DEPARTMENT OF SYSTEMS ENGINEERING AND AUTOMATION

Universidad Carlos III de Madrid

Reconstruction and Recognition of Confusable Models using Three-Dimensional Perception

Jorge García Bueno

PhD Thesis

Leganés, 2013

PhD Thesis

Reconstruction and Recognition of Confusable Models using Three-Dimensional Perception

Candidate

Jorge García Bueno

Adviser

Univ.-Prof. Luis Moreno Lorente

Review Committee

⋆ Chair (Presidente)	
⋆ Vocal (Vocal)	
⋆ Secretary (Secretario)	
Substitute (Suplente)	

Grade(Calificación)

Doctorado en Ingeniería Eléctrica, Electrónica y Automática

Leganés, December 2013

This thesis was funded by the Education Ministry of Spain.
FPU grant AP2009-3445
Robotics Lab – Universidad Carlos III de Madrid

Abstract

Perception is one of the key topics in robotics research. It is about the processing of external sensor data and its interpretation. The necessity of fully autonomous robots makes it crucial to help them to perform tasks more reliably, flexibly, and efficiently. As these platforms obtain more refined manipulation capabilities, they also require expressive and comprehensive environment models: for manipulation and affordance purposes, their models have to involve each one of the objects present in the world, coincidentally with their location, pose, shape and other aspects.

The aim of this dissertation is to provide a solution to several of these challenges that arise when meeting the object grasping problem, with the aim of improving the autonomy of the mobile manipulator robot MANFRED-2. By the analysis and interpretation of 3D perception, this thesis covers in the first place the localization of supporting planes in the scenario. As the environment will contain many other things apart from the planar surface, the problem within cluttered scenarios has been solved by means of Differential Evolution, which is a particle-based evolutionary algorithm that evolves in time to the solution that yields the cost function lowest value.

Since the final purpose of this thesis is to provide with valuable information for grasping applications, a complete model reconstructor has been developed. The proposed method holds many features such as robustness against abrupt rotations, multi-dimensional optimization, feature extensibility, compatible with other scan matching techniques, management of uncertain information and an initialization process to reduce convergence timings. It has been designed using a evolutionary-based scan matching optimizer that takes into account surface features of the object, global form and also texture and color information.

The last tackled challenge regards the recognition problem. In order to procure with worthy information about the environment to the robot, a meta classifier that discerns efficiently the observed objects has been implemented. It is capable of distinguishing between confusable objects, such as mugs or dishes with similar shapes but different size or color.

The contributions presented in this thesis have been fully implemented and empirically evaluated in the platform. A continuous grasping pipeline covering from perception to grasp planning including visual object recognition for confusable objects has been developed. For that purpose, an indoor environment with several objects on a table is presented in the nearby of the robot. Items are recognized from a database and, if one is chosen, the robot will calculate how to grasp it taking into account the kinematic restrictions associated to the anthropomorphic hand and the 3D model for this particular object.

Resumen

La percepción es uno de los temas más relevantes en el mundo de la investigación en robótica. Su objetivo es procesar e interpretar los datos recibidos por un sensor externo. La gran necesidad de desarrollar robots autónomos hace imprescindible proporcionar soluciones que les permita realizar tareas más precisas, flexibles y eficientes. Dado que estas plataformas cada día adquieren mejores capacidades para manipular objetos, también necesitarán modelos expresivos y comprensivos: para realizar tareas de manipulación y prensión, sus modelos han de tener en cuenta cada uno de los objetos presentes en su entorno, junto con su localización, orientación, forma y otros aspectos.

El objeto de la presente tesis doctoral es proponer soluciones a varios de los retos que surgen al enfrentarse al problema del agarre, con el propósito final de aumentar la capacidad de autonomía del robot manipulador MANFRED-2. Mediante el análisis e interpretación de la percepción tridimensional, esta tesis cubre en primer lugar la localización de planos de soporte en sus alrededores. Dado que el entorno contendrá muchos otros elementos a parte de la superficie de apoyo buscada, el problema en entornos abarrotados ha sido solucionado mediante Evolución Diferencial, que es un algoritmo evolutivo basado en partículas que evoluciona temporalmente a la solución que contempla el menor resultado en la función de coste.

Puesto que el propósito final de este trabajo de investigación es proveer de información valiosa a las aplicaciones de prensión, se ha desarrollado un reconstructor de modelos completos. El método propuesto posee diferentes características como robustez a giros abruptos, optimización multidimensional, extensión a otras características, compatibilidad con otras técnicas de reconstrucción, manejo de incertidumbres y un proceso de inicialización para reducir el tiempo de convergencia. Ha sido diseñado usando un registro optimizado mediante técnicas evolutivas que tienen en cuenta las particularidades de la superficie del objeto, su forma global y la información relativa a la textura.

El último problema abordado está relacionado con el reconocimiento de objetos. Con la intención de abastecer al robot con la mayor información posible sobre el entorno, se ha implementado un meta clasificador que diferencia de manera eficaz los objetos observados. Ha sido capacitado para distinguir objetos confundibles como tazas o platos con formas similares pero con diferentes colores o tamaños.

Las contribuciones presentes en esta tesis han sido completamente implementadas y probadas de manera empírica en la plataforma. Se ha desarrollado un sistema que cubre el problema de agarre desde la percepción al cálculo de la trayectoria incluyendo el sistema de reconocimiento de objetos confundibles. Para ello, se ha presentado una mesa con objetos en un entorno cerrado cercano al robot. Los elementos son comparados con una base de datos y si se desea agarrar uno de ellos, el robot estimará como cogerlo teniendo en cuenta las restricciones cinemáticas asociadas a una mano antropomórfica y el modelo tridimensional generado del objeto en cuestión.

A mis padres Rosalía y Jorge,
a mi hermano Luis,
a mi abuelo Luis.

Amat Victoria Curam
La victoria favorece a los que se preparan

Agradecimientos

No ha sido fácil. Nadie dijo que lo fuera, y sin embargo llegados a este punto no puedo sino agradecer el esfuerzo a toda la gente que me ha impulsado este tiempo.

Luis Moreno, gran profesor y mejor persona. Has conseguido formarme como pocos, abrirme caminos más allá de los libros y sobretodo apoyarme en cada una de mis decisiones. No tengo palabras para agradecerte este apoyo incondicional.

Mi familia, mis padres, mi hermano. Vuestro interés, insistencia, paciencia y sabiduría me han permitido llegar hasta aquí. Se que no ha sido fácil tampoco para vosotros, han sido muchas cenas escuchando mis batallas. Os quiero muchísimo.

A Alex, mi socio, compañero y ante todo un amigo único. Larga es la lista de batallas y experiencias que hemos vivido juntos los últimos años que no han hecho más que reforzar una amistad. Solamente puedo decirte una cosa: eres un genio. Muchísimas gracias a mis compañeros del Robotics Lab: Nico Burrus, Fer Monar, Concha, Santiago, Dolores y por supuesto a Miguel Fierro. La lista de historias contigo es interminable. Gracias a Víctor , David, Fran, Antonio y Alejandro por vuestras aportaciones en mis artículos.

Sin embargo, hay mucha más gente detrás que también me han aportado su granito de arena. La panda de toda la vida: Santi, Pablo, Kiko, Elena, Leyre, Raúl, Miguel, Javivi, Clemen, Mota, Fuentes, Marco. Y los amigos de la Universidad, en especial cuatro; Azu, Nacho, Chema y Diego Fogeda. A vosotros os debo tanto o más que a los demás por el cariño y el interés que me habéis demostrado día a día.

Finalmente, y no por ello menos importante, quiero agradecer el apoyo a Isabel. Con tu positivismo y alegría consigues cada día hacerme un poquito más feliz :)

¡gracias a todos!

Abbreviations

2D	**T**wo **d**imensional
3D	**T**hree **d**imensional
6D	**S**ix **d**imensional
CS	**C**onsensus **S**et
CWM	**C**ontinuous **W**ave **M**odulation
DE	**D**ifferential **E**volution
DOF	**D**egrees **o**f **F**reedom
EA	**E**volutionary **A**lgorithms
ECE	**E**uclidean **C**luster **E**xtraction
FLANN	**F**ast **A**pprox. **N**earest **N**eighbors w. **A**utomatic **A**lgorithm **C**onf.
FOV	**F**ield **O**f **V**ision
GA	**G**enetic **A**lgorithm
ICP	**I**terative **C**losest **P**oints
KNN	k **N**earest **N**eighbours
LSM	**L**east **S**quares **M**ethod
MSS	**M**inimal **S**ample **S**et
NARVP	**N**ormal **A**ligned **R**ange **V**alue **P**atch
NN	**N**earest **N**eighbours
R&D	**R**esearch **a**nd **D**evelopment
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
SV	**S**tereo **V**ision
SURF	**S**peeded **U**p **R**obust **F**eature
ToF	**T**ime **o**f **F**light
TSDF	**T**ransform **S**igned **D**istance **F**unction
VFH	**V**iewpoint **F**eature **H**istograms

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Generation by generation, human beings have claimed innovative technologies in order to face new demanding challenges. Prolongation of productive life expectancy and the irruption of new communications such as Internet has converted actual communities in a perfect breeding ground for independent living societies. With this problem in mind, developed countries are putting intense efforts into personal robots capable of interacting with people for their company. This challenge contemplates, among other assignments, the assistance in case of emergencies or cooperation during household chores such as laundry and dishwashing.

In order to fulfill these necessities, travail has been focused on new assistive robots becoming more intelligent as they are provided with improved perception skills for daily activities performances. These new generations of personal robots are prepared to interact with humans, learn new abilities, exchange previous experiences or take their own decisions according to their knowledge. The instinctive way of this interaction to happen passes through the creation of new perception channels, providing robots with relevant information of their surroundings. These technological advantages lead to convert them into smarter platforms.

The robots must be prepared to acquire new sources of information, detect patterns according to received data, react to these stimuli corresponding to learned behaviors by motion planning and finally perform actions based on the estimations. Manipulation skill is, without any doubt, one of the desirable abilities that covers all the previous requests. They have to synthesize the sensor data within the context of the actions and activities they perform. If the robot has to grasp a mug laying on a tray, it has to find a handle in a first stage and then plan a safe path to reach the object.

Those tasks are, unfortunately, rough to accomplish. Each attempt of the robot becomes different from the last in every aspect. The perception sensor does not provide exactly the same data values and therefore the subsequent steps of the chain might vary. However, by means of machine learning tools, the robots are able to innovate so that unexpected changes can be assumed successfully. The first usages of machine learning were applied by Elfes [1] in 1989 for environment mapping of an autonomous robot. After that, robots were programmed not only to perform a *single* task but to carry out a comprehensive *family* of tasks.

But the challenge is wider than that. Autonomous robots are not asked for manipulating objects solely, they must understand that if a mug is taken, it has to be grasped by its handle or that the sharp edge of a knife has to be dodged, turning this into one of the main issues of semantic perception for mobile manipulation.

This dissertation thesis proposes a set of improvements in several fundamental aspects for human-robot interaction. The aim of this research is to provide the robot with the ability of recognizing objects lying on a table that is situated on its vicinity. Furthermore, in case the object is not identified, the robot must be able to acquire sufficient views of the object so it would be distinguished in any upcoming occasion. Since the final purpose of the robot is to grasp a certain object, it has to comprehend the different views of the object and recreate a global model of the item. The challenge goes even further as the global model has to be sufficiently accurate and robust to permit others to analyze it in order to find optimal grasping points. Additionally to the above, the robot will be prepared to discriminate between confusable objects that might appear in the scene. In other words, the robot has to classify families of objects that could be easily confused due to a similar shape, size or color. That means that if two identical glasses of wine are disposed in front of the robot, it has to be prepared for classifying them favorably.

Thesis Outline and Contributions

The contributions of this thesis moves in harmony with the skills a robot must evolve in order to interact and grasp objects such as humans do in a natural way. Divided into three main parts, it is intended to reach the whole pipeline that the problem demands. Figure 1 illustrates the different parts of the thesis validating the continuity of the research. This dissertation deals with several facets of perception that have been separated and presented in a logical and natural way so that the reader can comprehend the challenges affronted in each part, the way those difficulties where solved and ergo in what manner they are beneficial in the successive robot tasks.

Each part has been splitted in different chapters within a common structure. The heading chapters on each part introduce the problem being faced while presenting the state-of-the-art and solutions to that specific problem. The latter chapters expose the author contributions and any experimental results necessary to support the solution.

The thesis starts with an extensive state of the art of the 3D perception techniques that exist up to these days. This section includes a novel proposal to integrate color information within spatial data on Time of Flight cameras and also an improved plane fitting algorithm that abides cluttered scenarios where the quantity of objects resting on a table would become an issue for the plane computation. The usage of a multi-modal optimizer provides with more stable results than classical techniques for estimating the parameters of a model.

The second part of the thesis addresses the three-dimensional model registration of unexplored objects based on their features. A wide study of the actual techniques for object reconstruction is exposed and a novel reconstruction technique is presented. The proposed system is an evolutionary-based scan matching method that reassembles objects with a limited quantity of views within uncontrolled acquisition processes and assumed to be performed without human intervention. The latter includes a hole filling filtering algorithm that introduces a probabilistic model that categorizes the certainty of observed and guessed points through a mixed model.

