République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie HOUARI BOUMEDIENE
Faculté d'informatique

Thèse
Présentée pour l'obtention du **diplôme** de **DOCTORAT** 3$^{\text{ème}}$ cycle
En : Informatique
Spécialité : Intelligence Artificielle
Par : IBELAIDEN Farah

Sujet

Positionnement visuel de caméra Basée sur une collection d'images RGBD

Soutenue publiquement, le 03/05/2023, devant le jury composé de :

Mme.	ACHOUR Nouara	Professeur	à l'USTHB	Présidente
M.	LARABI Slimane	Professeur	à l'USTHB	Directeur de thèse
Mme.	LAICHE Nacera	Maître de Conférences A	à l'USTHB	Examinatrice
Mme.	DAHMANI Djamila	Maître de Conférences A	à l'USTHB	Examinatrice
M.	DJEKOUNE Walid	Maître de Conférences A	à CDTA	Examinateur

This thesis is dedicated to:
The sake of Allah, my Creator and my Master;
My great teacher and messenger, Mohammed (May Allah bless and grant him);
My supervisor, Professor Slimane Larabi, whose expertise was invaluable in formulating the research questions and methodology. I would also thank him for his invaluable advice, continuous support, and patience during my PhD study;
My thesis committee president, Professor ACHOUR, for making her time available to read my thesis. I also thank Doctor LAICHE , Doctor DAHMANI and Doctor DJEKOUNE for their availability in reading and analyzing my work;
My family and many friends. A special feeling of gratitude to my loving parents whose words of encouragement and push for tenacity ring in my ears;
My husband, you have been a listener and a supporter of all my endeavors. Your partnership, steadfastness, and love sustain me;
My daughter, Eline, remember that all things are possible. Never be afraid to pursue your dreams and goals. You are precious gift from Allah. I love you without measure;
My brother and sisters, thank you for being there for me throughout the entire doctorate program. Missiva, Tin hinen and Ferhat you gave quiet encouragement and positive belief in my success that kept me going regardless of the challenge that I faced.

Abstract

Our aim in this thesis is to provide an automatic visual positioning system from depth images, to be used in different fields such as: robotics, virtual and augmented reality, visually impaired localization.

The work developed through our research focused on the architectural scene features. Hence, it fits into the class of automatic positioning methods based on the architecture that has recently appeared. Two objectives have been achieved through our thesis. The first one is the proposal of scene descriptor based on the architectural features uninfluenced by the external factors. The second one is the development of place recognition method based on this descriptor to identify the unknown location whose dataset does not need updates.

The exposed system takes as input a depth video acquired by the surrounding kinect sensor in the scene, divides it to select a set of key depth images, from which the planes are located and aligned to gradually calculate the 3D scene model. The 2D map is then derived and the scene descriptor is deduced and constitutes the core of the visual positioning algorithm.

The evaluation thus discloses the accuracy of the proposed descriptor in addition to its invariance and its stability against scenery changes, which justifies the efficiency of the visual positioning system presented.

Résumé

Notre objectif dans cette thèse est de proposer un système de positionnement visuel automatique à partir d'images de profondeur, utilisable dans différents domaines tels que : la robotique, la réalité virtuelle et augmentée, la localisation des malvoyants.

Le travail développé s'inspire des caractéristiques de la scène architecturale. Il s'inscrit donc dans la classe des méthodes de positionnement automatique basées sur l'architecture apparue récemment.

Deux objectifs ont été atteints à travers notre thèse. Le premier est la proposition de descripteur de scène basé sur les caractéristiques architecturales non influencées par les facteurs externes. Le second est le développement d'une méthode de reconnaissance de lieu basée sur ce descripteur pour identifier l'emplacement inconnu et dont l'ensemble de données n'a pas besoin de mises à jour.

Le système exposé prend en entrée une vidéo de profondeur acquise par le capteur kinect environnant dans la scène, la divise pour sélectionner un ensemble clé d'images de profondeur à partir desquelles les plans sont localisés et alignés pour calculer progressivement le modèle de scène 3D. La carte 2D est alors dérivée et le descripteur de scène est déduit et constitue le cœur de l'algorithme de positionnement visuel.

L'évaluation révèle ainsi la justesse du descripteur proposé ainsi que son invariance et sa stabilité face aux changements de décor, ce qui justifie l'efficacité du système de positionnement visuel présenté.

Table of contents

Table of Figures

Liste of Tables

1 Introduction

In last decades, Computer vision has been a relevant interdisciplinary research field. Developers and researchers have tried to mimic several human visual system aspects; most of them are very easy for humans whereas they are ultra-intricate to reproduce in an artificial system.

The large progress in computer vision tasks render them integral part of many intelligent systems enabling the visual information to be understood and interpreted.

One of the most fruitful topics of computer vision is undoubtedly "the visual positioning" or in other words "The place recognition" that aims to localize a camera in an unknown environment from several acquired views. It constitutes a challenge of great utility in providing a low-cost means for localization because of its utility in several applications from which we quote:

- Security and people tracking;
- Augmented reality (museum visits, games on scenes acquired by camera;
- Visually impaired localization.
- Intelligent robots.

The visual positioning topic may encompass many research directions (features description and detection, three dimensional scene reconstruction, objects detection and recognition); it may concern outdoor or indoor scenes. Many high-accuracy methods have been proposed for outdoor scenes, however they perform less well indoors due to the structure of the environment. This is why visual positioning in indoor scenes where people spend ninety percent of their time [1] remains a challenge for researchers.

The proposed methods for visually positioning may be differentiated according to their objectives into systems that recognize previously visited places and systems that calculate the $6DOF$ position (six degrees of freedom). Despite the difference of systems output, they intersect in scene representations (descriptors) which constitute their kernel. The database containing these descriptors is most of the time acquired at a single period, that's why the query may be non retrieved by the system years after. Permanent local changes of environment induce the need to update the dataset to take into account the modifications of scene [2], consequently, this expensive operation must be avoided. This is the factor that has been unkempt-ed by state-of-the-art methods focusing solely on improvements in positioning approaches.
In this insight, we aim through this thesis to overcome this inconvenient by proposing an architecture-based visual positioning method to recognize previously visited places invariant to scene modifications.
.

1.1 The goal

Through this thesis our goal is to propose a visual positioning system for visually impaired and blind people that permits them to localize themselves for autonomy purposes. The propounded system may also be used as a part of many applications of different domains (robotics, artificial intelligence, ...). It takes as input a sequence of depth images acquired by a depth sensor (Kinect version 2) and delivers the location of the depth camera. To do this, it retrieves the calculated scene descriptor from a set of location descriptors. Our main challenge is to propose a scene descriptor invariant to scene changes and compact for visual search.

1.2 Overview of our system

In this we propose a new system for place recognition based on the scene architecture (see figure 1.1).

It is constituted of four main modules:

- Input module: acquires partial depth views using the depth sensor "kinect version 2".
- Pre-processing module: denoises the acquired data.
- Processing module: includes four sub-modules:
  - The planes identification sub-module: takes as input the depth views and locates the planes of each one.
  - The registration sub-module: aligns the resulted polygons of the partial views to compute gradually the 3D map.
  - The 2D map determination sub-module: derives the 2D map from the computed 3D map to describe it geometrically.
  - The descriptors matching sub-module: compares the query descriptor against labeled dataset scene descriptors in order to identify the location.
- Output module: delivers the place recognition's result.

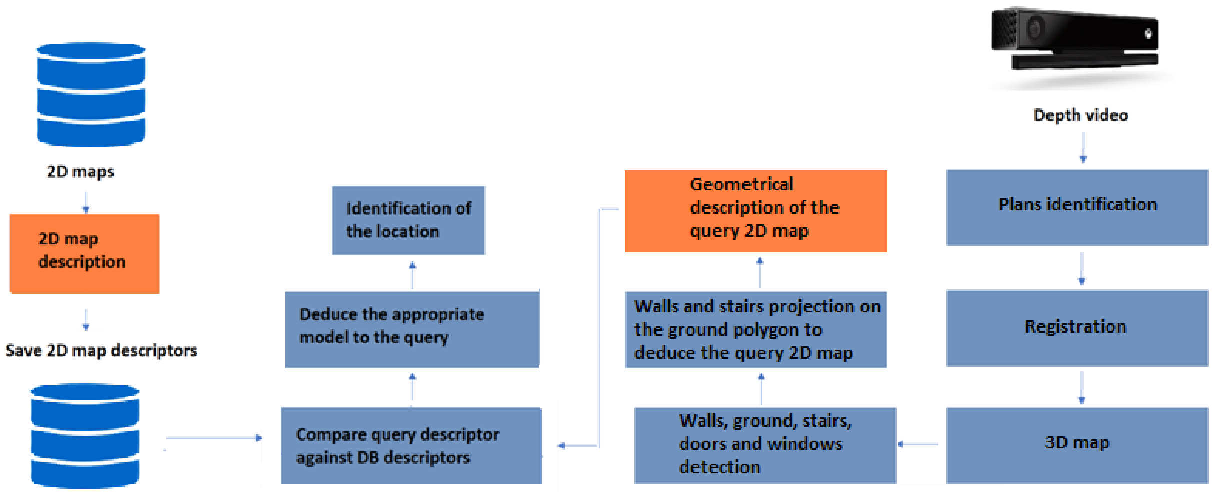
2D maps
2D map description
Save 2D map descriptors
Identification of the location
Deduce the appropriate model to the query
Compare query descriptor against DB descriptors
Geometrical description of the query 2D map
Walls and stairs projection on the ground polygon to deduce the query 2D map
Walls, ground, stairs, doors and windows detection
Depth video
Plans identification
Registration
3D map

Figure 1.1: The proposed system overview.

1.3 Summary of contributions

Our main contributions are:

- An architecture based scene descriptor from depth video by considering the scene delimitation, the windows and doors positions and dimensions, stairs characteristics on the basis of architectural norms.
- Visual positioning method based on non-rearrangeable parts of the scene (walls, stairs) that represent stable features over time, independent of scenery changes.

1.4 Outline of the dissertation

The remaining chapters are structured as follows:

- Chapter 2 is devoted to the presentation of the human scene interaction systems and components, the classification of the visual positioning methods and the scenes descriptors. As well as a review of the depth sensors and the relevant algorithms utilized by state of the art visual localization systems.
- Chapter 03 focuses on detail of different steps followed for architecture based scene descriptor computation and introduces the propounded visual positioning system founded on scenes descriptors matching.
- Chapter 04 exposes the system's implementation details and used tools, the dataset besides the results of the proposed system evaluation and its comparison to state of the art approaches.
- Then the thesis is concluded with a general conclusion and a set of future works.

2

Human-Scene Interaction and Visual Positioning Systems

2.1 Introduction

Computer systems may be broadly categorized on two categories. The first, for human-Scene interaction systems, and the second, for human computer interaction systems (the scripts).
The first category of systems provide users with the ability to interact with scenes by generating outputs in order to navigate and understand its content. While the the second category of systems gives users the possibility to interact with the computers in order to support their productivity and safety [3].
As our thesis topic fits into the first category, we will review through this chapter, the Human scene interaction systems proposed in the state of the art in subsection 2.2.1 and we summarize in 2.2.2 their common components and we will devote the next subsection (2.3) to the visual positioning.

2.2 Human Scene Interaction (HSI) Systems

2.2.1 HSI Systems: A review

The goal of human scene interaction systems is to extract from the scene any meaningful information received from the real life using acquisition sensors to transform it into instructions in order to enhance the user life in different domains from which we quote:

Navigation

Defined as the process of determining a proper and safe path for a robot to travel between an initial and final point, it is subdivided into indoor navigation and outdoor navigation.

The Outdoor navigation is also subdivided into structured and unstructured environments, while the indoor navigation is subdivided into map building based navigation and mapless navigation.

a)- Indoor navigation: Map based Navigation

The systems of this class explore the environment and build its map to start the navigation process.

In [4] The system contains two main modules built up with neural networks to identify edges, detect walls and output the proper steering commands to guide the robot at a distance of a wall or centered in a corridor.

[5] utizes RGB-D sensor to detect obstacles and free paths; both depth and color information were used to infer the presence or absence of obstacles.

[6] [7] offer a real-time scene reconstruction by using either stereo cameras (stereoSLAM) or a single camera (monoSLAM) [8] to infer the geometric model of a scene, and furnish the camera trajectory [9]. [10] proposed a fuzzy navigation system which was propounded to allow mobile robot to reach its full autonomy (by navigating, avoiding obstacles, follow moving object and construct a dense 3D map). Figure 2.1 depicts an illustrative example of mobile robot navigation with the constructed 3D map.

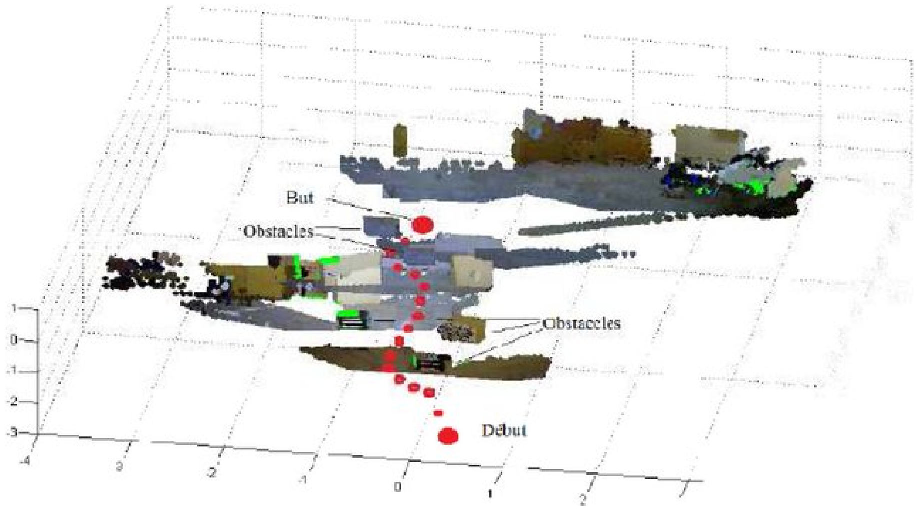

Figure 2.1: The results of mobile robot map based indoor navigation system with the constructed 3D map, Source: [10].

b)- Indoor navigation: Mapless based Navigation

Unlike map based navigation systems, these systems do not require an environment map as proposed in [11] and [12] that are based on appearance for robot navigation. [11] relies on a Bag-of-Visual-Words constructed from a database of Speeded Up Robust Features (SURF) extracted from scene images to provide matching between previously visited places as well as a measure of the probability of being in a new location.

In [13], corners are detected from RGB image using Harris and Stephens corner detector. Then, the image is divided into three regions of interest: center, left and right. After that, obstacles detection algorithm is applied to identify if the central region is safe. If this latter is not safe, they identify the free walkable space (left or right) and provide the user with the right orientation instructions for navigation.

And in [14] a deep neural network based visual navigation system has been proposed to allow the navigation from a few snapshots of the environment and directional guidance (see figure 2.2).

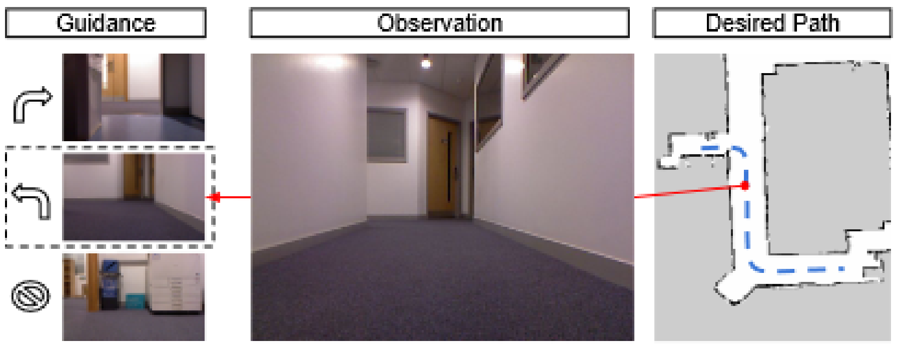

Figure 2.2: An example of indoor mapless based Navigation system with sparse directional guidance. The robot automatically selects from the provided guidance based on current observation. Source: [14]

c)- Outdoor navigation: In structured environments

It refers to the ability to detect and follow the lines of the road in order to navigate consistently [15] [16](see figure 2.3).

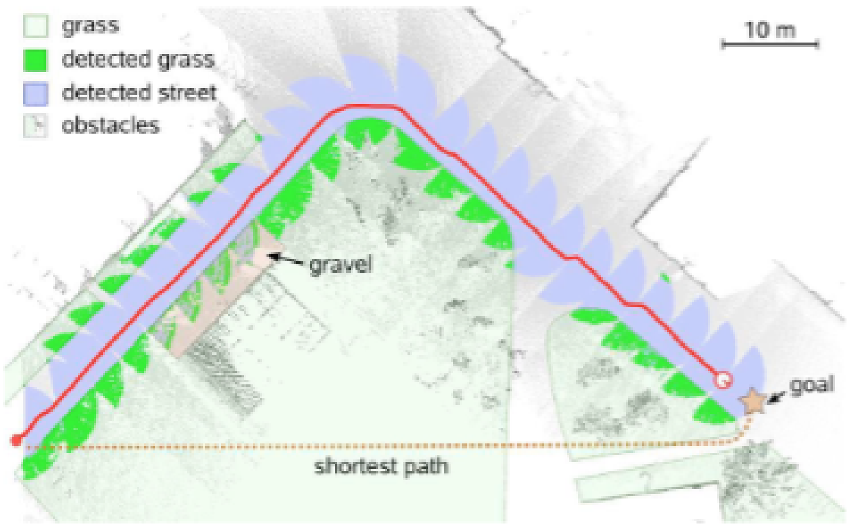

Figure 2.3: Outdoor navigation in structured environments, Source: [15]

In [17], authors propounded a system for obstacle avoidance in outdoor environments that uses RGB camera. In order to identify the most common outdoor obstacles (people, cars, bicycles,motorbikes, buses, traffic signs...) for a safe navigation.

[18] proposed a framework for a route following in an outdoor environment using a color camera and laser range scanner to capture environment images. The system firstly

produces a route description from the image data which is transformed by the reasoning module into world coordinates in order to compute the trajectory of the robot.

Among the methods based on Global Positioning System (GPS) which rely on information provided by satellite-based navigation methods, we cite [19] where an approach has been introduced based on GPS to provide the position and navigation instructions through an accessible keyboard and a speech synthesis interface to target the needs of visually impaired people.

d)- Outdoor navigation: In unstructured environments

There are two main contexts:

- The robot randomly explores the vicinity . In [20] an Efficient path search and trajectory optimization methods are proposed to generate trajectories, which can effectively avert different obstacles in off-road environments, such as dynamic obstacles, to reach the specified destination.
- The robot fulfills a mission with a target position. In this case, a map of the environment must firstly be created [21] and a localization algorithm is also required [22].

Object detection

Deals with identification and localization of objects present in images and videos. It depicts the most important and elementary tasks of many applications such as: Visual positioning (knowing the objects present in the scene allow us to recognize the location), self-driving cars, objects tracking.
R-CNN [23] is deep learning-based object detection method. It was proposed in order to decrease the complexity of Exhaustive Search algorithms by a selective search that extract approximately 2000 regions called "region proposals" from the image to reduce number of considered locations but it was still slow that's why the Fast R-CNN [24] was propounded as an amelioration of R-CNN where the convolutional operation is done once per image. SSD "Single Shot Detector" [25] has introduced a single multibox detector. It is faster than the previous object detection models such as R-CNN and its variants and the first version of YOLO "You Only Look Once".
YOLO [26] has a single neural network that predicts bounding boxes and class probabilities directly from entire images in one evaluation. It is extremely fast but it fails with the

small objects and this was solved later in versions: Yolo v2, Yolo v3 [27], Yolo v4 [28], Yolo v5 , Yolo v6 [29] and Yolo v7 which is the latest official Yolo version created by the original authors of the Yolo ensuring the trade-off between speed and accuracy. In general, it surpasses all previous object detectors in terms of both speed and accuracy [30] [31].

Human computer interaction

Some assistive systems were proposed to ameliorate the Human-Computer interactions (the computer is a part of the scene). As examples, we cite the eye-tracking systems that measure eye position, eye movement, and pupil size to detect zones in which the user has a particular interest at a specific time. There are a number of methods for measuring eye movement, as the optical methods, in which light, typically infrared, is reflected from the eye and sensed by a camera or some other specially designed optical sensor. The data is then analyzed to extract eye rotation from changes in the reflections [32].

In [33] a system for analyzing the structure of a web page based on visual information has been propounded by using the hierarchical segmentation reflecting the visual structure of the rendered page after converting it into image. Using image processing and computer vision techniques, the system identifies the edges and segment the input page into semantic categories without the need to know the programming languages or the tags used to write these components.

2.2.2 HSI: The components

The Human scene interaction systems components are generally four:

Input device:

The input device type impacts on the system accuracy and the running time. It specifies the aid system type which may be:

- Depth-based system: that uses depth sensors endowed of a depth camera which may be structured light based such as Microsoft Kinect v1 [34] and Asus Xtion [35] or time of flight scanners [36] [34], such as Microsoft Kinect v2 and ultrasonic [37].These systems use depth images that contains distances from the object to the camera.
- RGB-D based system takes as input RGB and depth images produced from the RGB and depth cameras.

- Stereo-based system uses a stereo camera with two lenses and a separate image sensor for each lens to allow the camera to simulate human binocular vision, and therefore gives it the ability to obtain the depth.
- RGB-based system uses RGB images of color cameras.

Output device

Depends on the input device. Most systems furnish their outputs in audio based interface [38] while the others provide it on vibration [39] or touch based interface [40].

Pre-processing

Pre-processing stage includes techniques that aim to produce accurate data by noise removal and redundancy reduction using a set of techniques such as normalization, data augmentation, grayscale conversion and image standardization. in order to furnish suitable data to algorithms properties.

Processing

Image processing includes methods that aim to perform operations on an image, in order to ameliorate its quality or extract from it the pertinent meaningful information. It encompasses techniques of image processing, computer vision (segmentation [41]), artificial intelligence (SVM) [42] , and augmented reality [43].

2.3 Visual positioning

For computer vision and robotics, the visual positioning covers many nominations such as image-based localization, place recognition, camera localization.

The visual positioning systems may be differentiated according to their outputs into:

(1) systems that catch the visual ability of humans and robots to recognize already visited places [44] [45] [46] [47].

(2) systems that output a position and orientation of the visual acquisition device [45] [48] [49].

The two main problems that encounter these systems are: how to excerpt the steady features over time? and how to render these features insensitive to the variation of illumination and differences of view angles. These issues make the problem of localization challenging and place the scene representation in its nucleus because it's a wrench of a robust approach of localization. That's why we initiate this section by introducing the scene representation in 2.3.1 Then we present in 2.3.2 the visual positioning systems classification.

2.3.1 Scene representations

The features used in state of the art of visual positioning systems may be categorized into:

a) Global features

That considers images as whole. They include:

a.1) Learned features

That become more popular with the advances of neural networks; [50] [51], [52], [53] and [54] have demonstrated that these later achieved good accuracy in the image retrieval task. Figure 2.4 illustrates an example of learned features.

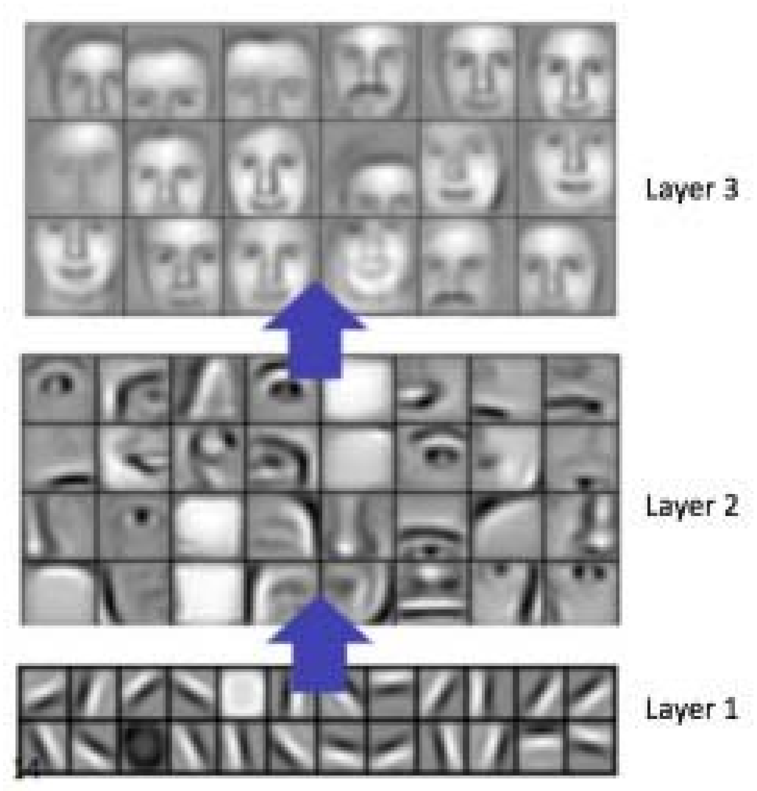

Figure 2.4: Learned features example [55]

b) Local features

Their Description occurs at pixel level and encompass:

b.1) Geometric features

Representing geometric primitive shapes semantically meaningful (like planar surfaces and line segments). In [56] authors described scenes by lines and planar surfaces as shown in figures 2.5 and 2.6.

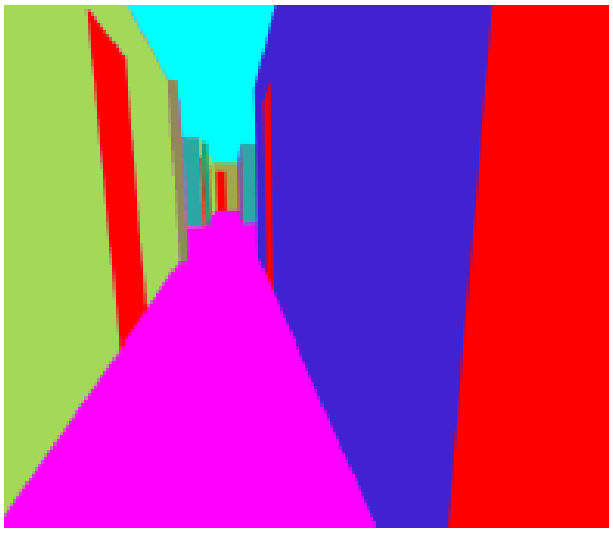

Figure 2.5: Example of planar surfaces that can act as geometric features [56]

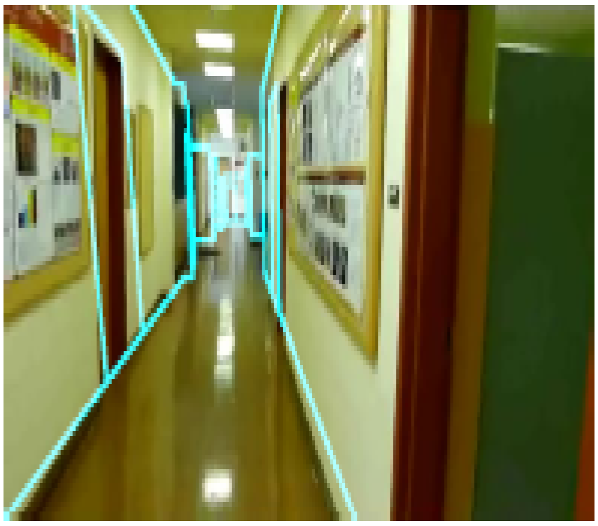

Figure 2.6: lines detected in an indoor scenes [57].

b.2) Point features

In Visual positioning SIFT (Scale-Invariant Feature Transform) [58] is used more than SURF (Speeded Up Robust Features) [59]. It offers high precision but can not satisfy requirements for real time applications because of its high run-time [50]. ORB (Oriented FAST and Rotated BRIEF) [60] is comparable to SIFT [58] in term of precision but it is significantly faster to compute [61]. An illustrative example of the above mentioned features is presented in figure 2.7.

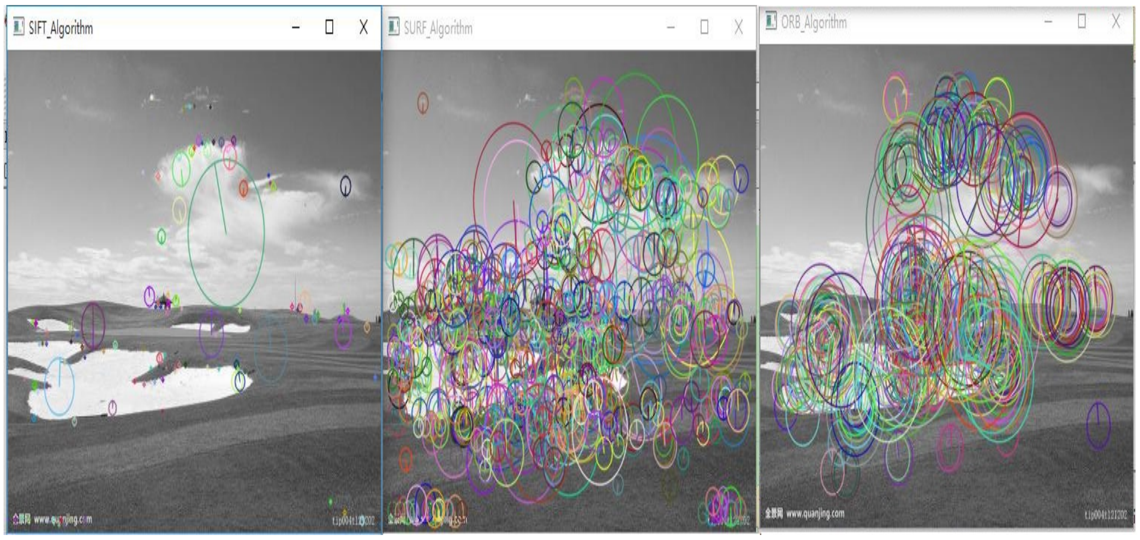

Figure 2.7: from right to left: ORB, SURF, SIFT features extracted from the same image [62].

c) Descriptors extracted from depth image

c.1) Point cloud descriptor

Also noted shape descriptor, it captures the surface geometry of a point and its surrounding neighbors. These descriptors may be categorized into:

c.1.1) Local descriptors from which we quote:

Point Feature Histograms (PFH) descriptor It is the most important descriptor. It tries to capture information of the geometry surrounding the point by analyzing the difference between the directions of normal in the vicinity as shown in figure 2.8. However it is too computationally expensive for that reason it is not suitable for real time applications [63] [64].

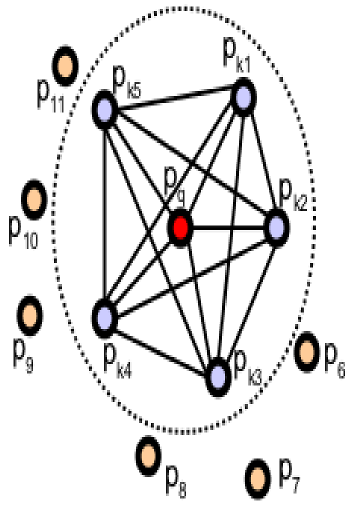

Figure 2.8: Point pairs established when computing PFH for a point [65].

Fast Point Feature Histograms (FPFH) descriptor Considers only the direct connections between the current key point and its neighbors, without additional links between neighbors (like it's represented in figure 2.9) [66] [67].

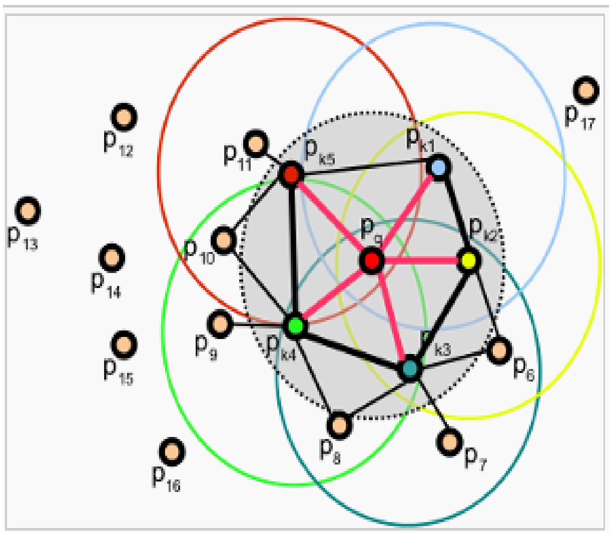

Figure 2.9: Point pairs established when computing the FPFH for a point [65].

c.1.2) Global descriptors

Compute a single feature vector for the entire input point cloud. They rely on the observation of the entire geometry and fail in environments with occlusions. They are suitable for 3D object recognition, geometric categorization and shape retrieval applications.

The Viewpoint Feature Histogram (VFH) is based on the FPFH. It is invariant to the object's pose and it is estimated for the whole cluster, not for every point. But FPFH is not robust to occlusion [68].

