

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Image, Information et Hypermédia

Présentée et soutenue par :

M. VINCENT ANGLADON

le vendredi 27 avril 2018

Titre :

Room layout estimation on mobile devices

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. VINCENT CHARVILLAT

M. SIMONE GASPARINI

Rapporteurs :

M. CARSTEN GRIWODZ, UNIVERSITE D'OSLO

M. DAVID FOFI, UNIVERSITE DE BOURGOGNE

Membre(s) du jury :

Mme LUCE MORIN, INSA DE RENNES, Président

M. PASCAL BERTOLINO, UNIVERSITE GRENOBLE ALPES, Membre

M. SIMONE GASPARINI, INP TOULOUSE, Membre

M. TOMISLAV PRIBANIC, UNIVERSITE DE ZAGREB, Membre

M. VINCENT CHARVILLAT, INP TOULOUSE, Membre

Acknowledgments

I would like to thank my thesis committee for giving me the opportunity to work on an exciting topic, and for the trust they placed in me. First, Simone Gasparini, I had the pleasure to have as advisor, who kept a cautious eye on my scientific and technical productions. I am certain this great attention to the details played an important role in the quality of my publications and the absence of rejection notice. Then, my thesis director, Vincent Charvillat, who was always generous in original ideas and positive energy. His advice helped me to put a more flattering light on my works and feel more optimistic. Finally, Telequid, which funded my works, with a special thought to Frédéric Bruel and Benjamin Ahsan for their great patience.

I would also like to thank my referees: Prof. Carsten Griwodz and Prof. David Fofi for their constructive feedbacks, as well as all the members of the jury for the great interest they manifested during the defense.

In the context of the Cogito project, I had the pleasure to collaborate with Tomislav Pribanić, Tomislav Petković, and Matea Đonlić, who gave me valuable feedbacks and with who I had pleasant exchanges.

The different atmospheres in which I floated played also a significant role. First, the pleasant ambiance in the IRIT laboratory at the ENSEEIHT school, in particular, the REVA team, where I enjoyed discussing with the permanents, the post-docs, and the Ph.D. students, with its delicious breaks and seminaries, the motivating atmosphere of the company, simultaneously busy and relaxed, and finally the precious mood of my friends and the members of my family, in particular, my parent I deeply thank for their great support.

Résumé

Titre : Création de plans d'intérieur avec une tablette

Mots-clés : scène intérieur, plan, reconstruction 3D, mobile, smartphone, tablette, capteur de profondeur, point de fuite, nuage de points, intéraction utilisateur

L'objectif de cette thèse CIFRE est d'étudier et de tirer parti des derniers appareils mobiles du marché pour générer des 3D des pièces observées. De nous jours, ces appareils intègrent un grand nombre de capteurs, tel que des capteurs inertiels, des caméras RGB, et depuis peu, des capteurs de profondeur. Sans compter la présence de l'écran tactile qui offre une interface pour interagir avec l'utilisateur.

Un cas d'usage typique de ces modèles 3D est la génération de plans d'intérieur, ou de fichiers CAO 3D (conception assistée par ordinateur) appliqués à l'industrie du bâtiment. Le modèle permet d'esquisser les travaux de rénovation d'un appartement, ou d'évaluer la fidélité d'un chantier en cours avec le modèle initial. Pour le secteur de l'immobilier, la génération automatique de plans et modèles 3D peut faciliter le calcul de la surface habitable et permet de proposer des visites virtuelles à d'éventuels acquéreurs. Concernant le grand public, ces modèles 3D peuvent être intégrés à des jeux en réalité mixte afin d'offrir une expérience encore plus immersive, ou pour des applications de réalité augmentée, telles que la décoration d'intérieur.

La thèse a trois contributions principales. Nous commençons par montrer comment le problème classique de détection des points de fuite dans une image, peut être revisité pour tirer parti de l'utilisation de données inertielles. Nous proposons un algorithme simple et efficace de détection de points de fuite reposant sur l'utilisation du vecteur gravité obtenu via ces données. Un nouveau jeu de données contenant des photos avec des données inertielles est présenté pour l'évaluation d'algorithmes d'estimation de points de fuite et encourager les travaux ultérieurs dans cette direction.

Dans une deuxième contribution, nous explorons les approches d'odométrie visuelle de l'état de l'art qui exploitent des capteurs de profondeur. Localiser l'appareil mobile en temps réel est fondamental pour envisager des applications reposant sur la réalité augmentée. Nous proposons une comparaison d'algorithmes existants développés en grande partie pour ordinateur de bureau, afin d'étudier si leur utilisation sur un appareil mobile est envisageable. Pour chaque approche considérée, nous évaluons la précision de la localisation et les performances en temps de calcul sur mobile.

Enfin, nous présentons une preuve de concept d'application permettant de générer le plan d'une pièce, en utilisant une tablette du projet Tango, équipée d'un capteur RGB-D. Notre algorithme effectue un traitement incrémental des données 3D acquises au cours de l'observation de la pièce considérée. Nous montrons comment notre approche utilise les indications de l'utilisateur pour corriger pendant la capture le modèle de la pièce.

Abstract

Title: Room layout estimation on mobile devices

Keywords: indoor, room layout, floor plan, 3D reconstruction, mobile, smartphone, tablet, depth sensor, vanishing point, point cloud, user interaction

Room layout generation is the problem of generating a drawing or a digital model of an existing room from a set of measurements such as laser data or images. The generation of floor plans can find application in the building industry to assess the quality and the correctness of an ongoing construction w.r.t. the initial model, or to quickly sketch the renovation of an apartment. Real estate industry can rely on automatic generation of floor plans to ease the process of checking the livable surface and to propose virtual visits to prospective customers. As for the general public, the room layout can be integrated into mixed reality games to provide a better immersiveness experience, or used in other related augmented reality applications such room redecoration.

The goal of this industrial thesis (CIFRE) is to investigate and take advantage of the state-of-the art mobile devices in order to automate the process of generating room layouts. Nowadays, modern mobile devices usually come a wide range of sensors, such as inertial motion unit (IMU), RGB cameras and, more recently, depth cameras. Moreover, tactile touchscreens offer a natural and simple way to interact with the user, thus favoring the development of interactive applications, in which the user can be part of the processing loop.

This work aims at exploiting the richness of such devices to address the room layout generation problem. The thesis has three major contributions. We first show how the classic problem of detecting vanishing points in an image can benefit from an a-priori given by the IMU sensor. We propose a simple and effective algorithm for detecting vanishing points relying on the gravity vector estimated by the IMU. A new public dataset containing images and the relevant IMU data is introduced to help assessing vanishing point algorithms and foster further studies in the field.

As a second contribution, we explored the state-of-the-art of real-time localization and map optimization algorithms for RGB-D sensors. Real-time localization is a fundamental task to enable augmented reality applications, and thus it is a critical component when designing interactive applications. We propose an evaluation of existing algorithms for the common desktop set-up in order to be employed on a mobile device. For each considered method, we assess the accuracy of the localization as well as the computational performances when ported on a mobile device.

Finally, we present a proof of concept of application able to generate the room layout relying on a Project Tango tablet equipped with an RGB-D sensor. In particular, we propose an algorithm that incrementally processes and fuses the 3D data provided by the sensor in order to obtain the layout of the room. We show how our algorithm can rely on the user interactions in order to correct the generated 3D model during the acquisition process.

Contents

Acknowledgments i

Résumé iii

Abstract v

1 Introduction 1
- 1.1 Ph.D. context . . . 2
- 1.2 Room layout generation . . . 2
- 1.3 Industrial state of the art . . . 3
  - 1.3.1 Acquisition devices . . . 3
  - 1.3.2 Modeling software and services . . . 6
- 1.4 Another solution . . . 7
- 1.5 Contributions . . . 8

2 Positioning 9
- 2.1 Assumptions . . . 10
- 2.2 Challenges . . . 11
  - 2.2.1 Visualization and User interaction . . . 12
  - 2.2.2 System Acquisition . . . 13
  - 2.2.3 System Localization . . . 14
  - 2.2.4 System Data interpretation and modeling . . . 14
  - 2.2.5 User privacy . . . 15
- 2.3 Conclusion . . . 15

I RGB 17

3 Vanishing Points for Image Understanding 19
- 3.1 Introduction . . . 20
- 3.2 Background . . . 21
  - 3.2.1 Vanishing point detection . . . 21
  - 3.2.2 Inertial data . . . 23
- 3.3 A fast and simple Vanishing Point (VP) detector using Inertial Motion Unit (IMU) data 24
- 3.4 Evaluation . . . 26
  - 3.4.1 The York Urban database . . . 26
  - 3.4.2 A new dataset . . . 27
- 3.5 Live VP estimation . . . 33
- 3.6 Conclusion . . . 34
- 3.A Restriction of uncertainty polygons to a set of orthogonal solutions . . . 36

II RGB-D 39

4 RGB-D Localization 41
- 4.1 Introduction . . . 42
- 4.2 Range imaging sensors . . . 42
  - 4.2.1 Definition of a range imaging sensor . . . 42
  - 4.2.2 Mobile range imaging technologies . . . 43
  - 4.2.3 Considered technologies . . . 47
- 4.3 Visual odometry evaluation . . . 47
  - 4.3.1 Taxonomy . . . 48
  - 4.3.2 Implementations on mobile devices . . . 50

4.3.3 Previous benchmarks . . . 51
4.3.4 Tested visual odometry algorithms . . . 52
4.3.5 Algorithms selection . . . 56
4.3.6 Mobile experiments . . . 60
4.3.7 Benchmark conclusions . . . 64
4.4 Planar approaches . . . 65
4.4.1 Overview and related works . . . 66
4.4.2 Our approach . . . 71
4.4.3 Planar approaches conclusion . . . 78
4.5 Conclusion . . . 80
4.A Additional planar Least Square (LS) Simultaneous Localization And Mapping (SLAM) results . . . 82

5 Room layout Estimation **85**
5.1 Introduction . . . 86
5.2 Background . . . 86
5.2.1 Room layout estimation . . . 86
5.2.2 Scene Semantic . . . 86
5.2.3 User interaction . . . 87
5.3 Analysis of user driven approaches . . . 88
5.3.1 Point and Line selection . . . 88
5.3.2 Plane selection . . . 89
5.4 The proposed layout generation pipeline . . . 91
5.4.1 Visibility Polygon . . . 92
5.4.2 Layout Generation . . . 93
5.4.3 Wall labeling . . . 96
5.4.4 Implementation details . . . 97
5.5 Evaluation . . . 97
5.6 Conclusion . . . 99
5.A Distance Measurement Evaluation . . . 103
5.A.1 First Experiment . . . 103
5.A.2 Second Experiment . . . 103

6 Conclusion **105**
6.1 Summary . . . 105
6.2 Limitations and perspectives . . . 106
6.3 Publications . . . 107
6.4 Collaborations . . . 107

Acronyms

API Application Programming Interface. 11, 45

AR Augmented Reality. 12, 89

BA Bundle Adjustment. 69, 70

BIM Building Information Modeling. 87

BOM Bill of Material. 44

CAD Computer Aided Design. 2, 3, 6

CIFRE Industrial Conventions for Formation and Research. 2

CNN Convolutional Neural Network. 22, 23, 34

DOF Degree Of Freedom. 12

EM Expectation Maximization. 21

fov Field of view. 107

FPS Frame per Second. 13

ICP Iterative Closest Point. 50, 53, 54, 66

IMU Inertial Motion Unit. vii, 2, 4, 5, 8, 13–15, 19–21, 23–26, 32–34, 64, 71, 76, 105, 107

Intel RSDK Intel® RealSense™ Smartphone Developer Kit. 44

IR Infrared. 44–47

Kinect^{SL} Kinect for Xbox 360. 6, 42, 45–47, 60, 67

Lidar Light Detection And Ranging. 6, 7

LS Least Square. viii, 41, 69, 70, 73, 76, 79, 80, 82, 83

PCA Principal component analysis. 74

PP Planar Patch. 68, 69, 71, 73–80, 82, 90–97

RANSAC RANdom SAmple Consensus. 21, 69, 73

RMS Root Mean Square. 26

SDK Software Development Kit. 13, 47, 56, 65, 71, 72, 75, 77, 81

SfM Structure from Motion. 35, 70

SL Structured Light. 44–47

SLAM Simultaneous Localization And Mapping. viii, 41, 48, 51, 57, 67, 69, 70, 76, 79, 80, 82, 83, 91, 105, 106

Tango TDK Project Tango Tablet Development Kit. 11, 44, 45, 47, 48, 65, 66, 71–73, 78, 80, 92, 104–107

ToF Time of Flight. 3, 5, 6, 44, 46, 47

TSDF Truncated Signed Distance Function. 49, 51

VIO Visual Inertial Odometry. 47, 65, 71, 73, 75, 76, 97

VO Visual Odometry. 20, 42, 46–49, 51, 65, 78, 80, 91, 105–107

VP Vanishing Point. vii, 8, 15, 19–34, 36, 105, 107

vSLAM Visual Simultaneous Localization And Mapping. 35, 49, 50, 56, 64, 65, 76, 77, 80, 106

List of Symbols

s Line segment. 95

π Planar Patch. 66, 74, 75, 95

$\mathbf{\Pi}$ Infinite plane. 69

Θ SLAM state. 70, 76–78

CHAPTER 1

Introduction

Contents

1.1 Ph.D. context **2**
1.2 Room layout generation **2**
1.3 Industrial state of the art **3**
1.3.1 Acquisition devices 3
1.3.2 Modeling software and services 6
1.4 Another solution **7**
1.5 Contributions **8**

1.1 Ph.D. context

THIS work is the result of an industrial collaboration between the R&D company Telequid and the VORTEX team at IRIT laboratory, in the context of a CIFRE convention. Telequid is an R&D company focused on media interaction: social TV, cloud video edition and "second screen" application through mobile and web applications using innovative technologies for sound (1D), image (2D) and video processing. The core applications provided by the company range from media retrieval, video edition up to augmented reality applications. The objective of this thesis responds to the company's goal of exploring the 3rd dimension, in order to further enrich the visual interaction and communication capabilities of its services and integrate them in its current software and hardware platform. Telequid wanted to leverage the new capabilities of the mobile devices in terms of sensing, specifically 3D sensors, and eventually to propose a new service. After an assessment period of some 3D sensors, a floor plan generation application was chosen as a target application.

1.2 Room layout generation

Room layout generation is the problem of generating a drawing or a digital model to scale of an existing room. Our goal is to design a mobile application relying on mobile sensors (IMU, RGB camera, depth camera) and, possibly, on user actions, to produce a model of the room layout. This model would be consistent with the geometry of the observed room and at scale, *i.e.* the room dimensions could be extracted from the model. Depending on the needs of the user, the model could be imported into another application for further use.

A typical usage of room layouts is the generation of floor plans or 3D Computer Aided Design (CAD) models for the building industry: the model can be used to sketch the renovation of an apartment or to assess the quality and the correctness of an ongoing construction w.r.t. the initial model. In the context of the real estate industry, the automatic generation of floor plans can ease the process of checking the livable surface and to propose virtual visits to prospective customers. Some notable rental platform and estate agencies such as AirBnB, Habiteo, and Orpi already propose 3D visits on their website, as illustrated in Figure 1.1. Room layouts can also be used for indoor navigation to

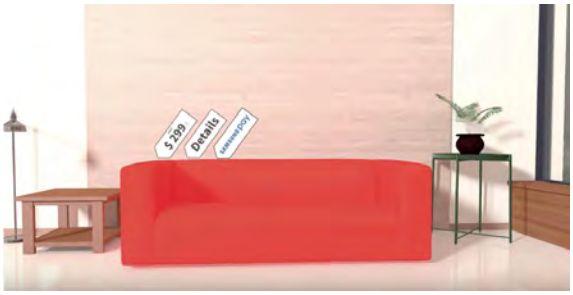


(a) Habiteo 2.5D tour. Image courtesy of Habiteo. **(b)** 2.5 indoor maps generated by Pintore *et al.* [147]

Figure 1.1: Room layout usages for the real estate.

help people localize themselves in large areas such as shopping malls or airports, or even for robot navigation in human-made environments. A new demand for layouts and 3D models is also coming from companies performing energy audits to analyze and assess energy efficiency of buildings: the 3D model of the room(s) integrating building envelope (*i.e.* walls, doors, floors, *etc.*) can be fed in a tuned thermal model to provide an estimate of the thermal efficiency of the building without costly and time-consuming measurements. For the general public, the room layout can be integrated into mixed reality games to provide a better immersiveness experience, or used in other related augmented reality applications such room redecoration as shown in Figure 1.2.

The different consumers of room layouts (geometers, architects, real estate agents, individuals, *etc.*) have different requirements. While geometer's floor plans require high accuracy, architects may



(a) Samsung VuildUs virtual reality application enables to visualize virtual furniture in a reconstructed room and to check their sizes fit the available space. Image courtesy of VuildUs.



(b) HomeAR augmented reality furniture visualization. Image courtesy of HomeAR.

Figure 1.2: Room layout usages for individuals

need less accurate models that can be easily (possibly, automatically) annotated to identify and localize water intakes, electrical plugs, the presence of a suspended ceiling, *etc*. In the next section, we will see there are existing room layout estimation solutions which can satisfy some of these specific requirements.

1.3 Industrial state of the art

We classified the existing solutions into two categories:

- *acquisition devices*, to take individual measurements or capture 3D point clouds;
- *modeling software and services*, to generate CAD models or floor plans from measurements or 3D point clouds.

When a product combines the two solutions, we call it an *all-in-one* solution. The creation of the room layout can be performed during the acquisition, we call this workflow *online*, or after the acquisition (*offline* workflow). In the case of an online process, missing data or measurements may lead to incomplete plans, thus requiring costly do-overs on site.

1.3.1 Acquisition devices

We separated the manual measurement and the point cloud acquisitions devices. These devices can also be classified in term of accuracy, cost, public (individuals or professionals), efficiency, *etc*., as we did in Table 1.1, which also highlights *all-in-one* products and technologies which appeared during this Ph.D.

1.3.1.1 Manual measurements

Laser rangefinders are individual measurement devices relying on a (usually pulsed) laser Time of Flight (ToF) technology. They offer high accuracy ($3\,\mathrm{mm}$ of absolute error) for prices starting at €70. The operator performs the measurements by holding the device at one of the extremities of the object to measure and aims at the other extremity with the laser. Each measurement has to be reported on a freehand sketch or a mobile application. Figure 1.3 shows the three steps of this process. This task can be challenging when dealing with furnished or cluttered environments, preventing the direct measurement of certain distances. The accuracy of the measurements is mainly affected by the difficulty to hold the device perfectly level: in our experiments (see Section 5.A), we demonstrate that the relative error of the measurements is around $1\,\mathrm{mm\,m^{-1}}$.

Angles and distances measurements Another method to estimate a room layout is to consider a fixed point, seen as the origin of the scene, and to measure the angles between two successive room corners (w.r.t. the origin), and the distances from the origin to each corner. The measurement device performs rotational movements only. The main advantage of this approach is the automation of the

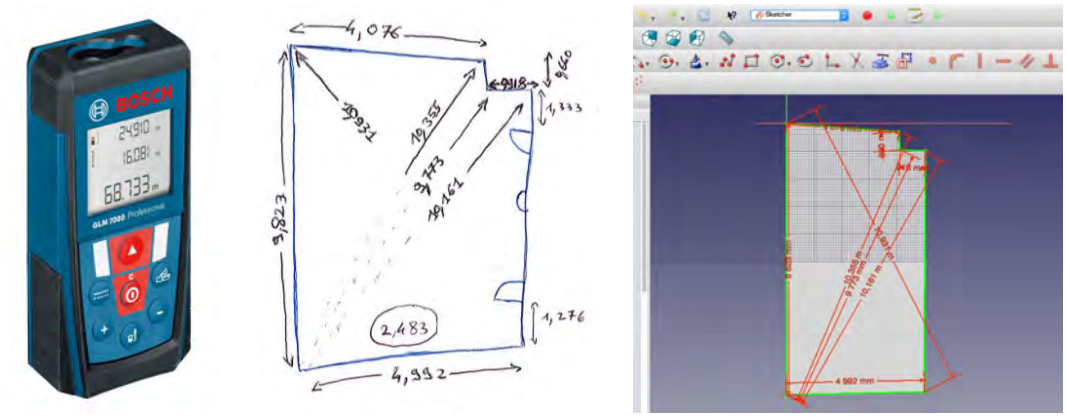


Figure 1.3: Left: Bosh laser rangefinder. Middle and right: freehand sketch of a room and drawing with constraints in FreeCad.

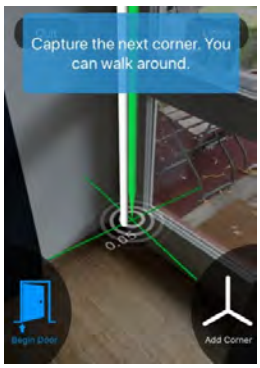
room layout drawing from the measurements, *i.e.* an *online workflow*: the user can immediately see whether there is a problem with a measurement or not. For example, Measurix illustrated in Figure 1.4a is a solution combining a Pocket PC, a tripod, a laser rangefinder, and a system to measure its orientation, after each measurement, a segment is displayed on the Pocket PC. The main disadvantage of these approaches is they can only handle rooms with a geometry corresponding to a star domain [192][1]. Arkisketch (see Figure 1.4b) proposes another solution with a laser rangefinder, which can be plugged into a smartphone using the phone IMU to estimate the laser orientation. The Magic Plan application shown in Figure 1.4c is an alternative not requiring a rangefinder: the distances are evaluated by trigonometry from the device orientation, the distance of the device to the floor (assumed constant and calibrated) and optionally the ceiling height. Implementation details of a similar system can be found in [146]. In term of accuracy, the use of a tripod guarantees to Measurix small measurement errors, Arkisketch is affected by the imperfect rotation of the user smartphone, and Magic Plan can suffer from high errors (superior to $10\,\mathrm{cm}$) with large rooms. Very recently, augmented reality frameworks such as ARKit and ARCore allow to track horizontal planes (usually the ground), localize the mobile device, which is not constrained to rotational movements. They enable a new way to perform manual measurements: the user can perform selections on a plane, such as selecting horizontal lines or points on the plane and get the relevant measurement. The latter solution is being considered by Occipital TapMeasure and Magic Plan (latest version for devices compatibles with ARKit). At the time of writing the AR frameworks do not provide a simple access to the 3D data.



(a) Measurix solution. Image courtesy of Measurix.



(b) Archisketch rangefinder. Image courtesy of Archisketch.



(c) MagicPlan application. Image courtesy of Sensopia.

Figure 1.4: Three hardware and software solutions to perform manual room layout estimation from angles and distances measurements.

[1] A set of points S is called a *star domain* if there exists a point $p \in S$ such that all points of S are connected to p by a line segment included in S. In our case, it means there must be at least one location from which all the wall corners are visible.

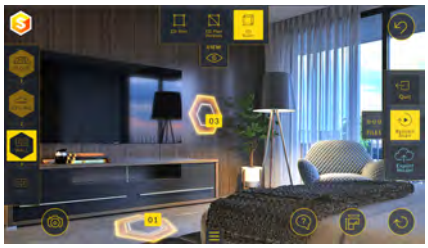
1.3.1.2 Point clouds

Another category of devices can aggregate multiple distance measurements at a given moment and build point clouds. The acquisition process, instead of measuring the walls, scans the full room with as much as details as possible, and the measurements are extracted from the point cloud later.

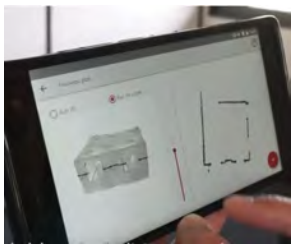
Some devices are static: they are placed on a tripod the operator has to displace between each acquisition. Other devices are mobile, allowing faster acquisitions and a larger diversity of viewpoints. In both cases, the various acquisitions have to be aggregated w.r.t. the device position and orientation. For static devices, there are only a few pose estimations to perform, generally from the point clouds or manually with targets. On mobile devices, instead, the pose estimation must be performed at least at the acquisition rate of the depth and image sensors. We will see later that it implies a lower accuracy of the camera pose.

Smartphones and tablets started to integrate a depth sensor in 2014, mainly with Project Tango devices and the Structure Sensor by Occipital. Earlier devices, such as the Phi.3D, use desktop depth sensors. We detail depth sensors technologies for mobile devices in Section 4.2 on page 42. Their range is limited ($4.5\,\mathrm{m}$ maximum) and the measurement error increases with the distance (around $1\,\mathrm{cm}$ error at a $3\,\mathrm{m}$ distance). Applications to create floor plans on these devices emerged then, such as:

- Easybuild with an online workflow where the user selects each wall in a precise order, Figure 1.5a displays the selected walls;
- MyCaptr, with an offline workflow, where at the end of the scan, the user can select an horizontal slice of the point cloud which is processed to extract the walls, as shown in Figure 1.5b;
- Canvas.io (see Figure 1.5c), with an offline workflow where all the 3D data is sent to their paying modeling service.



(a) Easybuild for Google Tango devices. Image courtesy of Wosomtech.



(b) MyCaptr for Google Tango devices. Image courtesy of LevelS3D.



(c) Canvas.io for iOS devices with the Structure Sensor. Image courtesy of Occipital.

Figure 1.5: Three applications for mobile devices equipped with depth sensors to estimate room layouts.

Portable solutions combine a 2D ToF laser range scanner, an IMU and a wide angle RGB camera such as the Paracosm PX-80 and the Kaarta Contour. Some solutions require a backpack, to carry the acquisition unit (Geoslam Zeb-Revo) or even the sensors (Leica Pegasus). Although these sensors are very accurate, localization errors lead to final measurements errors around a few cm. They are designed for professionals and cost more than $\in 10000$.

Trolleys solutions allow to carry heavier hardware and to generate more regular point clouds since the sensors are not affected by the movements of the operators during the walk. It is more suitable for large indoor spaces with a flat floor. Existing commercial, illustrated in Figure 1.7 include Viametris iMS 3D, Navvis M3 Trolley, both use sensors similar to the Portable solutions mentioned earlier, and Applanix-Trimple Timms, which uses a 3D laser sensor typical of tripods solutions.



(a) Countour. Image courtesy of Kaarta.



(b) PX-80. Image courtesy of Paracosm.



(c) Zebrevo. Image courtesy of Geoslam.



(d) Pegaasus backpack. Image courtesy of Leica.

Figure 1.6: Four portables solutions to capture point clouds in indoor scenes. Although not visible, the four solutions include a screen to monitor the point clouds acquisition and registration.



(a) iMS 3D trolley. Image courtesy of Viametris.



(b) The M3 trolley. Image courtesy of Navvis.



(c) The Timms trolley application. Image courtesy of Applanix.

Figure 1.7: Three trolleys solutions to capture point clouds in indoor scenes.

Scanners on tripods use generally a phase-shift laser ToF technology which offers long range (70–1000m) and high distance accuracy (1 mm of error). They are often referred as Light Detection And Ranging (Lidar) and their use are more suitable for large scenes, especially when high accuracy is needed, and there are a lot of details to capture. Figure 1.8a depicts a Leica P30 Lidar. They are very expensive ($>$ €20000), generally heavy and fragile, and their use is restricted to geospatial professionals.

Matterport and iGuide shown in Figure 1.8b and Figure 1.8c propose all-in-one solutions to generate floor plans and virtual visits, with more affordable hardware (around €3400 for Matterport). Matterport 3D camera uses three rotating 3D sensors similar to the Kinect for Xbox 360 (Kinect^{SL}), which have a limited range (4.5 m maximum) and a lower accuracy, as we have seen for Smartphones and tablets solutions. iGuide (formerly Planitar) 3D sensor is a fixed 2D laser scanner. Both products send all the photos and 3D data to their servers where the floor plans are generated, probably with the help of humans operator, and delivered with one or two business days. Matterport requires manual operation from the clients to handle windows, mirrors and to clean the point cloud before generating a floor plan.

1.3.2 Modeling software and services

The modeling step produces the CAD models and/or the floor plans. Their input can either be measurements or a point cloud. In the first case, an operator can use a drawing software such as AutoCad LT (Figure 1.9a) to report measurements, apply constraints (orthogonality, parallelism, *etc.*) in order to draw a model. In the second case, a point cloud processing and drawing software is used, such as PointCab (Figure 1.9b) and Revit (Figure 1.9c). They offer functions to create automatically or semi-automatically CAD models from point clouds. These software applications are generally very expensive (generally more than €2000), and require high-end computers to load and handle several hundreds of millions of 3D points. The creation of an apartment CAD model can take several hours



(a) P30 Lidar. Image courtesy of Leica.



(b) Matterport scanning solution. Image courtesy of Matterport.



(c) The iGuide acquisition system. Image courtesy of iGuide.

Figure 1.8: Three tripod scanning solutions.

Cat.	Product	Device motion	Accuracy	Workflow	Cost	Public	Capture/Layout time
Manual measurements	Laser rangefinder	Free motion	High	Offline	Low	General public	Fast / Slow
	Magic Plan	Rotation	Low	Online	Low	General public	Fast / Fast
	Measurix	Rotation	High	Online	*N.A.*	Trained staff	Fast / Fast
	Magic Plan ARKit, Tap Measure	Free motion	Medium	Online	Low	General public	Fast / Fast
	Archisketch	Rotation	Medium	Online	Medium	General public	Fast / Fast
Point cloud based	Easybuild	Free motion	Medium	Online	Medium	General public	Fast / Fast
	Canvas.io, GeoCV, MyCaptr	Free motion	Medium	Offline	Medium	General public	Medium / Slow
	Phi.3D	Free motion	Medium	Offline	High	Professional	Medium / Slow
	Portable solutions	Free motion	Medium	Offline	Very high	Professional	Medium / Slow
	Trolley solutions	Wheeled	Medium	Offline	Very high	Professional	Medium / Slow
	Matterport, iGuide	On tripod	High	Offline	High	Trained staff	Slow / Slow
	Lidars	On tripod	Very high	Offline	Very high	Professional	Slow / Slow

Table 1.1: Comparison of acquisition devices and services to estimate room layouts. Product names in bold refer to *all-in-one* solutions. We highlight in gray the products that appeared after the beginning of this Ph.D. thesis in 2014. The cost column refers to the price of hardware and software acquisition, and it assumes the users owns a mobile device with ARKit support.

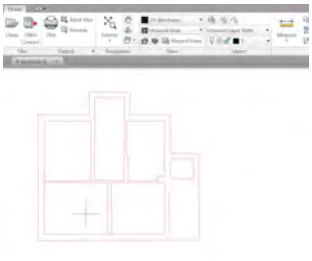
even for an experienced user. Some companies, such as Snapkin propose services to create a model from a point cloud using both automatic and manual operations. The delivery of the result can take several days. *All-in-one* solutions such as GeoCV, MyCaptr, Canvas.io, Matterport, iGuide internalized such a service.

1.4 Another solution

In 2014, at the beginning of the Ph.D., the solutions compared in Table 1.1 showed that Magic Plan was the only solution for the general public to offer an *online* workflow (*i.e.* where the room layout can be generated during the acquisition). As explained earlier, online solutions are interesting as they



(a) Floor plan drawing with AutoCAD LT. Image courtesy of Autodesk.



(b) Automatic 2D vectorization from a sliced point cloud. Image courtesy of PointCab.



(c) Plane fitting on a point cloud with Revit. Image courtesy of Autodesk.

Figure 1.9: Drawing and point cloud processing software.

allow to have direct feedback so that errors can be immediately noticed and possibly corrected directly on-site. The main limitation of Magic Plan was its low accuracy, which comes from the assumption that the device height is constant, the imperfect rotation of the user and the difficulty to select a corner hidden by furniture. We wanted to propose a new online solution designed for the general public with higher accuracy. The choice to design a mobile application was motivated by the wide diffusion of mobile devices and their increasing computing capabilities, which has enabled real-time image processing and the blooming of mobile computer vision applications. Moreover, the recent trend of equipping mobile devices with 3D sensors is making such devices a powerful tool for developing applications that are aware of the 3D structure of the scene, beside the already existing capability of sensing the movements and the orientation, granted by the different inertial sensors. Finally, one of the most significant innovations introduced by modern mobile devices is the unique way of interacting with the user through the touchscreen: the user can easily intervene and interact with the application with the gestures on the screen, possibly avoiding or correcting errors, in a *user-in-the-loop* paradigm.

1.5 Contributions

The goal of this thesis is to take advantage of modern mobile devices, and in particular devices with depth sensors. A comparison and a study of their limitation are carried out in Section 4.2. We also investigate and propose solutions to the scientific challenges underlying the room layout generation problem in the context of mobile devices. In particular, the interest in the user interaction field is linked to the presence of a touchscreen. The presence of sensors such as RGB cameras, IMU, depth sensors led us to consider various fields such as computer vision, image processing, point cloud processing, localization and mapping issues.

In Chapter 2 we will detail the various issues related to the problem of room layout estimation on mobile devices. This chapter articulates the rest of the thesis since each of the following chapters will focus on particular challenges identified there. For the purpose of image understanding, we study in Chapter 3 the problem of VP estimation in images. In Chapter 4 we will study the challenges of real-time pose estimation and the problem of map optimization in the context of plane features. Assuming the device can then reliably localize and map the environment, we address the problem of layout estimation from 3D data and user inputs in Chapter 5.

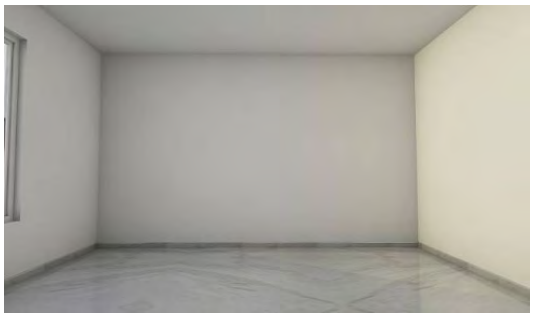
CHAPTER 2
Positioning

Contents

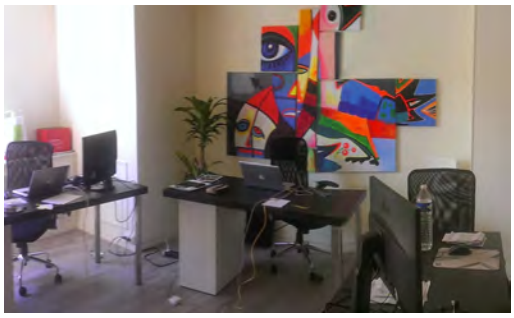
2.1 Assumptions **10**
2.2 Challenges **11**
2.2.1 Visualization and User interaction 12
2.2.2 System Acquisition 13
2.2.3 System Localization 14
2.2.4 System Data interpretation and modeling 14
2.2.5 User privacy 15
2.3 Conclusion **15**

THERE are multiple paths from a problem to a solution. The path of this thesis is the reflect of the investment performed among some scientific and technical challenges initially identified and continuously updated. Thus, to explain our works, we are going to detail these different challenges, and which chapters they are related to.

2.1 Assumptions



(a) The ideal and easiest case: an empty room of reasonable size with a rectangular shape. Image courtesy of Getty Image.



(b) The case we chose to handle: reasonable sized room with clutter and weak Manhattan assumption.

Figure 2.1: The ideal room and a typical office room.

Scene hypotheses All rooms are not like the one in Figure 2.1a: rectangular and empty. It exists a large variety of indoor scenes designed by architects, the most original projects may have non-vertical walls or curved walls and ceiling, as illustrated in Figure 2.2. Open plan designs (where there are subtle or no separation between the functional parts of the living area), erase the concept of rooms and make the understanding of the scenes more complex. Issues can also originate from the room decorations, as show in Figure 2.3: furniture and decorations may hide completely a wall, mirrors and bay window may not be perceived by imaging sensors, wall cladding alter the flatness of the walls and thus their detection.



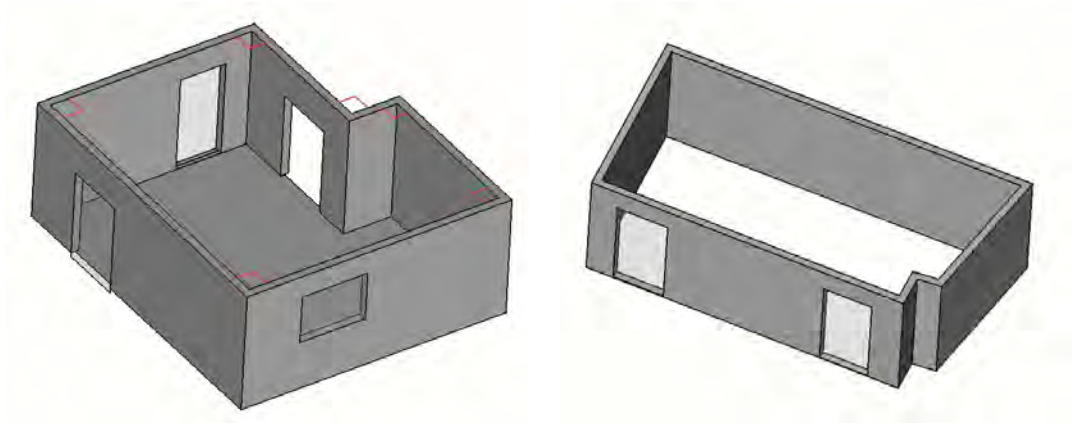
Figure 2.2: There is a large variety of indoor rooms designed by architects. From left to right: slanted walls in the Asymmetric Valley House, curved walls and ceiling in Perivolas hotel, mezzanine floor or the difficulty to separate rooms. Images courtesy of PlanBureau, Perivolas, and Mint Tiny Homes.

Faced with the diversity of the rooms, it is necessary to restrict the indoor spaces we will handle. We discarded indoor scenes with non vertical wall because they are very uncommon. The *Strong/Weak Manhattan World* hypotheses correspond to two very common hypotheses on the geometry of the room. The *Weak Manhattan* assumption (Figure 2.4b, sometimes called Soft Manhattan) states that the scene is made of a horizontal planar floor, with planar walls orthogonal to the floor, and the ceiling parallel to it. The *Strong Manhattan* assumption (Figure 2.4a) imposes the walls are also orthogonal or parallel each other. Since indoor space from European cities rarely comply with the Strong Manhattan hypothesis, we opted for the Weak one. Given the accuracy and the limited range of conventional depth sensors, we restricted ourselves to reasonable size rooms (under $200\,\mathrm{m}^2$) and a ceiling height limiter to $5\,\mathrm{m}$. Objects occluding the walls (decorations, furniture, *etc.*) are generally referred as *clutter*. We decided to handle cluttered scenes, which are very common, knowing some specific cases,



Figure 2.3: Decorations can make the rooms very challenging to capture. From left to right: rock face wall cladding produces non-flat surfaces, a massive furniture which prevents from capturing the wall behind, bay windows, which cannot be perceived by imaging sensors. Images courtesy of Gosford Quarries, Produce.com.sg, and budget-maison.com

such as full occlusion of the walls could not be handled. Figure 2.1b illustrates our room hypotheses: a reasonable sized room, with vertical planar walls, following the Weak Manhattan assumption, and with reasonable clutter.



(a) Strong Manhattan: the walls are orthogonal to the floor and between themselves.

(b) Weak Manhattan: the walls are orthogonal to the floor but can have any orientation.

Figure 2.4: Strong and weak Manhattan assumptions.

Hardware choices The choice of the device can also have an impact on the results: different computational power, memory resources, quality of the sensors, developer Application Programming Interface (API), *etc*. We opted first for the Apple iPad Air tablet which was at the time a high-end device with interesting performances. Its choice was also motivated by the reduced fragmentation of Apple devices, the presence (at the time) of a more advanced camera API and its compatibility with the Structure Sensor accessory. We later considered the Project Tango Tablet Development Kit (Tango TDK), which is a more powerful device with a built-in depth sensor.

2.2 Challenges

In this section, we will introduce and overview the different challenges related to our problem. Zlatanova *et al*. [227] proposed a list of open problems that need to be taken into account when dealing with indoor mapping. Starting from the challenges identified by Zlatanova *et al*., we adapt and enrich them to the specificities of our topic. In particular, we recall that we are first considering the hypothesis of *Strong Manhattan* scenes observed with a high-end mobile device, without depth sensor, and then the hypothesis of *Weak Manhattan* scenes, observed from a high-end mobile device with a depth camera. Moreover, our solution is considering two levels, the man and the machine, working jointly. The machine refers to the chosen hardware and the developed software, which sense and analyze the

scene, possibly making errors. The human controls the process and should be able to intervene and interact to provide guidances or corrections.

2.2.1 Visualization and User interaction

The touchscreen on mobile devices is an interface between the user and the systems. It is both an input and an output of the system. Abowd and Beale framework [44] is a model of the computer-machine interactions presented in Figure 2.5, which also provides an interpretation of the possible issues in using an interface. Following this model, the user formulates his intentions to the user interface input, which is interpreted by the system. The latter performs computations and changes its state, which is represented on the user interface output. The user observes this representation and formulates new intentions. As we can see, the user acts in a loop with the system through the user interface. It means the user interface (input and output) and the system should be designed jointly. Thereafter, we will call *visualization* the user interface output, and *user interaction* its input.

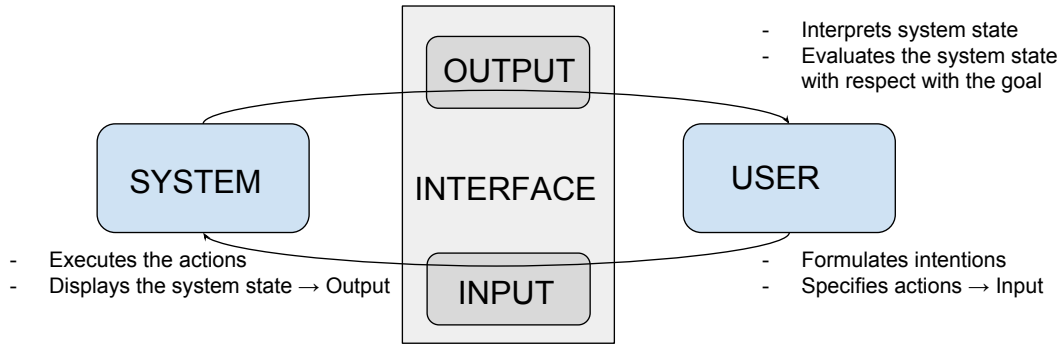


Figure 2.5: Abowd and Beal framework.

Augmented Reality The user interface can have different communication channels, called *modalities*, between the system and the user. In our case, the two input modalities are the touchscreen and the 6 Degree Of Freedom (DOF) device tracking allowing Augmented Reality (AR) interactions. With AR applications, the user can interact on the image of the camera displayed on the screen, while moving at the same time the device, which changes the image displayed on the screen. For example, positioning a virtual object (*e.g.* a wall corner) in an augmented view can be done with the touch screen (pan and rotation gestures), by moving/rotating the mobile device until the object is located and aligned where it should be, or by combining the two modalities. Another challenge with AR applications is to design interactions which offer high level of accuracy, taking into account the viewpoint can be shaky due to issues with the tracking or uncontrolled user movements.

Usability goals One important challenge consists in designing a user interface which is *effective* (it makes possible to achieve the task), *efficient* (it enables to perform the task fast), and *easy to use* (can be understood with none or little training), which are some of the usability goals defined by Rogers *et al.* [151]. These criteria can be evaluated with *user studies*, which monitor for example the success rate, the number of interactions, the interaction duration and user feedbacks on several groups of users.

Cooperation and interactions design Something is missing in Figure 2.5: the system receives non-user inputs. In our case, it can capture the physical environment with its camera (and eventually a depth sensor), which may be sufficient to find the solutions in some scenarios. When the user and the system work jointly to produce a solution several other issues arise: how the interactions affect the automatic process? How is used the input from the user to correct the model? At which extent the user can override what the machine is producing? Which task can be influenced by the user and which others are considered fully automatic? On the one hand, we can find *automatic* approaches such as PolyFit [126] and Murali Scan2BIM [125], where the room layout can be estimated without the assistance of the user. The user may intervene in the form of corrective actions at the end of the

system processing, as in [123, 148]. On the other hand, there are *user-driven* approaches such as Magic Plan Figure 1.4c and Tap Measure, which let the user selects all the corners. The system may intervene in solutions like Easybuild and O-Snap [9] where the user manually selects each surface to keep. The system reduces the user effort by detecting the corresponding planar surfaces and connects them. To find a good balance between user and effort repartition, it should be known in the one hand, what the system can achieve (*e.g.* localization, partially detect room corners, walls, clutter, find some connections between walls, *etc.*), and in the other hand, what user interactions to consider. These user interactions should be designed in term of behaviors, which define the place of the user with respect to the system: initialize, validate, correct, ask, answer, finalize, *etc.*, including manual modes (*e.g.* model/create, select, edit, *etc.*), provided by the different modalities which enable interaction medium such as gestures, widgets (button, scrollbar, menu, ...), device motion, *etc.*

2.2.2 System Acquisition

Acquisition is the step where information (mostly spatial, such as distances, angles, structure, *etc.*) of the considered scene is captured. These data are the input of two important problems we detail later on: localization and modeling.

As detailed by Khoshelham and Zlatanova [95], a wide range of sensors can be considered: IMU, RGB camera, depth sensors, radio-based sensors, *etc.* The main challenges associated with the use of sensors are:

- the choice of the sensor (which sensors to consider and which technologies);
- the calibration process, which should be transparent to the user or simple;
- the handling of the sensor data noise;
- data fusion between different sensors with different characteristics.

In the following, we further discuss these points for different sensors present in mobile devices.

2.2.2.1 Mobile device sensors

IMUs combine inertial (accelerometer and gyroscopes) and magnetic sensors. The fusion and integration of these sensor data can be used to estimate:

- the roll and pitch angles of the device, accurately in the long term;
- the heading angle of the device with a lower accuracy or only in the short term (*i.e.* during a short period of time);
- the position of the device on the short term only as demonstrated in [133].

The choice of a high-end mobile device generally guarantees the presence of gyroscopes and higher-quality sensors. Their calibration is performed by the manufacturer (*e.g.* temperature sensitivity, scale factor error, sensor axis misalignment, *etc.*) to provide "usable" raw inertial data, but not noise-free. While mobile Software Development Kits (SDKs) provide functions to retrieve the device orientation, it can be convenient to use its own implementation, in order to add support to the device position estimation and to handle manufacturer flaw such as the iPhone 5S orientation bias [1]. It is a difficult task because of the complexity of the filter and the addition of a custom calibration process for the end-user.

RGB cameras offer high resolution images suitable for accurate applications such as motion estimation. Again, the choice of the image sensor comes from the choice of the mobile device: higher-end smartphones offer additional options: video image stabilization, higher Frame per Second (FPS), *etc.* Mobile SDKs rarely provide the intrinsic parameters of the cameras, which imply the design of a calibration process.

[1] https://gizmodo.com/heres-why-the-iphone-5s-accelerometer-is-so-screwed-up-1445966306

Depth cameras are either built-in in some mobile devices or pluggable. We will detail in Section 4.2 the various technologies and their associated limitations. In general, the main issues for consumer depth cameras in the context of mapping is the quality of the provided data: the measurements noise increase significantly with the distance and some materials like glasses or reflective surface may affect the accuracy of the measurements.

Radio sensors and Microphones can be used for the localization, but they require referenced emitters and only offer a low accuracy both in the short and long term. For this reason, they are not considered in this thesis.

2.2.2.2 Sensor fusion

When multiple and different sensors are available, it is interesting to combine them as they can compensate their single limitations, thus providing more reliable measurements. We will discuss more in detail how in Section 3.2.2 to employ multiple sensors to improve the estimation of vanishing points. When employing multiple sensors, calibration is needed in order to address two main problems:

- time synchronization, since the sensors may not use the same clock, or time information may not be present;
- sensor-to-sensor frame transformation and especially orientation alignment.

Again the calibration should be transparent for the end-user or easy to perform.

The main challenge is then the data fusion between the sensors, *i.e.* integrating the various sensor data to produce more consistent and accurate measurements than the one provided by each individual sensor. Usually, this requires the design of a filter to combine the sensor data with their different noise model and acquisition rates.

2.2.3 System Localization

During the acquisition process with a hand-held device, the user may need to move around to capture the entire space, introducing the challenge of the device localization. The use of GLONASS-GPS signals is not possible in indoor environments because building materials significantly reduce their signal strength, and their positioning accuracy is very low (around $2\,\mathrm{m}$). We previously saw systems relying on a radio system or an IMU only could not be considered either. Cameras (RGB and range imaging) can, however, be used for precise indoor localization, with multiple setups (one or several cameras, with the eventual inclusion of an IMU) and assuming good image quality and the presence of texture. When the device visits an unknown space, the localization relies on a *dead reckoning* process: the current device position is estimated from the previously estimated location. [1] Each error leads to an offset in the subsequent positions, which may increase over time with the accumulation of the errors. It is a component which has to be reliable: it should provide accurate result in constant (or bounded) time, at a high frame rate, and obviously be completely crash-free and leak-free despite the complexity of the processing. It should be also versatile, to handle well different environments (low texture, flat scenes, low luminosity, *etc.*). In Chapter 4 we detail the components of an RGB-D localization algorithm, we explain how to evaluate it, and which factors can impact their accuracy.

2.2.4 System Data interpretation and modeling

After the data is acquired by the various sensors, it has to be processed to generate a model, a simplified geometric representation of the captured data (here a room layout) with semantic information (*e.g.* doors, walls, ceiling, *etc.*).

[1] We see here the interaction scheme can have an impact on the localization accuracy. If the user is forced to revisit a place (*e.g.* force to perform a trajectory which loops), the accuracy is increased. In contrast, if the user is asked to perform a long, complex and shaky interaction, *e.g.* to perform a painter movement with the device to select the walls, more positional errors may be accumulate.

Data interpretation can be seen as a segmentation process where some parts of the data are aggregated and assigned a label. The segmentation can originate from the extraction of features: for example, planes can be detected and classified as wall, floor or ceiling. But not necessarily: individual points or surface elements can be segmented individually as well, and then aggregated, as in [74]. The challenge is to obtain the highest sensitivity and precision, with a low computational cost, coping with data noise for the feature extraction process, and handling the large diversity of the indoor scenes seen in Section 2.1 on page 10 for the segmentation process.

Data modeling takes the interpreted data and computes (in the 2D case) a polygon representing the 2D shape of the scanned room. It should minimize both a model fitting and a model complexity criterion. On one hand, a high weight on the complexity term enables to obtain a simplified solution when there is for example too many outlier data, on the other hand, a complex scene may rather need a relaxed complexity. The modeling can also take advantage of extracted features. Some monocular approaches rely on line segments analyzed with the help of VP (detailed in Chapter 3), while methods taking advantage of 3D data can take into consideration planar surfaces, detailed in Chapter 5).

Data interpretation and modeling are tied since generating a 2D room layout is giving a wall label to the features generating the layout. The two processes share common challenges, such as handling an acquisition under progress, handling missing data, and eventual speed constraints.