Third part of this dissertation tackles the capacity of the robot to classify objects segmented during the preceding assignments. Thereby a meta-classifier is presented designed to let the robot discriminate through a list of regular objects. These objects were previously exposed to the robot in a table. The classifier has been conceived to correlate objects by extracting shape features, geometrical characteristics and also color and texture patterns. Several distance metrics have been tested in order to determine the reliability and performance of the classifier with confusable objects.

The latter part of the thesis presents the primary tests of all the contributions in order to validate the perception aptitudes of the mobile manipulation robot MANFRED–2. Besides, it integrates a safe path planning of the arm in order to reach the barycenter of a certain object. Experimental results yield good robustness in the supporting plane fitting for noisy environments and high recognition rates during the classification stage.

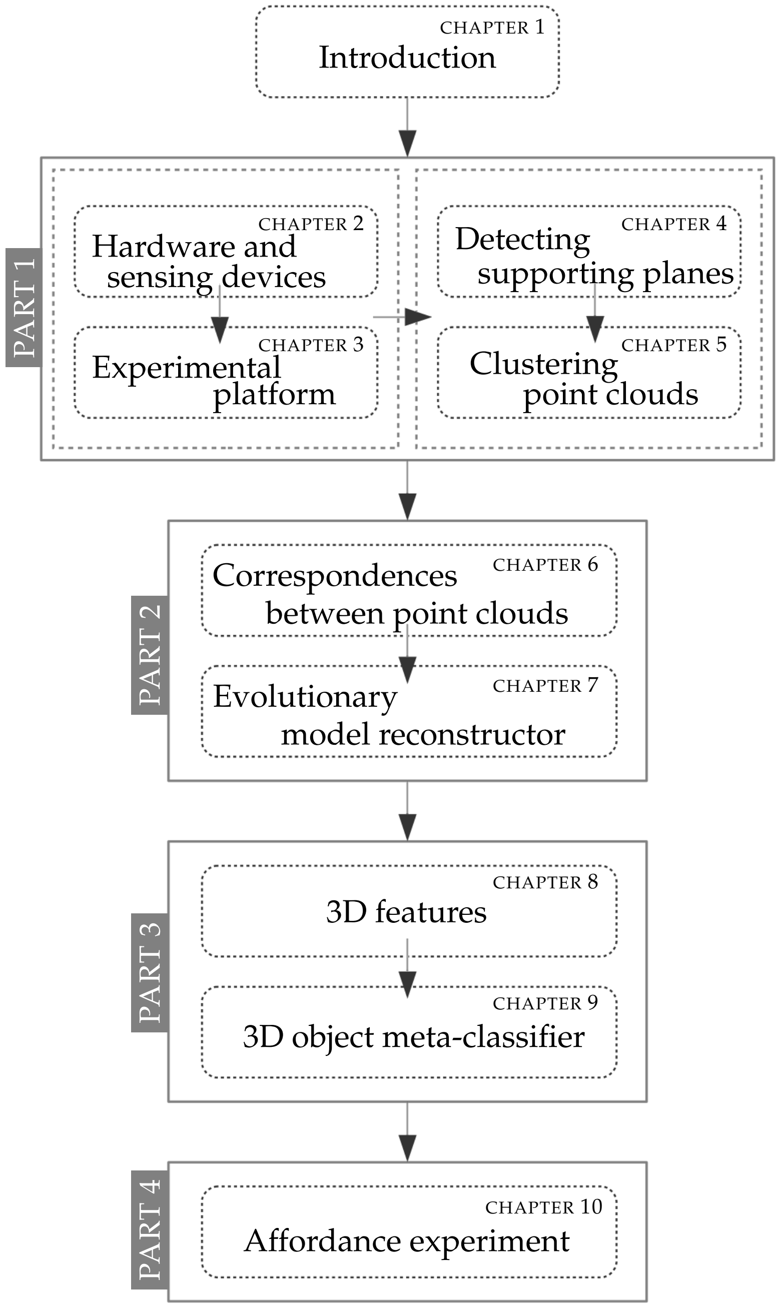


Figure 1.1: *Outline of the thesis.* General view of the thesis, conformation of the four parts and the chapters contained in each section.

PART I

CLUSTERING POINT CLOUDS

Chapter 2

Hardware and sensing devices

Robots require from sensors to interpret their environment. Perception technology has changed with the time allowing researchers to improve and investigate further on algorithms that enhance the way artificial machines interpret and react to external responses. Just because humans are living in a three–dimensional world, they are provided with an adequate set of tools to describe and locate different objects in any surrounding scene. These given features include motion, relative position, size and evolution of perceived objects. The demand of spatial perception has been satisfied by nature providing animals and humans with at least two eyes [2]. This stereo vision ability allows humans to process an image flow inside the brain and compute precisely depth measures of the observed environment.

Before *Time-Of-Flight* (ToF) technology was introduced, there were several methods to acquire and estimate 3D point clouds: those classical stereo vision algorithms which are based on correspondence matching such as dense depth maps generated with Dynamic Programming proposed in Gong *et al.* [3], block-matching approach [4] or even improved methods based on various consecutive frames to enhance the results as Davis *et al.* [5]. Besides, instead of passive systems like stereo vision, active sensors came up removing problems such as illumination conditions, unfocused scenes or image artifacts. This alternative is essentially a laser scanning line that deliver a specific signal and measure the received answer. These methods have been widely implemented in location problems, SLAM, environment modeling, surgery or industrial applications.

2.1 Stereo Vision

Stereo Vision systems comprise two perspective cameras with limited *Field Of Vision* (FOV). Any physical point is found in the observed 3D-space using both cameras. For each pixel in one image, the appropriate location in the other view must be found. Assuming that both cameras are perfectly calibrated, undistorted and rectified, image planes for both images are coplanar if optical axis are exactly parallel. In that case, both cameras would have equal focal lengths $f_l = f_r$ and also equal *principal points* [1] ${c_x}^l = {c_x}^r$, as Figure 2.1 describes.

In order to do that, the pinhole model can be easily imposed in both cameras. Equation 2.1 yields the corresponding relation between depth and disparity on pixels location using the triangulation principle.

[1]A principal point is where the principal ray intersects the camera plane. This intersection depends just on the optical axis of the lens.

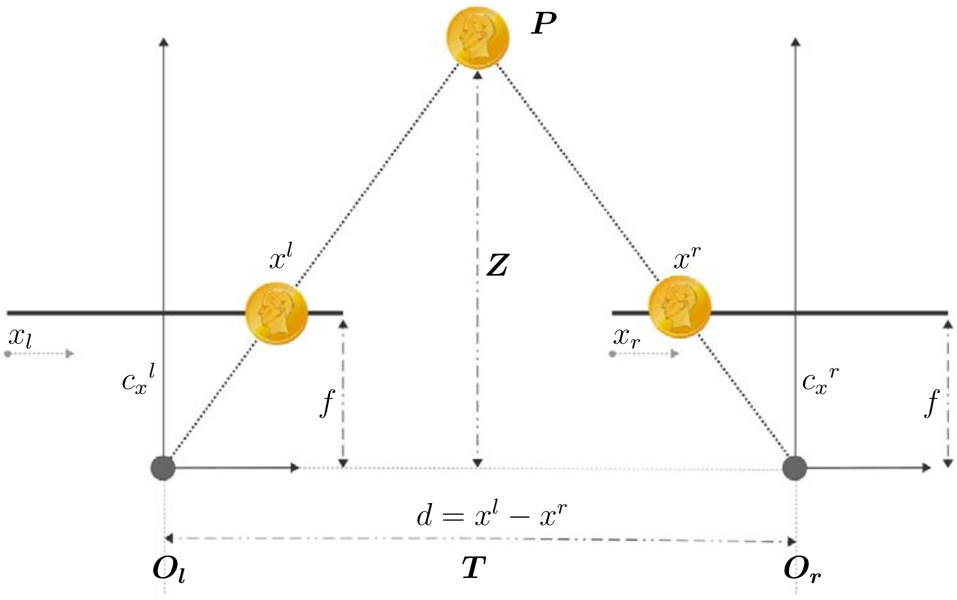


Figure 2.1: *Stereo Vision.* Perfect model of a stereo-pair for depth acquisition [6]

$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \rightarrow Z = \frac{fT}{x^l - x^r} \tag{2.1}$$

where $x^l - x^r$ is defined as *disparity*. For this action, the most obvious drawback found is the *correspondence problem*, that is, to match the *pixel-wise* pairs in both images. This operation requires a large consumption process in terms of computing resources and time to achieve good results due to the fact that pixels are not easy to find. Several applications have been developed during the last years in order to decrease computing times using new concepts such as parallel computing using CUDA proposed by Martín in [7], dynamic programming or Parallel cells such as Liu *et al.* [8].

2.2 Time-Of-Flight Technology

Once again, nature has beaten humans when talking about intelligence. Several thousands of years have passed since bats or dolphins were able to see without proper eyes. Those species use this sensor for both navigating and object tracking. This feature makes them able to detect and locate external actions or events in order to escape or attack. Humans have applied time-of-flight measurement systems later on, for instance when measuring the unknown depth of a well listening to the returned echo after a stone was thrown. These ToF methods are based on the propagation time of sound instead of light. It was in the 17^{th} century when Galileo Galilei performed an experiment to estimate the speed of light [9]. To do that, he took two people handling a torch and placed them at the top of two mountains one kilometer far. If one of them turned it on, the other would do the same and *viceversa*. With this simple experiment, he tried to measure the light speed neglecting the time of reaction of the contributors.

There are two mainly approaches currently employed in ToF technology [10]. The first one uses modulated, incoherent light and it is based on a phase measurement that is possible to be implemented in standard CMOS or CCD technology. The second solution is based on an optical shutter technology having been first used in studio cameras and later on miniaturized cameras. In Figure 2.2 the most notable ToF cameras and their respective manufacturers are represented.

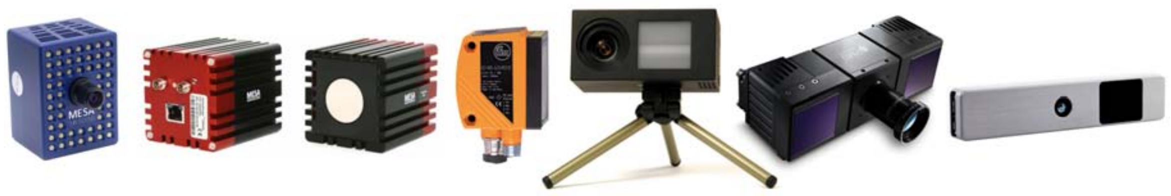


Figure 2.2: *Comercial ToF cameras.* (a-c) SR3000 and SR4000 from Mesa Imaging AG ©, (d) O3D100 from IFM Electronic ©, (e) CanestaVision ®, (f) PMD[Vision] ©CamCube 2.0, (g) PMD[Vision] ©CamBoard nano.

The basic principle of a ToF camera is represented in Figure 2.3 and it is resumed by Lange *et al.* [11] as follows: *A source emits a light pulse and starts a highly accurate stopwatch. The light pulse travels to the target and goes back. Reception of the light pulse by the detector mechanism stops the stopwatch, which now shows the time of flight of the light pulse.* Considering the fact that the light pulse travels the distance twice (forth and back) and that the speed of light is 299.792.458 m/s, then a measured time of 6.67 ns corresponds to a distance of one meter. As it is logical, the hardest problem here is to create a highly accurate time measurement system able to deal with nano and pico seconds. For instance, a resolution of 1 cm requires a time interval of 70 pico seconds.

Since ToF cameras are highly compact devices, the active light source and receiver are located very closely avoiding shadowing effects. That is, illumination

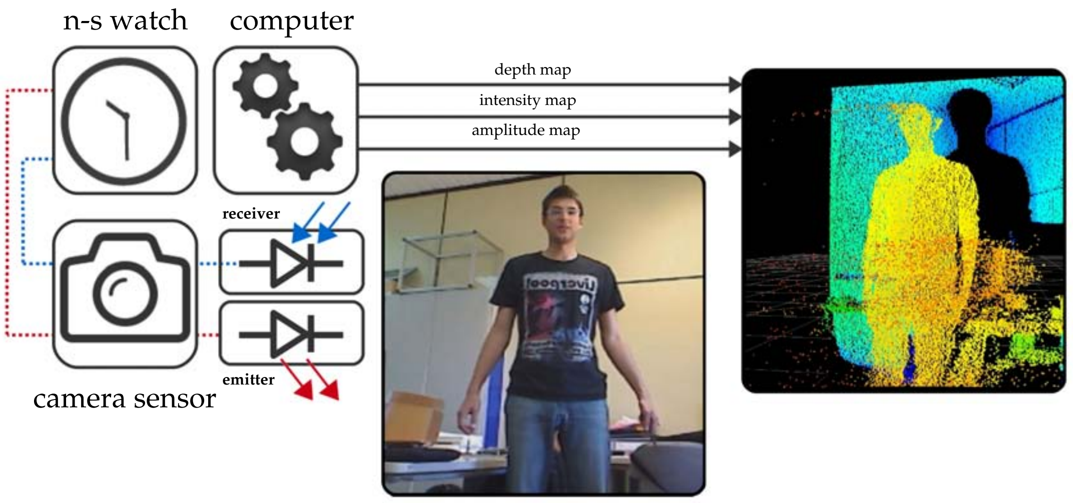


Figure 2.3: *ToF technology explanation.* Schema of the connections between the scene and the depth map. A nano-temporizer is activated when the led emitter (in red) pushes an infrared signal. Blue led receives the returned signal and stops the timer. The time-delay represents the distance to the object.

and observation directions are collinear [12]. Furthermore, one of the most important conditions of ToF sensors in general is that emitter and detector are operated synchronously, extracting the time of flight as accurate as possible.