Clustered Viewpoint Feature Histogram (CVFH) descriptor Instead of computing a single VFH histogram for the whole cluster, the image is firstly divided into smooth regions using region-growing segmentation. Then, the VFH histogram is computed for every region [69].

The Ensemble of Shape Functions (ESF) descriptor It is a combination of three different shape functions that describe certain properties of points cloud: distances, angles and area. This descriptor is very unique because it does not require normal information. It does not need any pre-processing, as it is robust to noise and incomplete surfaces [70].

c.2) Depth image descriptor

c.2.1) Binary descriptors

Like LBP used in [71] and its variants (CLBP, CS-LBP, CS-LDP, XCSLBP) that are considered as texture descriptors. They codify local primitives (curved edges, spots and flat area), figure 2.10 represent an illustration of the basic LBP operator.

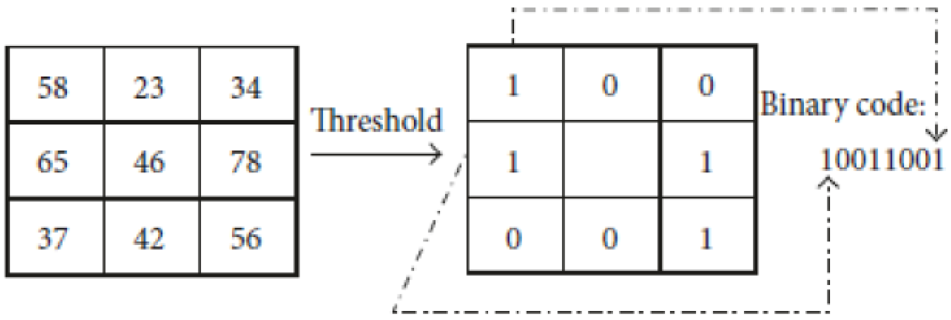

Figure 2.10: Illustration of the basic LBP operator.

d) Hybrid features

Consists of features that are neither local nor global, they combine several types of descriptors as:

d.1) Patch features

Considering region of interest in an image. In [49] [72] [73] authors have demonstrated their efficiency. Figure 2.11 show an example.

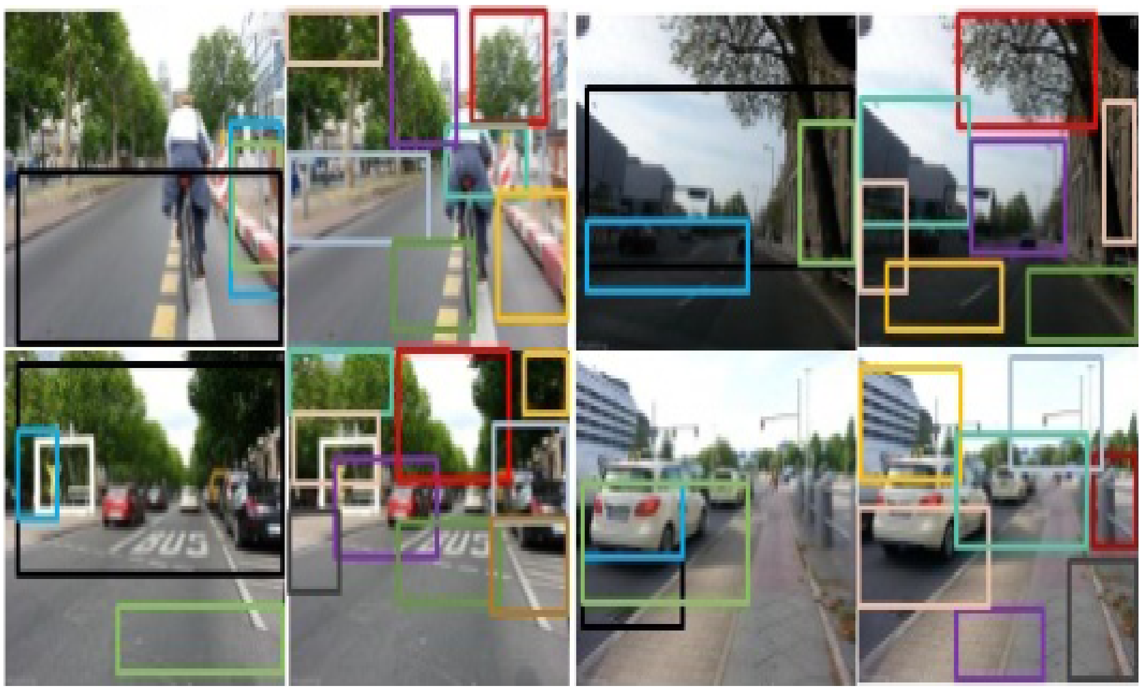

Figure 2.11: Example regions of interest.

d.2) Combined features

Uses an approach to pair various descriptors in order to increase the result of the retrieval step [67] [56] [53].

2.3.2 Visual positioning systems classification

The visual localization methods can broadly be divided into:

Features based methods

They may be classified into: 2D based methods, 3D based methods and topological based methods.

The 2D based methods have tackled the problem of visual localization as an image retrieval problem [74]. Figure 4.2.4 gives an overview of this system class.

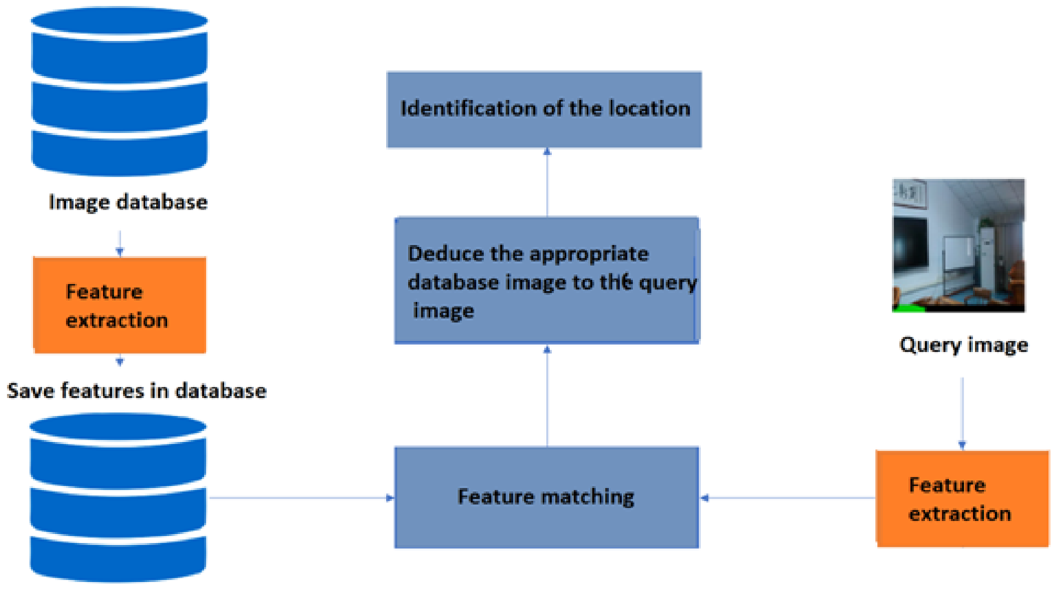
Image database
Feature extraction
Save features in database
Identification of the location
Deduce the appropriate database image to the query image
Feature matching
Query image
Feature extraction

Figure 2.12: 2D based methods overview.

[47] addresses one of the key problems in place recognition that is the presence of commons objects to different places (trees, road markings...); and demonstrates that the localization can be significantly improved by automatic identification and suppression of these objects from database images. Whereas, authors in [44] have proved that these commons structures can form distinguishing features for many places by a modification of their weights in the bag of visual word model.

The 3D based methods require three dimensional model (3D) of the scene, a large dataset of features extracted from database images used for 3D reconstruction [48] [45] and efficient retrieval method to search the most similar dataset image to the query to be used for the 6DOF pose computation. [75] [46] (see figure 2.13).

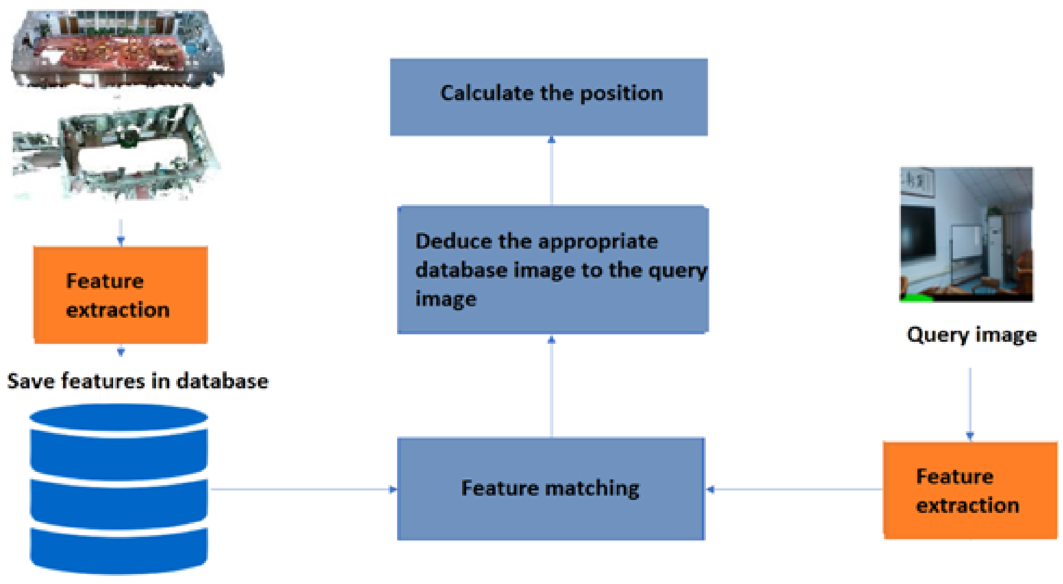

Figure 2.13: 3D based methods overview.

We distinguish from them the 2D-3D based methods as in [49] where an approach has been proposed to compute the 6DOF position of a camera using perspective-n-point (PNP) algorithm [76]. The results have demonstrated the precision of computed position. Whereas, in [48] the 6DOF position of the camera has been computed on the basis of epipolar constraint, the efficiency of the proposed solution decreases as the area of indoor scene increases due to accumulated alignment errors. The 3D-2D based methods match features of the 3D model against features of the query image. In [77] authors have compared between 2D-3D based methods and 3D-2D based methods and affirm that 3D-2D search reaches a better effectiveness but it is slower than 2D-3D search notably for large scale scenes because model features are more than query features. Based on this insight, authors in [78] attempted to prove that 3D-2D matching can be more quickly than 2D-3D through their modified 3D-2D based matching approach, however results showed significant reduction in term of precision.

The topological based methods are founded on adjacency (topological) maps (see figure 2.14) that depict the environments by nodes representing locations to which a set of reference images are assigned and arcs for the adjacency relationships between these locations. Like in [79] where a real time system was proposed to track the mobile robot's position by limiting the search of the most similar reference image to the query image on the currently believed location and its immediate neighbors identified from the topological map to reduce the localization time.

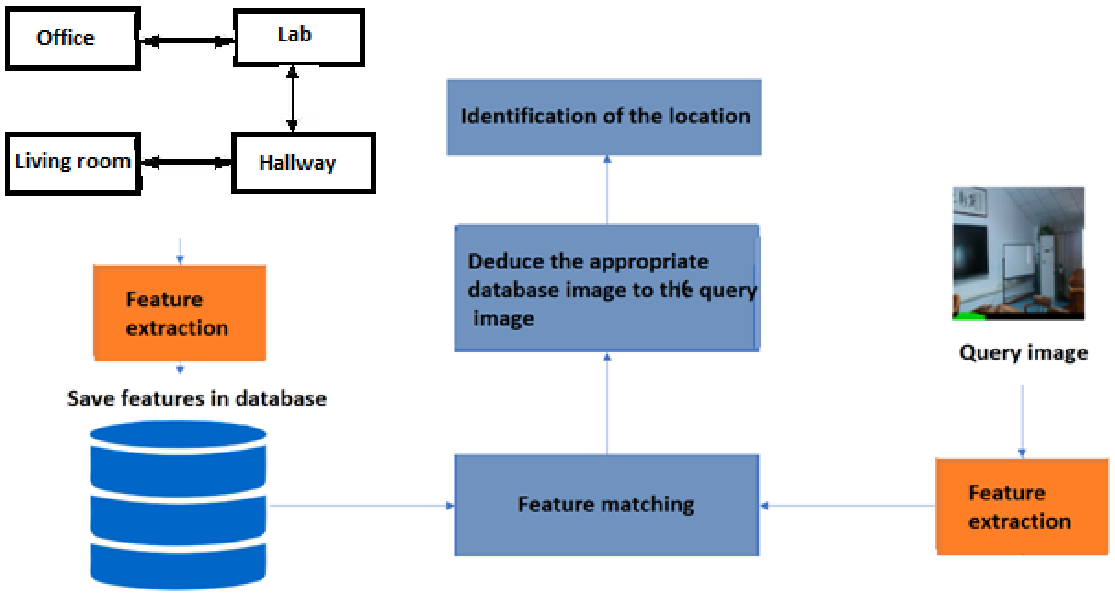
Office
Lab
Living room
Hallway
Feature extraction
Save features in database
Identification of the location
Deduce the appropriate database image to the query image
Feature matching
Query image
Feature extraction

Figure 2.14: topological based methods overview.

The neural network based methods

Use features extracted from a network as an image representation [80]. They may be decomposed into: (1) Methods that tackle the problem of place recognition as classification problem [81] as in [82] where authors propounded enhancement on the retrieval stage of the visual localization system [83] by a combination of multiple learning-based feature extractors. (2) Methods that deal the problem of localization as regression of pose estimation. Such as in [84] where researchers have proposed a simple algorithm that consists of a convolutional neural network trained end to end to regress the camera orientation and position in real time.

Segmentation based methods

Use results of objects segmentation as basis information for place recognition (for example if there is a bed it is a bedroom) [85] (see figure 2.15).

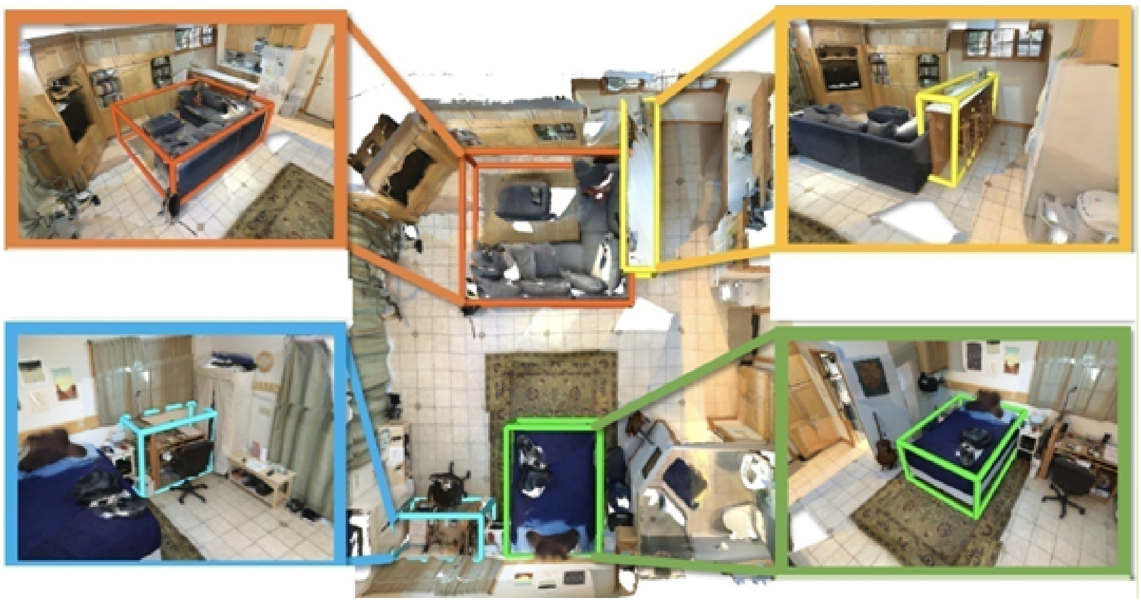

Figure 2.15: Segmentation based methods [85].

Architecture based methods

Appeared recently, founded on the scene architecture. In [86], a robot positioning system has been proposed using a convolutional neural network to predict edges of image and compute robot position by matching the extracted edges to the environment floor-plane. In our presented work in [87], an architecture-based scene descriptor has been proposed from partial depth views (captured by surrounding kinect). The planes of each view were identified and aligned with those of the previous frames in order to build gradually the 3D scene model; Then walls will be located and projected on the ground to define the 2D (two dimensional) map which will be described geometrically to define the scene descriptor used to recognize the scene. The proposed descriptor has been improved by including the information of windows, doors, and stairs on the basis of simple architectural norm in order to increase the descriptor discrimination [88]. Details will be given in next chapters.

According to the type of input data, methods of all aforementioned classes may be assembled into (1) methods that only use RGB image [47] [89]; (2) methods that use depth image [80]; (3) Category of those that associate depth image with other input data: like in [90] [52] [57] where authors used depth image with RGB image, and [67] where IR image were associated to the depth image.

2.3.3 Discussion

Despite all the improvements made to feature-based methods, they remain highly sensitive to high light variations and large modifications of decor that are frequent in daily life. Learning based methods require a large amount of data for the training phase, which is costly in terms of computation time. Furthermore, their generalization ability in large dynamic indoor environments is low. Movable objects (such as chairs) induce the accuracy decrease of segmentation based place recognition methods because they do not meet the criterion of stability over time. Lately, a lot of interest has been attracted towards architectural features because they are unchangeable and stable.

Most of the proposed methods require data-set updates due to the standing local modifications of the indoor environment. RGB images are more informative then depth images however they are considered as time consuming. Depth images are more suitable for applications designed for real time because of their significantly low computational runtime and their less sensitivity to lighting changes.

2.4 Conclusion

As the thesis topic "visual positioning " fits under the category of human scene interaction systems, We have firstly presented through this chapter some relevant systems of this category.
Then, we have presented an in- depth studies of the visual positioning methods of the state of the art, which we have enclosed by taken the decision to divert our research towards the depth images because they are more suitable for real time applications and to concentrate on architectural scene characteristics because they are stable and unchangeable.
And we will devote the next chapter to the depth acquisition and processing.

3 Depth acquisition and processing

3.1 Introduction

Point clouds are the more appropriate 3D representation for treating real world data notably when the scenes forms geometrics and measures are required.
The device Kinect is among the most low cost depth cameras that produced it.
Initially, it was introduced for gaming, however speedily after its release, it was widely used in different research domains.
In 3.2 we present the different versions of this sensor in addition to a comparison of data produced from each version. afterwards, we review the relevant approaches and concepts used by state of the art visual localization methods.

3.2 Kinect

Kinect Xbox 360 has been a revolution in 3D sensing [91] [92]. It represents a microsoft motion sensor that allows users to interact without an intermediary device (controller or marks) by face and voice recognition. Although it was initially meant for gaming, the technology has been shortly after its release widely used by scientists, robotics enthusiasts and researchers for real world applications development (virtual shopping, mapping and SLAM, education, visual positioning and other areas). Latterly another kinect version has been released. And in 2019, Microsoft released the Azure Kinect promoted as a developer kit with advanced AI sensors for building computer vision and speech models, the three versions are shown in figure 3.1.

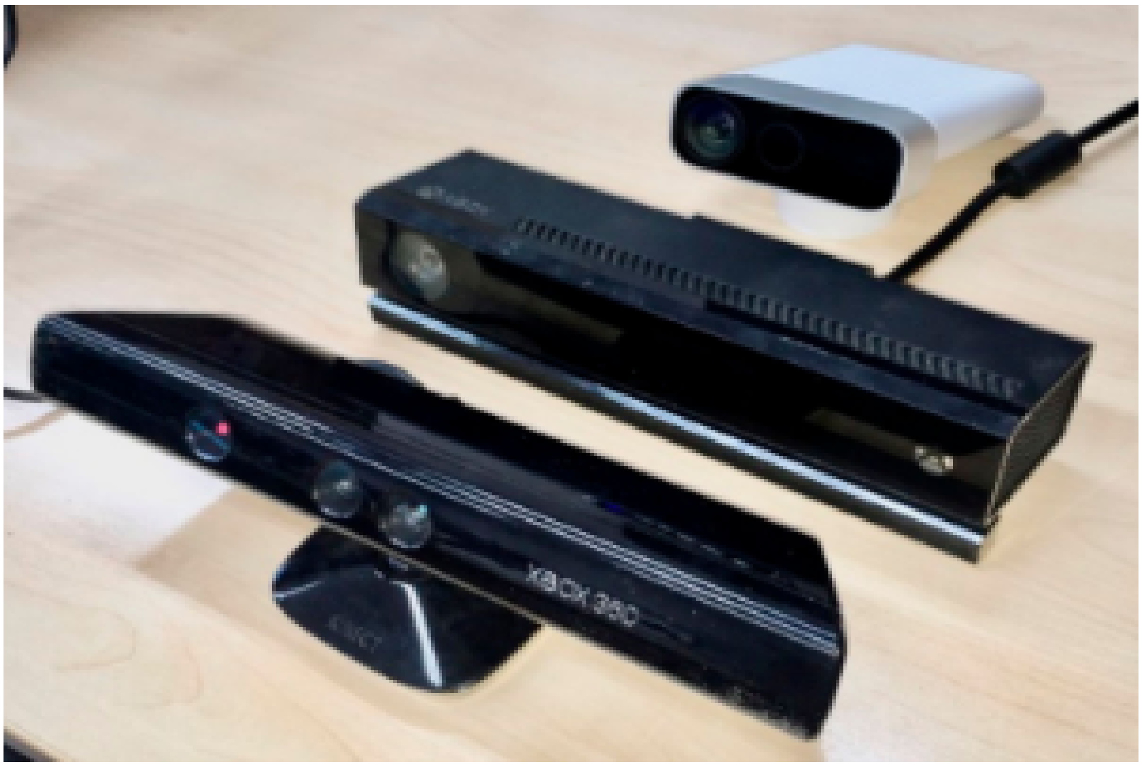

Figure 3.1: front to back: kinect v1, kinect v2, kinect azure.

3.3 The kinect's structure

Figure 3.2 illustrates the structure of kinect that consists on:

- IR projector: That transmits the IR signals to the IR depth sensor.
- RGB camera: which provides RGB images.
- IR camera: provides depth images.
- Microphone array: Amplifies sound from one direction and suppresses those coming from the other directions.

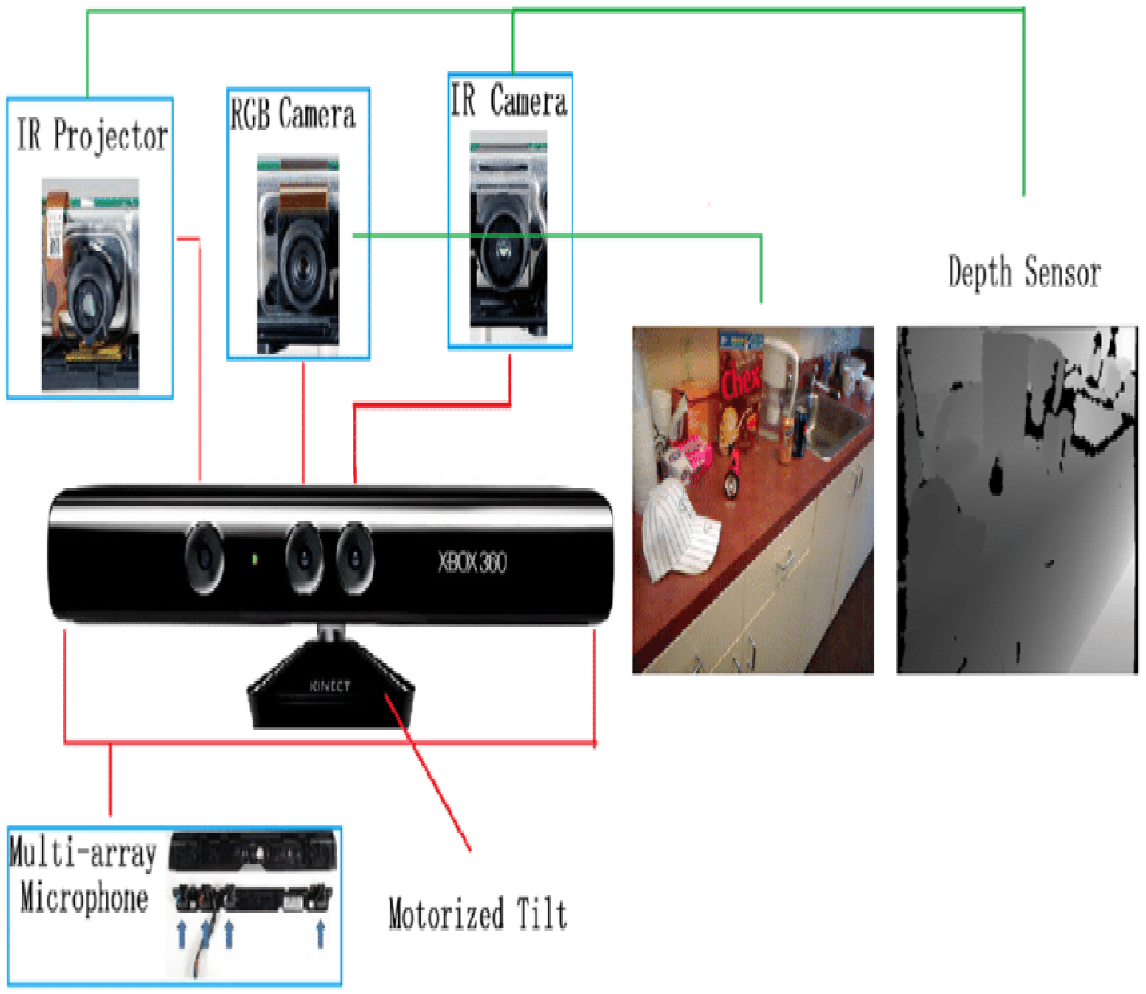

Figure 3.2: Kinect structure.

Figure 3.3 shows the different types of images produced by this later:

- RGB image: Which is a digital image that includes color information for each pixel in the form of red green and blue components;
- Depth image: Is an image with one channel that contains the distance between the viewpoint and the real-world points.
- IR image: is simply the intensity of reflected IR radiation emitted by the Kinect, where the value of a pixel is determined by the amount of infrared light reflected back to the camera.

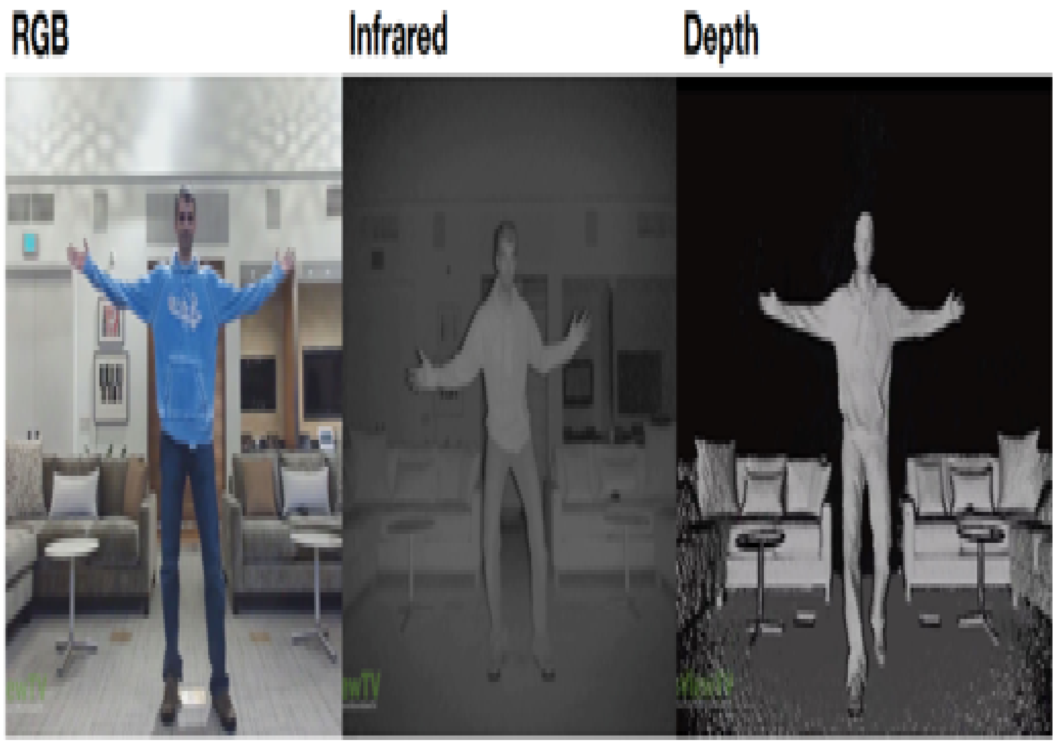

Figure 3.3: Types of images produced by the Kinect.

The following table summarizes the kinect versions characteristics.

	Kinect v1	Kinect v2	Kinect azure
Operating principle	Structured light-pattern projection	Time of flight	Time of flight
Depth image resolution	320×240 px	512×424 px	NFOV unbinned—640×576 px (30 fps) NFOV binned—3$320 \times 288$ px (30 fps) WFOV unbinned—1024×1024 (15 fps) WFOV binned—512×512 (15 fps)
Rgb image resolution	680×480 px	$1920 * 1080$ px	3840×2160 px (30 fps)
Field of View	$58° \times 45°$	$71° \times 60°$	NFOV unbinned—$75° \times 65°$ NFOV binned—$75° \times 65$ WFOV unbinned—$120° \times 120°$ WFOV binned—$120° \times 120°$
Weight	750 g	1390 g	520 g

Table 3.1: Characteristics of each kinect version.

3.4 Intrinsic parameters

They represents the parameters that allow to compute the 3D positions of the captured points, in the coordinate system of the camera, by the following formula:

$$p(D(u,v)) = \begin{pmatrix} P_x = \frac{(u-cd_x)z}{fd_x} \\ \\ p_y = \frac{(u-cd_y)z}{fd_y} \\ \\ p_z = D(u,v) \end{pmatrix} \tag{3.1}$$

Where D is the depth image, $D(u,v)$ is the depth of pixel (u,v), fd_x, fd_y are the focal lengths on x, y respectively, cd_x, cd_y the focal point coordinates.

Algorithm 1 Simplified RANSAC Algorithm

```
1: Initialize a group of 3 points.
2: repeat
3:     Choose randomly a sample of three points.
       Compute the parameters of the plane comprising the three chosen points $L_i$.
4:     repeat
5:         Calculate the distance between the point $p_j$ and the plane $L_i$.
           Add $p_j$ to the subset (inliers) of close points.
6:     until for each point $p_j$
7:     If there are enough points the subset is accepted
8: until the maximum number of iterations is reached
```

3.5 Planar regions extraction from RGBD images

The planar regions extraction methods may be categorized into three categories [93].

3.5.1 RANSAC(Random Sample Consensus) based methods

The classic RANSAC was initially introduced by [94] in 1981 broadly used for shape detection.

It performs plane detection iteratively by randomly choosing three points, computing the plane defined by them, and calculating how many points (in the dataset) lie on this plane within some tolerance threshold. The number of points found is called the score of the plane. The algorithm stops when it reaches stability, based on a low probability of finding a plane with higher score than the previous ones [93].

The RANSAC algorithm (summarized in 6) and its variations [95] [96] [97] are effective to detect large flat areas, however they tend to simplify complex scenes or those that contain many planes, such as stairs, where the steps are considered part of the same inclined plane, thus these methods are often combined with other techniques as (Minimum Description Length) (MDL) [98] and (Normal Coherence Checking) (NCC-RANSAC) [99], to add other constraints to discern planes with more precision. Even with these improvements these methods remain requiring high computational time that's why they are not suitable for real time applications.

3.5.2 Hough transform-based methods

Hough transform-based algorithms were originally used in electronic systems for the detection and binding of lines defined by a set of aligned two-dimensional points in an intensity image [100]. The idea on which this method is based is to transform all the image points in the coordinate system of the parameters (see Figures 3.4) and store the number of occurrences of the parameters in another image, at the end of this step, each point of the new image will contain the number of occurrences of parameters a and b (which are the coordinates in this new image), the points with a high number of occurrences (peaks) represent straight lines in the original image.