2.2.5 User privacy

It is quite natural to consider his own home as an intimate area and to be cautious when sharing a picture of it. Things tend to be different with a 3D scan, though, as it combines photos, a 3D model, and, potentially, the recognition of all the owner's objects. Knowing the exact location of the building, the size of the house and all the rooms, the brands of the furniture, *etc*., may be used to infer or reveal the apparent wealth, tastes, and other private and sensitive information, which can be exploited for commercial purposes. For the time being, there are concerns about possible future sale of users' floor plan collected by Roomba robotic vacuum iRobot are already a concern[2]. On the technical side, preserving the user privacy requires identifying the sensitive data, and to protect it: the data can be simply discarded once the processing has been completed, encrypted if needed to be stored. Some data, even harmless in appearance such as local features extracted from images, used to perform visual place recognition, can affect the user privacy, as demonstrated by Angelo *et al*. [41] who partly recovers images from a set of features. On the user side, it is very important is to establish a trust feeling with clear privacy policies, supported by adequate technical choices: for example, an application that does not require an Internet connection can be perceived as "safer" as it may assure a more control of the data (*i.e*. no cloud storage). Similarly, the release of the source code can help to verify whether the software complies with the data privacy policy.

2.3 Conclusion

In this chapter, we have presented our work hypotheses and identified the different challenges of the room layout estimation problem. To face the diversity of the indoor scenes, we categorized and restricted the set of the considered scenes. In Chapter 3 we consider *Strong Manhattan* scenes observed with high-end mobile device, without depth sensor and we propose a new method to estimate VP combining the data from both the camera and the IMU. VPs are an important geometric cue for orientation estimation and for image understanding tasks. In the next chapter we instead relax the Manhattan constraint, and we consider the case of *Weak Manhattan* scenes using a mobile device equipped with depth sensors. In particular, in Chapter 4 we will tackle the localization problem with the help of depth sensors and in Chapter 5 we will study different solutions for the modeling problem while respecting the user privacy.

[2] https://www.theguardian.com/technology/2017/jul/25/roomba-maker-could-share-maps-users-homes-google-amazon-apple-irobot-robot-vacuum

Part I

RGB

CHAPTER 3

Vanishing Points for Image Understanding

Contents

3.1 Introduction . . . 20
3.2 Background . . . 21
3.2.1 Vanishing point detection . . . 21
3.2.2 Inertial data . . . 23
3.3 A fast and simple VP detector using IMU data . . . 24
3.4 Evaluation . . . 26
3.4.1 The York Urban database . . . 26
3.4.2 A new dataset . . . 27
3.5 Live VP estimation . . . 33
3.6 Conclusion . . . 34
3.A Restriction of uncertainty polygons to a set of orthogonal solutions . . . 36

3.1 Introduction

In the context of the estimation of the room layout of *Strong Manhattan* scenes (defined in Section 2.1 on page 10) without depth sensor, the understanding of the scene relies mainly on the analysis of RGB images. Image understanding requires the analysis of the geometric properties of the image: since the perspective projection is a non-invertible mapping between the 3D dimensional scene and the 2D image plane, the depth information is lost, thus making image interpretation a challenging task [13]. Studying and analyzing the geometric properties of an image is thus crucial to recover the spatial layout of the scene.

A well-known geometric entity that can be used as a strong cue for image understanding is the *vanishing point*. Under the perspective projection, parallel lines in the scene are mapped to a pencil of lines that intersect in a so-called Vanishing Point (VP), an image point that is the projection of the intersection of the parallel lines at infinity. In a calibrated camera, a vanishing point gives the 3D direction of the pencil of lines. Detecting a VP can thus provide a strong constraint on the scene geometry. Strong Manhattan scenes consist of three orthogonal dominant directions, *i.e.* there are three main sets of parallel lines. By detecting these three orthogonal VPs associated to the sets of parallel lines, some information about the camera and the scene can be inferred: *e.g.* the camera can be calibrated [30, 212] and its rotation w.r.t. the scene can be estimated [7, 86, 101]. Vanishing points can be used as priors to constrain the 3D reconstruction of such scenes [59], indoor and outdoor scene understanding and reconstruction from a single image [75, 73, 100, 121] and as a fundamental cue for recovering the spatial layout of the scene [163, 184]. Figure 3.1 illustrates these last two applications. Recently, VPs have received a lot of interest in many works dealing with Visual Odometry (VO) robustness [57, 226, 183, 29].

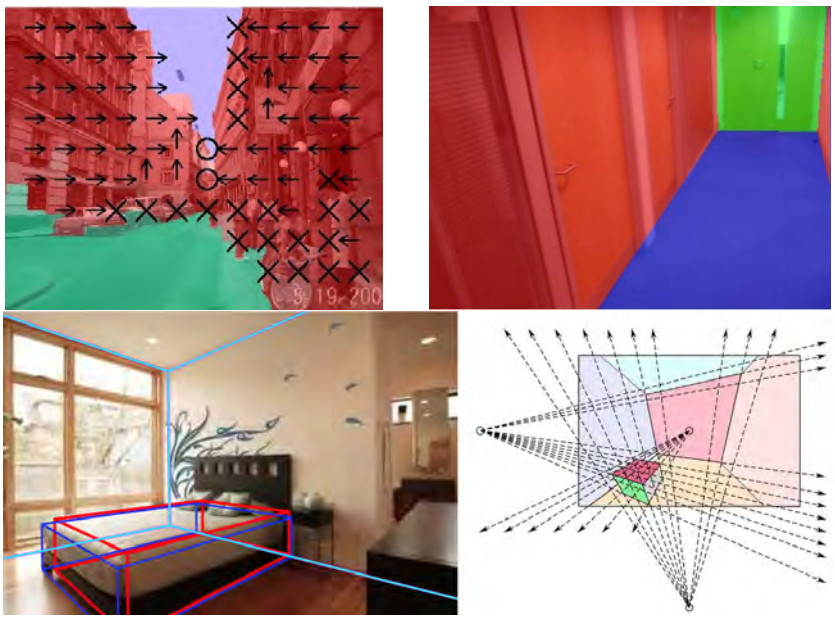


Figure 3.1: VP application examples. Top row: illustration of an outdoor scene segmentation application from [75] and VP guided semantic VO from [57]. Bottom row: illustration of room layout (cyan lines) and box-shaped objects (blue and red lines) estimation from [184].

Another important source of information that can help the interpretation of a scene is the inertial data. In the last years, we witnessed the development and the large diffusion of mobile devices equipped with Inertial Motion Unit (IMU), such as accelerometers, magnetometers and gyroscopes. Thanks to such sensors, the absolute orientation and the gravity vector of the camera can be estimated for each taken picture. Inertial data has been widely used in robotics in combination with the visual data in order to estimate the movement and the pose of the robots [38]. The recent diffusion of mobile devices has fostered their adoption in many computer vision and multimedia applications, such as 3D reconstruction [200], in order to provide a better estimation of the camera movement, especially when

the visual data is affected by, *e.g.*, occlusions and motion blur.

In this chapter, we will mainly focus on the problem of VP estimation, we will see how to take advantage of IMU data to ease this process. We will provide some background on VPs computation from image processing and on inertial sensors in Section 3.2. In Section 3.3, we will describe our approach where the IMU data is considered as a prior for estimating the VPs, and present the results of its evaluation in Section 3.4. Section 3.5 presents a mobile application which integrates our estimation algorithm on a stream of images and IMU data. Finally, we will present further use and limits of VP and conclude in Section 3.6.

3.2 Background

3.2.1 Vanishing point detection

In this section we focus on the VP detection in "Manhattan scenes", in which there exist three dominant, mutually orthogonal directions. For this reason, the VPs corresponding to their converging points in the image plane are called orthogonal VPs.

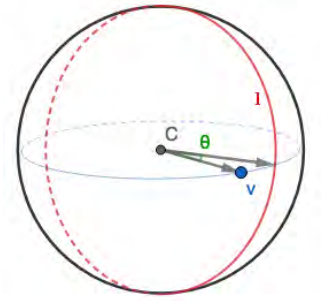


Figure 3.2: Overview of classical VPs estimation approaches relying on feature clusterization. In a first step, features are extracted from the image, here line segments in red. In a second step, the segments are clustered. Each color corresponds to a cluster associated with a vanishing direction. In a third step, the VPs are estimated on each cluster (considering the three biggest clusters), here computing the mutual intersection of the line segments. The VP corresponding to the red line segments is displayed in green.

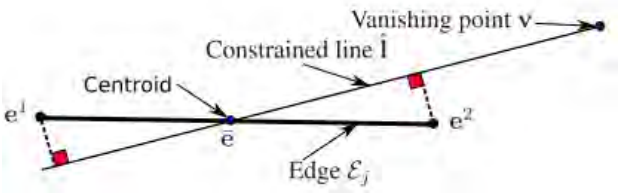
The detection of vanishing points requires the extraction of geometric features in the image, such as image gradients, lines or line segments. Line segments can be extracted with advanced image processing techniques, such as the LSD algorithm [67] based on an a-contrario approach, which has been shown to have a better control of the ratio of false-positive detection. The use of image gradients [34] and other low-level features, provides local orientation information and are mostly used to detect a single dominant VP.

A common approach is to cluster the features to estimate the VPs. Each cluster contains a pencil of lines (or other features) corresponding to parallel 3D lines of the scene. This task can be considered as a classic "chicken-and-egg" problem: if the feature clustering is known, then the VPs can be easily estimated as the point that minimizes a certain distance measure w.r.t. the features of each cluster. Conversely, if the VPs are given, the feature clustering is easily solved by assigning each feature to the "closest" VP (w.r.t. a certain distance measure). Figure 3.2 illustrates this approach. Various techniques have been suggested in the literature for the clusterization of the lines: Hough based methods [13], RANdom SAmple Consensus (RANSAC) frameworks [1, 168, 213] and J-linkage algorithm [201]. The last step relies on the estimation of the VP for each cluster as the point that minimizes the sum of a *consistency measure* between the considered lines and the VP solution. The *consistency measure* is a distance between a point and a line (or line segment), which evaluates the proximity of the point to the line (or line segment.) Several formulations have been proposed for this measure, such as point-line distance error functions [8, 168, 201], orientation error functions [42, 129, 179], or probabilistic error functions [36, 216]. Figure 3.3 illustrates some of them. Finally, the VP is commonly refined with an iterative process such as the Expectation Maximization (EM) approach proposed by [101]. The final orthogonal triplet is then chosen among the possible solutions or, as in [168] the orthogonality constraint can be enforced during the VP estimation process.

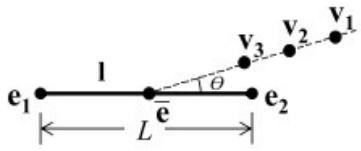
To perform the clusterization and compute the consistency measure, several working spaces have been proposed: the image space [168, 201], the Gaussian sphere [13, 36, 101], and dual spaces [89,



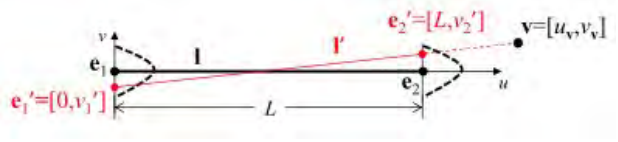
(a) The consistency measure is the angular distance θ between the great circle of the line l red and the VP v in blue.



(b) The consistency measure considered in [201] is the perpendicular distance between the line segment endpoints and the line $\hat{l}$ joining the segment centroid to the VP v. Image courtesy of [201].



(c) The consistency measure considered in [43] is the angular distance θ between the line l and the VP v_1. Images courtesy of [216].



(d) The consistency measure considered in [216] is proportional to the probability of displacement of the line segment endpoints e_1, e_2 (modeled with 1D Gaussian) to be aligned with the line l' joining the line segment centroid to the VP v. Images courtesy of [216].

Figure 3.3: Four consistencies measures. Figure 3.3a is defined on the Gaussian sphere, whereas Figures 3.3b to 3.3d are defined on the image space.

109]. The Gaussian sphere is a unit sphere centered on the optical center of the camera. The projection of the infinite lines detected in the image space corresponds to great circles on the Gaussian sphere. This projection requires the intrinsic parameters of the camera to be known. As illustrated in Figure 3.4, the intersection points of these circles correspond to the associated VPs. The main advantage of this representation is that it can handle infinite points, corresponding to the red circle in Figure 3.4. In addition, a uniform discretization of the Gaussian sphere (in spherical coordinates), offers a uniform angular resolution for all the vanishing directions. The projection of this grid on the image plane generates a discretization of the image plane where the center of the image has a higher resolution, and the distant areas have a lower one. This discretization encodes well the expected accuracy of VPs depending on their location: VPs near to the image center are very accurate since a small perturbation of the associated line segments leads to a small change of the VPs. Conversely, VPs farther away from the image center can be highly changed by small perturbations of their associated line segments. On the Gaussian sphere, it is straightforward to compute the distance between a projected line segment and a point. Therefore, common consistency measures on this space consider a distance between a great circle (corresponding to an infinite line) and a point. Defining a consistency measure on a sphere is not very practical. For this reason, the image plane is the preferred space for defining a consistency measure. Another motivation is that the VP estimation error comes mainly from the accuracy of the extracted features. Since the extraction was performed in the image space, it is more convenient to use this space to model the uncertainty of the features. Dual spaces allow taking advantage of specific properties from another space: for example, the method proposed by Lezama *et al.* [109] is robust as they cast the line clustering problem as a search of 2D points alignments, solved with a known a-contrario method.

Other methods do not follow a clusterization-estimation process but they rather try to solve the problem globally: Bazin *et al.* [17] try to find the rotation (*i.e.* a triplet of VP) that maximizes the number of clustered segments. Antunes *et al.* [8] follow a global approach in which the clusterization and the VP estimation are solved simultaneously as an *Uncapacitated Facility Location* problem.

More recently, the use of Convolutional Neural Networks (CNNs) has become a popular tool for computer vision tasks, and we find such approach to estimate VPs. Zhai *et al.* [221] take advantage of

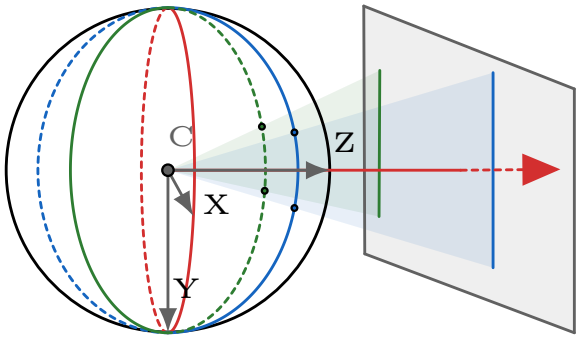


Figure 3.4: The Gaussian sphere is a unit sphere centered on the optical center of the camera. Here we represented the camera coordinate axis (X, Y, Z), with Z aligned with the optical axis. Infinite points correspond to the red circle, which is the intersection of the plane $Z = 0$ and the Gaussian sphere. Infinite lines of the scene or the image plane can be projected on the Gaussian sphere, generating great circles. In this example, the intersections of the vertical blue and green lines correspond to two intersection points of great circles of the sphere: $(\mathbf{0} \ \pm \mathbf{1} \ \mathbf{0})$ (zenith and nadir). The Gaussian sphere enables to easily map the dual relationship between 2D lines and 2D points with the image space. For example, the line dual to the zenith point in the Gaussian sphere is the great circle orthogonal the zenith direction $(\mathbf{0} \ \mathbf{1} \ \mathbf{0})$.

a CNN to generate horizon line candidates, using a prior information for the detection of the horizon line and the zenith VP from line segment features. Kluger *et al.* [98] propose a two-step approach using line segments as features, where a CNN performs the clustering step. Finally, Shuai *et al.* [187] propose a full neural approach, where the input is an image and the output the position of one VP (assumed visible in the image). They obtain successful results on scenes where even few line segments can be extracted.

3.2.2 Inertial data

In the last decades, we witnessed a notable breakthrough in microelectronics which brought low-cost miniaturized silicon sensors to common mobile devices such as smart-phones and tablets. In particular, IMUs usually consist of accelerometers measuring the acceleration of the device, gyroscopes measuring the rate of change of the device's orientation and magnetometer sensitive to the Earth's magnetic field. The IMU measurements can provide accurate information on the device orientation, as well as its velocity and position over a short period of time. Figure 3.5 illustrates how the device orientation can be obtained from these sensors. Accelerometers sense the total device acceleration, which is the sum of the translational device acceleration and the gravity. Gyroscopes sense the device rotational speeds, which after integration give the device orientation change during a short period of time. The fusion of the gyroscopic and the acceleration data allows separating the translational acceleration from the gravity vector, the latter providing the device roll and pitch angles with a high accuracy. The device structure and nearby ferromagnetic materials can interfere with the magnetometer. For this reason, the yaw angle provided by this sensor is extremely noisy. The fusion of the gyroscopic and the magnetometric data allow to stabilize the yaw, but its accuracy is far inferior to the estimated pitch and roll. Nymoen *et al.* [133] evaluated the IMU data of an Apple iPod Touch with a motion tracking system and their results corroborate these differences of accuracy. Finally, the translational device speed can be obtained (up to a constant) by integrating the translational acceleration, with an additional integration to obtain the position.

On the one hand, IMUs can provide measurements at a high frame rate with a low computational cost. On the other hand, they are usually corrupted by different types of error sources such as sensor noises, scale factor and temperature dependent bias, which are nonlinear and difficult to characterize [186]. They provide derived measures (acceleration and angular velocity), which need to be integrated to compute the current position and attitude, thus causing error accumulation and a significant drift in the position and the attitude over the course of time. These problems can be mitigated by employing optimal estimation and filtering techniques such as a Kalman filter [90].

We saw these sensors can provide a reliable pitch and roll angles on the long term, as well as its yaw angle, speeds and device position, which can only be considered for short periods of time.

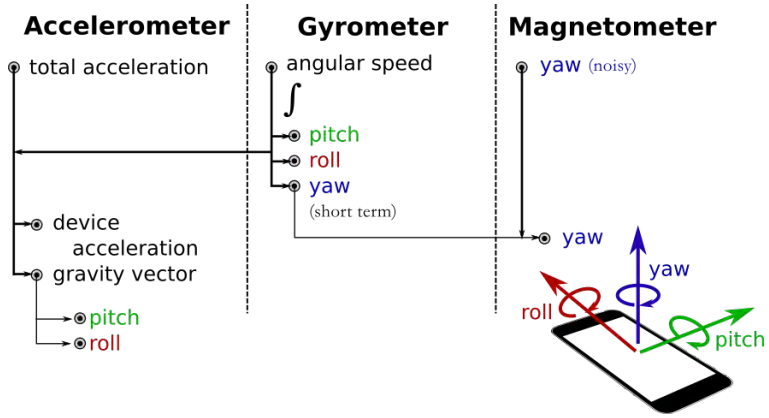
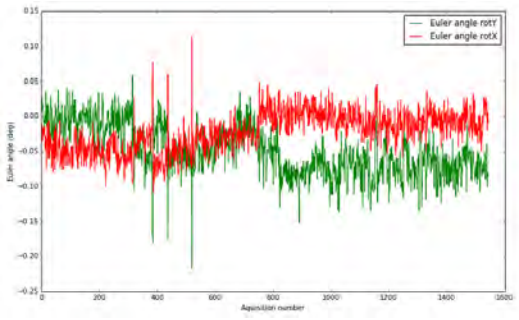
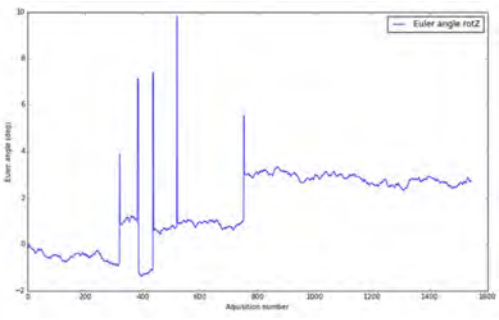


Figure 3.5: IMU data can be used to reliably compute the pitch and roll angles of a device by fusion of the accelerometer and the gyroscope data. The accelerometer can estimate the gravity vector alone, but it is sensitive to translational accelerations. The gyroscope only senses the rotational speed of the device, and after integration, it can give also the orientation of the device, up to a constant. The integrated values suffer from important drift in the long term, but contrary to the accelerometer, it is not sensitive to translational accelerations The fusion of the two sensors data enables to compensate each other's weakness, and accurately estimate the pitch and roll angles. Finally, the yaw angle (or heading) can be estimated from the fusion of the magnetometer data which is very noisy and the integrated gyroscopic data.

Figure 3.6 illustrates the deviations of the roll, pitch and yaw angles of an iPhone 4 standing still for 53 s: the yaw angle suffers from a 4° drift, while the roll and pitch angles suffer from a 0.2° drift. All these measurements can be obtained in real-time, and even when the device is in motion.



(a) The red and green curves correspond to the roll and pitch angles (in degree) respectively during 53 s (1600 samples).



(b) The blue curve corresponds to the yaw angle (in degree) during 53 s (1600 samples).

Figure 3.6: Roll, pitch and yaw angles of an iPhone 4 standing still on a table, provided by the iOS Core Motion framework (acquisition rate at 30 Hz). Two components of the noise can be observed here. On small windows of time (around 50 samples), the white noise makes the angular values fluctuate very rapidly. On larger windows (500 samples), the sensor bias slowly affects the average value of the angle.

The fusion between inertial and visual data is thus becoming an interesting topic because of their complementarity. Inertial data is indeed computationally cheap but suffers from drift and measurement noise; visual data can provide more precise and stable measurements but it is, in general, computationally more expensive. In the case of VP detection, the orientation and gravity vector provided by the IMU can be used as priors for driving and easing the process of VP detection.

3.3 A fast and simple VP detector using IMU data

In this section we propose a simple and fast method to estimate VPs, taking advantage of an IMU. We will try to demonstrate then the simplicity of this method can lead to accurate results.

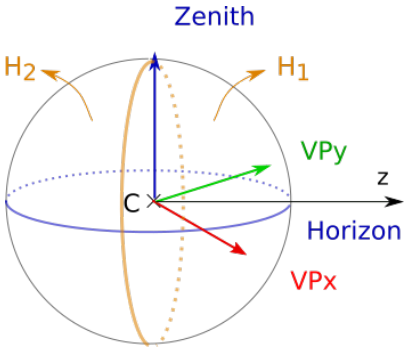


Figure 3.7: In the Gaussian sphere, the three VPs form an orthogonal frame. C is the optical center of the camera. The knowledge of the zenith enables to reduce the search of the two VPs orthogonal to the zenith, VP_x and VP_y on the horizon line.

Our approach relies on the use of the IMU, which provides the gravity vector, *i.e.* the direction of the vertical lines of the scene, and thus equal to the opposite of the zenith. Under the assumption of a Manhattan scene, there are two other VPs: VP_x and VP_y, which form an orthogonal frame with the zenith (see Figure 3.7). This means the detection problem of VP_x and VP_y can be thus reduced to the search of two orthogonal VPs along the horizon, *i.e.* along a line instead of the 2D space of the image plane.

Our method relies on the extraction of line segment features to detect the VPs. We performed a comparison of four algorithms: [120, 201, 130, 67] and chose [67], which obtained the highest sensitivity and precision. We estimate an approximated value of the zenith from the gravity vector. To refine this VP, we use the Gaussian consistency measure (see Figure 3.3a) to filter the segments converging to the approximated zenith, and solve the following linear least squares estimation problem:

$$\min_{\mathbf{v}} \sum_{\mathbf{l} \in L} (\mathbf{l}^{\mathsf{T}} \mathbf{v})^2 \tag{3.1}$$

To estimate VP_x and VP_y, we consider the intersections of the remaining lines (not converging to the zenith) with the great circle H of the horizon in the Gaussian sphere (the red circle in Figure 3.7), which is the dual of the zenith. As mentioned earlier, working in this space enables to consider infinite VPs (corresponding to the invisible intersection of the parallel lines in the image space), and offers a uniform angular resolution for all the vanishing directions.

The Gaussian sphere can be divided into two hemispheres H_1, H_2 along the plane $z = 0$ (see Figure 3.7, the two hemispheres are separated by the orange great circle), where H_1 corresponds to the points of the Gaussian sphere with $z \geqslant 0$. The two hemispheres are anti-symmetric since, in projective geometry, two homogeneous 3-vectors p and $-p$ correspond to the same 2D point. Therefore the search of VP_x and VP_y can be restricted to the semicircle corresponding to the points of H satisfying the constraints $z \geqslant 0$. In addition, since VP_x and VP_y are orthogonal, we only search for one of the two, let's call it VP_x, which belongs to a quarter of circle of H. The clustering is performed with a 35 bins accumulator, corresponding to 35 equal circle arcs of the quarter of circles. We refine VP_x with the same method employed for the zenith.

The parameters influencing our approach are: the intrinsic calibration parameters (a bad calibration can deteriorate the results), the choice of the consistency measure, the threshold for the value of the consistency measure, which determine whether the line is considered as converging or not to the VP to refine and the number of bins: a low number of bins decreases the accuracy of the clustering, a very high number of bins can prevent from finding the correct maxima (the intersections corresponding to the expected maxima are spread on too many bins), and a minimal number of filtered lines by the consistency measure to consider the VP as reliable.

3.4 Evaluation

3.4.1 The York Urban database

We first start to evaluate our work on the *York Urban Database* [42, 43], published in 2008, which was the first extensive dataset for VPs estimation algorithms evaluation in Manhattan scenes. It is the most popular dataset used by most of the works to assess the effectiveness of the proposed method.

3.4.1.1 Results

We used the ground truth VPs to simulate pitch and roll angles. The line segments considered as features were computed with the LSD algorithm [67]. In a second experiment, we applied a random perturbation (3° maximum) to simulate the offset bias of IMUs. We evaluated our results on each VP separately, considering the angular consistency measure defined in [43] (see Figure 3.3c) on the ground truth line segments and the corresponding estimated VP. We computed the Root Mean Square (RMS) of these measures and we generated a cumulated histogram of the RMS represented in Figure 3.8. VP_z corresponds to the zenith, while VP_x, VP_y correspond to the VPs on the horizon.

Figure 3.8 shows the results of our evaluation. Overall, the results obtained with the simulated IMU are comparable to Tardif J-Linkage algorithm (JL) [201] results, whereas when we apply a random noise on the simulated IMU, the obtained results are worse than the other two methods. This demonstrates our approach is significantly affected by the accuracy of the IMU. For VP_x, Figure 3.8a shows JL [201] are slightly better than our best results, whereas, for VP_y, Figure 3.8a shows our best results are much better than JL. For the estimation of VP_z, our results (without random noise on the simulated IMU) should be the best ones since they correspond to the ground truth zenith. As shown in Figure 3.8c, the results of JL [201] for the estimation of VP_z are slightly better than ours without noise. This may be surprising as we were using the GT as initial guess.

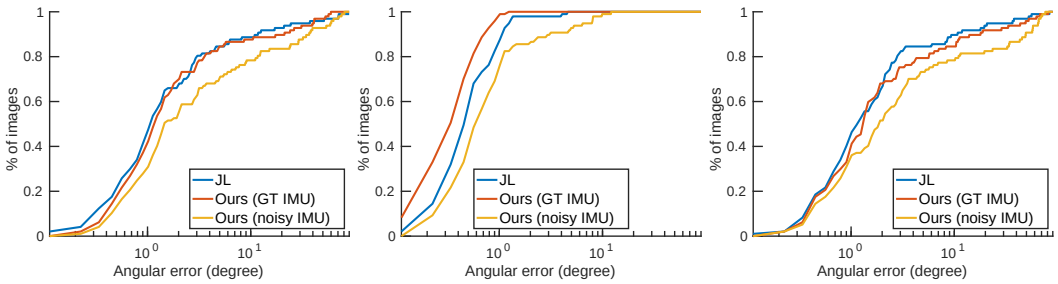


Figure 3.8: Cumulative histograms of the RMS error for VP_x, VP_y, VP_z respectively (higher is better).

3.4.1.2 Discussion

In this dataset, the ground truth VPs are estimated using the algorithm proposed by [36] using a statistical framework where each VP is estimated separately. Because no orthogonality constraint is enforced, an orthogonal frame is fitted to each triplet to enforce the constraint. This yields to an orthogonal solution which is not necessarily optimal given the statistical distribution of the line intersections used for the estimation. The resulting Manhattan directions, indeed, can be quite far from the line segments intersections as it can be seen in Figure 3.9. The obtained orthogonal solution might be a biased solution that may not be suitable to be used as a reference to evaluate and compare VPs estimation algorithms. Other datasets, such as the *PKU Campus Database* [110, 111] suffer from the same problem.

The evaluation of VPs algorithms is not consensual: some authors use their own consistency measure applied to the clustered line segments and the ground truth vanishing point as metric [42, 201, 8]. Bazin *et al.* [17] compare the number of inliers lines. In recent works, only the estimated horizon is compared with the ground truth [212, 216, 221, 109, 98]. Also, these evaluation approaches assume the provided reference VPs are not biased, which is not the case of the previously mentioned datasets.



Figure 3.9: The green square is the estimated VP by [42] (associated horizon in red). The light green point (associated horizon dashed) is the VP after orthogonalization of the Manhattan directions: it lies far from the common intersection zone of the associated line segments. This last VP cannot be used as a reference to evaluate algorithms.

This first evaluation was a proof-of-concept for testing our algorithm on the existing datasets. In order to evaluate the effectiveness of our approach, we created a new dataset embedding IMU data as well.

3.4.2 A new dataset

In this section, we describe how we created our dataset and the results of the evaluation we carried out on it.

3.4.2.1 Ground truth creation

The construction of a vanishing points dataset requires two elements: photos and reference vanishing points. Creating reference VP is a hard and challenging task, even if the images are manually annotated: as pointed out in [216], many deviations from a perfect imaging system such as camera noise, camera calibration errors, line segment extraction error, *etc*. affect the estimation of the ground truth orthogonal VPs, and only optimal or sub-optimal solution can be found for them. The only way to have *real* ground truth data for the VP would be the use of synthetic images, in which all the parameters are known by design, or using real images and highly accurate and costly instruments (*e.g*. electronic theodolites) to measure the actual attitude of the camera w.r.t. the Manhattan scene. Also, the generation of reference VPs from the ground truth line segments relies on an arbitrary choice of a VP estimation algorithm. Figure 3.10 illustrates this problem.

A more meaningful approach that we are proposing in this dataset is to provide an uncertainty region for the locations of the VPs, as opposed to single points. This information can be used to reject or accept the solution of an algorithm (the solution is respectively outside or inside the region).

We decided to compute the reference VPs with hand-labeled line segments, which must be accurately drawn. The uncertainty of a ground truth line segment comes from the selection of the two extrema, which can be modeled with circular regions of uncertainty around the extrema (see Figure 3.11). Shufelt [188] was the first to introduce the error modeling for line segment endpoints in a VP detection algorithm. The true position of a line segment endpoint is assumed to lie *among all the possible locations within its pixel*. The lines connecting all these possible endpoints sweep an area which is bounded by two lines, $\mathbf{l}_1$ and $\mathbf{l}_2$ as in Figure 3.11. This area is called a *double wedge* [19] (the gray area in Figure 3.11). In his proposed method, the Gaussian sphere is divided into accumulators, each wedge region is projected on the sphere and the corresponding accumulators are incremented. The maxima on the sphere then represent the directions of the VPs.

More recently, Xu [216] introduced a probabilistic consistency measure, which models the uncertainty of endpoint locations with a 1D Gaussian which is then used in an EM framework to estimate the VPs. Contrary to Xu, we followed a geometrical approach because our objective is to compute a

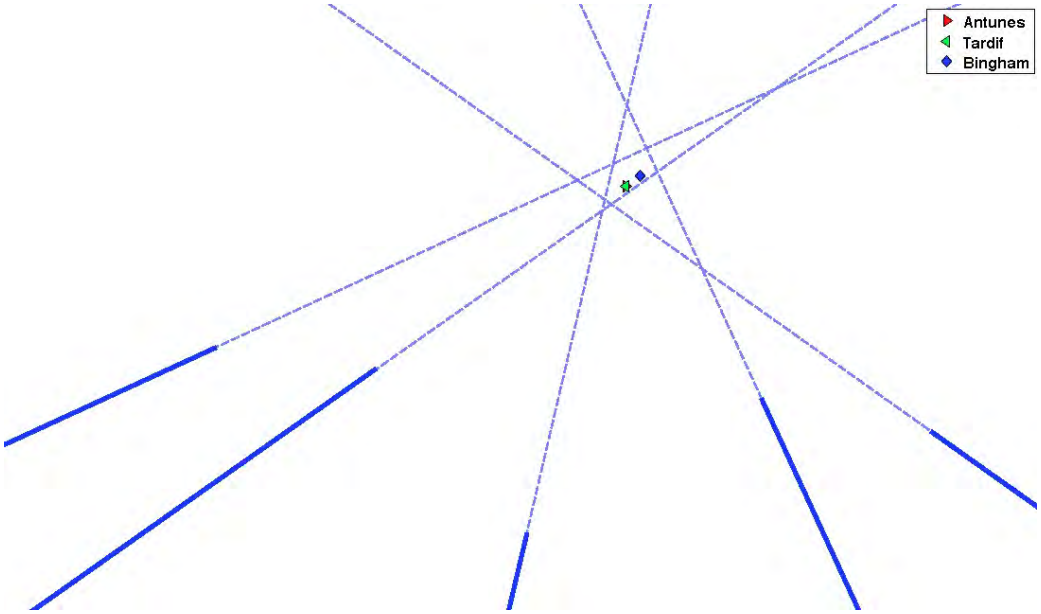


Figure 3.10: The manual creation of the line segments does not generate a single intersection point. Several VP estimation approaches are possible, here [36, 201, 8] in blue, green and red respectively, return different solutions. Which one is the best?

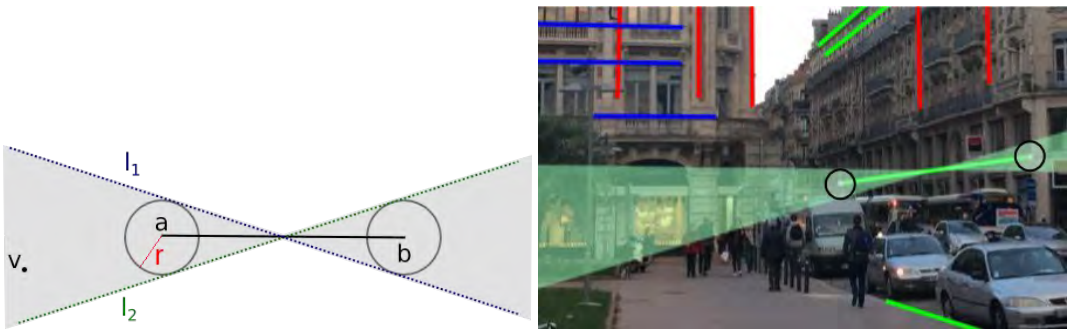


Figure 3.11: The uncertainty of a ground truth line segment is modeled with circular regions of uncertainty around the two extrema $\mathbf{a}$ and $\mathbf{b}$. The lines connecting all these possible endpoints sweep an area bounded by two lines, called *double wedge* (the area in grey), in which the associated VP $\mathbf{v}$ should lie. We denote r the radius of the circular region of uncertainty of the line segment endpoints. On the right, we represented a double wedge corresponding to one of our ground truth line segment.

confidence region for the solution, rather than finding one VP solution. In the confidence regions, all the possible VPs are equiprobable since we do not assume it is less unlikely to commit a two-pixels error on an endpoint rather than one pixel. In this sense, our approach to finding the regions is closer to [188], except we work in the image plane, and we do not use accumulators but compute the exact geometric intersection of the double wedges.

Assuming the real line associated with the annotated line segment is contained in its double wedge, the intersection of all the double wedges of a given line segment cluster forms a region in which the VP should lie (see Figure 3.12). This region can be empty when an outlier line segment is created, or when the operator who creates the ground truth is too inaccurate.

As illustrated in Figure 3.13, double wedges model the uncertainty of the line segments, as they naturally take into account the length of the segments. In general, long line segments should be more robust as they mitigate the annotation error of the two extrema. A long line segment, indeed, has a thinner double wedge, and thus it will contribute to narrow down the uncertainty region of the associated VP. Conversely, short segments have wider wedges which do not help to reduce the uncertainty region.

We reformulate the double wedge intersection problem in term of Boolean operations on half-

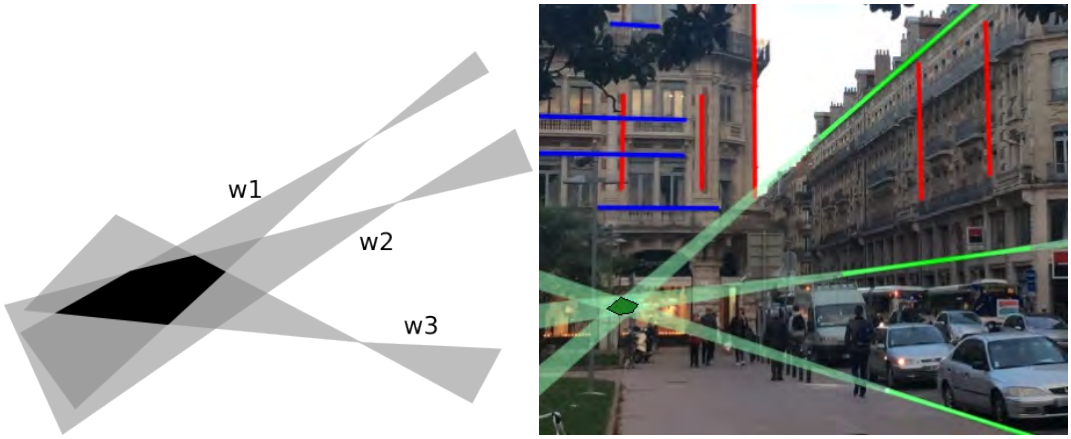


Figure 3.12: The intersection of the different double wedges $w1, w2, w3$ associated to the image of parallel lines of the scene is a convex polygon in which the VP should lie. On the right, we represented on a photo from our dataset, the intersection of the double wedges corresponding to the ground truth line segments.

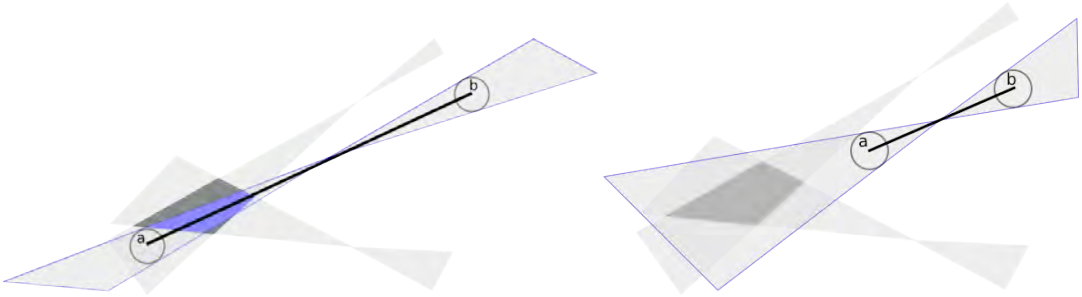


Figure 3.13: Left: Long line segments provide more information: they narrow down the uncertainty region of the VP. Right: short ones have no influence on the uncertainty region.

planes. Let $\mathbf{l}_1$ and $\mathbf{l}_2$ be the bounding lines (see Figure 3.11) of a segment $\overline{\mathbf{a}\,\mathbf{b}}$. Without loss of generality, consider the half-planes h_1 and h_2 bounded by $\mathbf{l}_1$ and $\mathbf{l}_2$ respectively, and both containing $\mathbf{a}$. The double wedge w associated to $\overline{\mathbf{a}\,\mathbf{b}}$ is defined as

$$w = (h_1 \cap h2) \cup \left(h_1^{\complement} \cap h_2^{\complement}\right), \tag{3.2}$$

where $h_i^{\complement}$ denotes the complementary of h_i, *i.e.* the other half of the plane.

The intersection of the double wedges of all the line segments thus requires the computation of the intersections and unions of the half-planes h_i of each line segment, which is a well-known computational geometry problem treated in [19]. The computation of the intersection is performed in the projective plane, which is equivalent to performing the computation on the Gaussian sphere: this allows us to compute the intersection of parallel lines and to handle the case of VPs at infinity.

3.4.2.2 Data collection methodology

The dataset contains 114 photographs (40 indoor and 74 outdoor). The photos were taken at different moments of the day and therefore have various exposures. A majority of the indoor scenes contain low levels of clutter (chairs, sofas, ...). In contrast, a majority of outdoor scenes contain a lot of occluding objects such as trees and vehicles, making the estimation of VP more challenging. The photos were taken holding the camera in different attitudes in order to have a sufficient variety of poses: post-hoc analysis revealed a mean and maximal absolute angular value between the camera principal axis and the horizon of $6.7°$ and $26°$ respectively. Figure 3.14 shows some selected photos from the dataset.

We collected the photos using an iPad Air 1 running iOS 8 in landscape mode with a 1920×1080 resolution and using the following iOS capture presets: automatic white balance, auto exposition, and fixed focus. The auto-focus was disabled because it can add a significant random lag between the moment the shutter button is pressed and the effective shot of the photo.



Figure 3.14: Some photos of the dataset with their ground truth line segments (red, green, blue), the horizon line computed from the IMU data (cyan line) and the polygon of the uncertainty region computed on the red line segments. In the photo on the left, the horizon line computed with the IMU data does not intersect the yellow polygons because of the bias of the IMU data.

Instead of using the raw data values of the accelerometers, gyroscopes and the magnetometer, we used the `CMDeviceMotion` class of the iOS SDK which provides high-level data such as the gravity and the attitude of the device through sensor fusion algorithms not detailed in the official documentation. A 30 Hz sampling rate was set to collect the IMU data. We developed a specific application for recording the device orientation provided by the `CMDeviceMotion` class along the taken photos. The source code of the application is available for download at the dataset website http://ubee.enseeiht.fr/tvpd .

Camera and IMU calibration The camera was calibrated offline using Bouguet camera calibration toolbox [24] to estimate the intrinsic parameters. Experiments on the IMU sensors holding the iPad on a try square shown that in the worst case, we could obtain 2° of error on the roll and pitch values. This bias is visible in the Figure 3.14, where the horizon lines computed with the IMU data do not intersect the polygons of the uncertainty regions associated to the VPs orthogonal to the zenith (VPx and VPy). In addition, no calibration of the IMU sensors is performed since our observations revealed that the flatness of the ground is less reliable and repeatable than the orientation values returned without setting a reference attitude, *e.g.* the ground.

A known issue affecting mobile devices is the synchronization between the data provided by the IMU sensors and the image provided by the camera [71]. Our preliminary experiments demonstrated that for our device the IMU data could be not synchronized w.r.t. the orientation computed using the image. The mean lag between the IMU data and the camera frames was found to be 16 ms with a standard deviation of 140 ms. To take into consideration this uncertainty and provide smoother data, we computed the attitude matrix as the average rotation [39] over a time window covering the mean lag measured during the preliminary experiments.

Line segments creation In order to generate the ground truth, a web application has been developed (see Figure 3.15) to let the users accurately draw the line segments and to associate them with one of the three Manhattan directions (as in [42]). The source code of the application is also available for download at the dataset web-page. A post-hoc analysis revealed a mean of 16.7 segments per photo. We assumed a 4-pixels accuracy around the endpoints clicked by the users, in other words, the radius r of the circular region of uncertainty of the line segments endpoints is 4 pixels.

This value has been determined experimentally as the average value that ensured that the intersection of the double wedges was not empty and contained the VP solution provided by [8].

We also made comparisons on the York Urban Database, see Figure 3.16. As expected, the VPs computed with [36] lie in our uncertainty regions. Since the orthogonalization process is independent of the line segments, the orthogonalized VP do not always lie in the uncertainty regions (see Section 3.4.1).

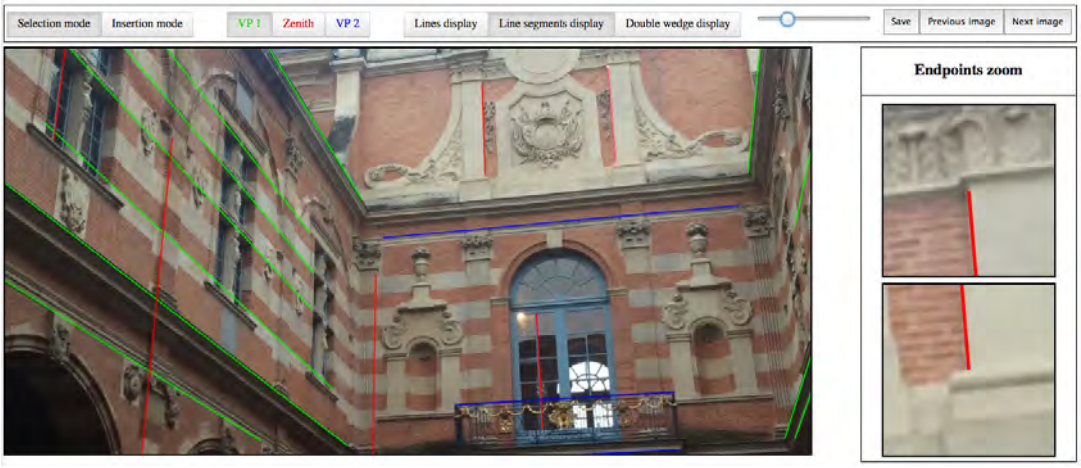


Figure 3.15: The web application used to annotate the images with line segments.

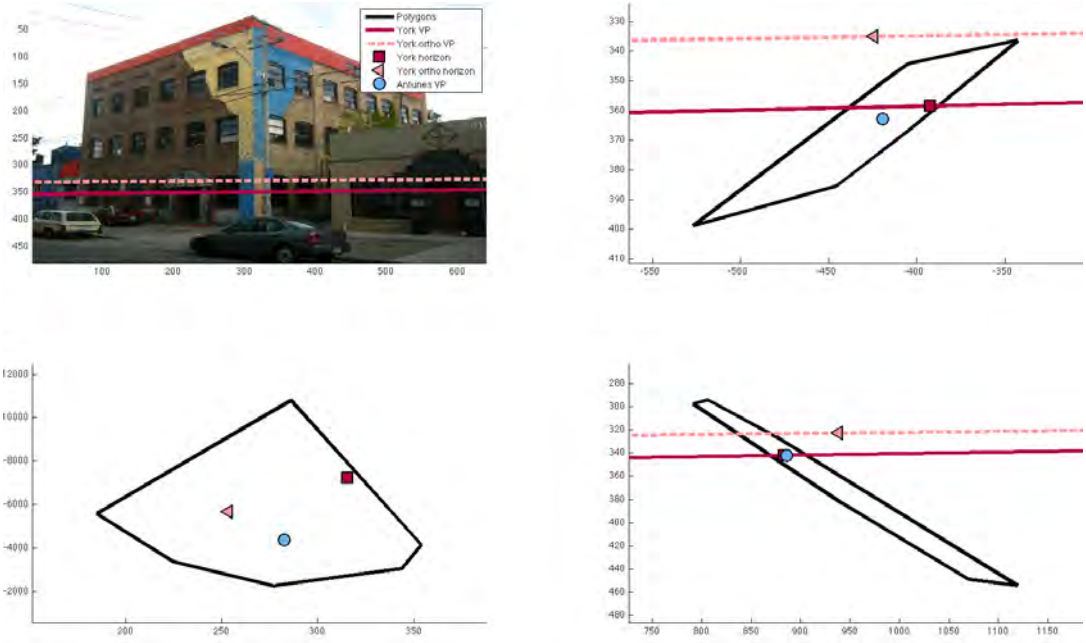


Figure 3.16: Comparison on the York Urban Database. The red squares (VP provided by the dataset using [36], associated horizon in red) and the blue circles (VP computed using [8]) lie in our uncertainty regions. The orthogonalized VP are represented with pink triangles (associated horizon dashed) are not in our polygons.

3.4.2.3 Results

We saw earlier the uncertainty polygons can be used to evaluate a VP estimation algorithm: a point-in-polygon test can be used to validate or reject the VP if it is inside or outside the polygon. Besides the segments, the size of these polygons depends on the circle radius r of the uncertainty on the line segment endpoints. The lower is r, the smaller is the polygon. But what if we want a test defined on a continuous domain (independent of r) instead of a binary test? To address this question, we propose the evaluation metric

$$D(\mathbf{v}, S) = \max_{s_i \in S} D_{\text{JL}}(\mathbf{v}, s_i) \tag{3.3}$$

where S is a cluster of segments and $D_{\text{JL}}(\mathbf{v}, s) = \text{dist}(\mathbf{e^1}, \hat{l})$ is the consistency measure defined by Tardif [201], where $\hat{l}$ is the line joining the midpoint of the segment s with $\mathbf{v}$, and $\mathbf{e^1}$ is an endpoint of s. As illustrated in Figure 3.3b, $D_{\text{JL}}(\mathbf{v}, s)$ gives the minimum values of the radius r of the double wedge associated to the line segment s, so that the corresponding vanishing point $\mathbf{v}$ lies in it. Our evaluation

metric $D(\mathbf{v}, S)$ gives the minimum value of the radius r of the cluster S so that the vanishing point $\mathbf{v}$ lies in the corresponding uncertainty polygon. The isocontours of $D_{\text{JL}}(\mathbf{v}, s)$ displayed in Figure 3.17, correspond to the uncertainty polygons defined for different values of r.

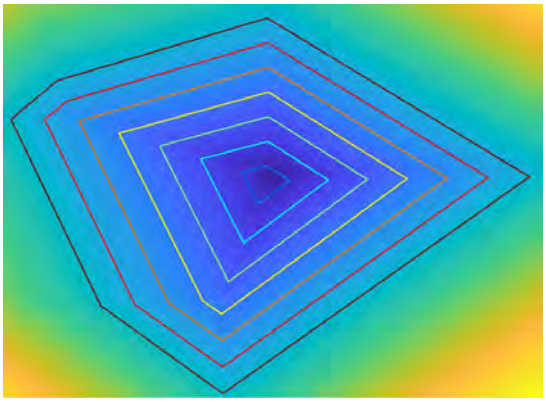


Figure 3.17: Different uncertainty regions computed for different values of r, the radius of the circular region of uncertainty of the line segment endpoints. Our evaluation metric $D(\mathbf{v}, S)$ is represented by a color map. $D(\mathbf{v}, S)$ gives the minimal radius r of all the double wedges of a cluster of segments so that it lies in the associated uncertainty polygon. The uncertainty polygons correspond to the isocontours of $D_{\text{JL}}(\mathbf{v}, s)$.

Method	VP_x	VP_y	VP_z
JL [201]	32–44%	54–60%	75–80%
Ours	46 %	57 %	64 %

Table 3.1: Percent of VPs lying in the ground truth uncertainty polygon using the point-in-polygon test. Best results are displayed in bold. Since JL [201] returns non-deterministic results, the worst and best results obtained are reported.