ToF cameras are able to return three different sources of information. Firstly, a range map with a resolution of 204×204 pixels `float` precision in centimeters. This information it is obviously used to acquire object distances, scene segmentation and object modeling (Figure 2.4.b). Secondly, an intensity image that reveals the texture and brightness for each item inside the scene. That information becomes crucial for pattern recognition and camera calibration (Figure 2.4.c). Finally, an amplitude image that contains an estimation of the committed error measuring the time of flight for every pixel (Figure 2.4.d). Exploiting this information may increase the reliability of distance for each pixel, as it is considered with the discussion of the shading constraint by Bohme *et al.* in [13]. In Figure 2.4.a the original scenario to obtain a better understanding is displayed .

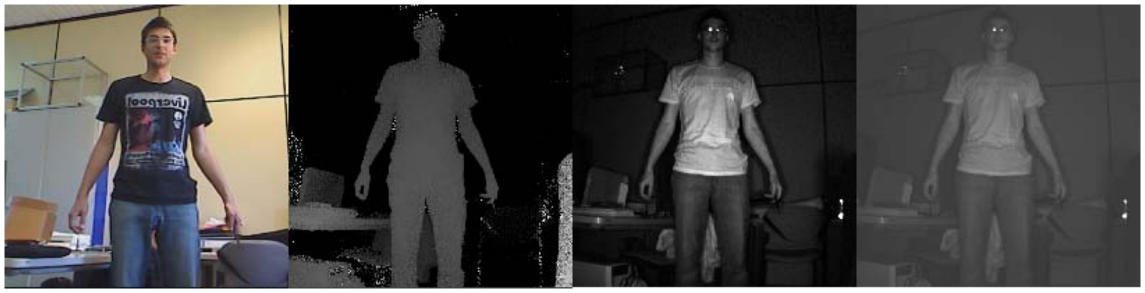


Figure 2.4: *ToF outputs.* Original RGB scenario. Depth map, intensity image and amplitude map are the three sources of information given by a ToF sensor (PMD Camera)

2.2.1 Intensity Modulation Principle

The most important companies have focused out their prototypes following this kind of ToF-principle such as **Mesa Imaging** [2] (Figure 2.2.a), **PMDTech electronics** [3] (Figure 2.2.b) and **CanestaVision Camera Modules** [4](Figure 2.2.c). The intensity modulation principle is based on the on-chip correlation of the incident optical signal s, which comes from a modulated *near infra-red* (NIR) source and is reflected by the objects inside the scenario, with a reference signal g, which posses an internal offset τ:

$$c(\tau) = s \otimes g = \lim_{T\to\infty} \int_{-T/2}^{+T/2} s(t) \cdot g(t+\tau) dt \tag{2.2}$$

Choosing s and g as sinusoidal signals:

$$g(t) = \cos(\omega \cdot t), \qquad s(t) = b + a \cdot \cos(\omega \cdot t + \phi) \tag{2.3}$$

where ω represents the modulation frequency, a is the amplitude of the incident optical signal, b corresponds to the correlation bias and ϕ is the phase offset due to the incident object distance. The convolution of those signals yields

$$c(\tau) = \frac{A}{2} \cos(\omega \cdot \tau + \phi) + b \tag{2.4}$$

Every pixel of the sensor samples the amount of modulated light reflected by any object four times every period at equal intervals m_1 to m_4. These four values are

[2]Mesa Imaging – http://www.mesa-imaging.ch
[3]PMDTech electronics – http://www.pmdtec.com
[4]Canesta Vision – http://canesta.com/

sufficient to recover the sinusoidal signal easily. The phase offset between the emitted light and the received signal is

$$\phi = \arctan(\frac{m_4 - m_2}{m_3 - m_1}), \qquad m_i = c\,(i, \frac{\pi}{2}), i = 1, \ldots, 4. \tag{2.5}$$

and this value determines the range of the object in the scene which is given by

$$R = \frac{c}{4\pi\omega} \cdot \phi, \ c \approx 299.792.458 m/s \tag{2.6}$$

The intensity of the objects in the image can be recovered from the average light reflected as

$$I = \frac{m_1 + m_2 + m_3 + m_4}{4} \tag{2.7}$$

The amplitude of the measured sinusoidal can be expressed as

$$A = \frac{\sqrt{(m_3 - m_1)^2 + (m_4 + m_2)^2}}{2} \tag{2.8}$$

and therefore it allows the prediction of the quality of the measurement ΔR as

$$\Delta R = \frac{c}{2\omega\sqrt{8}} \frac{\sqrt{I}}{2A} \tag{2.9}$$

With all this information, it is possible to obtain in real time not only depth values for each pixel but also the reliability or estimated error for each pixel.

2.2.2 3D Related Problems

Common arising problems

There exist several problems when processing images of captured objects. *Aperture problem* leads the list exhibiting the limited aperture angle of the camera optics giving information of a partial environment. *FOV problem* arises when fixing the field to a specific application. 3D information gets lost when projecting information into a planar sensor, that is, the *optical projection problem*. Finally, the detection of fast object movements cannot be sampled due to the sampling rate limit given by the camera, what is known as *sampling problem*. In addition to all these problems, the error committed measuring gray level changes in pixels (*divergence problem*) has to be considered.

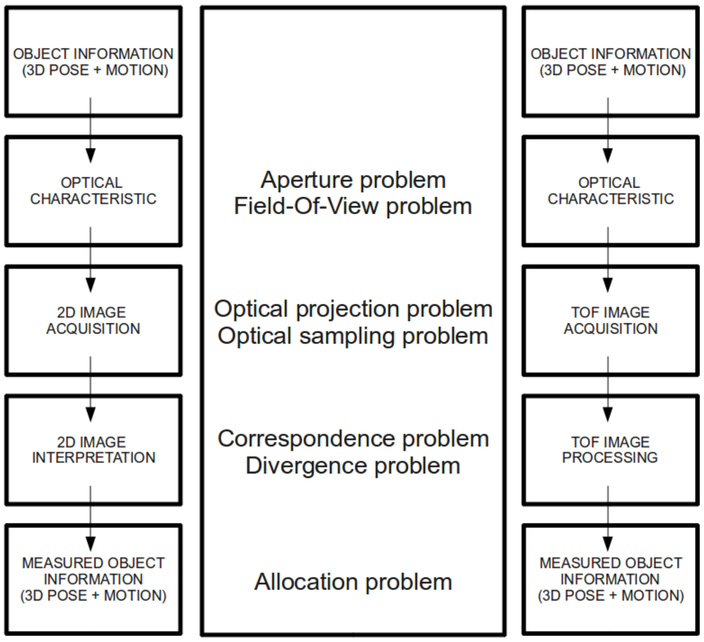


Figure 2.5: *Functional block.* Image processing chain comparison between SV and ToF systems by Hussmann *et al.* [14].

ToF cameras do not have all the problems shown in Figure 2.5. Only *sampling* and *aperture* problems appear. Other issues such as FOV problem do not affect just because hardware setup can be changed easily. There are neither correspondence, divergence nor allocation problems due to the fact that each pixel of sensor calculates a range value and therefore no object information is lost by the optical projection on a planar sensor.

Field of View problem

ToF cameras do not depend on geometrical parameters contrary to SV systems where the distance between cameras implies different triangulation possibilities and therefore a range of depth resolutions. The usage of active modulated light source makes ToF systems more effective and reliable.

Two different approaches are the most common modulation techniques used for depth measurement. The first one is *pulsed modulation*, introduced 20 years ago by Moring *et al.* [15] and it is not very frequently used by the literature. The alternative is *Continuous Wave Modulation* (CWM) explained by Beheim *et al.* in [16]. CWM is nowadays used in ToF systems because this method does not require high rise and fall times allowing then several sources of light to be used. Usually, square waves or sinusoidal waveforms are applied for modulation. The idea is simply to measure the phase between sent signal and received signal instead of measuring the time to go and return of a single light source. Once modulation frequency f_{mod} is established, the measured corresponding phase means directly the time-of-flight [11]. Using Equation 2.6 and replacing ϕ by the frequency response of the modulation

$$\phi = (\frac{\varphi_0}{360^o} + N \cdot 360^o) \ with \ N = 0, 1, 2, 3, ... \tag{2.10}$$

In case of modulation, $2 \cdot \pi\omega$ in 2.6 equals to f_{mod}, and therefore the range of the camera can be expressed as

$$R = (\frac{c}{2 \cdot f_{mod}}) \cdot (\frac{\varphi_0}{360^o} + N \cdot 360^o) \ with \ N = 0, 1, 2, 3, ... \tag{2.11}$$

Studying the above Equation 2.11, if f_{mod} is set to 20Mhz as [17] recommends for typical PMD cameras, the *Non Ambiguity Range (NAR)* turns to

$$NAR = \max(R) = \frac{c}{2 \cdot f_{mod}} = \frac{3 \cdot 10^8 m/s}{2 \cdot 2 \cdot 10^7 s^{-1}} = 7.5\, m \tag{2.12}$$

giving an idea of the spaciousness those devices can take in. Equations 2.11 and 2.12 demonstrates that NAR depends only on the frequency f_{mod} applied to obtain a larger or shorter distance region.

Correspondence problem

As mentioned before, one of the most relevant problems that emerge in ToF technology is the correspondence problem, also named by Chung *et al.* as *correspondenceless* in [18]. The idea is quite simple, SV systems need rich textured frames to offer good reliability. That is because disparity has to be found between equivalent pixels in both images. If gray values are quite similar along the *epipolar lines*, disparity levels would be erroneous inducing into bad disparity maps.

A small jug has been captured in the laboratory (Figure 2.6.a) and computed with a SV system (Figure 2.6.b) and then with a ToF system (Figure 2.6.c). It is trivial to find out how background is computed erroneously when it is poorly textured. Contrary to ToF cameras, SV systems require several settings such as disparity window size or maximum disparity levels to be defined beforehand, otherwise depth maps will not correspond to observed systems. Furthermore, due to the fact that SV systems run without active lighting, they generate shadows creating false positives and hence, they cannot estimate the 3D information of the objects due to the correspondence problem.

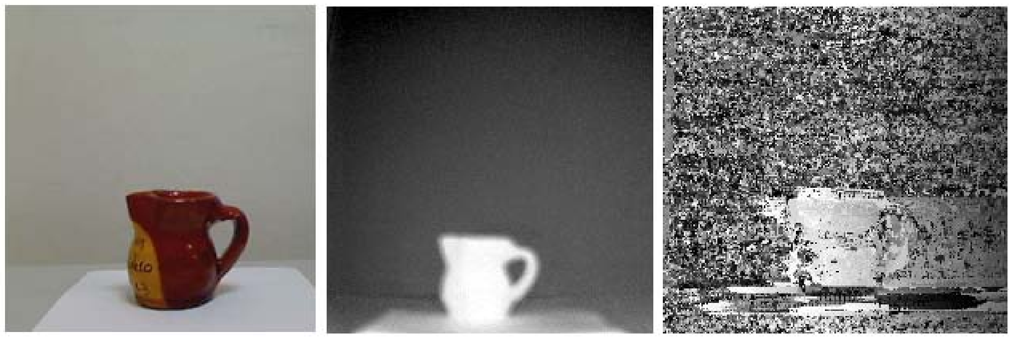


Figure 2.6: *Correspondence problem.* Real example comparing SV and ToF technologies for a small jug in the laboratory.

Influence of the Temperature onto the distance measurement

As Kahlmann *et al.* explains in [19], CCD and CMOS photo sensors are extremely temperature dependent and therefore an increment of temperature causes a higher rate of thermal generated electrons. There happen two effects on the sensor:

- *Internal Temperature.-* The internal heating of the sensor creates a self-induced error that requires from approximately 10 minutes (studied in [19]) to get stabilized when measuring a fixed target.

- *External Temperature.-* The measured distance increases with temperature (see Figure 2.7). It can be also pointed out that the problem is systematic and thus could be fixed with a calibration procedure. The determined drift lies around 8mm/^{o}C. That means that during the first few minutes, while the camera warms up, the measured distance increases.

Other error sources

In first place, Lindner and Kolb [20] perceive a periodic error related to the measured distance. The error has a wave length of approximately 2m. They account this error to the fact that the evaluation of the distance assumes a impeccably sinusoidal light source, which in practice is not granted.