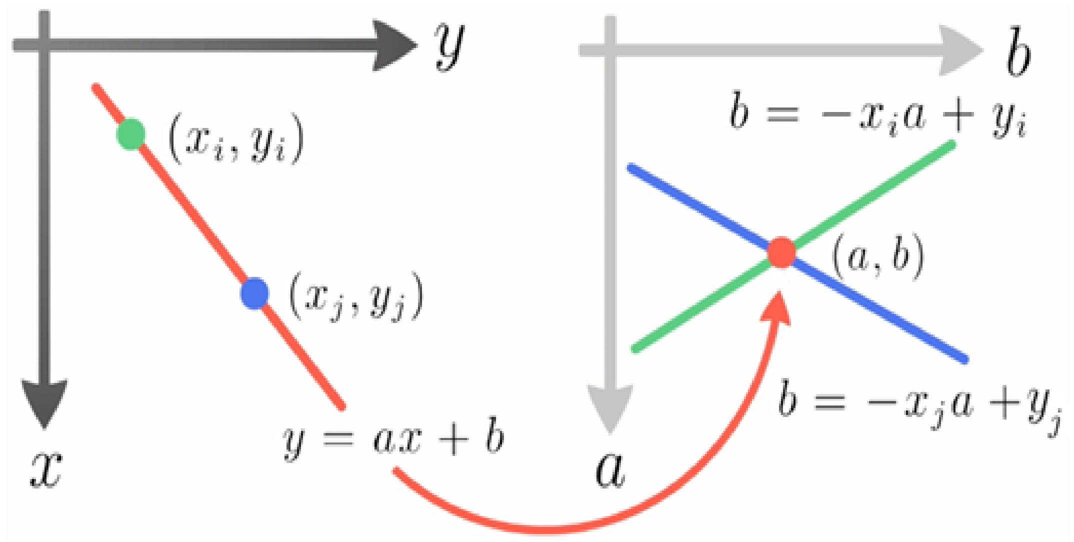

Figure 3.4: Transformation of points in the coordinate system parameter.

Much research has been proposed in order to use the same principle in the detection of planes in a depth image (RGB-D) by extending it to 3D space. [101] have proposed an algorithm which subdivides the image into several zones containing the points which belong to the same plane with a Quaternary tree (quadtree), the detection is carried out with the voting principle of Hough transformation, and the covariance matrix (see figures 3.5, 3.6). However these methods suffer from the computational intensity relating to the transformation of points and the voting system, other methods have been proposed to overcome this weakness such as (Randomized Hough Transform) [102] which calculates the parameters of the planes in a probabilistic way.

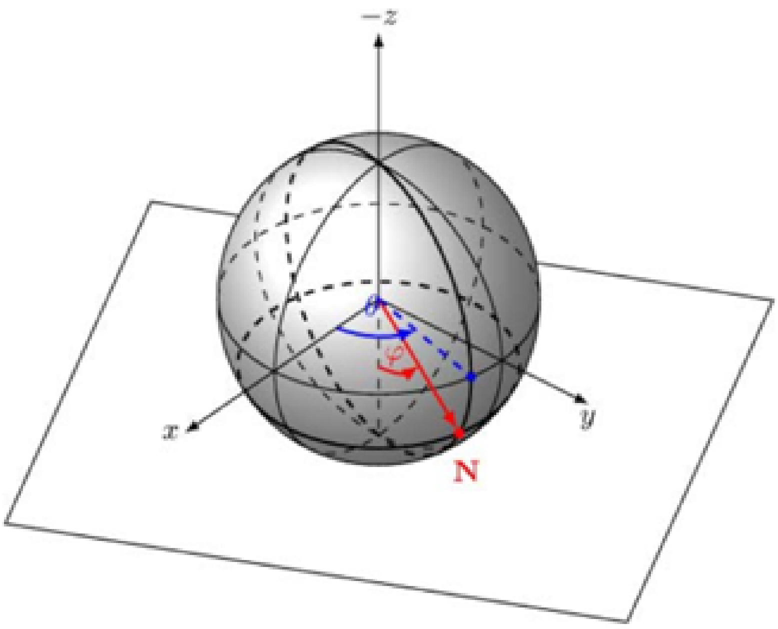

Figure 3.5: The 3-dimensional hough space (θ, ϕ, ρ) [102]. where θ is the angle of the normal vector on the $xy-plane$, ϕ the angle between the $xy-plane$ and the normal vector in z direction and ρ the distance to the origin of the coordinate system.

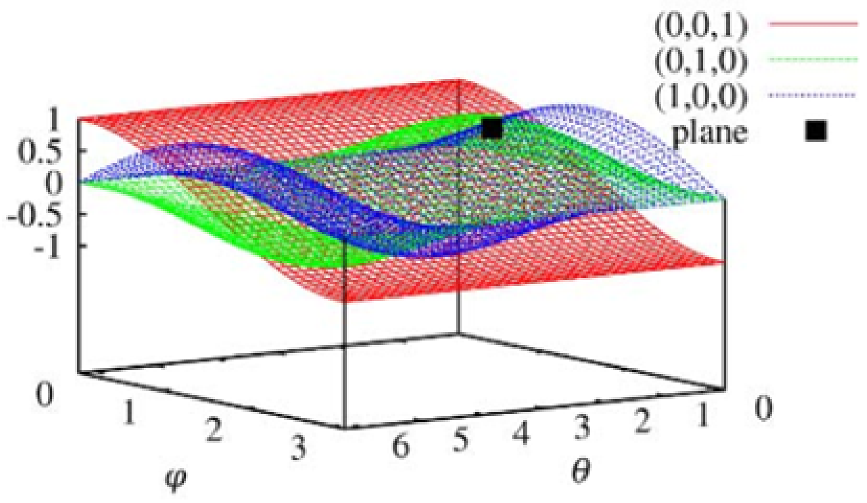

Figure 3.6: Transformation of three points from 3D into Hough Space. The intersection of the curve (marked in black) depicts the plane spanned by the three points [102].

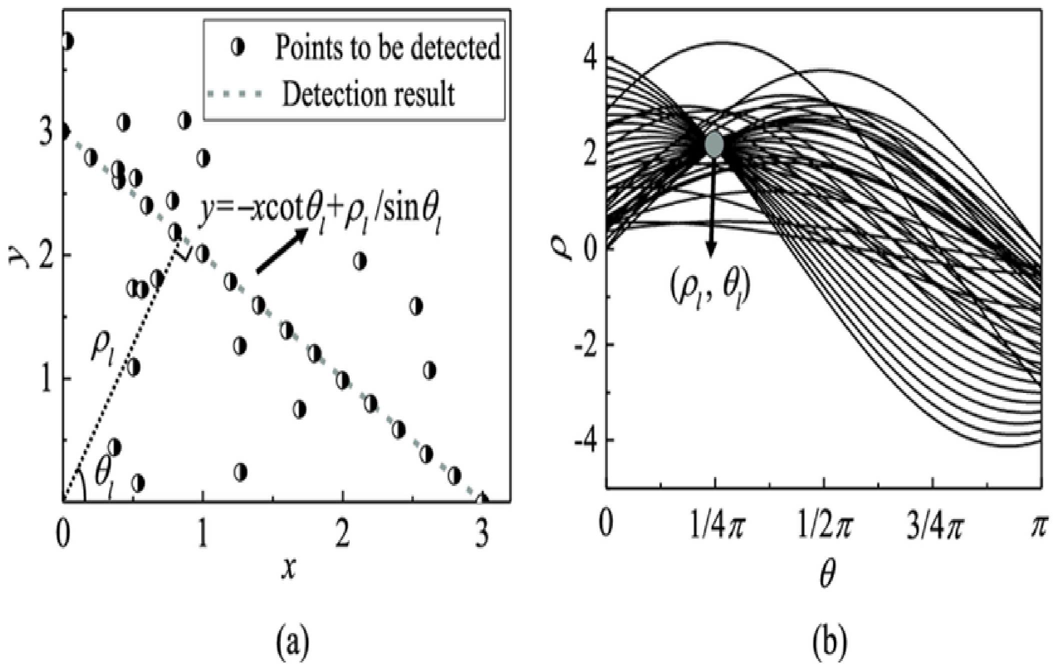

Figure 3.7: Example of line detection with the Hough transformation, (a) points in a Cartesian coordinate system, (b) the same points in a polar coordinate system [103].

3.6 Surface Growing based methods

The principle of these methods is the expansion of the seed region (see figure 3.8) until the fitting error exceeds certain thresholds. Various techniques have been proposed to handle different region types, (e.g: 3D voxels, 2D rectangular regions). [100] proposed an efficient algorithm to detect planes in unorganized point clouds. It first partitions the entire 3D space into a large number of small voxels and then performs clustering by merging the seed with its nearest 26 voxels. [101] proposed a normal constrained region growing method for plane segmentation. Local normal lines are estimated by deducing the local mesh and smoothing via the bilateral filter. These methods are very efficient, nevertheless they have a major limitation which is the choice of initial zones size. if one chooses a large size, one cannot detect the small flat zones, otherwise, the calculation will take a long time. Another method [93] is proposed to overcome the problem of the division size which consists in making it dynamic.

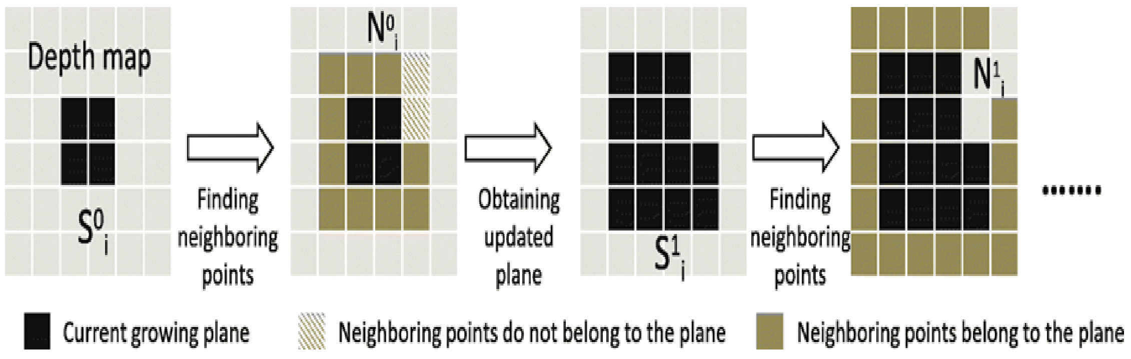
Depth map
S^0_i
Finding neighboring points
N^0_i
Obtaining updated plane
S^1_i
Finding neighboring points
N^1_i
....
Current growing plane
Neighboring points do not belong to the plane
Neighboring points belong to the plane

Figure 3.8: Example of grown based method.

3.7 Registration of two RGB-D images

In the 3D reconstruction of static scenes, there are two families of image registration methods:

- Image to image methods: that estimate the adequate transformation between two successive images, they are used in so called unknown scenes.
- Image to model methods: assume the availability of the scene model, to estimate the transformation that brings an image back to it, in order to locate the camera or to enrich it.

In the context of 3D scene reconstruction, the registration task (also called alignment) consists in finding the translation and rotation parameters allowing to correctly align the superimposed views of the captured scene in order to reconstruct, from these partial surfaces, a complete scene representation.
One of the main tasks in this process is the estimation of the local pose (translation and rotation relative to the data of the previous image, called the target image or destination) of the camera corresponding to each RGB-D image (as schematized by figure 3.9).

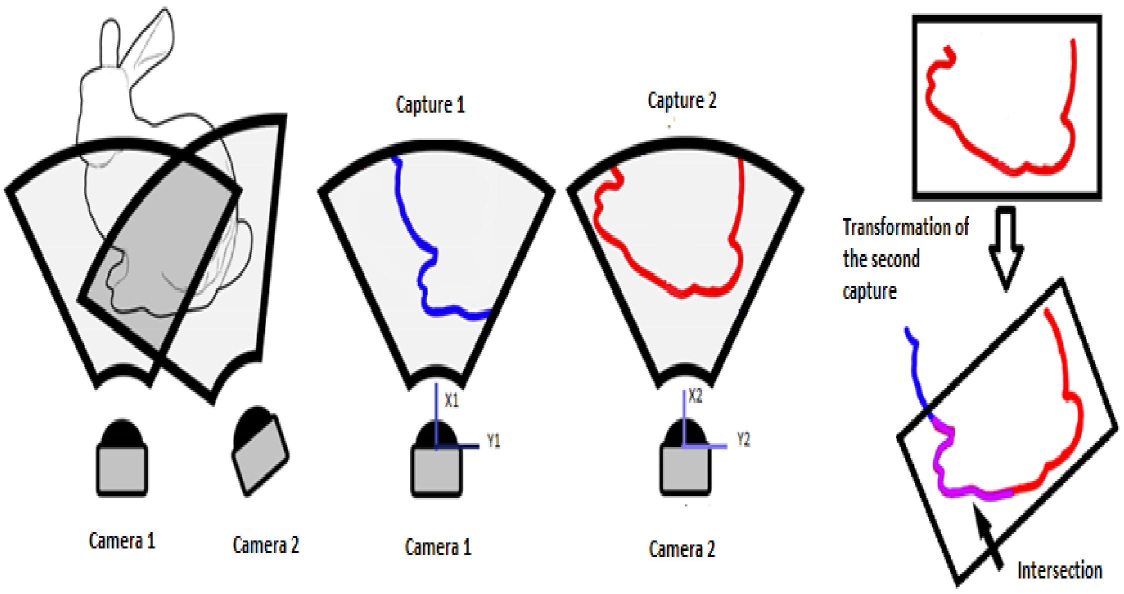

Figure 3.9: Registration example [104].

The techniques used for 3D reconstruction can be classified into two categories:

Sparse correspondence: compute features of each image (descriptors or points of interest like SIFT or SURF) , then using correspondences, the spatial transformation may be calculated.

Dense correspondence: capture and then merge several point clouds with depth sensors (RGB-D), and this is the technique we used.

Most recent approaches use a search for dense matches in a well-defined order in conjunction with other similarity criteria to choose the best point of the target image to associate with a given point of the source image (figure 3.10). This is done by checking its neighborhood in the space of the target image (the spatial neighborhood (also called point-to-point distance) [105] [106].
Where the Euclidean distance (p_1, p_2) between the two points p_1 (x_1, y_1, z_1) and $p_2(x_2, y_2, z_2)$ is equal to:

$$||p_1 - p_2|| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Other methods use the point-to-plane distance (the length of the normal vector dropped from the given point onto the given plane) as illustrated by figure 3.10. As a neighborhood criterion, the plane to be found here is one of the planes bearing the surfaces of the target image [107].

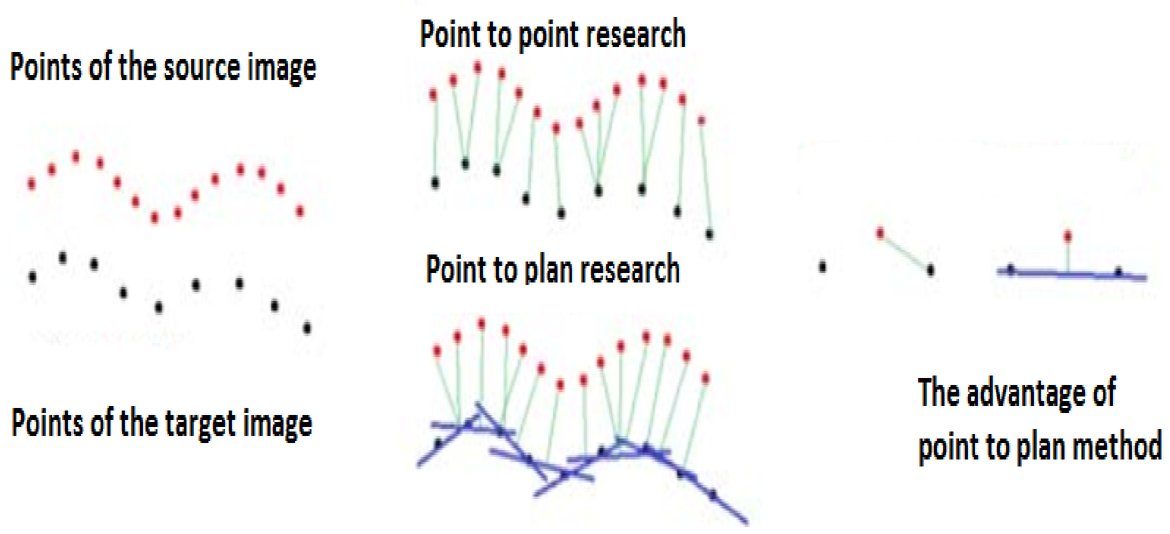

Figure 3.10: Point-to-point vs. point-to-plane methods, we have illustrated here only two dimensions to simplify the concept.

The best combination is one that minimizes the sum of the distances between the points in the source image and their correspondents in the target image, one by one.

3.7.1 The DARCES method based on RANSAC

The method (Data-Aligned Rigidity Constrained Exhaustive Search) [108] is designed for data of two images with partial overlap, it considers a sphere passing through three points (which is the minimum number to determine a transformation) in the source image, then it looks for three points that belong to the same sphere in the destination image, once the points are defined, we can find the transformation and apply it to all the points of the source image. After applying the estimated transformation, we determine the set of overlap (which is the set of points that belong to both images), the number of points of this set is used as a criterion (if it is large enough) for the validity of the estimated transformation. In this method, the RANSAC algorithm intervenes to randomly find the triples of the points to be considered. The tests carried out on this algorithm [108] were too slow, because this one is generally done on more than three points.

Algorithm 2 Simplified ICP

```
1: repeat
2:     Association of points by nearest neighbor criteria.
       Estimation of transformation parameters.
       Transform the points using the estimated parameters.
3: until the maximum number of iterations is reached, or the minimum distance is reached.
```

3.7.2 ICP method

The ICP (iterative closest point) algorithm [109] (2) aims at finding the transformation between a point cloud and some reference surface (or another point cloud), by minimizing the square errors between the corresponding entities. It requires an initial approximation of the transformation, without which its execution can be slow, and it can give erroneous results. For this reason, it is better suited to dense reconstructions where the images to be aligned come from close captures.

We can mathematically express the Distance function (mean squared error) that the ICP algorithm tries to minimize as follows:

$$Dist(T(S), D) = \sum_{s \in T(s)} \sum_{d \in D} g((s-d)^2) \tag{3.2}$$

Where S and D are the source and destination images respectively, s and d are two points belonging to the images S and D respectively, $Dist$ is the total distance between the two images, g is the distance between the two points s and d.

From a practical point of view, the basic ICP algorithm is part of a registration process, which includes the following steps [110]:

- Selection: The sampling of input point clouds where only part of the data is considered, this step can result in the calculation of linear, plane entities, which will be used as correspondences.
- Estimate the correspondences between the points (or entities) in the sub-sampled point clouds: find for each point of the source image, the point closest to it, this search is done using kd trees.
- Rejection: Filter matches, to reduce the number of outliers that are too far from their correspondents, where the RANSAC algorithm is often used.
- Alignment: assigning an error measure and reducing its value to find the optimal transformation (ICP).

3.8 $k - d$ trees

The $k - d$ trees are binary trees, where k denotes the dimension of the nodes. Each non-terminal node divides the space into two half-spaces. The points located in each of the two half-spaces are stored in the left and right branches of the current node. For example, if a given node divides the space according to a plane normal to the direction (Ox), all the points of coordinate x lower than the coordinate of the point associated with the node will be stored in the left branch of the node. Similarly, points with a coordinate x greater than that of the considered point will be stored in the right branch of node [111]. The algorithmic complexity of the nearest neighbor search in the tree is $O(nlogn)$.

3.9 $red - black$ trees

Is a sub type of $k - d$ trees. It is a binary search tree in which each node has information of its color, which is red or black. Figure 3.11 shows an example of red-black tree.

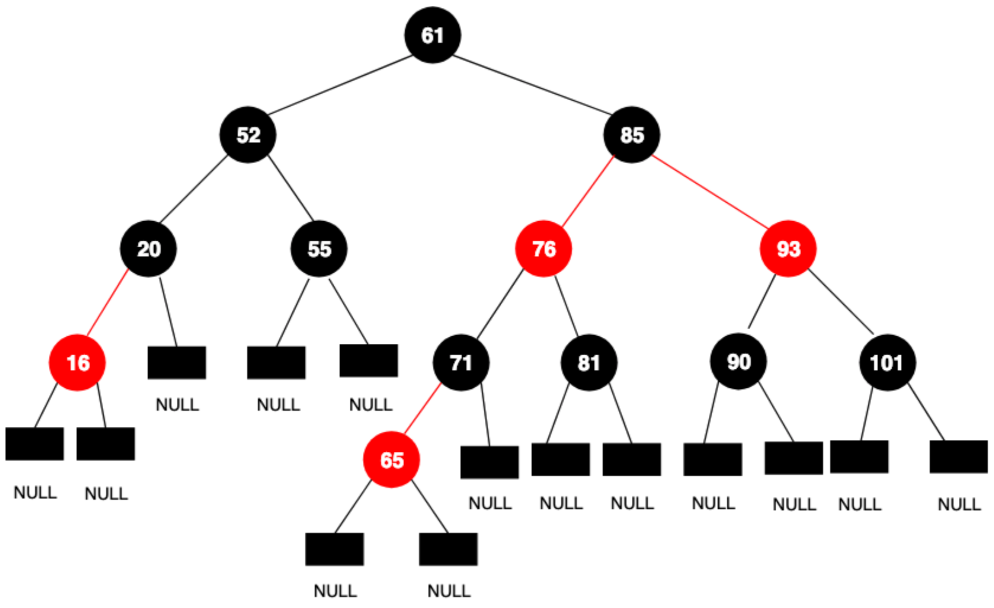

Figure 3.11: Example of red and black trees [112].

The tree's structure is very particular where the following rules are used:

- A node can only be red or black;
- The root is black;
- The children of a red node are black;
- All nodes have 2 children (that are themselves nodes or NIL leaves which have

no value and no children). Their color is always black and it is taken into account when calculating the black height.

By limiting how nodes can be colored on any path from root to leaf, red-black trees ensure that no path is more than twice as long as the others, so that the shaft is approximately balanced. The complexity of the insertion, deletion and search operations is of the order of $O(logn)$. Each node of the tree contains the color, identifier, left child, right child fields. If a child of a node does not exist, the corresponding pointer field of the node contains the value NIL [113].

3.10 Conclusion

In this chapter, we have presented in more details the kinect depth sensor, the data that it produces and its different versions.

After that, we have reviewed the most commonly used algorithms of registration and planar regions extraction in addition to the presentation of some visual localization basic concepts.

In the next chapter, we will introduce our proposed architecture based scene descriptor.

4 The visual positioning system

4.1 Introduction

The visual positioning system represents a technology that enables the devices (such as a smartphone or a drone) and visually impaired people to determine their location in a physical world using visual cues.
One of the main advantages of this system is its ability to work indoors environments where GPS signals may be weak or unavailable. This makes it particularly useful for applications (such as indoor navigation, augmented reality, and autonomous vehicles.
Through this chapter, we present our proposed visual positioning system, based on non-rearrangeable parts of the scene representing the stable features over time. This renders our method able to recognize places even with the standing local changes which may occur in the indoor environment.
Since the kernel of any robust visual localization system begins from an accurate stable scene descriptor, we will focus in section 4.2 on detailing our scene descriptor that we have inspired from the architectural floor plane. Hereafter, the detail of the place recognition process founded on descriptors matching.

4.2 3D Scene reconstruction and 2D Map Computation

Scene depth video is acquired using a depth camera tied up in a cart moved to cover fully the scene. In the case of scenes having a simple structures and small dimensions the camera is merely surrounded (see figure 4.1); whereas in the case of scenes of intricate structures and big dimensions, it is rotated and translated in order to cover all scene details (see figure 4.2).

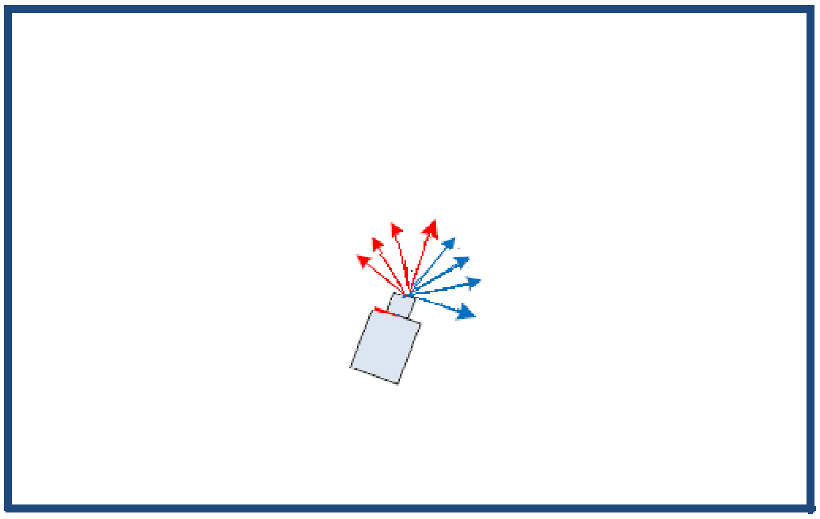

Figure 4.1: Data acquisition in scenes of simple structure and small dimensions.

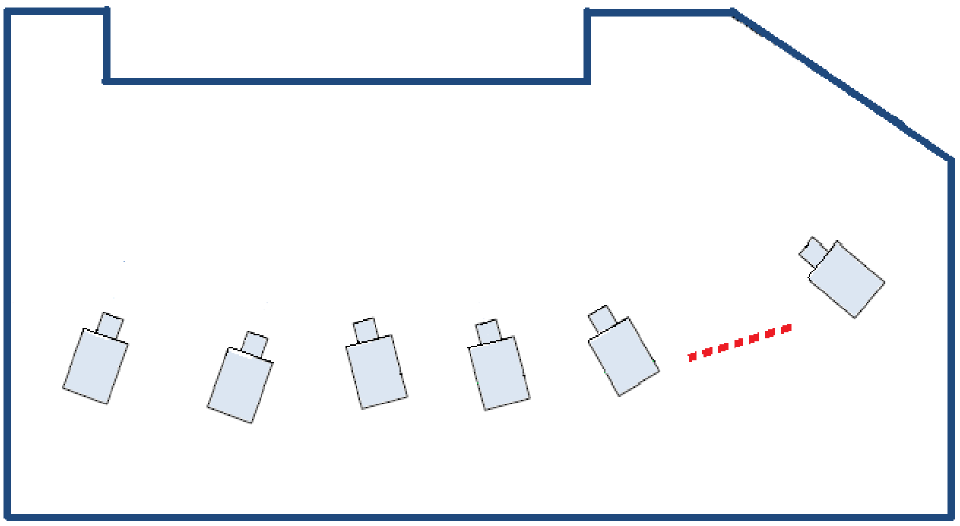

Figure 4.2: Data acquisition in scenes of intricate structure and big dimensions.

Once the scene video is recorded, a set of key depth frames are selected then the planes are located and aligned to gradually compute the 3D model, from which a 2D map is derived and used to deduce the scene descriptor as shown in the diagram of figure 4.3.

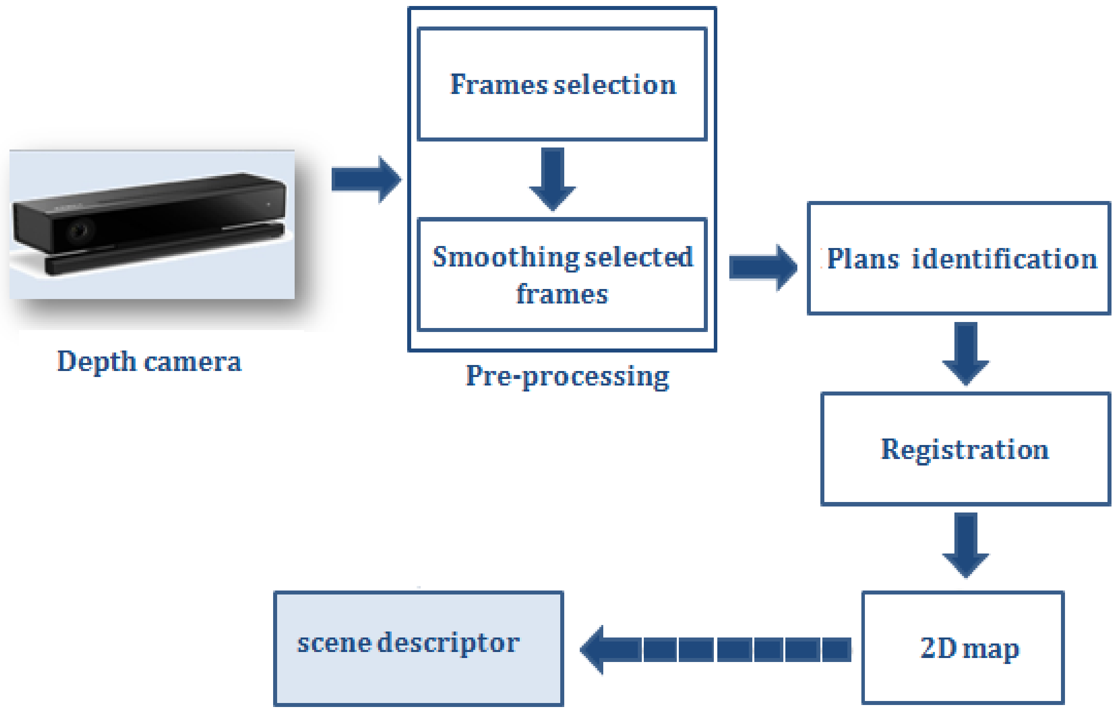

Figure 4.3: Scene description process.

4.2.1 Preprocessing

The preprocessing includes:

- Frames selection consists of selecting a key-frame after each constant number of frames [114] [115] with a key frame interval experimentally fixed to 100 frames whose duration is equivalent to 4 seconds). This implies a decrease in the processing time and data storage space, reduction of alignment errors (what will be explained in section 4.2.4 through figure 4.14).
- Smoothing key depth frames by applying the median filter to reduce the noise.

4.2.2 Planes identification

Planar regions extraction

Planar regions are extracted from each selected key depth frame by dividing recursively the depth image using the quad tree algorithm into rectangular areas (defined by

their average position computed from the ensemble of points that compose them, the normal vectors). The intermediate region R that satisfies to both conditions of flatness $F_l(R)$ and smoothness $S_m(R)$ are considered as planar whereas the others that are not small enough (whose width is greater than a fixed threshold) are recursively subdivided into four regions.
Figure 4.5 summarizes the planar regions extraction process.

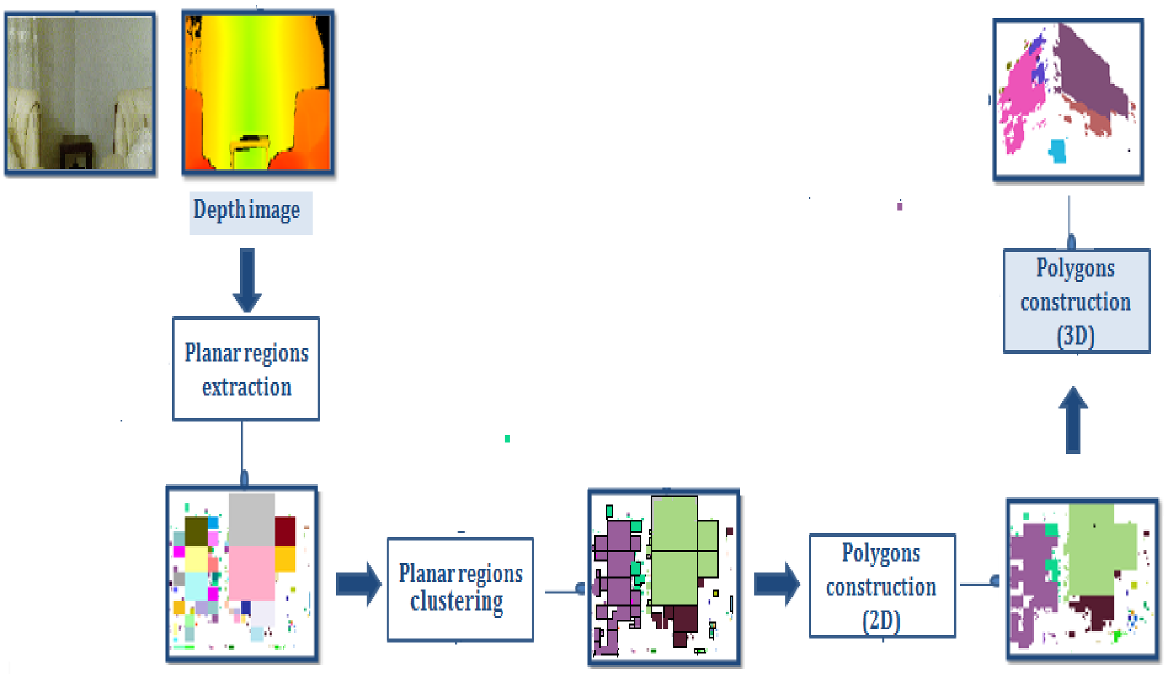
Depth image
Planar regions extraction
Planar regions clustering
Polygons construction (2D)
Polygons construction (3D)

Figure 4.4: Planes identification from depth key frame (the RGB image is not used in the framework)

Smoothness test

To ascertain the smoothness of a region [93], the depth change indication (DCI) map [116] is firstly computed in order to detect the wide changes of depth noted I_V, defined formally by the equation 4.1:

$$DCI(u,v) = \begin{cases} 1 & \max\limits_{(m,n)\in F} |I_V(u,v) - I_V(m,n)| \leq f_s, \\ 0 & otherwise \end{cases} \tag{4.1}$$

Where F is the $4-$neighbors of (u,v), f_s is the smoothness threshold function, (m,n) is the coordinates of a pixel in the depth image.