As in the comparison on the York Urban Database, we used line segments from the LSD algorithm [67] and compared our approach with JL [201]. Table 3.1 and Figure 3.18 shows the results obtained on our dataset with the binary point-in-polygon test and the metric $D(\mathbf{v}, S)$, respectively. Given the point-in-polygon test, our approach estimated successfully 46–64% of the VPs of our dataset. A similar test carried out on the York Urban Database showed JL [201] and our approach obtained 70–100% of point-in-polygon success. This means the images from our dataset are more challenging than the ones of the York Urban Database. The two evaluations show the zenith is the VP detected with the most of success and accuracy, and again, JL approach [201] performs better than ours. Regarding the two other VPs, our results are again very similar to the ones obtained with JL approach [201].

These results demonstrate the use of an IMU allows to accurately estimate VPs with a simple and efficient algorithm and to achieve similar results than more complex and time-consuming approaches such as [201].

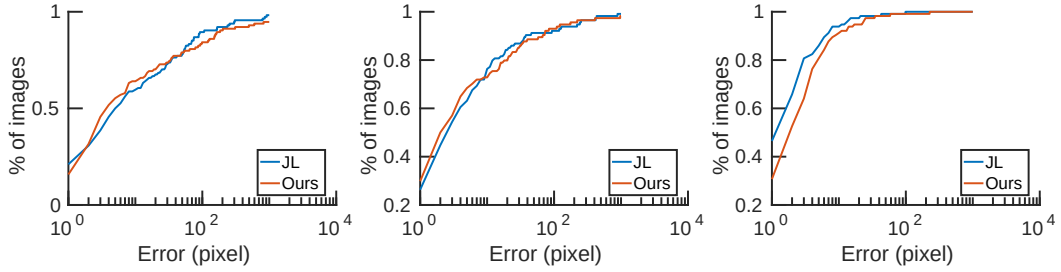


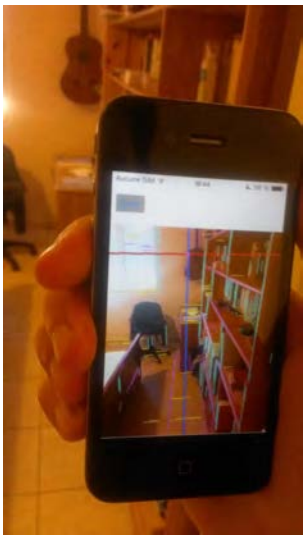
Figure 3.18: Cumulative histograms of the error for VP_x, VP_y, VP_z, respectively, using the metric $D(\mathbf{v}, S)$ (see Equation (3.3)) on our dataset (higher curve is better).

3.5 Live VP estimation

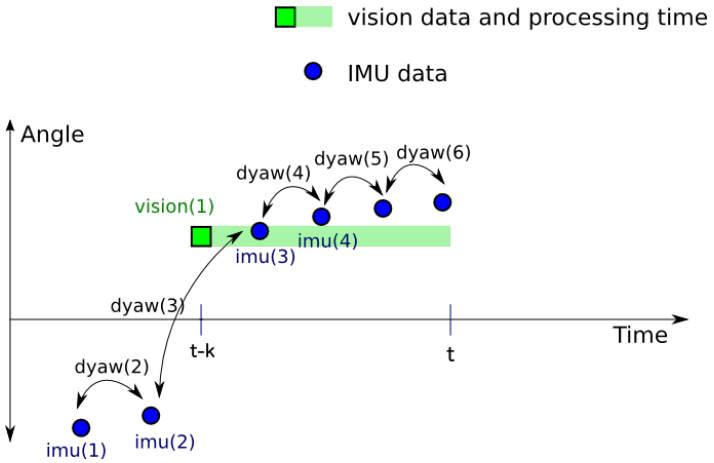
We integrated our VPs estimation algorithm into a proof-of-concept mobile application on an iPhone 4 in order to evaluate our approach on a stream of images and IMU data. Since all the images from the stream cannot be usable for VPs extraction (*e.g.* the image can be too blurred, or the viewpoint does not contain enough line segments), we also tried to take advantage of the inertial data to provide an orientation when a pure vision approach would fail.

We implemented our tracking algorithm using a Kalman filter [90] with a single state variable representing the yaw angle correction between the orientations estimated from the inertial data and the VPs. As we saw in Section 3.2.2, the yaw angle estimation from initial data is the least reliable and a visual observation of this angle is enough to reveal a significant drift. Figure 3.19a shows a screenshot of our application, which displays on the screen a visual compass based on the IMU data corrected with the VPs estimation. The IMU data is used in the prediction step, while the orientation provided by the VPs is used for the measurement update.

On a 2010 iPhone 4, the extraction of the line segments with [67] approach took $300\,\mathrm{ms}$ on 640×480 images, and $9\,\mathrm{ms}$ for the extraction of the VPs. As a reference, it must be noted that the same code was running approximately 10 times faster on a 2013 iPad Air and we expect that it could be as fast on a more recent version of the iPhone. Also, in comparison, the VP estimation of [201] is 100 times slower than our approach, on a desktop computer. With the slow processing time on the iPhone 4, there was a large difference of acquisition rate between our input data: the yaw angle estimated from the VPs at $3\,\mathrm{Hz}$, and the estimation from the inertial data provided at $30\,\mathrm{Hz}$ by the iOS Core Motion framework. It also created a problem of delayed measurement illustrated in Figure 3.19b: between the moment an image was captured at time $t-k$ and the end of the VPs extraction at time t, several values of the yaw angles had been obtained from the IMU.At time t, a rollback of the state at time $t-k$ was performed to apply the measurement update, then the predictions from the IMU data between $t-k$ and t were performed again, on the corrected state this time. Figure 3.20 illustrates the results obtained with the Kalman filtering.



(a) Live VPs estimation on an iPhone 4. The red horizontal line corresponds to the horizon. The blue line is the vanishing line dual to VP_y. The intersection of the blue and red line corresponds to VP_x.



(b) Illustration of the delayed measurement problem.

Figure 3.19

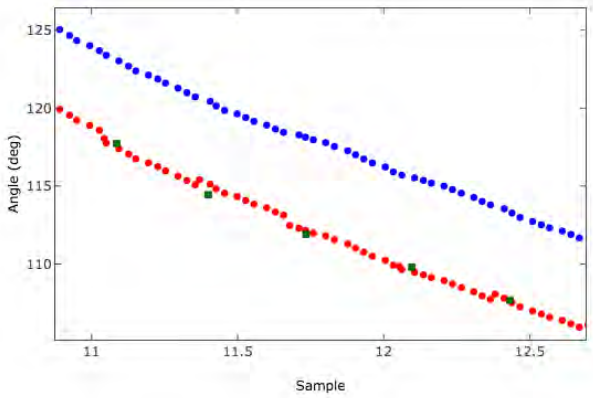


Figure 3.20: The yaw angles estimated from the IMU and the VPs are represented by the blue and green points respectively. The red points represent the yaw angle resulting of our filtering. As expected, the red points are sticking to the values estimated from the VPs (with a delay equal to the time between two VPs estimation). A smaller jitter can be observed on the red points, showing the VPs estimation were lacking accuracy, probably due to the low resolution of the images considered. The experiment had started with $0°$ of difference between the inertial and VP-based estimation of the yaw angle. After less than $11\,\mathrm{s}$, there were $5°$ of difference.

3.6 Conclusion

In this chapter we introduced a novel VP estimation algorithm: as it takes advantage of the IMU data, this fast and lightweight approach can be easily integrated into other mobile applications, such as the room layout reconstruction. Our method demonstrated accurate results comparable with other state-of-the-art algorithms such as [201]. To evaluate the effectiveness of our approach, we created a new dataset, which, to the best of our knowledge, is the only one including IMU data. Instead of providing *real* ground truth data for the VPs, we opted for a more meaningful approach consisting in computing uncertainty regions for the location of the vanishing point. These regions are provided in the form of polygons and are computed by intersecting the double wedges of the ground truth line segments. The works related to the creation of this dataset have been presented at the oral session of an international conference: [5] Vincent Angladon *et al.* "The Toulouse vanishing points dataset". In: Proceedings of the 6th ACM Multimedia Systems - MMSys 2015, ACM Press, 2015.

Vanishing points are an interesting geometric cue for image understanding, especially for the segmentation of indoor scenes in order to detect the walls, the floor, and the ceiling. The segments belonging to the wall corners can be clustered and the associated vanishing points can give an initial guess about the orientations of the planes they belong to. This problem is in general ill-posed though, as there are several possible spatial layouts which could explain a set of line segments. Therefore more information should be gathered from the image to find the most coherent of the layouts. Lee *et al.* [103] propose a geometric approach where the analysis of the area swept by the line segments w.r.t. their VP define orientation maps corresponding to the orientations in the image of the planes of the walls, ceiling, and floor. Hedau *et al.* [73] use contextual information from a prior segmentation of the image to select the most appropriate layout candidate. Schwing *et al.* [184] jointly optimize the layout and the detected objects, with constraints to ensure the position of the objects is coherent with the estimated layout. More recently, real-time approaches have been proposed, such as Yang *et al.* [218] who take advantage of a CNN.

Despite these efforts, the failure rate of these methods is still quite high ($> 10\,\%$) to consider their use in an industrial application. Figure 3.21 illustrates an example of spatial layout estimation failure on an image of our database with the method proposed in [73]. They also revealed to be quite sensitive to outlier segments (line segments corresponding to textures and clutter in the room), and missing (undetected) line segments.

Multi-view approaches have also been proposed, such as [56, 58, 12], but they all require as input

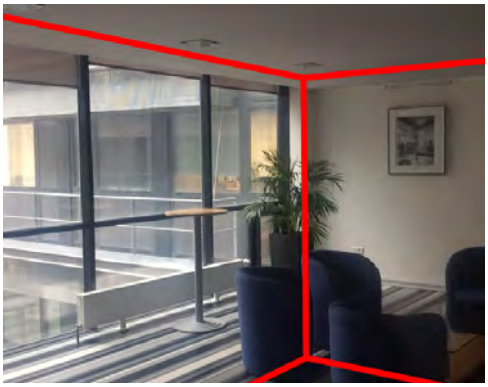


Figure 3.21: An example of spatial layout estimation failure using [73] approach.

the camera poses and a sparse point cloud, estimated from a Structure from Motion (SfM) or a Visual Simultaneous Localization And Mapping (vSLAM) algorithm. More recently, joint approaches are being proposed [37, 174, 219], where the vSLAM algorithm enables the room layout estimation, and the estimated planes are used to improve the robustness of the vSLAM algorithm. At the time we were investigating this topic, SfM or a vSLAM algorithm did not perform well on mobile devices (*e.g.* ARKit and ARCore did not exist). In the following chapters, we will rather consider the use of depth sensors for mobile devices, which allows obtaining more directly the orientation of planar structures.

3.A Restriction of uncertainty polygons to a set of orthogonal solutions

Once we have computed the uncertainty polygon for each VP, under the assumption of strong Manhattan World we could refine the size of these polygons by enforcing the mutual orthogonality of the VPs. Let P_1, P_2, and P_3 be the polygons resulting of the double wedge intersections. We want to compute for each P_i the regions $P_i^+ \subseteq P_i$ containing all the mutually orthogonal triplets. More formally, we would like to compute P_1^+, P_2^+, P_3^+ so that

$$\forall (i,j,k) \in \{(1,2,3),(2,1,3),(3,1,2)\}, \forall \mathbf{p_i} \in P_i^+, \exists \mathbf{p_j}, \mathbf{p_k} \in P_j^+, P_k^+ : \begin{cases} \mathbf{p_i}^\mathsf{T}\mathbf{p_j} = 0 \\ \mathbf{p_i}^\mathsf{T}\mathbf{p_k} = 0 \\ \mathbf{p_j}^\mathsf{T}\mathbf{p_k} = 0 \end{cases} \tag{3.4}$$

where $\mathbf{p_i}$, $\mathbf{p_j}$ and $\mathbf{p_k}$ are expressed in homogeneous normalized camera coordinates.

In order to refine the polygons to enforce the orthogonality constraints, we devised an iterative algorithm, Algorithm 1, that exploits the duality of the projective space. At each iteration, it updates the polygon as the result of the intersection of the polygon itself and the duals of the other two polygons. Figure 3.23 illustrates the obtained polygons after the first iteration of the algorithm, while Figure 3.24 illustrates the different iterations. In practice, we remarked that, due to the uncertainty of the camera intrinsic parameters, after several iterations, we sometimes found empty intersections of the polygons.

Algorithm 1: Restriction of the set of solutions to a set of orthogonalized solutions

Data: P_1, P_2, P_3
Result: $\hat{P_1^+}, \hat{P_2^+}, \hat{P_3^+}$ an approximation of the solution

$P_1', P_2', P_3' \leftarrow P_1, P_2, P_3$;
while *no convergence of* (P_1', P_2', P_3') **do**
  $P_1'' \leftarrow P_1' \cap {P_2'}^\perp \cap {P_2'}^\perp$;
  $P_2'' \leftarrow P_2' \cap {P_1'}^\perp \cap {P_3'}^\perp$;
  $P_3'' \leftarrow P_3' \cap {P_1'}^\perp \cap {P_2'}^\perp$;
  $P_1', P_2', P_3' \leftarrow P_1'', P_2'', P_3''$;
end
$\hat{P_1^+}, \hat{P_2^+}, \hat{P_3^+} \leftarrow P_1', P_2', P_3'$;

Computing the dual of a polygon The dual of a polygon P is the set of locus of points $P^\perp$ such that

$$\mathbf{x} \in P^\perp \Leftrightarrow \exists \mathbf{y} \in P : \mathbf{y}^\mathsf{T}\mathbf{x} = 0 \tag{3.5}$$

where $\mathbf{x}$ and $\mathbf{y}$ are expressed in homogeneous coordinates in the image plane. On the Gaussian sphere, $P^\perp$ is represented by a continuous set of great circles. In order to compute $P^\perp$ we rely on the following property that can be proved for a generic triangle:

Property 3.1. *The dual of a triangle is the dual of its border.*

Proof. We recall that the dual of a point is a line (and vice-versa). For a point $\mathbf{A}$ in the projective space, all the vectors $\mathbf{l}_i$ satisfying $\mathbf{l}_i^\mathsf{T}\mathbf{A} = 0$ are the set of all the lines passing by $\mathbf{A}$. In the dual space, $\mathbf{A}$ is a line and $\mathbf{l}_i$ are all the points lying on such line.

If we now consider a line segment between two points $\mathbf{A}$ and $\mathbf{B}$, it can be easily shown that the dual of a line segment is a double wedge [19]. The duals of the two extremal points $\mathbf{A}$ and $\mathbf{B}$ are two lines $\mathbf{l}_\mathrm{A}$ and $\mathbf{l}_\mathrm{B}$ that intersect in one point $\mathbf{O}$, which is the dual of the line supporting the segment. As a generic point move along the line segment from $\mathbf{A}$ to $\mathbf{B}$, it spans on the dual space a set of lines passing by $\mathbf{O}$ and limited by the two lines $\mathbf{l}_\mathrm{A}$ and $\mathbf{l}_\mathrm{B}$. This defines a double wedge in the dual space.

Consider now a point $\mathbf{D}$ not lying on the segment $\overline{\mathbf{AB}}$ and the pencil of lines passing through it. In the dual space, this set of lines is a set of collinear points, lying on the line $\mathbf{l}_\mathrm{D}$ (the dual of $\mathbf{D}$). If we consider the subset of those lines of the pencil crossing the segment $\overline{\mathbf{AB}}$, in the dual space they are all the points of $\mathbf{l}_\mathrm{D}$ that are also contained in the double wedge of $\overline{\mathbf{AB}}$: indeed, each of those points

are the intersection of $\mathbf{l_D}$ with a line of the double wedge corresponding, in the primal plane, to the intersection point on $\overline{\mathbf{AB}}$.

Let's now consider the triangle $\mathbf{ABC}$ in Figure 3.22. If we consider a point $\mathbf{D}$ inside the triangle, by definition all the lines of the pencil passing through $\mathbf{D}$ cross at least one of the three segments forming the triangle. In the dual space that means that the line $\mathbf{l_D}$ is completely contained in the union of the three double wedges. Hence the dual of the triangle is completely contained in the dual of the border.

□

Since any polygon can be triangulated it is straightforward to extend this property to the polygon and prove that the dual of a polygon is the union of the duals of its triangles.

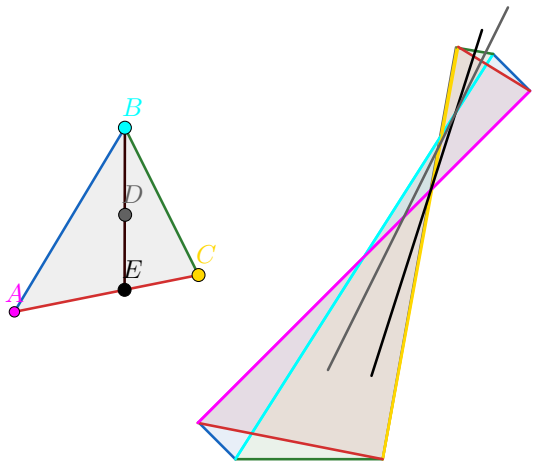
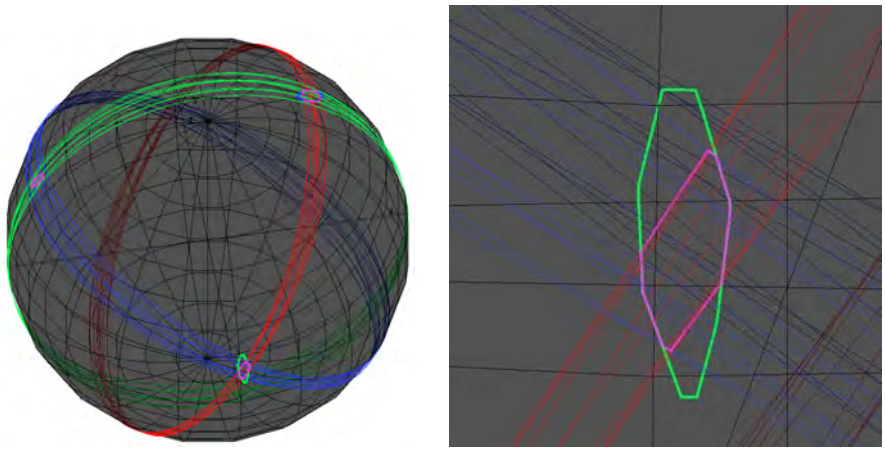


Figure 3.22: The duals of the points A, B, C, D, E are represented on the right with magenta, cyan yellow, black and gray lines respectively. The duals of the line segments $\overline{AB}, \overline{BC}, \overline{AC}$ are represented with blue, green, red double wedges respectively.



(a) **(b)**

Figure 3.23: The red, green and blue polygons are intersected with their orthogonal represented with pencils of great circles of the same color. The intersection, after one iteration, is displayed in magenta. Figure (b) is a zoom of figure (a)

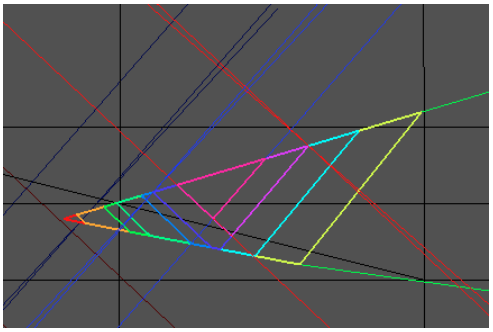


Figure 3.24: The obtained polygons after the different iterations of Algorithm 1. In green: the original polygon, in magenta the final polygon after 10 iterations. The blue and red lines are the boundaries of the dual of the two other polygons.

Part II

RGB-D

CHAPTER 4

RGB-D Localization

Contents

4.1 Introduction . . . 42
4.2 Range imaging sensors . . . 42
4.2.1 Definition of a range imaging sensor . . . 42
4.2.2 Mobile range imaging technologies . . . 43
4.2.3 Considered technologies . . . 47
4.3 Visual odometry evaluation . . . 47
4.3.1 Taxonomy . . . 48
4.3.2 Implementations on mobile devices . . . 50
4.3.3 Previous benchmarks . . . 51
4.3.4 Tested visual odometry algorithms . . . 52
4.3.5 Algorithms selection . . . 56
4.3.6 Mobile experiments . . . 60
4.3.7 Benchmark conclusions . . . 64
4.4 Planar approaches . . . 65
4.4.1 Overview and related works . . . 66
4.4.2 Our approach . . . 71
4.4.3 Planar approaches conclusion . . . 78
4.5 Conclusion . . . 80
4.A Additional planar LS SLAM results . . . 82

4.1 Introduction

In the context of indoor scene reconstruction, the objective is to correctly identify the walls, assumed as vertical planes, possibly while the user is moving around in the scene. This requires the ability to keep track of the movement of the user, and, more precisely, to localize or follow the movement of the device in the scene. Visual Odometry (VO) is the task of estimating the 3D pose (*i.e.*, its position and orientation) of the camera from visual data [131] and, more in general, from RGB-D data exploiting the 3D information provided by depth sensors [127]. It is a key component which enables to consider scenarios where the user can move freely in an environment and remains localized. Designing a VO algorithm is a challenging task: it should be fast on the target platform in order to follow the camera frame rate and to let CPU resources for the other component, as well as robust, since an error in the localization can lead to incorrect interpretation of the RGB-D data (*e.g.* incorrect position of objects, duplication of a wall *etc.*). While VO has been thoroughly addressed in the literature using desktop computers, the limited computational power and hardware of mobile devices set new challenges for adapting or designing new and more efficient algorithms able to process RGB-D data. It is then interesting to evaluate the performances of current algorithms on mobile settings.

In this chapter, we introduce in Section 4.2 various range imaging cameras integrated on mobile devices, explaining the limitation of each technology, with a focus on the sensors considered during this thesis. In Section 4.3 we give an overview on VO and we perform an evaluation of various RGB-D VO algorithms. The compared approaches are generic, *i.e.* there is no assumption on the scene. We then consider the particular case of indoor scenes made of dominant planes in Section 4.4, which gives details on the plane extraction problem from point cloud and describes how it is possible to take advantage of the detected planes to improve the localization and the reconstruction.

4.2 Range imaging sensors

In the past years, we have witnessed the development of consumer-grade depth sensors such as the Kinect for Xbox 360 (Kinect^{SL}). Although these sensors were initially meant for gaming and entertainment, they found a large interest in various communities, such as computer vision, robotics, and biomechanic communities, as they can provide 3D data at relatively low cost. In an escalating trend, manufacturers are now focusing their efforts on reducing the size of these sensors in order to offer mobile devices the possibility to better sense and understand the world around them. Embedding depth sensors on everyday mobile devices can foster a whole new range of consumer applications, from augmented reality to 3D reconstruction and scene understanding.

4.2.1 Definition of a range imaging sensor

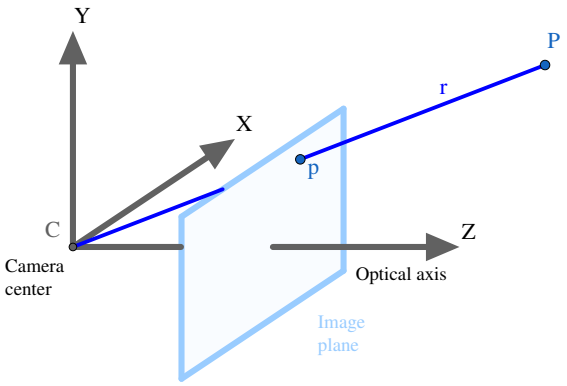


Figure 4.1: The camera pinhole model.

The classic camera pinhole model, illustrated in Figure 4.1, projects a 3D point P of the scene to an image point p on the image plane. The point p corresponds to the intersection of the ray r passing

through the camera center C and P. Thus, the 3D point P cannot be recovered from p, because any point lying on the ray r is mapped to the same pixel p.

Range imaging sensors are a subset of the existing depth sensors, special cameras that produces range images (also called depth map) whose pixel values are proportional to the distance between the points C and the 3D points P along the z axis, *i.e.* the camera optical axis. Contrary to the classic cameras, range imaging sensors provide dense 3D measurements, with the 3D point cloud organized in the form of a matrix (hence the name of organized point cloud): this allows a faster processing of the point cloud thanks to more efficient neighboring search algorithms, image integral computation, *etc*.

4.2.2 Mobile range imaging technologies

First attempts of mobile range imaging products came in 2013 with desktop sensors fixed on tablets as depicted in Figure 4.2. Later, the PrimeSense Capri, the first range imaging sensor designed for mobile devices was unveiled during the Google I/O 2013 with demonstrations on Nexus 10. It was followed by several sensors designed for mobile devices too, compared in Table 4.1.

Distance measurements are subject to noise, which depends on the technology adopted by the manufacturer. In the following paragraphs, we describe different real-time[1] range imaging camera technologies, compatible with an integration on a mobile device, along with their strengths and limitations. We refer the reader to [55] for further information on range imaging sensors.



Figure 4.2: Desktop range imaging sensors fixed on tablets. On the left: Lynx Laboratories tablet. On the right, DOT Product DPI-8X.

4.2.2.1 Stereo vision

Principle Stereo vision is a passive depth perception approach which relies on triangulation. When a scene is observed from two different viewpoints (producing two images), corresponding image points of the scene are imaged at different points in the two images. This (apparent) displacement of corresponding image points is called parallax. The search of corresponding points in the two images is called the correspondence problem and the displacement of corresponding points in the image is called the disparity. The disparity of objects closer to the camera are larger than more distant objects because, with the parallax effect, their displacement in the image is longer. Finding the correspondences between the two images is a complex problem we will not discuss here, we refer the readers to Szeliski's book [198] for more details. If the transformation between the two viewpoints is known, the 3D positions of the matched points can be computed via triangulation It is generally applied to binocular vision systems where the camera displacement is known and constant. Its application to monocular vision system is more complex as it requires to reliably estimate the pose of the camera.

[1] With a frame rate superior to 5 Hz
[3] Limited to 12k points

Technology	Manufacturer	Product name	Range	Acquis. rate	Resolution	Integrations
Stereo	Intel®	RealSense™ Snapshot	N/A	N/A	N/A	Dell Venue 8 7000 series (2015)
	Lips®	U-Focus	0.2–0.6m	30 fps	640×480	N/A
IR SL	Intel®	RealSense™ 3D Camera ZR300	0.55–2.8m	30 fps	480×360	Intel RSDK (2016)
	Intel®	RealSense™ 3D Camera R200	0.55–3.5m	60 fps	480×360	HP Spectre X2 (2015), Microsoft Surface Pro 4 (2016)
	Heptagon	Mora and Zora	0.2–1.5m	N/A	640×480	N/A
	Lips	LIPSedge M5	0.3–4m	60 fps	640×480	N/A
	Mantis Vision	MV4D	0.5–4m	5 fps	320×240[3]	Tango TDK (2014) and Mantis Vision Aquila 3D tablet (2014)
	Occipital	Structure	0.4–3.5m	30 fps	640×480	Pluggable on iOS devices (since 2014)
	PrimeSense	Capri	0.5–4m	5 fps	320×240	Tango Peanut Phone (2014)
	STMicroelectronics?		Unknown	Unknown	Unknown	Apple iPhone X (2017)
	Bellus3d	Face camera Pro	0.25–0.6m	Unknown	Unknown	Pluggable on Android devices (2018)
ToF	Infineon and PMD	Real3™	0.1–4m	45 fps	224×172	Lenovo Phab 2 Pro (2016) and Asus Zenfone AR (2017)
	Sony Depth-sensing Solutions	DepthSense® 541	0.1–5m	60 fps	40K points	A Sony Xperia prototype (2017)

Table 4.1: Comparison of various range imaging sensors designed for mobile devices.

Limitations and advantages Stereo vision is a passive technology, therefore it is dependent on good lighting conditions of the scene and the presence of texture to solve the correspondence problem. Because the two images come from different camera positions, some regions may be visible in the first image, while being occluded in the second one. Occlusions, repetitive textures, and specular reflections lead to mismatches and incorrect range measurements which can be partially filtered [198]. Also, transparent surfaces are obviously not sensed. As demonstrated in [55], the range measurement errors are quadratic with the distance to the objects. The resolution of the measurements can be improved on the software side with additional computational cost, using the highest image resolution available and sub-pixel disparity computation.

Some approaches, such as the plane sweeping and the block matching strategies can be extremely parallelized and are suitable for hardware implementations on FPGAs or ASICs [112], allowing real-time stereo depth sensing capabilities to be low power and independent of the host CPU. Finally, stereo vision has a low Bill of Material (BOM) cost and can be used outdoor, contrary to the other techniques we describe afterward.

Integration on mobile devices Monocular dense stereo on mobile devices has been proposed by [141, 182] with GPU implementations. They achieved real-time performances but required a high-end device (an iPhone 6 and a Tango TDK respectively). Since 2014, the Google Camera application can also compute range images on any Android mobile devices, but not in real-time. The depth information is used to simulate a Bokeh effect on the photo and is stored in the EXIF data of the image file. It can be viewed on the website `http://depthy.me`.

Smartphones with a stereo rig started in 2011–2012 with the HTC EVO 3D and the LG Optimus 3D series. Saez *et al.* [173] used one of these devices to implement a stereo vision algorithm which could perform 30000 range measurements at 9 fps on the host CPU. Similar mobile devices were

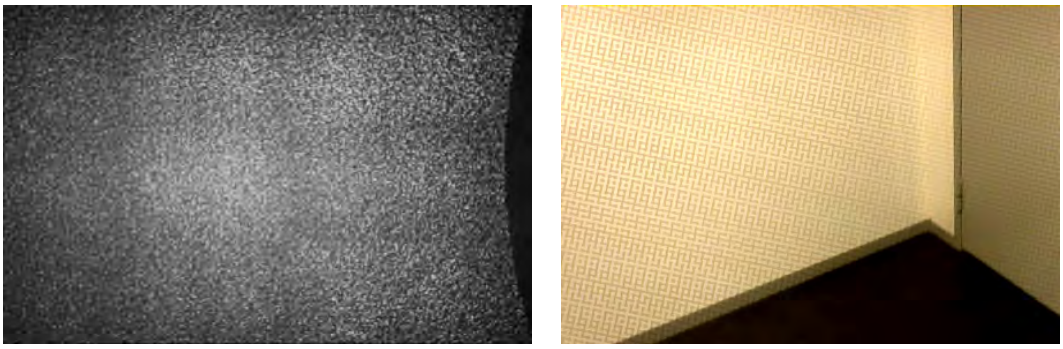
brought up to date in 2014 with the HTC One M8, until now with the iPhone 7 Plus and the Huawei P10. Unfortunately, they do not necessarily integrate dedicated hardware or SDK to perform stereo vision, thus their usage is restricted to image enhancement instead of real-time depth sensing. Since iOS 11, the API provides live depth map on dual camera devices. However, it is not possible to access the two camera live streams, contrary to the Android API which offers this possibility but does not provide depth map estimation.

A three cameras rig was proposed by Intel® with the RealSense™ Snapshot, integrated in the Dell Venue 8 7000 series tablet. The computed range images could only be used with specific applications for photo edition (refocusing, background removal, ...) and taking measurements in images.

When the camera sensors are very close each other, the technology is called *camera array*. The baseline is shorter, which is less optimal for long distance measurements. For example, the U-Focus from Lips is a 4×4 camera array with a range which cannot exceed 0.6 m. Other camera arrays were designed for mobile devices, involving LinX Imaging and Pelican Imaging before their acquisition by Apple and Tessera FotoNation respectively.

4.2.2.2 Structured light

Principle Structured light, also called active stereo, is a particular case of stereo vision: a known pattern considered as a reference image is projected into the scene. A camera distant from the projector observes the image of the projected pattern which is deformed by the shape of the scene. Solving the correspondence problem between the observed pattern and the known one, followed by the triangulation step, enables to perform the range measurements. We refer the reader to [14] for further explanations. The patterns generally considered are stripes and speckles (random dots), with near IR or visible light. For example, PrimeSense technology (behind the Kinect^{SL} among others) relies on IR speckles as depicted in Figure 4.3a.

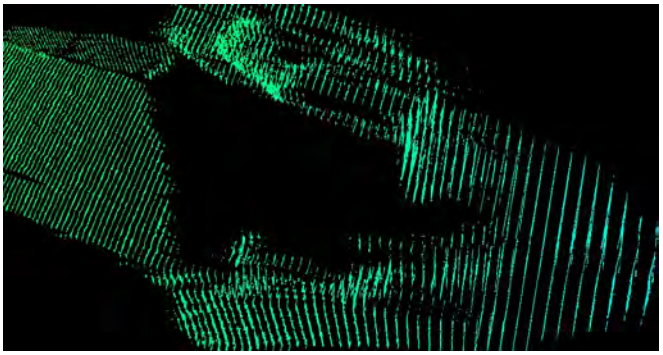


(a) SL pattern from the Structure Sensor. **(b)** SL pattern from the Tango TDK

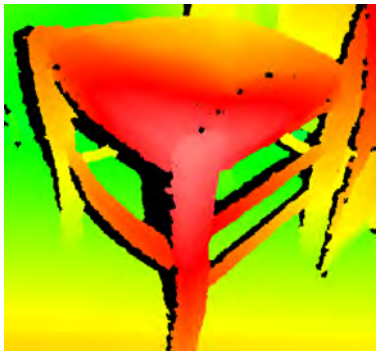
Figure 4.3: SL patterns

Limitations and advantages The main advantage of active methods such as SL is they do not require the presence of texture in the scene unless the albedo of the material is null for the pattern projector wavelength. In term of limitations, the SL approaches inherit some of the stereo vision: the depth resolution (illustrated in Figure 4.4a), the sensitivity to specular reflections, and the problem of the occlusions depicted in Figure 4.4b. Sarbolandi *et al.* [177] gives a complete list of the limitations of this technique, among them, the sensibility to ambient background light: patterns should be visible with high contrast to solve the correspondence problem, which means that it is preferable to have a projector with a strong light power (or a dark enough scene), and to avoid strong IR lights (*e.g.* light bulb, sun) when using a near IR pattern.

Integration on mobile devices SL with visible light has been attempted on mobile devices with a built-in light projector by Pribanić, Đonlić and Petković [153, 45]. They used two consumer-grade mobile devices: the Samsung Galaxy Beam and the Lenovo Yoga Tab 3 Pro which were not designed for this use, and demonstrated it was possible to compute decent range images using the host CPU.



(a) With the PrimeSense depth sensors, the disparity values are computed with 1/8 of pixel accuracy. Therefore only discrete value are obtained, which leads to a discretization of the depth data commonly called the quantization effect. Here the building appears to be cut into slices.



(b) Stereo vision approaches (active and passives) are sensitive to occlusions, leading to a shading effect, here on the left of the hand and the body.

Figure 4.4: Some limitations of the SL and stereo technologies

Primesense popularized consumer grade depth sensing using near IR SL with the Kinect^{SL}. Their technology was used in the Google Tango Peanut Phone and the Structure Sensor by Occipital [138]. Some sensors can support outdoor scenes, such as the Intel® RealSense™ ZR300 and R200 with the addition of a second IR camera. The Table 4.1 gives a complete list of SL range imaging sensors designed for mobile devices.

Range measurements noise Various noise models of the Kinect^{SL} have been proposed. Their purpose is to study the parameters having an impact on the measurements accuracy and to estimate the depth uncertainty for the algorithms relying on range measurements. Khoshelham *et al.* [94] proposed a noise model of the Kinect^{SL} depending on the sensor to objects distance only. He validated experimentally the random error measurement grows quadratically with the distance to the objects, up to $4\,\mathrm{cm}$ at the maximal range. His model was extended by Dryanovski *et al.* [49] to predict the depth uncertainty at object boundaries and to improve the robustness of his VO algorithm. Indeed, the range measurement errors are more important at object edges, areas which have a higher importance for some VO approaches, as we will see in the Section 4.3. Later, Nguyen proposed [128] another noise model that took into account the orientation of the surface w.r.t. the sensor. The derived measurement uncertainty has been integrated into the KinectFusion [127] pipeline to demonstrate it can help improve the registration the reconstruction.

4.2.2.3 Time of Flight

Principle ToF is another active technology, which senses distances by measuring the time the light has traveled from an emitter to an object and back to the camera. The most common approaches are the Pulsed and Continuous Wave (CW) Intensity Modulation described in [40]. An intensity-modulated light signal is sent by the emitter, generally in the near-IR spectrum. Since the light is an electromagnetic wave with a phase which varies periodically with the time, the travel time translates into a phase shift which is measured for all the pixels of the camera. The phase shift is assumed not to exceed one period of the phase.

Limitations and advantages ToF sensors do not require baseline, the emitter can be located next to the camera, allowing very compact designs. As described in [177], these sensors share some limitations with the SL approaches: incorrect measurements at objects boundaries, and sensitivity to background ambient light and to specular reflections. Finally, when the light is scattered by a surface, various reflected rays may indirectly reach the camera, leading to a multi-path effects (increasing the measured distance), which is considered as one of the major error sources of ToF cameras.

Integration on mobile devices Infineon in collaboration with PMDTechnologies developed the first ToF range imaging sensor which was integrated into consumer smartphones: the Lenovo Phab

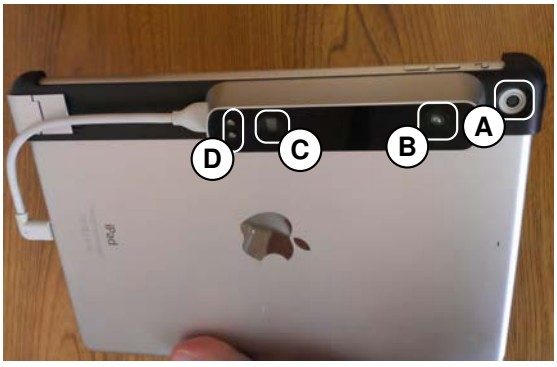
2 Pro and the Asus Zenfone AR. More recently, Sony SoftKinectic designed is own ToF sensor which was integrated in Sony Xperia prototype demonstrated at the MWC 2017.

4.2.3 Considered technologies

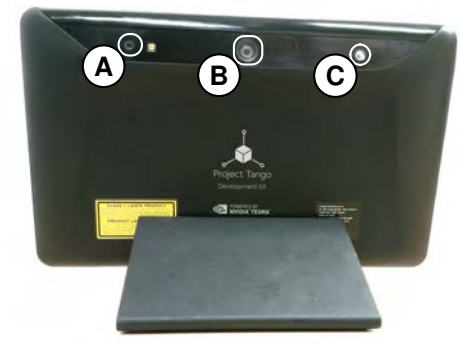
In the rest of the thesis, the range imaging sensors considered will be the Structure Sensor from Occipital and the Mantis Vision MV4D via the Project Tango Tablet Development Kit which were acquired during the realization of this Ph.D.

The Structure Sensor is a SL external range imaging sensor with its own battery, which can be fixed with a bracket on compatible iOS devices. The RGB images are provided by the rear camera of the considered device: therefore there is no exact synchronization between the depth sensor and the RGB camera, and the removal or the insertion of the sensor requires to run a calibration procedure in order to obtain range images aligned with the RGB images. A free and proprietary ObjectiveC SDK offers an RGB-D VO algorithm we evaluate in Section 4.3, as well as textured meshing. As described in Table 4.1, this sensor offers range images at VGA resolution and 30 fps.

The Project Tango Tablet Development Kit illustrated in Figure 4.5b, integrates a RGB-IR camera as depth sensor, with a IR pattern emitter, and a fisheye camera. Compared to the Structure Sensor, such design does not require the user to perform a calibration between the sensors. The downside is it prevents from obtaining depth maps and RGB images simultaneously, since the camera is shared for the two usages: during 1 s the camera provides 17 RGB images and 5 depth images. Also, a synchronization failure of the emitter with the camera can lead to missing depth data or the display of the pattern on the RGB image, as occurred in Figure 4.3b. Its 5 fps frame rate and the 12k points resolution are inferior to those of the Structure Sensor. A Java and C SDK offers a Visual Inertial Odometry (VIO) algorithm, global map optimization and textured mesh generation.



(a) Structure Sensor by Occipital attached with a bracket to our iPad Air. iPad RGB camera (A), Structure IR camera (B), IR projector (C), IR light-emitting diode (D).



(b) Tango TDK image sensors – RGB-IR camera (A), Fisheye camera (B), IR pattern emitter (C).

Figure 4.5: The two tablets and depth sensors considered along this thesis.

4.3 Visual odometry evaluation

The study of RGB-D VO algorithms occurred while we were evaluating the recently released Structure Sensor by Occipital [138]. At that moment, the focus of the company was more on objects reconstruction than scenes reconstruction, which translated in poor localization performances. This was penalizing, as it implied the generated floor plans would have incorrect measurements. Several odometry algorithms designed for the Kinect^{SL} had been published and released open-source. We wanted to perform a benchmark of these approaches, to select one with high performances in term of accuracy and speed (for porting on a mobile device). The expected benefit was twofold: reduce the drift issues experienced with the closed source VO algorithm provided by the Structure SDK, and use an editable localization algorithm.

In this section, we will study five selected real-time VO algorithms among the most accurate in the literature and for which an implementation was available. We will present a first evaluation and comparison of their performances in terms of accuracy and resource consumption (memory and CPU usage) on desktop settings using a standard dataset for the evaluation of RGB-D VO algorithms [196]. Then, we will detail the results of a second similar evaluation on mobile settings, for which we considered the Structure Sensor. This benchmark will not include the Tango TDK, which was not in our possession at the time of the tests, and it uses a different category of VO algorithm taking advantage of inertial data but not using a depth sensor, as detailed in [195].

4.3.1 Taxonomy

Simultaneous Localization And Mapping (SLAM) is a process by which a system can build a map of an environment, and at the same time, can use this map to deduce its location. A SLAM system is divided into two part: a front-end and a back-end. The front-end is dependent on the considered sensors. Assuming they include a camera, its principal component is the VO algorithm, which localizes the camera at the sensor frame rate with a limited accuracy (*i.e.* the estimated trajectory is only locally consistent). The back-end is input agnostic, it performs a global optimization of the map and the trajectory by leveraging loop closures. It should be noted that sometimes, SLAM named algorithms may only refer to the back-end. In the following, we will focus on the VO algorithm.

Visual Odometry is an incremental process which estimates the 3D pose of the camera from visual data. The latest visual frame is registered against previous data to compute a rigid body transformation between the current and the previous pose. Similarly to wheel odometry, the accuracy of the current pose depends on the reliability of the previous pose. Therefore VO is prone to accumulation of errors and the resultant trajectory can only be considered locally consistent.

A VO algorithm can be qualified small or large-baseline depending on its ability to handle large changes of camera viewpoint between consecutive frames. Large camera displacement can occur during fast camera movements or with a low camera frame rate. In this paper, we will focus on small-baseline RGB-D VO using Primesense-based depth sensors with the assumption of static environments.

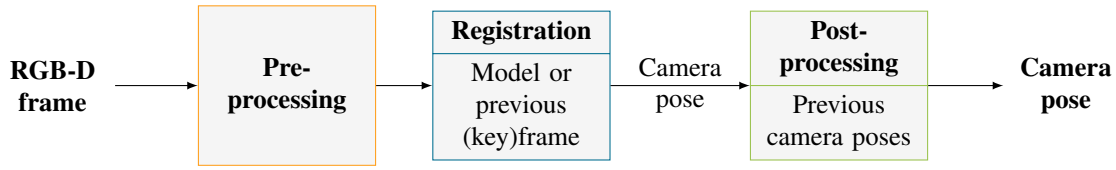


Figure 4.6: The different components of a VO pipeline.

Generally, a VO algorithm can be seen as a registration procedure with a preprocessing and a post-processing step as depicted in Figure 4.6. The depth maps provided by depth sensors are noisy and can have missing values due to occlusions. Some VO implementation [127, 161] add a filtering preprocessing step in order to enhance the depth maps and improve the registration step. To this end, bilateral filters [203] can be applied to the raw depth data to reduce the noise and the missing data, yet preserving the discontinuities.

The registration process takes as input the latest RGB-D frame and a previous frame (or a model) to compute the current pose of the camera. There are different strategies for aligning two frames. In the *frame-to-frame* matching strategy [93, 194], the current RGB-D frame is aligned with the previous one. This strategy quickly leads to a large drift of the estimated trajectory as the pose errors are cumulated. To mitigate this effect, the *frame-to-keyframe* strategy samples the sequence of RGB-D frames into keyframes, usually having a larger spatial displacement among them [81, 135, 136]. The current frame is then aligned w.r.t. to the previous keyframe, until a new keyframe is selected. The selection of the keyframe is important in order to have an homogeneous sampling of the scene, and it often relies on an heuristic evaluating the image quality (*e.g.* no motion blur) or the redundancy of the visual information. For example, in [81] a threshold on the number of matched visual features is used as an indicator of the overlapping part of the scene between the frames and of the movement of the camera. Other methods [138] uses the IMU sensor and a threshold on the estimated rotations and translations to select the keyframe. Another strategy, called *frame-to-model*, consists in building a

model of the explored scene and using this model to align the new frames. The model can be a sparse 3D point cloud, as *e.g.* for CCNY [49], or a voxel grid of Truncated Signed Distance Function (TSDF) (Truncated Signed Distance Function) for KinectFusion [127]. For the latter model, a synthetic view of the surface is generated, usually in the form of a depth map at a predicted camera pose to perform the registration. This strategy significantly reduces the small-scale drift and it is more accurate than the *frame-to-keyframe* strategy [127]. Moreover, *frame-to-model* strategy allows to recover after tracking failures and relocalize the device w.r.t. the model [22]. On the other hand, they still suffer from large-scale drift and may require a heavy memory usage, which can be reduced by using only a subset of the model. For example, CCNY [49] subsamples the 3D point cloud and Kintinuous [211] only loads the part of the scene taken into consideration. Aligning a frame to a model requires more computation than aligning a frame with another one. To speed up the registration process, it is necessary to take into consideration only the part of the model that has an overlap with the current frame. A common approach for the voxel grid of TSDF is to generate a depth map from the model at a predicted pose and to compare the current frame with this depth map [127, 141]. We refer the reader to [122] for more detailed information on the different registration methods.

On some *(key)frame-to-frame* VO approaches [223], a local optimization post-processing step is added to refine the latest camera pose and reduce the trajectory drift. It cannot be applied on *frame-to-model* strategies, unless the model can be updated when the previous camera poses are refined by the optimization process.

During the registration process, only two frames are taken into account in order to efficiently compute the camera pose at the current time with a closed form expression. The idea is that the current pose could have been computed with earlier frames than the previous (key)frame. The local optimization process takes several (key)frames as input of an optimization problem, often a windowed bundle adjustment problem, and returns refined camera poses [223]. We refer the reader to [178] for a more detailed survey on camera pose optimization for monocular VO.

A taxonomy for VO registration approaches has been proposed in [53], classifying the approaches into three main categories: *image-based*, *depth-based* and *hybrid-based* (see Figure 4.7). In the following, we briefly summarize each category w.r.t. the methods evaluated in this paper. We also recommend Yousif *et al.* review [220] on RGB-D VO algorithms which includes monocular VO and Visual Simultaneous Localization And Mapping (vSLAM) algorithms.

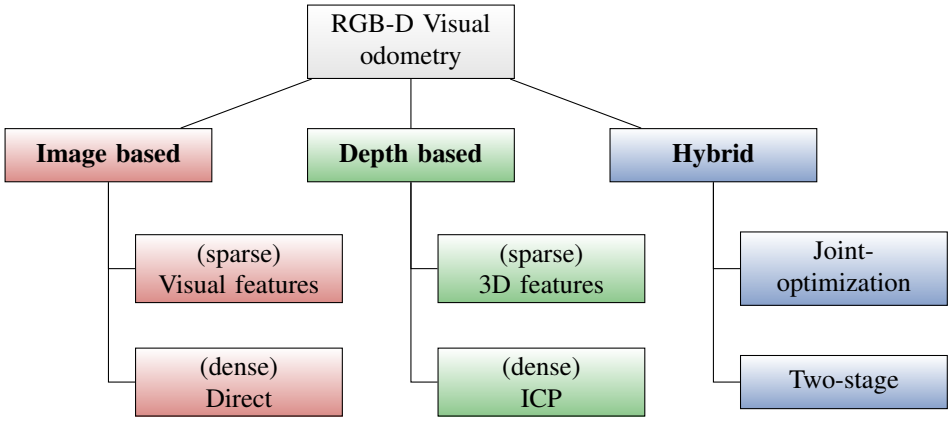


Figure 4.7: Summary of the three classes taxonomy of registration approaches proposed for RGB-D VO.

The *image-based* methods rely on the information of the RGB image [81, 93, 162] and it can be further divided into *feature-based* methods and *direct* methods. The formers are sparse methods as they use local image features to register the current frame w.r.t. a previous (key)frame. On desktop computers, SIFT [115] and SURF [16] features are commonly used for their high robustness [52]. On the other hand, their computational cost makes them unsuitable for mobile devices Hudelist *et al.* [83], and other computationally cheap features such as BRISK [108], BRIEF [28] and ORB [169] must be used. These methods perform well in highly textured scenes while they tend to fail in poor light conditions and under fast motion of the camera, as the features are not robust to motion blur. Moreover, the features are generally located at objects boundaries where the depth information provided by sensors based on Structured Light technology [138] is the least reliable, thus affecting the registration

accuracy. The *direct* methods [93] are instead dense method, as the registration uses all the pixels of the images. Under the assumption that the luminosity of the pixels is invariant to small viewpoint changes, they estimate the camera motion that maximizes a photo-consistency criterion between the two considered RGB-D frames. These methods work even in poor light conditions and low textured scenes, and they can handle object occlusions. On the other hand, the viewpoint displacement between the considered frames must be small, thus limiting the range of application to smooth and relatively slow movements.

The *depth-based* algorithms rely mostly on the information of the depth images [162, 135, 194]. The sparse *3D feature-based* methods rely on the extraction of salient features on the 3D point clouds. The rigid body transform can be computed by matching the descriptors associated to the features extracted in two frames. As for the feature-image-based algorithms, the majority of these features are located at objects boundaries and areas with high curvatures. Again, due to the limitations of the Structured Light technology, the depth values have low accuracy or can be missing in these areas, leading to bad repeatability of the features and poor registration accuracy. The Iterative Closest Point (ICP) methods refer to a class of registration algorithms which try to iteratively minimize the distance between two point clouds without knowing the point correspondences [20, 33]. The alignment error is computed with a given error metric such as point-to-point or point-to-plane distance, and the process is repeated until this error converges or the maximal number of iterations is reached. Each iteration improves the point clouds alignment, which in return enables the heuristic association function to output more correct matches, and so on. Weighting strategies [170] are used for robust registration and filtering outliers due to sensor noise or the non overlapping parts of the 3D point clouds. Similarly to the direct-image-based methods, Iterative Closest Point (ICP) converges well under the assumption of small viewpoint changes, as it avoids local minima and converges to the desired solution. Coarse-to-fine approaches have also been proposed to improve the convergence [127, 160]. For an exhaustive review of ICP algorithms, we refer the reader to [149]. *Depth-based* algorithms can work well in poor light conditions as they rely on the 3D data, but on the other hand, they might fail with scenes having low structure (*e.g.* only few planar surfaces).