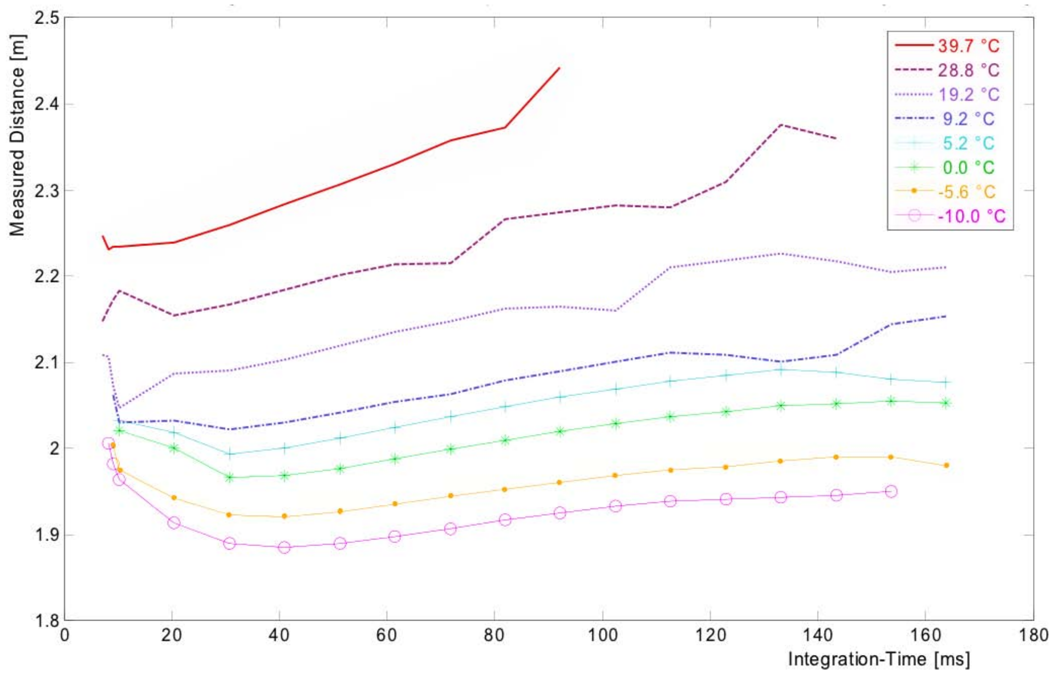


Figure 2.7: *Temperature effect on ToF.* Relation between Integration-Time and Measured Distance with respect to different external temperatures for a fixed object (extracted from [19]).

Another source of error is the time elapsed to transfer the signal from the sensor to the CPU. This error rests on the relative location of the sensor within the array (i.e. the pixel in the image) as expressed by May *et al.* [21]. Additionally, since the distance prediction depends on the amount of reflected light, the intensity of the optical signal (i.e. the brightness) affects the distance measurements. However, a low intensity leads to a wrong signal to noise relation, misleading the measurement erratically. Another error arises from the shutter time of the camera, i.e. the time over which the camera integrates the image.

Longer integration times tend to shift the image towards the camera [19] [22]. Kahlmann *et al.* [19] also report that the internal temperature of the camera, as well as the external temperature, influence the depth measurements. But even after the temperature has been stabilized, Kahlmann *et al.* announce a little deviation that, according to them, is due to a cool down which occurs in between taking the individual images. Finally, Gudmundsson *et al.* [23] discusses in the effects of multiple reflections on the distance measurements.

2.3 RGB-D Systems

After the arrival of ToF technology, devices were able to extract pretty small frames at fast rates containing 3D raw maps without texture information. This absence of color information led to emerging strategies to embed indirectly the chromaticity of the scene in the depth map using custom calibration techniques.

Both devices were calibrated using Bouguet stereo method thanks to the gray-scale output of the PMD camera as explained by the author in Bueno *et al.* [24] and Burrus *et al.* [25]. It is thus possible to get color information for each pixel of the depth image by back-projecting the pixel back to 3D using the estimated depth, and then projecting it onto the color image. Figure 2.8 contains the imagery system developed in the laboratory using a PMD [vision] CamCube 2.0 camera with a resolution of 204×204 pixels and a QuickCam Pro 9000 camera with a resolution of 640×480 pixels to provide color information. The goal of this setup is to mix up depth information with texture with good reliability.

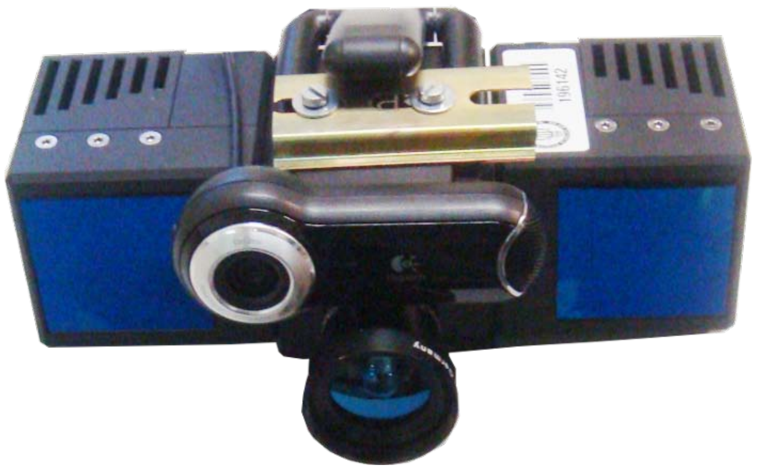


Figure 2.8: *RGB-D custom system.* Placement of the ToF and RGB cameras to generate RGB-D maps using Zhang calibration.

Raw data from the ToF camera are very noisy and suffer from various bias. A 5×5 median filter is used to remove spurious values, and the bias are reduced using similar techniques as [26]. Radial bias becomes the most significant distortion: pixels which are further from the optical center tend to get higher depth estimates. To reduce it, ground truth depth values were computed using checkerboards, and then compared to ToF measurements. As shown in Figure 2.9, a polynomial function is then fitted to the observed errors and is then used as a depth correction

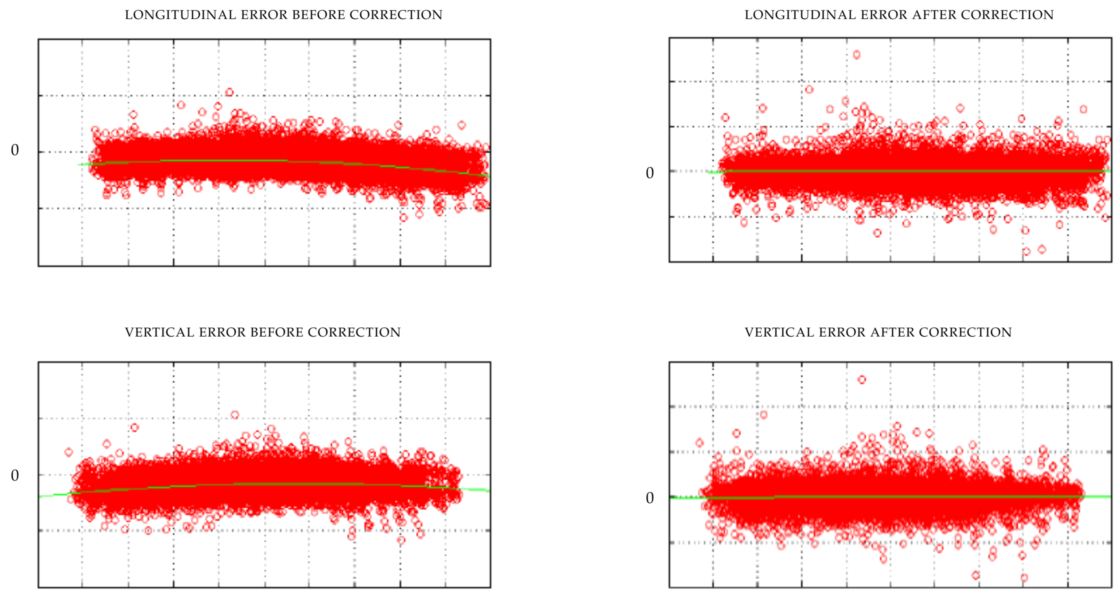


Figure 2.9: *Correction of depth measurements.* Each red circle corresponds to an error computed by comparing a ground truth depth value with its ToF estimate. On the left side, values are plotted before correction and show the fitted correction polynomial (green). This polynomial is then used to apply a depth offset, and corrected values are plotted on the right side.

offset. ToF measurements also suffer from relatively strong bias due to the varying reflectivity of the objects: dark objects usually appear closer. However, this bias turns out to be difficult in order to improve the reliably and therefore this problem has not been faced during this work.

Two different alternatives have been used in order to obtain and interpret the intrinsic and extrinsic parameters for both cameras. That is, the individual and internal variables (focal length, center of image, etc.) and external variables (distance between cameras and relative rotation).

Firstly, OpenCV library [5] has been used. This library offers a complete set of *API's* that allow recognizing the desired parameters with a batch of chessboard's pattern frames. For this project, 20 frames of chessboard pattern in different poses have been used. It is important to remark that the more changes in pose, the better the results become. That yields from the fact that the parameters are obtained iterating on those frames's poses. Some of the results before artifacts removal are shown in the following Tables. It is easy to interpret the results of the calibration.

[5]More information in `http://opencv.org/`

Secondly, those results have been checked with the library provided by Jean-Yves Bouguet, called *Camera Calibration Toolbox for Matlab* [6]. The values were almost the same, with slight changes in distortion factors reason why they are not represented here.

Focal lengths are normal and centers of image are close to the real centers in both cameras. For this work, images have been resized into 640×480 pixels for Logitech's frames and into 500×500 pixels for PMD camera. This inconsistence is changed and fixed later with a physical offset of the cameras. Intrinsic values for both cameras can be shown in Table 2.1.

CALIBRATION RESULTS		
Intrinsic	Logitech Camera	ToF Camera
f_x	540.475185	773.901272
f_y	541.428144	771.584695
c_x	307.210290	230.014322
c_y	226.635229	244.194863

Table 2.1: Intrinsic parameters of both cameras for the first approach

The centers are shifted due to distortions. For the distortion parameters, Table 2.2 describes the results obtained.

CALIBRATION RESULTS		
Distortion	Logitech Camera	ToF Camera
k_1	0.0288288	-0.34804000
k_2	0.4753549	-1.83188588
p_1	0.0042067	0.00453175
p_2	0.0004481	0.00006298
k_3	-3.8845537	26.118679

Table 2.2: Extrinsic parameters. Relation between both cameras in 3D world coordinates

It is important to highlight the large radial and tangential distortion that appears in the PMD camera. Figure 2.10 represents the effect of both distortions in

[6]More information in `http://www.vision.caltech.edu/bouguetj/calib_doc/`

the image and how it is rectified.

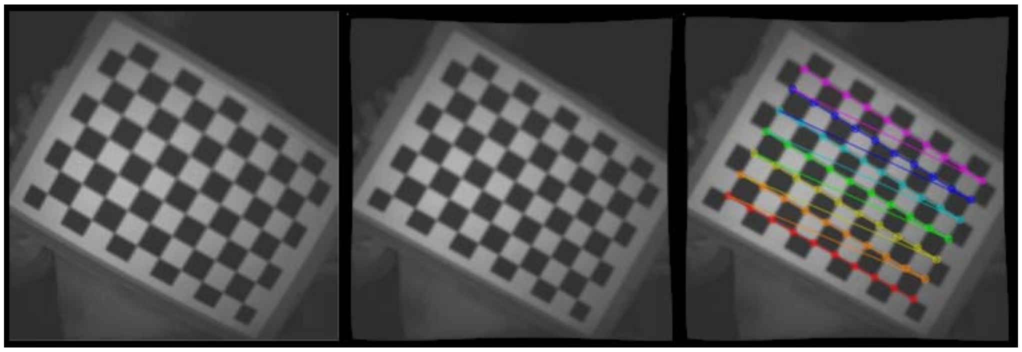


Figure 2.10: *Distortion correction*. Left image shows the original image from the 3D camera and central image shows the undistorted correction. Last image shows the chessboard corner detection from OpenCV library.

Finally, the extrinsic parameters for translation (in millimeters) and rotation (in radians) are listed in the following lines and represented in Figure 2.11.

$$T = [0.843615, -41.014156, 7.2423722] \tag{2.13}$$

$$R = \begin{bmatrix} 0.9999887 & -0.0021876 & 0.0042038 \\ 0.0025968 & 0.9949743 & -0.1000964 \\ -0.0039637 & 0.1001062 & 0.9949688 \end{bmatrix} \tag{2.14}$$

That means that there is a spatial offset of approximately 4 cm in the Y axis and 0.7 cm in the Z axis and they are over the same vertical plane.