Once the DCI is calculated, the region smoothness $S_m(R)$ may be verified by:

$$S_m(R) = \begin{cases} 1 & |R| = \sum\limits_{(u,v)\in R} DCI(u,v), \\ 0 & otherwise \end{cases} \tag{4.2}$$

$|R|$ is the size of the intermediate region R. As interpreted by the equation 4.2 a region is considered as smooth if its pixels do not have a high difference of depth with their neighbors.

Flatness test

The flatness of an intermediate region R is provided by the function $F_l(R)$ defined as follows:

$$F_l(R) = \begin{cases} 1 & \begin{cases} MSE(R) < T_m \; and \\ Curvature(R) < T_c \end{cases}, \\ 0 & otherwise \end{cases} \tag{4.3}$$

Where: $MSE(R)$ represents the value of Mean Square Error [117], the $Curvature(R)$ is the curvature of a region in the depth image which is computed using the covariance matrix explained in what follows.
So if the MSE and the curvature are lower than T_m and T_c (depicting respectively the MSE and Curvature thresholds), the region R is considered as flat.

Covariance matrix

The origin of the covariance matrix comes from the field of statistics; it is used to estimate the parameters of a model represented by an algebraic equation. It is also used to calculate the standard errors of estimators [118]. A variance/covariance matrix is a square matrix that includes the variances and covariance associated with several variables. The diagonal elements of the matrix contain the variances of the variables, while the off-diagonal elements contain the covariance between all possible pairs of variables. Indeed if we suppose a set (E) of points $p_i = (x_i, y_i, z_i)^T$, where $i \in 1 \dots n$ then the covariance matrix for this set will be defined by the matrix:

$$C = \sum_{i=1}^{n} (p_i - u)(p_i - u)^T \tag{4.4}$$

$$u = \frac{1}{n}\sum_{i=1}^{n}(p_i) \tag{4.5}$$

Where u is the mean of all points (E). For a given region (R), the covariance matrix can be written as:

$$C(R) = \begin{pmatrix} C_R(x,x) & C_R(x,y) & C_R(x,z) \\ C_R(y,x) & C_R(y,y) & C_R(y,z) \\ C_R(z,x) & C_R(z,y) & C_R(z,z) \end{pmatrix}$$

$$C_R(\phi,\omega) = \frac{1}{n}(\sum_{i=1}^{n}(p_{\phi i} * p_{\omega i}) - \sum_{i=1}^{n} p_{\phi i} * \sum_{i=1}^{n} p_{\omega i}) \tag{4.6}$$

Where $C_R(\phi,\omega), \phi,\omega \in \{x,y,z\}$ is the variance/covariance of the x, y and z coordinate values of the points (belonging to the region R) $p_i = (x_i, y_i, z_i)^T, i \in \{1 \dots n\}$.
The mean square error (MSE) (Mean Square Error), is the average of the squared distances of the points with respect to the estimated plane (see formula 3.2). According to [118],[93] and [98]:

$$MSE(R) = \frac{\lambda_0}{|R|} \tag{4.7}$$

$$Curvature(R) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{4.8}$$

Where λ_0, λ_1 and λ_2 are the eigenvalues of the covariance matrix (C), in increasing order. Once obtained, the $MSE(R)$ and $Curvature(R)$ values for a given region, we can deduce if it is flat or not by applying the formula 4.3.

4.2.3 Polygons construction

In section 4.2.2, we have estimated the boundaries of each planar region in the 2D space (its average position as well as the normal vector of the plane to which it belongs).

Using these properties, The resulting planar regions are merged into subsets [93], each one containing the regions that belong to the same plane (Figure 4.5). For the following reasons:

- It reduces the data storage space because the space needed to store polygons (depicted by a set of vertices) is significantly lower than the space required for planar regions (defined by all points contained in their surfaces).

- It contributes to decreasing the processing time.
- The existence of effectual libraries for polygons manipulation (intersection, union, surface, tiling ...) so the manipulation of polygons is a more common practice in artificial vision than the manipulation of planar regions.

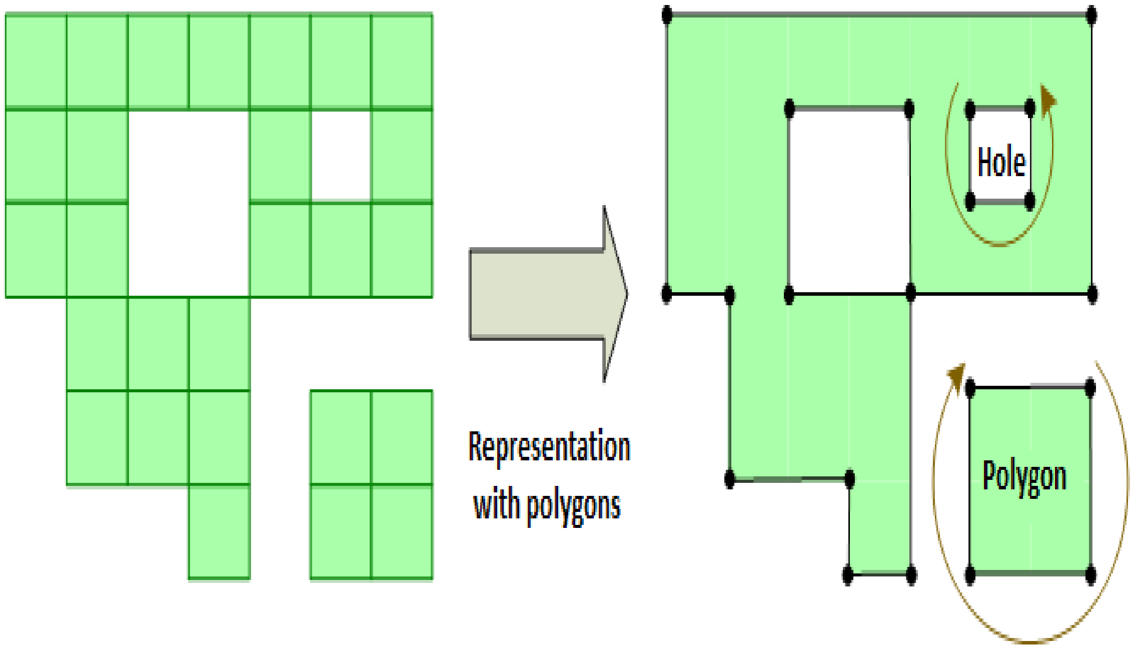

Figure 4.5: Polygons construction.

Two regions $R1$ and $R2$ are considered as belonging to the same plane if the following conditions are fulfilled:

$$\begin{cases} ||\vec{n1} - \vec{n2}|| < L_{nom} \\ |d_1 - d_2| < L_{dist} \end{cases} \tag{4.9}$$

Where $\{\vec{n_1}(a, b, c), d_1\}$ and $\{\vec{n_2}(e, f, g), d_2\}$ are parameters of planar regions $R1$ and $R2$ characterized by normal vectors n_1 and n_2, distances from regions to the coordinate system center $d1$, $d2$.

4.2.4 3D and 2D map computation

3D map Computation

We apply a registration of key frames in order to compute the 3D map. As the global coordinate system (of the scene) is attached to the first keyframe, we estimate the geometric local transformations $\Delta T_i, \in 1 \ldots n$ between each two successive images (I_i, I_{i+1}), the coordinate system of the first image (defined by the first pose of the camera) is considered the global (scene) coordinate system. The global transformation T_i of an image I_i, is the cumulation (figure 4.6) of all the local transformations ΔT_i temporally ordered, it is defined by:

$$\Delta T_n = \prod_{i=1}^{n} \Delta T_i \tag{4.10}$$

The set of polygons of the first depth key frame do not undergo any transformation while the set of polygons extracted from depth key frames I_i, $i \in 2..n$, will be transformed by the transformation ΔT_i in order to be merged with polygons of the previous frames as summarized by figure 4.6.

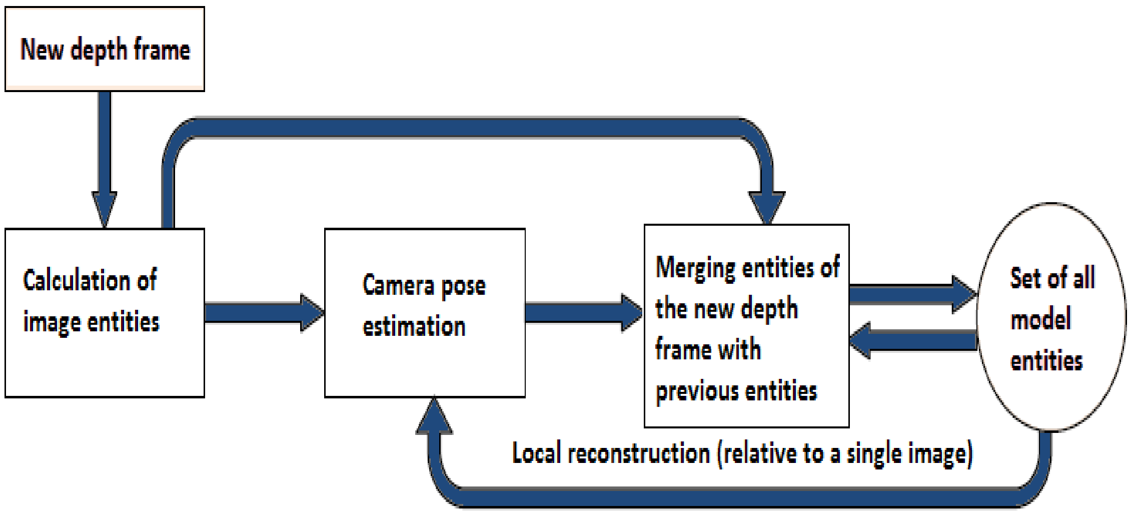
New depth frame
Calculation of image entities
Camera pose estimation
Merging entities of the new depth frame with previous entities
Set of all model entities
Local reconstruction (relative to a single image)

Figure 4.6: Diagram summarizing the registration process.

Figures 4.7 illustrate the results of the registration process. As the camera moves forward, new polygons representing the large plane regions are built and merged with the previous polygons, so that at the end a 3D scene model will be obtained .

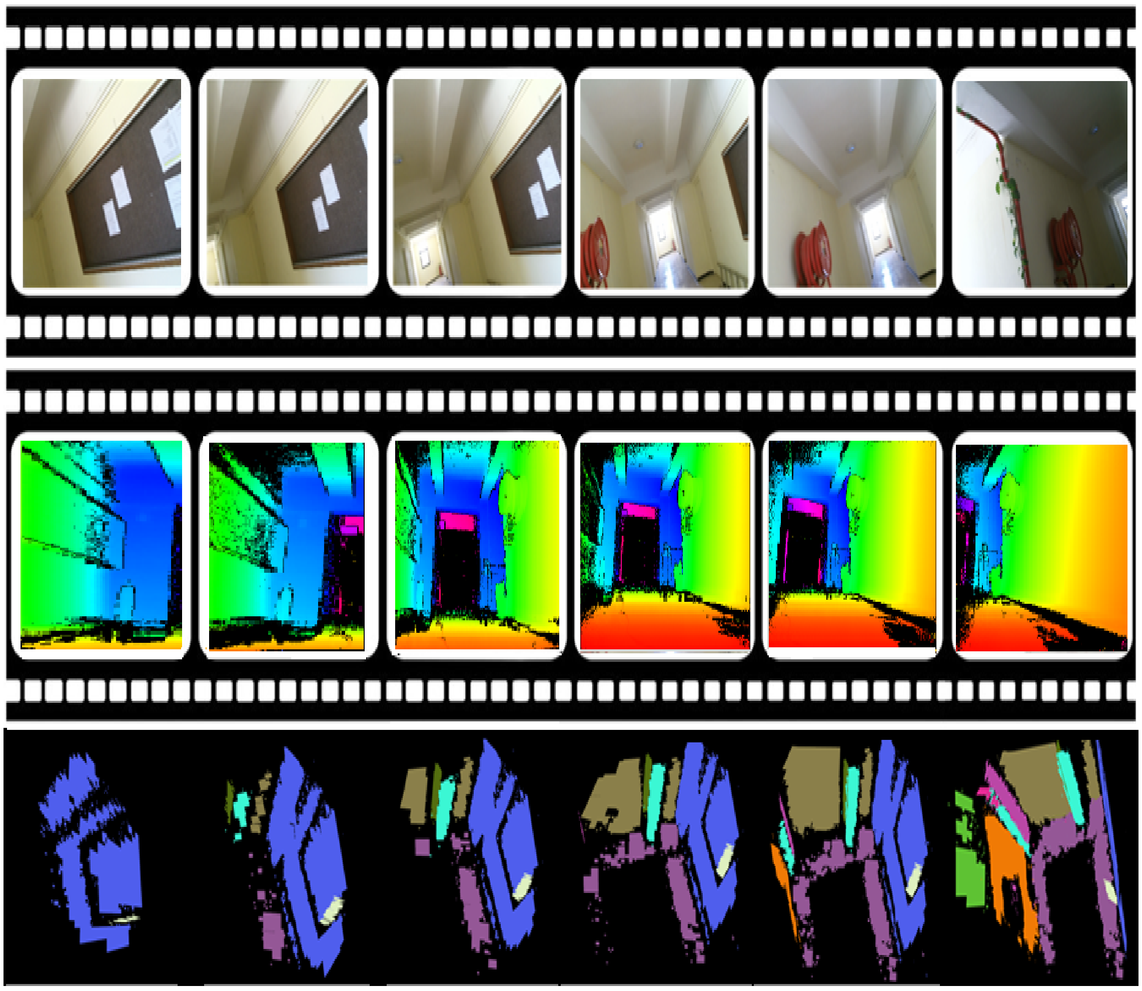

Figure 4.7: From top to bottom: RGB images acquired by rotating a Kinect sensor, The associated depth frames, the partial results of the 3D map construction.

Ground detection

The ground polygon is located as a polygon having a normal vector parallel to the $Yaxis$ or almost parallel for the case of an inclined kinect where we tolerate a certain angle between the normal vector and the $Yaxis$ (less than 30 degrees experimentally determined). Figure 4.8 shows ground polygons detected in different depth frames acquired from an inclined kinect where the $Yaxis$ is colored in green. The case of angle greater than 30 degrees represents the case of a very tilted kinect towards the ground where the ground is captured more than the scene details. Since the visually impaired person does not move in the scene by leaning the head completely to the ground, we did not give importance to this case. We recall that the OX, OY and OZ are the axis of the global coordinate system (of the scene) attached to the coordinate system of the kinect in the first key frame.

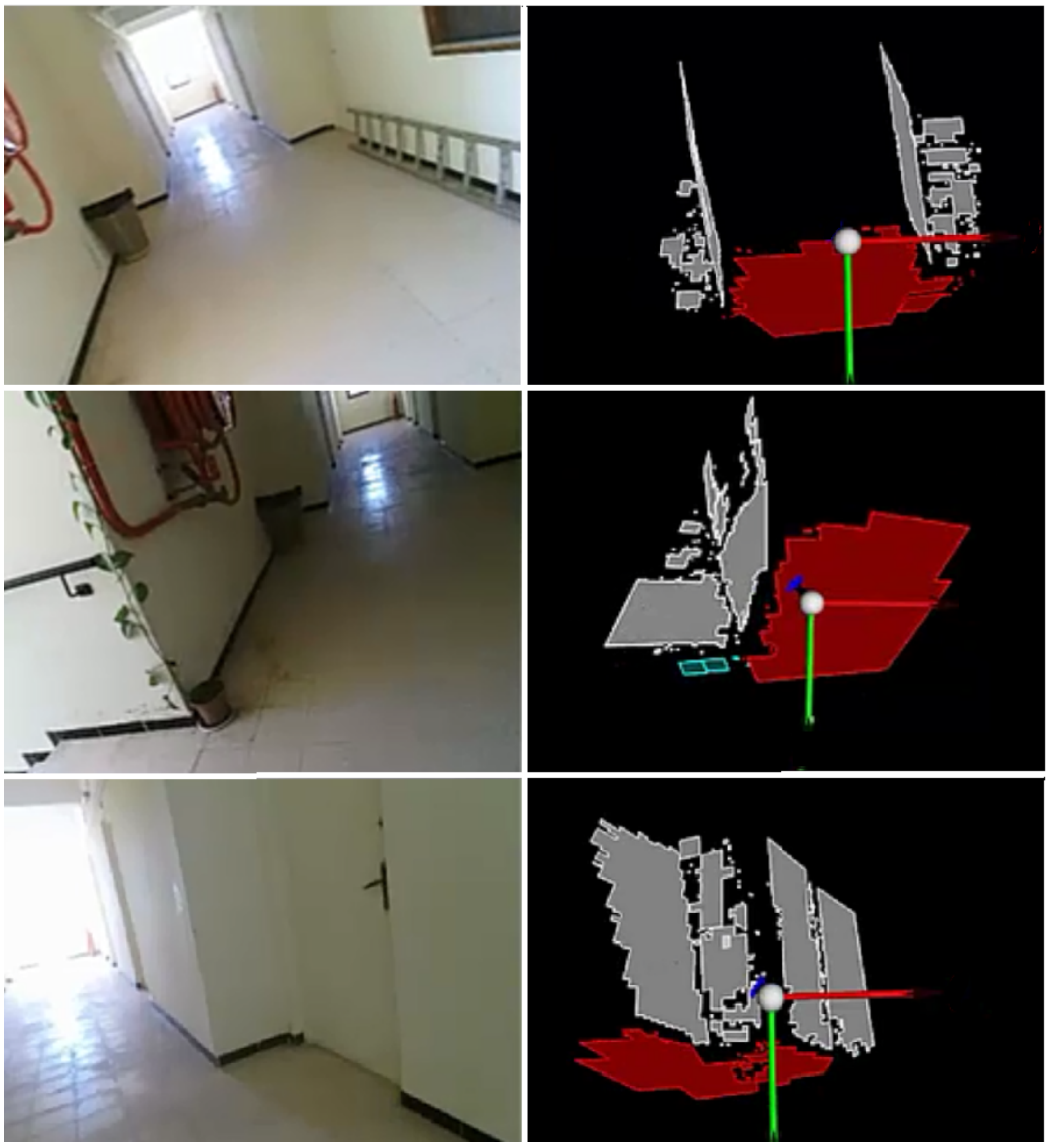
Figure 4.8: From left to right: rgb image added just for a scene visualisation, the detected ground planes colored in red

Figure 4.9 illustrates the ground polygon detected in the 3D indoor scene map (shown in red).

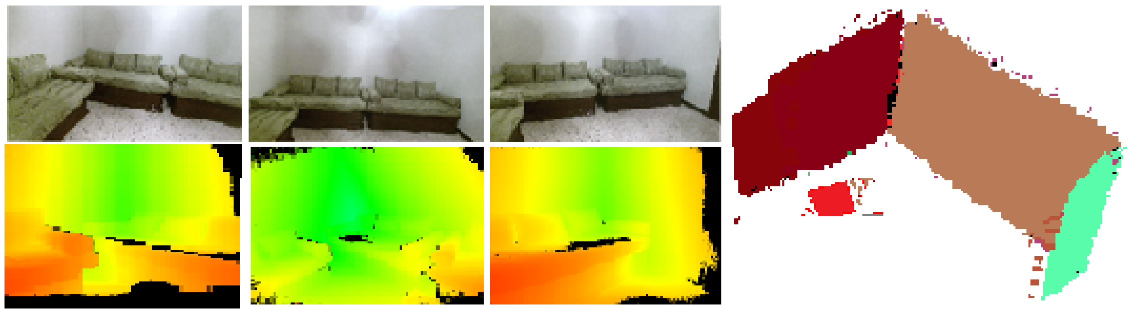

Figure 4.9: From right to left: depth frames, 3D computed scene map and the same map after the ground detection.

2D map Computation

In our first solution version [87], we computed the 2D map by directly projecting the wall planes located in the 3D map on the ground polygon to define the scene boundaries. However, the obtained map did not reflect the complete architectural features, we therefore improved this later by including the missing information related to the doors, windows, and stairs. For that we have used the architectural norms of doors and windows positioning following the international organization for standardization (ISO) (see figure 4.10) which has:

- Fixed the position of windows (between 1.2m and 2.10 m).
- Circumscribed the door height to 2.10 m.

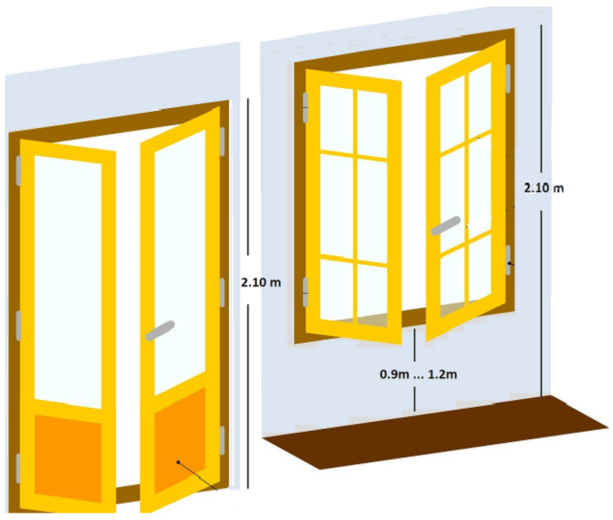

Figure 4.10: Doors and windows positioning following the international organization for standardization (ISO).

As, these advertised heights may be broadened depending on the case of the planned arrangement (example: Hospital, Restaurant, Performance hall,...) but despite all the possible modifications, the conducted study confirms that the margin between 1.20 m and 2.10 m remains a common openness margin. We note here that this margin itself was narrowed in height to ensure that the possible errors in the 3D model (due to noise) will not affect the results as shown by figure 4.11.

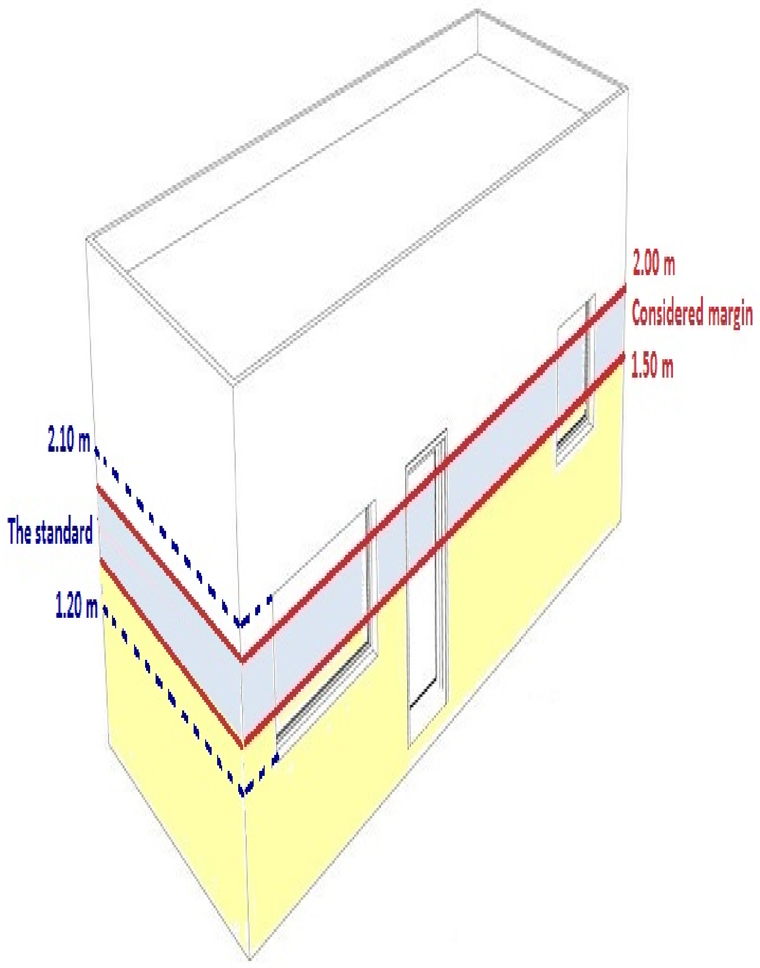

Figure 4.11: Schema illustrating the 3D model cups. According to the spandrel heights, the overtures of windows and doors are between 1.20 m and 2.10 m, however, the merging between 1.50 m and 2.00 m is considered as first cutting limits.

The 2D map is computed as follow:

The areas of the orthogonal polygons to the ground are computed then they are stored in an auto-balanced red-black tree (explained in section 3.11) [119] in descending order of their areas. So that small planes whose areas are small will be filtered out and the highest polygons with big areas are considered as walls and a cup of them

(shown by blue in figure 4.11) is projected on the ground to represent them in the 2D map. By line segments with labeled discontinuities indicating the presence of door or window (see figure 4.12).
The overtures of doors have been distinguished from those of windows on the basis of the standard spandrel heights advocated by ISO. Figure 4.11 illustrates the door and window openings. The door overtures are detected in both yellow and blue cups whereas the window overture is visible only in the first (blue).

Figure 4.12 shows an example of the door and window overtures detected in an indoor scene.

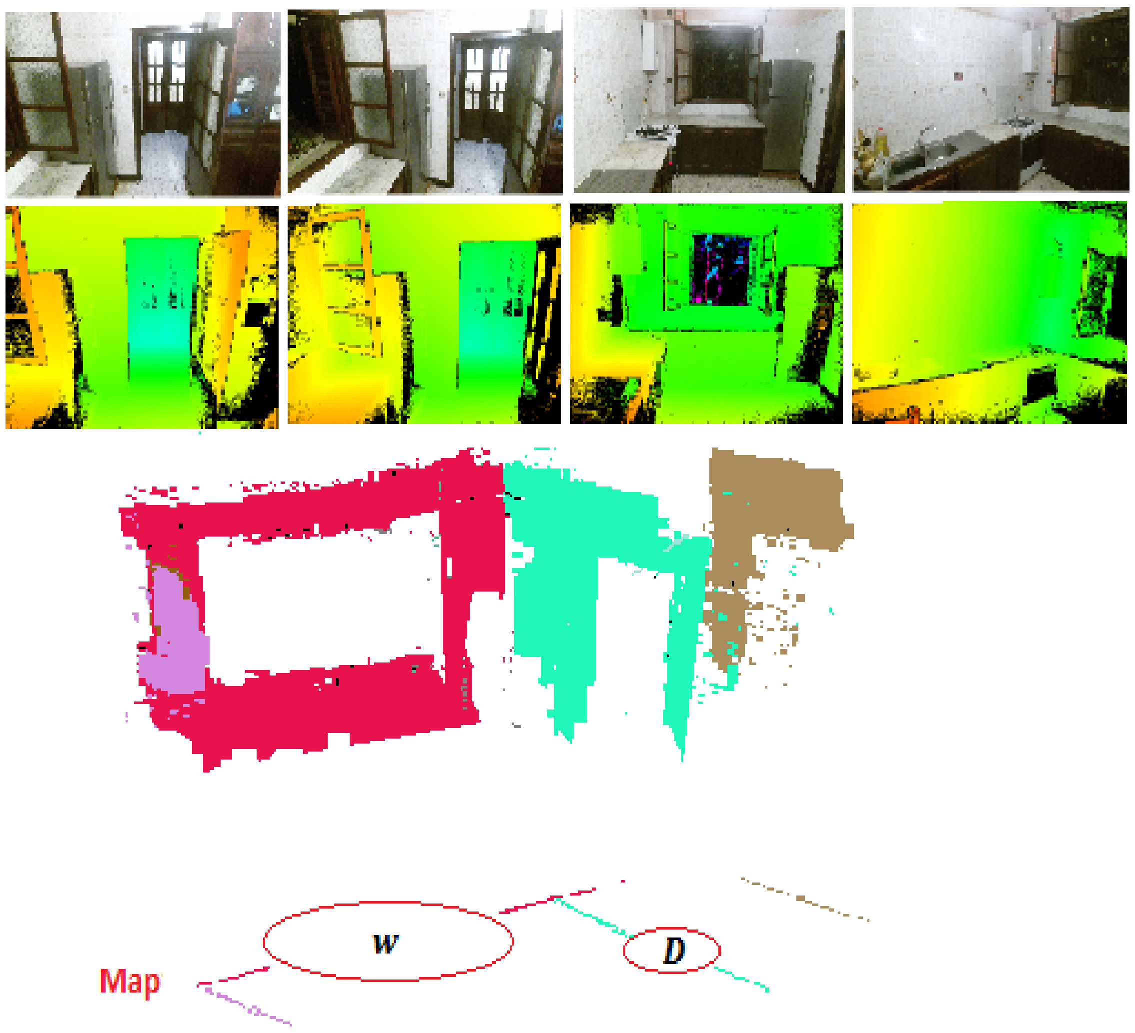

Figure 4.12: Example illustrating the overtures of window and door computed in the indoor scene shown by the RGB images in the first row and depth images in the second row.

In the example of figure 4.13, the gray, pink and mauve polygons were identified as walls whereas the blue one (cloth cabinet) was filtered out despite its large surface because it is not among the highest scene polygons.

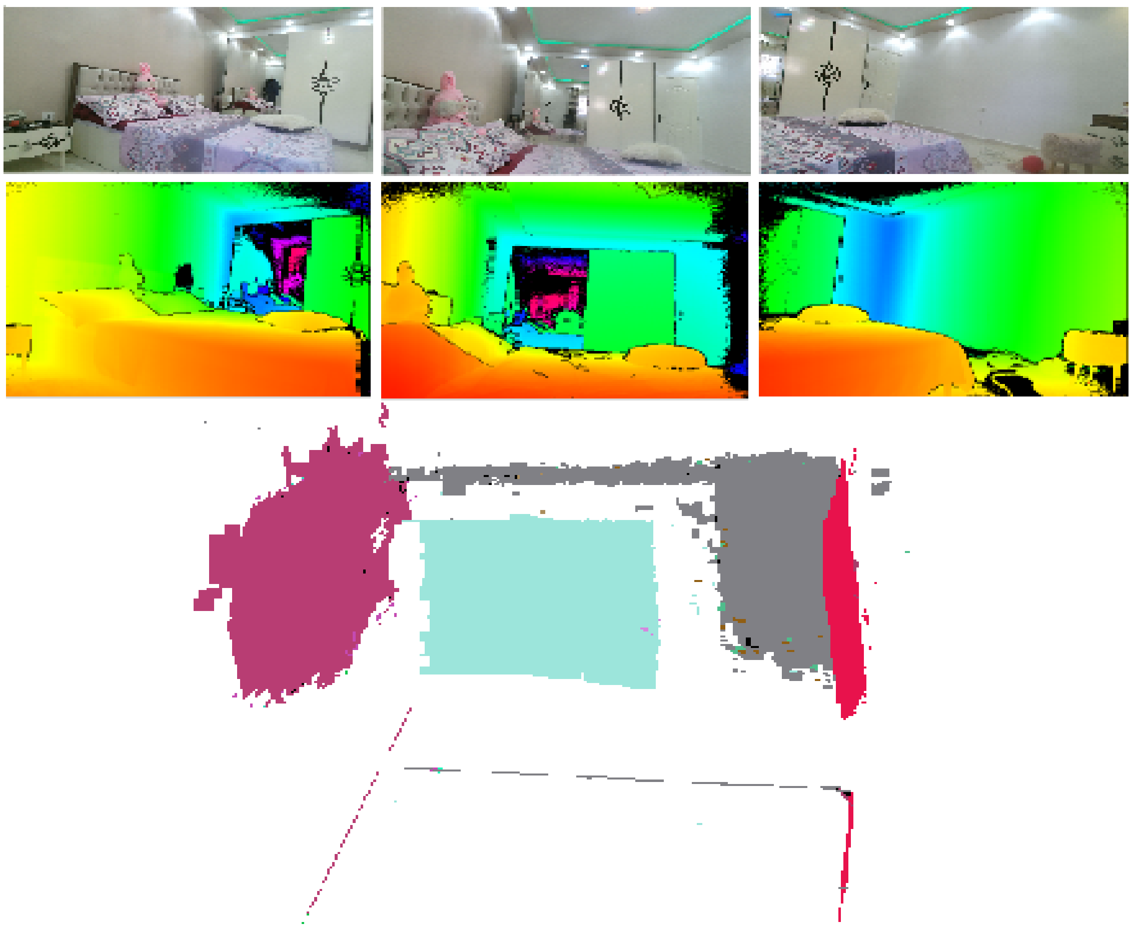

Figure 4.13: 3D and 2D map computed in the indoor scene shown by the top row of RGB images.

Through figure 4.14 we highlight another benefit of the key frames selection step that is the alignment errors reduction.