Finally, *hybrid* algorithms try to combine the best of the two worlds in order to handle scenes having either low structure or little texture [136, 159]. They can be divided into *two-stage* methods and *joint-optimization* methods. The two-stage methods use one approach (usually a sparse method) to compute an initial guess of the registration and use a second approach (usually a dense method) to refine the transformation or just compute it in case of failure of the first approach [49]. The joint-optimization strategy consists in designing an optimization problem which combines equations from depth-based and image-based approaches [159, 211].

4.3.2 Implementations on mobile devices

Developing real-time VO algorithms is more challenging on mobile devices due to their limited memory and processing power. Fine optimizations can be performed using SIMD instructions of the embedded CPU and OpenGL ES shaders can be used for processing parallelizable tasks on the GPU. However, this highly increases the complexity of the implementation and requires low-level programming skills. On modern mobile devices, one can also take advantage of the Inertial Motion Unit (IMU), and eventually integrate the estimated rotation as a prior knowledge into the registration algorithm.

Regarding monocular VO algorithms, Schöps *et al.* [180] achieved a 30 FPS tracking performance with a partial porting of the semi-dense LSD-Slam algorithm on a Sony Xperia Z1 phone. No code was publicly released though. Commercial solutions also emerged in the past years, proposed by 13th Lab, Metaio, and RealityCap, before their recent acquisition by Occulus VR, Apple and Intel ® respectively. The Google Project Tango [62] also proposes a proprietary monocular Visual and Inertial Odometry (VIO) algorithm. It is designed for dedicated hardware using, in particular, a fisheye camera such as the Tango Yellowstone tablet and the Intel RealSense® smartphone™.

Lately, with the recent development of depth sensors for mobiles such as the Structure Sensor [138] and Mantis Vision MV4D [119], new proprietary RGB-D VO algorithms for mobile devices have been developed and they are available through their relevant SDKs. Presumably, these advances will lead to more interests in the academic research on mobile RGB-D VO, even if developing algorithms that fully exploit the low-level hardware capabilities of the device is challenging. For example, Brunetto *et al.* proposed a RGB-D vSLAM algorithm based on SlamDunk which can run on a Samsung Galaxy

Tab Pro 10.1 tablet [25], but due to the lack of low level optimizations, it could only reach 10 FPS.

Visual Odometry is, however, an important component to enable computer vision applications on mobile devices. Klingensmith [97] proposed a real-time mapping solution for indoor scenes that takes advantage of the depth sensor and the VIO algorithm by Google Tango. Instead of allocating a fixed grid of 3D voxels as in the traditional approaches, he creates on-demand chunks of voxels according to the observations of the scene, which is appropriate for indoor environments as they contain a lot of free space. Schöps [182] addresses the outdoor mapping problem using the same hardware, which prevents the use of the depth sensor. With the help of the provided VIO algorithm, he computes and filters depth maps from the fisheye camera and fuse them with a TSDF approach too in order to reconstruct the scene. Live 3D reconstruction is a very challenging problem which requires performing the VO and the mapping in real time. Prisacariu *et al*. [156, 155] jointly estimate the pose and the visual hull of the model with a probabilistic framework. The system is robust to motion blur, lack of texture and can run at 20 FPS on an iPhone 5, but the resulting model is coarse and cannot contain concavities. Tanskanen *et al*. [200] propose a monocular visual features tracking approach combined with a multi-resolution stereo depth map estimation. The tracking runs at 15-30 FPS on a Samsung Galaxy S3 but the generated dense 3D point cloud is only refreshed at 0.3-0.5 FPS while being GPU optimized. Kolev *et al*. [99] improve the accuracy of the reconstructed model with a surfel approach combined with weighted depth maps, but at the cost of a lower frame rate. Ondrúška *et al*. [141] propose a faster solution (25 FPS on an Apple iPhone 6) which can generate medium accuracy 3D reconstructed models. They use a direct method for the tracking, compute the depth maps by dense stereo matching and perform the mapping with a TSDF approach.

4.3.3 Previous benchmarks

Assessing and comparing the quality and the accuracy of VO algorithms is an important task. This work relies on previous benchmarks that have been published in the last years, mostly in the robotics community. Sturm *et al*. [196] introduced and publicly released the TUM dataset, a collection of different RGB-D image sequences meant to benchmark SLAM and VO algorithms. Even if in the paper no algorithms evaluation is carried out, it has become a seminal work as the dataset has become a sort of standard for benchmarking new algorithms in the spirit of other computer vision datasets, such as *e.g*. the KITTI dataset [60].

Morell-Gimenez *et al*. [122] performed a comparison of registration methods on scenes mapping and object reconstruction scenarios. For the scenes mapping scenario, which is our topic of interest, they evaluated five different algorithms: DVO [93], KinFu (an implementation of KinectFusion [127]), an ICP approach, an imaged-based visual feature approach using a combination of FAST [167] keypoints and BRIEF [28] descriptors, and an hybrid two-stage approach combining the two last ones where the refinement step is provided by the ICP algorithm. The last three approaches were implemented by Morell-Gimnenez *et al*. using the Point Cloud Library [171]. The results show that DVO and Kinfu are the most accurate algorithms on the "fr1" scenes of the TUM dataset. The paper does not report any information about the computational time and the memory consumption as the main objective of the work was to assess the quality and the accuracy of each method.

Handa created the ICL-NUIM dataset [72] composed of synthetic images of indoor scenes generated with POVRay. Although the main focus of the dataset is to provide a method to benchmark the surface reconstruction accuracy, it has been used to evaluate different VO algorithms, thanks to the ground truth provided by the synthetic data. The following algorithms are compared on a desktop environment: DVO [93], Fovis [81], RGB-D [193], ICP KinectFusion flavour [127] and Kintinuous [211]. The evaluation on all scenes from ICL-NUIM with the ATE metric showed a clear advantage to KinectFusion ICP registration while Fovis gives the less accurate results.

More recently, Fang and Zhang [53] compared different open-source VO implementations: Libviso2 [61], Fovis [81], DVO [93], FastICP, Rangeflow [85], 3D-NDT [3], CCNY [49], and DEMO [224]. The evaluation is performed on two scenes of the TUM dataset and on a challenging dataset created by the authors with illumination changes, fast motion and long corridors. The metrics taken into consideration are the accuracy of the estimated camera motion and the performances of the algorithms (runtime and CPU usage). The authors [53] provide an analysis of the success and failure cases of the different algorithms w.r.t. the environment. In particular, the study shows that there is no algorithm performing well in all environments and some guidelines to choose a VO algorithm depending

on the environment are proposed. For example, when the scene is well illuminated, image-based and hybrid methods are recommended, whereas depth-based methods are only really interesting in low light environments.

The evaluation we are proposing in this paper is similar in spirit to the mentioned works, but our comparison is focused on the evaluation of algorithms for the mobile experiment, in which computational cost and memory consumption are strong constraints. To the best of our knowledge, this is the first attempt at benchmarking state-of-the-art algorithms on mobile devices equipped with a depth sensor. Our benchmark is similar in spirit to [53], but aimed at testing VO algorithm on mobile devices. For this reason, we tested some algorithms that were not considered in [53] and, due to our needs, we considered both the CPU and memory usage of the algorithms (whereas [53] only assesses the CPU usage).

4.3.4 Tested visual odometry algorithms

For our evaluation, we selected the algorithms to test based on two main criteria. Firstly, we only considered the methods that performed better in other benchmark studies (see Section 4.3.3). Secondly, the most important criterion was the availability of the code (or a SDK in the case of [138]), so that it could be ported and tested on a mobile device. According to these criteria we selected DVO [93], Fovis [81], MRSMAP [194], the 3 algorithms of the OpenCV RGB-D module [161] , and the VO algorithms that come with the Occipital sensor [138]. For brevity purpose, we denote OCV (ICP, RGB-D, RgbdICP) the three OpenCV algorithms we took into consideration. As pointed out in Section 4.3.3, only DVO and Fovis were considered for the benchmark in [53]. Table 4.2 provides a classification of the considered methods according to the taxonomy described in Section 4.3, and Table 4.3 collects more technical details about the code available for each method.

In the remaining part of this section, we briefly review the algorithms considered for our analysis. For each algorithm, we present a block diagram of the main pipeline. In the diagrams, blocks that are vertically aligned in the pipeline are blocks that can potentially run in parallel.

Algorithm	Method class	Registration	Matching Strategy	Local optimization
Fovis [81]	Image-based	Feature-based	frame-to-keyframe	No
OCV RGB-D [162]	Image-based	Direct	frame-to-frame	No
DVO [93]	Image-based	Direct	frame-to-frame	No
OCV ICP [160]	Depth-based	ICP	frame-to-frame	No
MRSMAP [194]	Depth-based	Feature-based	frame-to-frame	No
STTracker depth [135]	Depth-based	ICP	frame-to-keyframe	Unknown
STTracker color [136]	Hybrid	Unknown	frame-to-keyframe	Unknown
OCV RgbdICP [159]	Hybrid	Joint-optimization strategy	frame-to-frame	No

Table 4.2: Overview of the different approaches proposed in the evaluated VO algorithms.

4.3.4.1 Fovis

Fovis [81] is a fast visual odometry library developed for micro aerial vehicles (MAV). The visual odometry (front-end) represented by the Figure 4.8 is performed on the MAV and the global consistency of the trajectory (back-end) is enforced off-board. The registration is feature-based with a frame-to-keyframe matching strategy, employing FAST keypoints computed on multiple scales and on subdivisions of the images to ensure a uniform repartition of the keypoints over the image. Each feature is assigned to a descriptor containing the pixel values of the 9×9 patch centred in the keypoint. The descriptors are matched across frames using an $L1$ distance. The matches are then validated using the associated 3D points: for each frame, the distances among the associated 3D points are calculated and compared with those of the other frame. This allows retaining the inlier features used to estimate the rigid body motion with Horn *et al*. method [79]. Several refinements are then applied to improve the robustness of the computed camera pose.

Algorithm	Release year	License	ROS binding	SW dependencies	HW dependencies
Fovis [81]	2011	GPLv3	Yes	Eigen	(x86 SSE2)
OCV RGB-D [162]	2012	MIT	No	(Eigen)	No
DVO [93]	2013	GPLv3	Yes	Eigen, OpenCV, (PCL)	x86 SSE2
OCV ICP [160]	2012	MIT	No	(Eigen)	No
MRSMAP [194]	2012	BSD	Yes	GSL, TBB, OpenCV, Boost, PCL	No
STTracker depth [135]	2014	Closed	No	No	iPhone, iPad
STTracker color [136]	2014	Closed	No	No	iPhone, iPad
OCV RgbdICP [159]	2012	MIT	No	(Eigen)	No

Table 4.3: Technical overview of the evaluated VO algorithms. Dependencies in brackets are optional.

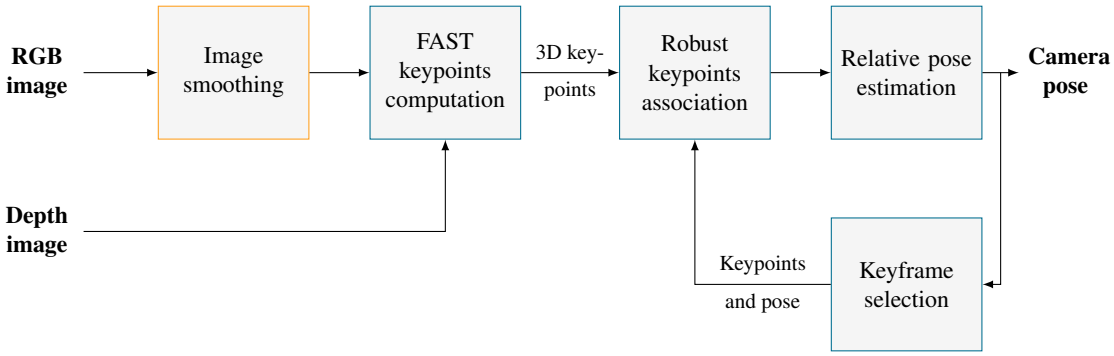


Figure 4.8: The pipeline of the Fovis algorithm. The preprocessing and registration steps are displayed in orange and blue respectively.

4.3.4.2 OpenCV RGB-D module

Maria Dimashova developed the OpenCV RGB-D module which is available in the `opencv_contrib` repository [161]. It offers a visual odometry algorithm which comes into three flavours : ICP, RGB-D and RgbdICP.

OCV RGB-D Figure 4.9 illustrates the RGB-D flavour [162], which is based on a direct image-based approach inspired by Steinbrucker *et al*. works [193] with a frame-to-frame matching strategy. Two hypotheses are made. First, the light intensity of a 3D point is considered to be constant among successive frames. Then, the angular and translational speeds are supposed to be constant between two frames. The algorithm finds the transformation relating two frames by minimizing the difference in intensity between the warped current RGB-D frame and the previous one. The first hypothesis enables to define the objective function as the sum of the square pixel intensities between the back-projected frame and the previous one. Thanks to the second hypothesis, it is then possible to reduce the minimization problem to a linear least square problem. Finally, to ensure better robustness with large motion change, the authors apply a coarse to fine approach by working on an image pyramid.

OCV ICP The ICP flavour [160], shown in Figure 4.10, is inspired by the point cloud registration algorithm of KinectFusion [127]. KinectFusion ICP variant is based on a projection based heuristic association function with a point-to-plane error metric. Assuming a small rotation between the two frames, the minimization of the point-to-plane error is reduced to a linear least square problem. A coarse-to-fine scheme is used to speed up the point cloud registration. It requires the computation of image pyramids for the depth frames and the normal maps. A notable difference with KinectFusion point cloud registration is that OpenCV is frame-to-frame whereas the other is frame-to-model.

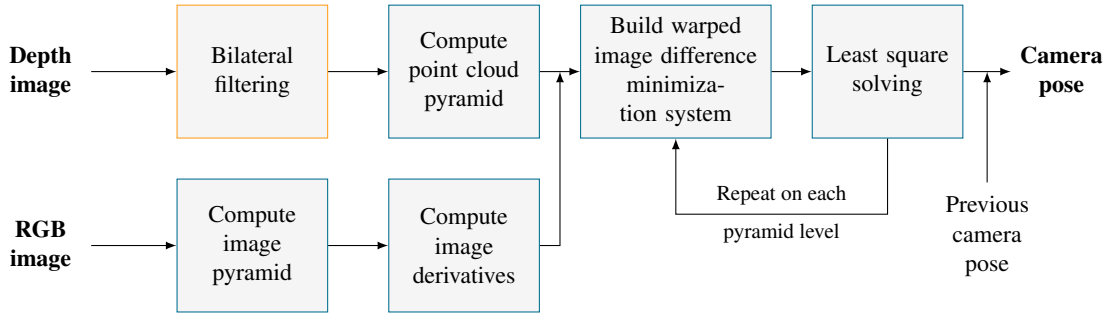


Figure 4.9: The pipeline of the OCV RGB-D algorithm.

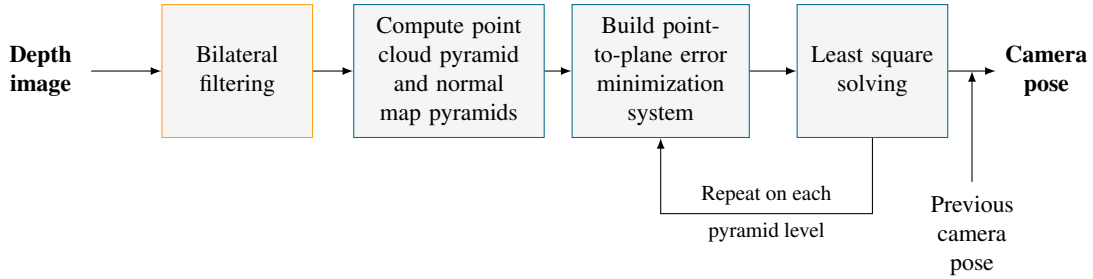


Figure 4.10: The pipeline of the OCV ICP algorithm.

OCV RgbdICP We have seen previously OCV RGB-D and OCV ICP were reduced to linear least square problems. As illustrated in Figure 4.11, the joint-optimization hybrid approach of OCV RgbdICP takes into consideration the concatenation of the equations of the two problems and solves it. It is the same scheme Whelan proposed with his RGB-D and ICP Integration [211].

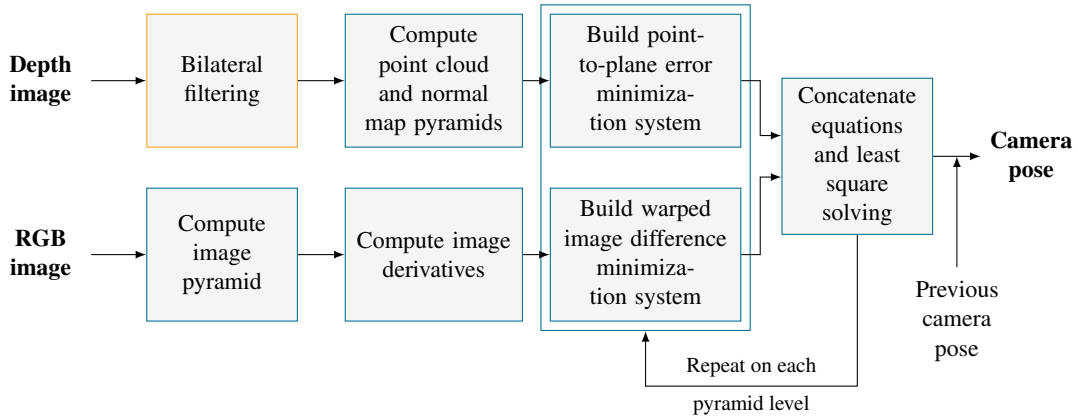


Figure 4.11: The pipeline of the OCV RgbdICP algorithm.

4.3.4.3 Dense Visual Odometry

Dense Visual Odometry (DVO) [93] depicted in Figure 4.12 is a direct image-based method with a frame-to-frame matching strategy. As in Steinbrucker *et al*. works [193] described earlier, a residual is defined with the difference of pixel intensities between the registered RGB-D frames. The minimization is performed with a coarse-to-fine approach in a probabilistic way, defining a likelihood of the transformation given the residual, and with the use of a sensor model and a motion model. The sensor model takes the form of a weighting function giving more or less importance to a given residual which partially originates from sensor noise. The motion model expresses the probability of a transformation. It can depend on IMU data if available. The proposed motion model assumes a constant velocity

in order to avoid jitter in the estimated trajectory.

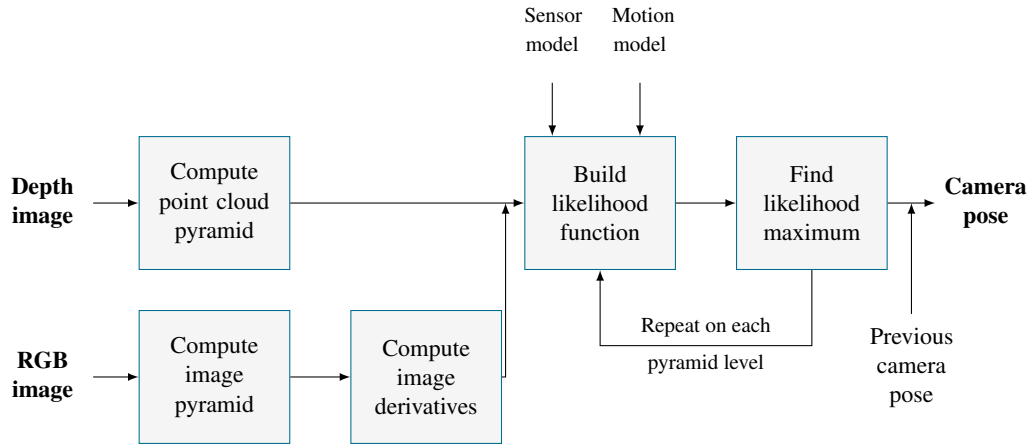


Figure 4.12: The pipeline of the DVO algorithm.

4.3.4.4 MRSMAP VO

Stückler *et al*. [194] proposed a 3D feature-based approach with a frame-to-frame matching strategy in which each frame is viewed as an octree of surfels. The originality of the approach is that multiple levels of resolution can be used simultaneously since each parent node of the octree encodes the information of their children node. The uncertainty of 3D points w.r.t. the camera is modelled by using smaller surfels for points closer to the camera. For optimization purpose, the coloured 3D points are not stored in the nodes: the local geometry and the colour distribution of the 3D points are instead encoded by a 6D multivariate normal distribution of 3D points coordinates and the three components of the colour in the L$\alpha\beta$ space. Each surfel is associated to a shape-texture descriptor which encodes the difference of colour and normal orientations between the adjacent surfels in the form of three bins histograms. The registration of a RGB-D frame illustrated in Figure 4.13 is performed at the level of their octree representation. The surfels of the two octrees are first associated with a coarse-to-fine approach using the shape-texture descriptor of the surfels. Then a likelihood based on the difference of the local geometry encoded by the associated surfels is maximized in order to compute the transformation between the two frames.

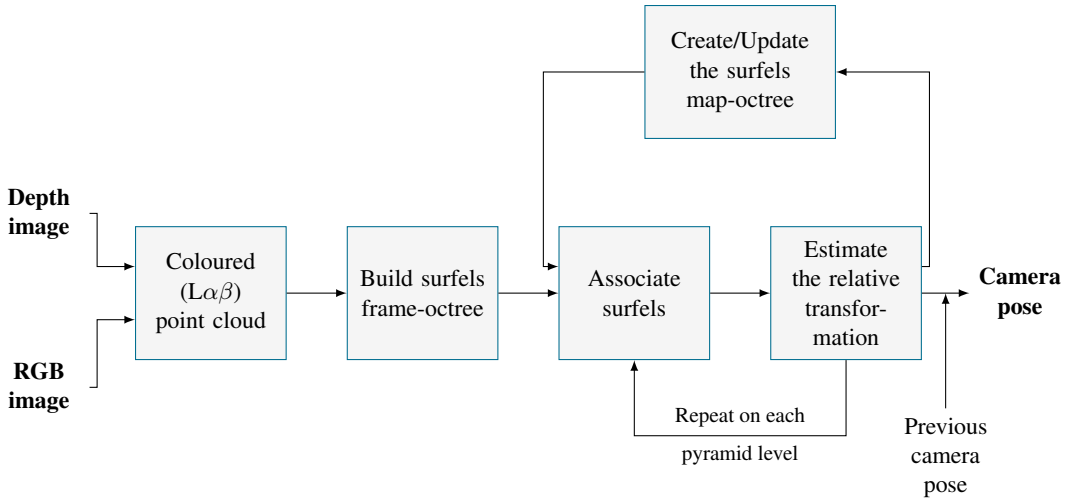


Figure 4.13: The pipeline of the MRSMap algorithm. The nodes association and transformation estimation steps are parallelized for each node and association respectively.

4.3.4.5 Occipital STTracker

Structure is a depth sensor manufactured by Occipital using Primesense's technology and it uses structured light to estimate the depth. The sensor does not support any RGB camera and it has to take advantage of the mobile device rear camera to retrieve the RGB frames. Occipital provides an iOS SDK with a VO algorithm in two flavours: depth-based [135] and hybrid [136].

4.3.5 Algorithms selection

In order to limit the number of algorithms evaluated on the mobile devices, we performed a selection based on three criteria: the accuracy, the runtime and the memory footprint. As mentioned in Section 4.3.3, most of the algorithms that we are considering were not used in previous benchmarks [53, 122]. For these reasons, we needed an assessment of their performances in terms of accuracy and resources consumption. Since all the algorithms mentioned earlier were designed for embedded or desktop computers, we chose the latter platform which also enabled us to easily perform memory monitoring. On the other hand, accuracy evaluations are not dependent on the computing platform.

4.3.5.1 Description of the dataset

As we mentioned earlier, the RGB-D TUM dataset for vSLAM evaluation was interesting for the evaluation because it offers various indoor acquisitions scenarios with ground truth trajectories. It is divided into three sets of sequences: "fr1", "fr2" and "fr3".

The "fr1" sequences provide various scenes recorded in an office environment. They include two simple scenes for debugging purpose: "xyz" and "rpy" with respectively translation only and rotation only sensor movements, and two very challenging scenes : "floor" which as the name suggests has low structure, and "360" with a high rotational motion and, thus motion blur.

The "fr2" sequences were recorded in a large industrial hall. Compared to the "fr1" sequences they are generally longer and have a slower camera motion. It also contains two debugging series and a "desk" scene. Three scenes are very challenging: "360 hemisphere", "large no loop" and "large with loop", due to the low texture and the distant 3D points.

Finally, the "fr3" sequences feature a scene with a desk and various evaluation series to evaluate the performances of algorithms on scenes with structure and/or texture.

At the time of writing, the `STTracker` class which implements the VO algorithm is designed to be used with the RGB-D frames of the Structure Sensor only. The evaluation of this algorithm on the TUM RGB-D vSLAM dataset was very difficult and required us to write an intermediate software layer which supplied the Structure SDK with the required data.

4.3.5.2 Description of the metrics

Parameters All VO algorithms have parameters which must be tuned in order to give the best results. To simplify the experiments, we took the parameters recommended by the author's algorithms in their respective articles. For the Structure STTracker, we took the parameters `STTrackerQualityAccurate` and `STTrackerDepthAndColorBased`.

Accuracy evaluation There are two well know metrics that can be used to estimate the accuracy of the estimated camera poses over time, the Absolute Translational Error (ATE) and the translational Relative Pose Error (RPE) [196]. They both assume that the ground truth and the estimated trajectory are aligned, time-synchronized and equally sampled. At a given time step ATE computes the Euclidean distance between the estimated camera position and its ground truth. The ATE is then defined as the mean squared error (RMSE) of these distances all along the trajectory. This metric is more suitable for vSLAM evaluation because it assesses the global consistency of the estimated trajectory relatively to the ground truth.

The RPE is instead used to measure the local accuracy of the estimated trajectory over a fixed time interval Δ. Considering a sequence of estimated camera poses (roto-translations) $\mathtt{P}_i \in SE(3), i = 1, \ldots, n$ and their corresponding ground truth $\mathtt{Q}_i \in SE(3), i = 1, \ldots, n$, the relative pose error $\mathtt{E}_i$ at time i is defined as

$$\mathtt{E}_i = \left(\mathtt{Q}_i^{-1}\,\mathtt{Q}_{i+\Delta}\right)\left(\mathtt{P}_i^{-1}\,\mathtt{P}_{i+\Delta}\right)^{-1}$$

The overall RPE of the sequence is then defined as the RMSE of the translation of each $\mathtt{E}_i$. The RPE better represents the drift of the trajectory over time, which is useful for the evaluation of visual odometry systems.

The accuracy comparison of the algorithms was performed during the desktop experiment. We used the RPE metric with a time interval $\Delta = 1\,\mathrm{s}$. For each experiment, we computed and plotted the RMSE, the mean and the standard deviation of the RPE values. We also plotted the graphs of the RPE over the time for visual inspection purpose, in order to highlight the experiments with high and narrow error peaks which would be masked by the RMSE measure.

As shown in Section 4.3, many parameters influence VO algorithms performances. For a proper comparison, we should compare individually, for each VO algorithm, its registration performance, using the same rigid body transformation estimation function, the same pre-processing and post-processing steps. This would be unpractical, for this reason, we only compared the full pipeline of the algorithms, as an end-user would use it.

Performance evaluation Memory consumption evaluation can be quite controversial. It can be heavily impacted by the optimizations performed by the kernel and the presence of a garbage collector. For this reason, we provided values intended to give a general idea of the memory consumption of the evaluated algorithms.

We used a computer with an Intel ®Core™i7-2600 CPU and 6 GB of RAM for the desktop experiment. We monitored the performances of the VO algorithms by recording every second the process information status given on GNU Linux operating systems by the files `/proc/pid/stat` and `/proc/pid/statm`. To evaluate the memory consumption, we took into consideration the maximum value of the Resident Set Size, also called virtual memory high water mark (VmHWM), and the maximum value of the program data (Pgm Data).

The Resident Set Size is the actual part of the virtual memory used by the process which is mapped into the RAM. Therefore it is a good indicator of the RAM requirements of the target platform. The program data is the sum of the stack size, the heap size, and the size of the global plus static variables (data+bss), in other words, it is the sum of VmData and VmStck. It is mainly affected by the heap size and may be partially mapped into the RAM.

We did not take into account the Virtual Memory Size, which is the total amount of virtual memory used by a program. It includes the size of the binary and its linked shared libraries, the stack and heap usage. Unused shared libraries can dramatically increase the Virtual Memory Size, leading to misinterpretations.

To monitor the runtime performances, we took into consideration the number of processed frames per seconds and the CPU usage[4]. Due to the Quad-Core CPU of our desktop computer, our CPU load value is between 0 and 400 %.

In order to ensure the runtime performances were not affected by intensive I/O operations, we also monitored the total I/O delays provided by `delayacct_blkio_ticks`. Since the I/O delays of all the monitored algorithms was negligible compared to execution time, we did not include their values in our results.

4.3.5.3 Accuracy results

Figure 4.14, Figure 4.15and Figure 4.16 represent the bar graphs of the RPE of the evaluated algorithms on the different scenes of the RGB-D TUM dataset for SLAM. In order to ease the comparison, the different classes of VO algorithms are clustered with different hues: shades of red, green, and blue for the image-based, depth-based, and hybrid algorithms.

A first simple observation of the different graphs is that the accuracy results significantly vary from a scene to another. As stated by Fang [53], there is no algorithm which outperforms the others in all environments. The results have to be analysed w.r.t. the scene characteristics. Therefore the choice of VO algorithm depends on the target environment. Apart from the challenging scenes we described earlier and correspond to higher RPE values, the slower "fr2" scenes obtain better results than the "fr1" scenes. This illustrates well the importance of speed on the VO performances.

[4]We use the UNIX definition for CPU usage as $\sum_{c \in NumCores} \frac{time_spent_on_core_c}{elapsed_time}$, where $elapsed_time$ is the delta of system clock between the start and the end of the execution, and $time_spent_on_core_c$ are the number of system clocks spent on each of the $NumCores$ of the machine [68].

As the intuition suggests, the hybrid and image-based methods are the most accurate when the environment has texture and no structure such as the scenes "fr1 floor", "fr3 nostructure texture near withloop" and "fr3 nostructure texture far". Similarly, the environments with structure and low texture favour the hybrid and depth-based algorithms as shown by the scene "fr3 structure notexture near". Nevertheless, with the scene "fr3 structure notexture far", which has noisier depth data, the accuracy of the ICP algorithm is comparable to the image-based algorithms. On this scene, the 3D feature-based approach of MRSMAP enables to achieve the lowest RPE. When the environment is neither flat nor textureless, *e.g.* the "fr3 structure notexture near" scene, we reproduced Fang [53] results, in which image-based or hybrid-based methods are more robust than depth-based methods. However, surprisingly the addition of texture on the "fr3 structure notexture near" scene deteriorated the results of the depth-based methods. It must be noted that the results reported in [122] for the "fr3" scenes show that the image-based methods have a higher RPE than the depth-based methods on the textured scenes with low structure and vice-versa for the untextured scenes with low structure. After a comparison with our DVO results, we found out that this discrepancy of results was due to the inversion of the plot labels and to an incorrect scaling on the y-axis ticks in the paper of [122]. Also, the accuracy difference between the depth-based and image-based methods is very important, which might be explained by the lack of structure in the scene. In contrast, the scenes recorded in the office also show the hybrid and image-based methods are more robust, but the accuracy difference with the depth-based methods is slighter. A trend emerges if we compare the most accurate algorithm on each scene of the "fr1" and "fr2" series: OCV RgbdICP and Fovis have the lowest RPE on the scenes "fr1" and "fr2" respectively. DVO and OCV RGB-D generally come behind or between. There are some exceptions to this trend, but not enough to draw conclusions on them.

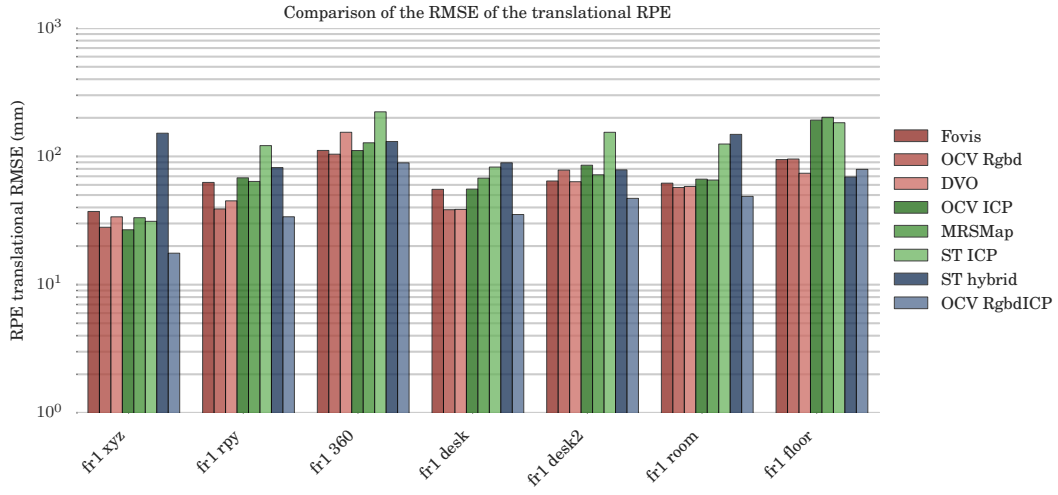


Figure 4.14: RPE comparison on the fr1 sequences of the TUM dataset.

4.3.5.4 Performance results

Runtime performances Table 4.4 illustrates the runtime performances performed with VGA frames on the TUM fr1 desk scene. It shows that only Fovis can run at the rate of the depth sensor which is 30 FPS. All the imaged-based algorithms, DVO, Fovis and OCV RGB-D, can run at a frame rate superior to 20 FPS, which is fast enough for real-time applications such as augmented reality.

The CPU load column from the Table 4.4 illustrates that all the algorithms do not fully take advantage of the multiple cores of the CPU. Surprisingly, the fastest algorithm, Fovis only uses one thread, while the slowest, MRSMAP use several ones. Also, the hybrid method OCV RgbdICP does not take well advantage of threads, while its image-based and its depth-based approaches could be run in parallel.

Memory consumption The memory performance evaluation illustrated by Table 4.4 reveals that several algorithms require more than 500 MB of program data. In contrast, the peak value of the

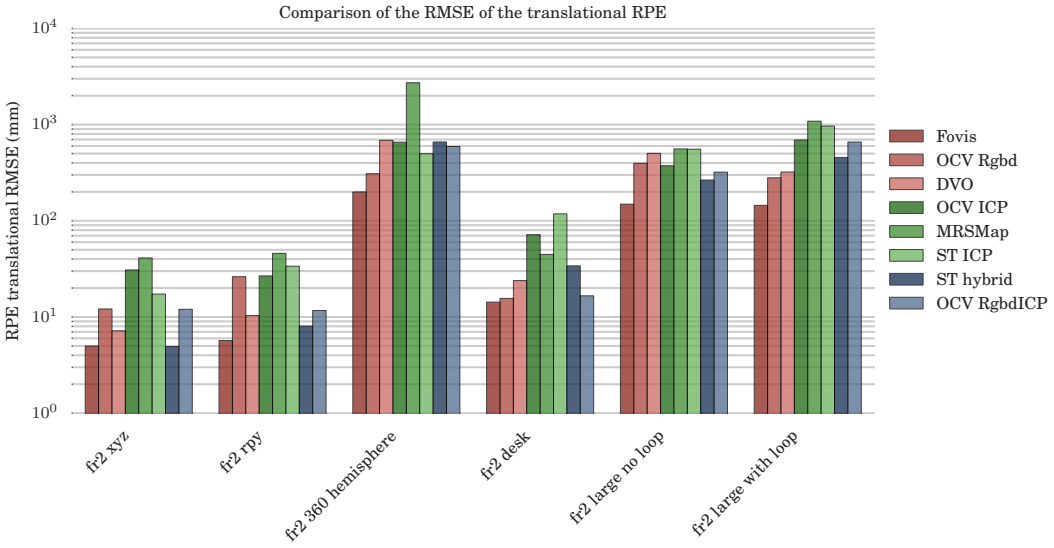


Figure 4.15: RPE comparison on the fr2 sequences of the TUM dataset.

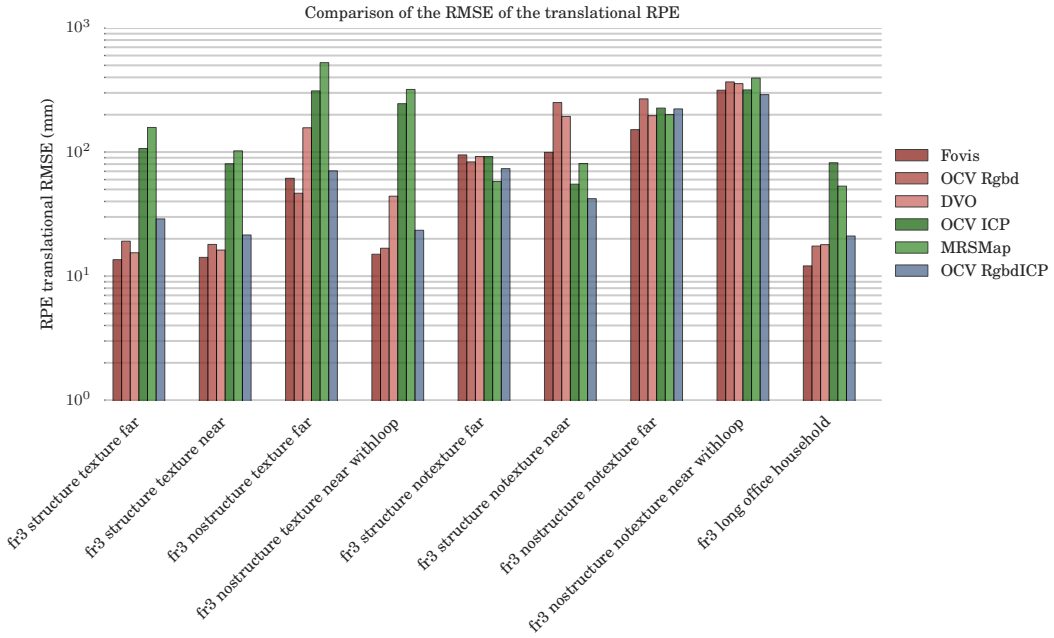


Figure 4.16: RPE comparison on the fr3 sequences of the TUM dataset.

Resident Set Size (VmHWM) is generally below 100 MB which is low enough for mobile devices. Again, Fovis is the least demanding algorithm with only 25 MB of maximal memory mapped into the RAM while MRSMap, the most demanding uses 300 MB. This comparison between the program data and the VmHWM also demonstrates that generally only a small amount of the program data is mapped into the RAM.

4.3.5.5 Experiments conclusion

From this evaluation, we selected the algorithms to evaluate on the iPad. Concerning the depth-based methods, we selected OCV ICP over MRSMAP for being slightly more accurate, less CPU-demanding and easier to compile on the iPad. We selected OCV RgbdICP which was our only hybrid algorithm, while for the image-based methods, we kept Fovis for its best runtime performance and high accuracy

Name	FPS	CPU load (%)	VmHWM (MB)	Pgm Data (MB)
MRSMap	9.3	200	332	811
OCV ICP	17.8	101	70	622
OCV RgbdICP	11.9	101	74	626
OCV Rgbd	22.7	94	50	602
DVO	23.8	288	89	536
Fovis	103.9	91	25	24

Table 4.4: Performance evaluation on the "TUM fr1 desk" scene with VGA frames performed on a desktop computer.

while we dropped DVO as it has many x86 optimizations and a similar accuracy to OCV RGB-D.

4.3.6 Mobile experiments

4.3.6.1 Second accuracy experiment: Structure Sensor acquisitions

Description of the dataset and the metrics The evaluated algorithms had their parameters optimized to give the best results on some series of the TUM dataset which used a Kinect^{SL} as depth sensor. While the two sensors share the same core technology, we wanted to check with a second experiment whether we could observe a different accuracy trend with the Structure Sensor. We connected the Structure Sensor on the iPad Air used in the previous experiment and we used a dedicated application to record locally the trajectory estimated from the STTracker VO algorithm, the RGB-D frames and the IMU data.

We recorded three scenes in three different rooms (r1, r2, r3) with various luminosity levels, denoted hl, ml and ll for high, medium and low luminosity respectively. For each scene, We also recorded different camera motions, which we denoted hs, ms and ls for high, medium and low camera speed respectively.

In the absence of motion capture cameras that could provide a ground truth for the device motion, we constrained the camera motion on an horizontal plane (*e.g.* like a ground floor or the surface of a table), using a home-made mount to secure the device in a vertical position. Therefore, instead of evaluating the drift w.r.t. a known pose, we will rather evaluate the planarity of the device trajectory. We also ensured the trajectories had their start and stop positions identical by putting the mount in contact with the same reference object at the beginning and the end of the recording. This guarantees almost perfect loop closure, any error made is negligible in comparison to the expected drift, as measured from the desktop experiment.

As for the metric to evaluate the accuracy of the algorithms, we use both the loop closing error and the RPE. The loop closing error is defined as the distance between the endpoints of the estimated trajectory divided by the path length, and it is used by most of the authors [53, 223] to compare VO algorithms. This metric evaluates the performances of the algorithms globally rather than locally, for each time step. On one hand, algorithms with a low RPE may be penalized by this metric because of one major drift; on the other hand, algorithms with high RPE may have a smaller distance because of compensations between the various drifts, somewhat like in a perfect random walk. Therefore RPE can give a better insight about the performance of the algorithm all over the device motion.

In our case, given that the trajectory is supposed to be planar and we cannot have a ground truth for the device pose, we instead evaluate the RPE as the drift along the z-axis component (*i.e.* the normal to the plane of the motion) to assess the quality of the estimated trajectory

Results and analysis The Figure 4.17 represents the RPE along the z-axis for the evaluated algorithms. We compared our metric with the loop closing error. In contrast, with the experiments on the TUM datasets, the OCV RGB-D and OCV ICP algorithms do not perform well. With the Structure Sensor dataset, the STTracker hybrid algorithm has generally the highest accuracy, matched only by OCV RgbdICP. These results show that the trends are very different on some algorithms for the Kinect^{SL} RGB-D TUM dataset and our Structure Sensor dataset.

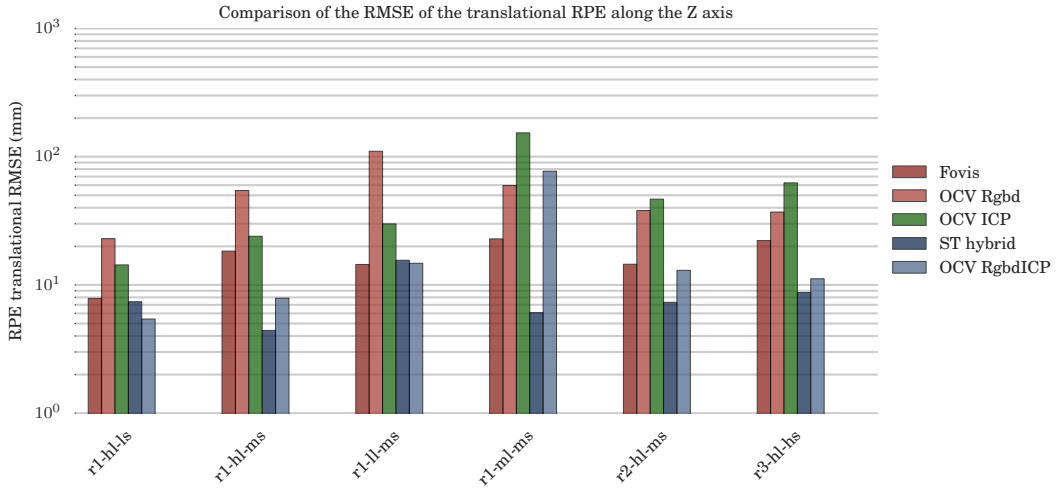


Figure 4.17: Comparison of the RMSE of the translational RPE along the z-axis.



Figure 4.18: Loop closing error defined as the distance between the endpoints of the estimated trajectory divided by the path length.

Figure 4.18 represents the same evaluation with the start-end distance metric. The trends between the two figures seem very similar. However, on "r1-hl-ms", "r1-ll-ms", "r2-hl-ms" and "r2-hl-ms" the ranking of the Fovis, OCV RGB-D and OCV ICP algorithms is very different with the two metrics.

4.3.6.2 Performance evaluation

The performance experiment was carried out on four mobile devices: two iOS devices, iPhone 5 and iPad Air and two Android devices, Memo Pad 7 and Tango TDK. As mentioned in the Section 4.3, the Tango TDK has a dedicated VIO module requiring different input: thus, it cannot be fairly compared with the RGB-D VO algorithms, which are the object of this article. Table 4.5 displays the characteristics of these mobile devices. The PassMark CPU benchmark assesses trough various intensive parallel computational algorithms how fast a CPU is. The scores give an indication of the CPU speed, the faster is the processor, the higher is the score. We took the value of the Android[5] and iOS[6] ranking available on November 17th, 2016. The scores indicate the iPhone 5 and the Memo Pad 7 can be considered as middle performance devices, whereas the iPad Air and the Tango TDK can be

[5] http://www.androidbenchmark.net/cpumark_chart.html
[6] http://www.iphonebenchmark.net/cpumark_chart.html

considered as high performances devices.

We compiled the previously selected VO algorithms with the `-O3` optimization option and used the "TUM fr1 desk" scene to evaluate their performance, with two different image resolutions: VGA (640×480) and QVGA (320×240).

Manufacturer	Model	CPU	CPU PassMark score	RAM (GB)
Apple	iPad Air	Apple A7	37517	1.0
Apple	iPhone 5	Apple A6	23914	1.0
Asus	Memo Pad 7 K013	Intel ® Atom ™Z3745	27807	0.86
Google	Tango Yellowstone	Nvidia Tegra K1	38503	3.7

Table 4.5: The mobile devices used for the performance evaluation with some of their hardware specifications and their PassMark score (the faster the CPU the higher the score).

Algorithm	Device	QVGA (FPS)	VGA (FPS)
Fovis	iPad Air	92.0	24.1
	iPhone 5	38.7	10.2
	Memo Pad 7	81.6	20.1
	Tango Yellowstone	96.0	26.0
OCV RGB-D	iPad Air	28.6	6.7
	iPhone 5	13.7	3.6
	Memo Pad 7	8.9	2.4
	Tango Yellowstone	17.2	4.3
OCV ICP	iPad Air	23.8	5.5
	iPhone 5	9.7	2.5
	Memo Pad 7	7.9	2.1
	Tango Yellowstone	14.4	3.6
ST ICP	iPad Air	43.7	42.6
	iPhone 5	23.3	20.9
ST hybrid	iPad Air	36.4	28.3
	iPhone 5	19.2	16.7
OCV RgbdICP	iPad Air	14.3	3.2
	iPhone 5	6.4	1.6
	Memo Pad 7	4.3	1.2
	Tango Yellowstone	8.1	2.0

Table 4.6: Performance evaluation on the "TUM fr1 desk" scene with QVGA (320×240) and VGA (640×480) images performed on the four mobile devices.

Table 4.6 shows the results of the performance experiment. For each algorithm, the frame rate (fps) is reported for each device for which it was possible to port the algorithm. It must be noted that we chose to report the frame rate as a performance measure as it can be used as a reference for assessing the suitability of the algorithm for a given application. In general, computer vision applications are usually considered to be real-time when they are able to assure a minimal throughput in terms of images processed per second. This clearly depends on the target application and the type of time constraints that must be guaranteed [18]. Visual odometry can be used to enable applications like augmented reality or more general robotic applications, and in general, for this range of applications, a frame rate of $15\,\text{fps}$ is commonly considered a minimal threshold to assure the responsiveness of the application.

As it can be noted from the table, downsampling the images from a VGA resolution to QVGA clearly improves the computational performances up to a factor of 4. However, this has sometimes an

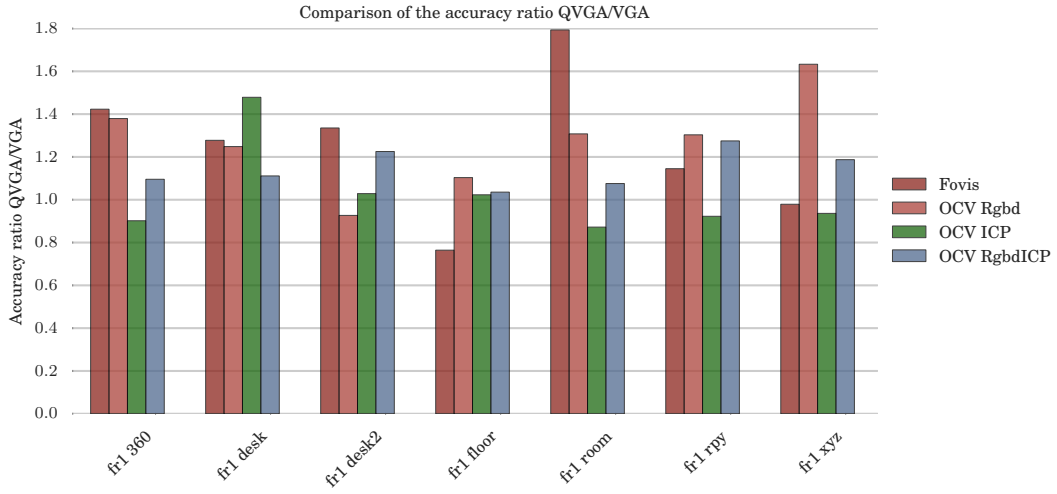


Figure 4.19: Comparison of the RPE ratio QVGA/VGA on the fr1 scenes of the TUM dataset.

impact on the estimated trajectory and the accuracy of the results. Figure 4.19 shows the ratio between the VGA and the QVGA accuracy: using half resolution generally worsens the achievable throughput (ratio greater than 1), with some exceptions in which we observed a slight improvement of accuracy (ratio lesser than 1).

In the case of the STTracker, it can be noted that resolution does not affect the performances, as there is only a slight difference between the two resolutions. Since the code is not available and the documentation is not clear on this point, we can only speculate that the algorithm always downsamples input VGA images to ensure a high frame rate.

More generally, Fovis is the only algorithm that can achieve high frame rates at VGA resolution on high-end devices, and it anyway outperforms the other algorithms on iPhone 5. It is worth noting that Fovis' SSE2 optimizations were disabled when running the tests on the Memo Pad 7 as they led to a slight loss of performance. All the other algorithms fail to reach real-time performances at VGA resolution, even on high-end devices: OCV RGB-D and OCV ICP are the only ones passing 5 fps on iPad Air. When using QVGA resolution, the performances of all algorithms improve and generally the iPad Air is the device getting higher frame rates for every considered algorithm. As for the OpenCV family of algorithms, OCV RGB-D only fails to achieve real-time performances on Memo Pad 7, OCV ICP can provide high frame rates only on the most powerful devices, iPad Air and Tango TDK, while OCV RgbdICP only comes close to the threshold of 15 fps on iPad Air.

Concerning the devices, the iPad Air is twice as fast as the iPhone 5, and the Memo Pad 7 is twice as slow as the Tango TDK, with the exception of the Fovis algorithm.