2.4 Light Coding

Light coding technology (also known as structured light technique) was developed originally by Curless and Seitz [27]. It was lately developed in a single microchip by PrimeSense *Ltd.*, a company founded in 2005 in Israel. Their mayor contribution was a novel 3D depth sensor that works by coding the scene with near-IR light, which is invisible to the human eye [28]. This sensor alternative projects a patter of pixels in the scenario and captures the deformation for those pixels once they are projected into the objects. If the pair IR emitter–camera is well calibrated,

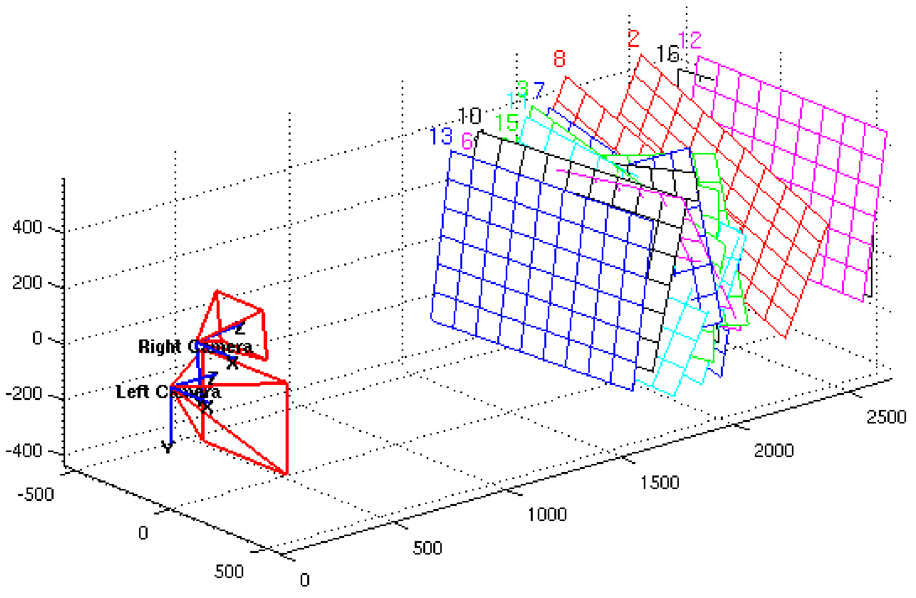


Figure 2.11: *Stereo calibration*. Calibration of both cameras and its representation using the Matlab Toolbox provided by California Institute of Technology.

it is possible to estimate the depth for each pixel based on the matrix deformation. The estimation is done triangulating signals from emitter, camera and pixel matrix positions. According to Kramer *et al.* [29], the depth map is constructed by analyzing a speckle pattern of IR laser light at 830nm. This principle, which was firstly proposed by [27], was improved by PrimeSense by the combination of two classic computer vision techniques:

- *Depth from focus.*- Based on the principle that objects positioned further away will be more blurry and those closer will be more focused. Using a special *astigmatic* lens with different focal length in x and y directions, a projected circle then becomes an ellipse whose orientation depends on depth as shown in Fig. 2.12.

- *Depth from stereo uses parallax.*- If you look at the scene from a different angle, objects that are close gets shifted to the side more than those which are further. Kinect devices analyzes the shift of the speckle pattern by projecting from one location and observing from another.

The pattern used in Kinect devices is part of the PrimeSense patent (see Fig. 2.13, [30]), generated from a set of diffraction gratings with special care to lessen the effect of zero-order propagation of a center bright dot. However, there exit several techniques listed by Pan *et al.* in [31] to achieve this patterns. Some of them are

FIG. 2

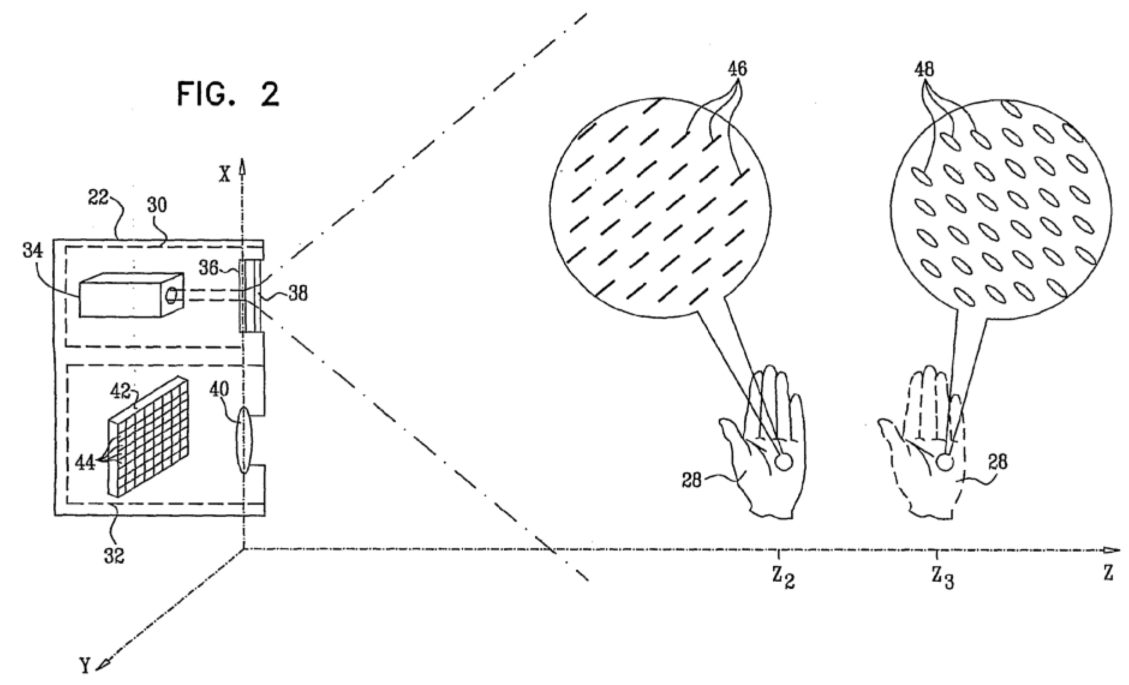


Figure 2.12: *Light coding.* A projected circle then becomes an ellipse whose orientation depends on depth (Freedman *et al.* , US Patent 2010/0290698).

colored strips, colored disordered coding, colored grid of points, black and white strip, black and white dot matrix or a black and white grid, as in this particular case.

The solution then uses a standard off-the-shelf CMOS image sensor to read the coded light back from the scene. Their first chip release, with more than 20M users is Carmine (PS1080), a multi-sense system which provides a synchronized depth image, color image and audio stream.

Kinect device contains three vital pieces that work together to detect the motion and create the physical image on the screen: an RGB color VGA video camera, a depth sensor, and a multi-array microphone.

1. The RGB camera detects the red, green and blue color components. It has a pixel resolution of 640×480 (VGA) and a frame rate of approximately 30 fps. This helps in facial and body recognition, texture comparison or features extraction.

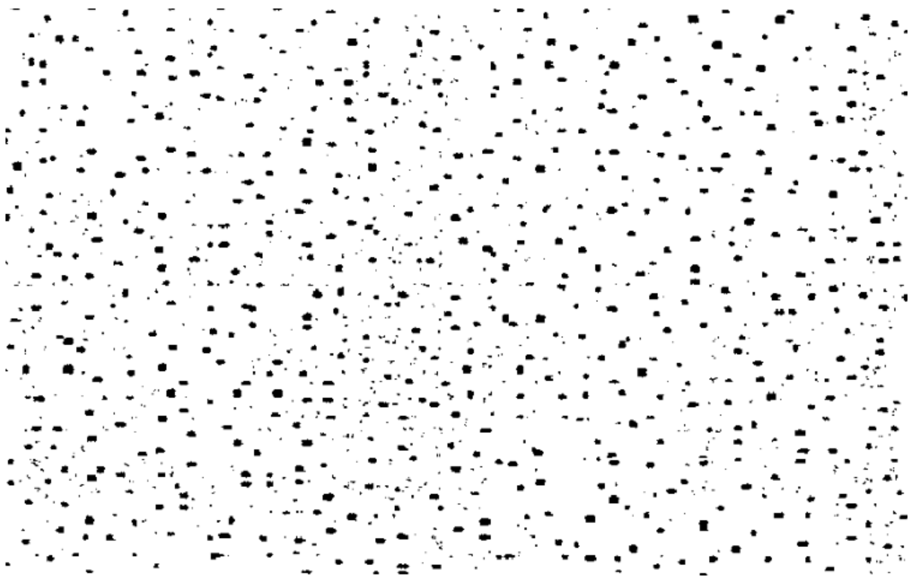


Figure 2.13: *Light coding.* IR pattern patented by PrimeSense for Kinect 3D sensing (Shpunt et al., US 2008/0106746).

2. Depth sensor contains a monochrome CMOS sensor and infrared projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the user's body by transmitting invisible near-infrared light and measuring its ToF after it reflects off the objects.

3. The microphone is actually an array of four microphones that can isolate the voices of the user from other background noises allowing players to use their voices as an added control feature. This feature is not yet used up to day in robotics but the microphones will be surely exploited in HRI applications in a few years.

Calibration parameters are not easily modified but since Kinect devices are massively produced, most of the distortion parameters, focal distance and optical parameters are customized by manufacturers. Khoshelham *et al.* [32] developed a deep study of accuracy and resolution for Kinect and determined that calibration error might reach less than 1 mm. for a workspace of one meter depth. With this information in mind, the sensor will be assumed to be undistorted and calibrated with the manufacturer estimations.

As every image processing solution, speed turns into a critical constraint. Most of the challenges require real-time execution enabling robots to interact with the

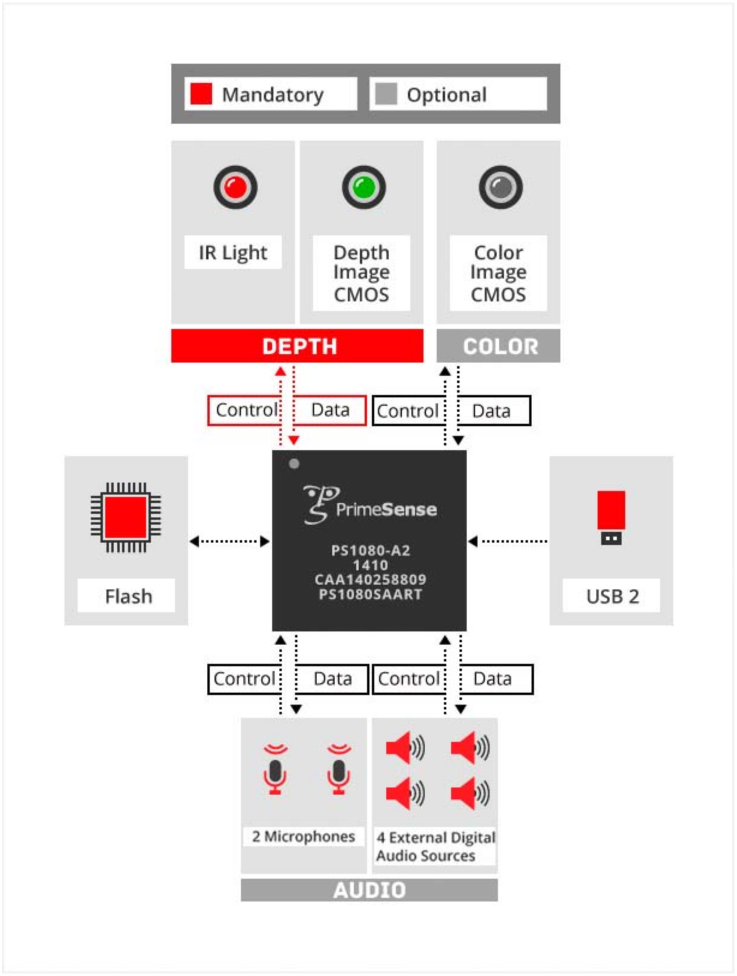


Figure 2.14: *Light Coding sensor Carmine PS1080.* Main structure of PrimeSense SoC's Carmine (PS1080), the processor for Kinect sensing devices.

environment as fast as it changes. Since the first step in perception is data acquisition, the interface has to provide data as faster as possible avoiding bottle necks. For this study, an Asus Xtion Pro Live camera is used. It contains a PrimeSense Carmine 1.8 3D Sensor. Table 2.3 contains the benchmark results for depth and color streams. The results have been extracted in the laboratory taking a sample of one thousand frames at different positions. The variability has been computed analyzing the output depth variations of the sensor in a fixed position. There are several features of the camera that are not used in this work such as the built-in microphones. However, it is not ruled out the possibility of using in the future to introduce some human-robot interaction.

MOST RELEVANT FEATURES	
Avg. frame-rate	**29.5498 Hz**
Variability	1.2 mm
Min range	800 mm
Max range	3500 mm
Resolution	VGA (640x480)
Field of View (Horizontal, Vertical, Diagonal)	57.5, 45, 69
Spatial x/y Resolution (2-Sigma Values) @2m	3.4
Depth Resolution (2-Sigma Values) @2m	1.2
Built-in Microphones	2
Data Format	16 bits
External Digital Audio Inputs	4 inputs
Dimensions: Width x Height x Depth	18 x 2.5 x 3.5 cm
Power Supply	USB (5V)
Maximal Power Consumption	2.25 W

Table 2.3: Benchmark for the Asus Xtion Pro Live camera

That means that the frame rate of the whole system will be at most the acquisition rate of 29.5 fps. Therefore, the three technologies presented in the previous sections can be summarized in Table 2.4.

OVERALL 3D TECHNOLOGIES			
Description	**ToF Camera**	**Stereo Vision**	**Structured Light**
Correspondence problem	No	Yes	Yes
Extrinsic calibration	No	Yes	Yes
Auto illumination	No	No	Yes
Untextured surfaces	Good performance	Bad performance	Good performance
Depth range	0.3 – 7.5 m.	Base-line dependent	Light-power dependent
Image resolution	Up to 204×204	High resolution. Camera dependent	
Frame rate	Up to 25 fps.	Typically 30 fps. Camera dependent	

Table 2.4: Highlights and drawbacks for the three 3D depth computation methods here proposed.