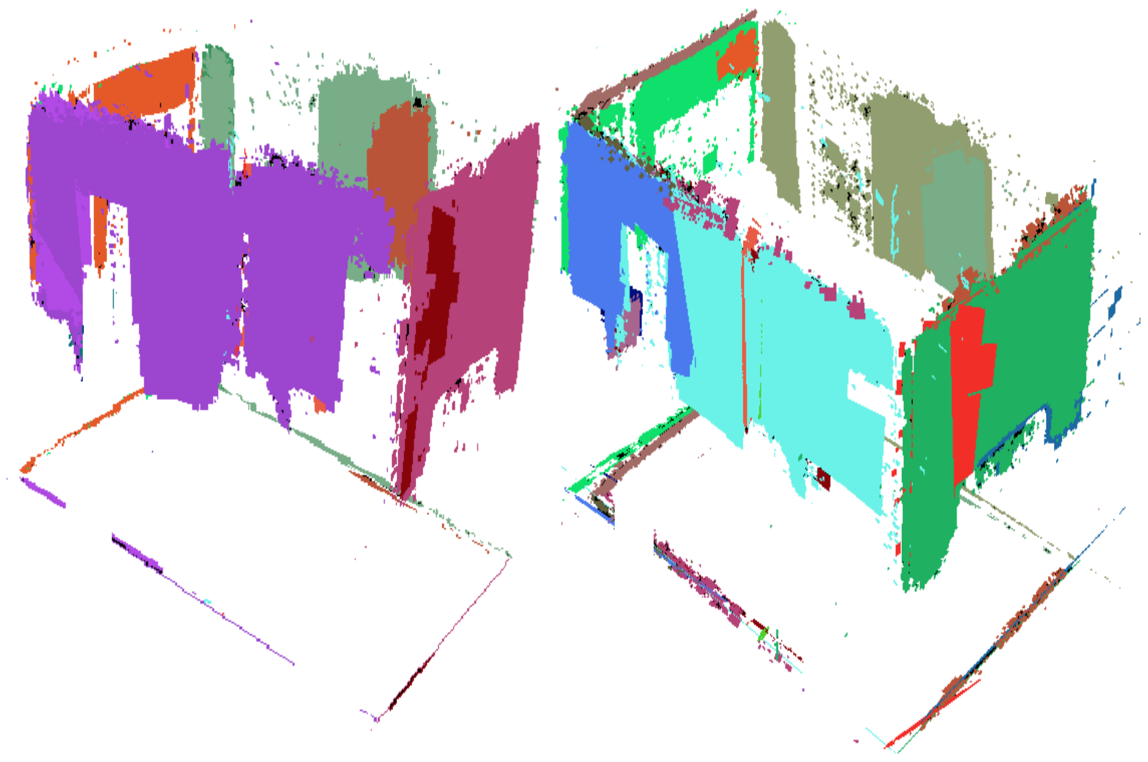

Figure 4.14: From left to right: Maps computed with selected frames, maps computed using all video frames.

Stairs detection and representation in the 2D map

After the 2D map computation, the detected stair railings in the 3D map are represented in the 2D map by rectangular shapes with parallel lines regularly spaced inside and a direction added to illustrate the type of the stair steps (ascending or descending) as illustrated in figure 4.15.

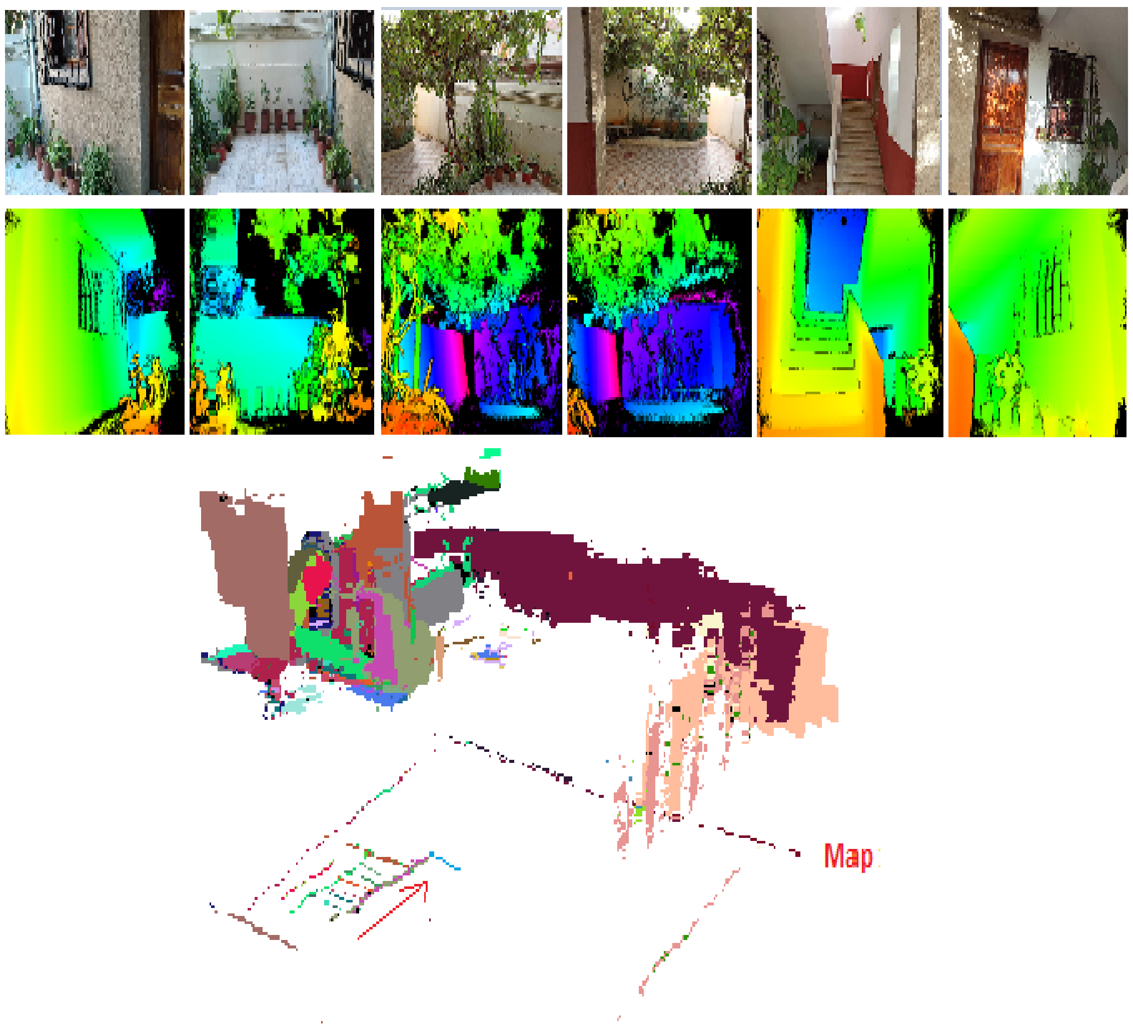

Figure 4.15: From top to bottom: RGB images, depth images, 3D model accompanied with 2D map. The 2D map contains two apparent delimitations (The outside edges represent the scene borders and the rectangular shape illustrates the projection of the captured stair railing whose parallel lines contained in its interior represent the staircase steps).

For the descendant stair steps only treads polygons are detected in the 3D model however for ascending stairs, both treads and risers polygons are spotted as shown in figure 4.16).

Algorithm 3 Stairs detection

```
Input: $3Dmap$ Polygons of the constructed 3D map
Output: $stairs$ the detected stair railing
 1: for P ∈ 3Dmap do
 2:     if $P \parallel groundPolygon$ then
 3:         Add(P, Condidatestairs);
 4:     end if
 5: end for
    //function that outputs the parallel polygons regularly spaced
 6: stairs.polygons = regularySpaced(Condidatestairs, 0.12, 0.20);
 7: for P ∈ 3Dmap do
 8:     if $P \perp groundPolygon$ then
 9:         Add(P, CondidaterisersPolygons);
10:     end if
11: end for
    //function that outputs the parallel polygons regulary spaced
12: risersPolygons = regularySpaced(CondidaterisersPolygons, 0.27, 0.35);
13: if risersPolygons = ∅ then
14:     Stairs.type= $"descendant"$;
15: else
16:     Stairs.type= $"ascendant"$;
17: end if
18: return $Stairs$
```

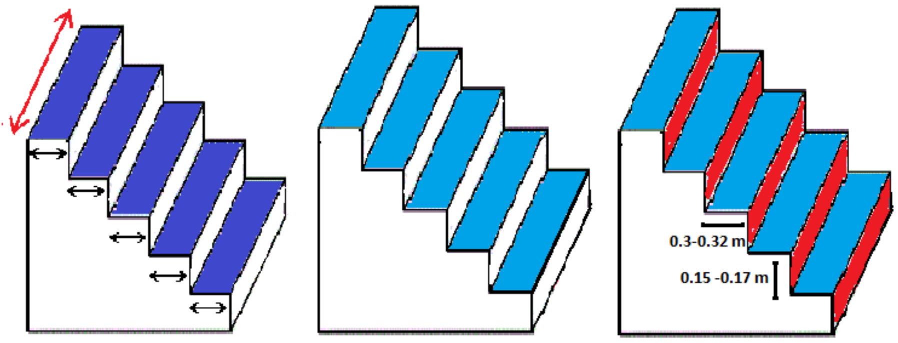

Figure 4.16: From left to right: Stairs schema, treads polygons shown in blue on descendant stairs 3D model, Polygons of treads in blue and risers in red on ascendant stairs 3D model.

Algorithm 3 summarizes the stair railing detection.
The algorithm 3 uses the function $regularySpaced$ that we have defined on the basis of stair sizing architectural norms.

Which have fixed the treads width between $0.30m$ and $0.32m$ and the risers height between $0.15m$ and $0.17m$ as shown though figure 4.16.

Algorithm 4 regularySpaced ($Condidatestairs$, H_{inf}, H_{sup})

```
Output: $RS$ the regularly spaced polygons
 1: $P =$ first_element($Condidatestairs$);
    //from the second element to the end of Condidatestairs
 2: for G $\in$ Condidatestairs do
 3:     d=Distance(P.BaryCenter, G.barycenter);
 4:     if $d \leq H_{sup}$ and $d \geq H_{inf}$ then
 5:         Add(P, RS);
 6:         P=G;
 7:         remove(P,Condidatestairs);
 8:     end if
 9: end for
10: add(P, RS);
11: return $RS$
```

The $regularySpaced$ function 4 is called in algorithm 3 twice. The first time, with a set of polygons parallel to the ground and $0.12m$, $0.20m$ representing respectively the lower and upper bounds of the extended risers height interval to($0.12 - 0.20m$) due to the possible errors in the 3D model in order to identify the set of treads stairs representing polygons regularly spaced with a height in the upper-mentioned interval.

The second, with a set of polygons perpendicular to the ground and 0.27, $0.35m$ depicting the limits of the extended treads width interval ($0.27-0.35m$) to locate the risers polygons spaced with a width belonging to this interval.

4.2.5 Descriptor computation

We define on the 2D map the following elements (see figure 4.17):

- Line Segment L_k defining a wall in the 2D map.
- Corner Q_k is the intersection of the two successive lines segments L_{k-1} and L_k.
- d_k length of the line L_k demarcated between two successive corners Q_k and Q_{k+1}. We specify that the segment length is set to zero when one of its two corners is not detected.
- Angle α_k between two successive line segments L_{k-1} and L_k is the angle between their corresponded direction vectors V and W (see figure 4.18). That is computed as follow: $\alpha_k = \cos^{-1}[\frac{(V.W)}{(|V||W|)}]$, where $V.W$ is the scalar product of the two vectors and $|V||W|$ is

the product of their norms.
- $(t_{k,j}, l_{k,j}, d_{k,j})$: Indicates the presence of the j^{th} opening on the wall k of type $t_{k,j}$ (equal to 0 for windows and to 1 for doors), of length $l_{k,j}$, at a distance $d_{k,j}$ from the corner k.
- $(t_{k,i}, l_{k,i}, w_{k,i}, ds_{k,i})$ indicates the presence of the i^{th} stair nearest to wall k, of type $t_{k,i} = 1$ (if the stairs are ascending otherwise $t_{k,i} = 0$), of dimensions $(l_{k,i}, w_{k,i})$ at a distance $ds_{k,i}$ from the nearest wall k (see figure 4.17).

The extracted 2D map is described geometrically with a set of $n - uplets$. Each one, related to the vertex number i contains the pair of (length of next segment and associated angle), a triplet of values (type of opening (door or window), its length and its distance from the vertex i) for each opening in segment connecting the vertices i and $i+1$. At the end, information about stairs is added. Each stair of the 2D map is coded with its type (ascendant or descendent), its dimensions (length, width), its distance from the nearest wall (see figure 4.17):
If we describe the 2D map of figure 4.17, where the walls number $1, 3$ and 4 contain respectively $1, 1$ and 2 windows and the wall number 2 contains a door, we obtain the following descriptor.

$D = \{Q_0(\alpha_0, d_0), Q_1(\alpha_1, d_1, (0, l_{1,0}, d_{1,0})), Q_2(\alpha_2, d_2, (1, l_{2,0}, d_{2,0}), (1, L, W, ds_{2,0}))$
$Q_3(\alpha_3, d_3, (0, l_{3,0}, d_{3,0})), Q_4(\alpha_4, d_4, (0, l_{4,0}, d_{4,0}),$
$(0, l_{4,1}, d_{4,1}))\}$.

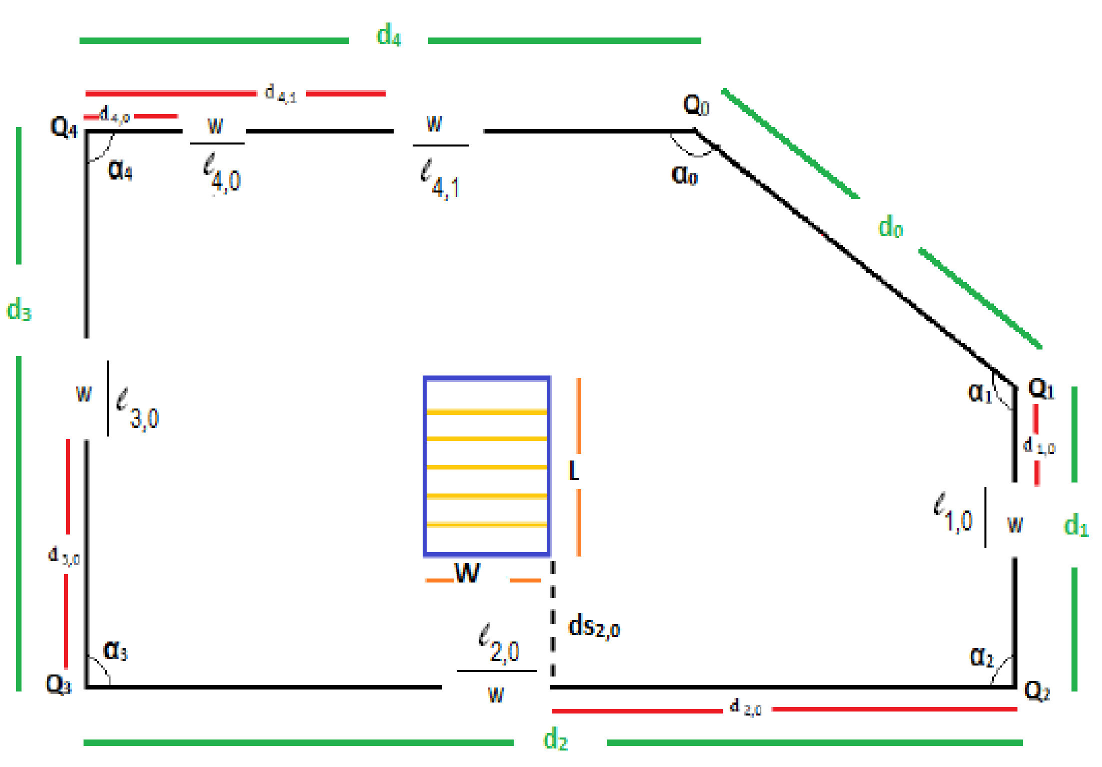

Figure 4.17: Example of closed 2D map computed in an indoor environment.

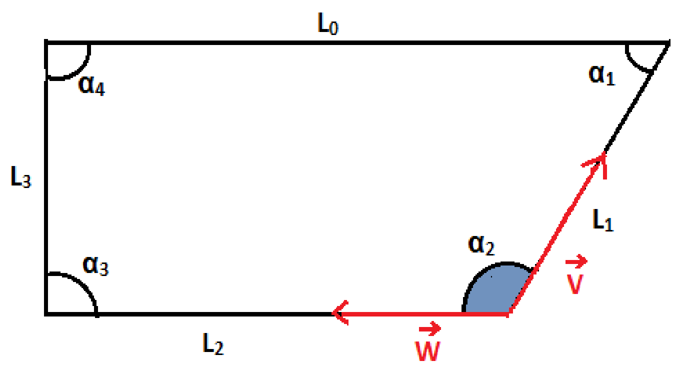

Figure 4.18: Angle α_2 between the successive lines L_1 and L_2 and their corresponded direction vectors.

Considering the 2D map of figure 4.19, where the wall number 4 is not demarcated so its length is set to zero and the descriptor will be as follow.

$$D = \{Q_0(\alpha_0, d_0), Q_1(\alpha_1, d_1, (1, l_{1,0}, d_{1,0})), Q_2(\alpha_2, d_2, (0, l_{2,0}, d_{2,0})), Q_3(\alpha_3, d_3), Q_4(0, \alpha_4)\}.$$

Algorithm 5 The scene descriptor computation

```
Input: Video sequence of depth frames
Output: $D_s$ the computed descriptor of the scene
1: for Each frame do
2:     Extracting planar areas from selected frame;
3:     Clustering each set of planar areas that belong to the same plane into subsets;
4:     Construction of polygons from subsets of planar regions;
5:     Merge the obtained polygons with polygons of the previous frames to build
       gradually the 3D map;
6: end for
7: Determination of 2D map from computed 3D map;
8: $D_s$ computation on the basis of determined 2D map;
9: return $D_s$
```

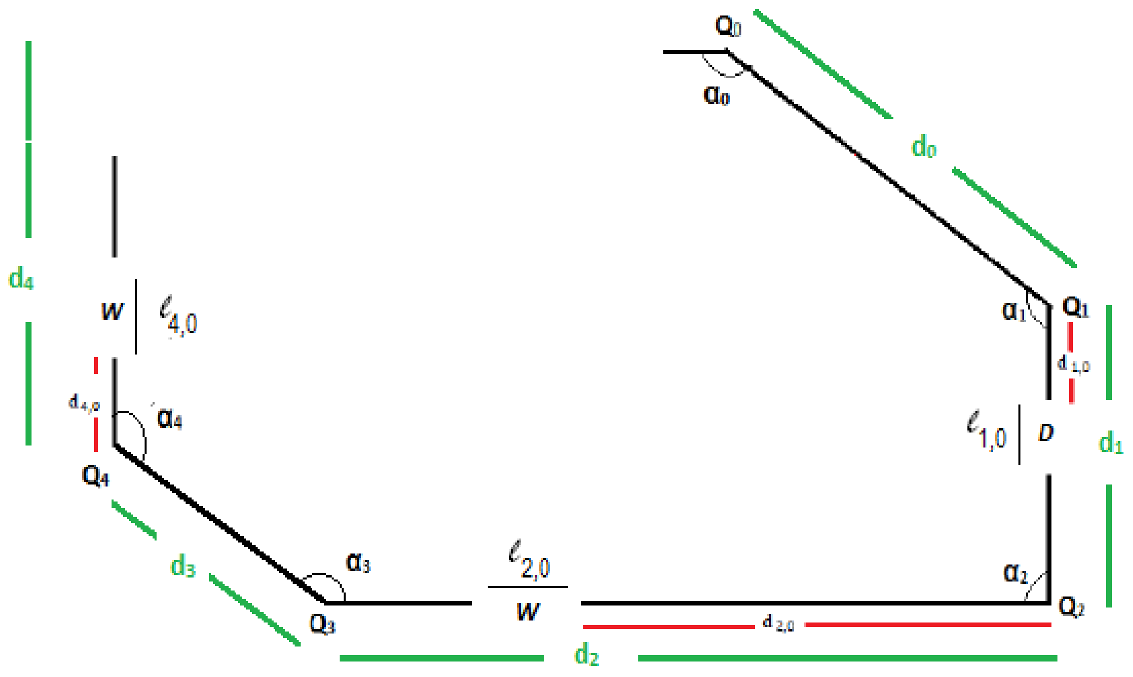

Figure 4.19: Example of an unclosed 2D map computed in an indoor environment.

Algorithm 5 summarizes the scene descriptor computation method.

4.3 The Place Recognition method

The purpose of this section is the identification of the unknown location.

Figure 4.20 summarizes the place recognition process that uses a dataset containing the different scenes descriptors its construction framework is detailled in the next chapter.

The process takes as input a depth video acquired by a depth sensor (Kinect V2); From which it computes the query descriptor using the explained method in section 4.2.5 then searches for the most similar dataset descriptor to this later based on the matching principle presented in 4.3.1.

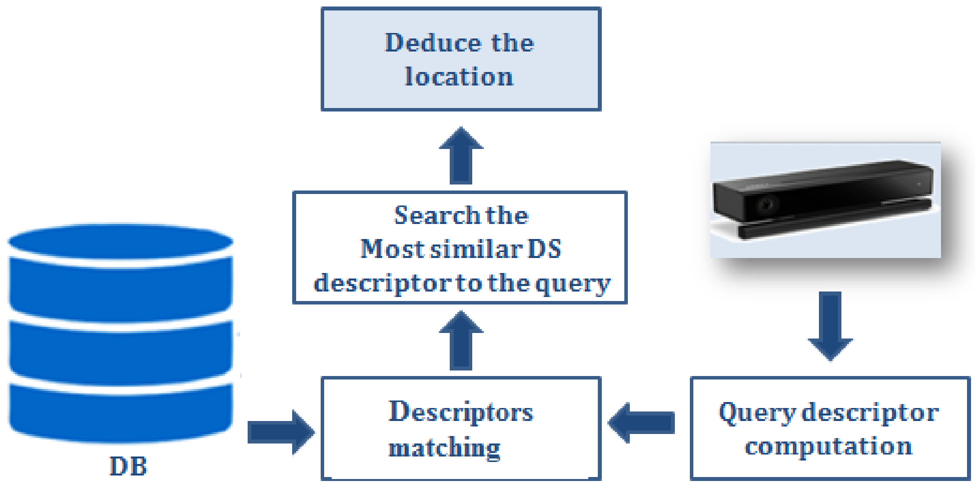
Deduce the location
Search the Most similar DS descriptor to the query
Descriptors matching
Query descriptor computation
DB

Figure 4.20: Diagram of the Place Recognition process that takes as input a scene depth video, calculates the corresponding descriptor, compares it to the data-set descriptors and identifies the location.

4.3.1 Descriptors Matching

Similarity measure

Let $D_q = \{Q_k(\alpha_k, d_k, (t_{k,j}, l_{k,j}, d_{k,j}), (t_{k,i}, L_{k,i}, W_{k,i}, ds_{k,i})\}$ be the descriptor of an unknown scene (see subsection 4.2.5).
Let D_p be the descriptor of a scene from the dataset.

$$D_p = \{Q'_k(\alpha'_k, d'_k, (t'_{k,j}, l'_{k,j}, d'_{k,j})), (t'_{k,i}, L'_{k,i}, W'_{k,i}, ds'_{k,i})\}$$

The similarity measure Sim is computed as the sum of $SimC$ $SimO$ and $SimS$ representing the similarities measures of corners, overtures and stairs (see equations: 4.14, 4.11, , 4.12, 4.13) as follow:

$$SimC(D_q, D_p) = \sum_{k=0}^{nq}(\beta_1 * \|\alpha_k - \alpha'_k\| + \beta_2 * \|d_k - d'_k\| \tag{4.11}$$

$$SimO(D_q, D_p) = \sum_{p=0}^{no_k}(\beta_3 * \|t_{k,p} - t'_{k,p}\| + \beta_2 * (\|d_{k,p} - d'_{k,p}\| + \|l_{k,p} - l'_{k,p}\|)) \tag{4.12}$$

$$SimS(D_q, D_p) = \sum_{i=0}^{ns_k}(\beta_2 * (\|W_{k,i} - W'_{k,i}\| + \|L_{k,i} - L'_{k,i}\| + \|ds_{k,i} - ds'_{k,i}\|) + \beta_3 * \|t_{k,i} - t'_{k,i}\|)) \tag{4.13}$$

$$Sim(D_q, D_p) = SimC + SimO + SimS \tag{4.14}$$

β_1, β_2 and β_3 are the coefficients associated to each feature according to its importance in the determination of similarity measures between places (they were chosen on the basis of experimentation).

Finding of the best match

As the starting point of dataset descriptor for the comparison is unknown, the similarity measure is computed between D_q and D_p for all possible starting points j considering the clockwise and anticlockwise directions. For each dataset descriptor the measures are computed and the minimal one is selected and saved for the considered descriptor.

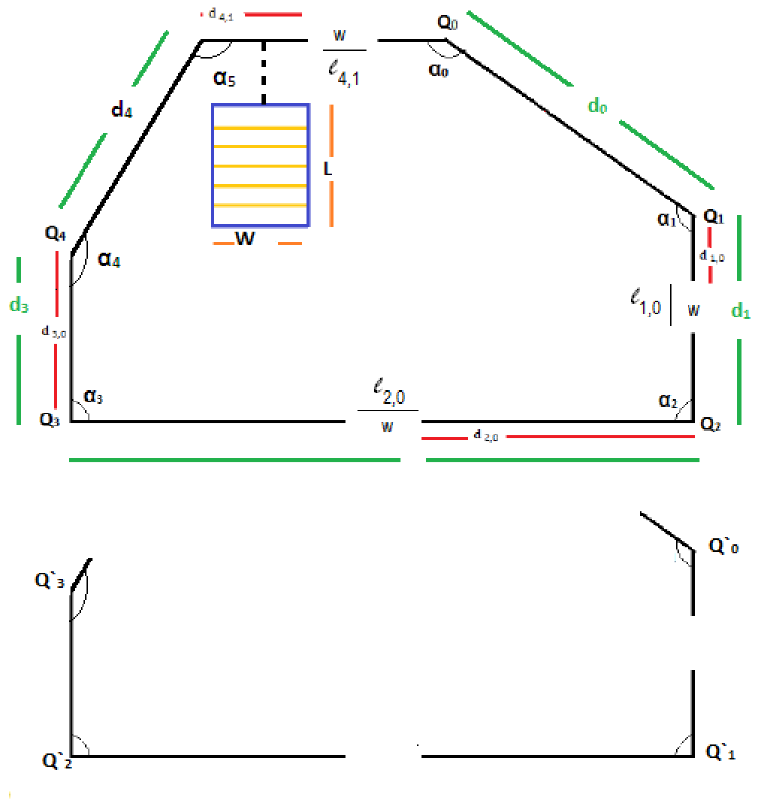

Figure 4.21: From top to bottom: 2D map of the dataset descriptor, 2D map of the query descriptor.

In figure 4.21, we present a 2D map of the dataset descriptor (shown on the top of the figure) and a map of the query descriptor (on the bottom).

The matching between these descriptors consists on computing the similarities measures between the query descriptor and the different parts of the dataset descriptor, where each part contains 4 corners (the query descriptor corners number). It considers each corner of the dataset descriptor as a starting point accompanied with its three successive ones in both the clockwise and anti-clockwise directions. The parts likely to be similar to the query are the succession of corners:

$(Q_0, Q_1, Q_2, Q_3;$
$Q_0, Q_5, Q_4, Q_3;$
$Q_1, Q_2, Q_3, Q_4;$
$Q_1, Q_0, Q_5, Q_4;$
and so on.
Once the measures are computed, the minimal one is saved for the considered dataset

descriptor.

In the same way, the difference measures with the remaining descriptors of the dataset are computed.

Then the one having a minimum score is considered as appropriate to the query and its location is transferred to that of the query.

Note that all possible states of apertures (open, close) were considered for comparison as illustrated by figure 4.22 (the openings are circled in red).

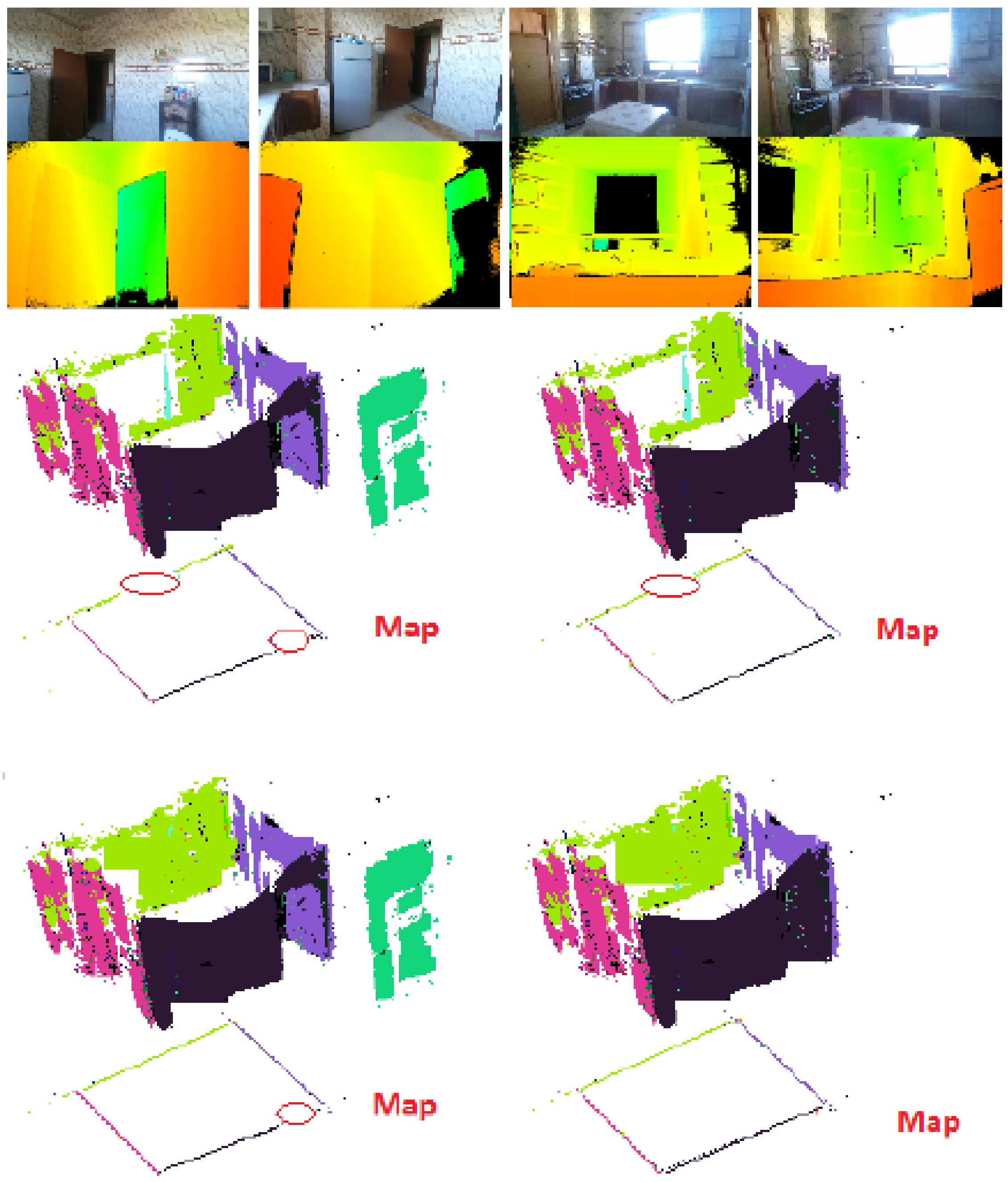

Figure 4.22: The 2D maps constructed in the same indoor scene considering the different possible states of its apertures: the door and window (open, close).

The Place Recognition steps are summarized in the algorithm 6.
Algorithm 6 performs the matching between the query descriptor and n dataset descriptors. Considering each corner of each dataset descriptor as a starting point in the two directions (clockwise and anticlockwise) so the number of comparisons is $2n$ (the dataset descriptor corners number). Each comparison involves the computation

Algorithm 6 Place Recognition Algorithm

```
Input: $D_q$: the query descriptor having $m$ corners
Output: $Location$ the query descriptor location
1: for each $D_s \in \mathbb{DS}$ do
2:     for each $Starting\ corner\ Q_j\ of\ D_s$ considering the clockwise and anticlockwise
       directions do
3:         Compute Similarity Measure $Sim(Q_j, D_s, D_q)$;
4:         $Select\ Sim^*\ such\ that\ Sim^*(Q_j, D_s, D_q)\ is\ minimal$
5:     end for
6:     $Save\ the\ location\ of\ D_s\ the\ similar\ to\ D_q$ ;
7: end for
8: return $Location\ of\ the\ query\ descriptor$
```

of similarity measure between the m corners. So, the algorithm in total executes $2nm$ iterations.
The algorithm complexity is then of the order of $O(n^2)$.