4.3.6.3 Discussion

Even if modern mobile devices have CPUs with 2 or 4 cores up to 2.3 GHz, their computational power cannot be exploited at their full potential for a long period of time without draining the battery and risking some over-heating of the device. Since they are designed to be power efficient, their frequency is often throttled down and their instructions set is reduced. Moreover, the current hardware architectures of mobile devices have reduced L1 and L2 cache and a reduced instruction set. Therefore, when optimizing the implementation, developers should pay particular attention to the memory accesses, for example taking advantage of the pre-fetching and maximizing the processing on small blocks of data.

As a general rule, polymorphism should be limited or used carefully, as it may introduce performance overheads and it may lead to indirect function calls which are less likely to be optimized at compile time [217]. For example, in a study over a large set of programs Driesen *et al.* [48] showed that, in average, 5.3 % of the time is used to deal with polymorphism, and 13.7 % for "all virtual" versions of the program. For the examined algorithms, we can note that OpenCV highly uses polymorphism both for data structures and algorithms, while at the other end, Fovis uses polymorphism only for the abstraction of the input data source.

Standard computer vision libraries such as OpenCV and ROS [158] are extremely useful tools for developing, prototyping and testing algorithms. On the other hand, these libraries were originally designed mostly for desktop environments, and only recently the porting to mobile environment has been started. Despite these efforts, at the moment of writing, they still lack of adequate and complete code optimization, supporting *e.g.* specific instruction set like ARM-NEON that could fully exploit the specific hardware of modern mobile devices. Moreover, the use of `float` over `double` data type is recommended for runtime and memory performance: even if the most recent ARM processors are 64 bit CPUs, for the time being there are no instructions that support double precision operations [10], which introduces type conversion overheads that significantly affects the performances [113].

Parallelization can also improve the performances of the algorithm when ported in the mobile environment. Computationally intensive algorithms for image processing can be parallelized by means of shaders that can speed-up the computation by running on the GPU of the device. For example, all the pre-processing blocks of Figures Figure 4.8–Figure 4.11 used to pre-process the RGB-D image using classic image processing algorithm (bilateral filtering, image smoothing, *etc*.) can be implemented as a shader. Multithreading is also another natural way to optimize the pipeline execution. As showed in Section 4.3.4 when describing the algorithm pipelines, all the blocks that are vertically aligned can be actually run in parallel by different threads. This is especially adapted for the algorithms that require to processing both the depth and the RGB image, such as OCV ICP, OCV RGB-D and DVO. Splitting the processing of each input image into different threads will certainly benefit the performance of the algorithm, and optimize the resource consumption. More sophisticated parallelization can also be employed combining pipelining and look-ahead strategies [165], so that the device resources can be fully exploited. For example, a pipelined version of OCV RGB-D (*c.f*. Figure 4.9) could reserve two threads for processing the two input images as they are available instead of waiting for the whole pose estimation process to end. This allows improving the throughput of the algorithm at the cost of more complexity of the implementation.

An IMU offers a good combination with VO algorithms because of its complementary with visual sensors. Inertial data is computationally cheap, it deals well with rapid movements, but it suffers from drift and measurement noise. On the other hand, visual data can provide more precise and stable measurements, which can be used as reference to prevent the inertial measurements from drifting. Inertial data are particularly adequate for algorithms based on iterative solvers that need a first initial solution, and algorithms where a rough estimate of the motion is used. For the Fovis algorithm, for example, the "features matching" step takes advantage of the knowledge of the rotation between the frames: the method uses pixel errors between images to infer the rotation, whereas as stated by the authors, the IMU data could provide the same kind of information at a lower computational cost. The DVO algorithm is designed to be used with a motion prior, again an inertial sensor can fulfil this task. Brunetto *et al*. [25] demonstrated that higher robustness of the pose estimation of their features image-based vSLAM algorithm SlamDunk could be achieved with the use of IMU data from a Samsung tablet.

Considering the scenario of a mobile augmented reality application where 24 fps or higher is recommended, Fovis with QVGA images appears to be the best choice since it can run at high frame rates on middle and high performance mobile devices. In the case iOS, only the devices with high-end specifications such as the iPad Air can achieve real-time performances, with the exception of the STTracker algorithms, which are specifically optimized for this environment.

4.3.7 Benchmark conclusions

In this section, we presented a classification and a theoretical review of RGB-D VO algorithms. We tested and analysed the performances on a mobile device of six visual odometry algorithms designed for RGB-D sensors on different mobile devices covering the two most common mobile operating systems, iOS and Android. We selected the most promising algorithms to test based on previous benchmarks found in the literature, and our tests on desktop environment to assess the computational and memory performances before porting them to the mobile environment. The performances of each algorithm were analysed in terms of accuracy and time and memory consumption, which is a fundamental aspect when deploying a visual odometry algorithm on a device with limited resources. We assessed and confirmed the algorithms accuracy on the state-of-the-art RGB-D TUM dataset and we collected the time and memory consumption of the algorithms to get a first rough estimation of the

resources needed.

After selecting the most promising algorithms in terms of accuracy and resources consumption, we run several tests on mobile devices to assess both the actual performances on the mobile devices and the accuracy on our own dataset.

In general, results showed that only high-end devices such as iPad Air can guarantee some adequate frame rate at normal resolution (VGA). Reducing the resolution of the input image proved to increase the throughput, yet sometimes at the expense of the accuracy. On the other hand, algorithms provided with the Structure SDK, the ST hybrid [136], can achieve a good accuracy and faster execution times even at full resolution. Since the code for the latter is closed-source, we can only expect that the code is specifically designed and optimized for the mobile settings. As for the open-source algorithms, only Fovis could achieve frame rates up to 24 fps and perform adequately on all the four available mobile devices. This could be explained by the fact the algorithm was already designed for running on micro aerial vehicles, thus privileging a simpler implementation adapted to limited resources environment. Results also showed that implementations relying on standard computer vision libraries such as OpenCV, are still lacking a proper support for mobile architectures. These tools are quite useful for quickly prototyping the implementation of an algorithm, but, for the time being, they are still oriented to the desktop environment and their complexity is not well suited for mobile environments.

In the light of the evidence shown in this paper, it appears clearly that designing a VO algorithm for mobile environments requires to thoughtfully adapt the implementation to the limited resources available to achieve a good trade-off between accuracy and throughput. The VO algorithms provided by the hardware makers such as Occipital are the best bases upon which building an application, as they are finely tuned for the specific environment. On the other hand, developing an original VO algorithm requires a thorough design of the algorithm from the ground-up.

4.4 Planar approaches

We have seen previously VO algorithms suffer from drift. These localization errors can be limited with a global map and trajectory optimization algorithm. Due to the amount of data considered, these algorithms are slower than VO algorithms, so they are usually run punctually on a separate thread. The Google Tango SDK proposes a fast global map optimization algorithm [116], optimized for the mobile platform. However, it seems the speed was trade at the expense of the accuracy. Figure 4.20 illustrates this problem: some walls remain thick or duplicated after the global optimization, and in some cases, the solution is locally deteriorated. This effect can sometimes be explained by the absence of texture in some areas of the scenes (*e.g.* painted walls), leading to the absence of local feature points. In these areas, the drift could be reduced if the Google Tango VIO algorithm took advantage of the 3D data provided by its range imaging sensor, but would it be enough? Several studies [106, 35, 70] demonstrated that some depth-based and hybrid vSLAM approaches such as Kinect Fusion [127], Sun3Dsfm [190], Kintinuous [211] and DVO-SLAM[7] [92] failed to reconstruct globally consistent large scenes. In the absence of planar constraints, the reconstruction of large scenes generates curvy floors and walls.

In order to improve the geometric consistency of our reconstructed scenes, we decided to consider planar primitives. Planes are interesting features: they can be considered for large-baseline registration since they remain visible after important camera displacements, they do not suffer for the presence of repetitive texture or their absence, and they provide the structure of the room to reconstruct. Walls are static and there is no reason they become completely occluded, thus they are suitable high-level landmarks.

The study of planar approaches occurred after we had selected the Tango TDK as target platform, at the expense of the Structure Sensor. In this section we will give some background on planar approaches for VO and global optimization, detailing the different components: extraction of the planes, fusion of planes, registration and optimization with planes. We will then present our attempt to build a similar system and the obtained results on acquisitions from the Tango TDK.

[7] Note that the DVO [93] algorithm we evaluated earlier is an older version which is image-based, whereas [92] is a hybrid approach.



Figure 4.20: Top down views of point clouds from room scans with the Tango TDK colored with a jet color map applied on the points depending on their altitude (red for lower points, blue for the higher ones). The left and right images represent the point clouds before and after the provided global map optimization was performed, respectively. Green and red ellipses represent area which were improved and deteriorated by the map optimization, respectively. Orange ellipses represent area where the walls remained duplicated or thick after the map optimization.

4.4.1 Overview and related works

Planar approaches follow a common framework we summarize in Figure 4.21. The range images are processed in order to extract sets of 3D points corresponding to candidate walls, generally called *planar patches* or planar regions. They are associated with the eventual help of registration prior (*e.g.* ICP, wheel odometry, *etc.*), and fused with the previously extracted planar patches. Then, the pose registration can be estimated by estimating (or correcting) the camera pose which aligns the associated planar patches. Finally, the global consistency of the fused planar patches can be improved with an optimization procedure on the camera pose and the detected planes.

4.4.1.1 Planar patch extraction

A planar patch π is defined as a list of 3D points associated with the equation of the fitted infinite plane and the boundaries of this patch. We define the planar patches extraction problem as a function which

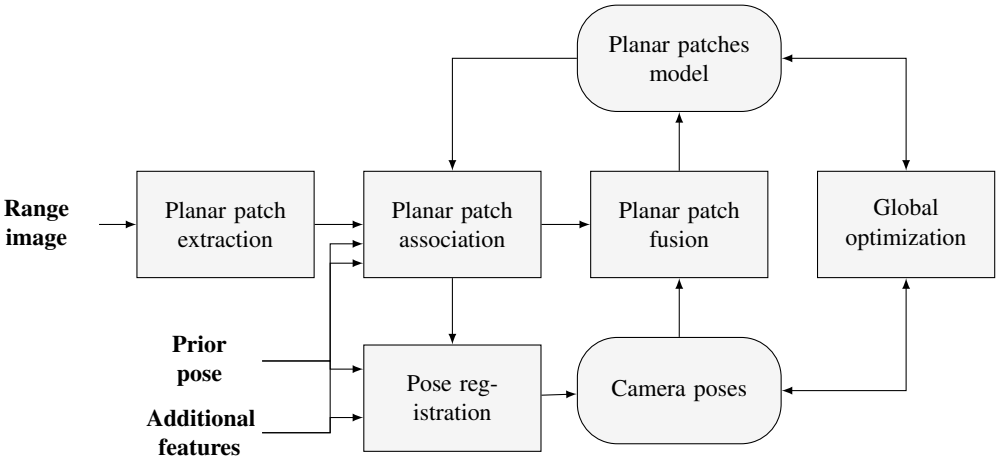


Figure 4.21: The common pipeline of planars SLAM approaches. Rectangles with rounded corners denote data, while the other rectangles represent functional components.

Reference	Plane extraction	Plane uncertainty	Additional features	Prior pose	Global optimization
[145]	Region growing	Yes	No	No	Graph-based SLAM (translation estimation only)
[107]	Normal map	Yes	No	Yes	Graph-based SLAM
[46]	Hough Transform	No	SIFT	No	Bundle adjustment
[199]	RANSAC	No	SURF	No	Bundle adjustment
[175]	Normal Map	No	No	ICP	No
[87]	Normal map	Yes	No	No	Graph-based SLAM with plane observation and potential odometry constraints
[117]	Split & merge	No	RGB Image	No	Graph-based SLAM with odometry and plane observation constraints
[80]	RANSAC	No	RGB Image	No	Graph-based SLAM with odometry and plane observation constraints

Table 4.7: Comparison of SLAM planar approaches sorted by ascending year of publication.

takes as input a range image and returns a list of planar patches.

The extraction of fast planar patches in range images has been extensively studied since the 90's. Hoover *et al.* [78] compares some of the proposed algorithms of that time. We can note that older plane fitting approaches [91, 143, 209] (especially when the density of the point cloud is low) take into consideration the sensor noise model, whereas approaches posterior to the Kinect^{SL} release do not [46, 54]. In this section, we briefly recall some popular strategies presented in recent papers. For a more detailed review of planar regions extraction, we refer the reader to [54, 84, 150].

RANSAC based approaches [204] apply iteratively the RANSAC algorithm to find the most dominant plane of the scene. The inlier points are removed and the dominant planes of the remaining points are repeatedly extracted.

Hough Transform approaches [23, 46] consider each 3D point and fill an accumulator which votes for all the possible planes the point can lie on. The peak values of the accumulator indicate the possible planes. However, the cluster associated with one peak may contain non-adjacent points.

Region growing algorithms [144, 150] take a random seed point and some neighbors and extend this set of points by taking into consideration neighboring points. A plane is estimated on this set of points and a new point is considered valid when its distance to the plane is small enough. The planar patch keeps growing iteratively until no valid point can be found in the neighboring of the patch. A new seed point is picked until all points have been considered. Some points may be incorrectly segmented, particularly at the intersection of two planar patches as illustrated in Figure 4.22

Normal map segmentation computes the normal map and segments it with a region growing approach [107] or a voting approach [76]. On each normal cluster, the plane parameters are estimated. The clusters must be segmented again to separate possible parallel planes.

Split and merge approaches perform multiple planar extractions on partitions of the data and merge the contiguous similar planar patches. Feng *et al*. [54] proposes a block splitting function working in the depth map and a merging strategy which balances the size of the patches for performance issue. Because of the size of the block, the planar segmentation is coarse, and a refinement process with a pixel-wise region growing approach is required to increase the resolution of the segmentation. Depending on the size of the block, this approach may not detect thin planar patches.

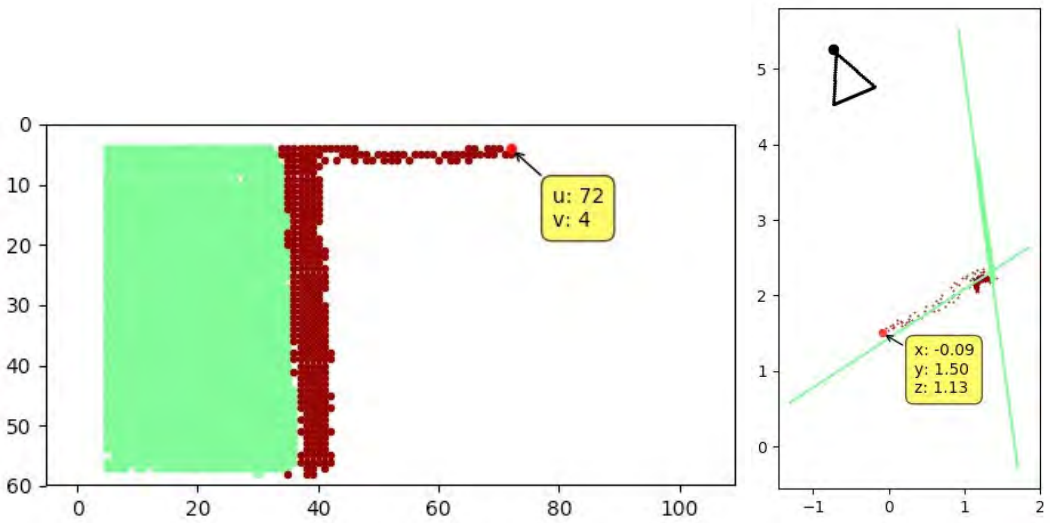


Figure 4.22: A part of the ceiling (undetected here) was segmented in the red planar region (which corresponds to a vertical beam). A normal map approach would have correctly segmented the range image since the normals of the incorrectly classified red points are collinear with the gravity vector.

A boundary of the 3D points associated to the planar patch is generally computed, in the form of an oriented bounding box, a convex or concave hull of the 3D points projected on the fitted plane.

These recent papers do not address the issue of choosing among several plausible planar segmentations. For example, aligned furniture, thin sliding doors, plinths on a wall, half height wall tile *etc*. create adjacent parallel planar surfaces, and depending on the noise level, it can be difficult to determine whether one or two Planar Patches (PPs) should be detected. A previously seen PP can be later perceived as two PPs after a displacement of the user to get closer to this ambiguous area and reduce the range measurement noise. Another issue concerns the non-planarity of some parts of the surface which can be observed for very large planar surfaces, especially on the ceiling and the floor, *e.g*. to allow the drainage of water, or because of sagging in some areas of the structure. Current approaches may over-segment these areas and also propose changing results depending on the noise level. It is a difficult problem, which strongly depends on the level of details and the granularity of model that is required by the application.For the sake of this work, we prefer an over-segmentation since it easier to design a user interaction to fuse PPs than to split PPs.

4.4.1.2 Planar patch association and fusion

A model of planar patches in the world coordinates is usually created with the patches of the first frames, which are brought into a global world coordinate system thanks to the known camera pose. For each successive frame, the extracted planar patches are associated with the model via an *association function*. The associated planar patches of the model are updated with a *fusion function*, which generally performs a weighted mean of the estimated plane parameters and computes the union of the patches boundaries.

Given a list of planar patches from the model $M = \{\pi_i^m\}$ and a list of planar patches from the current frame $C = \{\pi_i^c\}$, the *association function* computes a list of planar patches pairs where each patch π_i^c appears only once.

The planar patches association step is usually tackled by comparing metrics such as the normal angle difference and an offset distance between the considered planes. For example, [107, 117] compare the distances of the planes to the origin, which depends on the origin choice and the estimated normal. As illustrated in [46, Figure 5], "the closeness of planes parameters is not equal to the closeness of planar patches". To cope with this issue, [208] and [87] compute a vector between the gravity centers of the planar patches, which is projected on one of the plane normal to define a PP-to-PP distance. [21, 117] consider an additional metric based on the overlapping of the patches, while [164] adds the comparison of the planar patches dominant color. [199] also takes advantage of the RGB images with the help of visual local features (SURF). The use of uncertainty metrics on the planar patches favors the use of probabilistic approaches. For example, [210] performs a χ^2 test with the SPmodel of the planar patches while [107] defines two Mahalanobis distances with the covariance matrices of the points of the planar patches and the covariance matrix of the plane parameters.

The pairs of patches $\{(\pi_i^m, \pi_j^c)\}$ with the lowest metric are kept, but some outliers may remain. They are filtered by [107, 144] with algorithms inspired by RANSAC.

4.4.1.3 Planar registration

Given two consecutive frames and the set of corresponding features, the planar registration is generally achieved by searching the transformation minimizing a distance between the matched features. Plane-only approaches consider a plane-to-plane distance only: for example [107] uses a Mahalonobis distance which takes into account his uncertainty model. This approach suffers from an important issue: depending on the number of associated planes and their configuration, the camera pose may be not enough constrained. For example, to estimate the translational component of the transformation $\mathbf{t_{c\to m}}$ between planes of the current frame $\{\mathbf{\Pi}_i^c\}$, to planes of the model $\{\mathbf{\Pi}_i^m\}$, a minimum of three associated planes with linearly independent normals are required. This requirement appears in Equation (4.1), where we use the Hessian plane model: $\mathbf{\Pi}_i = \begin{bmatrix} \mathbf{n_i} \\ -d_i \end{bmatrix}$, where $\mathbf{n_i}$ is the unit vector of the normal of $\mathbf{\Pi}_i$ and d_i the distance of the plane from the origin. It also reveals a problem of numerical stability, since three similar (nonidentical) $\{\mathbf{n_i^m}\}$ vectors may still give a (degenerate) solution. Such a situation can be detected with a numerical rank analysis as proposed in [145].

$$\begin{bmatrix} \mathbf{n_1^m}^\mathsf{T} \\ \vdots \\ \mathbf{n_i^m}^\mathsf{T} \\ \vdots \\ \mathbf{n_n^m}^\mathsf{T} \end{bmatrix} \mathbf{t_{c\to m}} = \begin{bmatrix} d_1^m - d_1^c \\ \vdots \\ d_i^m - d_i^c \\ \vdots \\ d_n^m - d_n^c \end{bmatrix} \tag{4.1}$$

The probability of degeneracy can be reduced by considering additional features: [46, 199] also minimize the distance between 3D points corresponding to image local features and matched via their descriptors, while [80, 117] include a photometric residual corresponding to a dense and semi-dense registration approaches.

4.4.1.4 Global optimization

The main approaches usually solve a global optimization of the structure (the map), and the camera poses (the localization) are the Bundle Adjustment (BA) and the Least Square (LS) Simultaneous

Localization And Mapping (SLAM).

Bundle Adjustment (BA) [205] takes its essence with SfM for adjusting a bundle of rays corresponding to the observation of local image features by adjusting the camera poses and the 3D positions of the features. Formally, it corresponds to a nonlinear LS problem which jointly optimizes all the camera poses and all the observed features. Therefore, the resulting equations and the resolution approaches are similar to those of LS SLAM. Contrary to the latter, there is no constraint between the camera poses (the order of consideration of the camera poses w.r.t. the time does not matter), and the approach is not necessarily probabilistic.

LS SLAM [27, 66] is also called Maximum-a-Posteriori SLAM or graph-based SLAM. It is a probabilistic approach, which takes its essence from the robotics community, where a common problem consists in correcting the positions of observed features and the positions of a robot: the order of positions matters and an estimation of the difference between two consecutive positions is often known (*e.g.* with wheel encoder). With this formulation, the constraints of the problem come from the measurements of the different sensors (and not only from a camera sensor as for BA), and the trajectory and the map are updated over time, as the robot moves and new data is acquired. Its objective is to estimate a variable (a.k.a. state) Θ, usually including the camera poses and the landmarks, which maximizes the coherence of predicted values $h_k(\Theta)$, with measurements z assumed independent. Formally it corresponds to Equation (4.2), considering the independence of the measurements and applying Bayes' theorem.

$$\Theta^* = \arg\max_{\Theta} \mathbb{P}(\Theta \,|z_1, \dots z_N) = \arg\max_{\Theta} \mathbb{P}(\Theta) \prod_{k=1}^{N} \mathbb{P}(z_k|\,\Theta) \tag{4.2}$$

The terms $\mathbb{P}(z_k|\,\Theta)$ are usually assumed Gaussian, which implies the error between a measurement and a prediction is affected by the covariance matrix $\Sigma_{\mathbf{K}}{}^{-1}$:

$$\mathbb{P}(z_k|\,\Theta) \propto \exp\left(-\frac{1}{2}||h_k^{\mathrm{o}}(\Theta) - z_k^{\mathrm{o}}||^2_{\Omega_k}\right) = \exp\left(-\frac{1}{2}(\mathbf{h}_{\mathbf{k}}^{\mathrm{o}}(\boldsymbol{\Theta}) - \mathbf{z}_{\mathbf{k}}^{\mathrm{o}})^{\mathsf{T}}\, \Sigma_{\mathbf{K}}{}^{-1}\, (h_k^{\mathrm{o}}(\Theta) - z_k^{\mathrm{o}})\right) \tag{4.3}$$

The inverse of the covariance matrix is called the information matrix $\Omega_k = \Sigma_{\mathbf{K}}{}^{-1}$.

If there is no prior knowledge, the prior probability $\mathbb{P}(\Theta)$ can be removed from the maximization, otherwise, it can also be modeled with a Gaussian distribution:

$$\mathbb{P}(\Theta) \propto \exp\left(-\frac{1}{2}||h_0^{\mathrm{o}}(\Theta) - z_0^{\mathrm{o}}||^2_{\Omega_0}\right) \tag{4.4}$$

It follows that Equation (4.2) becomes Equation (4.5).

$$\Theta^* = \arg\min_{\Theta} -\log\left(\mathbb{P}(\Theta) \prod_{k=1}^{N} \mathbb{P}(z_k|\,\Theta)\right) = \arg\min_{\Theta} \sum_{k=0}^{N} ||h_k(\Theta) - z_k||^2_{\Omega_k} \tag{4.5}$$

This problem can be represented by a *factor graph* where the nodes (called *variable nodes*) represent the variables of the state to optimize, and the edges represent the constraints between the variables. These constraints correspond to different measurements and are labeled with *factor nodes*.

Variables parametrization defines how the variables (usually the planes and the transformations) are represented. The parametrization should be ideally minimal in order to enable the inversion of the covariance matrix Σ_k. For the parametrization of the plane, [46, 107, 117] consider spherical coordinates: the normal is parametrized with two angles, the third component being the distance of the plane from the origin. The main downside of this approach is the gimbal lock problem, when two axes become aligned during the optimization. To avoid this issue, Hsiao and Kaess [80, 87] normalize the plane parameters on the unit sphere, which is viewed and handled as a rotation in a 3-dimensional Lie group. For the transformations, there is a similar difficulty to find a minimal parametrization of the rotation. Hsiao *et al*. consider a spherical representation, while [46, 117] consider twist coordinates of the Lie algebra $\mathrm{so}(3)$.

Regarding the expression of the covariances matrices Σ_k associated with the variables representing the planes, there is little mention about it in the literature. Ma *et al*. [117] assume the uncertainty of the plane parameters is isotropic. In contrast, [80, 107] take into account the 3D points used for the estimation of the plane to estimate the covariances associated to each variable.

4.4.2 Our approach

Our system is divided into two parts: a front-end which handles the sensors data at the speed rate of the acquisition, and a back-end which performs corrections on the data produced by the front-end at a slower rate (see Figure 4.23). In the front-end, the images from the fisheye camera and the inertial data are processed by the VIO component from the Tango middleware, which provides camera poses at the camera frame rate. The PPs components take the depth data as input to extract, associate and fuse the planar patches. Thanks to the known camera pose, the PPs are brought into a global world coordinate system, so that they can be associated and then fused with the existing patches of the model by association and fusion functions. On the back-end side, the Tango global optimization component [116] optimizes and updates the camera poses after loop-closures detection. As we have seen earlier (Figure 4.20), this component reduces on average the errors of the global model due to the drift of the VIO algorithm, but some walls may remain duplicated despite the optimization. For this reason, we designed a back-end component relying on planar primitive to improve the consistency of the PPs model: it is run at the end of the scan to avoid competition with the back-end provided by the Tango SDK.

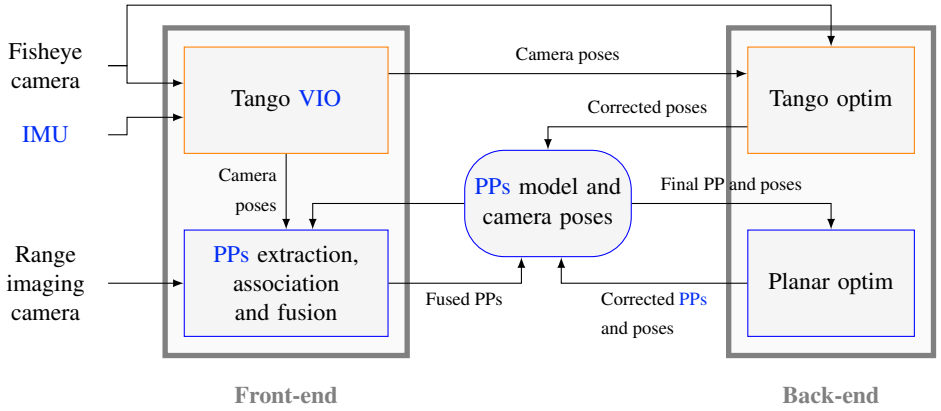


Figure 4.23: Pipeline of the interactions between our modules and Tango components. The components provided by Tango are displayed in orange, and the components we designed are in blue. The input of our system are the data provided by the IMU, the fisheye camera and the range imaging camera. Front-end components ingest sensors data and process them in real-time, while the back-end components perform corrections of the front-end outputs, at a slower pace. At the end of the scan, the global optimization of the planar primitives map and the camera poses is performed to remove ghost walls.

4.4.2.1 Noise model

In order to perform the extraction of planes, we needed a noise model of the Tango TDK depth sensor to predict the accuracy of the depth data and adjust accordingly some thresholds sensitive to the data noise. We designed an experiment where the tablet was observing multiple walls from various distances and orientations. The PPs of the walls were extracted via the method described in Section 4.4.2.3, and we considered the distance between the camera sensor and the PP centroid $\overline{\mathbf{X}} = {}^{1}/_{N_p} \sum_{i=1}^{N_p} \mathbf{X}$ where $\mathbf{X}$ is a point of the extracted PP. The noise of depth data was estimated by computing the PP thickness $\sigma = 2\sqrt{\lambda_3}$, where $\lambda_1 > \lambda_2 > \lambda_3 > 0$ are the eigenvalues of the covariance Cov_π of the points:

$$\text{Cov}_\pi = {}^{1}/_{N_p} \sum_{\mathbf{X} \in \pi} (\mathbf{X} - \overline{\mathbf{X}})(\mathbf{X} - \overline{\mathbf{X}})^\mathsf{T}. \tag{4.6}$$

The limit of this approach is that it assumes the plane thickness is constant, which is an approximation. Figure 4.24 shows the measured thickness of the planes for different orientations and distances. We can observe that some planes with a significant orientation of the plane w.r.t. the sensor have a lower thickness than other planes parallel with the image plane of the sensor (low orientation value). For this

reason, the orientation of the plane w.r.t. to the camera is not taken into account in our noise model, as in the model proposed by [94].

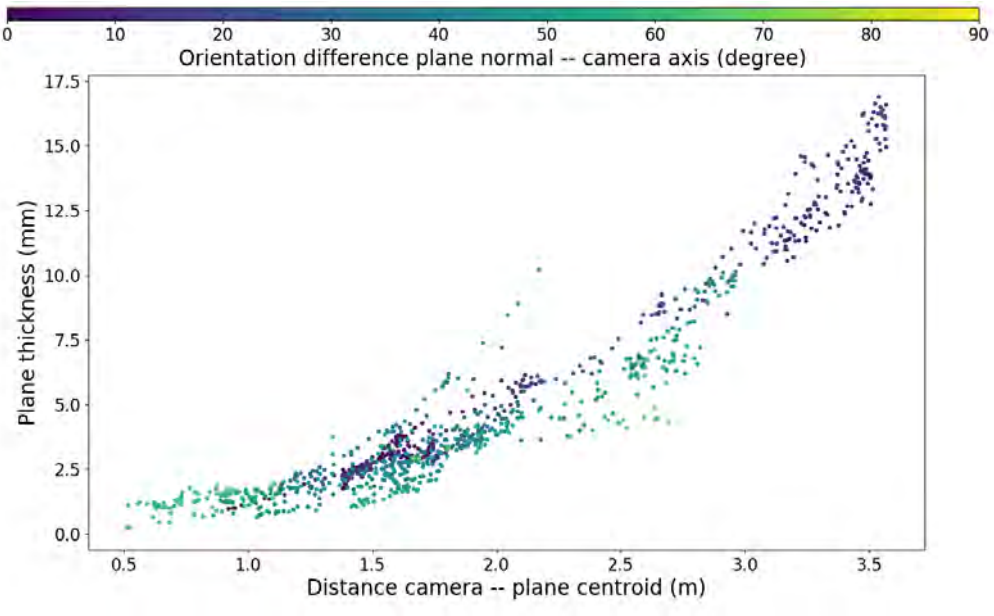


Figure 4.24: Noise model of the Tango TDK depth sensor. Each point represents the thickness of an observed plane at a certain orientation (represented by the colormap) and distance (the x-axis). We can observe that some purple points (planes perpendicular to the camera axis) are noisier (larger thickness) than some planes more tilted w.r.t. the sensor (green points).

4.4.2.2 Depth data pre-processing

The Tango SDK does not give access to the range image through the API, only the list of the 3D points is available to the developer. Hence the range image has to be generated by projecting the provided 3D points with an extra computation cost. Moreover, range image processing requires a relatively dense depth map. Unfortunately, as explained in Figure 4.25, projecting the 3D points with the provided intrinsic camera parameters results in very sparse depth maps. It is then necessary to reduce the range image size by projecting them on a smaller image (at the cost of the loss of 3D points), to obtain denser range images, suitable for various processings such as the computation of integral images [77].

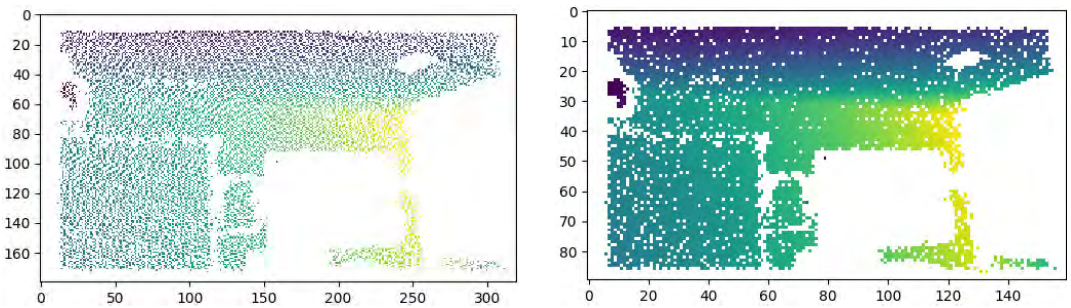


Figure 4.25: Left: range image obtained by projecting (using the provided camera intrinsic parameters) the 10k 3D points corresponding to one frame acquired by the Tango TDK. The obtained depth map is very sparse, around 27% of the pixels of the image have a value. Also 5% of the 3D points are lost (*i.e.* 5% of the 3D points project on a common pixel). Right: the range image obtained after dividing by two its dimensions, 35% of the 3D points are lost, and despite this downsampling, the depth map still contain many holes to use [77]. To obtain a dense range image, we reduce the image size with a 0.33 - 0.4 scale factor, leading to a loss 67% - 54% of the 3D points.

4.4.2.3 Planar patch extraction

The planar patch extraction module takes as input the range image generated in Section 4.4.2.2, which is segmented into regions corresponding to the same plane equation. This segmented image is then processed to generate a list of PPs.

We first selected the normal map segmentation approach. This choice was motivated by the simplicity of the approach, the high performances obtained by [76] (1.95 ms to process 10 k points), and the possibility to consider more generic primitives in the future, such as cylinders. The normal estimation was performed using integral images [77]. This algorithm requires dense range images, thus we reduced the image size of a 0.33–0.4 scale factor, leading to a 67 %–54 % loss of the 3D points, respectively. As illustrated in Figure 4.26 this step revealed to be quite sensitive to the noise.

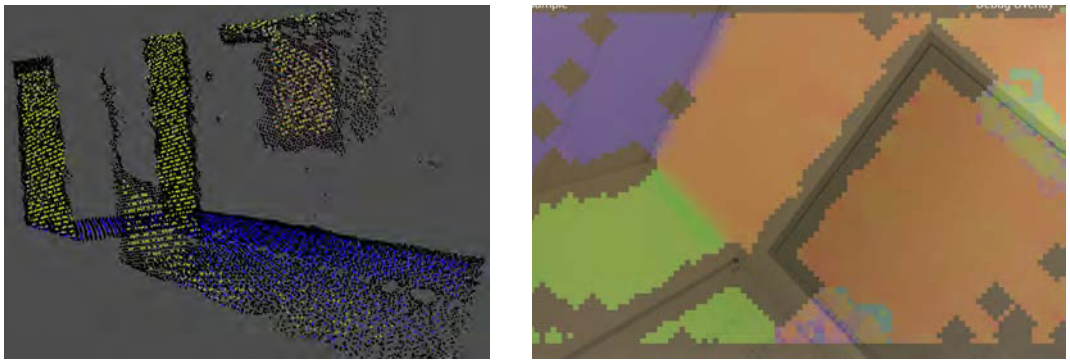


Figure 4.26: Left: points cloud from the Tango TDK colored according to the estimated normals (the RGB components are proportional to the three directions of the estimated normal). The normals are computed via [77]. 3D points with no normal (*e.g.* points lost by the downsampling) are displayed in black. Right: tests on the Tango TDK where we display the downsampled normal map on the RGB image. Missing 3D data create large holes because we downsampled the depth map and a neighborhood of points with similar depth is required to compute the normal around a point.

Switching to a region growing approach inspired by the algorithm described by Poppinga *et al.* [150] allowed us to extract smaller planar patches (the downsampling is not required) and to achieve higher performances. The algorithm (see Algorithm 2) proceeds by taking a point P_1 and two neighbors P_2, P_3 from the point cloud PC. They form an initial PP Π for which we estimate the plane equation. We try to extend this PP by considering neighboring points. The `seedList` contains the points that can be considered as a seed to explore nearby points. For each neighboring point P_{neigh}, we compute its distance d_{neigh} to the plane estimated from the points of Π. Our noise model gives us an upper bound DistanceThreshold of the thickness of a plane located at a given distance from the camera. We consider this upper bound to define the maximum distance allowed between Π. Now, suppose the distance d_{neigh} is smaller than DistanceThreshold, we add P_{neigh} to the `seedList` and Π, and estimate again the plane equation associated to Π. When the PP cannot be extended (all the points of `seedList` have been considered), it is added to the list of PPs `lPP`. The points of this list will never be considered again by the algorithm. A segmented range image I is generated from the list of the PPs: each pixel of I corresponding to a plane Π_i is assigned with a label (i). For performance reasons, in our implementation, the image I also plays the role of the variable `lPP`, allowing fast verifications for the membership test of a point P_{neigh} to an existing PP. We then perform a filtering of the image I to remove isolated points. In a post-processing step, we compute the rectangular boundary, of each extracted PP, filter out outlier points with the RANSAC algorithm and refine the estimated plane with an LS optimization on the inlier 3D points.

4.4.2.4 Planar patch association and fusion

In our case, we use the camera poses provided by the Tango TDK as pose prior. We remind that the provided VIO algorithm running on the device does not use the depth data to estimate the camera pose and is subject to small estimation errors, which may accumulate. For these reasons, the planar patches do not perfectly overlap when using this pose prior.

Algorithm 2: Region growing PP extraction.

input : Point cloud PC
output: Segmented image I

IPP $\leftarrow \varnothing$;
foreach *point* P_1 *of* PC **do**
  select points P_2, P_3 in $(PC \setminus \text{IPP}) \cap \text{neighbourhood}(P)$;
  $\Pi \leftarrow \{P_1, P_2, P_3\}$;
  DistanceThreshold $\leftarrow$ `NoiseModel` (Pi);
  `UpdatePlaneEstimation` (Π);
  SeedList $\leftarrow \{P_1, P_2, P_3\}$;
  foreach *point* P_{seed} *of* SeedList **do**
    foreach *point* P_{neigh} *of neighbourhood*$(P) \setminus (\Pi \cup \text{IPP})$ **do**
      if $dist(\Pi, P_{neigh}) <$ DistanceThreshold **then**
        $\Pi \leftarrow \Pi \cup P_{neigh}$;
        `UpdatePlaneEstimation` (Π);
        DistanceThreshold $\leftarrow$ `NoiseModel` (Pi);
        SeedList $\leftarrow$ SeedList $\cup P_{neigh}$;
      end
    end
  end
  IPP $\leftarrow$ IPP $\cup \Pi$;
end
We create the image I *from* IPP *by assigning a label* i *to the pixels of* I *corresponding to points of the plane* Π_i *of* IPP;
`// Removal of isolated points of` I
foreach *point* P *of* I **do**
  $\Pi \leftarrow$ plane associated with P NbPointSamePlane $\leftarrow 0$;
  foreach *point* P' *of neighbourhood*(P) **do**
    if $P' \in \Pi$ **then**
      NbPointSamePlane $\leftarrow$ NbPointSamePlane $+1$;
    end
  end
  if NbPointSamePlane $< \max(|neighbourhood(P)|/2, 1)$ **then**
    remove P' from Π;
  end
end

For the *association function*, we first compare the normal angle differences and the PPs offset distances to define candidate pairs of associated patches. We defined our PPs offset distances as the mean of the distances between the points of the rectangular boundary of one PP and the infinite plane of the other PP. We validate the pairs with an overlapping test between the rectangle boundaries.

After two planar patches have been associated, we need to create a new planar patch combining the two. For disambiguation purpose, from now on, we will call unit-PPs the PPs obtained after extraction, and fused-PPs, the PPs obtained after the fusion process and corresponding to the union of several unit-PPs.

Similarly to [21, 107], we wanted to avoid storing the 3D points associated with the planar patches. We use the Principal component analysis (PCA) algorithm to estimate the infinite plane equation of the fused-PPs. We use [107, Equations (7-8), (18)] to compute the fused patch covariance matrix and the fused patch centroid from their respective covariance matrices and centroids. We then project the boundary of the two associated PPs on the new estimated plane and compute the rectangular boundary covering the two previous ones.

After this first fusion process, we consider the updated fused-PP of the model π_i^m and check whether their fusion (which increased the rectangular boundary) can lead to new associations. This may occur, *e.g.*, during a loop closure where a patch π_j^c is merged with a patch π_{i1}^m and the updated

model patch overlaps with another model patch π^m_{i2}. We loop until no further association can be found.

The unit-PPs that compose a fused-PP may not be perfectly aligned. It is the role of the global optimization component to improve their alignments, at the end of the scan. For this reason, we need to keep track of the unit-PPs by storing their covariance matrices, their centroids and camera pose matrix. During the scan, the global optimization performed by the Tango SDK may also correct previous camera poses, which leads to an update of the associated covariances matrices and centroids.

4.4.2.5 Planar registration

In this paragraph, we explain why, contrary to the planar approaches of the literature mentioned earlier, we did not opt for a planar registration component. To cope with the problem of pose ambiguity mentioned earlier, the most robust optimization approach is to consider jointly PPs and photometry as in [46, 199, 117]. We could not intervene on the Tango VIO algorithm, which is closed source. The solution would have been to develop a new odometry algorithm, and optimize it for mobile platforms, which is a long process and beyond the scope of this thesis. There was also the risk to perform redundant work with Google, and could logically attempt to improve its odometry algorithm with the use of the depth sensor. For all these reasons, we attempted to patch the existing algorithm with a new component built upon the Tango VIO algorithm.

We started with a test of Pathak *et al*. [145] planar registration with a frame-to-frame approach we applied to systematically correct at each frame the camera poses from the Tango VIO algorithm. The choice of the frame-to-frame approach over a frame-to-model one was to enhance the potential limitations of the solution. Figure 4.27 shows that significant drifts of the Tango algorithm are correctly corrected, and in the overall, the wall looks thinner. However, the right wall lost its flatness, meaning that poses accurately estimated at that moment were degraded by the planar registration algorithm. In fact, it is not surprising the camera pose estimated with a noisy point cloud from a 160×80 resolution depth sensor be less accurate than the pose computed with a 1280×720 resolution camera during optimal condition (presence of texture and low motion blur). There are many possible sources of errors: limitations of the depth sensing technology, limitations of the range imaging hardware, calibration, *etc*... which lead to inaccuracy of the range measurements. On the registration side, we found the main source of errors was the plane extraction. Under-segmentation of the range image (*e.g*. when there are adjacent similar planar surfaces which are seen as a single PP) or over-segmentation (*e.g*. when a nonplanar surface is seen as a planar one) lead to inaccuracies and spurious PPs respectively. Also, the use of a rectangle boundary cannot correctly model concaves planar surfaces or with holes (*e.g*. because of the presence of a TV on the wall, an opened window, ...) and can affect the overlap test of the association function. As the number of planar features is quite low in a given frame, the association function cannot take advantage of consensus approached to filter incorrect matches.

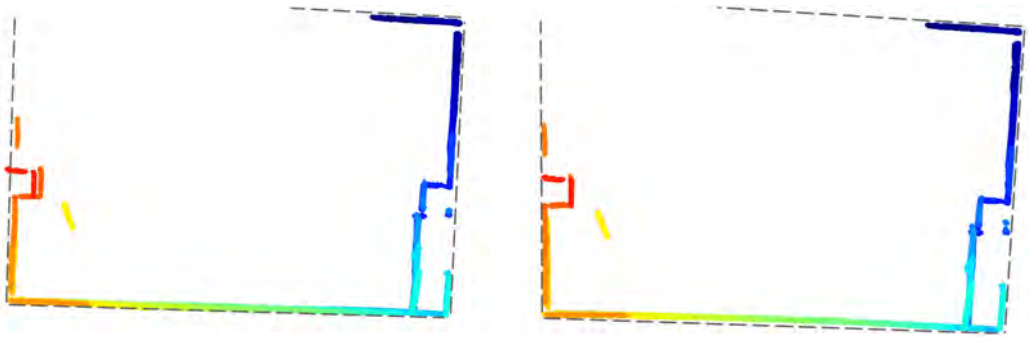


Figure 4.27: Left: Top down view of the point cloud of a room partially scanned displayed with a jet colormap (the first frame corresponds to the blue points, while the last frame corresponds the red points). The camera poses come from Tango VIO algorithm. The effect of the VIO drift is visible on a pillar on the left of the image. Right: point cloud after correction of the poses via a frame-to-frame planar registration algorithm. The duplication effect of the pillar is corrected and the walls appear thinner, but the right wall appears more slanted, deforming the room. We added gray dashed line to highlight the effects of the deformations.

Clearly, a frame-to-model approach (compared to frame-to-frame) would exhibit straight flat walls and would reduce the drift, but still, registration errors can occur and assuming the following regis-

trations are correct, it would generate a deformed PPs model. Probably, a better registration-patch strategy would be not to correct the Tango VIO poses when the images are enough textured, and perform planar registration otherwise or when a defect of PPs alignment is detected. But in both cases, these corrections would not globally fix the model and would be difficult to merge with the pose corrections performed by the Tango global map optimization.

4.4.2.6 Planar global optimization

To this point, our system can build incrementally a PPs model which is updated when a new PP is detected or fused with the model, or when a loop closure is detected. We have seen earlier that Tango global map optimization algorithm could fail to generate a map coherent in term of room structure (see Figure 4.20). The objective is to correct globally the room structure (and consequently all the camera poses), so that walls which appeared duplicated, become thin planar surfaces.

Problem simplification. As we seen in Section 3.2.2 on page 23, the estimation of the gravity vector from the IMU is quite accurate. It implies the pitch and roll angles provided by the Tango VIO should be as well accurate, which we verified experimentally: no drift nor offset was observed, and the vertical structures remained vertical. However, the yaw (heading) estimation of the camera pose and its position could be incorrect, as illustrated in Figure 4.20 and Figure 4.27. For the use case of the generation of a 2D floor plan, a top-down representation of the scene is considered: the planes corresponding to the walls can be represented by line segments. An error on the z component of the camera position does not affect the final result. For all these reasons, we considered a 2D simplification of the traditional planar-map optimization problem, a 2D approach where only the heading, the x and y components of the camera position would be globally optimized. In comparison with the planar approaches described in Section 4.4.1.4, the PPs measurements are here seen as a set of line segments s_i, and the 3D camera poses are seen as 2D poses $\boldsymbol{\xi}$, initialized with the poses provided by the Tango vSLAM algorithm. We want to find 2D corrected poses $\boldsymbol{\xi}$ and supporting lines $\mathbf{l}$ of the segments, so that the application of the corrected poses to the unit-PPs improves their alignment within the fused-PPs they contribute to.

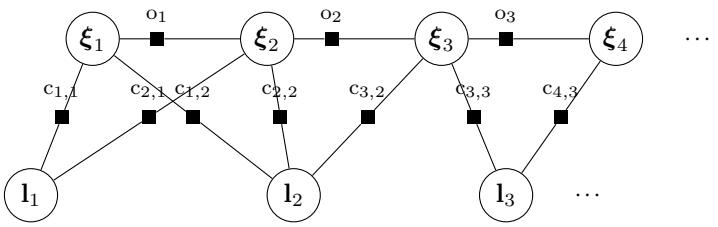


Figure 4.28: Factor graph of our planar LS SLAM approach. The nodes $\boldsymbol{\xi}$ and $\mathbf{l}$ represent the corrected poses and the support lines. The factor nodes relate to the odometry measurements o_k and the plane measurements $c_{k,i}$ (seen as line measurements).

LS SLAM formulation We chose a *LS SLAM* (see Section 4.4.1.4) to solve this problem. The *factor graph* representation illustrated in Figure 4.28 shows the relations between our variable and factor nodes. There are two types of *variables nodes* in our *state vector* Θ: the support lines $\mathbf{l}$ (the lines supporting the line segments) and the corrected poses $\boldsymbol{\xi}$, hence, the variables to be optimized are $\Theta = (\mathbf{l}_1, \ldots, \mathbf{l}_M, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N)$. We have two types of constraints connecting the variables nodes: the *odometry factors* and the *support line factors*. The poses computed by the Tango vSLAM define the odometry factors o_k associated with the measurements $z_k^o \in Z^o$ and their measurement prediction is $\widehat{z_k^o} = h_k^o(\Theta)$. The observation of a planar patch (seen as a line $\mathbf{l}_i$) from the pose $\boldsymbol{\xi}_k$ defines a *support line* factor $c_{k,i}$, associated with the measurement $z_{k,i}^c \in Z^c$. Its predicted measurement is $\widehat{z_{k,i}^c} = h_{k,i}^c(\Theta)$

We want to minimize the difference between the predicted measurements ($\widehat{z^o}$, $\widehat{z^c}$) and the real measurements (z^o, z^c), which translates into Equation (4.7) (see Section 4.4.1.4 for the details), where

N_o and N_c correspond to the number of odometry measurements and PPs observations respectively.

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} \mathbb{P}(\Theta \,|Z^o, Z^c) = \dots \\ &= \arg\min_{\Theta} -\log \left(\prod_{k=1}^{N_o} \mathbb{P}(z_k^o|\,\Theta) \prod_{\substack{1\le k\le N_o \\ 1\le i\le N_c \\ z_{k,i}^c \in Z^c}} \mathbb{P}(z_{k,i}^c|\,\Theta) \right)\end{aligned} \tag{4.7}$$

Odometry measurement model. A difference of pose measurement $\boldsymbol{\xi}$ is given by the change of 2D camera orientation ψ and the 2D translation $[t_x t_y]$. The function q^ξ (see Equation (4.8)) maps this minimal parametrization with a matrix representation.

$$\begin{aligned}\mathrm{q}^\xi \colon \mathbb{R}^3 &\longrightarrow \mathcal{M}_{2,3}(\mathbb{R}) \\ \boldsymbol{\xi} = \begin{pmatrix}\psi \\ t_x \\ t_y\end{pmatrix} &\longmapsto \begin{pmatrix}\cos\psi & -\sin\psi & t_x \\ \sin\psi & \cos\psi & t_y\end{pmatrix}\end{aligned} \tag{4.8}$$

We model the odometry measurement error with a normal distribution $\mathcal{N}(0,\, \Omega^{o\,-1})$. In the absence of reliable information about the odometry uncertainty provided by the Tango SDK, Ω^o is constant and defined as follows:

$$\Omega^o = \begin{pmatrix}\frac{1}{\sigma_\psi^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_t^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_t^2}\end{pmatrix}$$

with $\sigma_\psi^2 = 0.0013$ and $\sigma_t^2 = 0.0016$ defined experimentally.