Chapter 3

Experimental Platform

MANFRED V2 Robot (**MAN FR**i**E**n**D**ly Version 2) is a eight *Degrees Of Freedom* (DOF) mobile manipulator designed and built by Robotics Lab research group from the Systems and Automation Engineering Department in Carlos III University of Madrid. MANFRED-2 is used as a general rule to test any algorithm developed by the researches. This includes MSc. students, PhD. students and Professors. It is used as an *adhoc* experimental platform for R&D in any field related with robotics. Most of the technical details have been extracted from Rascón [33] and Monar [34].

The main fields of application of MANFRED-2 are 2D and 3D location and mapping, machine learning, safe path planning, manipulation, grasping, affordance, multi-robot coalition formation, object recognition, SLAM, etc., allowing the robot to get on with its surroundings autonomously. Those environments are in any case designed for robots, increasing the difficulty during the perception of anything around. Manfred has been designed to avoid obstacles, localize itself inside a building or even turn the light on, but in this thesis, its main feature is focused on 3D perception. It has to be capable of recognizing home items such as cups, plates, glasses, mugs or any easy to handle objects, localize them and learn new elements.

The robot has been designed following the standards of rovers, used for spacial missions and also communication satellites. It has a differential mobile base (with two DOF) and a lightweight anthropomorphic arm of six DOF. It is composed of a list of independent sub-systems such as the sensory system, the locomotive system, the on-board computer, the power distribution system, drivers, etc., that are interconnected in such a way that it can be modified or improved in the near future with less efforts. The modular design separates successfully the hardware level (mechanical and electric interfaces) from the software level (communication interfaces between applications). This characteristic simplifies the subsystems integration. Figure 3.1 presents in MANFRED-2 and some of the features and subsystems that the robot includes.

The platform has a total of eight DOF, six in the arm and two in the base. The whole mechanical structure is connected through a common control board. It can be tele-operated and remotely controlled using a computer. The robot is formed by four main topics:

1. *Mechanical and electric structure.*- Main physical structure, drivers, batteries and any device required to let the robot move.

2. *Sensory interfaces.*- Sensors and all devices in charge of perceiving externals

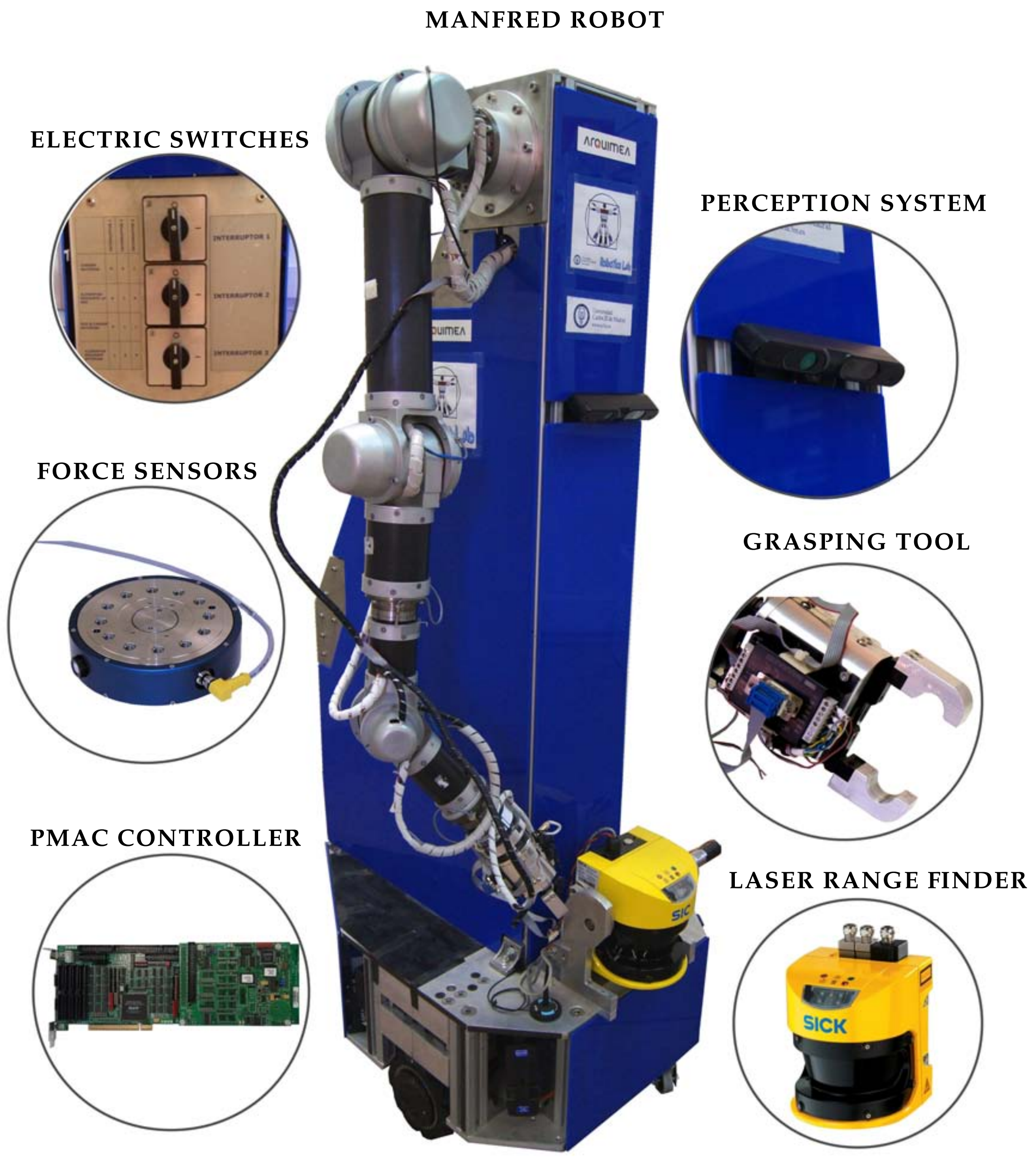


Figure 3.1: *Experimental platform.* Photo of MANFRED-2 and some of the subsystems integrated in the platform. The modular design separates successfully the hardware level (mechanical and electric interfaces) from the software level (communication interfaces between applications).

signals in order to make the robot decide how to behave.

3. *Control system.-* Description of the robot control system. Architecture and technical description of its main parts.

4. *Software layer.-* Information about the software platform, the operative system and other technical details about MANFRED-2 programming languages and internal structure.

3.1 Mechanical and Electric Structure

MANFRED-2 was designed with a very desirable expectation resumed in various specifications: high mobility for arm and base movements, mechanical and electrical robustness, high movement repeatability and straightforward integration of new parts and fixing based on a modular principle. Those requirements have been established for both the anthropomorphic arm and the wheeled base, in order to perform accurate, safe and human-like movements. Task performance has been identified as a must, and for this reason the mechanical design is pretended to be robust and reliable.

First sight improvements with respect to the previous robot version (formally named MANFRED) are a better force balance, several communication enhancements and a new software control. However, the force balance have been the most noticeable change, allowing the new version to reproduce critical arm tasks without taking extra risk. To do that, the base components (batteries, wiring, etc.) have been re-organized counterbalancing the arm when the robot is accomplishing a specific task. This tweak made the robot more stable and robust, making the grip achieve further elements inside the workspace. The most significant mechanical and electrical features are listed down below including the highlights versus the prior version of the platform.

- Stability has been improved moving some of the elements of the base to a lower height. The center of mass of the base has been displaced vertically.

- Rigidity has been boosted changing two elements: principal mast has been extended to the bottom plate and more columns have been added between the actual plates.

- Better calibration accuracy thanks to an improved height adjustment in the drive wheels.

- Redesign of the control panel to integrate safety button and switches in the same location.

- Casing attachment has been improved to facilitate the access to internal pieces with magnet strips.

3.1.1 Physical Description

MANFRED-2 structure is divided into two main parts: an octagonal base that supplies with electricity the platform and the principal mast where the arm is attached.

The base, with a height of 250 mm, is formed by two octagonal aluminum pieces one over the other one. Four batteries are integrated within the base, including the power electronic boards required by the engines in charge of the wheels. As mentioned before, the set of batteries of $12V$ and $45A/h$ disposed in serial connection are used as counterweight. Several DC/DC converters are included to cover the current demand of any electrical or electronic device plugged into MANFRED-2. Furthermore, the robot can be connected to a physical electrical network if demanded. Some of those elements can be seen in Figure 3.2.

Wheels are provided by Rockland Bayside, model DX 6-20-1M-P Servo-Wheels, and use a *brushless* engine and a 20:1 reduction gear. Each wheel is provided also with an optical encoder, model HEDS-5540-A06 with a resolution of 512 steps per revolution. There are also three passive wheels in permanent contact with the basement, providing a better stability to the base.

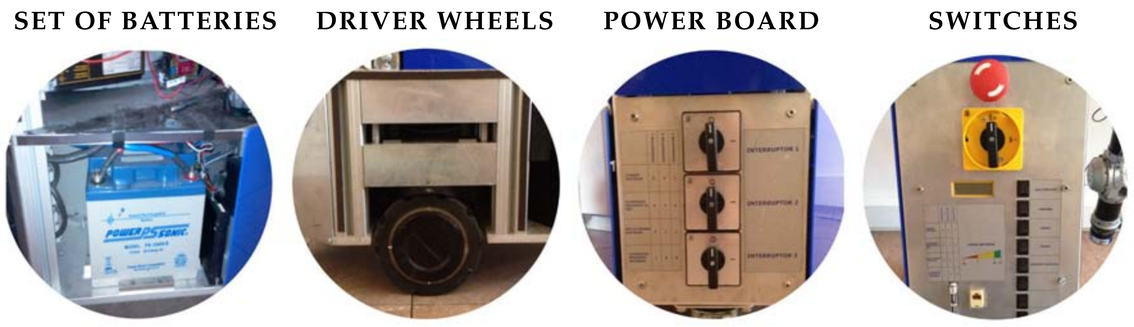


Figure 3.2: *Mechanical and electric details of the experimental platform.* MANFRED-2 is composed by a set of 24V batteries, two driven wheels, a control board with power configuration and a set of switches to control the sensors.

The principal mast is 150 mm height, fixed properly to the base. The mast supports the arm and the 3D camera among other sensors. The home position of the arm is designed in such a way that if current is stopped, it can move to that position without damaging other elements. Mast is designed with two anchor blocks so another extra arm can be attached in a near future. Actually, a new version of the robot is being designed and several new features will be listed afterwards.

Other elements found in the mast are 48 V DC/ 24 V DC to feed the SICK 3000 Laser, 48 V DC/ 12 V DC to feed the Asus Camera, 48 V DC/ $\pm$ 15 V DC traco-power converters, fuses, supply power, power drivers for the arm, communication wirings, and connectors such as RS-232, VGA, USBs and security switch buttons.

Table 3.1 covers the weighty elements of the platform

MANFRED-2 WEIGHTS		
Device	unit weight [kg]	total weight [kg]
12V Battery	15.40	61.60
Support structure	29.00	29.00
Driver wheels	7.00	14.00
Wiring & comms.	6.00	6.00
Drivers	0.68	5.44
Computer	5.00	5.00
Electronic components	2.75	2.75
DC/DC converters	2.00	2.00
Caster wheels	0.42	1.26

Table 3.1: Weights of the main devices installed in the platform.

3.1.2 Robotic Arm LWR-UC3M-1

MANFRED-2 has been provided with a six DOF robotic arm, developed by several researchers in the department. It has been named LWR-UC3M-1 (**L**ight**W**eight a**R**m UC3M-1) and it is designed to perform a large amount of manipulation tasks: object grasping, robot-human tasks, open doors or switch the light in a room. It is composed of rigid elements made of carbon fiber connected by revolution joints.

The DOF of the robot are delegated in the following form:

- SHOULDER with two DOF allowing movements in the antero-posterior and transverse axis.

- ELBOW with one DOF permitting the same movement as humans do.
- WRIST with 3 DOF, one for pronation-supination movement, another one for abduction and adduction and the last for flexion-extension depending on the relative position with respect to the previous joint.

The arm was designed with several material proposals such as magnesium, aluminum alloys, titanium alloys, lithium and carbon fiber. Magnesium was rejected because its mechanical properties do not allow an easy mechanization. Titanium is a high expensive material, lithium might become dangerous in direct contact. Accordingly, aluminum has been used for almost the whole structure while some links are made of carbon fiber. The links, hollow cylinders are used as transfer elements for wiring, electronic components, control and electrical parts.

However, a new version of the arm LWR-UC3M-2 is being developed these days. It has several new improvements that are listed in Table 3.2 and gathers the most important highlights of this new arm with respect to the actual situation.

ROBOTIC ARMS COMPARISON		
Characteristic	LWR-UC3M-1	LWR-UC3M-2
weight [Kg]	18.00	11.00
max load [Kg]	4.50	5.00
range [mm]	955	1000
DOF	6	7
encoders	relative	absolute
tool	grip	hand

Table 3.2: Comparison between the actual robotic arm LWR-UC3M-1 and the new approach LWR-UC3M-2.