4.4 Conclusion

We have presented a new visual positioning system inspired from the architectural floor plan. It includes two main stages.

The first, for the scene description that takes as input a depth video acquired using a depth camera surrounded in the scene. From which a set of key depth frames are selected to locate and register the scene planes in order to gradually compute the 3D map, from which a 2D map is derived and used to deduce the scene descriptor.

The scene descriptor take into account the lengths of walls, the angles between walls, the opening in the walls so as the presence of stairs.

The second, for the place recognition process which is based on the matching principle. Its purpose is the identification of the unknown location.
By searching for the most similar dataset descriptor to the query descriptor in order to transfer its location to that of the query.
The most notable advantages expected from this system are the accuracy and the efficiency despite the frequent scenery modifications that the scenes may undergo and the illumination variations.

In addition to the stability of the system’s dataset as it does not need updates.
These foreseeable advantages are explained by the fact that the scene descriptors manipulated by this method are based on non-rearrangeable parts representing steady features over time.
This is what we are going to prove through the "Experimental results" chapter after the introduction of some details related to the system’s implementation.

5
Experimental results

5.1 Introduction

Through the previous chapter, we presented our visual positioning approach based on stable features which was inspired from the architecture.
And we will devote this one to the presentation of some details (in 5.2) and used tools (in 5.3) for the system implementation as well as the used dataset in 5.4.
Section 5.5 introduces the system's performance and effectiveness assessment. By disclosing the scenes descriptors evaluation in the first subsection 5.5.1 and the place recognition approach assessment in the second 5.5.2 accompanied with results of its comparison to the relevant state of the art methods (Fabmap [11] and FastABLE [120]). Then, we exibit the computational complexity results in the third 5.5.3.

5.2 Implementation details

5.2.1 Planar regions extraction

A planar region is a rectangular region defined by its top left pixel, its width and its height, it contains a set of points belonging to the same plane. It is defined by its mathematical equation:

$$ax + by + cz + d = 0 \tag{5.1}$$

Where the parameters a,b and c define the normal vector of the plane, d is the orthogonal distance of this plane from the origin of the reference system. The approach explained in diagram 5.1 is mainly based on the algorithm proposed in [93], it consists in dividing the depth image into several rectangular regions following a recursive quadtree subdivision. During the recursive iterations, the intermediate regions (R) are stored in a stack (S), they undergo two tests of flatness (flatness) and continuity (smoothness), if a region is large enough (its width is greater than T_{width}) and it satisfies both conditions it will be added set of planar regions (R_{plane}), otherwise it will be subdivided into four regions which will in turn be stacked in the stack (S).

5.2.2 Covariance matrix

The covariance matrix can be calculated using equation(4.6), but [117] offers a faster method, based on the calculation of integral images [121].
The integral image (I_O) is a cumulative representation of the image O, it has the same size as the original image, so that the value of each pixel $I_O(i,j)$ of the integral image is equal to the sum of the pixels of the original image located above and on the left of the pixel $O(i,j)$. More formally:

$$I_0(i,j) = \sum_{i' \leq i, j' \leq j} O(i',j') \tag{5.2}$$

Thanks to this representation, the sum of the values in a rectangular region can be calculated in only 4 accesses to the integral image, and therefore in constant time whatever the size of the region. This sum can be calculated by induction:

$$\begin{cases} S(i,j) = S(i,j-1) + O(i,j) \\ I_0(i,j) = I_0(i-1,j) + S(i,j) \end{cases} \tag{5.3}$$

Where $S(i,j)$ is the cumulative sum from row i to column j, in this way the integral image can be computed with a single pass through the original image.
Once the integral image has been calculated the sum of the pixels (i,j) (figure 5.2) of any rectangular region $R(p,l,h)$, defined by its upper left point $p(i_R,j_R)$, its length

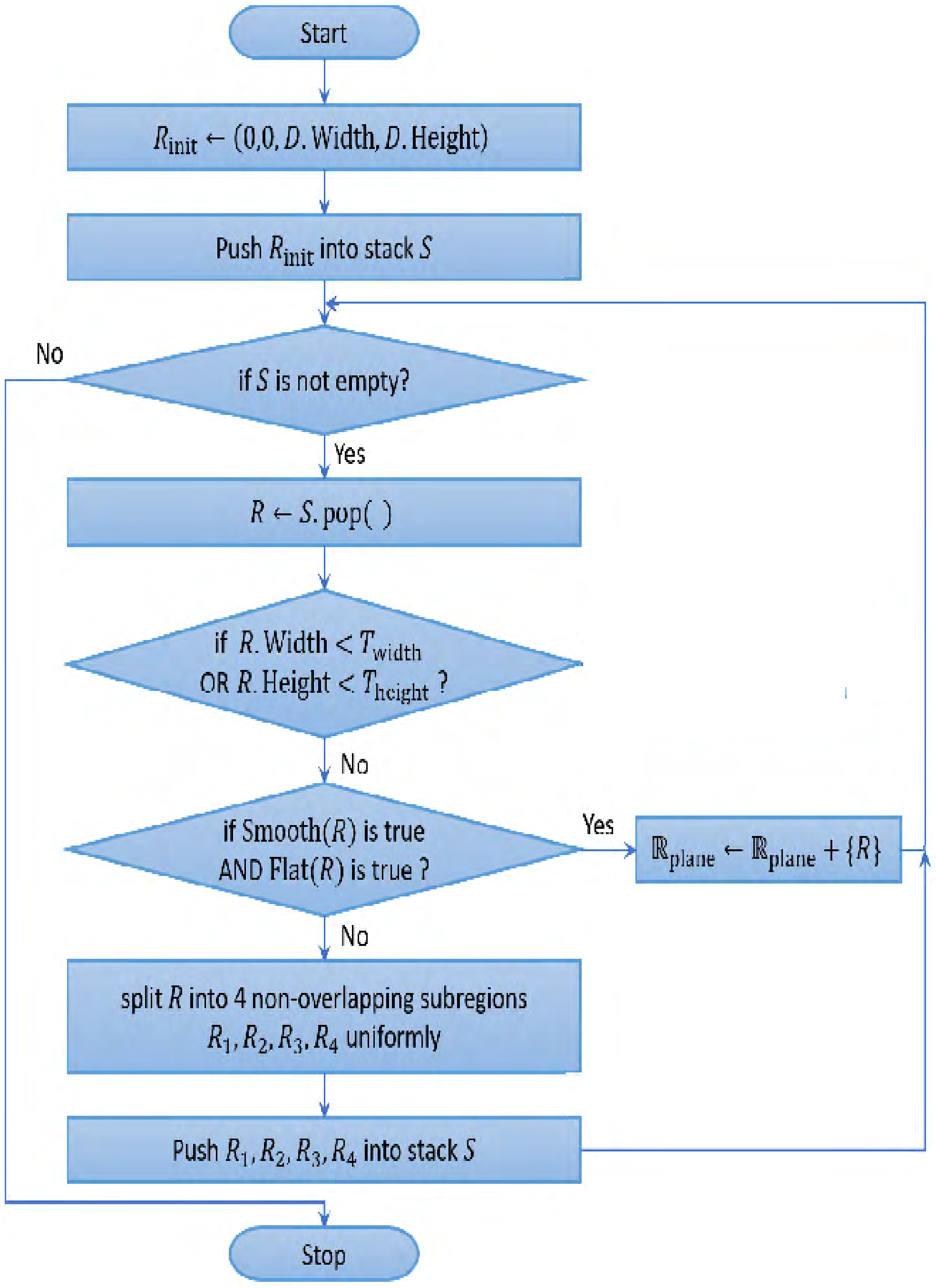
Start
$R_{\text{init}} \leftarrow (0,0, D.\text{Width}, D.\text{Height})$
Push R_{init} into stack S
No
if S is not empty?
Yes
$R \leftarrow S.\text{pop}(\ )$
if $R.\text{Width} < T_{\text{width}}$ OR $R.\text{Height} < T_{\text{height}}$?
No
if Smooth(R) is true AND Flat(R) is true ?
Yes
$\mathbb{R}_{\text{plane}} \leftarrow \mathbb{R}_{\text{plane}} + \{R\}$
No
split R into 4 non-overlapping subregions R_1, R_2, R_3, R_4 uniformly
Push R_1, R_2, R_3, R_4 into stack S
Stop

Figure 5.1: Image subdivision algorithm [93]

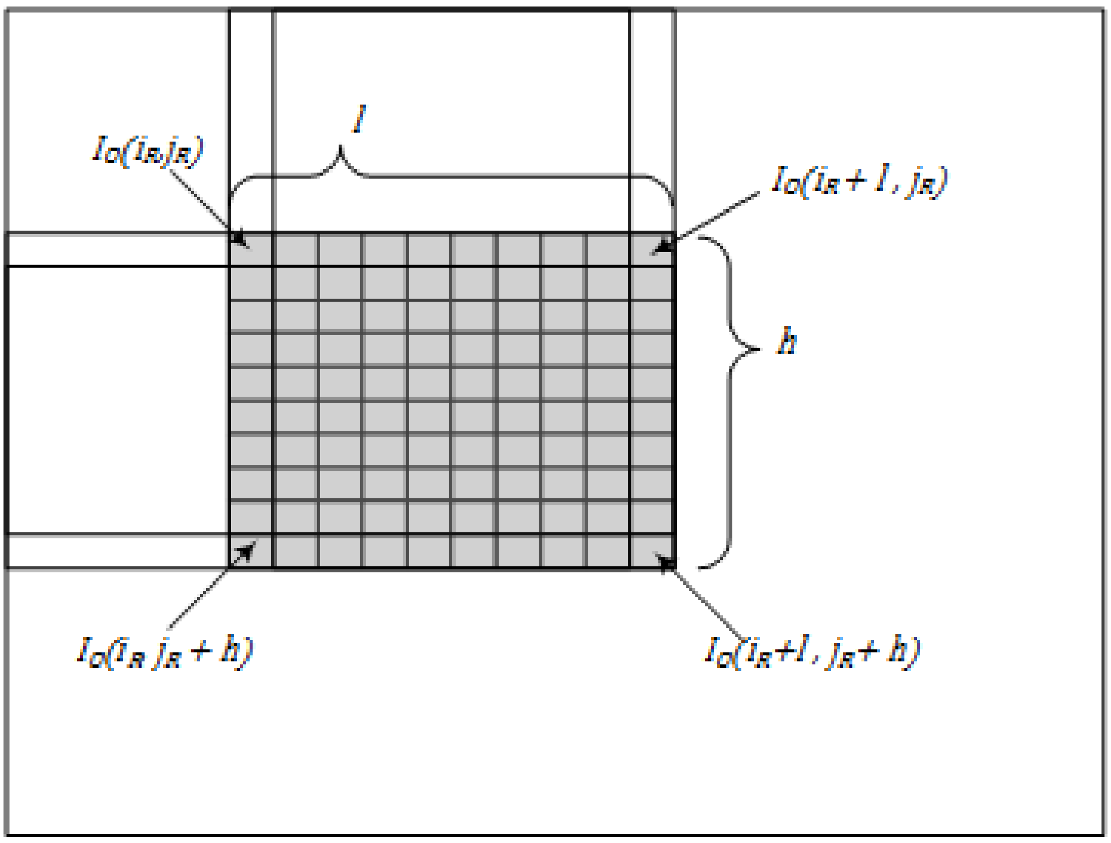

Figure 5.2: The pixels sum computation in a rectangular region located in an image.

l and height h, is evaluated with the formula:

$$\sum_{i_R \le i \le i_R + l, j_R \le j \le j_R + h} O(i,j) = \frac{1}{|R|}(I_0(i_R, j_R) + I_0(i_R + l, j_R + h) - I_0(i_R + l, j_R) - I_0(i_R, j_R + h) \tag{5.4}$$

To use this method, nine integral images are calculated corresponding to the sums necessary for the calculation of the covariance matrix 4.6, namely: I_{Ox},I_{Oy},I_{Oz}, I_{Oxx},I_{Oxy},I_{Oxz},I_{Oyy},I_{Oyz},I_{Ozz}, where O_x,O_y,O_z,O_{xx},O_{xy},O_{xz},O_{yy},O_{yz},O_{zz} are the images containing x coordinates, y, z and their products one by one.

5.2.3 Planar regions clustering

We built a data structure called "Region Compound Tree (RCA)", the latter includes four red and black trees, three of them contain the coordinates of the normal vectors of the regions previously estimated , and the fourth contains their distances to the origin of the camera coordinate system. The distance of a region R from the origin is calculated using the coordinates of the average position u of the region (Formula 4.5) and the equation

Algorithm 7 fusion of planar regions

```
Input: $R_{plane}$ //array of planar regions
Output: $ACR$
1: $ACR \leftarrow R_{plane}$
2: for $R \in R_{plane}$; $R \neq R_{plane}$ do
3:     $E_{nearest} \leftarrow ACR.nearestRegions(R)$
4:     IF ($E_{nearest} \neq \emptyset$) THEN
5:     $R.children \leftarrow E$
6:     $update(R)$
7:     ELSE $ACR \leftarrow R$
8: end for
```

of the plane(Formula 5.1) defined by its parameters a ,b and c previously calculated:

$$d = -(au_x + bu_y + cu_z) \tag{5.5}$$

In this way, for a given region, the search for the region closest to it in the tree is done in an optimized time. We added an array $R.children$ to the structure of the region R, to indicate the references of the regions which belong to the same plane as it. Before inserting a planar region R in the compound tree ACR, we compare it with the regions already inserted, if there is a region R_{close} which belongs to the same plane as R, we insert R in the array $R_{close}.children$, if not, we insert the R region in the ACR tree.

- Planar regions comparison: Two regions are considered as belonging to the same plane if the conditions in 4.9 are fulfilled.
- The update function: is used to re-evaluate the parameters of a region each time another region is added as a child of it. At the end of the Algorithm execution 7, the ACR tree will only contain the planar regions belonging to distinct planes, and the array (children) of each one will contain the regions belonging to the same plane as this one.

5.2.4 Polygons construction

We have chosen the "Clipper" library to unite several planar regions belonging to the same plane into polygons, this library was chosen for:

- Its efficiency in processing polygons with holes (figure 4.5).
- The availability of its implementation in open-source.
- the polygonization of a set of non-adjacent regions, part of the same plane, gives a set of linked polygons and not a single polygon. Which is a supported structure

by Clipper library.

Clipper library terminologies

- Path: corresponds to an ordered sequence of vertices defining a single geometric contour which is a line (an open path) or a polygon (a closed path).
- Line: or polyline is an open path containing two or more vertices.
- Polygon: usually refers to a two-dimensional area bounded by a closed. This area may also contain a number of "holes".
- Outline: synonym of path.
- Hole: is an enclosed area in a polygon that is not part of it. A "hole polygon" is a closed path that forms the outer boundaries of a hole.
- the Polygon Filling Rule: with a list of closed paths, define which areas (bounded by paths) are inside and which are outside (holes), paths with a sequence clockwise are by convention inside, the opposite for holes (figure 4.5).
- Union: operation of merging two or more polygons to obtain a single one, this operation eliminates collinear points and duplicate lines (figure 4.5).

To apply the union on a set of co-planar regions, it suffices to construct a path for each region containing its four angular points (the corners), and pass these paths to the Clipper library, the result of this operation is a set of polygons (figure 4.5).

At the end of this step, all the subsets of the planar regions are represented as 2D polygons (in image space) then the 3D coordinates (camera space) of the vertices of these polygons are computed using the previously presented formula. The (3D) polygons obtained are supposed to be flat (their vertices belong to the same plane), but because of the noisy nature of the Kinect data, we do not obtain perfect polygons see the right side of figure 5.3. To overcome this problem, the positions of polygon vertices have been projected onto their respective planes (left side of figure 5.3).

The planar regions extracted from the depth image shown in figure 5.4 whose characteristics are illustrated in figure 5.5, are given in figure 5.6.

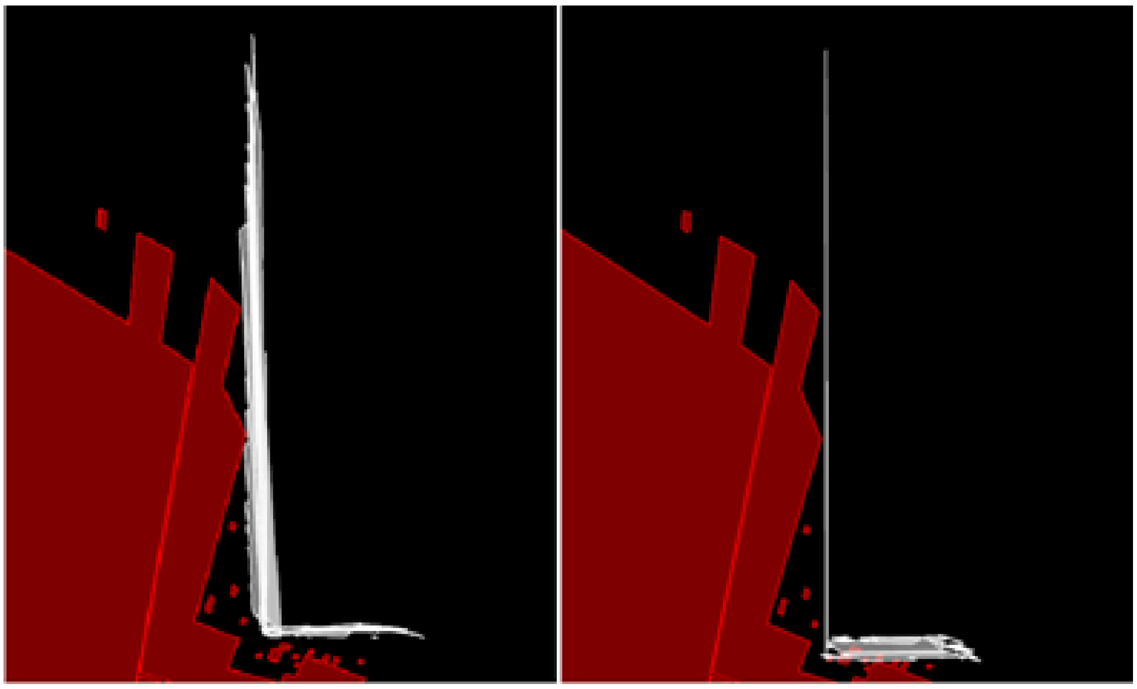

Figure 5.3: Projection of the polygons on their respective planes

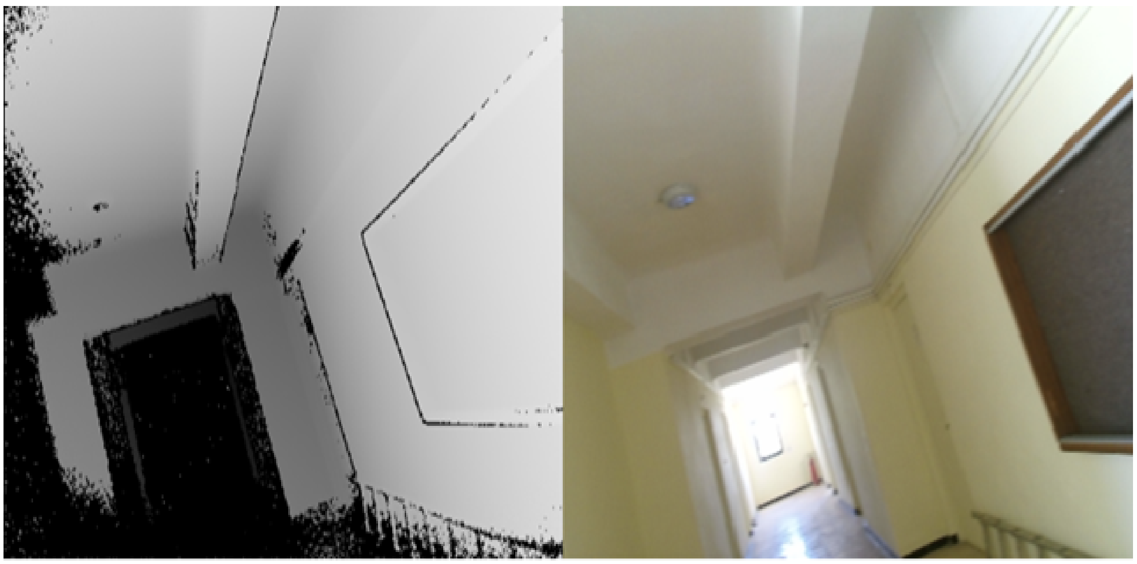

Figure 5.4: RGBD image.

In figure 5.4 the RGB image (on the right side) was added just for a better visualization of the captured depth scene image. And the depth image (in the left side) shows the pixels farther from the camera with a darker color, while black color represents absence of objects on these points.

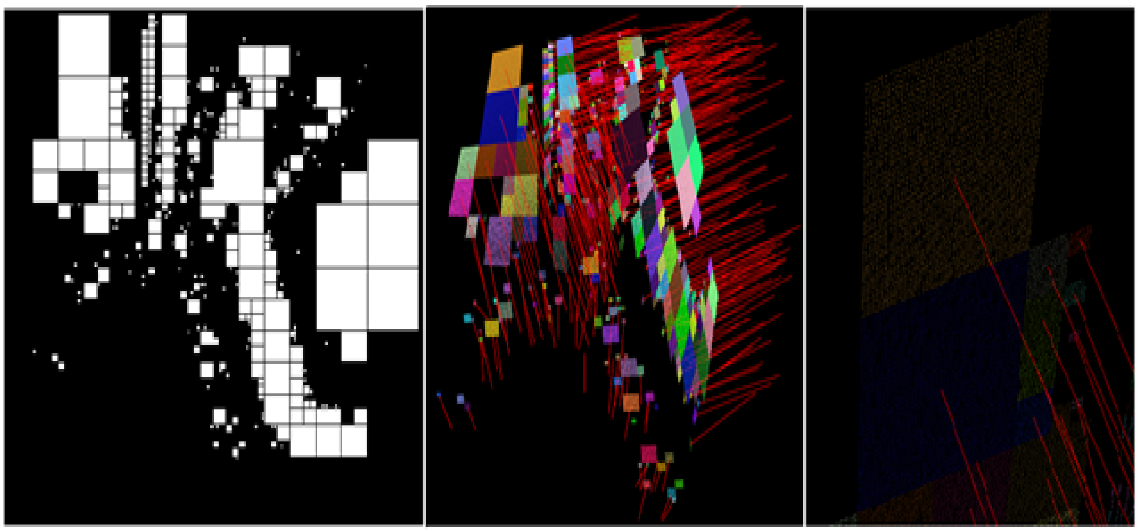

Figure 5.5: planar regions extraction.

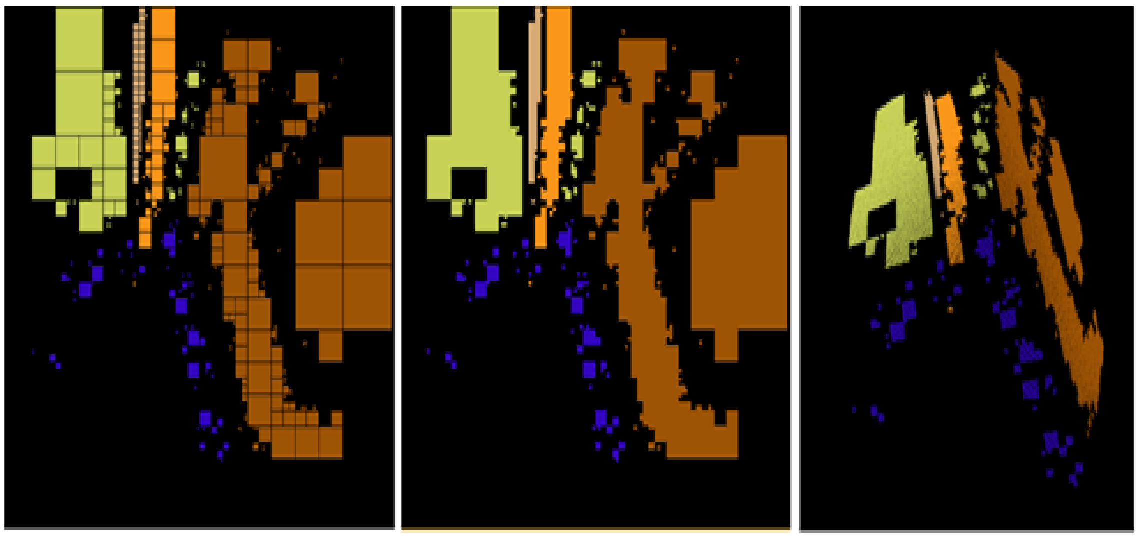

Figure 5.6: planar regions clustering.

We note that the depth information from the Kinect is very noisy, this noise is accentuated according to the distance of the objects from the camera and it is useful for image corners.
Figure 5.7 shows a part of a polygon constructed from a subset of coplanar regions, in this step a set of polygons are constructed for each captured depth image and all points and regions data are removed to liberate the memory.

Figure 5.6 illustrates the merging results of regions belonging to the same plane colored

Figure 5.7: polygon construction.

by the colors of their respective regions (a random color has been assigned to each region's subset to distinguish it). The right side and the central are in 2D space while the left side is in 3D space.

5.2.5 3D map Computation

After extracting planar regions defined by data structures which includes for each region, the following parameters:

- The position and dimensions of the region;
- The parameters of the plane which carries the region;
- The points center gravity.

In this step we will use this data to apply the variant ICP (point-to-plan). The algorithm has four main steps; we have adapted each step to our case. The ICP is an iterative algorithm (see figure 5.8, algorithm 6), so each iteration uses the results obtained at the last iteration to give a better result, until the result converges (distance less than M_{dist}) or although we have exceeded the maximum number of iterations N_{it} allocated to the algorithm.

We can formulate the problem that we are trying to solve by the ICP algorithm as follows: Let $E_{source} = s_i$, $E_{target} = c_i$, the sets of points from the source image and the target

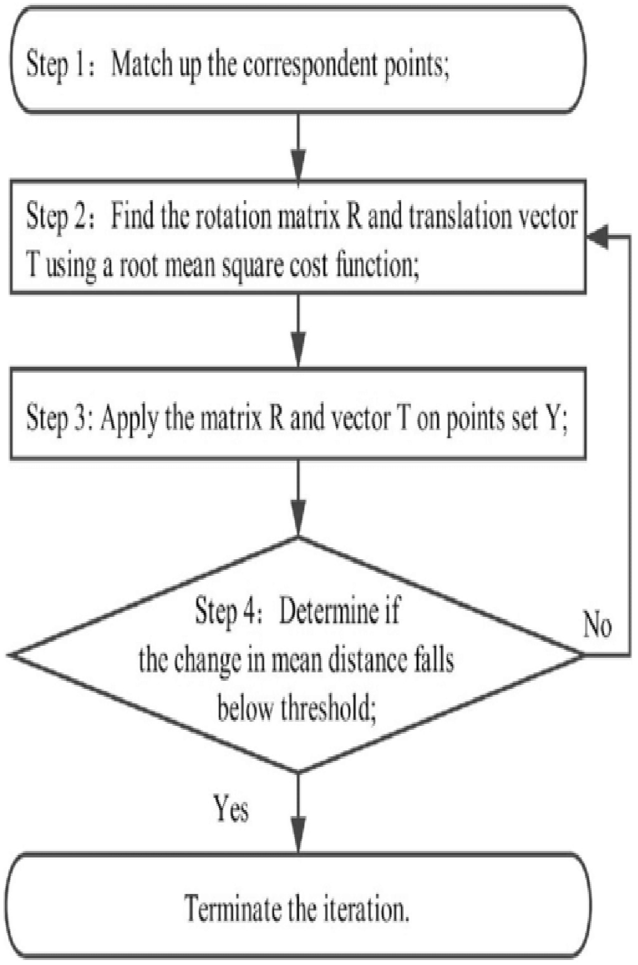
Step 1: Match up the correspondent points;
Step 2: Find the rotation matrix R and translation vector T using a root mean square cost function;
Step 3: Apply the matrix R and vector T on points set Y;
Step 4: Determine if the change in mean distance falls below threshold;
No
Yes
Terminate the iteration.

Figure 5.8: ICP algorithm diagram

image respectively, each point s_i is linked to its corresponding c_i by the relation:

$$c_i \approx Rs_i + T \tag{5.6}$$

Where R and T are the rotation and the translation of the transformation between the two sets E_{source} and E_{target} , the distance between them is $Dist$, which can be formulated as:

$$Dist = \sum_{i=1}^{n} ||c_i - Rs_i - T|| \tag{5.7}$$

The goal of the ICP algorithm is to find the transformation that minimizes this distance $(Dist)$.

Selection

Points reduction was made with the extraction of the planar regions (section 4.2.2), indeed the application of the algorithm on the totality of the points (resulting points cloud from depth images) render the operation slow and does not meet the fixed objective (execution in real time), that's why the input data of the ICP algorithm will be:

- The gravity centers (named point in the rest of this chapter);
- The normal vectors that are necessary to compute the distance of points from the source to the target image in each iteration to estimate the correspondences.
- The curvatures for more precision in the correspondence choice.

Match estimation

The match estimation is the process of matching points s_i in the source point cloud E_{source} to their nearest neighbors c_i in the target cloud E_{target}. That's a greedy approximation to find the ideal correspondences (s_i, c_i) between the two point clouds.

A naive way to find the nearest neighbor is to perform an exhaustive search in all the target points of the nearest neighbor of each source point, various optimized data structures have been proposed for fast searches, such as red and black trees (3.11) and k-d trees (3.8). These data structures have logarithmic search times, $O(NlogN)$ for k-d trees.

For more precision during the search, the ICP algorithm can perform a reciprocal search (Reverse), (after having determined the corresponding points of the source image in the target image, we seek the corresponding points of the target image in the source image, in this way we filter the points which change corresponding after the inverse operation).

Rejection (filtering)

This operation consists in rejecting the points which can falsify the estimate of the optimal transformation, part of these points is the set of points located at the edges of the image and which do not belong to the two images, in other words the points which do not belong to the intersection of the two images must be rejected from the matching operation. As it is assumed that the captured images are temporally and spatially close, we can set a distance threshold between two regions (their centers) at Mx_{dist}; all correspondents whose distance exceeds this threshold are rejected.

Alignment (transformation matrix)

We will assume that the transformation is (isometric) rigid (the scene does not present any deformation between the two image captures, so the dimensions of the scene remain the same regardless of the angle of capture). The objective of this operation is to minimize the error between two point clouds (point-to-point error minimization). Consider the point clouds $AP = (p_i)$,$i \in 1 \ldots n$ and $Q = (q_i)$, $i \in 1 \ldots n$, the goal of the alignment operation is to find the transformation of Q (translation and rotation) which will make it possible to minimize point-to-point distances (in the sense of least-squares minimization).

Given the set of pairs $(q_i; p_i)$ representing the relationship between the two sets P and Q. As already commented, the goal of the ICP algorithm is to find the best transformation between two sets of points minimizing the distance between them. so, what we should do is to solve the equation 5.7 which represents this problem.

- Translation calculation: Represents the translation between their centers calculated using formula 4.5.
- Rotation calculation: The solution of the minimization equation 5.7 is calculated using the singular value decomposition [122] of the rotation matrix R.

Data fusion and representation

A "globalPolygons (GP)" vector is provided to contain the polygons of the whole scene, as the capture of the scene progresses, the polygons extracted from each depth image are inserted into this vector, after transforming them by the corresponding global transformation.

Knowing that each polygon has its own parameters (normal vector, distance from the center of the coordinate system, the paths of the polygons it includes (see 4.2.2), its surface), before inserting a polygon a search is made to check if this polygon belongs to the same plane as another polygon already existing, to do this an algorithm similar to that of the merging of planar regions (4.2.3) is used.

We build a data structure "Tree Composed of Polygons (TCP)", composed of four red and black trees, three of them contain coordinates of the normal vectors of the polygons, and the fourth contains the distances of this latter with respect to the origin of the camera coordinate system, each node of the trees contains a reference of the corresponding polygon in the vector of polygons GP.

Before inserting a polygon g into the TCP, we compare it with the polygons already inserted, if there is a polygon g_{close} which belongs to the same plane as g, we replace g_{close} by the union of the two polygons g and g_{close}, if not, we insert the polygon g in the TCP tree.

The calculation of the polygon resulting from the union of the two polygons g and g_{close} is done with the Clipper library. Even this latter is able to perform operations on the polygons in 3D, it is much more efficient in 2D, to take advantage of this efficiency we calculate the union of the polygons in 2D that's why the polygons are firstly transformed from 3D to 2D (for more details refer to section A.1) and once the result is obtained in 2D, it is expressed in the 3D coordinate system. Indeed, the operation of orthogonal projection on the plane. Algorithm 8 summarizes the fusion method used.