We obtain:

$$\mathbb{P}(z_k^o|\,\Theta) = \exp\left(-\frac{1}{2}||h_k^o(\Theta) - z_k^o||^2_{\Omega_k^o}\right) \tag{4.9}$$

where z_k^o is the odometry measurement associated to the frame k defined as follows:

$$z_k^o = (\mathrm{q}^\xi)^{-1}\left(\mathrm{q}^\xi(\boldsymbol{\xi}_k^{Tango})^{-1}\,\mathrm{q}^\xi(\boldsymbol{\xi}_{k+1}^{Tango})\right) \tag{4.10}$$

and $\boldsymbol{\xi}_k^{Tango}$ denotes the $\mathrm{k}^{\mathrm{ith}}$ pose provided by the Tango vSLAM component. The prediction of an odometry measurement is defined as:

$$h_k^o(\Theta) = h^o(\boldsymbol{\xi}_k, \boldsymbol{\xi}_{k+1}) = (\mathrm{q}^\xi)^{-1}\left(\mathrm{q}^\xi(\boldsymbol{\xi}_k)^{-1}\,\mathrm{q}^\xi(\boldsymbol{\xi}_{k+1})\right) \tag{4.11}$$

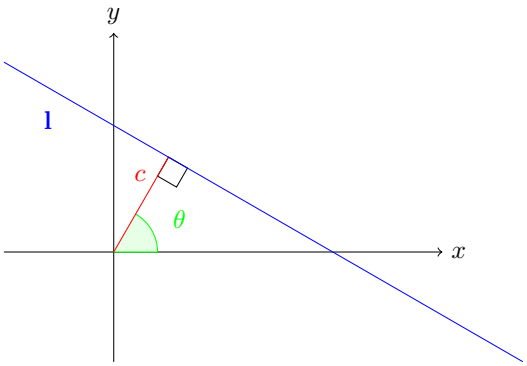
Lines measurement model. A line measurement is given by its orientation θ and its distance c to the origin, as illustrated in Figure 4.29. We model the measurement error with a normal distribution $\mathcal{N}(0,\, \Omega^{c\,-1})$. Similarly to [107], Ω^c depends on the covariance matrix of the points of the associated unit-PP: σ_c is the line width uncertainty, and σ_θ the normal orientation uncertainty, where $\sigma_c = \sqrt{\lambda_2}$, $\sigma_\theta = \arctan(\sigma_c/\sqrt{\lambda_1})$ and $\lambda_1 > \lambda_2 > 0$ are the eigenvalues of the aforementioned covariance matrix. The difference with [107] is that we do not use this uncertainty model to perform registration or plane association but for our measurement model. Figure 4.29b shows a graphical representation of the uncertainty values in the form of a rectangular box of width $2 * \sigma_c^2$ and length $2 * \sqrt{\lambda_1}$. This parametrization has the advantage of being minimal, which enables to define a full rank information matrix as follows:

$$\mathrm{l} = \begin{pmatrix}\theta \\ c\end{pmatrix} \quad \Omega^c = \begin{pmatrix}\frac{1}{\sigma_\theta^2} & 0 \\ 0 & \frac{1}{\sigma_c^2}\end{pmatrix} \tag{4.12}$$

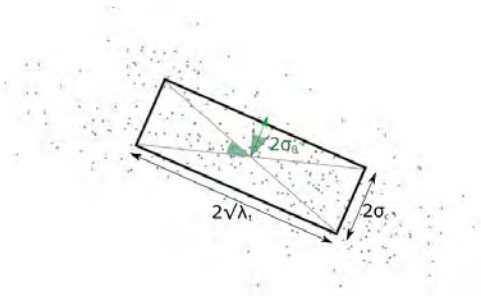
We can transform a line l into a Cartesian representation with the function q^l defined by:

$$\begin{aligned}\mathrm{q}^\mathrm{l} \colon (-\pi, \pi] \times \mathbb{R}^+ &\to \mathbb{R}^3 \\ \mathrm{l} = \begin{pmatrix}\theta \\ c\end{pmatrix} &\mapsto \begin{pmatrix}\cos\theta \\ \sin\theta \\ -c\end{pmatrix}\end{aligned} \tag{4.13}$$

It follows the line $\mathbf{l}$ can be described as: $\{(x, y) \in \mathbb{R}^2 \,|(x, y, 1)\, \mathrm{q}^{\mathrm{l}}(\mathbf{l}) = 0\}$



(a) Polar representation of a line $\mathbf{l}$ with the coefficients θ and c.



(b) Uncertainty model of a line computed from the covariance of the associated point cloud. λ_1, λ_2 are the eigenvalues of the covariance matrix and define an *uncertainty box* of the line segment. σ_θ is the angular uncertainty of the normal.

Figure 4.29: Line parametrization and uncertainty model.

We obtain:

$$\mathbb{P}(z^{\mathrm{c}}_{k,i}|\,\Theta) = \exp\left(-\frac{1}{2}||h^{\mathrm{c}}_{k,i}(\Theta) - z^{\mathrm{c}}_{k,i}||^2_{\Omega^{\mathrm{c}}_{k,i}}\right) \tag{4.14}$$

where the measurement $z^{\mathrm{c}}_{k,i}$ is the line $l_{k,i}$ (in polar coordinates) corresponding to the i$^{\text{th}}$ PP extracted at the frame k and the prediction of a line support measurement is defined as:

$$h^{\mathrm{c}}_{k,i}(\Theta) = h^{\mathrm{c}}(\boldsymbol{\xi}_k, \mathbf{l}_i) = (\mathrm{q}^{\mathrm{l}})^{\text{-}1}\left(\mathrm{q}^{\xi}(\boldsymbol{\xi}_k)^{\text{-}\mathsf{T}}\,\mathrm{q}^{\mathrm{l}}(\mathbf{l}_i)\right) \tag{4.15}$$

Solution. Combining Equations (4.16) to (4.15), we obtain the final non linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{k=1}^{N_{\mathrm{o}}} ||\widehat{z^{\mathrm{o}}}_k - z^{\mathrm{o}}_k||^2_{\Omega^{\mathrm{o}}_k} + \sum_{\substack{1\leq k\leq N_{\mathrm{o}} \\ 1\leq i\leq N_{\mathrm{c}} \\ z^{\mathrm{c}}_{k,i}\in Z^{\mathrm{c}}}} ||\widehat{z^{\mathrm{c}}}_{k,i} - z^{\mathrm{c}}_{k,i}||^2_{\Omega^{\mathrm{c}}_{k,i}} \tag{4.16}$$

Considering all the frames would be unpractical in term of performances. We select a subset of the frames (keyframes) by comparing the rotational and translational distance of the current frame with the previous keyframe.

We compute a solution of Equation (4.16) with the Scipy implementation of the Broyden-Fletcher-Goldfarb-Shanno algorithm. This suboptimal approach takes 7 min to find the minimum of a problem involving 43 keyframes. In comparison, specialized C++ libraries such as [88, 102] performing local linearization and taking advantage of the sparsity of the linearized equations can solve problems of similar size in a few seconds.

Figure 4.30 illustrates our results on a trivial problem made of two PPs. As expected, the minimum corresponds to camera poses which enable to align the uncertainty boxes of the associated unit-PPs. When the scene contains more constraints, such as Figure 4.31, all the uncertainty boxes cannot be perfectly aligned, but a compromise can be found. For this scene, we can observe the dark blue uncertainty boxes had the biggest width uncertainty, their alignment in the solution is not as good as the other boxes. Section 4.A presents additional results on acquisitions with the Tango TDK.

4.4.3 Planar approaches conclusion

In this section, we saw traditional RGB-D VO and global map optimization algorithms could fail to faithfully reconstruct indoor scenes. and it was relevant to consider PP features since they encode the



Figure 4.30: Left: Initial PP model generated from two frames, before optimization. The light and dark blue line segments correspond to the fused-PPs. Each fused-PP is made of two unit-PPs, represented with black *uncertainty boxes* (defined in Figure 4.29b). Right: After global optimization, the *uncertainty boxes* of the unit-PPs have a better alignment.

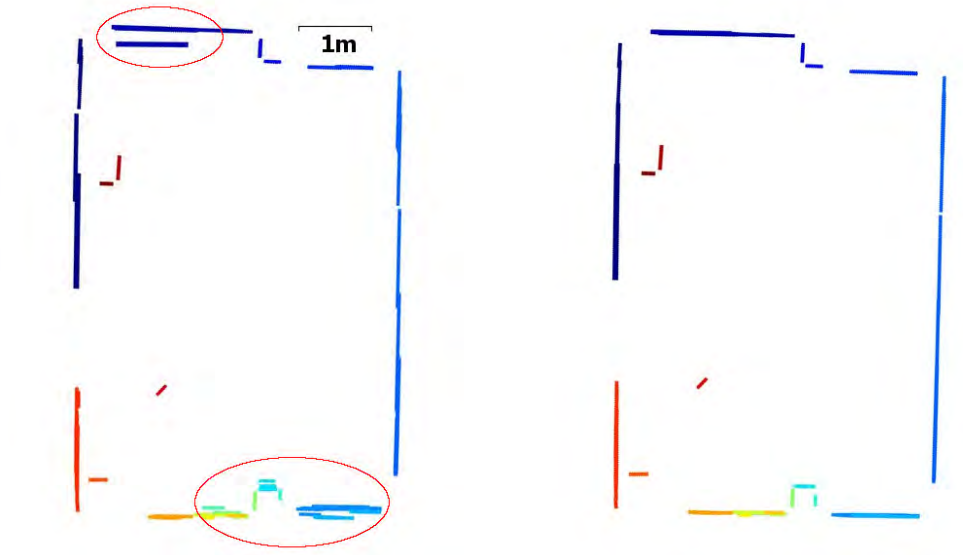


Figure 4.31: Left: initial planar uncertainty rectangles. Two uncertainty boxes are displayed with the same color if there are associated with the same fused-PP. The red ellipsis illustrates the regions where the odometry drift caused a notable duplication of the wall structure. Right: *uncertainty boxes* (defined in Figure 4.29b) drawn after the correction from our planar LS SLAM. To illustrate the enhancement of our method, we had disabled the Tango global map optimization.

layout of the room, they enable large-baseline registration, and offer a memory efficient data structure. We studied their classical components: PPs extraction, association, fusion, registration, global optimization and proposed a solution to implement them on a Tango device. The originality of our approach comes from the adaptation with existing Tango components, and the use of an uncertainty model into our global optimization component. Related works and our results demonstrate such approach can be successfully considered to improve the coherency of the reconstruction.

Unlike previous works, we would like to highlight the cons and pitfalls of planar approaches. Because of depth data noise, planar registration performs local corrections which can lack accuracy in the short term. Figure 4.27 illustrates this problem. On the long term, it is more interesting as planes are large features which encompass multiple measurements. The redundancy of the observations can be considered to compensate the uncertainties of the measurements, with the help of an uncertainty model. Besides the noise of the range measurements, which can affect the PPs extraction, the scene itself may contain planarity defects and small reliefs, such as a thin painting on a wall, wall cladding, *etc*. This issue is not addressed in the literature nor in our works. It is a difficult problem since a slight

curvature can hardly be observed locally, and an observation of the entire planar surface may also fail to sense the curvature, depending on the measurement noise which increases with the distance. It can lead to incorrect extracted planes and an incorrect registration. The registration can be strongly affected by incorrectly matched PP. As the number of planar surfaces present in a frame is generally low, the association function cannot use consensus algorithms to improve the robustness of the matches and the camera pose may not be fully constrained. As regards global optimization, the problem of incorrect plane associations can be mitigated with the help of robust LS SLAM approaches such as [197] or a line process as in [35]. The use of local image features such as [46, 199, 164] or global approaches such as [117, 80] can solve the problem of lack of features and missing camera pose constraints. The joint optimization on PPs and image information enables to be robust to the lack of structure or texture, similarly to hybrid VO approaches seen previously. One reproach to our work is we did not perform such a joint optimization. The reason is we wanted to take advantage of Tango vSLAM which runs very quickly on the Tango TDK, instead of wasting CPU resources computing a second time images descriptors (which are not accessible in the public Tango API). Our work can be seen as a patch over the Tango platform. This was a safe decision considering we had little information about Project Tango roadmap and we wanted to avoid investing a lot of time on works that could become redundant with future Tango functionalities.

To conclude, planar approaches reveal to be successfully improving the reconstruction results of indoor scenes. While it appears intuitive to use it for such scenes, the aforementioned issues should be taken into consideration. Their use is more suited to global optimization than local optimization (*e.g.* VO). Other challenging scenarios can be taken into account, such slightly deformed planar surfaces or curvy walls which have not been addressed in the literature yet.

4.5 Conclusion

In this chapter, we have tackled the problems related to localization with RGB-D sensors mentioned in Chapter 2:

Sensor limitations and mainly range measurements noise has an important impact on the accuracy of the reconstructed model, the extraction of PPs and the registration algorithms giving high importance to depth data such as depth-based approaches. The use of sensor models enable to predict the accuracy of the measurements and uncertainty models can help the fusion of data.

Real-time localization is very desirable for our purpose, to display an augmented view of the scene and to keep computational resources for the processing of the range measurements. It is the role of the front-end localization component, which relies on a VO algorithm to provide the pose of the camera at the highest possible frame rate. Optimizing a VO algorithm without sacrificing the accuracy is a difficult task and at the moment, few VO algorithms can handle VGA frames at camera frame rate.

Robust localization is critical since the position of the range measurements are relative to the camera pose. An incorrect estimation of the device motion can lead to incorrect reconstruction, *e.g.* duplicated walls, holes, ... Indoor scenes may contain little texture, have large flat structures, varying illuminations, which is challenging for VO algorithms. We have seen in Section 4.3 that VO approaches using RGB data suffered less from drift, and that hybrid approaches obtained in overall the best results. We saw that front-end localization components were not expected to provide a globally coherent map and trajectory. It is the role of the back-end, which runs at a slower pace, to perform a global optimization of the map and the trajectory.

Planar approaches suffer from various problems, such as the lack of simultaneous planar surface in a frame and the ambiguities during the extraction of PPs. Such approach is more interesting for back-end component rather than front-end since PPs are large features which encompass multiple measurements and it is more relevant to ensure the global consistency of the planes during loop-closing to avoid the double walls effects. Despite these limitations, we demonstrated it was possible to obtain successful results where the walls, the ceilings, and floors remain flat.

Adaptation with existing technologies One of the main issues that we had to face was working with such a cutting-edge technology, not yet mature to be available to the general public. Since the SDKs was closed source, it was difficult, if not impossible, to have full control of all the sensors and their data and, more in general, what happened "under the hood" of the system. For this reason, we chose to build upon what was provided by the API and the SDKs, rather than proposing our own full solution that would have required access to data and hardware that was not available.

Related publications. The benchmark of the different VO algorithms has been published in Journal of Real-Time Image Processing: [6] Vincent Angladon *et al*. "An evaluation of real-time RGB-D visual odometry algorithms on mobile devices". In: Journal of Real-Time Image Processing (2017), pp. 1–18, Springer Verlag, in press.

4.A Additional planar LS SLAM results

This section shows additional results of our planar LS SLAM approach presented in Section 4.4.2.6.

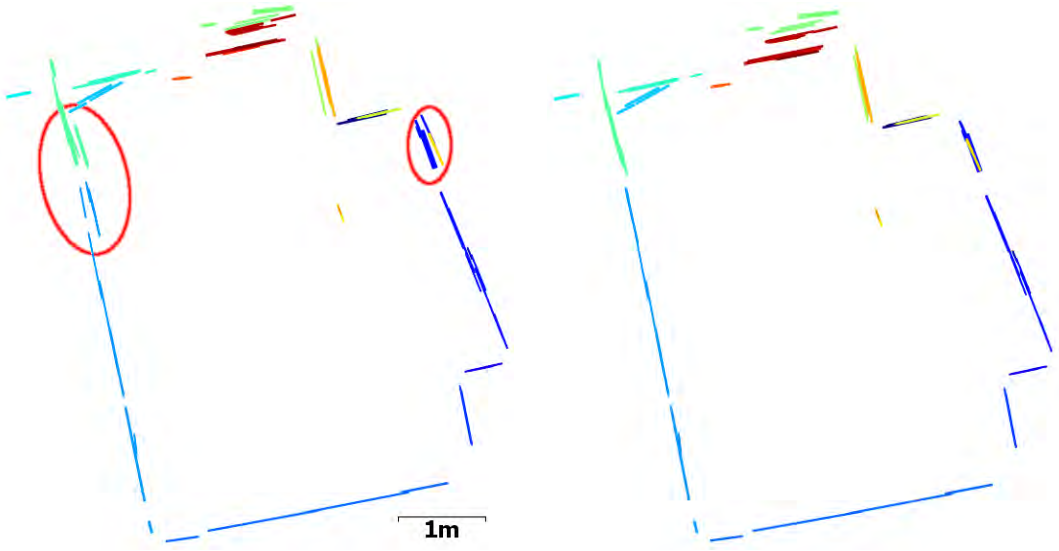


Figure 4.32: Left: initial planar uncertainty rectangles. Two uncertainty boxes are displayed with the same color if there are associated with the same fused-PP. The red ellipses illustrate the regions where the odometry drift caused a notable duplication of the wall structure. Right: *uncertainty boxes* drawn after the correction with our approach.



Figure 4.33: Left: initial planar uncertainty rectangles. Right: *uncertainty boxes* drawn after the correction with our approach.

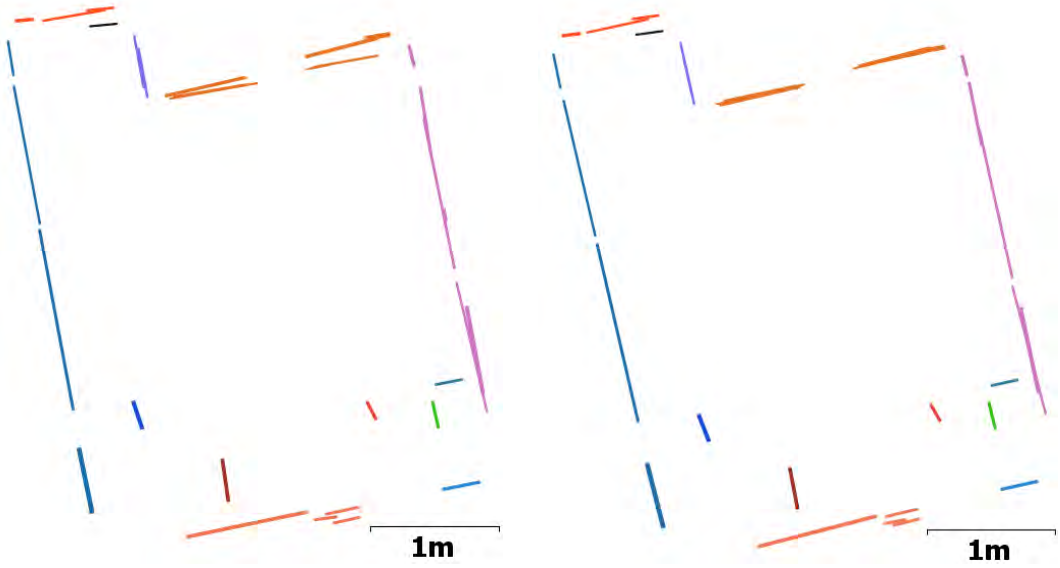


Figure 4.34: Left: initial planar uncertainty rectangles. Right: *uncertainty boxes* drawn after the correction with our approach.

CHAPTER 5

Room layout Estimation

Contents

5.1 Introduction . . . 86
5.2 Background . . . 86
5.2.1 Room layout estimation . . . 86
5.2.2 Scene Semantic . . . 86
5.2.3 User interaction . . . 87
5.3 Analysis of user driven approaches . . . 88
5.3.1 Point and Line selection . . . 88
5.3.2 Plane selection . . . 89
5.4 The proposed layout generation pipeline . . . 91
5.4.1 Visibility Polygon . . . 92
5.4.2 Layout Generation . . . 93
5.4.3 Wall labeling . . . 96
5.4.4 Implementation details . . . 97
5.5 Evaluation . . . 97
5.6 Conclusion . . . 99
5.A Distance Measurement Evaluation . . . 103
5.A.1 First Experiment . . . 103
5.A.2 Second Experiment . . . 103

5.1 Introduction

In the context of room layout estimation with a depth sensor, we have seen previously how to localize the device and detect the planes corresponding to the walls, the ceiling, and the floor. Now, we want to generate a floor plan or a 3D model of the observed scene. In addition, we would like to design an interactive approach that can run on a mobile device and is able to generate the model as the scan progresses, so that errors or missing data can be covered without do-overs. The main challenge of our problem is how to properly design a system that has the user in its processing loop: as discussed in Section 2.2.1 an efficient and effective interface is required to take into account the user inputs and improve the solution automatically generated by the system.

We remind our work hypotheses: we assume the considered rooms are made of a horizontal ceiling and floor and vertical planar walls, not necessarily orthogonal w.r.t. each other, *i.e.* weak Manhattan world assumption. They can contain clutter (furniture, movable objects, ...) occluding the walls. The considered hardware is the Tango TDK integrating a depth sensor with a working range of $4.2\,\mathrm{m}$, therefore our approach is limited to medium size rooms with a $5\,\mathrm{m}$ ceiling height maximum. The tablet natively supports motion tracking by means of a localization module, which requires the scene to contain sufficient texture, *i.e.* the walls of the ceiling or the floor can be of uniform color, but not all of them.

In this chapter, we first present a brief review on indoor scene layout estimation in Section 5.2, with a focus on two related topics: semantic analysis and user interactions. In Section 5.3 we review some existing approaches for user interactions in existing applications and some preliminary experiments we carried on to improve their user-driven scheme. The results of this experiments influenced some of the design choices for our approach described in Section 5.4. We validate the relevance of our approach in Section 5.5, with a comparison between different existing mobile applications, while Section 5.6 concludes the chapter with some remarks.

5.2 Background

5.2.1 Room layout estimation

One of the first approaches for generating a floor plan recovered the topology of the room by considering the adjacencies of the extracted segments [125, 206, 215], which works well for rooms with low clutter, and when the walls are well separated from the clutter. However, most of them are off-line methods that cannot be easily adapted for real-time applications on mobile devices. Murali *et al.* is the only approach considering data from a Project Tango device. They use the Tango Constructor application to scan the different rooms, and then process the final point cloud. The layout of the rooms is retrieved by searching for boxes among the extracted intersecting planes.

For scenes fulfilling the Manhattan World assumption, Xiao and Furukawa [214] propose a constructive solid geometry (CSG) approach for Manhattan World scenes where the extracted line segments are used to enumerate additive or negative rectangle candidates modeling the sliced point cloud.

In Stambler *et al.* [191], the room layout is the solution of an optimization problem that takes into account a room shape probability, the number of walls and openings (to penalize complex layouts) and a wall probability defined on each extracted surface patch.

The use of cell-complex is very popular for both single rooms [26, 152] and multiple rooms [124, 123, 139, 140] layout estimation. The line segments are replaced by infinite lines which partition the 2D space into polygonal cells. A graph connecting the adjacent cells is defined: instead of considering topological information on the segments, adjacency relationship between the cells is taken into account. The inside/outside label of the cells are computed with a graph-cut algorithm. This approach is robust to missing data because the extension of the segments is automatically considered. The edges separating differently labeled regions represent the room layout.

5.2.2 Scene Semantic

The segmentation of 3D primitives can be performed individually, *i.e.* considering each primitive independently, or globally, *i.e.* considering the adjacent primitives which are a contextual information.

The problem of clutter classification is often ignored as authors estimate that the retrieved boundary of a room consists only of walls [140, 152]. The planar patches included in this boundary are the support of the walls, the others form the clutter. Experiments proved this assumption to be sometimes wrong. For example, shutters behind a window or a wall behind an open door, create a protrusion in the expected boundary. Missing scan data may also lead to an incorrect boundary when an obstacle such as a wardrobe completely covers the vertical extent of a wall.

First works in the field of RGB-D semantic considered the RGB-D frames as input [11, 69, 172, 189], which raised the problem of fusing the labels between different frames with the global model of the reconstructed scene. This problem was tackled by [74, 118], in which a Bayesian fusion is proven to provide good results. In the following paragraphs, we consider only approaches working with 3D data, we summarize in Table 5.1.

Primitive planar approaches generally perform a classification of the planar patches by assigning them a label, whether they represent a wall, a clutter, a floor or a ceiling. In Mura *et al.* [124] and Murali *et al.* [125], walls correspond to patches with a height close to the ceiling-floor distance. In a later work, Mura [123] proposes a global approach with a graph of the adjacencies of the planar patches. The nature of the path in the graph between a considered planar patch and specific nodes (ceiling, floor, identified walls) defines its clutter/wall labeling. This method relies on the assumption that the data is almost complete and the adjacencies between the planar patches are correctly identified. Xiong *et al.* [215] propose also a global approach based on a classifier of the local features of the planar patches, combined with another classifier, which takes into account the adjacency relationships of the patches and the previous predictions. The features include the patch's orientation, the area of the bounding rectangle, its height, the point density and the aspect ratio.

Pre-segmented data approaches also consider large compact regions which are then classified. Kim *et al.* [96] consider a 3D voxel space where planar patches are detected as well as objects. The segmentation relies on a Conditional Random Field (CRF) approach on the voxels, in which label consistency is enforced on the planes and the detected objects. Anand *et al.* [2] over-segment the colored point cloud into segments with a smoothness constraint. Each segment is then classified taking into consideration individual features and object-object relations (*e.g.* a keyboard is very likely to lie on a table). Despite the absence of planar patches extraction, the walls are correctly segmented with the knowledge of the local curvatures in the features. Stambler and Huber [191] classify smooth surface patches (which are mainly planar) with a voting approach from an individual, per-point classification. The features take into consideration the height of the point, the curvature and information about the points located behind (relatively to its normal) the considered point.

Point cloud approaches directly consider the 3D points and they are now dominated by deep learning approaches such as [157]. While such approaches often give very promising results, because of their black-box nature it is difficult to verify whether they actually learn actual contextual (global) information or local features. Using point clouds, whether they are pre-segmented or not, offers the advantage of a finer segmentation and the identification of a large range of objects such as windows, doors, cupboards, beds, *etc*. A high semantic level approach can take more advantage of the contextual information. For example, identifying a doorknob could influence the segmentation of the neighboring points to the door class and thus easily consider the planar patch as clutter. This accuracy gain is not necessary for our needs and would imply a higher computational cost. Also, it would not spare us from incorrect labels to be corrected by the user.

5.2.3 User interaction

Automatic approaches for floor plan generation or scan-to-Building Information Modeling (BIM) can be prone to errors. These errors can come from missing data, clutter, sensor noise or some special architectural elements in the scene (*e.g.* small walls, large openings, cavities in the walls, obstacles fully covering a wall, *etc*.). Depending on the progress of the scan, it may not be possible for an algorithm to determine whether a planar patch corresponds to a wall or clutter.

The user interaction schemes proposed in the literature are usually corrective (the user intervenes the end to correct a proposed solution) or user-driven (no solution can be computed without user

Article	Input	Context	Learning
[124]	Planar patches	None	None
[123]	Planar patches	Adjacent planar patches	None
[215]	Planar patches	Adjacent planar patches	Stacked Generalization
[96]	3D Voxels, planar patches and objects	None	SVM
[2]	Over segmented point cloud	Inter-object relationship	SVM
[191]	3D Points and planar patches	None	Randomized Decision Forest
[157]	3D colored Points	Deep	Deep learning

Table 5.1: Comparison of several 3D segmentation approaches relevant to our works. We refer the reader to [32] for a more complete review on indoor scene understanding approaches.

interaction). Corrective schemes [9, 123, 148] are commonly considered for offline approaches, while user-driven schemes are mostly found in online approaches.

In offline approaches, the user has the possibility to interact at the end of the automatic process to perform final corrections. For example, [9] proposes a simplified 3D edition interface to let the user perform corrections, taking advantage of the estimated planar polygons and the 3D point cloud to during the creation of new planar polygons. Mura *et al.* [123] let the user transform a wall planar patch to a clutter one or vice-versa, and to cope with missing data by letting the user extend some planar patches. Pintore *et al.* [148] propose a user interface to edit the estimated room layout in the panoramic image space.

In the opposite direction, other approaches are completely user-driven and they can only generate the floor plan through the user inputs. Magic Plan [185], Tap Measure [137], Tango Measure [65], and [146, 166] propose an interaction scenario where the user manually captures wall corners at the floor level or the ceiling level. Similarly, [64] proposes an interaction where the user can select the planar patches of the walls he wants to keep. The order of this selection is crucial as it determines the connections between the walls (or the corners) and thus the topology of the layout. Missing one corner or one wall during the scanning may lead to an incorrect result.

5.3 Analysis of user driven approaches

In the following, we briefly review the main available solutions and some preliminary tests on user interactions we carried out. The objective is to highlight the choices made in term of user interface and interaction for reconstructing the room layout. This analysis determined some of the design choices for our application.

5.3.1 Point and Line selection

Point selection. With Magic Plan [185] (as well as Tango Measure [65] and TapMeasure [137]), the selected entity is a 3D point. It is difficult to accurately select a point with a simple finger touch on the screen device, as the device is held with the other hand and, possibly, moved around. A common solution considered by the three applications is to use a crosshair overlaid on the camera image in a fixed position so that the user need to aim at the corner rather than select it with the finger. The user has to adjust two rotations of the device (see Figure 5.1) to aim the crosshair to the point to select, which takes some time. Moreover, corners with the floor are often occluded by furniture, preventing from selecting the corner accurately. The applications Magic Plan and TapMeasure allow the user to create a corner when it is occulted, with an obvious trade-off on the accuracy.

Line selection. As an attempt to ease the interaction, we experimented the use of a vertical line crosshair to select wall corners. This vertical line is parallel to the gravity vector and represents the wall corner to select. Figure 5.2 illustrates this interaction: the user has to adjust only one rotation (around the gravity center) to align the crosshair with the wall corner to select. The main advantage is that the full vertical extent of a wall corner is less likely completely occluded by furniture.

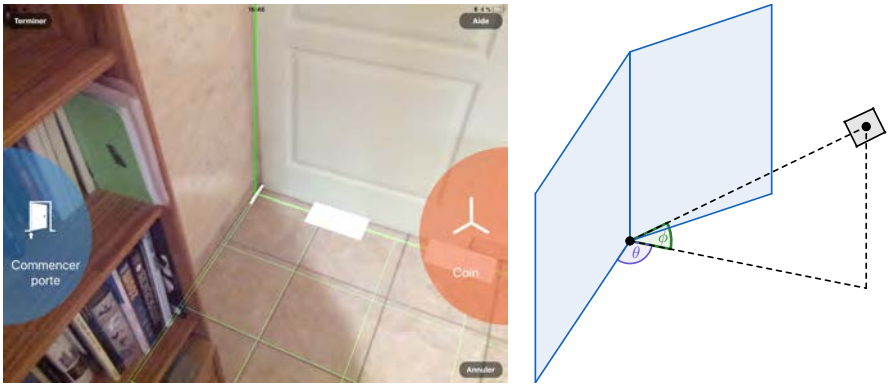


Figure 5.1: Selection of a 3D point with a crosshair with Magic Plan, the user has to adjust two rotations: θ and ϕ.

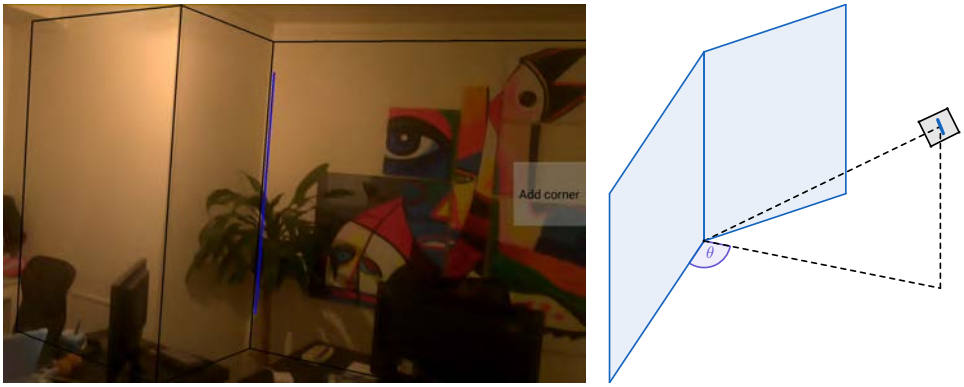


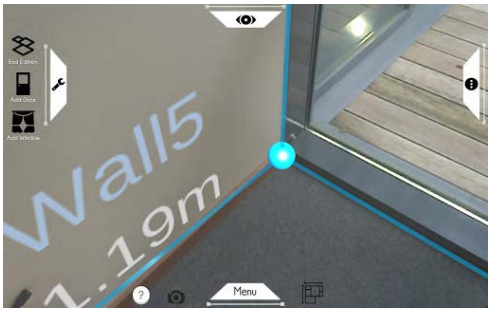
Figure 5.2: Selection of a 3D line with a crosshair with our own prototype, the user has to adjust only one rotation: θ.

Accuracy improvements. The point and line crosshair selection methods can be affected by a potential involuntary shaking of the user's hands. To improve the accuracy of the pointing interaction, several solutions have been proposed. The *Freeze-Set-Go* interaction [105], freezes the camera view after the selection of the entity. The user can improve the entity position on the still image, moving the device at its convenience, then the AR mode is restored. This technique was chosen by Easy Build to correct the position of the room corners, as depicted in Figure 5.3a.

The *Snap-To-Feature* interaction [104] consists in snapping the user selection to salient elements of the image: the user input is used as a rough guess for the region where the entity can be found. For example, the Tango Measure application optionally proposes to detect the edges and planes around the crosshair by processing the 3D point cloud. Upon detection, the crosshair is displaced on the detected element, as shown in Figure 5.3b. Unfortunately, sometimes the detected edge is not the desired one, which makes the experience very frustrating for the user.

5.3.2 Plane selection

The manual selection of the planes corresponding to walls can be performed in a more efficient fashion, in terms of user interaction. Planes are indeed large entities and they do not require an accurate user input to be selected. The application FloorPlanEx [64] lets the user add a wall with a touch input anywhere on the camera image and it then estimates the corresponding plane equation by considering the 3D points around the touched area. A crosshair can eventually be drawn, as in Figure 5.4a to notify the user when there are sufficient 3D data to estimate a plane and how it would look like. This approach is more sensitive to the sensor position w.r.t. to the wall, though. If the device is too close to the wall, a small part of it is observed, and the estimated plane may lack accuracy (and it may later appear misaligned w.r.t. to the wall). If the device is too far from the wall or with a too skew orientation w.r.t. the wall, the depth sensor noise is higher, thus reducing the accuracy of the plane estimation.



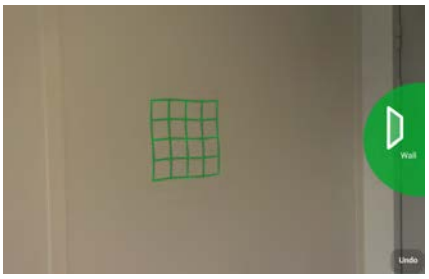
(a) *Freeze-Set-Go* interaction proposed by Easy Build to accurately position the room corners. Image courtesy of Wosomtech.



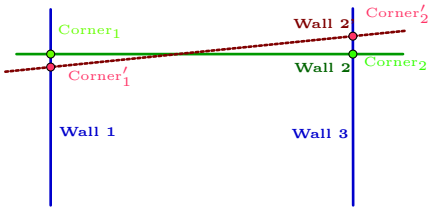
(b) *Snap-To-Feature* interaction proposed by Tango Measure. Here the application seems to propose the most salient edge in the green circle, which corresponds here to the edge between the wall and the skirting. The edge between the floor and the skirting cannot be snapped here because it is less salient, while we aim the device to this edge.

Figure 5.3: *Freeze-Set-Go* and *Snap-To-Feature* interactions.

Moreover, with this approach, the boundaries of the estimated plane are computed by intersecting the previously selected plane and the plane that will be selected next. This means that the boundaries of a plane will be shown only when the next plane is selected, which does not give the user a feedback about the quality of the estimated plane. Indeed, an incorrect plane estimation affects the estimation of the two adjacent wall corners, as illustrated in Figure 5.4b The longer is the wall, the higher will be the error obtained on the corner positions, whereas with the two previous methods, this error does not depend on the wall dimensions, as illustrated in Figure 5.5.



(a) Plane selection with Magic Plan for Project Tango devices.



(b) An incorrect estimation of the plane of a wall (Wall2') affects the position of the two adjacent corners: Corner$'_1$ and Corner$'_2$, computed as the intersection of Wall2' with Wall1 and Wall3.

Figure 5.4: The plane selection interaction.

Incorrect user interactions. With the applications FloorPlanEx, EasyBuild and Magic Plan for Google Tango, some particular cases of intuitive walls selections can lead to an incorrect room layout (see Figure 5.6). These cases can be detected when the intersection point of the consecutive infinite lines corresponding to the selected planes, is far outside the other points of the generated room layout. The aforementioned applications either display the incorrect layout, either remove the incorrectly selected plane.

Conclusion. We saw previously the easiest and most efficient interaction was the selection of planes, but the proposed applications were lacking accuracy because they were only considering a small part of the 3D data. Thereafter, we consider for our proposed approach, a plane selection method too. In the next section, we will improve the accuracy obtained with this interaction by considering fused-PPs as seen in Section 4.4.2.4. This solution enables to take advantage of additional 3D points, corresponding to the 3D data from several viewpoints, for which the full range image has been segmented into PPs.



(a) In this interaction, the user selects the positions of the corners of a $2.5\,\mathrm{m}$ long wall. We assume the uncertainty of the corner position is $3.5\,\mathrm{cm}$ in the 2D space. The black points correspond to random positions of the wall corners according to the considered accuracy. The blue lines (joining those corners) correspond to possible walls.

(b) In this interaction, the user selects the plane of a wall. We assume the (observed) plane is $2.5\,\mathrm{cm}$ thick and $1\,\mathrm{m}$ long. The black points correspond to the observed point cloud. The blue lines correspond to possible planes.

Figure 5.5: Wall selection accuracy for two interaction scheme. For both schemes, the endpoint accuracy at the extremities of the wall is around $3.5\,\mathrm{cm}$. Neglecting the VO drift impact, the endpoint accuracy is constant in the first case. Whereas, the accuracy decreases in the second case when the wall is longer or observed from a more distant position.



Figure 5.6: Example of incorrect selection of the walls, though it is intuitive to do this: PPs 2 and 3 correspond to the same plane. The generation of the layout performed by FloorPlanEx and Magic Plan for Google Tango will consider the intersection of the planes 2 and 3, which corresponds to an infinite point. This problem can, in fact, be detected. In the case the planes 2 and 3 are distinct, the correction of the layout (here presented with an orange line segment) requires to know their boundary.

Still, localization drift between viewpoints can reduce the expected accuracy gain, an issue which can be counterbalanced with a planar SLAM algorithm, as presented in Section 4.4.1.4.

The tested user-driven approaches require as many user inputs as there are walls in the scene. In order to be more efficient, our approach should be able to retrieve some walls without user input.

5.4 The proposed layout generation pipeline

From the analysis of the existing applications, it appears that the major drawbacks are (i) the number of interactions required by the user, and (ii) the order in which the user has to scan the room. We want to design an application that requires fewer interactions and would be fully automatic in a best-case scenario. We also aim at relaxing the constraint of scanning the room one wall after another in order to give the user more freedom of movement in the room: the user should be able to move freely and eventually to come back to some parts of the rooms that may need more refinement.

Our method is mainly based on two inputs, the Planar Patches (PPs) and the visibility polygon [19], *i.e.* the polygonal region representing the space explored by the device. The PPs are computed incrementally from the depth maps as discussed in Section 4.4.2.3 and they can be modified by the user interaction to change their classification into “wall” or “clutter”. The visibility polygon is built incrementally as well as the union of the camera frustums; it is used both as a visual feedback for the user to show the part of the room that has been covered and as first rough computation of the room

layout.

Figure 5.7 summarizes the proposed approach and the components of our pipeline. The Tango TDK middleware provides at each iteration the depth map of the scene, as well as the camera pose. The depth map is processed in order to extract sets PPs. Thanks to the known camera pose, the PPs are brought into a global world coordinate system, so that they can be associated and then fused with the existing patches of the model. Whenever the model is updated, the visibility polygon of the discovered area(s) is also updated (Section 5.4.1), which can be used as a visual feedback for the user. The visibility polygon is also used by the labeling module (Section 5.4.3) to automatically classify the planar patches as *wall* or *clutter*, thus enabling the disambiguation between actual walls and other objects that may be lying inside the room. This last task can take advantage of the interaction of the user, who can correct and change the automatic labeling of the vertical planar patches into wall or clutter. This interaction scheme allows obtaining a good repartition of the tasks between the user and the system: it lets to the system tasks it can reliably perform (localization, plane detection, room layout generation from a set of selected plane), and lets the user intervene on tasks the system cannot do well (estimating whether a PP corresponds to a wall or not). Finally, the layout generation module (Section 5.4.2) computes the room layout from the vertical planes with *wall* labels and the boundary given by the visibility polygon. When the labeling is correct, the user has no interaction to do, and all the PPs corresponding to planes are selected for him. In the absence of the labeling module, this approach would only relax the constraint of scanning the room one wall after another (thanks to the layout generation module) but would remain user-driven.

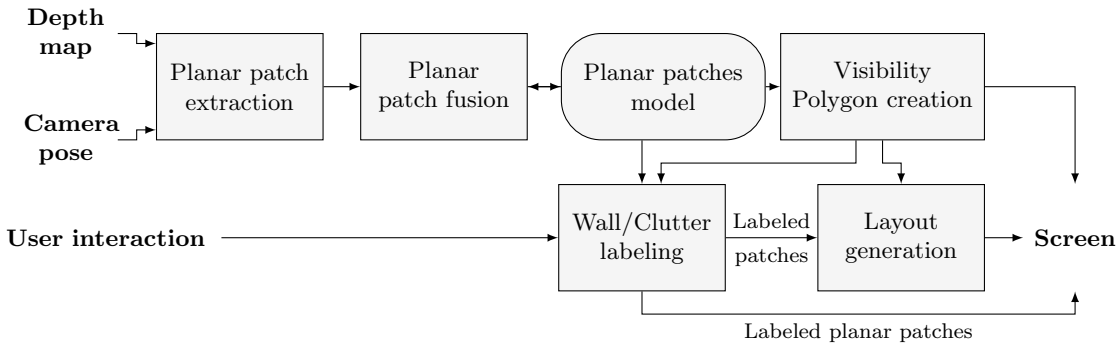


Figure 5.7: The pipeline of the our room layout generation algorithm.

5.4.1 Visibility Polygon

During a scan, the user needs to have first visual feedback showing a rough representation of the explored space. In existing applications, the discovered area is represented by an occupancy grid [50], or by displaying the progress of the reconstruction such as in Canvas.io [134] and Tango Constructor. Occupancy grids are not very appealing because the grid is not necessarily aligned with the walls and despite the use of thresholding techniques, they have a low robustness to noisy data which may create floating areas. Visibility polygons [19] have been considered by Zhang *et al.* [225] and the Tango Floor Plan Example2 application [63] (which is quite recent and posterior to our work). They represent the area observed by the camera in the form of a polygon as the result of the union of all camera frustums. They can also be used to represent approximately the room layout. In Tango Floor Plan Example2, this polygon is computed from the reconstructed mesh, while in [225] they use a polygon clipping approach.

Construction of the visibility polygon. We consider the 2D space corresponding to a top-down view of the scene and we denote s the line segment obtained from the projection of a vertical PP on a horizontal plane.

At a given frame, the viewing polygon is a polygon representing all the area observed by the camera. Assuming there is no unobservable 3D data such as glass, the edges of this polygon would be contributions from the segments s or from the camera view frustum. A simple and intuitive way to

compute the visibility polygon would be to compute the union of all the viewing polygons, for all the considered frame. As the PPs model is updated, the visibility polygon needs to integrate the data and get updated as well. To this end, we compute the visibility polygon from the segments s corresponding to the vertical PPs of the model and all the camera positions that observed s. We consider the visibility polygon P^s associated with each line segment s, which is the union of all the triangles t_s formed by the camera position and the two extremities of s. Each triangle is made of one wall segment and two frustum segments. An example of polygon P^s associated with a segment s observed from multiple frames is depicted on the left of Figure 5.8. After each update of the vertical planar patch, s is updated. We then update the vertices of P^s which were located on s. The geometric union of all the polygons P^s forms the visibility polygon, as illustrated on the right of Figure 5.8. We implemented the computation of the visibility polygon with the help of the geometry engine GEOS [142].

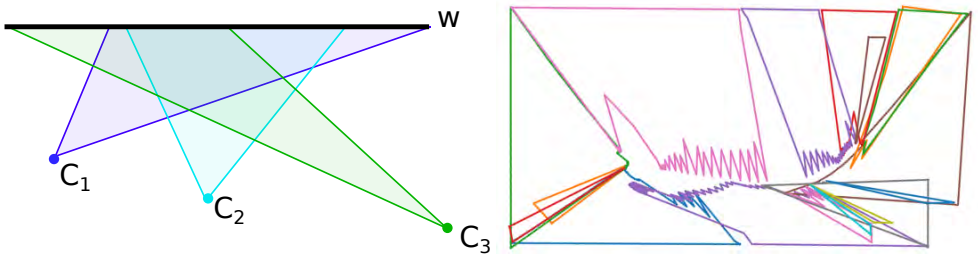


Figure 5.8: Left: the three blue, cyan and green triangles t_s represent the part of the camera frustum viewing s and associated with the camera poses c_1, c_2 and c_3 respectively. Their union form P^s: the visibility view polygon associated with the line segment s. Right: the visibility polygon is the union of the visibility view polygons P^s, here represented with different colors.

Use of the visibility polygon for rough layout estimation. The visibility polygon is also interesting because it can give a preliminary, rough overview of the shape of the room being scanned, *e.g.* by applying a polygon simplification algorithm such as the Ramer–Douglas–Peucker one [47] or the Visvalingam-Whyatt one [207]. In Visvalingam-Whyatt algorithm, each vertex is associated a triangle formed by its adjacent vertices. Vertices with the smallest area are removed (leading to a change of the adjacent triangles areas), until a threshold on the number of remaining vertices or area removed is reached. This method highly depends on the sampling, *i.e.* the repartition of the vertices on the boundary of the polygon, since a high sampling will produce smaller triangles, which will be removed first.

Ramer–Douglas–Peucker algorithm is a curve simplification algorithm: the results depend on the point of the polygon boundary selected as both first and last point of the curve, which will not be removed. It considers the Hausdorff distance to assess whether the simplification of a part of the curve into a segment is possible. This operation is repeated on the curve, in a recursive fashion. This method is less sensitive to the density of sampling.

These simplification algorithms can give suitable results for scenes with low clutter, but provide unsatisfactory results for other scenes, as illustrated in Figure 5.9. In addition, they do not let space for user interaction, except changing a simplification threshold, which in some case, can lead to both under and oversimplified areas. For this reason, we do not use this method.

5.4.2 Layout Generation

The layout generation module takes as input the visibility polygon and the PPs model with their wall/clutter labels assigned either by the user or by the labeling module. The output of the module is a room layout obtained by taking into account these two inputs. In this module, the PPs of the model are seen as 2D segments, *i.e.* their projections on the plane.

We modeled our problem as a 2D geometric graph problem: each node corresponds to an extremity of a segment and nodes belonging to a 2D segment are linked by an edge [19]. We first build a geometric graph for the PP segment and a geometric graph for the boundary of the visibility polygon. Then we compute the *overlay* of these two graphs [19], *i.e.* we compute a new (undirected) graph that combines the information in the two graphs. Figure 5.10 shows a simple example of the two graphs

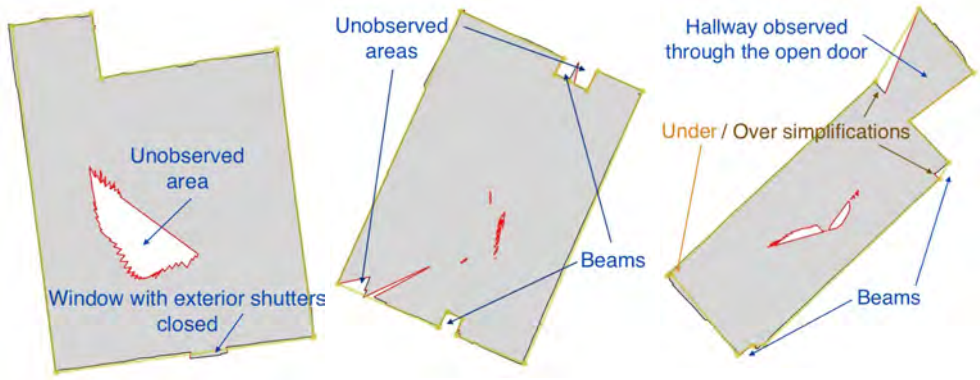


Figure 5.9: Two examples showing the simplification of the visibility polygon with the Visvalingam-Whyatt [207] algorithm can produce either a satisfactory layout (on the left) or an incorrect layout (on the right). The visibility polygon is displayed in gray, with frustum lines displayed in red. The layout is displayed in yellow.

and the computed overlay. Given this new representation, computing the layout corresponds to the problem of finding a cycle in the overlay graph.

In general, because of the topology and the incorrect user interactions, there may not be uniqueness of the solution, as illustrated for example in Figure 5.11. There are multiple criteria that can be considered to select the cycle to retain: the geometric area or perimeter of the polygon drawn by the path, the number of nodes, *etc*. In our case, we consider the number of segments selected or not by the user. We want to maximize the number of user-selected segments first, and then, for two paths having the same number of user-selected segments, we want to minimize the number of non-user-selected segments. The segments outside the path are discarded, whereas the others correspond to the (virtually) selected PPs. The adjacencies of the segments correspond to their order of appearance in the path, and thus to the order of plane selection seen in the interaction scheme described in Section 5.3.2. The room layout is estimated as in Section 5.3.2, considering the retrieved ordered selection of the PPs: the boundaries of each wall is computed by intersecting the previously selected plane and the plane that will be selected next. In case of incorrect configuration between two PPs, as in Figure 5.5, we connect their adjacent extremities (corresponding to the orange line segment in the figure).

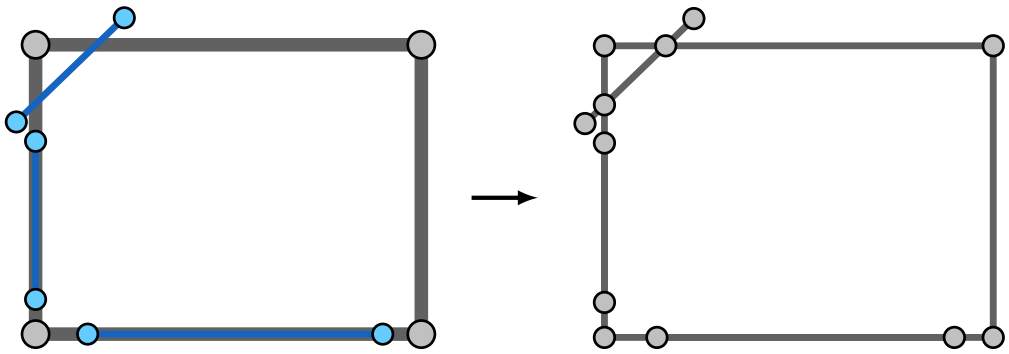


Figure 5.10: Toy example of the computation of the overlay of two graphs: the figure on the left shows the geometric graph of the visibility polygon boundary in gray and the geometric graph of the 2D PP segments in blue. The figure on the right shows the final overlay graph, which is an undirected graph.