The addition of the new arm in MANFRED-2 will turn the platform into a fully anthropomorphic robot with two arms. The following Figure 3.3 contains both versions, on the left the new LWR-UC3M-2 arm and on the right the actual version LWR-UC3M-1.

The actual arm is located on the lateral side of the mobile robot's mast in such a way that the computer vision sensors and laser telemetry systems are not interfered with the arm. The arm joints are equipped with DC brushless motors from Kollmorgen and Harmonic Drives model HFUC-2UH for reducing the speed and increasing the torque. An optical encoder HEDS550 is included to measure the rotations and a presence inductive sensor is attached on each motor to find out the

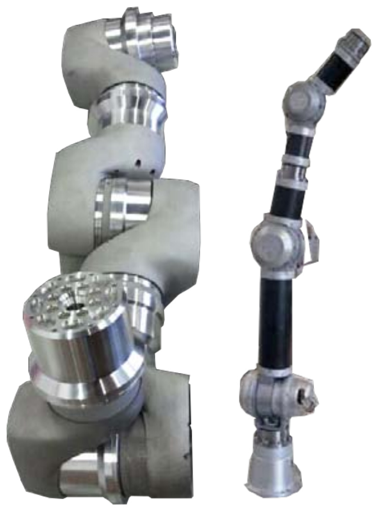


Figure 3.3: *Robotic arms LWR-UC3M-1 and LWR-UC3M-2* . Left image shows the new version of the arm that will be provided in the near future to MANFRED-2. Right image shows the actual version LWR-UC3M-1.

home position for each joint. LWR-UC3M-1 encoders are relative, as shown in Table 3.2. This means that encoders are designed to return relative information from the home position to the actual position.

Home position has been designed to have each motor relaxed with no energy consumption. To convert any relative measurement to absolute values, the robot arm has to be moved to the home position, and from there, to the desired point. The home function has been implemented direct into the PMAC2-PCI.

3.1.3 Anthropomorphic Robot Hand: Gifu Hand III

Two of the most challenging tasks that are being faced these days in the research group are the control and integration of the anthropomorphic robot hand named Gifu Hand III from the Virtual System Laboratory in Gifu University [35] in MANFRED-2. Some of the technical specifications of the hardware are listed in Figure 3.4.

The Gifu Hand is a five-fingered hand driven by built-in servomotors and has 20 joints with 16 DOF. The hand is controlled with ART-Linux real-time operating system. The thumb has four joints with four DOF and each of the fingers has four joints with three DOF each. The movement of the first joint of the thumb and of the fingers allows adduction and abduction, and that of the second joint to the fourth joint allows ante-flexion and retro-flexion.

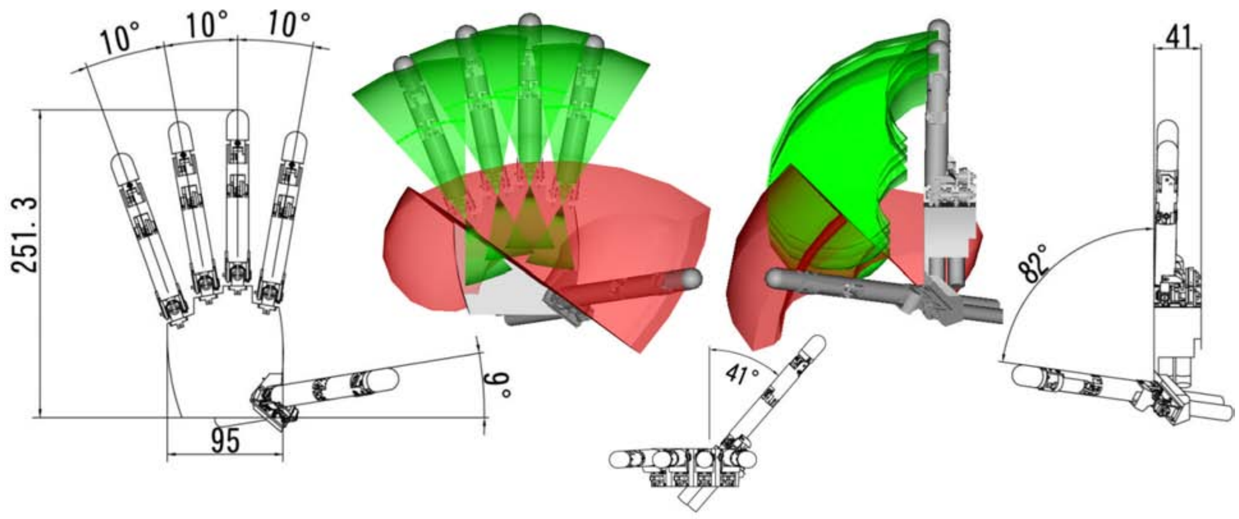


Figure 3.4: *Structure of Gifu Hand III*. Representation of the robotics hand from different views and mobility space for each finger. In red the space for the thumb, in green for the rest.

The most advanced characteristics of this hand are: a high response (the minimum bandwidth of the robot hand is 7.4 Hz while the human hand is even lower than 5.5 Hz) and a large *opposability* of the thumb. Furthermore, it can be enveloped with a distributed tactile sensor that consists on a grid pattern of electrodes and uses conductive ink in which the electric resistance changes in proportion to the pressure on the top and bottom of a thin film. This sensor can be easily attached to the hand providing with valuable outputs and statistics for grasping researches.

3.2 Sensory Interfaces

Several sensors are attached to MANFRED-2 including lasers for 2D location, motorized lasers for 3D location and mapping, odometry sensors or perception sensors. Each of them transforms a physical variable into a logical variable, allowing the robot to acquire, transform, interpret and react to any external impulse performing tasks.

3.2.1 Force Sensor

In order to let MANFRED-2 perform safe and effective manipulation tasks such as holding the doorknob or pressing a switch, a force/torque sensor is included in the end point of the arm from JR3 manufacturer, model 67M25A50-I40. It can measure as far as 11 kg with a weight of 0.175 kg at 8 measures per second rate. It has been displayed in Figure 3.1. Force sensor is implemented with six strain gauges proportional to the force applied in X,Y and Z axis. However, this sensor requires of a digital signal converter (DSP) to process the output. In this case, a ADSP-2184 sensor from Analog Devices is in charge of converting the analog signal into a digital value. Finally a PCI card connects the DSP with the computer where data is then processed.

3.2.2 Laser Telemetry Sensors

Navigation, localization and mapping are abilities that require from external information to be successfully carried out. The aim of telemetry sensors is to provide with 2D or 3D data representing which are the obstacles around the robot. If the robot has to avoid objects on its way, recognize where it is positioned in a workspace or estimate a motion planning, measuring the surroundings is quite imperative.

Two different sensors are included in MANFRED-2. Both are laser range finders and their details are listed above:

- SICK S3000.- A bi-dimensional laser telemeter SICK S3000 Professional CMS is located at the base of the robot. Its assignment is to detect objects with a minimum size of 35 cm in a range between 0.1–10 m. Examples of these obstacles are people feet, chairs, boxes and walls. Its angular resolution is 0.25^o (with a 180^o opening range) and has a power consumption of 800mA with 24V. It is connected to the computer directly using a USB 2.0 port. It has been mounted on a movable stand to permit pitch movements so a 3D scenario can be modeled (see Figure 3.5) with consecutive scans. Measurement error is below 10 mm.

- HOKUYO UTM-30LX.- This telemeter has a better opening range reaching 270^o with a distance range between 0.1–30 m. This sensor is located at 1 meter above the ground to intercept not only navigation obstacles but also further elements such as walls, room openings or doors. Due to its 0.25^o angular resolution, it acquires 1081 measures per scan in 25 ms. It is connected to the computer directly using a USB 2.0 port and consumes about 12V and 700mA.

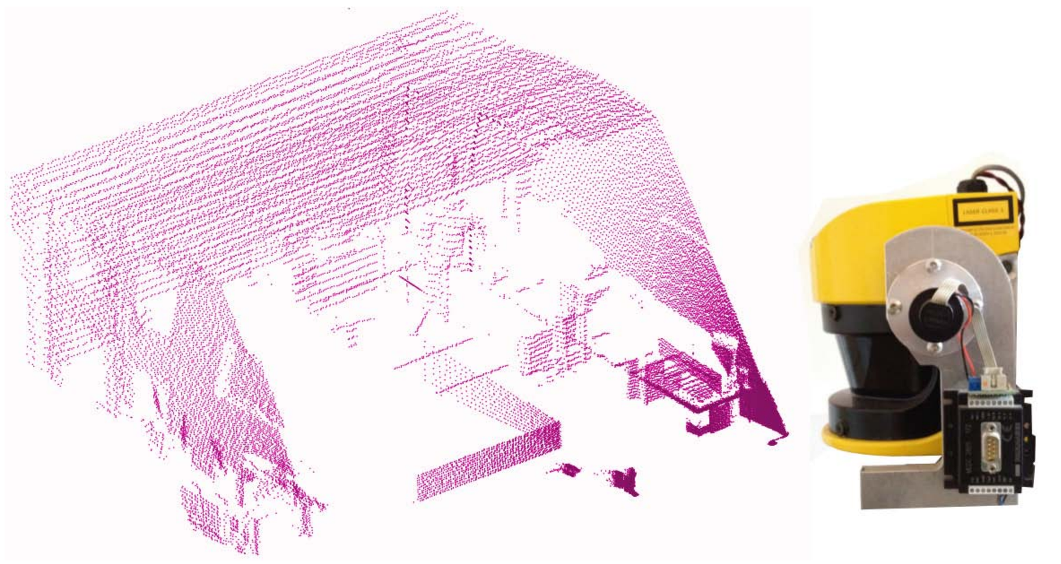


Figure 3.5: *Laser Telemetry Sensor*. Result of the SICK 3000 range laser sensor scanning in the laboratory on the left; a detail of the pitch-movable stand on the right.

3.3 Control System

The platform has been provided with eight motors distributed in two parts: two in the base and six in the arm. All those motors have to be controlled constantly. For this reason, a controller card has been installed.

The PMAC2-PCI is a Programmable Multi-Axis Controller card developed by Delta Tau Data Systems Inc. It is a high performance device able to control eight axis simultaneously with high precision. Tunning its more than a thousand configuration variables, PMAC2-PCI is more than enough to control the eight DOF. It is connected to a computer via PCI bus or using a serial port. PMAC2-PCI (see Figure 3.6) is able to run its own applications independently as an industrial computer. The DSP that has been incorporated is DSP56002 of 24 bits with an operation frequency of 40 MHz. It works in real time and it supports multi-task operations.

However, PMAC2-PCI controller card cannot be connected directly to the motors. It requires an intermediate interface. In this case, control commands to the motors are transmitted using an additional card named ACC-8E. Each 8E card is fed with $\pm$15V, has four 18-bit D/A converters and is able to command two motors using an analog input. Thus four of those cards are needed to control the full platform. Furthermore, each 8E card is plugged in the control card through a 100-pin

Figure 3.6: *PMAC2-PCI*. The control unit for MANFRED-2 manipulator robot is an eight axis controller card PMAC2-PCI.

flat bus named JMACH.

There exist two guidebooks named *Software reference manual* and *PMAC2 user manual* that explain how to configure and make use of the controller card. Set-up and starting operations of PMAC2-PCI card are extremely arduous assignments that must be performed with certainty. Configuration variables are referenced as I-variables and there exist 1025 altogether. Manufacturer provides with a software that allows tunning the modification of the configuration parameters and includes other interesting tools such as:

- *Terminal window.*- For sending ASCII encoded commands to the card.
- *Watch window.*- Track and modify internal variables in real time.
- *Position window.*- Trace motor movements, encoder positions, speed and tracking errors.
- *Tunning Pro.*- Configure each PMAC2-PCI parameter such as PID, filters, DAC calibration files, etc.

PMAC2-PCI controller card is able to execute different programming commands:

- *Motion programs.*- Controller card executes a task line by line. Each task corresponds with a specific command such as motor movement.The PMAC2-PCI controller card can store and execute up to 256 motion programs and is ready to execute another program or terminal commands.

- *Programmable Logic Controller.-* If a program needs to be executed asynchronously during a large amount of time, they have to be PCL coded. These snippets are exactly equal to motion programs with the only difference that they include a header at the beginning of the code to classify them as PCL. Once the program is run, the controller card will execute the code on each cycle as long as it is possible.

Chapter 4

Detecting supporting planes

This chapter proposes a novel plane fitting algorithm that improves the robustness of state of the art alternatives for this. It is based on DE algorithm. As this optimizer is used multiple times throughout the document, this chapter is going to focus on the problem statement and not on the optimizer particularly. Nevertheless, the genetic algorithm will be described in detail in the subsequent chapters. As a main advantage of this contribution, the plane fitting proposal yields better results in cluttered scenarios where the quantity of objects resting on a table would become an issue for the plane computation.

As this thesis is mainly based on perception, the Asus Xtion Pro Live 3D sensor as three-dimensional sensor device has been selected. This camera version is categorized as a *developer* version of the original Kinect camera with several modifications to make it simpler: tilt and pan motors are avoided and the microphone array has been reduced to just two of them. This sensor is positioned in the front upper part of MANFRED-2 as shown in Figure 3.1. From this position, the robot perceives the environment as humans do. Figure 4.1 shows an example of a room with objects and how the robot perceives that environment.