5.3 The used tools

To allow the subsequent use of our project, we exclusively followed the object-oriented paradigm, in fact, this modular programming mode allowed the adaptation of the code according to the need and the acquisition material available, for example, to use another camera model than Kinect V2, it suffices to adapt the class responsible for reading the data.
The development is entirely done under the Microsoft Visual Studio version 19 environment, using the C++ programming language version 14, the latter is essential in the fields of robotics and artificial vision, the main reason behind this choice is the fact that the C++ language can be used on all platforms (Windows, Lunix, OS... etc.) and all hardware devices (PC, mobile device, on-board card... etc.), in addition to this the speed and optimization of calculations.
The Kinect Studio V2.0 tool which is part of the "Kinect SDK" was also used for recording captures and emulating the Kinect, in fact, we were not obliged to use the Kinect for the development and debugging

Algorithm 8 Polygons fusions

```
Input: $G_i$ //set of polygons of the image $I_i$
 1: for $g \in G_i$ do
 2:     $g_{nearest} \leftarrow ACP.nearestPolygon(g)$

 3:     if $g_{nearest} \neq \emptyset$ then
 4:         $ACP.delete(g)$;

 5:         $M = projection(reference_{3D}, plane_g)$;

 6:         $g.path_{2D} = g.path_{3D} * M$;

 7:         $g_{union}.path_{2D} = g.path_{2D} \cup g_{nearest}.path_2D$;

 8:         $g.path_{3D} = g.path_{2D} * M^-1$;

 9:         $ACP \leftarrow g_{union}$;
10:     else
11:         $g.path_{2D} = g.path_{3D} * M$;

12:         $ACP \leftarrow g$;
13:     end if
14: end for
```

Libraries

Microsoft Kinect SDK

The Kinect Software Development Kit (SDK) 2.0 for Windows enables developers to create apps that support gesture and voice recognition, using Kinect sensor technology on Windows.

There are open source alternatives to this library (libfreenect, OpenNI+SensorKinect) that allow the acquisition of Kinect data under other operating systems than Windows. As part of our project, we only used the "DepthBasics" class, which serves as an interface to read RGB-D images from the Kinect.

OpenCV

The name of this library is coming from the acronym (Open Computer Vision), it is a free graphics library specializing in image processing, we needed this library, in particular the "Mat" class to contain RGB-D images, DCI Map (4.2.2) and integral images (4.2.2), this class provides easy and optimized ways to read, write, browse images and matrix calculation (4.2.2). It was also a great help in debugging codes, with its ability to display images graphically.

PointCloudLibrary

The Point Cloud Library (PCL) [123] is an open source project for image and point cloud processing, it contains the implementation of many algorithms for pattern recognition, segmentation, noise elimination. . . etc.
We used this library for registration with the ICP (5.6) algorithm.

Clipper Library

The open-source Clipper library is used to perform line and polygon clipping, logical operations (intersection, union, exclusivity, etc.) on lines and polygons.
This library is based on the splitting algorithm of Vatti [124].
We needed this library to merge the polygons extracted from RGB-D images using its ability to perform union operations between polygons, it was also used to calculate the surfaces of the latter.

OpenSceneGraph (OSG)

Is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling.
Entirely written in standard C ++ and OpenGL, despite the existence of many display libraries, we preferred this one because of its simplicity, indeed it represents a good compromise between richness in functionality and efficiency.
We used this library for the 3D display of point clouds, lines, polygons, etc.

5.4 The used datasets

The public datasets may be classified into two classes.

The first, encloses those that contain unsorted images list [125] [126] suitable for evaluation of 2D based methods and neural network based methods, they furnish query images captured under various conditions.
The second for sorted images lists [84] destined for 3D based methods assessment. They do not consider big changes between query and database images due to relying on feature matching for ground truth generation.

None of these data-sets can be used to evaluate our system based on coarse scene information, because the evaluation of this later requires a set of videos (sorted images lists) taken under various acquisition conditions and these datasets do not furnish the required data.

So we built our dataset [127] by capturing depth videos of different indoor scenes using the depth camera.

For each scene, We have acquired videos for the dataset covering all scenes details. And query videos overcasting different scenes parts from different positions and under various conditions in order to be used for evaluation.

The recorded videos were used to compute the scenes descriptors D_{scene} (using the explained method in the previous chapter subsection 4.2.5) then the couples $\{D_{scene}, Location\}$ ($Location$ indicate the identification of scene) were stored in the database accompanied with RGB images (saved for evaluation to compare the results of proposed system against those of the state of the art methods).

Through figure 5.9, we present an example of indoor working environment with scenes of various geometric forms and dimensions (rectangular (scene 01), squarish (scene 00), T (scene 06), L (scene 08), N (scene 07) shapes, scenes of small surfaces $(10m^2)$, medium $(30m^2)$ and large$(70m^2)$).

5.5 Experiments

The used evaluation metrics

We use the Precision, Recall and F1 score which measure the correctness of system predictions. The precision-recall curve illustrates the tradeoff between precision and recall. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results. Precision P is defined as follow:

$$P = \frac{T_p}{T_p + F_p} \tag{5.8}$$

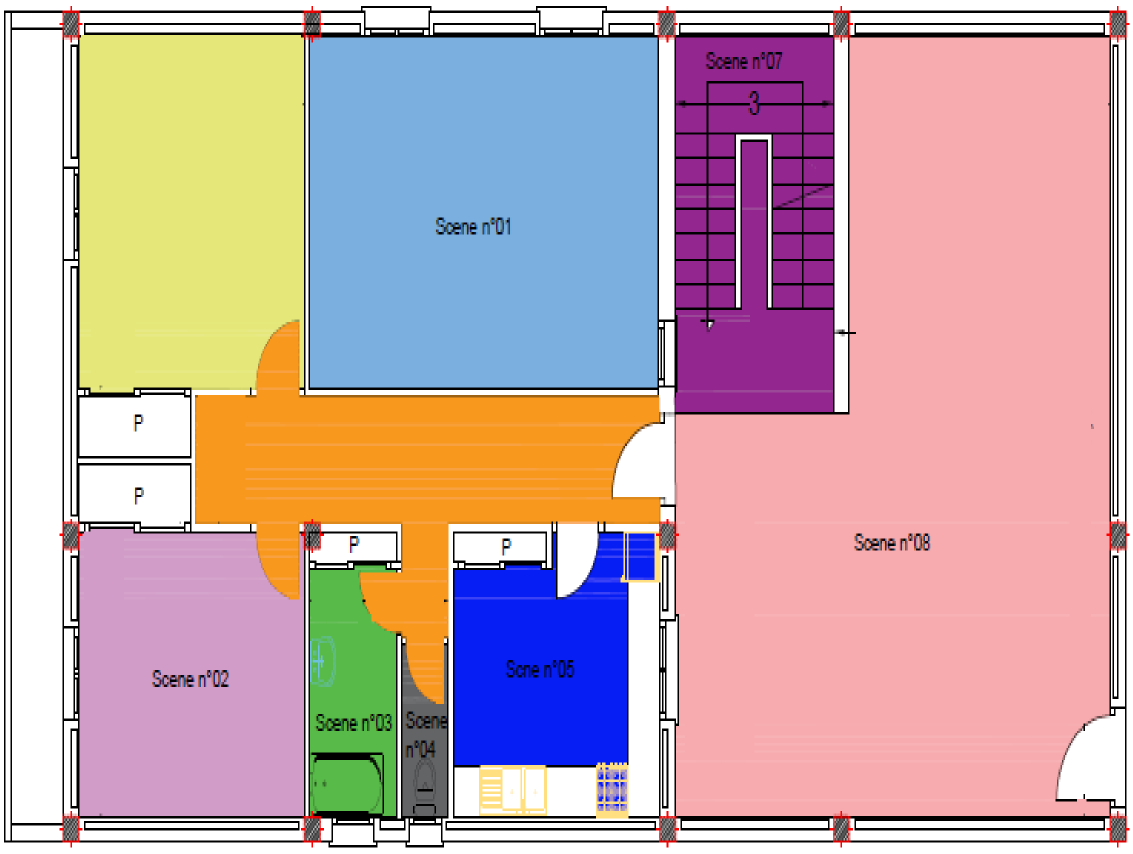

Figure 5.9: Experimental environment floor-plan.

Recall R is defined as:

$$R = \frac{T_p}{T_p + F_n} \tag{5.9}$$

Where T_p is the true positive, F_p false positive, F_n false negative. These quantities are also related to the F1 score, which is defined as the harmonic mean of precision and recall.

$$F1 = 2\frac{P \times R}{P + R} \tag{5.10}$$

5.5.1 Scenes descriptors computation

As the proposed descriptors represent plane views of the scenes, we have evaluated their precision by comparing the derived 2D maps from which these descriptors were computed against real plane views.

Figures 5.10, 5.11 and 5.12 show the average error and standard deviation of calculated dimensions, angles and overtures in 2D maps of seven scenes sample.

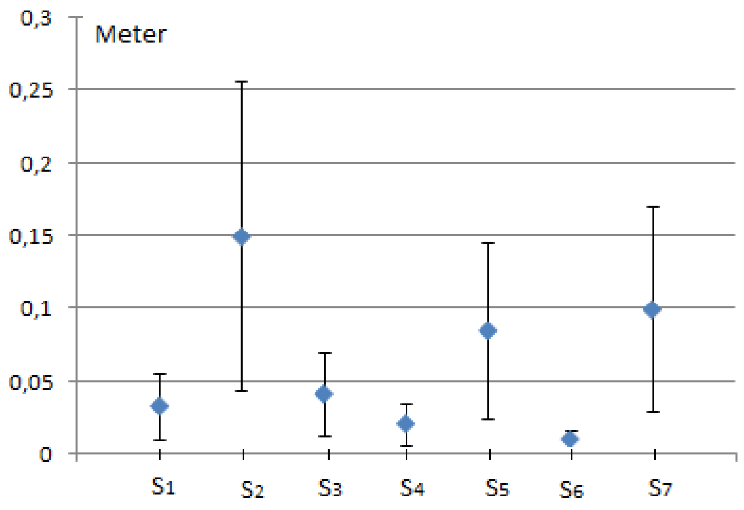

Figure 5.10: Average errors and standard deviations for computed walls lengths for seven scenes $(s_1, ..., s_7)$.

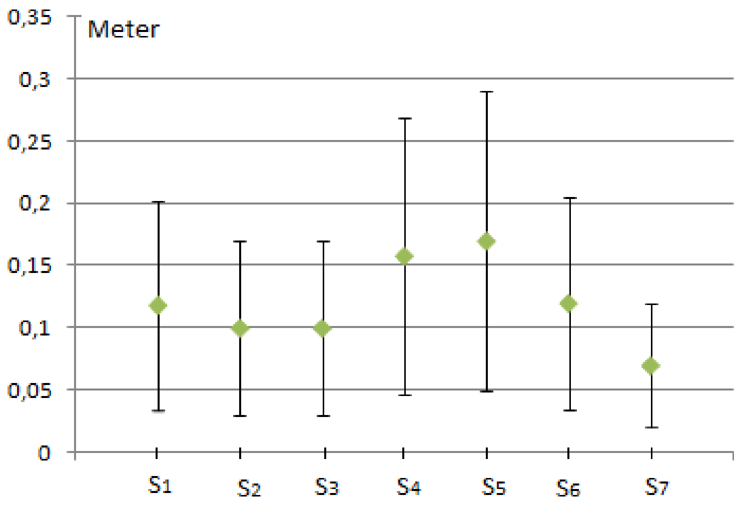

Figure 5.11: Average errors and standard deviations for calculated apertures for seven scenes $(s_1, ..., s_7)$.

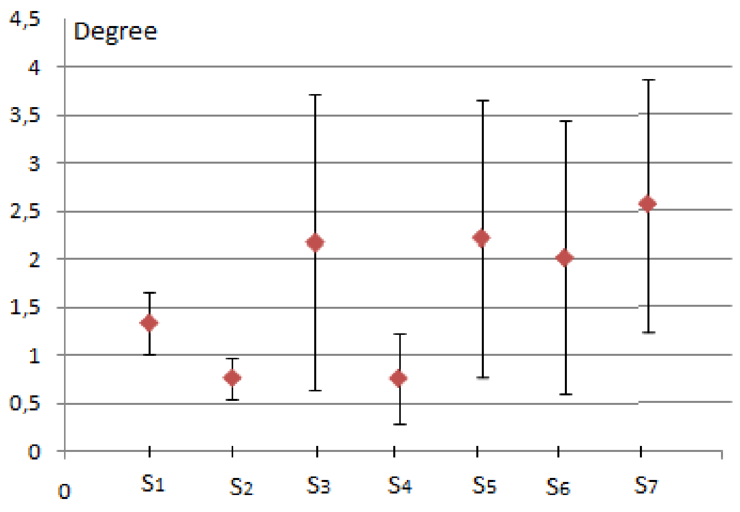

Figure 5.12: Average errors and standard deviations for calculated angles for seven scenes $(s_1, ..., s_7)$.

Compared to the ground truth, computed measures from 2D maps are accurate (see figures 5.10, 5.11 and 5.12), moreover the doors overtures have been correctly identified and distinguished from those that represent windows as shown in figures 5.13 and 5.14 where the orange plane in 5.14 was not represented in the 2D map because only walls planes were projected.
Note also that apertures dimensions are independent of the sash opening (in whole or in partial) because they are defined as overtures of the same plane.

5.5.2 Evaluation of Place Recognition Approach

Evaluation results

Only our constructed dataset was used to evaluate the system performance because none of the public datasets furnish the processed data by our system. The data was organised on two sets of depth videos :
- The first, contains those that was used to construct model descriptors,
- The second for test data which may be subdivided into eight subsets considering light variations, scenery changes, appearance and disappearance of movable objects and persons (like summarized in table 5.1) in order to investigate the effect of changing acquisition conditions.

We have selected open source algorithms for comparison to establish the evaluation

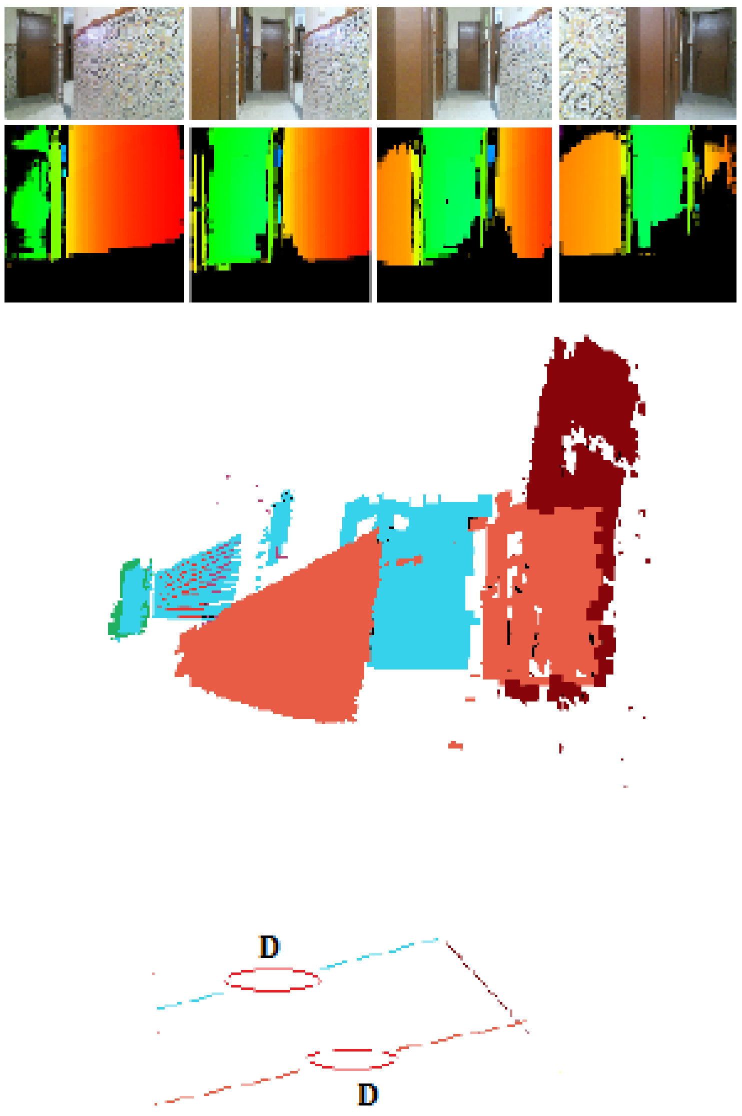

Figure 5.13: Overtures of doors correctly identified.

of both state-of-the-art methods and the proposed method on the same constructed dataset for a reliable comparison.

Among these open source approaches we have selected the most relevant: FastABLE [120] and FABMAP [11] for a complete comparison considering a method that uses a sequence of images (FastABLE) and a method that utilises a single image (FABMAP).

The precision-recall curves of the three approaches are presented in figures 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22 and 5.23 for each subset of test data and in table 5.2 the F1

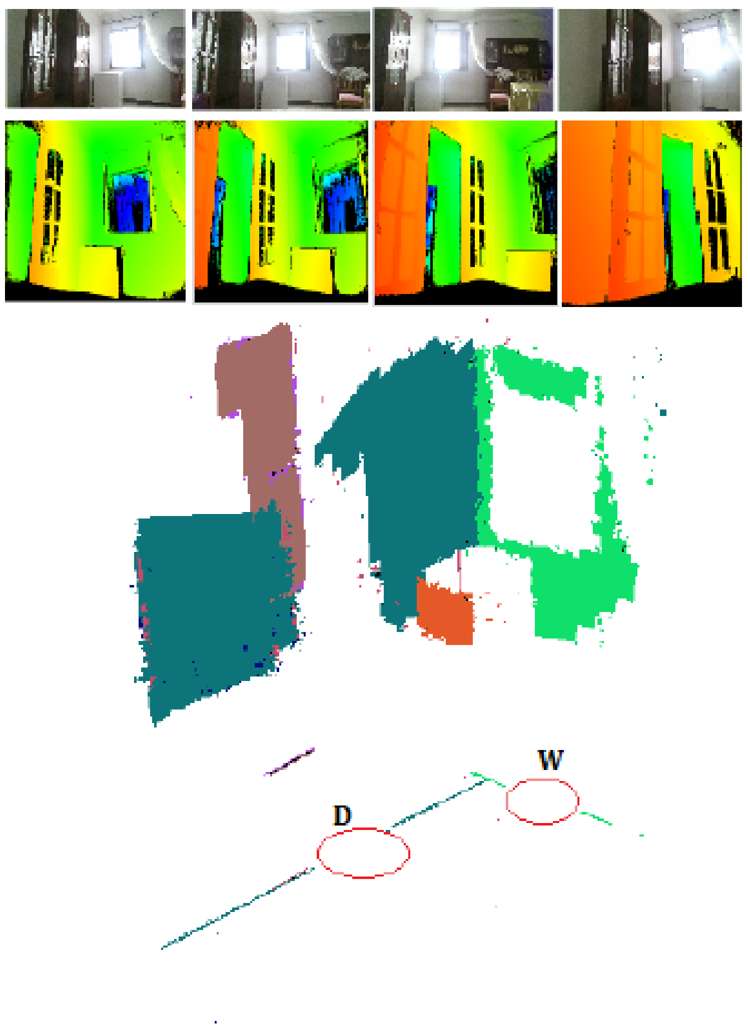

Figure 5.14: Overture of door and window correctly identified and distinguished.

scores are given.

The exposed results in figures 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22 and 5.23 and table 5.2 show that:

- For the first subset, FastABLE and FABMAP offer better results than the proposed approach;

- In case of scenery changes, the place recognition results decrease in both state of the art methods and still stable in our approach;

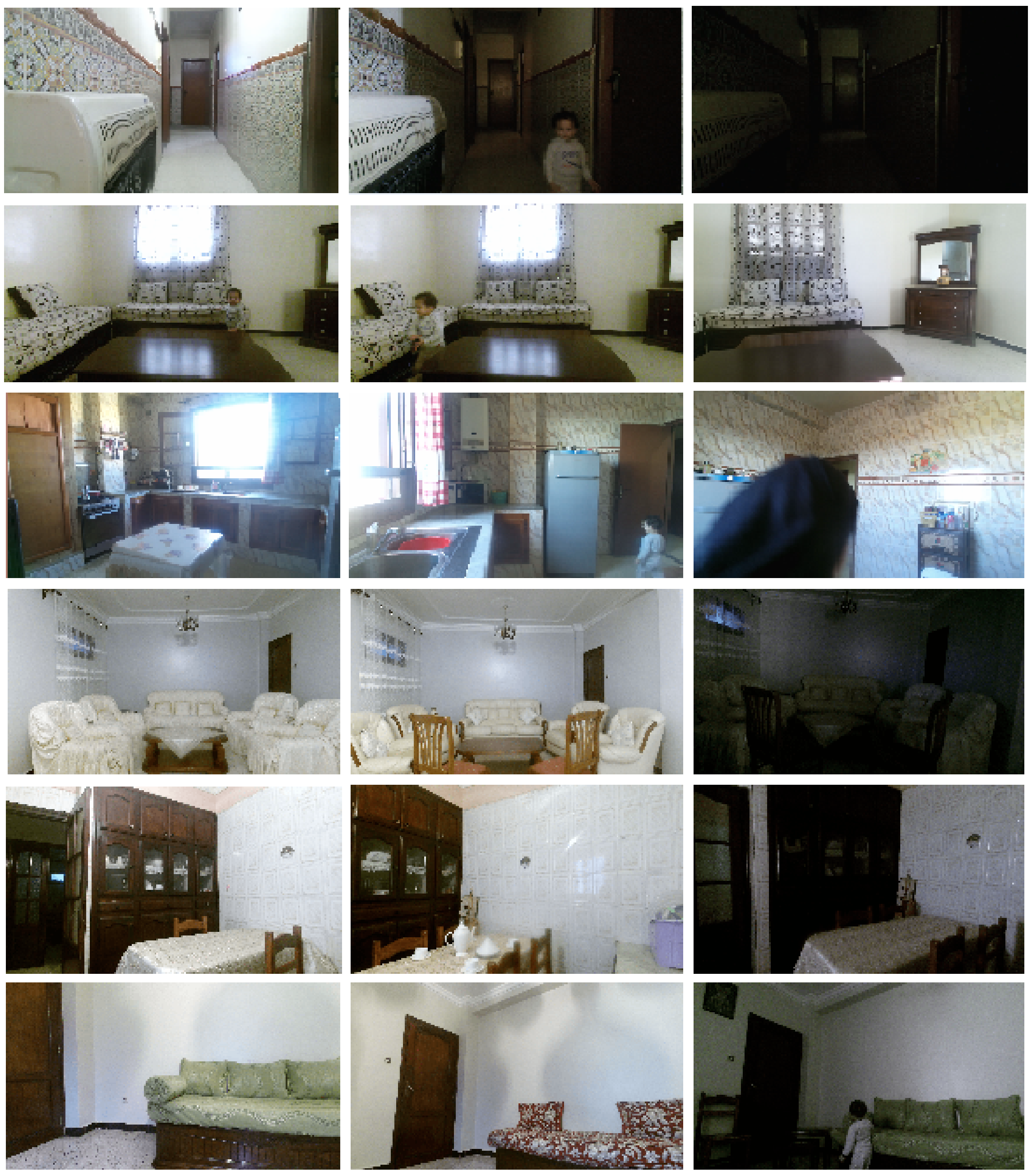

Figure 5.15: Acquisition conditions of test data.

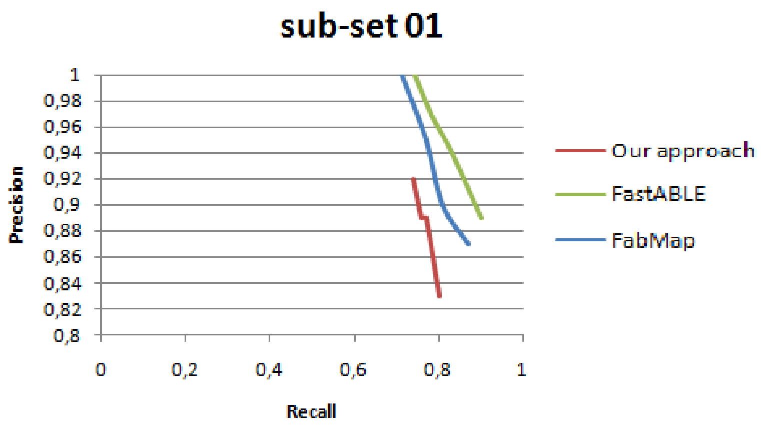

Figure 5.16: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 01 of test data.

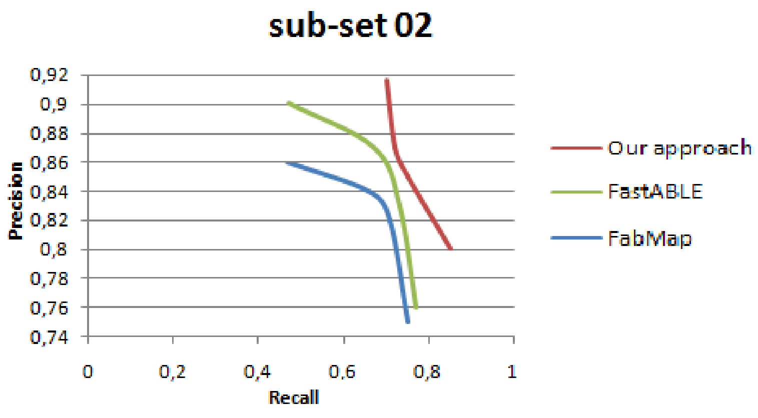

Figure 5.17: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 02 of test data.

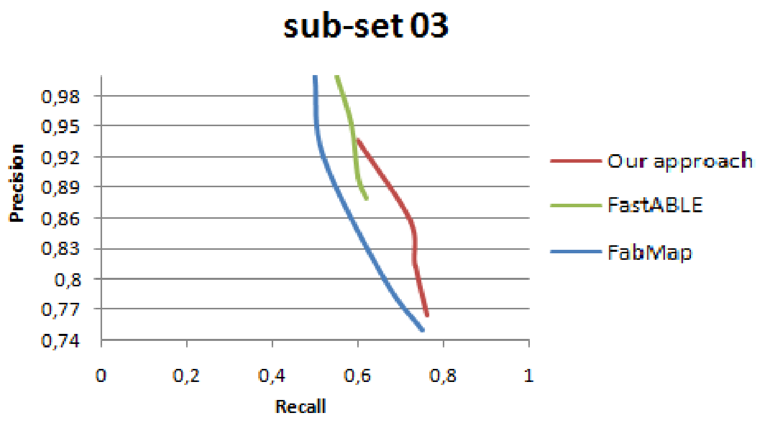

Figure 5.18: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 03 of test data.

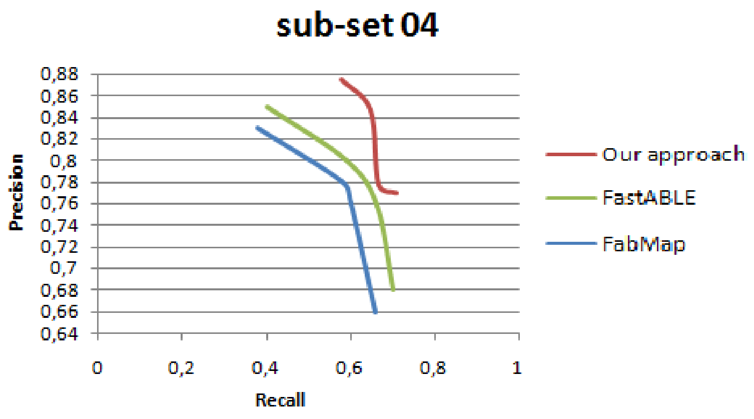

Figure 5.19: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 04 of test data.

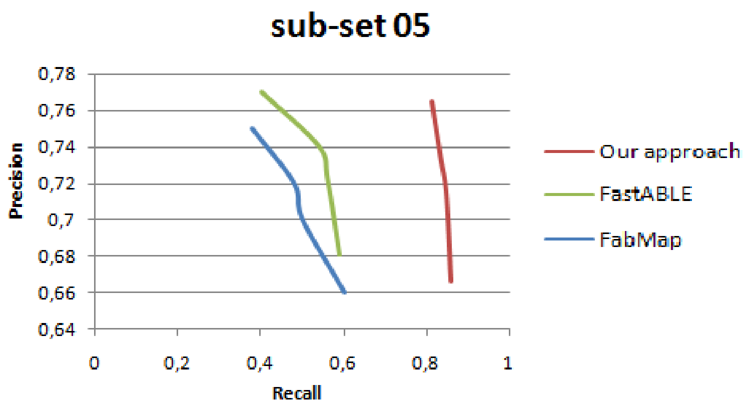

Figure 5.20: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 05 of test data.

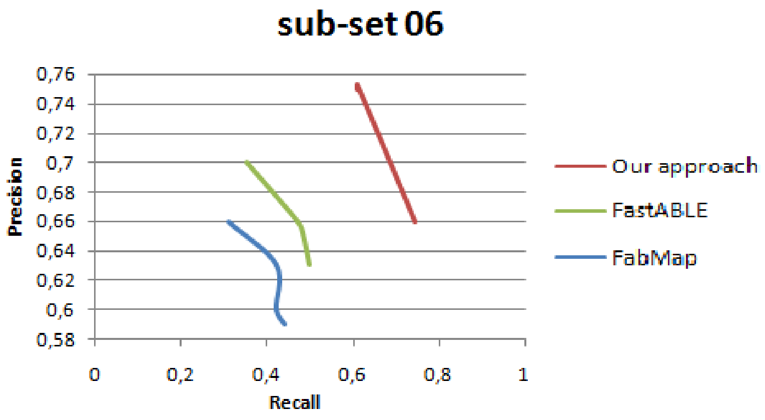

Figure 5.21: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 06 of test data.

Scene	lighting variation	Dynamic objects	Scenery changes
Subset 1	No	No	No
Subset 2	No	No	Yes
Subset 3	No	Yes	No
Subset 4	No	Yes	Yes
Subset 5	Yes	No	No
Subset 6	Yes	No	Yes
Subset 7	Yes	Yes	No
Subset 8	Yes	Yes	Yes

Table 5.1: Subsets of data test.

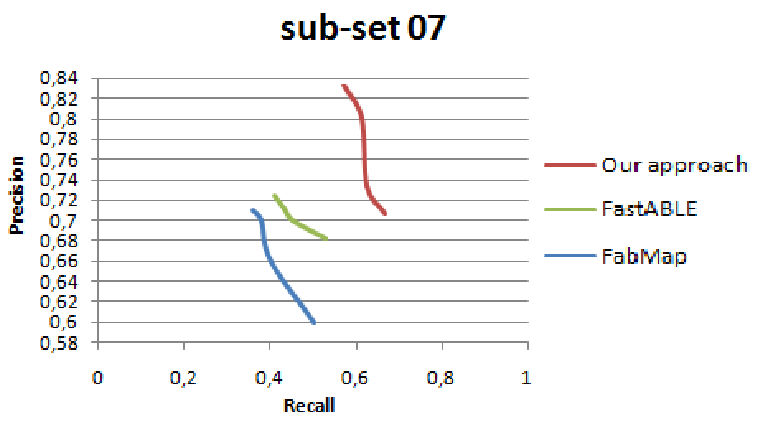

Figure 5.22: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 07 of test data.

- Concerning sensitivity to light changes, FastABLE and FABMAP are more responsive than the proposed method, figure 5.24 shows 3D and 2D maps constructed in very low light scene in order to affirm that our system remains accurate even in very low light.

Subset	Proposed approach F1	FastABLE F1	Fabmap F1
Subset 1	**0.8210**	0.8762	0.8549
Subset 2	**0.8071**	0.7399	0.7220
Subset 3	**0.7669**	0.7213	0.7146
Subset 4	**0.7243**	0.6681	0.6406
Subset 5	**0.7739**	0.6083	0.5789
Subset 6	**0.6890**	0.5353	0.4788
Subset 7	**0.6852**	0.5528	0.5080
Subset 8	**0.6639**	0.4466	0.4193

Table 5.2: F1 scores of the proposed system, FastABLE and Fabmap.

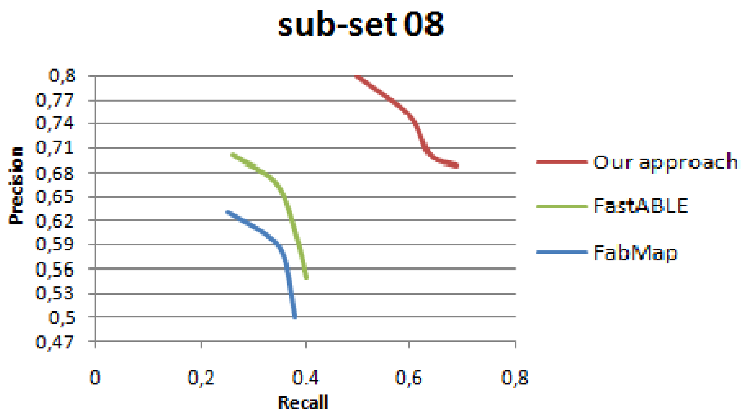

Figure 5.23: Precision-recall curves of the proposed method, FastABLE and FABMAP for the subset 08 of test data.