Solving and discussion. Generating the room layout correspond, in the graph domain, to the problem of finding a path in the graph corresponding to a cycle. In general, our graph can have multiple cycles. Enumerating all the cycles of a graph has an exponential complexity, as demonstrated by Bax [15]. Instead, we try to create a direct acyclic graph, where the solution path can be found in linear time (w.r.t. the number of nodes and edges). For that purpose, we modify the overlay graph and we as-

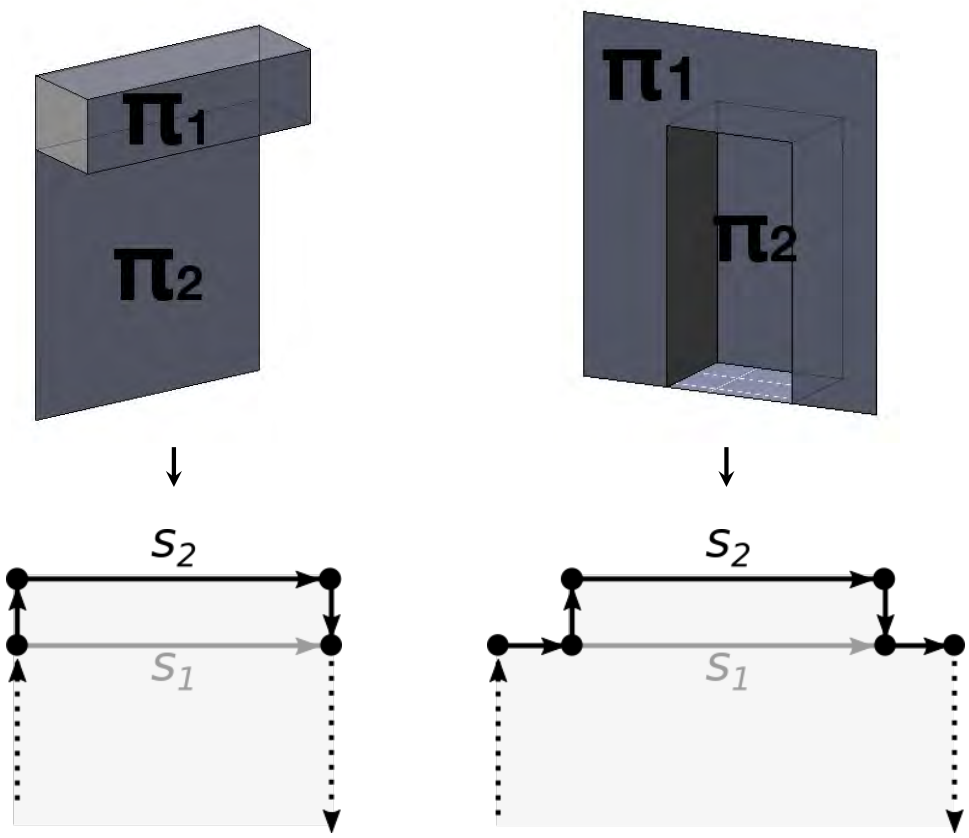


Figure 5.11: Two examples of scenes and user selections where there is no unique solution. Let's assume the user selected both the planes π_1 and π_2 (incorrect interaction from the user) displayed in the first row. Their respective segments in the top-down view (second row) are s_1 and s_2. The obtained layout with our approach will discard s_2 because the paths traversing this segment contain more non-user-selected segments. A correct interaction would have been the selection of π_1 or π_2, instead of both.

sign a direction to the edges corresponding to the segments of the visibility polygon boundary[1]: these directed edges are set to follow a clockwise orientation along the boundary of the visibility polygon. All the other edges are left undirected edges.

In order to find our minimal cycle, we assign to each node a pair (u_n, c_n) containing the number of distinct user-selected u_n and non-user-selected c_n traversed segments. The pair is initialized to zero. As we perform a traversal of all the edges of the graph we update the value of the pair as it follows. For each user selected PP corresponding to a traversed edge joining a node n to n', we update (u'_n, c'_n) as follows:

$$(u'_n, c'_n) = \begin{cases} (u_n + 1, c_n) & \text{if } u_n + 1 > u'_n \text{ or } (u_n + 1 = u'_n \text{ and } c'_n > c_n) \\ (u'_n, c'_n) & \text{otherwise} \end{cases} \tag{5.1}$$

And for each non-user selected PP corresponding to a traversed edge joining a node n to n', we update (u'_n, c'_n) as follows:

$$(u'_n, c'_n) = \begin{cases} (u_n, c_n + 1) & \text{if } u_n > u'_n \text{ or } (u_n = u'_n \text{ and } c'_n > c_n + 1) \\ (u'_n, c'_n) & \text{otherwise} \end{cases} \tag{5.2}$$

This way, a node n has its values u_n, c_n updated when we found a better path to join it. In order to retrieve the optimal path at the end of the graph traversal, we store for each updated node, the

[1]Note that such edges, in general, correspond to a partition of the segments of the boundary: the overlay computation can indeed split the edges corresponding to segments of the boundary if the latter contain or are crossed by 2D PP segments. In the example of Figure 5.10 we can see that the left vertical edge of the boundary is split into 4 edges in the overlay graph, as it contains one PP segment and is crossed by another PP segment.

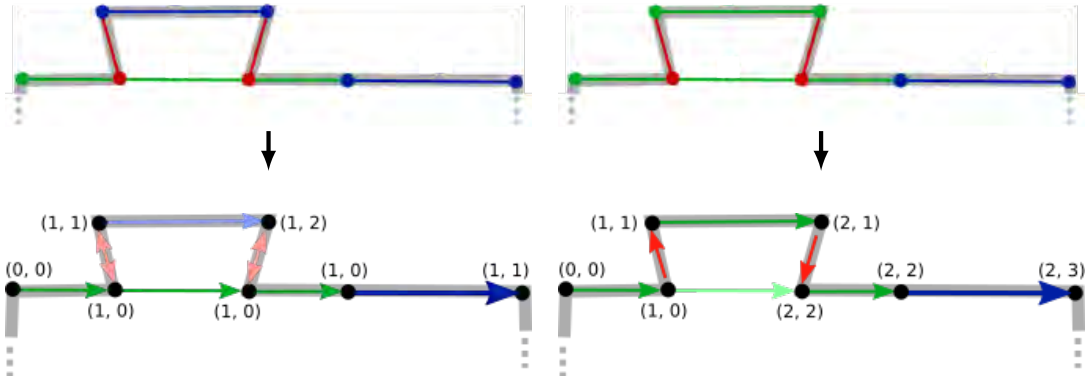


Figure 5.12: Two toy examples, one for each column, illustrating the creation of the directed acyclic graph from the overlay graph. We depict in green the edges corresponding to PP segments selected by the user, in blue the others PP segments, the visibility polygon is displayed in gray, with frustum edges in red.
First row: in the two cases three PPs are detected, with one user-selected segment in the first case and two in the second.
Second row: after a breadth-first traversal of the graph, the generated directed acyclic graph with the pairs (u_n, c_n) displayed for each node. The edges not corresponding to the optimal path are represented with faded colors.

previously traversed node. The final layout is estimated by considering the segments traversed by the optimal path and intersecting their respective infinite lines as in Section 5.3.

5.4.3 Wall labeling

Clutter often consists of irregular shapes, such as plants, sofas, *etc*., which are easily discarded as we consider only planar primitives. It can also consist of piecewise planar shapes such as cupboards, radiators, *etc*., we would like to detect. To position our work with the literature presented in Section 5.2.2, we clarify our classification requirements. The classification should follow an online approach with a low computational cost. Consequently, we avoided considering the *Point cloud approaches* and *Pre-segmented data approaches*. These methods provide a fine grain segmentation with numerous classes we estimated to be unnecessary to our use case. This accuracy gain would imply a higher computational cost and would not spare us from incorrect labels to be corrected by the user. Instead, we naturally chose a *primitive planar approach*, trading accuracy for efficiency. Each PP is classified independently with a machine-learning approach.

We hand-selected the following features to form the feature vector describing our PPs:

- the distance d^c between the highest point of the vertical planar patch and the estimated ceiling;
- the distance d^f between the lowest point of the vertical planar patch and the estimated floor: *i.e*. intuitively, a planar patch close to the ceiling and floor is likely to be a wall;
- the (horizontal) length l of the segment s, *i.e*. longer segments are likely to be walls;
- the distance d^v between a segment s and the exterior boundary $\partial\,\mathrm{P^v}$ of the visibility polygon $\mathrm{P^v}$, defined as $d^v = \sup_{p\in \mathrm{s}} d(p, \partial\,\mathrm{P^v})$, *i.e*. the maximum distance of its extremities from the boundary: the farther an extremity is from the boundary, more likely the PPs corresponds to clutter.

We compute a wall probability $\mathbb{P}(\mathrm{s})$ for each segment s of the model with a Multi-layer Perceptron classifier using one hidden layer and a logistic sigmoid activation. This choice was performed experimentally after comparing the prediction of several classifiers with different parameters. This classifier was trained beforehand against 900 manually labeled vertical planar regions from our training dataset. When a new PP is detected, we compute its corresponding $\mathbb{P}(\mathrm{s})$. This probability is updated when the PP is modified or when the ceiling/floor estimation changed. The user can, at any moment, change the PP label by touching it in the augmented view. Any label set by the user will remain as it is. Our implementation based on the Python Scikit-learn module can label 50 planar patches in $29\,\mathrm{ms}$.

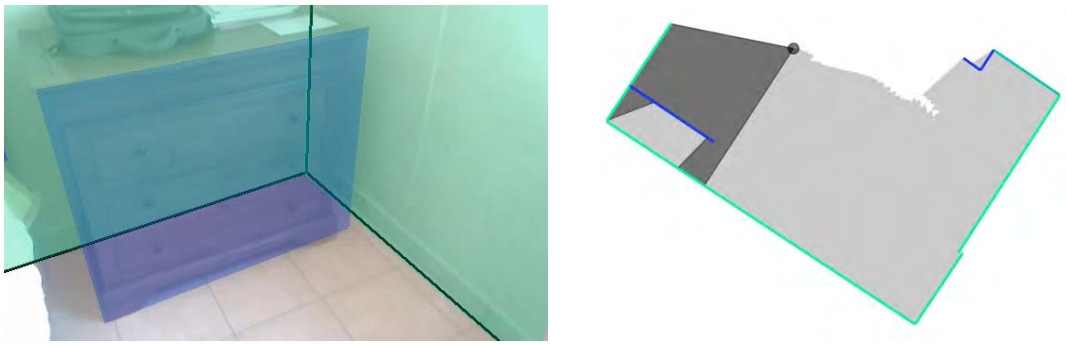


Figure 5.13: Visualization of the scan progress of the scene House2. We represented with a green-to-blue color scale the wall probability of the PPs. A high wall probability will be displayed in green, and in blue for a low probability. Left: augmented view of the device camera with the detected and classified planar patches. The estimated room layout is displayed with black lines. Right: visibility polygon in light gray, camera view polygon in black.

5.4.4 Implementation details

Our Python implementation of this last version takes $200\,\mathrm{ms}$ to generate the graph, and $11\,\mathrm{ms}$ to compute the optimal path and generate the layout. We consider improving the speed of the graph creation step by recycling the graph created during the union operations performed to compute the visibility polygon.

For a satisfying user experience, the layout of previously seen areas should not change when the user visits a new part of the scene. When the ceiling and the floor are detected, our method is suitable for incremental changes of the model: the wall probability $\mathbb{P}(\mathrm{s})$ of the previously observed segments s does not change, which means the computed path restricted to the previously seen segments is the same and has the same cost.

Visualization and interaction We propose a collaborative interaction scheme where the interaction is optional and can be performed at any time. Figure 5.13 shows the two views of our interface. As the user moves to capture new parts of the scene, the first view (Figure 5.13, left) displays the camera image augmented with the fused-PPs colored relatively to their wall probability and the estimated layout. The second view (Figure 5.13, right) provides a top view of the scene with the visibility polygon and the estimated layout too. The two views are updated at the frame rate of the depth sensor, giving an immediate feedback to the user who can decide to visit the area (s) with missing data. Displaying the fused-PPs enables the user to have a first feedback on the layout being built and it possibly can help her or him to move closer to the walls to improve the reconstruction. A longer observation of a wall allows it to be represented with a larger PP having extremities close to the wall corners, to gather more 3D points, which helps to improve the estimation of the plane and the wall probability.

While the areas of the walls close to the ceiling are usually less likely to contain clutter, we do not force the user to observe these areas. This approach is less comfortable for the user as he has to keep the device aimed towards the ceiling. Moreover, the ceilings are usually poorly textured (*e.g.* uniform painting), which may affect the reliability of the Google Tango VIO and lead to a poor localization accuracy. Nevertheless, our approach can in general handle this kind capturing scheme.

5.5 Evaluation

In this section, we compare the geometric accuracy of the room floor plans generated with our approach, Magic Plan [185] (run on iPad Air 1, which does not support ARKit), TapMeasure [137] (run on an iPhone 6 with ARKit support) and the FloorPlanEx from Google [64]. At the time of writing, we only designed a Python desktop prototype which can replay a recorded scan or process live data transmitted by the application. The presented results were obtained by performing the interactions on replays of the scans.

Evaluation protocol. We considered five indoor scenes Lab1^{MW}, Lab2, House1^{MW}, House2^{MW}, and House3^{MW}, where MW denotes the scenes respecting the Manhattan World assumption. The ground truth room layouts of these scenes were created with a Bosh DLE 50 laser rangefinder. We preliminarily assessed the quality of the measurements of the Tango TDK w.r.t. the rangefinder in another experiment detailed in Section 5.A. We evaluated the geometry accuracy of the obtained layouts with the ground truth and the reproducibility of the measurements by repeating the measurements five times. Each estimated layout was aligned with the ground truth by computing the transformation which minimizes the distances between their corresponding vertices. The mean value of these distances defines our residual error. We also evaluated the user effort during the use of the considered mobile applications. Magic Plan, TapMeasure, and FloorPlanEx are user-driven applications where the user selects the walls and the corners, respectively. The number of interaction is equal to the number of corners (plus one for Magic Plan). For Magic Plan, we did not count the interactions performed to estimate the ceiling height and to calibrate the device-floor distance. For our approach, we evaluated the number labels corrections on the planar patches and the number of patches merging.

Results and analysis. Table 5.2 reports the quantitative evaluation for each approach, while Figure 5.15 and Figure 5.16 show a qualitative comparison of the obtained floor plans. As explained in Section 1.3.1.1, Magic Plan estimates distances from the device orientation instead of taking advantage of a localization module or a depth sensor. Consequently, even the best results of the application (min residual column) are generally less accurate than the results obtained with FloorPlanEx and our approach. Due to the amount of clutter, most of the corners were captured on the ceiling, which reduces the accuracy of the measurements according to the application recommendations. Magic Plan assumes the angle between two consecutive walls is either $90°$ or $45°$: for this reason, the results are unsatisfactory on the scene Lab2, which does not fulfill the MW assumption. We can also observe the residual increases with the area of the room, which is coherent when there is an error with the device height estimation.

TapMeasure takes advantage of ARKit for the localization, but it is not designed to use a depth sensor. Similarly to Magic Plan, the position of the corners is computed from the device orientation If the user does not move, the position of the corners can be more accurate than Magic Plan since the height of the device (w.r.t. the ground plane) is known at any moment. Contrary to Magic Plan, TapMeasure does not allow to add corners on the ceiling. If there are objects lying on the floor, the user has to select the position of the corner through the clutter, which inevitably affects the accuracy of the generated layout. The accuracy of the localization depends on the type of the scenes, and it affects the final results as well. For the scenes Lab1^{MW} and Lab2, we can speculate the tracking was working well since the layout generated by TapMeasure were more accurate than FloorPlanEx (including the min and max residual columns). In contrast, for the scene House2^{MW}, the tracking was quite poor since TapMeasure obtained the worst results (less accurate than Magic Plan). Figure 5.14 illustrates several tracking issues obtained during the evaluation.

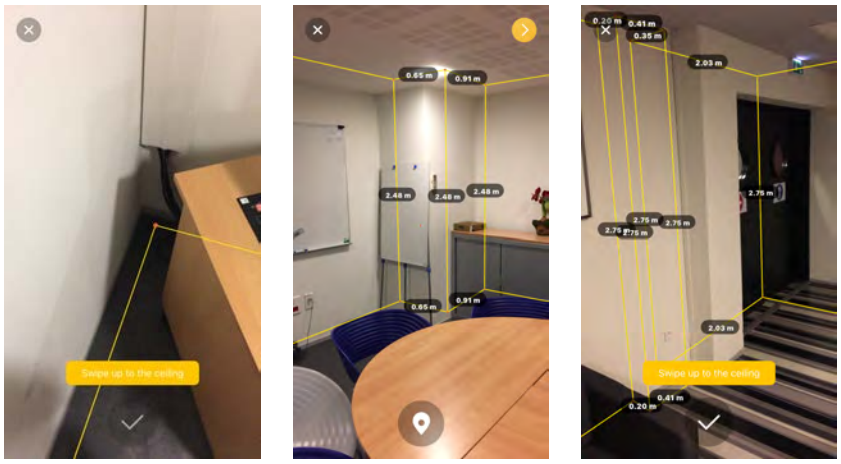


Figure 5.14: Three examples of ARKit localization drift during the use of TapMeasure.

For selling or renting a property in France, for example, the Alur law defines the maximum error of

the measured surface of the apartment to be less than 5% of the whole surface. The errors in estimating the area from the FloorPlanEx application and our approach are below or equal to this threshold, which may not be enough for some official uses. The results show that our method is generally more accurate and provides more repeatable results than the FloorPlanEx: we consider the 3D points from multiple frames to estimate the planes of the walls when the FloorPlanEx only considers the points from one frame.

The last column of Table 5.2 describes the degree of interaction in terms of the number of interactions required to complete the scan. It confirms our approach generally requires fewer screen interactions than user-driven approaches, except for the scene house3^{MW}, which contained a fireplace, high furniture, and many curtains. The Lab1^{MW} was also quite challenging because of the presence of a high cupboard and pillars which were incorrectly labeled as wall. In term of displacement effort, they are mandatory for FloorPlanEx and our approach, in order to capture 3D data with a reasonable noise in medium-sized rooms. Also, the movements of the user cannot be too fast with the Tango TDK in order to avoid localization issues. Magic Plan, instead, required the user to move toward the room center (in order to maximize the measurement accuracy) and to rotate on himself during the scan. This enables to scan simple scenes (star-shaped and moderate clutter) quite easily and fast, while complex scenes with a lot of clutter and occlusion are not handled well.

Scene	Method	Mean and Max area err.		Mean and σ residuals		Min and Max residuals		Nb Inter.
Lab1^{MW} ($25\,m^2$)	Ours	2.3%	4.2%	**29 mm**	**14 mm**	**14 mm**	**48 mm**	**3.25**
	FloorPlanEx	**2.2%**	4.5%	47 mm	26 mm	18 mm	84 mm	4
	TapMeasure	2.3%	**3.8%**	46 mm	24 mm	18 mm	76 mm	5
	Magic Plan	12%	17%	164 mm	52 mm	106 mm	231 mm	5
Lab2 ($47\,m^2$)	Ours	**1.1%**	**2.4%**	**38 mm**	**3 mm**	**35 mm**	**43 mm**	**0.75**
	FloorPlanEx	3.3%	4.3%	73 mm	19 mm	45 mm	100 mm	6
	TapMeasure	3.2%	6.6%	66 mm	15 mm	50 mm	86 mm	7
	Magic Plan	15%	24%	264 mm	65 mm	199 mm	329 mm	7
House1^{MW} ($11\,m^2$)	Ours	**1.8%**	**2.4%**	**30 mm**	12 mm	**14 mm**	**44 mm**	**0.5**
	FloorPlanEx	2.6%	4.5%	53 mm	**9 mm**	38 mm	60 mm	6
	TapMeasure	13.4%	23.8%	107 mm	60 mm	39 mm	184 mm	7
	Magic Plan	4.9%	8.8%	66 mm	18 mm	46 mm	87 mm	7
House2^{MW} ($13\,m^2$)	Ours	**3.0%**	5.0%	**29 mm**	12 mm	**17 mm**	**49 mm**	**0**
	FloorPlanEx	3.3%	**4.2%**	41 mm	**11 mm**	28 mm	58 mm	4
	TapMeasure	6.2%	7.9%	66 mm	**11 mm**	56 mm	82 mm	5
	Magic Plan	5.4%	9.2%	50 mm	23 mm	22 mm	89 mm	5
House3^{MW} ($48\,m^2$)	Ours	**1.9%**	**2.3%**	**32 mm**	**5 mm**	**27 mm**	**37 mm**	7
	FloorPlanEx	2.8%	4.0%	105 mm	23 mm	78 mm	144 mm	**6**
	TapMeasure	7.5%	13.7%	122 mm	42 mm	60 mm	177 mm	7
	Magic Plan	15.4%	24.9%	233 mm	79 mm	163 mm	344 mm	7

Table 5.2: Results of the geometry accuracy and reproducibility comparison experiment. Manhattan scenes have "MW" in their name. We use bold font to indicate the best result for each scene.

5.6 Conclusion

In this chapter, we proposed a brief review of room layout estimation methods in the literature as well as some recent mobile applications. This review revealed that the few current online approaches are mostly user-driven and require many user interactions to generate the final floor plan. Our proposed approach overcomes these limitations: the room model can be generated online, incrementally during the scan progress and the proposed interaction scheme is not user-driven. The user, indeed, can optionally collaborate with the system, by selecting or removing (previously selected) planes corresponding to walls. The system reduces the user effort by classifying the detected planes either as clutter or as

walls, and by retrieving the topology of the scene, a task which is let to the user in the user-driven approaches. We evaluated our method on a desktop computer, using scans recorded from a Project Tango mobile device. The comparison with other mobile applications demonstrates that, in general, we achieve higher accuracy. In term of efficiency, the number of user interactions depends on the complexity of the scenes, which may contain clutter data incorrectly classified as walls.

Related publications. An alternative approach to generate a simplified layout of the scene has been presented at the Multimedia Modeling (MMM 2018) conference: Vincent Angladon *et al*. "Room Floor Plan Generation on a Project Tango Device". In: Multimedia Modeling (MMM 2018), Part II, Lecture Notes in Computer Science 10705 proceedings, Springer Verlag, in press. [4]

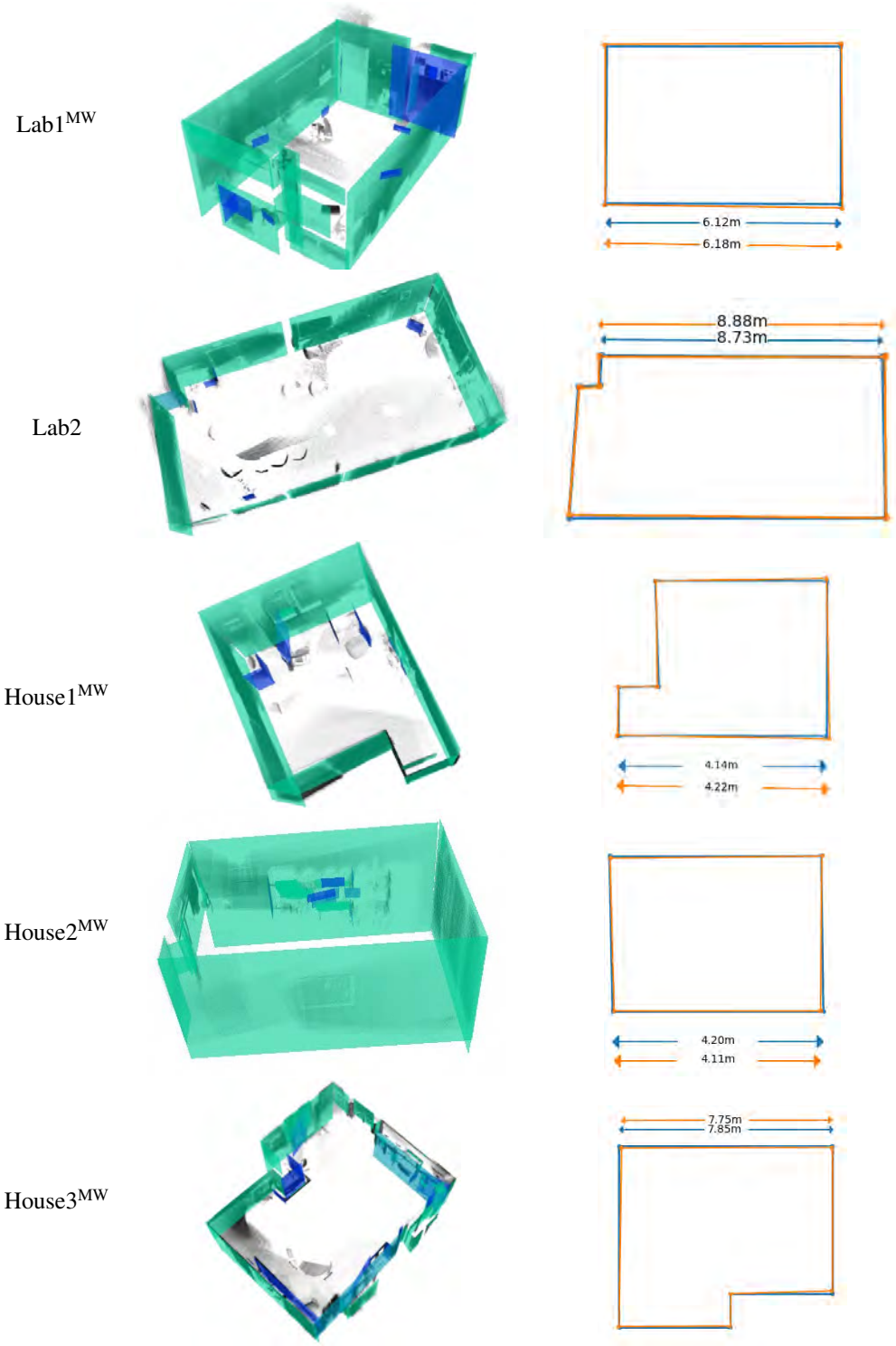


Figure 5.15: First column: the point clouds and the labeled planar patches (blue: clutter, green: walls) of the scanned rooms. Second column: comparison of our approach in orange with the ground truth in blue.

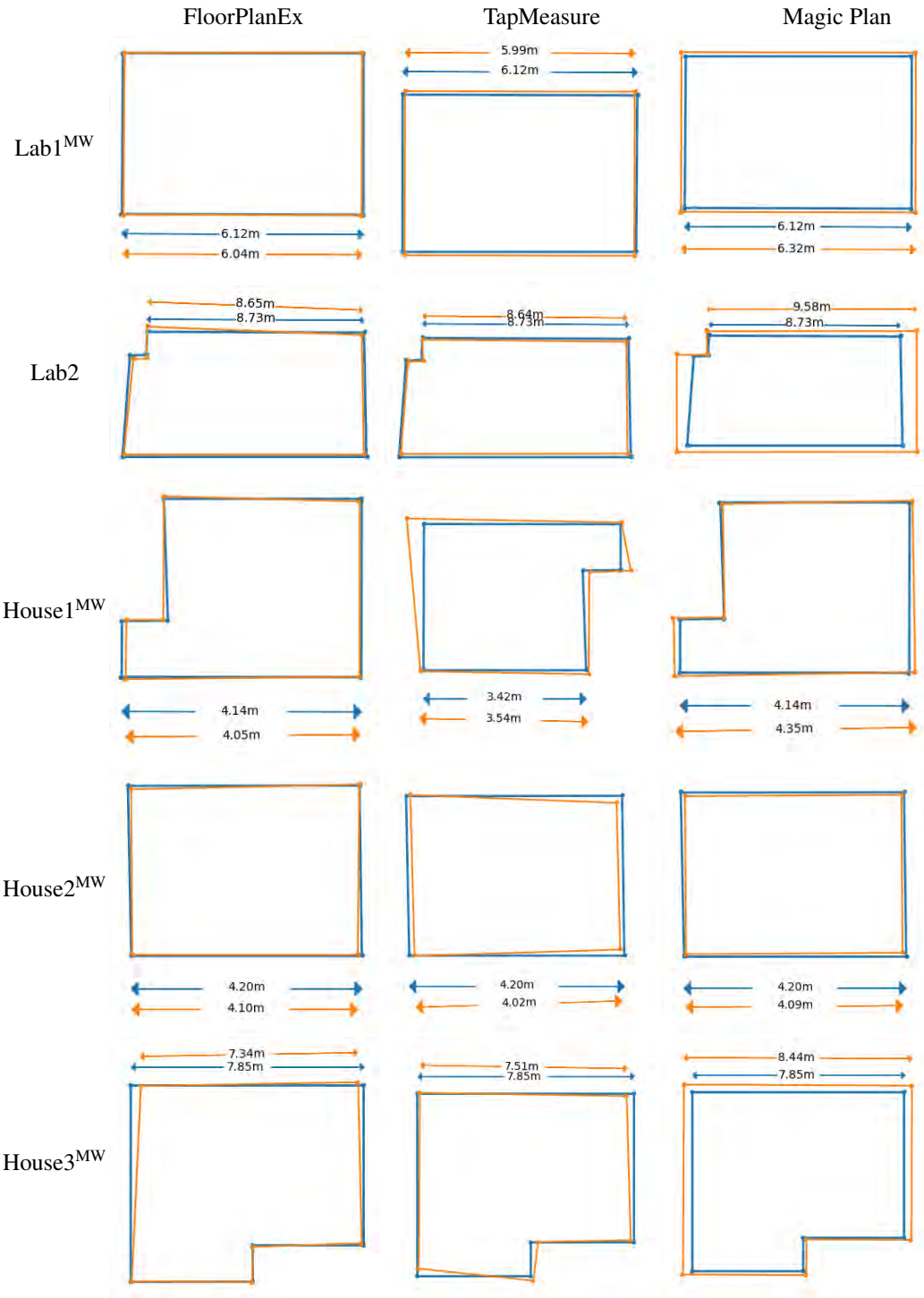


Figure 5.16: First row: comparison of the room layout obtained with FloorPlanEx in orange and the ground truth in blue. Second and third row: same comparison with TapMeasure and Magic Plan results.

5.A Distance Measurement Evaluation

In this section, we assess the accuracy of the distance measurements performed with a Tango TDK and with a Bosh DLE 50 laser rangefinder, which has been used for creating ground-truths for the room measurements evaluation.

5.A.1 First Experiment

Tango TDK has two main sources of inaccuracies: the depth sensor noise, and the drift performed by the provided localization module, which relies on a Visual Inertial Odometry (VIO) algorithm. There are two opposing strategies to perform a measurement with a Tango TDK: minimize the displacements of the tablet but have inaccurate depth perception over longer distances, or accept possible localization drifts in order to obtain more accurate depth measurements.

Protocol. We asked five candidates to perform the same distance measurement with the Measure App application [65] for Google Tango between two corners separated by a distance of $7.2\,\mathrm{m}$. With this application, the user has to point the tablet to the extremities of the object to measure. The position of the two 3D points of the measurement in the world space depends on the estimated camera position and the depth measurements. We instructed the candidates to perform a first measurement by minimizing their displacement, then we instructed them to move closer ($1.5\,\mathrm{m}$ approximately) to the anchor points of the measurement.

The depth measurement error is quadratic with the distance between the tablet to the measured object. During the first scenario, this distance was around $4.2\,\mathrm{m}$ and the inaccuracy of the depth sensor was estimated to $24 \pm 9\,\mathrm{mm}$ with a plane fitting test, whereas in the second scenario the distance was around $1.5\,\mathrm{m}$ with an estimated $5 \pm 1\,\mathrm{mm}$ depth error.

The VIO drift depends on the number and the position of the tracked keypoints in the fisheye camera frames: therefore textureless scenes and the motion blur may affect the motion estimation. The scene of the experiment was quite challenging for the VIO algorithm because the walls and the ceiling were textureless, the ground had a repetitive pattern and few furnitures were present. Also, when the user gets closer to an untextured wall, the VIO algorithm is more likely to drift.

Results. The standard deviations of the two measurements strategies were $81 \pm 25\,\mathrm{mm}$ and $39 \pm 25\,\mathrm{mm}$ respectively. The uncertainty comes from the truncation at $1\,\mathrm{in}$ of the measurements provided by the application. The results demonstrate that higher accuracy can be obtained with the second strategy, which means the average VIO algorithm localization error (on a few meters path) is lower than the depth error.

5.A.2 Second Experiment

Protocol. We performed a second experiment in which we asked the candidates to perform the same three measurements, M1, M2, and M3 respectively, both with a laser rangefinder and the Tango TDK to assess the repeatability of the measurements.

The absolute error of the laser rangefinder is constant ($1.5\,\mathrm{mm}$ in our case). However, like any experimental activity the measurements are effected by random errors in the use of the device (*e.g.* the device is not hold perfectly level) and small structural errors of the scene: the floor may not be perfectly flat, the walls may not be perfectly orthogonal to the ground, *etc*. Consequently, the absolute error is proportional to the measured distance with a coefficient equal to $1 - \cos\alpha$ where α is the elevation angle (angular difference with a vector parallel to the ground plane). For this reason, we considered relative errors instead of absolute errors in the results of this experiment.

Rangerfinder measurements are simple to perform when the walls can be used as support for the device, which was the case for the first measurement M1. When the walls are not parallel, as for the third measurement M3, the user has one degree of freedom to adjust the azimuth orientation of the laser. Between two opposite wall corners of a diagonal, the crutch of the rangefinder has to be used to be in contact with the corner. There are two degrees of freedom, the azimuth, and the elevation

angle, to point the device to the opposite corner at the same height. This was the case for the second measurement M2, which is also the longest distance measured among the three. Longer distances are more challenging to measure because a small orientation change of the rangefinder introduces more important displacements of the laser dot. Therefore, with the rangefinder we expect to obtain a higher repeatability with M1 than M3, and to obtain the lowest repeatability with M2. With the Tango TDK device, the three measurements M1, M2 and M3 are equally challenging and the distance length does not introduce difficulties for the operator because he can move during the measurement.

Results. Table 5.3 summarizes the standard deviation of the relative errors of the measurements performed by the candidates. The best repeatability was obtained for all devices, with the easiest measurement M1. With the Tango TDK, the repeatability decreases with the length of the measurement which can be explained by the VIO algorithm drift. Whereas with the laser rangefinder, the repeatability varies less, which means it is more reliable for the measurements despite the difficulties experimented by the users. Contrary to what we expected, $\sigma_{\text{Rangefinder}}$ is lower for M2 than for M3, while M2 was a more difficult measurement. One possible explanation is that most of the candidates understood M2 was a difficult measurement, and and they took their time to perform the measurement, sometimes with several attempts because they wanted to check the consistency of their results. In comparison, no candidate performed several measurements for M3.

Measurement	$\sigma_{\text{Rangefinder}}$	σ_{Tango}
M1 (7.2 m)	$0.7\,\text{mm}\,\text{m}^{-1}$	$5.5 \pm 3.5\,\text{mm}\,\text{m}^{-1}$
M2 (11.9 m)	$0.83\,\text{mm}\,\text{m}^{-1}$	$22.6 \pm 2.1\,\text{mm}\,\text{m}^{-1}$
M3 (9.7 m)	$1.03\,\text{mm}\,\text{m}^{-1}$	$10.4 \pm 2.6\,\text{mm}\,\text{m}^{-1}$

Table 5.3: Results of the three measurements M1, M2, M3 performed by the candidates. $\sigma_{\text{Rangefinder}}$ and σ_{Tango} represent the standard deviation of the relative errors of the measurements.

Chapter 6
Conclusion

Contents

6.1 Summary 105
6.2 Limitations and perspectives 106
6.3 Publications 107
6.4 Collaborations 107

6.1 Summary

All along this manuscript, we studied different components related to the problem of room layout estimation on mobile devices, under two points of view: RGB and RGB-D approaches. In the context of the study of RGB approaches, we focused on the problem of VPs estimation to help image understanding, and in particular the estimation of room layout from a single image. We explored the use of inertial sensors with the purpose of developing a mobile application and we devised a novel, fast and accurate solution to estimate the VPs of an image using IMU data. The comparison of our approach with a state of the art approach revealed we obtained similar accuracy while being several magnitudes faster. In order to assess our method we create a new dataset with annotated ground truth line segments and IMU data, which can be used for the evaluation of VPs algorithms. Instead of giving the ground truth position of the VPs, we provide uncertainty regions where the VPs may lie, in order to take into account the uncertainties of the manual labeling process.

In the context of analyzing RGB-D approaches, we considered two devices. We started with the very recent Structure Sensor, which revealed to have some issues related to the provided SLAM component. SLAM algorithms generally consist of two elements: a front-end made of a fast odometry algorithm called VO, which is mainly reliable in the short term (*e.g.* a few minutes), and a back-end, which handles loop closures and ensure the estimated trajectory is globally consistent. This component is very critical, since even minor deviations or misalignments affect the quality of the measurements and of the augmented reality application. We proposed a benchmark of the most promising VO algorithms based on RGB-D sensors that could be suitable for a mobile device: we evaluated their accuracy on several datasets and we assessed their computational efficiency and memory consumption. This evaluation revealed that only one open-source algorithm (Fovis [81]) could be run with QVGA (320×240) images at the rate of the camera, on all the considered devices. On the high-end devices, it was also the only algorithm that could be run with VGA (640×480) images. Otherwise, the VO algorithm provided with the Structure Sensor obtained the best performances on the dataset using this sensor.

We continued our experiments with the Tango TDK, which revealed to have a more accurate SLAM algorithm. Contrary to Structure Sensor, it does not use the data of the depth sensor for estimating the motion of the device but it rather uses a RGB approach. The first experiments showed that such choice enables the real-time motion estimation, but it also introduces some drift that affect the on-line 3D reconstruction using the RGB-D data. To mitigate this effect, we investigated planar SLAM approaches and we developed our own plane extraction algorithm to satisfy the constraints of the provided depth sensor. Our proof of concept planar SLAM runs on top of the existing Tango components and it considers the uncertainty on the estimated planes to relax the constraints where the uncertainty is the higher.

Planes of an indoor scene generally correspond to its structural elements (walls, ceiling, and floor). For this reason, it is a popular approach for the estimation of room layouts from 3D data. While previous works or existing applications are mostly offline (the room layout is estimated without user intervention, at the end of the acquisition procedure) or user-driven (the floor plan is fully generated by the user), we proposed a novel approach where the user interactions are part of the processing loop. A comparison with existing mobile applications revealed our method was generally more accurate and could require fewer user inputs.

6.2 Limitations and perspectives

The floor plan estimation problem addressed in this thesis covers several topics: image understanding, vSLAM, semantic analysis and user interactions. Such broad spectrum of topics clearly leave space for improvements and further investigations. In what follows, we discuss some limitations and perspectives of our works.

Concerning the works on vSLAM algorithms, the recent release of the localization frameworks ARKit and ARCore propose an interesting track to explore, as they both provide support for plane detection. In comparison with Project Tango, these technologies offer a broader impact to users, as they do not require to buy additional hardware, and they are meant to enable augmented reality applications. Existing approaches for room layout based on multiple images [12, 57, 56, 58] could be revisited to take advantage of these technologies. However, according to our preliminary experiments, these frameworks are, for the time being, lacking the accuracy necessary for generating good quality plans. An interesting extension of our works on planar SLAM would be a tight integration (and not separated as we proposed to accommodate the existing Tango components) with an existing feature-based open-source VO algorithm, in order to take advantage of the uncertainty information on the odometry and to fully constrain the camera pose from the matched features when few planes are visible.

A finer segmentation (at the level of objects) can improve the recognition of walls and clutter, as well as enabling the detection of doors and windows. Deep learning techniques are nowadays very popular and they can be used to train specific networks on datasets such as [31] to provide a semantic segmentation of the scene. Tateno *et al*. demonstrated [202] such approach could allow a Tango TDK to segment in real-time the point cloud generated during the acquisition. The understanding of the scene provided by a semantic analysis can also be considered to fill holes in the scenes and perform scene completion, as proposed by [51] who also considers 3D data from the same device.

Regarding our works related to the estimation of the room layout, RGB images could be also integrated in the pipeline. Color information, and especially textures are very valuable for the semantic analysis of indoor scenes but also to create textured 3D models. It is a difficult problem since the model may be incomplete, the superposition of overlapping RGB images may create a blurry texture due to the imperfect accuracy of the camera tracking and quality of the images. Huang *et al*. [82] combine hole filling in meshes, inpainting algorithms, and texture optimization approaches to enhance the quality of the generated textured models. Inpainting can also be considered, in the context of a diminished reality application where the clutter would be virtually removed of the scene as proposed by Zhang *et al*. [222] Zhang *et al*. [222] propose such an application where an illumination model is recreated to add further realism to the textures. Once the 3D model of the room is estimated, it can be interesting to add some 3D objects (furnitures, equipments, ...) of the scanned room into the model. Sankar *et al*. [176] propose an application for the Tango TDK where some of the objects of the scenes are recognized and added to the generated model.

Regarding the user experience, it can be interesting to guide the user during the scan [114, 132] in order to optimize the time spent in the apartment, as well as designing more intuitive user interfaces [181].

While our proof-of-concept application demonstrates the feasibility of our approach, more engineering work is required to release a mobile application. Some of the components of our pipelines, such as the planar model optimization and the room layout estimation require better optimizations and to be implemented in a cross-platform language such as C++. Additional features such as data export, the aggregation of multiple rooms in the floor plan, and the detection (or the manual selection) of doors and windows should be considered.

6.3 Publications

The research work presented in this thesis has been published in international conferences and journals. In the following list, we report the publications with a brief description of the contribution.

[4] Vincent Angladon, Simone Gasparini, and Vincent Charvillat. "Room Floor Plan Generation on a Project Tango Device". In: *MultiMedia Modeling*. Springer International Publishing, 2018, pp. 226–238. DOI: 10.1007/978-3-319-73600-6_20. URL: https://doi.org/10.1007/978-3-319-73600-6_20
In this paper, we proposed a collaborative and online method to generate accurate floor plans from a Tango TDK. We take advantage of the specific capabilities of this tablet (presence of a built-in depth sensor and a localization module) to propose a collaborative user interaction scheme. Our approach relies on the extraction and the fusion of planes extracted from the 3D data, and generally a specially crafted graph to model the problem.

[5] Vincent Angladon, Simone Gasparini, and Vincent Charvillat. "The Toulouse vanishing points dataset". In: *Proceedings of the 6th ACM Multimedia Systems Conference on - MMSys '15*. ACM Press, 2015. DOI: 10.1145/2713168.2713196. URL: https://doi.org/10.1145/2713168.2713196
In this paper we presented a new dataset for the evaluation of VPs estimation algorithms. Its originality is the addition of IMU data synchronized with the camera, allowing VPs estimation algorithms taking advantage of this sensor to be compared. For our dataset, we also propose VPs of reference in the form of uncertainty regions, which express the uncertainty on the extremities of the ground truth line segments.

[6] Vincent Angladon, Simone Gasparini, Vincent Charvillat, Tomislav Pribanić, Tomislav Petković, Matea Đonlić, Benjamin Ahsan, and Frédéric Bruel. "An evaluation of real-time RGB-D visual odometry algorithms on mobile devices". In: *Journal of Real-Time Image Processing* (2017). in press, pp. 1–18. DOI: 10.1007/s11554-017-0670-y
In this paper, we exposed the results of an evaluation of several VO algorithms designed to take advantage of a depth sensor. We compared their accuracy, their memory and CPU consumption on a desktop computer and several mobile devices. We highlighted the solutions which obtained satisfactory performances to be run on a mobile device.

6.4 Collaborations

[154] Tomislav Pribanić, Tomislav Petković, Matea Đonlić, Vincent Angladon, and Simone Gasparini. "3D Structured Light Scanner on the Smartphone". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 443–450. DOI: 10.1007/978-3-319-41501-7_50. URL: https://doi.org/10.1007/978-3-319-41501-7_50
This paper is the result of a collaboration with the University of Zagreb in the context of the Project Cogito. It describes a method to capture a dense depth map with the Samsung Galaxy Beam, a smartphone with a built-in projector, which is used to project a pseudorandom dots pattern. As the projector and the rear camera do not have a common Field of view (fov), a deflection adapter for the projector is designed, as well as a calibration procedure. The article shows the resulting 3D point cloud obtained with this method and a comparison with point clouds of the same objects obtained with a more powerful projector.

Bibliography

[1] D G Aguilera, J Gómez Lahoz, and J Finat Codes. "A new method for vanishing points detection in 3D reconstruction from a single view". In: *Proceedings of the ISPRS Working Group V/4 Workshop*. 2005.

[2] Abhishek Anand et al. "Contextually guided semantic labeling and search for three-dimensional point clouds". In: *The International Journal of Robotics Research* 32.1 (Nov. 2012), pp. 19–34. DOI: 10.1177/0278364912461538. URL: https://doi.org/10.1177/0278364912461538.

[3] Henrik Andreasson and Todor Stoyanov. "Real time registration of RGB-D data using local visual features and 3D-NDT registration". In: *Proceedings of International Conference on Robotics and Automation (ICRA) Workshop on Semantic Perception, Mapping and Exploration (SPME)*. IEEE Computer Society, 2012.

[4] Vincent Angladon, Simone Gasparini, and Vincent Charvillat. "Room Floor Plan Generation on a Project Tango Device". In: *MultiMedia Modeling*. Springer International Publishing, 2018, pp. 226–238. DOI: 10.1007/978-3-319-73600-6_20. URL: https://doi.org/10.1007/978-3-319-73600-6_20.

[5] Vincent Angladon, Simone Gasparini, and Vincent Charvillat. "The Toulouse vanishing points dataset". In: *Proceedings of the 6th ACM Multimedia Systems Conference on - MMSys '15*. ACM Press, 2015. DOI: 10.1145/2713168.2713196. URL: https://doi.org/10.1145/2713168.2713196.

[6] Vincent Angladon et al. "An evaluation of real-time RGB-D visual odometry algorithms on mobile devices". In: *Journal of Real-Time Image Processing* (2017). in press, pp. 1–18. DOI: 10.1007/s11554-017-0670-y.

[7] M.E. Antone and S. Teller. "Automatic recovery of relative camera rotations for urban scenes". In: *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*. Vol. 2. IEEE Comput. Soc, 2000, pp. 282–289. ISBN: 0-7695-0662-3. DOI: 10.1109/CVPR.2000.854809. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=854809.

[8] Michel Antunes and Joao P Barreto. "A Global Approach for the Detection of Vanishing Points and Mutually Orthogonal Vanishing Directions". In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2013)* (June 2013), pp. 1336–1343. DOI: 10.1109/CVPR.2013.176. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6619020.

[9] Murat Arikan et al. "O-snap". In: *ACM Transactions on Graphics* 32.1 (Jan. 2013), pp. 1–15. DOI: 10.1145/2421636.2421642. URL: https://doi.org/10.1145/2421636.2421642.

[10] ARM. *ARM Compiler toolchain Assembler Reference v5.0 - NEON instructions*. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0489c/CJAJIIGG.html. 2016.

[11] Dan Banica and Cristian Sminchisescu. "Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in RGB-D images". In: (June 2015). DOI: 10.1109/cvpr.2015.7298974. URL: https://doi.org/10.1109/cvpr.2015.7298974.

[12] Sid Yingze Bao et al. "Understanding the 3D layout of a cluttered room from multiple images". In: *IEEE Winter Conference on Applications of Computer Vision*. IEEE, Mar. 2014. DOI: 10.1109/wacv.2014.6836035. URL: https://doi.org/10.1109/wacv.2014.6836035.

[13] S Barnard. "Interpreting perspective images". In: *Artificial Intelligence* 21.4 (Nov. 1983), pp. 435–462. ISSN: 00043702. DOI: 10.1016/S0004-3702(83)80021-6. URL: http://linkinghub.elsevier.com/retrieve/pii/S0004370283800216.

[14] Joan Batlle, E Mouaddib, and Joaquim Salvi. “Recent progress in coded structured light as a technique to solve the correspondence problem: a survey”. In: *Pattern recognition* 31.7 (1998), pp. 963–982.

[15] Eric T. Bax. “Algorithms to count paths and cycles”. In: *Information Processing Letters* 52.5 (Dec. 1994), pp. 249–252. DOI: 10.1016/0020-0190(94)00151-0. URL: https://doi.org/10.1016/0020-0190(94)00151-0.

[16] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded up robust features”. In: *Proceedings of the 2006 European Conference on Computer Vision (ECCV 2006)*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023. URL: http://link.springer.com/10.1007/11744023.

[17] Jean Charles Bazin et al. “Globally optimal line clustering and vanishing point estimation in Manhattan world”. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2012)*. 1. IEEE, June 2012, pp. 638–645. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6247731. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247731.

[18] M. Ben. *Principles of Concurrent and Distributed Programming, Second Edition*. Second. Addison-Wesley, 2006. ISBN: 9780321312839.

[19] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008. ISBN: 3540779736, 9783540779735. DOI: 10.1016/s0898-1221(98)90095-5. URL: http://dx.doi.org/10.1016/s0898-1221(98)90095-5.

[20] P. J. Besl and H. D. McKay. “A method for registration of 3-D shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791. URL: http://dx.doi.org/10.1109/34.121791.

[21] J. Biswas and M. Veloso. “Planar polygon extraction and merging from depth images”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 3859–3864. DOI: 10.1109/iros.2012.6385841. URL: http://dx.doi.org/10.1109/iros.2012.6385841.

[22] Johann Borenstein, H. R. Everett, and Liqiang Feng. *Navigating Mobile Robots: Systems and Techniques*. Natick, MA, USA: A. K. Peters, Ltd., 1996. ISBN: 1568810660.

[23] Dorit Borrmann et al. “The 3D Hough Transform for Plane Detection in Point Clouds: A Review and a New Accumulator Design”. In: *3D Res.* 2.2 (Mar. 2011), 32:1–32:13. ISSN: 2092-6731. DOI: 10.1007/3DRes.02(2011)3. URL: http://dx.doi.org/10.1007/3DRes.02(2011)3.

[24] J. Y. Bouguet. *Camera calibration toolbox for Matlab*. 2008. URL: http://www.vision.caltech.edu/bouguetj/calib%5C_doc/..

[25] Nicholas Brunetto et al. “Fusion of Inertial and Visual Measurements for RGB-D SLAM on Mobile Devices”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. IEEE Computer Society, Dec. 2015. DOI: 10.1109/iccvw.2015.29. URL: http://dx.doi.org/10.1109/iccvw.2015.29.

[26] Angela Budroni and Jan Boehm. “Automated 3D Reconstruction of Interiors from Point Clouds”. In: *International Journal of Architectural Computing* 8.1 (Jan. 2010), pp. 55–73. DOI: 10.1260/1478-0771.8.1.55. URL: https://doi.org/10.1260/1478-0771.8.1.55.