- The dynamic objects and persons in locations decrease slightly the place recognition results in the three methods;

- When scenes undergo big changes in appearance due to a combination of aforementioned acquisition conditions (subset 4, subset 6, subset 7 and subset 8), the accuracy of the proposed approach is higher than that of FastABLE and FABMAP.

5.5.3 The proposed system computational complexity

The proposed system was implemented using c++ programming language and executed in a laptop with an Intel Core i5-5200U CPU @ 2.20 GHz and 4 GB RAM.
Our system consists of the following processing stages:

- Planar regions extraction from depth frames;
- Merging of planar regions belonging to the same plane into polygon;
- 3D map Construction;
- 2D map derivation;
- Scene description;
- place recognition.

The computational complexity of each step is indicated in table 5.3, the measurements were taken from the execution of the proposed algorithm on different scenes (with a total of 1000 images per scene).

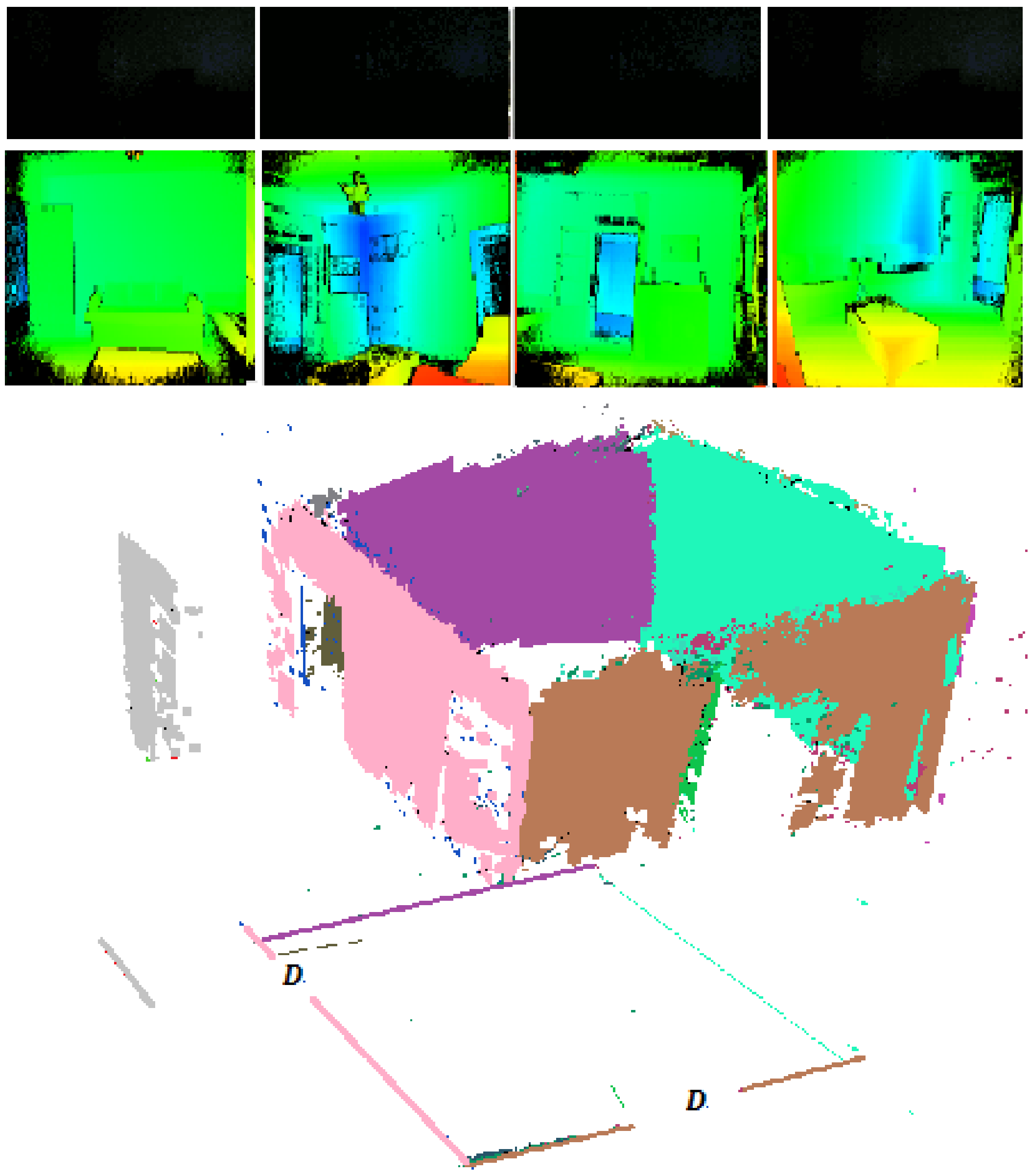

Figure 5.24: 3D and 2D map computed in a very low light scene.

Note that the multi-threading and GPUs were not used in the system implementation.

Despite a slight high of the system place recognition time in comparison to FastABLE and Fabmap, The proposed approach in this thesis is more suitable for applications where robustness to real scenarios (illumination variation, scenery modifications) is required.

	Execution time (s)		
	Max	Average	Min
Planar regions extraction	0.0107	0.0067	0.0028
Polygon construction	0.0042	0.0021	0.0001
3D map Construction	0.0309	0.0015	0.0004
2D map derivation	0.0080	0.0065	0.0050
Scene description	1.1222	1.3667	1.6112
Place recognition	0.0005	0.0004	0.0003
Total	1.1658	1.3772	1.6170

Table 5.3: Computational complexity of the proposed algorithm.

5.6 Conclusion

We have initiated this chapter by presenting the details and the used tools for the system's implementation. Then we have introduced the evaluation of the new propounded visual positioning method.

The conducted experiments demonstrate the scenes descriptors accuracy and the benefit of their use on place recognition. They also proclaim that our approach is uninfluenced by the scenery changes, the light variations and it outperforms the state of the art methods FastABLE [120] and Fast Appearance Based Mapping (Fabmap) [11] especially in the case of big changes in appearance due to a combination of acquisition conditions.

Among the most notable advantages of this system, we highlight too the no need of dataset update because it contains architecture based scene descriptors reflecting the non rearrangeable scene parts.
In conclusion, the proposed approach through this thesis is more suitable for applications where robustness to real scenarios is required.

6 Conclusion

We have proposed through this thesis a new visual positioning system based on two principle steps.

The first one whose objective is the scene description, takes as input a depth video acquired by rotating the kinect in the scene. Builds gradually a 3D model, locates the ground plane and identifies the stable features. The 3D model includes:
- the walls represented as the highest large polygons perpendicular to the polygon describing the ground,
- the doors and windows by openings belonging to the walls polygons,
- the stairs by parallel polygons regularly spaced and parallel to the ground.

Once these stable scene features are identified, they are represented on a 2D map, by line segments describing walls with labeled openings for doors, and windows, and rectangular shapes with parallel lines inside, for the stairs. Then this map is described geometrically to derive the scene descriptor.

The second step aims to recognize the unknown place by matching between the dataset descriptors and the resulting descriptor from the first stage.

A method for building the dataset was also proposed, to build the evaluation database containing descriptors of the already visited places.

The assessment of the system proclaims the accuracy of the proposed descriptor, and its robustness to the variation of the acquisition conditions and confirms that the used dataset does not require updates, since it is based on constant scene characteristics.

We therefore anticipate that our method will draw more attention from researchers to other architectural forms and features in the future.

6.0.1 Future works

As a confusion is strongly expected in the proposed approach, when different scenes present exactly a same architecture, we plan to study the possibility of this inconvenient elimination by the inclusion of the information relating to the non-movable objects in the scene descriptor, and to analyze their impact on the results of place recognition.

We plan also, to improve the proposed system by taking into consideration scenes with rounded walls.

we draw readers attention to the fact that the majority of visual positioning systems proposed in the state of the art use large databases, including for each scene several descriptors calculated under different conditions. However, the number of descriptors stored in the dataset used by our system is smaller.
And we aim to better propel the system by replacing the various descriptors of the dataset with a descriptor describing the entire interior environment derived directly from the architectural plan.

A appendix

A.1 3D to 2D transformation matrix

To determine the transformation matrix M which allows to express the polygons in the $2D$ coordinate system defined by a plan (normal vector $\vec{N}(x_N, y_N, z_N)$, and distance d) and an origin point $p_o(x_o, y_o, z_o)$, we must compute the corresponding translation and rotation.

- **Translation:**

  The translation between the two origins of the coordinate systems is only the vector defined by the origin of the original coordinate system and the center of gravity of the polygon that we want to transform.

- **Rotation:**

  The rotation is defined by an axis and an angle of rotation, the axis of rotation is the line carrying the vector V_{axe} which is the vector product between the axis ($\vec{Z_{axe}}$) of the reference of origin and the normal vector of the polygon($\vec{N}$):

$$\overrightarrow{V_{axe}} = \overrightarrow{Z_{axe}} \wedge \overrightarrow{N} \tag{A.1}$$

The rotation angle is calculated using the scalar product between the two vectors:

$$V_{angle} = \cos^{-1}(\overrightarrow{Z_{axe}}.\overrightarrow{N}) \tag{A.2}$$

The quaternion Q defining the rotation is given by $Q(Q_x, Q_y, Q_z, Q_w)$ Where:

$$\begin{cases} Q_x = \overrightarrow{V_{axe}}.x * \sin V_{angle}/2 \\ Q_y = \overrightarrow{V_{axe}}.y * \sin V_{angle}/2 \\ Q_z = \overrightarrow{V_{axe}}.z * \sin V_{angle}/2 \\ Q_w = \cos V_{angle}/2 \end{cases} \tag{A.3}$$

The quaternion Q can be written in the form of a matrix as follows:

$$\begin{pmatrix} 1 - 2Q_y^2 - 2Q_z^2 & 2Q_xQ_y - 2Q_zQ_w & 2Q_xQ_z + 2Q_yQ_w \\ 2Q_xQ_y + 2Q_zQ_w & 1 - 2Q_x^2 - 2Q_z^2 & 2Q_yQ_z - 2Q_xQ_w \\ 2Q_xQ_z - 2Q_yQ_w & 2Q_yQ_z + 2Q_xQ_w & 1 - 2Q_y^2 - 2Q_x^2 \end{pmatrix}$$

The transformation matrix M in homogeneous coordinates is written as follows:

$$\begin{pmatrix} & & & x_0 \\ & M_R & & y_0 \\ & & & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

To transform a point $p(x, y, z)$into the new coordinate system it is necessary to do a matrix multiplication with the homogeneous coordinates:

$$p_t = (x_t, y_t, 0, 1) = p(x, y, z, 1)^T.M \tag{A.4}$$

Finally the coordinates of point p in the new $2D$ coordinate system are $p_t(x_t, y_t)$.

A.2 Projection of a point onto a plan

The projection p' of a point p on a plane P is the intersection of the line pp' parallel to the normal vector $\vec{N}(x_N, y_N, z_N)$ of this plan and carrying the point p, with this same plan, we can compute the projected point p' with the formula:

$$\vec{Op'} = \vec{Op} - \vec{N}.(\vec{Op} - \vec{Oc}.\vec{N}) \tag{A.5}$$

where c is a point belonging to the plan P.

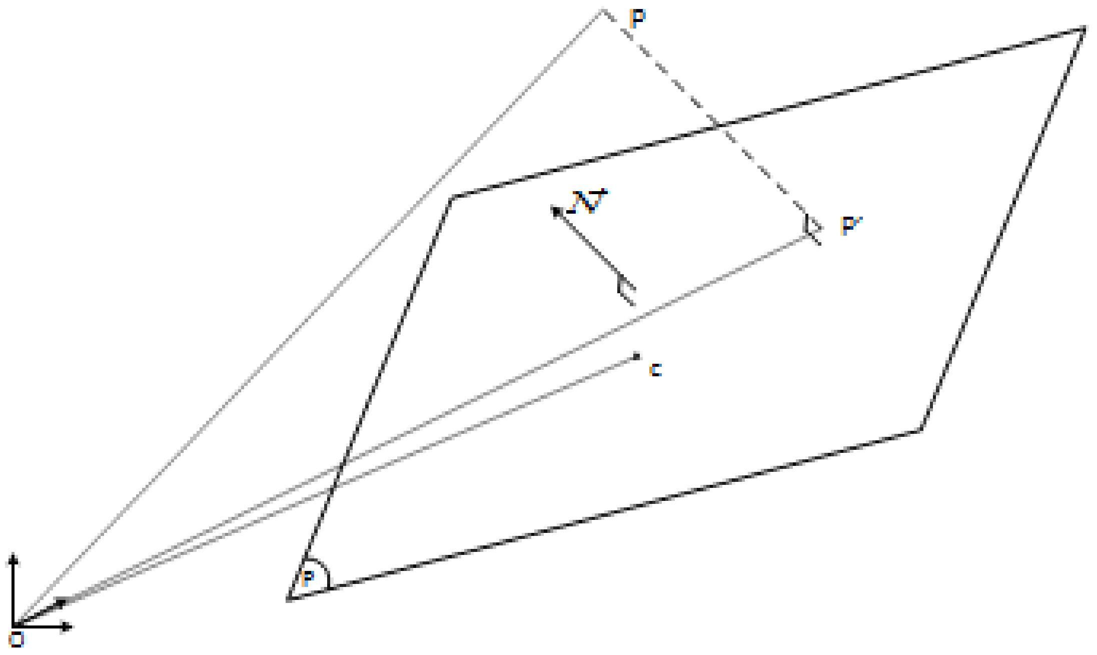

Figure A.1: projection of a point onto a plan.

B
Bibliography

Table of contents

Bibliography

[1] B. L. Diffey, “An overview analysis of the time people spend outdoors”, *British Journal of Dermatology*, vol. 164, no. 4, pp. 848–854, 2011.

[2] S. Lowry, N. Sünderhauf, P. Newman, *et al.*, “Visual place recognition: a survey”, *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[3] G. Kabanda, “Review of human computer interaction and computer vision”, GRIN Verlag Munich, Germany: 2019.

[4] M. Meng and A. C. Kak, “Neuro-nav: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment”, in *[Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 1993, pp. 750–757.

[5] A. Aladren, G. López-Nicolás, L. Puig, and J. J. Guerrero, “Navigation assistance for the visually impaired using rgb-d sensor with range expansion”, *IEEE Systems Journal*, vol. 10, no. 3, pp. 922–932, 2014.

[6] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, “Real-time 3d visual slam with a hand-held rgb-d camera”, in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, vol. 180, 2011, pp. 1–15.

[7] K. Konolige and M. Agrawal, “Frame-frame matching for realtime consistent visual mapping”, in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 2803–2810.

[8] J. Neira, A. J. Davison, and J. J. Leonard, “Guest editorial special issue on visual slam”, *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 929–931, 2008.

[9] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: large-scale direct monocular slam”, in *European conference on computer vision*, Springer, 2014, pp. 834–849.

[10] N. Abdelkrim, K. Issam, K. Lyes, and C. Khaoula, “Fuzzy logic controllers for mobile robot navigation in unknown environment using kinect sensor”, in *IWSSIP 2014 Proceedings*, IEEE, 2014, pp. 75–78.

[11] M. Cummins and P. Newman, “Appearance-only slam at large scale with fab-map 2.0”, *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[12] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, “Open-fabmap: an open source toolbox for appearance-based loop closure detection”, in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 4730–4735.

[13] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, M. Busso, and A. Rizzi, “Robust traffic lights detection on mobile devices for pedestrians with visual impairment”, *Computer Vision and Image Understanding*, vol. 148, pp. 123–135, 2016.

[14] L. Xie, A. Markham, and N. Trigoni, “Snapnav: learning mapless visual navigation with sparse directional guidance and visual reference”, in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 1682–1688.

[15] K. M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard, “Improving robot navigation in structured outdoor environments by identifying vegetation from laser data”, in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 1217–1222.

[16] S. Panzieri, F. Pascucci, and G. Ulivi, “An outdoor navigation system using gps and inertial platform”, *IEEE/ASME transactions on Mechatronics*, vol. 7, no. 2, pp. 134–142, 2002.

[17] Z. Bauer, A. Dominguez, E. Cruz, F. Gomez-Donoso, S. Orts-Escolano, and M. Cazorla, “Enhancing perception for the visually impaired with deep learning techniques and low-cost wearable sensors”, *Pattern recognition letters*, vol. 137, pp. 27–36, 2020.

[18] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra, “Vits-a vision system for autonomous land vehicle navigation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 342–361, 1988.

[19] J. Rivera-Rubio, K. Arulkumaran, H. Rishi, I. Alexiou, and A. A. Bharath, “An assistive haptic interface for appearance-based indoor navigation”, *Computer Vision and Image Understanding*, vol. 149, pp. 126–145, 2016.

[20] B. Zhou, J. Yi, X. Zhang, *et al.*, “An autonomous navigation approach for unmanned vehicle in outdoor unstructured terrain with dynamic and negative obstacles”, *Robotica*, pp. 1–24, 2022.

[21] J. Guivant, E. Nebot, J. Nieto, and F. Masson, “Navigation and mapping in large unstructured environments”, *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 449–472, 2004.

[22] E. Krotkov and M. Hebert, “Mapping and positioning for a prototype lunar rover”, in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 1995, pp. 2913–2919.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[24] R. Girshick, “Fast r-cnn”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[25] W. Liu, D. Anguelov, D. Erhan, *et al.*, “Ssd: single shot multibox detector”, in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: unified, real-time object detection”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[27] J. Redmon and A. Farhadi, “Yolov3: an incremental improvement”, *arXiv preprint arXiv:1804.02767*, 2018.

[28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: optimal speed and accuracy of object detection”, *arXiv preprint arXiv:2004.10934*, 2020.

[29] C. Li, L. Li, H. Jiang, *et al.*, “Yolov6: a single-stage object detection framework for industrial applications”, *arXiv preprint arXiv:2209.02976*, 2022.

[30] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, *arXiv preprint arXiv:2207.02696*, 2022.

[31] H.-K. Jung and G.-S. Choi, “Improved yolov5: efficient object detection using drone images under various conditions”, *Applied Sciences*, vol. 12, no. 14, p. 7255, 2022.

[32] A. Haro, I. Essa, and M. Flickner, “A non-invasive computer vision system for reliable eye tracking”, in *CHI’00 Extended Abstracts on Human Factors in Computing Systems*, 2000, pp. 167–168.

[33] M. Cormier, K. Moffatt, R. Cohen, and R. Mann, “Purely vision-based segmentation of web pages for assistive technology”, *Computer Vision and Image Understanding*, vol. 148, pp. 46–66, 2016.

[34] M. Kraft, M. Nowicki, A. Schmidt, M. Fularz, and P. Skrzypczyński, “Toward evaluation of visual navigation algorithms on rgb-d data from the first-and second-

generation kinect", *Machine Vision and Applications*, vol. 28, no. 1, pp. 61–74, 2017.

[35] R. A. Zeineldin and N. A. El-Fishawy, "Fast and accurate ground plane detection for the visually impaired from 3d organized point clouds", in *2016 SAI computing conference (SAI)*, IEEE, 2016, pp. 373–379.

[36] M. Bloesch and D. Rodriguez, "Kinect v2 for mobile robot navigation: evaluation and modeling", in *In Proceedings of the IEEE International Conference on Advanced Robotics (ICAR*, Citeseer, 2015.

[37] W. C. Simôes and V. De Lucena, "Blind user wearable audio assistance for indoor navigation based on visual markers and ultrasonic obstacle detection", in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2016, pp. 60–63.

[38] O. Halabi, M. Al-Ansari, Y. Halwani, F. Al-Mesaifri, and R. Al-Shaabi, "Navigation aid for blind people using depth information and augmented reality technology", *Proceedings of the NICOGRAPH International*, pp. 120–125, 2012.

[39] Y. H. Lee and G. Medioni, "Rgb-d camera based navigation for the visually impaired", in *Proceedings of the RSS*, Citeseer, vol. 2, 2011.

[40] C. Zatout and S. Larabi, "Semantic scene synthesis: application to assistive systems", *The Visual Computer*, vol. 38, no. 8, pp. 2691–2705, 2022.

[41] R. Abobeah, M. E. Hussein, M. M. Abdelwahab, and A. A. Shoukry, "Wearable rgb camera-based navigation system for the visually impaired.", in *VISIGRAPP (5: VISAPP)*, 2018, pp. 555–562.

[42] A. Bhowmick, S. Prakash, R. Bhagat, V. Prasad, and S. M. Hazarika, "Intellinavi: navigation for blind based on kinect and machine learning", in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, Springer, 2014, pp. 172–183.

[43] G. Bhorkar, "A survey of augmented reality navigation", *arXiv preprint arXiv:1708.05006*, 2017.

[44] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 883–890.

[45] K.-W. Chen, C.-H. Wang, X. Wei, *et al.*, "Vision-based positioning for internet-of-vehicles", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 364–376, 2017.

[46] R. Mur-Artal and J. D. Tardós, “Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras”, *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[47] J. Knopp, J. Sivic, and T. Pajdla, “Avoiding confusing features in place recognition”, in *European Conference on Computer Vision*, Springer, 2010, pp. 748–761.

[48] G. Feng, L. Ma, and X. Tan, “Visual map construction using rgb-d sensors for image-based localization in indoor environments”, *Journal of Sensors*, vol. 2017, 2017.

[49] E. Deretey, M. T. Ahmed, J. A. Marshall, and M. Greenspan, “Visual indoor positioning with a single camera using pnp”, in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2015, pp. 1–9.

[50] Y. Chen, R. Chen, M. Liu, A. Xiao, D. Wu, and S. Zhao, “Indoor visual positioning aided by cnn-based image retrieval: training-free, 3d modeling-free”, *Sensors*, vol. 18, no. 8, p. 2692, 2018.

[51] E. Sizikova, V. K. Singh, B. Georgescu, M. Halber, K. Ma, and T. Chen, “Enhancing place recognition using joint intensity-depth analysis and synthetic data”, in *European Conference on Computer Vision*, Springer, 2016, pp. 901–908.

[52] X. Song, S. Jiang, L. Herranz, and C. Chen, “Learning effective rgb-d representations for scene recognition”, *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 980–993, 2019.

[53] M. Koskela and J. Laaksonen, “Convolutional network features for scene recognition”, in *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 2014, pp. 1169–1172.

[54] H. Taira, M. Okutomi, T. Sattler, *et al.*, “Inloc: indoor visual localization with dense matching and view synthesis”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7199–7209.

[55] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network”, in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[56] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov, and I. Petrović, “Place recognition based on matching of planar surfaces and line segments”, *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 674–704, 2015.

[57] C. Zou, Z. Li, and D. Hoiem, “Complete 3d scene parsing from single rgbd image”, *CoRR*, vol. abs/1710.09490, 2017.

[58] D. G. Lowe *et al.*, “Object recognition from local scale-invariant features.”, in *iccv*, vol. 99, 1999, pp. 1150–1157.

[59] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: speeded up robust features”, in *European conference on computer vision*, Springer, 2006, pp. 404–417.

[60] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “Orb: an efficient alternative to sift or surf.”, in *ICCV*, Citeseer, vol. 11, 2011, p. 2.

[61] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, “An evaluation of the rgb-d slam system”, in *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 1691–1696.

[62] M. Bansal, M. Kumar, and M. Kumar, “2d object recognition: a comparative analysis of sift, surf and orb feature descriptors”, *Multimedia Tools and Applications*, vol. 80, no. 12, pp. 18 839–18 857, 2021.

[63] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, “Learning informative point classes for the acquisition of object model maps”, in *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Hanoi, Vietnam, 2008 2008.

[64] Y. Ye, T. Cieslewski, A. Loquercio, and D. Scaramuzza, “Place recognition in semi-dense maps: geometric and learning-based approaches”, in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 72–1.

[65] R. Hänsch, T. Weber, and O. Hellwich, “Comparison of 3d interest point detectors and descriptors for point cloud fusion”, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, p. 57, 2014.

[66] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration”, in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 3212–3217.

[67] Y. Zheng, P. Luo, S. Chen, J. Hao, and H. Cheng, “Visual search based indoor localization in low light via rgb-d camera”, *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 11, no. 3, pp. 349–352, 2017.

[68] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, “Fast 3d recognition and pose using the viewpoint feature histogram”, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 2155–2162.

[69] A. Aldoma, M. Vincze, N. Blodow, *et al.*, “Cad-model recognition and 6dof pose estimation using 3d cues”, in *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, IEEE, 2011, pp. 585–592.

[70] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3d object classification”, *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2987–2992, 2011.

[71] Y. Qiao and Z. Zhang, “Visual localization by place recognition based on multi-feature (d-λlbp”, *Journal of Sensors*, vol. 2017, 2017.

[72] H. Wen, R. Clark, S. Wang, *et al.*, “Efficient indoor positioning with visual experiences via lifelong learning”, *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 814–829, 2019.

[73] M. Uddin, “Scene classification using localized histogram of oriented gradients method”, *International Journal of Computer (IJC)*, vol. 20, no. 1, pp. 13–18, 2016.

[74] R. Sahdev and J. K. Tsotsos, “Indoor place recognition system for localization of mobile robots”, in *2016 13th Conference on Computer and Robot Vision (CRV)*, IEEE, 2016, pp. 53–60.

[75] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system”, *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[76] V. Lepetit, F. Moreno-Noguer, and P. Fua, “Epnp: an accurate o (n) solution to the pnp problem”, *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[77] T. Sattler, B. Leibe, and L. Kobbelt, “Efficient & effective prioritized matching for large-scale image-based localization”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1744–1756, 2017.

[78] Y. Li, N. Snavely, and D. P. Huttenlocher, “Location recognition using prioritized feature matching”, in *European conference on computer vision*, Springer, 2010, pp. 791–804.

[79] I. Ulrich and I. Nourbakhsh, “Appearance-based place recognition for topological localization”, in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Ieee, vol. 2, 2000, pp. 1023–1029.

[80] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[81] A. Oertel, T. Cieslewski, and D. Scaramuzza, “Augmenting visual place recognition with structural cues”, *arXiv preprint arXiv:2003.00278*, 2020.

[82] H. Fan, Y. Zhou, A. Li, S. Gao, J. Li, and Y. Guo, “Visual localization using semantic segmentation and depth prediction”, *arXiv preprint arXiv:2005.11922*, 2020.

[83] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, “From structure-from-motion point clouds to fast location recognition”, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 2599–2606.

[84] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: a convolutional network for real-time 6-dof camera relocalization”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[85] Z. Deng, S. Todorovic, and L. J. Latecki, “Unsupervised object region proposals for rgb-d indoor scenes”, *Computer Vision and Image Understanding*, vol. 154, pp. 127–136, 2017.

[86] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard, “Robot localization in floor plans using a room layout edge extraction network”, *arXiv preprint arXiv:1903.01804*, 2019.

[87] F. Ibelaiden, B. Sayah, and S. Larabi, “Scene description from depth images for visually positioning”, in *2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)*, IEEE, 2020, pp. 101–106.

[88] F. Ibelaiden and S. Larabi, “Visual place representation and recognition from depth images”, *Optik*, p. 169 109, 2022.

[89] C.-H. Chuang, Y.-N. Chen, and K.-C. Fan, “Image feature point matching for indoor positioning”, in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, The Steering Committee of The World Congress in Computer Science, Computer . . ., 2017, pp. 139–140.

[90] K. Pujar, S. Chickerur, and M. S. Patil, “Combining rgb and depth images for indoor scene classification using deep learning”, in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, IEEE, 2017, pp. 1–8.

[91] J. Shotton, A. Fitzgibbon, M. Cook, *et al.*, “Real-time human pose recognition in parts from single depth images”, in *CVPR 2011*, Ieee, 2011, pp. 1297–1304.

[92] C. Pheatt and J. McMullen, “Programming for the xbox kinect™ sensor: tutorial presentation”, *Journal of Computing Sciences in Colleges*, vol. 27, no. 5, pp. 140–141, 2012.

[93] Z. Xing and Z. Shi, “Extracting multiple planar surfaces effectively and efficiently based on 3d depth sensors”, *IEEE Access*, vol. 7, pp. 7326–7336, 2018.

[94] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”, *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[95] A. Nguyen and B. Le, “3d point cloud segmentation: a survey”, in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, IEEE, 2013, pp. 225–230.

[96] B. Xu, W. Jiang, J. Shan, J. Zhang, and L. Li, “Investigation on the weighted ransac approaches for building roof plane segmentation from lidar point clouds”, *Remote Sensing*, vol. 8, no. 1, p. 5, 2015.

[97] R. Schnabel, R. Wahl, and R. Klein, “Efficient ransac for point-cloud shape detection”, in *Computer graphics forum*, Wiley Online Library, vol. 26, 2007, pp. 214–226.

[98] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, “Fast plane detection and polygonalization in noisy 3d range images”, in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 3378–3383.

[99] X. Qian and C. Ye, “Ncc-ransac: a fast plane extraction method for 3-d range data segmentation”, *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2771–2783, 2014.

[100] P. V. Hough, “Method and means for recognizing complex patterns”, *US patent*, vol. 3, no. 6, 1962.

[101] E. Vera, D. Lucio, L. A. Fernandes, and L. Velho, “Hough transform for real-time plane detection in depth images”, *Pattern Recognition Letters*, vol. 103, pp. 8–15, 2018.

[102] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter, “The 3d hough transform for plane detection in point clouds: a review and a new accumulator design”, *3D Research*, vol. 2, no. 2, pp. 1–13, 2011.

[103] P. Cheng, W.-J. Li, W.-L. Chen, D.-L. Gao, Y. Xu, and H. Li, “Computer vision-based recognition of rainwater rivulet morphology evolution during rain–wind-induced vibration of a 3d aeroelastic stay cable”, *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 172, pp. 367–378, 2018.

[104] `www.3dreshaper.com`. (visited on 05/05/2023).

[105] R. A. Newcombe, S. Izadi, O. Hilliges, *et al.*, “Kinectfusion: real-time dense surface mapping and tracking”, in *2011 10th IEEE international symposium on mixed and augmented reality*, IEEE, 2011, pp. 127–136.

[106] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, “Real-time 3d reconstruction at scale using voxel hashing”, *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, pp. 1–11, 2013.

[107] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images”, *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[108] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, “Ransac-based darces: a new approach to fast automatic registration of partially overlapping range images”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1229–1234, 1999.

[109] Y. Chen and G. Mediom, “Object modeling by registration ofmultiple range images, in’image and vision computing’”, 1991.

[110] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, “Registration with the point cloud library: a modular framework for aligning in 3-d”, *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[111] J. L. Bentley, “Multidimensional binary search trees used for associative searching”, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[112] `https://algorithmtutor.com/Data-Structures/Tree/Red-Black-Trees/`.

[113] C. E. Leiserson, R. L. Rivest, T. H. Cormen, and C. Stein, *Introduction to algorithms*. MIT press, 1994, vol. 3.

[114] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, “An image-to-map loop closing method for monocular slam”, in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 2053–2059.

[115] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, “A comparison of loop closing techniques in monocular slam”, *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[116] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, “Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images”, in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 2684–2689.

[117] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, “Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images”, in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 2684–2689.

[118] W. Chojnacki, M. J. Brooks, A. Van Den Hengel, and D. Gawley, “On the fitting of surfaces to data with covariances”, *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1294–1303, 2000.

[119] R. Hinze *et al.*, “Constructing red-black trees”, in *Proceedings of the Workshop on Algorithmic Aspects of Advanced Programming Languages, WAAAPL*, vol. 99, 1999, pp. 89–99.

[120] M. R. Nowicki, J. Wietrzykowski, and P. Skrzypczyński, “Real-time visual place recognition for personal localization on a mobile device”, *Wireless Personal Communications*, vol. 97, no. 1, pp. 213–244, 2017.

[121] F. C. Crow, “Summed-area tables for texture mapping”, in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984, pp. 207–212.

[122] D. Kalman, “A singularly valuable decomposition: the svd of a matrix”, *The college mathematics journal*, vol. 27, no. 1, pp. 2–23, 1996.

[123] R. B. Rusu and S. Cousins, “3d is here: point cloud library (pcl)”, in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 1–4.

[124] B. R. Vatti, “A generic solution to polygon clipping”, *Communications of the ACM*, vol. 35, no. 7, pp. 56–63, 1992.

[125] D. M. Chen, G. Baatz, K. Köser, *et al.*, “City-scale landmark identification on mobile devices”, in *CVPR 2011*, IEEE, 2011, pp. 737–744.

[126] H. Jegou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search”, in *European conference on computer vision*, Springer, 2008, pp. 304–317.

[127] F. Ibelaiden and S. Larabi, “A benchmark for visual positioning from depth images”, in *2020 4th International Symposium on Informatics and its Applications (ISIA)*, IEEE, 2020, pp. 1–6.