[27] Cesar Cadena et al. “Simultaneous localization and mapping: Present, future, and the robust-perception age”. In: *CoRR, vol. abs/1606.05830* (2016).

[28] Michael Calonder et al. “BRIEF: Binary robust independent elementary features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Vol. 6314 LNCS. PART 4. Springer Verlag, 2010, pp. 778–792. ISBN: 364215560X. DOI: 10.1007/978-3-642-15561-1_56. URL: http://dx.doi.org/10.1007/978-3-642-15561-1_56.

[29] Federico Camposeco and Marc Pollefeys. "Using vanishing points to improve visual-inertial odometry". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015. DOI: 10.1109/icra.2015.7139926. URL: https://doi.org/10.1109/icra.2015.7139926.

[30] B. Caprile and V. Torre. "Using vanishing points for camera calibration". In: *International Journal of Computer Vision* 4.2 (Mar. 1990), pp. 127–139. ISSN: 0920-5691. DOI: 10.1007/BF00127813. URL: http://link.springer.com/10.1007/BF00127813.

[31] Angel X. Chang et al. "Matterport3D: Learning from RGB-D Data in Indoor Environments". In: *International Conference on 3D Vision (3DV)* abs/1709.06158 (2017). URL: http://arxiv.org/abs/1709.06158.

[32] Kang Chen, Yu-Kun Lai, and Shi-Min Hu. "3D indoor scene modeling from RGB-D data: a survey". In: *Computational Visual Media* 1.4 (Dec. 1, 2015), pp. 267–278. ISSN: 2096-0662. DOI: 10.1007/s41095-015-0029-x. URL: https://doi.org/10.1007/s41095-015-0029-x.

[33] Y. Chen and G. Medioni. "Object modeling by registration of multiple range images". In: *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA)*. Apr. 1991, 2724–2729 vol.3. DOI: 10.1109/robot.1991.132043. URL: http://dx.doi.org/10.1109/robot.1991.132043.

[34] Jiwon Choi et al. "Real-time vanishing point detection using the Local Dominant Orientation Signature". In: *Proceedings of the 2011 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*. IEEE, 2011, pp. 1–4.

[35] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. "Robust reconstruction of indoor scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5556–5565. DOI: 10.1109/cvpr.2015.7299195. URL: http://dx.doi.org/10.1109/cvpr.2015.7299195.

[36] R.T. Collins and R.S. Weiss. "Vanishing point calculation as a statistical inference on the unit sphere". In: *Proceedings of the 1990 IEEE International Conference on Computer Vision (ICCV 1990)*. IEEE Comput. Soc. Press, 1990, pp. 400–403. ISBN: 0-8186-2057-9. DOI: 10.1109/ICCV.1990.139560. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=139560.

[37] Alejo Concha et al. "Incorporating scene priors to dense monocular mapping". In: *Autonomous Robots* 39.3 (July 2015), pp. 279–292. DOI: 10.1007/s10514-015-9465-9. URL: https://doi.org/10.1007/s10514-015-9465-9.

[38] Peter Corke, Jorge Lobo, and Jorge Dias. "An Introduction to Inertial and Visual Sensing". In: *The International Journal of Robotics Research* 26.6 (June 2007), pp. 519–535. ISSN: 0278-3649. DOI: 10.1177/0278364907079279. URL: http://ijr.sagepub.com/cgi/doi/10.1177/0278364907079279.

[39] W.D. Curtis, A.L. Janin, and K. Zikan. "A note on averaging rotations". In: *Proceedings of the IEEE Virtual Reality Annual International Symposium*. IEEE, 1993, pp. 377–385. ISBN: 0-7803-1363-1. DOI: 10.1109/VRAIS.1993.380755. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=380755.

[40] Carlo Dal Mutto, Pietro Zanuttigh, and Guido M Cortelazzo. *Time-of-Flight Cameras and Microsoft KinectTM*. Springer Science & Business Media, 2012. Chap. CW Matricial Time-of-Flight Range Cameras.

[41] Emmanuel d'Angelo et al. "From bits to images: Inversion of local binary descriptors". In: *IEEE transactions on pattern analysis and machine intelligence* 36.5 (2014), pp. 874–887. DOI: 10.1109/tpami.2013.228. URL: http://dx.doi.org/10.1109/tpami.2013.228.

[42] Patrick Denis, James H. Elder, and Francisco J. Estrada. "Efficient Edge-Based Methods for Estimating Manhattan Frames in Urban Imagery". In: *Proceedings of the 2008 European Conference on Computer Vision (ECCV 2008)*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Springer Berlin Heidelberg, 2008, pp. 197–210. DOI: 10.1007/978-3-540-88688-4_15.

[43] Patrick Denis, James H. Elder, and Francisco J. Estrada. *York Urban Line Segment Database*. http://www.elderlab.yorku.ca/YorkUrbanDB/. 2008.

[44] Alan Dix et al. *Human-computer Interaction*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997. ISBN: 0-13-437211-5.

[45] Matea Đonlić and Tomislav Petković. "On Tablet 3D Structured Light Reconstruction and Registration". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. IEEE. 2017. DOI: 10.1109/iccvw.2017.290. URL: http://dx.doi.org/10.1109/iccvw.2017.290.

[46] Mingsong Dou et al. "Exploring High-level Plane Primitives for Indoor 3D Reconstruction with a Hand-held RGB-D Camera". In: *Proceedings of the 11th International Conference on Computer Vision - Volume 2*. ACCV'12. Daejeon, Korea: Springer-Verlag, 2013, pp. 94–108. ISBN: 978-3-642-37483-8. DOI: 10.1007/978-3-642-37484-5_9. URL: http://dx.doi.org/10.1007/978-3-642-37484-5_9.

[47] David H. Douglas and Thomas K. Peucker. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature". In: (Mar. 2011), pp. 15–28. DOI: 10.1002/9780470669488.ch2. URL: https://doi.org/10.1002/9780470669488.ch2.

[48] Karel Driesen and Urs Hölzle. "The Direct Cost of Virtual Function Calls in C++". In: *SIGPLAN Not.* 31.10 (Oct. 1996), pp. 306–323. ISSN: 0362-1340. DOI: 10.1145/236338.236369. URL: http://doi.acm.org/10.1145/236338.236369.

[49] Ivan Dryanovski, Roberto G. Valenti, and Jizhong Xiao. "Fast visual odometry and mapping from RGB-D data". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2013), pp. 2305–2310. ISSN: 10504729. DOI: 10.1109/icra.2013.6630889. URL: http://dx.doi.org/10.1109/icra.2013.6630889.

[50] Hao Du et al. "Interactive 3D Modeling of Indoor Environments with a Consumer Depth Camera". In: *Proceedings of the 13th International Conference on Ubiquitous Computing*. UbiComp '11. Beijing, China: ACM, 2011, pp. 75–84. ISBN: 978-1-4503-0630-0. DOI: 10.1145/2030112.2030123. URL: http://doi.acm.org/10.1145/2030112.2030123.

[51] Maksym Dzitsiuk et al. "De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. DOI: 10.1109/icra.2017.7989457. URL: https://doi.org/10.1109/icra.2017.7989457.

[52] F. Endres et al. "An evaluation of the RGB-D SLAM system". In: *Proceedings of the 2012 IEEE International Conference Robotics and Automation (ICRA)*. Vol. 1. c. IEEE Computer Society, 2012, pp. 1691–1696. ISBN: 9781467314046. DOI: 10.1109/ICRA.2012.6225199. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=6225199.

[53] Zheng Fang and Yu Zhang. "Experimental Evaluation of RGB-D Visual Odometry Methods". In: *International Journal of Advanced Robotic Systems* 12.3 (2015), pp. 1–16. ISSN: 1729-8806. DOI: 10.5772/59991. URL: http://www.intechopen.com/journals/international%5C_journal%5C_of%5C_advanced%5C_robotic%5C_systems/experimental-evaluation-of-rgb-d-visual-odometry-methods.

[54] C. Feng, Y. Taguchi, and V. R. Kamat. "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 6218–6225. DOI: 10.1109/icra.2014.6907776. URL: http://dx.doi.org/10.1109/icra.2014.6907776.

[55] Robert B. Fisher and Kurt Konolige. "Range Sensors". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 521–542. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_23. URL: https://doi.org/10.1007/978-3-540-30301-5_23.

[56] Alex Flint, David Murray, and Ian Reid. "Manhattan scene understanding using monocular, stereo, and 3D features". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126501. URL: https://doi.org/10.1109/iccv.2011.6126501.

[57] Alex Flint et al. "Growing semantically meaningful models for visual SLAM". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, June 2010. DOI: 10.1109/cvpr.2010.5540176. URL: https://doi.org/10.1109/cvpr.2010.5540176.

[58] Axel Furlan et al. "Free your Camera: 3D Indoor Scene Understanding from Arbitrary Camera Motion". In: *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association, 2013. DOI: 10.5244/c.27.24. URL: https://doi.org/10.5244/c.27.24.

[59] Y. Furukawa et al. "Manhattan-world stereo". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvprw.2009.5206867. URL: https://doi.org/10.1109/cvprw.2009.5206867.

[60] Andreas Geiger. "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012)*. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 3354–3361. ISBN: 978-1-4673-1226-4. DOI: 10.1109/cvpr.2012.6248074. URL: http://dx.doi.org/10.1109/cvpr.2012.6248074.

[61] Andreas Geiger, Julius Ziegler, and Christoph Stiller. "StereoScan: Dense 3D Reconstruction in Real-time". In: *Proceedings of the Intelligent Vehicles Symposium (IV 2011)*. IEEE Computer Society, 2011. DOI: 10.1109/ivs.2011.5940405. URL: http://dx.doi.org/10.1109/ivs.2011.5940405.

[62] Google. *ATAP Project Tango – Google*. http://www.google.com/atap/projecttango/. 2014.

[63] Google. *Floor plan API example for Google Tango devices*. https://github.com/googlearchive/tango-examples-java/tree/master/java_floor_plan_reconstruction_example. 2017.

[64] Google. *java_floor_plan_example create a floor plan by using the depth sensor of a Google Tango device to detect and measure walls in a room*. https://github.com/googlearchive/tango-examples-java/tree/release-biyelgee/java_floor_plan_example. 2016.

[65] Google. *Measure: augmented reality measurement application for Google Tango devices*. https://play.google.com/store/apps/details?id=com.google.tango.measure. 2016.

[66] G Grisetti et al. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. DOI: 10.1109/mits.2010.939925. URL: https://doi.org/10.1109/mits.2010.939925.

[67] Rafael Grompone von Gioi et al. "LSD: a Line Segment Detector". In: *Image Processing On Line* 2 (Mar. 2012), pp. 35–55. ISSN: 2105-1232. DOI: 10.5201/ipol.2012.gjmr-lsd. URL: http://www.ipol.im/pub/art/2012/gjmr-lsd/?utm%5C_source=doi.

[68] Neil J. Gunther. *White paper: UNIX Load Average – Part 1: How It Works*. White paper http://www.teamquest.com/pdfs/whitepaper/ldavg1.pdf. TeamQuest Corporation, 2010.

[69] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. "Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2013. DOI: 10.1109/cvpr.2013.79. URL: https://doi.org/10.1109/cvpr.2013.79.

[70] Maciej Halber and Thomas Funkhouser. "Fine-To-Coarse Global Registration of RGB-D Scans". In: (2017). DOI: 10.1109/cvpr.2017.705. URL: http://dx.doi.org/10.1109/cvpr.2017.705.

[71] Christopher Ham, Simon Lucey, and Surya Singh. "Hand Waving Away Scale". In: *Proceedings of the 2014 European Conference on Computer Vision (ECCV 2014)*. Ed. by David Fleet et al. Vol. 8692. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 279–293. ISBN: 978-3-319-10592-5. DOI: 10.1007/978-3-319-10593-2. URL: http://link.springer.com/10.1007/978-3-319-10593-2.

[72] Ankur Handa et al. "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM". In: *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*. IEEE Computer Society, 2014, pp. 1524–1531. ISBN: 9781479936847. DOI: 10.1109/icra.2014.6907054. URL: http://dx.doi.org/10.1109/icra.2014.6907054.

[73] Varsha Hedau, Derek Hoiem, and David Forsyth. "Recovering the spatial layout of cluttered rooms". In: *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision* (Sept. 2009), pp. 1849–1856. DOI: 10.1109/ICCV.2009.5459411. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459411.

[74] Alexander Hermans, Georgios Floros, and Bastian Leibe. "Dense 3D semantic mapping of indoor scenes from RGB-D images". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014. DOI: 10.1109/icra.2014.6907236. URL: https://doi.org/10.1109/icra.2014.6907236.

[75] Derek Hoiem, Alexei A. Efros, and Martial Hebert. "Recovering Surface Layout from an Image". In: *International Journal of Computer Vision* 75.1 (Feb. 2007), pp. 151–172. DOI: 10.1007/s11263-006-0031-y. URL: https://doi.org/10.1007/s11263-006-0031-y.

[76] Dirk Holz et al. "Robot Soccer World Cup XV". In: ed. by Thomas Röfer et al. Berlin, Heidelberg: Springer-Verlag, 2012. Chap. Real-time Plane Segmentation Using RGB-D Cameras, pp. 306–317. ISBN: 978-3-642-32059-0. URL: http://dl.acm.org/citation.cfm?id=2554542.2554572.

[77] Stefan Holzer et al. "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 2684–2689. DOI: 10.1109/iros.2012.6385999. URL: http://dx.doi.org/10.1109/iros.2012.6385999.

[78] Adam Hoover et al. "An experimental comparison of range image segmentation algorithms". In: *IEEE transactions on pattern analysis and machine intelligence* 18.7 (1996), pp. 673–689. DOI: 10.1109/34.506791. URL: http://dx.doi.org/10.1109/34.506791.

[79] Berthold K P Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *Journal of the Optical Society of America A* 4.4 (1987), pp. 629–642. DOI: 10.1364/josaa.4.000629. URL: http://dx.doi.org/10.1364/josaa.4.000629.

[80] Ming Hsiao et al. "Keyframe-based Dense Planar SLAM". In: *Proc. International Conference on Robotics and Automation (ICRA), IEEE*. 2017. DOI: 10.1109/icra.2017.7989597. URL: http://dx.doi.org/10.1109/icra.2017.7989597.

[81] Albert S. Huang et al. "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera". In: *Proceedings of the International Symposium of Robotics Research (ISRR 2011)*. 2011, pp. 1–16. DOI: 10.1007/978-3-319-29363-9_14. URL: http://dx.doi.org/10.1007/978-3-319-29363-9_14.

[82] Jingwei Huang et al. "3Dlite". In: *ACM Transactions on Graphics* 36.6 (Nov. 2017), pp. 1–14. DOI: 10.1145/3130800.3130824. URL: https://doi.org/10.1145/3130800.3130824.

[83] Marco A. Hudelist, Claudiu Cobârzan, and Klaus Schoeffmann. "OpenCV Performance Measurements on Mobile Devices". In: *Proceedings of International Conference on Multimedia Retrieval - ICMR '14*. New York, New York, USA: ACM Press, 2014, pp. 479–482. ISBN: 9781450327824. DOI: 10.1145/2578726.2578798. URL: http://dl.acm.org/citation.cfm?doid=2578726.2578798.

[84] R. Hulik et al. “Fast and accurate plane segmentation in depth maps for indoor scenes”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 1665–1670. DOI: 10.1109/iros.2012.6385868. URL: http://dx.doi.org/10.1109/iros.2012.6385868.

[85] Graeme Jones. “Accurate and Computationally-inexpensive Recovery of Ego-Motion using Optical Flow and Range Flow with Extended Temporal Support”. In: *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association, 2013, pp. 75.1–75.11. ISBN: 1-901725-49-9. DOI: 10.5244/C.27.75. URL: http://www.bmva.org/bmvc/2013/Papers/paper0075/index.html.

[86] Antonio Torralba. Joseph J. Lim Hamed Pirsiavash. “Parsing IKEA Objects: Fine Pose Estimation”. In: *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV 2013)*. 2013. DOI: 10.1109/iccv.2013.372. URL: http://dx.doi.org/10.1109/iccv.2013.372.

[87] Michael Kaess. “Simultaneous localization and mapping with infinite planes”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4605–4611. DOI: 10.1109/icra.2015.7139837. URL: http://dx.doi.org/10.1109/icra.2015.7139837.

[88] Michael Kaess et al. “iSAM2: Incremental smoothing and mapping using the Bayes tree”. In: *The International Journal of Robotics Research* 31.2 (Dec. 2011), pp. 216–235. DOI: 10.1177/0278364911430419. URL: https://doi.org/10.1177/0278364911430419.

[89] Mahzad Kalantari et al. “Détection automatique des points de fuite et calcul de leur incertitude à l’aide de la géométrie projective”. In: *RFIA 2008*. 2008, pp. 703–712.

[90] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), p. 35. ISSN: 00219223. DOI: 10.1115/1.3662552. URL: http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402.

[91] Yasushi Kanazawa and Ken-ichi Kanatani. “Reliability of plane fitting by range sensing”. In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*. Vol. 2. IEEE. 1995, pp. 2037–2042. DOI: 10.1109/robot.1995.525562. URL: http://dx.doi.org/10.1109/robot.1995.525562.

[92] Christian Kerl, Jurgen Sturm, and Daniel Cremers. “Dense visual SLAM for RGB-D cameras”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 2100–2106. DOI: 10.1109/iros.2013.6696650. URL: http://dx.doi.org/10.1109/iros.2013.6696650.

[93] Christian Kerl, Jurgen Sturm, and Daniel Cremers. “Robust odometry estimation for RGB-D cameras”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2013)*. IEEE Computer Society, 2013, pp. 3748–3754. ISBN: 9781467356411. DOI: 10.1109/icra.2013.6631104. URL: http://dx.doi.org/10.1109/icra.2013.6631104.

[94] Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and resolution of Kinect depth data for indoor mapping applications.” In: *Sensors (Basel, Switzerland)* 12.2 (Jan. 2012), pp. 1437–54. ISSN: 1424-8220. DOI: 10.3390/s120201437. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3304120%5C&tool=pmcentrez%5C&rendertype=abstract.

[95] Kourosh Khoshelham and Sisi Zlatanova. *Sensors for indoor mapping and navigation*. 2016.

[96] Byung-soo Kim, Pushmeet Kohli, and Silvio Savarese. “3D scene understanding by Voxel-CRF”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1425–1432. DOI: 10.1109/iccv.2013.180. URL: http://dx.doi.org/10.1109/iccv.2013.180.

[97] Matthew Klingensmith et al. “Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device”. In: *Proceedings of the 2015 Robotics Science and Systems Conference*. July 2015.

[98] Florian Kluger et al. “Deep Learning for Vanishing Point Detection Using an Inverse Gnomonic Projection”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2017, pp. 17–28. DOI: 10.1007/978-3-319-66709-6_2. URL: https://doi.org/10.1007/978-3-319-66709-6_2.

[99] K. Kolev et al. “Turning Mobile Phones into 3D Scanners”. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2014, pp. 3946–3953. DOI: 10.1109/cvpr.2014.504. URL: http://dx.doi.org/10.1109/cvpr.2014.504.

[100] J Košecká and W Zhang. “Extraction, matching, and pose recovery based on dominant rectangular structures”. In: *Computer Vision and Image Understanding* February 2005 (2005). DOI: 10.1016/j.cviu.2005.04.005. URL: http://www.sciencedirect.com/science/article/pii/S1077314205000482.

[101] Jana Košecká and Wei Zhang. “Video Compass”. In: *Proceedings of the 2002 European Conference on Computer Vision (ECCV2002)*. Ed. by Anders Heyden et al. Springer Berlin Heidelberg, 2002, pp. 476–490. DOI: 10.1007/3-540-47979-1_32. URL: http://link.springer.com/chapter/10.1007/3-540-47979-1%5C_32.

[102] Rainer Kummerle et al. “G2o: A general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. DOI: 10.1109/icra.2011.5979949. URL: https://doi.org/10.1109/icra.2011.5979949.

[103] D.C. Lee, M. Hebert, and T. Kanade. “Geometric reasoning for single image structure recovery”. In: *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvprw.2009.5206872. URL: https://doi.org/10.1109/cvprw.2009.5206872.

[104] Gun A. Lee and Mark Billinghurst. “A user study on the Snap-To-Feature interaction method”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, Oct. 2011. DOI: 10.1109/ismar.2011.6162899. URL: https://doi.org/10.1109/ismar.2011.6162899.

[105] Gun A Lee et al. “Freeze-Set-Go interaction method for handheld mobile augmented reality environments”. In: *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. ACM. 2009, pp. 143–146. DOI: 10.1145/1643928.1643961. URL: http://dx.doi.org/10.1145/1643928.1643961.

[106] Jeong-Kyun Lee et al. “Joint Layout Estimation and Global Multi-View Registration for Indoor Reconstruction”. In: *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV 2017)*. 2017. DOI: 10.1109/iccv.2017.27. URL: http://dx.doi.org/10.1109/iccv.2017.27.

[107] Tae-Kyeong Lee et al. “Indoor mapping using planes extracted from noisy RGB-D sensors”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012. DOI: 10.1109/iros.2012.6385909. URL: http://dx.doi.org/10.1109/iros.2012.6385909.

[108] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. “BRISK: Binary Robust invariant scalable keypoints”. In: *Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV2011)*. IEEE Computer Society, Nov. 2011, pp. 2548–2555. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126542. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126542.

[109] José Lezama, Gregory Randall, and Rafael Grompone von Gioi. “Vanishing Point Detection in Urban Scenes Using Point Alignments”. In: *Image Processing On Line* 7 (July 2017), pp. 131–164. DOI: 10.5201/ipol.2017.148. URL: https://doi.org/10.5201/ipol.2017.148.

[110] Bo Li et al. *PKU Campus Database*. http://www.cis.pku.edu.cn/vision/vpdetection/. 2012.

[111] Bo Li et al. "Vanishing point detection using cascaded 1D Hough Transform from single images". In: *Pattern Recognition Letters* 33.1 (Jan. 2012), pp. 1–8. ISSN: 01678655. DOI: 10.1016/j.patrec.2011.09.027. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167865511003060.

[112] Yanzhe Li, Kai Huang, and Luc Claesen. "SoC and FPGA oriented high-quality stereo vision system". In: *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*. IEEE. 2016, pp. 1–4. DOI: 10.1109/fpl.2016.7577366. URL: http://dx.doi.org/10.1109/fpl.2016.7577366.

[113] Nicolas Limare. *Integer and Floating-Point Arithmetic Speed vs Precision*. http://nicolas.limare.net/pro/notes/2014/12/12_arit_speed/#index3h2. 2014.

[114] Kok-Lim Low and Anselmo Lastra. "Efficient Constraint Evaluation Algorithms for Hierarchical Next-Best-View Planning". In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. IEEE, June 2006. DOI: 10.1109/3dpvt.2006.52. URL: https://doi.org/10.1109/3dpvt.2006.52.

[115] DG Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: http://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94.

[116] Simon Lynen et al. "Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization". en. In: *Robotics: Science and Systems XI*. Ed. by Lydia E. Kavraki, David Hsu, and Jonas Buchli. Robotics. Science and Systems XI; Conference Location: Rome, Italy; Conference Date: July 13-17, 2015; . RSS, 2015, p. 37. ISBN: 978-0-9923747-1-6. DOI: 10.15607/rss.2015.xi.037. URL: http://dx.doi.org/10.15607/rss.2015.xi.037.

[117] Lingni Ma et al. "Cpa-slam: Consistent plane-model alignment for direct rgb-d slam". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1285–1291. DOI: 10.1109/icra.2016.7487260. URL: http://dx.doi.org/10.1109/icra.2016.7487260.

[118] Lingni Ma et al. "Multi-view deep learning for consistent semantic mapping with rgb-d cameras". In: *arXiv preprint arXiv:1703.08866* (2017). DOI: 10.1109/iros.2017.8202213. URL: http://dx.doi.org/10.1109/iros.2017.8202213.

[119] Mantis Vision. *MV4D depth sensor for mobile devices*. http://www.mv4d.com/mobile.php. 2014.

[120] J. Matas, C. Galambos, and J. Kittler. "Progressive Probabilistic Hough Transform". In: *Procedings of the British Machine Vision Conference 1998*. British Machine Vision Association, 1998. DOI: 10.5244/c.12.26. URL: https://doi.org/10.5244/c.12.26.

[121] Branislav Micusik, Horst Wildenauer, and Jana Kosecka. "Detection and matching of rectilinear structures". In: *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2008)*. IEEE, June 2008, pp. 1–7. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587488. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4587488.

[122] Vicente Morell-Gimenez et al. "A comparative study of registration methods for RGB-D video of static scenes". In: *Sensors (Switzerland)* 14.5 (2014), pp. 8547–8576. ISSN: 14248220. DOI: 10.3390/s140508547. URL: http://dx.doi.org/10.3390/s140508547.

[123] C. Mura, O. Mattausch, and R. Pajarola. "Piecewise-planar Reconstruction of Multi-room Interiors with Arbitrary Wall Arrangements". In: *Proceedings of the 24th Pacific Conference on Computer Graphics and Applications*. PG '16. Okinawa, Japan: Eurographics Association, 2016, pp. 179–188. DOI: 10.1111/cgf.13015. URL: https://doi.org/10.1111/cgf.13015.

[124] Claudio Mura et al. "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts". In: *Computers & Graphics* 44 (2014), pp. 20–32. DOI: 10.1016/j.cag.2014.07.005. URL: http://dx.doi.org/10.1016/j.cag.2014.07.005.

[125] Srivathsan Murali et al. “Indoor Scan2BIM: Building Information Models of House Interiors”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017. DOI: 10.1109/iros.2017.8206513. URL: http://dx.doi.org/10.1109/iros.2017.8206513.

[126] Liangliang Nan and Peter Wonka. “PolyFit: Polygonal Surface Reconstruction from Point Clouds”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2353–2361. DOI: 10.1109/iccv.2017.258. URL: http://dx.doi.org/10.1109/iccv.2017.258.

[127] Richard A Newcombe et al. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2011)*. IEEE Computer Society, 2011, pp. 127–136. ISBN: 9781457721854. DOI: 10.1109/ismar.2011.6162880. URL: http://dx.doi.org/10.1109/ismar.2011.6162880.

[128] Chuong V. Nguyen, Shahram Izadi, and David Lovell. “Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking”. In: *Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. 3DIMPVT ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 524–530. ISBN: 978-0-7695-4873-9. DOI: 10.1109/3DIMPVT.2012.84. URL: http://dx.doi.org/10.1109/3DIMPVT.2012.84.

[129] Marcos Nieto and Luis Salgado. “Real-time robust estimation of vanishing points through nonlinear optimization”. In: *Proceedings of SPIE 7724, Real-Time Image and Video Processing 2010*. Ed. by Nasser Kehtarnavaz and Matthias F. Carlsohn. Apr. 2010, pp. 772402–772402–14. DOI: 10.1117/12.854592. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=749583.

[130] Marcos Nieto et al. “Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy”. In: *Pattern Analysis and Applications* 14.2 (Apr. 2011), pp. 149–163. DOI: 10.1007/s10044-011-0211-4. URL: https://doi.org/10.1007/s10044-011-0211-4.

[131] D. Nister, O. Naroditsky, and J. Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*. Vol. 1. IEEE Computer Society, June 2004, pp. 652–659. DOI: 10.1007/978-3-319-17885-1_101465. URL: http://dx.doi.org/10.1007/978-3-319-17885-1_101465.

[132] Bradley D. Null and Eric D. Sinzinger. “Next Best View Algorithms for Interior and Exterior Model Acquisition”. In: *Advances in Visual Computing*. Springer Berlin Heidelberg, 2006, pp. 668–677. DOI: 10.1007/11919629_67. URL: https://doi.org/10.1007/11919629_67.

[133] Kristian Nymoen et al. “Comparing motion data from an iPod touch to a high-end optical infrared marker-based motion capture system”. In: *Proceedings of the International Conference on New Interfaces For Musical Expression*. 2012, pp. 88–91.

[134] Occipital. *Fast, easy, mobile 3D scanning for home service pros and DIY warriors*. https://canvas.io. 2016.

[135] Occipital. *ST depth*. `STTracker` instance with the `kSTTrackerTypeKey` set to `STTrackerDepthBased`. 2013.

[136] Occipital. *ST hybrid*. `STTracker` instance with the `kSTTrackerTypeKey` set to `STTrackerDepthAndColorBased` 2013.

[137] Occipital. *The fastest way to measure (iOS application)*. https://tapmeasure.io/. 2017.

[138] Occipital Inc. *The Structure sensor*. http://structure.io. 2014.

[139] Sebastian Ochmann et al. “Automatic reconstruction of parametric building models from indoor point clouds”. In: *Computers & Graphics* 54 (2016), pp. 94–103. DOI: 10.1016/j.cag.2015.07.008. URL: http://dx.doi.org/10.1016/j.cag.2015.07.008.

[140] Sven Oesau, Florent Lafarge, and Pierre Alliez. "Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 90 (2014), pp. 68–82. DOI: 10.1016/j.isprsjprs.2014.02.004. URL: http://dx.doi.org/10.1016/j.isprsjprs.2014.02.004.

[141] P. Ondrúška, P. Kohli, and S. Izadi. "MobileFusion: Real-Time Volumetric Surface Reconstruction and Dense Tracking on Mobile Phones". In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (Nov. 2015), pp. 1251–1258. ISSN: 1077-2626. DOI: 10.1109/tvcg.2015.2459902. URL: http://dx.doi.org/10.1109/tvcg.2015.2459902.

[142] OSGeo. *GEOS: Geometry Engine, Open Source*. https://trac.osgeo.org/geos/. 2000.

[143] Kaustubh Pathak, Narunas Vaskevicius, and Andreas Birk. "Uncertainty analysis for optimum plane extraction from noisy 3D range-sensor point-clouds". In: *Intelligent Service Robotics* 3.1 (2010), pp. 37–48. DOI: 10.1007/s11370-009-0057-4. URL: http://dx.doi.org/10.1007/s11370-009-0057-4.

[144] Kaustubh Pathak et al. "Fast registration based on noisy planes with unknown correspondences for 3-D mapping". In: *IEEE Transactions on Robotics* 26.3 (2010), pp. 424–441. DOI: 10.1109/tro.2010.2042989. URL: http://dx.doi.org/10.1109/tro.2010.2042989.

[145] Kaustubh Pathak et al. "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation". In: *Journal of Field Robotics* 27.1 (2010), pp. 52–84. DOI: 10.1002/rob.20322. URL: http://dx.doi.org/10.1002/rob.20322.

[146] Giovanni Pintore, Marco Agus, and Enrico Gobbetti. "Interactive mapping of indoor building structures through mobile devices". In: *3D Vision (3DV), 2014 2nd International Conference on*. Vol. 2. IEEE. 2014, pp. 103–110. DOI: 10.1109/3dv.2014.40. URL: http://dx.doi.org/10.1109/3dv.2014.40.

[147] Giovanni Pintore et al. "Omnidirectional image capture on mobile devices for fast automatic generation of 2.5 D indoor maps". In: *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE. 2016, pp. 1–9.

[148] Giovanni Pintore et al. "Omnidirectional image capture on mobile devices for fast automatic generation of 2.5D indoor maps". In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2016. DOI: 10.1109/wacv.2016.7477631. URL: https://doi.org/10.1109/wacv.2016.7477631.

[149] François Pomerleau, Francis Colas, and Roland Siegwart. "A Review of Point Cloud Registration Algorithms for Mobile Robotics". In: *Foundations and Trends in Robotics* 4.1-104 (2015), pp. 1–104. ISSN: 1935-8253. DOI: 10.1561/2300000035. URL: http://www.nowpublishers.com/article/Details/ROB-035.

[150] J. Poppinga et al. "Fast plane detection and polygonalization in noisy 3D range images". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2008, pp. 3378–3383. DOI: 10.1109/iros.2008.4650729. URL: http://dx.doi.org/10.1109/iros.2008.4650729.

[151] Jenny Preece, Yvonne Rogers, and Helen Sharp. *Beyond Interaction Design: Beyond Human-Computer Interaction*. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN: 0471402494.

[152] M Previtali et al. "Towards automatic indoor reconstruction of cluttered building rooms from point clouds". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2.5 (2014), p. 281. DOI: 10.5194/isprsannals-ii-5-281-2014. URL: http://dx.doi.org/10.5194/isprsannals-ii-5-281-2014.

[153] Tomislav Pribanić et al. "3D Structured Light Scanner on the Smartphone". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 443–450. DOI: 10.1007/978-3-319-41501-7_50. URL: https://doi.org/10.1007/978-3-319-41501-7_50.

[154] Tomislav Pribanić et al. “3D Structured Light Scanner on the Smartphone”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 443–450. DOI: 10.1007/978-3-319-41501-7_50. URL: https://doi.org/10.1007/978-3-319-41501-7_50.

[155] V. A. Prisacariu et al. “Real-Time 3D Tracking and Reconstruction on Mobile Phones”. In: *IEEE Transactions on Visualization and Computer Graphics* 21.5 (May 2015), pp. 557–570. ISSN: 1077-2626. DOI: 10.1109/tvcg.2014.2355207. URL: http://dx.doi.org/10.1109/tvcg.2014.2355207.

[156] V.A. Prisacariu et al. “Simultaneous 3D tracking and reconstruction on a mobile phone”. In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2013)*. IEEE Computer Society, Oct. 2013, pp. 89–98. DOI: 10.1109/ismar.2013.6671768. URL: http://dx.doi.org/10.1109/ismar.2013.6671768.

[157] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *arXiv preprint arXiv:1612.00593* (2016). DOI: 10.1109/cvpr.2017.16. URL: http://dx.doi.org/10.1109/cvpr.2017.16.

[158] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA Workshop on Open Source Software*. 2009.

[159] Vincent Rabaud and Maria Dimashova. *OpenCV ICP and RGBD visual odometry*. Class `RgbdICPOdometry` and function `RGBDICPOdometryImpl()` in the file https://github.com/Itseez/opencv_contrib/tree/master/modules/rgbd/src/odometry.cpp. 2012.

[160] Vincent Rabaud and Maria Dimashova. *OpenCV ICP-based visual odometry*. Class `ICPOdometry` and function `RGBDICPOdometryImpl()` in the file https://github.com/Itseez/opencv_contrib/tree/master/modules/rgbd/src/odometry.cpp. 2012.

[161] Vincent Rabaud and Maria Dimashova. *OpenCV RGBD module*. https://github.com/Itseez/opencv_contrib/tree/master/modules/rgbd. 2012.

[162] Vincent Rabaud and Maria Dimashova. *OpenCV RGBD-based visual odometry*. Class `RgbdOdometry` and function `RGBDICPOdometryImpl()` in the file https://github.com/Itseez/opencv_contrib/tree/master/modules/rgbd/src/odometry.cpp. 2012.

[163] Srikumar Ramalingam et al. “Manhattan Junction Catalogue for Spatial Reasoning of Indoor Scenes”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2013. DOI: 10.1109/cvpr.2013.394. URL: https://doi.org/10.1109/cvpr.2013.394.

[164] Carolina Raposo et al. “Plane-based Odometry using an RGB-D Camera.” In: *BMVC*. 2013. DOI: 10.5244/c.27.114. URL: http://dx.doi.org/10.5244/c.27.114.

[165] Michel Raynal. *Concurrent Programming: Algorithms, Principles, and Foundations*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642320260, 9783642320262. DOI: 10.1007/978-3-642-32027-9. URL: http://dx.doi.org/10.1007/978-3-642-32027-9.

[166] Julian Rosser, Jeremy Morley, and Gavin Smith. “Modelling of Building Interiors with Mobile Phone Sensor Data”. In: *ISPRS International Journal of Geo-Information* 4.2 (2015), pp. 989–1012. ISSN: 2220-9964. DOI: 10.3390/ijgi4020989. URL: http://www.mdpi.com/2220-9964/4/2/989.

[167] E. Rosten, R. Porter, and T. Drummond. “Faster and Better: A Machine Learning Approach to Corner Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (Jan. 2010), pp. 105–119. ISSN: 0162-8828. DOI: 10.1109/tpami.2008.275. URL: http://dx.doi.org/10.1109/tpami.2008.275.

[168] Carsten Rother. “A new approach to vanishing point detection in architectural environments”. In: *Image and Vision Computing* 20.9-10 (Aug. 2002), pp. 647–655. ISSN: 02628856. DOI: 10.1016/S0262-8856(02)00054-9. URL: http://linkinghub.elsevier.com/retrieve/pii/S0262885602000549.

[169] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE Computer Society, 2011, pp. 2564–2571. ISBN: 9781457711015. DOI: 10.1109/iccv.2011.6126544. URL: http://dx.doi.org/10.1109/iccv.2011.6126544.

[170] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *Proceedings of the third International Conference on 3-D Digital Imaging and Modeling*. IEEE. 2001, pp. 145–152. DOI: 10.1109/im.2001.924423. URL: http://dx.doi.org/10.1109/im.2001.924423.

[171] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE Computer Society, May 2011. DOI: 10.1109/icra.2011.5980567. URL: http://dx.doi.org/10.1109/icra.2011.5980567.

[172] Ahmed Saad, Ghassan Hamarneh, and Torsten Moller. "Exploration and visualization of segmentation uncertainty using shape and appearance prior information". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1366–1375. DOI: 10.1109/tvcg.2010.152. URL: http://dx.doi.org/10.1109/tvcg.2010.152.

[173] Juan Manuel Sáez, Francisco Escolano, and Miguel Angel Lozano. "Aerial obstacle detection with 3-D mobile devices". In: *IEEE journal of biomedical and health informatics* 19.1 (2015), pp. 74–80.

[174] Marta Salas et al. "Layout aware visual tracking and mapping". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015. DOI: 10.1109/iros.2015.7353367. URL: https://doi.org/10.1109/iros.2015.7353367.

[175] Renato F Salas-Moreno et al. "Dense planar SLAM". In: *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE. 2014, pp. 157–164. DOI: 10.1109/ismar.2014.6948422. URL: http://dx.doi.org/10.1109/ismar.2014.6948422.

[176] Aditya Sankar and Steve M. Seitz. "Interactive Room Capture on 3D-Aware Mobile Devices". In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology - UIST '17*. ACM Press, 2017. DOI: 10.1145/3126594.3126629. URL: https://doi.org/10.1145/3126594.3126629.

[177] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. "Kinect Range Sensing: Structured-Light versus Time-of-Flight Kinect". In: (2015). DOI: 10.1016/j.cviu.2015.05.006. arXiv: 1505.05459. URL: http://dx.doi.org/10.1016/j.cviu.2015.05.006.

[178] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In: *IEEE Robotics & Automation Magazine* 18.4 (2011), pp. 80–92. ISSN: 1070-9932. DOI: 10.1109/mra.2011.943233. URL: http://dx.doi.org/10.1109/mra.2011.943233.

[179] G. Schindler and F. Dellaert. "Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments". In: *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*. Vol. 1. IEEE, 2004, pp. 203–209. ISBN: 0-7695-2158-4. DOI: 10.1109/CVPR.2004.1315033. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1315033.

[180] T. Schöps, J. Engel, and D. Cremers. "Semi-Dense Visual Odometry for AR on a Smartphone". In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2014)*. IEEE Computer Society, Sept. 2014. DOI: 10.1109/ismar.2014.6948420. URL: http://dx.doi.org/10.1109/ismar.2014.6948420.

[181] Thomas Schops et al. "Real-Time View Correction for Mobile Devices". In: *IEEE Transactions on Visualization and Computer Graphics* 23.11 (Nov. 2017), pp. 2455–2462. DOI: 10.1109/tvcg.2017.2734578. URL: https://doi.org/10.1109/tvcg.2017.2734578.

[182] Thomas Schöps et al. “3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices”. In: *Proceedings of the International Conference on 3D Vision*. Piscataway, NJ: IEEE Computer Society, 2015.

[183] Tobias Schwarze and Martin Lauer. “Minimizing odometry drift by vanishing direction references”. In: (2015).

[184] Alexander G. Schwing et al. “Box in the Box: Joint 3D Layout and Object Reasoning from Single Images”. In: *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV2013)*. IEEE, Dec. 2013, pp. 353–360. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.51. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751153.

[185] Sensopia. *MagicPlan: Create a floor plan in just a few minutes*. http://www.magicplan.com/magicplan/. 2012.

[186] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. “Analysis and Modeling of Inertial Sensors Using Allan Variance”. In: *IEEE Transactions on Instrumentation and Measurement* 57.1 (Jan. 2008), pp. 140–149. ISSN: 0018-9456. DOI: 10.1109/TIM.2007.908635. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4404126.

[187] Yan Shuai et al. “Regression convolutional network for vanishing point detection”. In: *Automation (YAC), 2017 32nd Youth Academic Annual Conference of Chinese Association of*. IEEE. 2017, pp. 634–638. DOI: 10.1109/yac.2017.7967487. URL: http://dx.doi.org/10.1109/yac.2017.7967487.

[188] J.A. Shufelt. “Performance evaluation and analysis of vanishing point detection techniques”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.3 (Mar. 1999), pp. 282–288. DOI: 10.1109/34.754631. URL: https://doi.org/10.1109/34.754631.

[189] Nathan Silberman and Rob Fergus. “Indoor scene segmentation using a structured light sensor”. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 601–608. DOI: 10.1109/iccvw.2011.6130298. URL: http://dx.doi.org/10.1109/iccvw.2011.6130298.

[190] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576. DOI: 10.1109/cvpr.2015.7298655. URL: http://dx.doi.org/10.1109/cvpr.2015.7298655.

[191] Adam Stambler and Daniel Huber. “Building Modeling through Enclosure Reasoning”. In: *3D Vision (3DV), 2014 2nd International Conference on*. Vol. 2. IEEE. 2014, pp. 118–125. DOI: 10.1109/3dv.2014.65. URL: http://dx.doi.org/10.1109/3dv.2014.65.

[192] Jean Chan Stanek. “A characterization of starshaped sets”. In: *Journal canadien de mathématiques* 29.4 (Aug. 1977), pp. 673–680. DOI: 10.4153/cjm-1977-070-2. URL: https://doi.org/10.4153/cjm-1977-070-2.

[193] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. “Real-time visual odometry from dense RGB-D images”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 3. IEEE Computer Society, 2011, pp. 719–722. ISBN: 9781467300629. DOI: 10.1109/iccvw.2011.6130321. URL: http://dx.doi.org/10.1109/iccvw.2011.6130321.

[194] Jürg Stückler and Sven Behnke. “Multi-resolution surfel maps for efficient dense 3D modeling and tracking”. In: *Journal of Visual Communication and Image Representation* 25.1 (2014), pp. 137–147. ISSN: 10473203. DOI: 10.1016/j.jvcir.2013.02.008. URL: http://dx.doi.org/10.1016/j.jvcir.2013.02.008.

[195] Jürgen Sturm. *Tracking and Mapping in Project Tango*. Dagstuhl Seminar on Vision for Autonomous Vehicles and Probes. Nov. 2015.

[196] Jurgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2012)*. IEEE Computer Society, 2012, pp. 573–580. ISBN: 9781467317375. DOI: 10.1109/iros.2012.6385773. URL: http://dx.doi.org/10.1109/iros.2012.6385773.

[197] Niko Sünderhauf and Peter Protzel. "Switchable constraints for robust pose graph SLAM". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 1879–1884. DOI: 10.1109/iros.2012.6385590. URL: http://dx.doi.org/10.1109/iros.2012.6385590.

[198] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. Chap. Stereo correspondence.

[199] Y. Taguchi et al. "Point-plane SLAM for hand-held 3D sensors". In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 5182–5189. DOI: 10.1109/icra.2013.6631318. URL: http://dx.doi.org/10.1109/icra.2013.6631318.

[200] P. Tanskanen et al. "Live Metric 3D Reconstruction on Mobile Phones". In: *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Dec. 2013, pp. 65–72. DOI: 10.1109/iccv.2013.15. URL: http://dx.doi.org/10.1109/iccv.2013.15.

[201] Jean-Philippe Tardif. "Non-iterative approach for fast and accurate vanishing point detection". In: *Proceedings of the 2009 IEEE International Conference on Computer Vision (ICCV2009)*. IEEE, Sept. 2009, pp. 1250–1257. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459328. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459328.

[202] Keisuke Tateno, Federico Tombari, and Nassir Navab. "Real-time and scalable incremental segmentation on dense SLAM". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2015. DOI: 10.1109/iros.2015.7354011. URL: https://doi.org/10.1109/iros.2015.7354011.

[203] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Computer Vision, 1998. Sixth International Conference on*. IEEE. 1998, pp. 839–846. DOI: 10.1109/iccv.1998.710815. URL: http://dx.doi.org/10.1109/iccv.1998.710815.

[204] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen. "Planar surface SLAM with 3D and 2D sensors". In: *2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 3041–3048. DOI: 10.1109/icra.2012.6225287. URL: http://dx.doi.org/10.1109/icra.2012.6225287.

[205] Bill Triggs et al. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372. DOI: 10.1007/3-540-44480-7_21. URL: http://dx.doi.org/10.1007/3-540-44480-7_21.

[206] Enrique Valero, Antonio Adán, and Carlos Cerrada. "Automatic method for building indoor boundary models from dense point clouds collected by laser scanners". In: *Sensors* 12.12 (2012), pp. 16099–16115. DOI: 10.3390/s121216099. URL: http://dx.doi.org/10.3390/s121216099.

[207] M. Visvalingam and J. D. Whyatt. "Line generalisation by repeated elimination of points". In: *The Cartographic Journal* 30.1 (June 1993), pp. 46–51. DOI: 10.1179/000870493786962263. URL: https://doi.org/10.1179/000870493786962263.

[208] J. Weingarten, G. Gruener, and R. Siegwart. "A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction". In: 2003. URL: http://www.isr.uc.pt/icar03/.

[209] Jan W Weingarten, Gabriel Gruener, and Roland Siegwart. "Probabilistic plane fitting in 3D and an application to robotic mapping". In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 1. IEEE. 2004, pp. 927–932. DOI: 10.1109/robot.2004.1307268. URL: http://dx.doi.org/10.1109/robot.2004.1307268.

[210] Jan Weingarten and Roland Siegwart. "3D SLAM using planar segments". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 3062–3067. DOI: 10.1109/iros.2006.282245. URL: http://dx.doi.org/10.1109/iros.2006.282245.

[211] T. Whelan et al. “Robust Real-Time Visual Odometry for Dense RGB-D Mapping”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2013)*. Karlsruhe, Germany: IEEE Computer Society, May 2013. DOI: 10.1109/icra.2013.6631400. URL: http://dx.doi.org/10.1109/icra.2013.6631400.

[212] H. Wildenauer and A. Hanbury. “Robust camera self-calibration from monocular images of Manhattan worlds”. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2012)*. IEEE, June 2012, pp. 2831–2838. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6248008. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6248008.

[213] Horst Wildenauer and Markus Vincze. “Vanishing Point Detection in Complex Man-made Worlds”. In: *Proceedings of the 2007 International Conference on Image Analysis and Processing (ICIAP 2007)*. IEEE, Sept. 2007, pp. 615–622. ISBN: 0-7695-2877-5. DOI: 10.1109/ICIAP.2007.4362845. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4362845.

[214] Jianxiong Xiao and Yasutaka Furukawa. “Reconstructing the world’s museums”. In: *International journal of computer vision* 110.3 (2014), pp. 243–258. DOI: 10.1007/s11263-014-0711-y. URL: http://dx.doi.org/10.1007/s11263-014-0711-y.

[215] Xuehan Xiong et al. “Automatic creation of semantically rich 3D building models from laser scanner data”. In: *Automation in Construction* 31 (2013), pp. 325–337. DOI: 10.22260/isarc2011/0061. URL: http://dx.doi.org/10.22260/isarc2011/0061.

[216] Yiliang Xu, Sangmin Oh, and Anthony Hoogs. “A Minimum Error Vanishing Point Detection Approach for Uncalibrated Monocular Images of Man-Made Environments”. In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2013)*. IEEE, June 2013, pp. 1376–1383. ISBN: 978-0-7695-4989-7. DOI: 10.1109/CVPR.2013.181. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6619025.

[217] Daoqi Yang. *C++ and Object-Oriented Numeric Computing for Scientists and Engineers*. New York, NY: Springer New York, 2001. ISBN: 978-1-4612-6566-5. DOI: 10.1007/978-1-4613-0189-9. URL: http://link.springer.com/10.1007/978-1-4613-0189-9.

[218] Shichao Yang, Daniel Maturana, and Sebastian Scherer. “Real-time 3D scene layout from a single image using Convolutional Neural Networks”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. DOI: 10.1109/icra.2016.7487368. URL: https://doi.org/10.1109/icra.2016.7487368.

[219] Shichao Yang et al. “Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments”. In: *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016. DOI: 10.1109/iros.2016.7759204. URL: https://doi.org/10.1109/iros.2016.7759204.

[220] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics”. English. In: *Intelligent Industrial Systems* 1.4 (2015), pp. 289–311. ISSN: 2363-6912. DOI: 10.1007/s40903-015-0032-7. URL: http://dx.doi.org/10.1007/s40903-015-0032-7.

[221] Menghua Zhai, Scott Workman, and Nathan Jacobs. “Detecting Vanishing Points Using Global Image Context in a Non-ManhattanWorld”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.610. URL: https://doi.org/10.1109/cvpr.2016.610.

[222] Edward Zhang, Michael F. Cohen, and Brian Curless. “Emptying, refurnishing, and relighting indoor spaces”. In: *ACM Transactions on Graphics* 35.6 (Nov. 2016), pp. 1–14. DOI: 10.1145/2980179.2982432. URL: https://doi.org/10.1145/2980179.2982432.

[223] Ji Zhang, Michael Kaess, and Sanjiv Singh. "A real-time method for depth enhanced visual odometry". English. In: *Autonomous Robots* 41.1 (2015), pp. 31–43. ISSN: 0929-5593. DOI: 10.1007/s10514-015-9525-1. URL: http://dx.doi.org/10.1007/s10514-015-9525-1.

[224] Ji Zhang, Michael Kaess, and Sanjiv Singh. "Real-time depth enhanced monocular odometry". In: *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE Computer Society, Sept. 2014, pp. 4973–4980. ISBN: 978-1-4799-6934-0. DOI: 10.1109/IROS.2014.6943269. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6943269.

[225] Ying Zhang, Chuanjiang Luo, and Juan Liu. "Walk&Sketch". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*. ACM Press, 2012. DOI: 10.1145/2370216.2370285. URL: https://doi.org/10.1145/2370216.2370285.

[226] Huizhong Zhou et al. "Structslam: Visual slam with building structure lines". In: *IEEE Transactions on Vehicular Technology* 64.4 (2015), pp. 1364–1375. DOI: 10.1109/tvt.2015.2388780. URL: http://dx.doi.org/10.1109/tvt.2015.2388780.

[227] S. Zlatanova et al. "Problems In Indoor Mapping and Modelling". In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-4/W4 (Nov. 2013), pp. 63–68. DOI: 10.5194/isprsarchives-xl-4-w4-63-2013. URL: https://doi.org/10.5194/isprsarchives-xl-4-w4-63-2013.