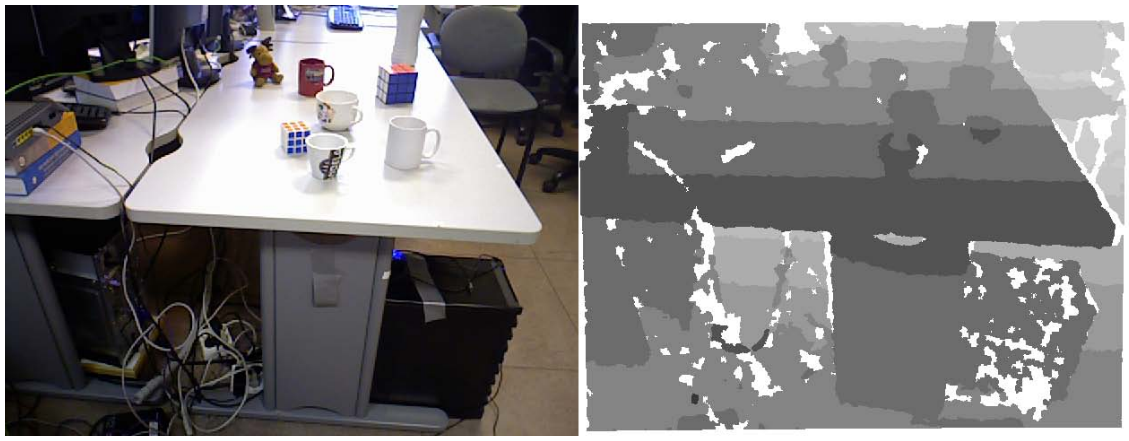


Figure 4.1: *MANFRED-2 Perception example* (a) Texture of the 3D scenario perceived by the robot. (b) Depth representation of the 3D scenario in gray scale. The darker the region, the closer to the sensor.

The robot has to interpret the information acquired by the camera and transform it into real data. Before light coding technology came up, 2D segmentation was performed using HSV color segmentation such as Dillmann *et al.* [36] or Rusu *et al.* [37] converting shape blobs into 3D poses using PCA to reduce dimensionality. This solutions were based on huge databases of different views for the same object and then compressed into *eigen-views*. Those techniques are slow and, more importantly, color dependent.

Lately, with the overrunning of stereo vision and RGB-D systems, 3D analysis started. Researchers such as Rusu *et al.* [37] began extracting supporting planes using stereo vision. These alternatives focus the problem to those points sufficiently close to the plane. They proposed a supporting plane extraction based on RANSAC (as will be explained below from Zuliani [38]). However, they claimed that according to the Extended Gaussian Image (EGI), most of the estimated point normals in their datasets are found around the principal XYZ directions. Hinterstoisser *et al.* [39] merged both visual features and 3D data avoiding planes detection. This solution required of more than 3000 templates comparison of each object to be detected but it can become extremely fast when programmed in GPU.

In order to provide MANFRED-2 with a successful object grasping, 3D map will be examined so any supporting planes will be extracted. As a general rule, any plane found on the raw point cloud will be chosen as a candidate. Afterwards, a filter will choose if there exist a principal supporting plane and populate it depending on several conditions:

- Percentage of *inliers* at a certain thresholded distance to the proposed supporting plane.
- Orientation based on the geometrical parameters of the proposed supporting plane.

Until now, multiple ways of ground segmentation have been proposed for robot navigation and obviously SLAM applications. Most of them were based on color maps provided by an on board camera, extracting texture features such as Cherian *et al.* [40]. Others such as Milella*et al.* [41] use a specific mixture of sensors such as radars and cameras to determine the path. In three-dimensional perception, there exist multiple ways to extract planes inside 3D point clouds. The majority try to fit multiple parametric models inside the data using a geometric arrangement of the input data points. For instance, Linarth in [42] is based on a Bayesian hypothesis using a Particle Filter to provide a robust estimation of the plane parameters taking in mind the non-linearities.

However, using exclusively depth data is possible to determine the planar segmentation. Enjarini and Gräser [43] proposed a gradient of depth feature to filter points belonging to surfaces based on their relative depth to neighbor pixels. Each pixel is then defined by two components: *Magnitude Gradient of Depth* (MGoD) and

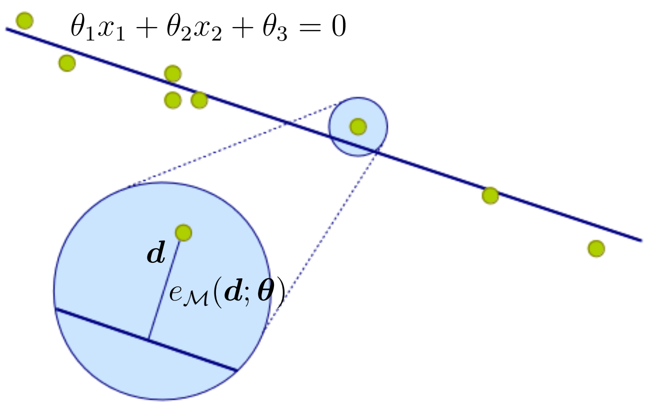

$\theta_1 x_1 + \theta_2 x_2 + \theta_3 = 0$
$\boldsymbol{d}$
$e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta})$

Figure 4.2: *Estimating 2D Lines.* Line fitting for 2D data.

Directional Gradient of Depth (DFoD). Both features are based on pixel-wise depth information. The author proposed in Bueno *et al.* [44] a ground truth method based on depth.

This thesis proposes a novel plane fitting method based on evolutionary algorithms as will be described later. Error measurement is generally performed applying a generalization of *least squares* (LS) problem. The LS technique is proved to be inefficient for nonlinear estimations but can be largely improved. The estimation filtering method is based on the optimization of the observation likelihood but without modeling a probabilistic distribution.

4.1 Estimating 2D Lines

In order to understand the 3D model fitting problem, it is recommended to start with a 2D simple case and extrapolate it to three-dimensional statement. Considering a set of N points $D = \{d_1, d_2, ..., d_N\} \subset \mathbb{R}^2$, the estimation of the best line (model) that fits such points is desired. There exist an error function that represents the consistency of the model such as

$$e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta}) = \frac{\theta_1 x_1 + \theta_2 x_2 + \theta_3}{\sqrt{\theta_1^2 + \theta_2^2}} \tag{4.1}$$

This error is a monotonically increasing function of the absolute value of the signed error. And the model $\mathcal{M}$ used to fit the measurements is $\theta_1 x_1 + \theta_2 x_2 + \theta_3 = 0$, $\boldsymbol{\theta} \in \mathbb{R}^2$ being the parameter vector. This fitting is known as orthogonal regression because of its nature: each sample point is evaluated measuring the orthogonal distance to the model itself.

If the fitting error is modeled as a Gaussian random variable with zero mean and σ_η standard deviation ($e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta}) \sim \mathcal{N}(0,\sigma_\eta)$), the maximum likelihood estimation will try to find the parameter vector $\boldsymbol{\theta}$ that maximizes the likelihood of the joint error distribution such as

$$\mathcal{L}(\theta) \equiv p[e_{\mathcal{M}}(\boldsymbol{d}_1;\boldsymbol{\theta}),...,e_{\mathcal{M}}(\boldsymbol{d}_N;\boldsymbol{\theta})] \tag{4.2}$$

where p represents the joint probability distribution function (pdf) of these errors. The estimate of the parameter vector that maximizes the probability of observing the signed errors $e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta})$ is given by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{4.3}$$

Assuming that error distributions are independent, it is valid to consider log-likelihood since logarithm operator is a monotonically increasing function which does not affect the maximization problem $\mathcal{L}^*(\boldsymbol{\theta}) \equiv \log \mathcal{L}(\boldsymbol{\theta})$, and therefore

$$\mathcal{L}^*(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p[e_{\mathcal{M}}(\boldsymbol{d}_i;\boldsymbol{\theta})] = \sum_{i=1}^{N} \log p[e_{\mathcal{M}}(\boldsymbol{d}_i;\boldsymbol{\theta})] = \tag{4.4a}$$

$$= \sum_{i=1}^{N} \left(\log \frac{1}{Z_G} - \frac{1}{2} \left(\frac{e_{\mathcal{M}}(\boldsymbol{d}_i;\boldsymbol{\theta})}{\sigma_\eta} \right)^2 \right) \tag{4.4b}$$

where $Z_G = \sqrt{2\pi}\sigma_\eta$ gives the normalization parameter for the Gaussian distribution. The maximum likelihood estimate of the parameter vector is given by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \sum_{i=1}^{N} \left(\log \frac{1}{Z_G} - \frac{1}{2} \left(\frac{e_{\mathcal{M}}(\boldsymbol{d}_i;\theta)}{\sigma_\eta} \right)^2 \right) = \arg\min_{\theta} \sum_{i=1}^{N} \frac{1}{2} \left(\frac{e_{\mathcal{M}}(\boldsymbol{d}_i;\theta)}{\sigma_\eta} \right)^2 \tag{4.5}$$

That represents the well known least square estimator.

4.1.1 Outliers, Bias and Breakdown Point

Assuming that there exists a true model (with a set of true parameters) of the observed data, an **outlier** its considered to be any datum failing a statistical hypothesis test such as Chi-square. Outliers are commonly comprised by noise side effects. Therefore inliers are considered all the subsets that are not outliers. There is another parameter named **bias** with a huge significance while studying the behavior of the estimation.

Let $D_I \subset D$ be a set of inliers and let $D_{I|O}(m)$ be the previous set after m inliers that have been replaced by random outliers. Bias measures the maximum perturbation that can be found using only inliers and then replacing some of them by outliers. It is defined as

$$bias_{\mathcal{M}}(m; D_I) \equiv \sup_{D_{I|O}(m)} \Omega \left[\boldsymbol{\theta}(D_I), \boldsymbol{\theta}(D_{I|O}(m))\right]_{\mathcal{M}} \tag{4.6}$$

where Ω function measures the distance between parameter vectors for each subset and any $p-$norm based distance can be used. Finally, **breakdown point** $\mathcal{BP}$ is defined as the minimum ratio of outliers that can increase the bias parameter consistently. That is,

$$\mathcal{BP}_{\mathcal{M}}(D_I) \equiv \min \left\{ \frac{m}{|D_I|} : bias_{\mathcal{M}}(m; D_I) = \infty \right\} \tag{4.7}$$

4.1.2 Minimal Sample Set

Minimal Sample Set (MSS) is considered as the minimum number of datum elements that permits to determine a parameter vector entirely. For instance, if the desired parametric model is a line, two points are required in order to identify the line uniquely. Likewise, if the model is a plane, three points are required.

4.1.3 Random Sample and Consensus

As stated during the previous section, the aim of this part of the process is to determine the location of any supporting plane that might exist inside the point cloud. Facing the problem that there might exist a large quantity of noise inside the data source (any object lying on the plane, artifacts due to shiny areas, any inconsistent point captured through the sensor, etc.), it is necessary to use a robust estimator ready to handle outliers even larger than 50% of the complete dataset.

RANSAC algorithm (RAN*dom* S*ample* A*nd* C*onsensus*) has been demonstrated to be a powerful tool in terms of robustness, convergence speed and stability. The following explanation aims to illustrate the base of this method and how to put it into practice to solve the supporting plane statement. RANSAC comprises two steps that are *iteratively* repeated:

- *Hypothesize.*- First MSSs are randomly selected. Parameter vector is computed using exclusively the selected elements of the MSS.

- *Test.*- Check how good each element fits in the entire dataset with the parameter vector generated during the previous step. All those elements consistent with the model take part of the **consensus set** (CS).

The more elements gathered in the CS, the better representation of the parameter vector is achieved. The algorithm stops when the probability of finding a better CS is below a threshold value.

Terminology and Preliminaries

Table 4.1 explains the nomenclature used during the rest of the section.

Symbol	Description
superscript $^{(h)}$	h^{th} iteration
$\hat{x}$	estimated value of the quantity x
N	number of elements of input dataset
$D = \{\boldsymbol{d}_1, ..., \boldsymbol{d}_N\}$	input data set
S	indicate a MSS
$\boldsymbol{\theta}\left(\{\boldsymbol{d}_1, ..., \boldsymbol{d}_h\}\right)$	parameter vector using the subset $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_h\}$
$f_{\mathcal{M}}$	smooth function whose zero level set contains all the points that fit the model $\mathcal{M}$ based on $\boldsymbol{\theta}$

Table 4.1: Terminology and preliminaries for RANSAC algorithm

The model space $\mathcal{M}$ (see Fig. 4.3) represents all those points that fulfill $f_{\mathcal{M}}$

$$\mathcal{M}(\boldsymbol{\theta}) \equiv \{\boldsymbol{d} \in \mathcal{R}^d : f_{\mathcal{M}}(\boldsymbol{d}; \boldsymbol{\theta}) = 0\} \tag{4.8}$$

In order to determine a CS, it is necessary to define an error function that measures the consistency of the datum on the proposed model space. Distance between datum $\boldsymbol{d}$ and model space $\mathcal{M}(\boldsymbol{\theta})$ is

$$e_{\mathcal{M}}(\boldsymbol{d}; \boldsymbol{\theta}) \equiv \min_{\boldsymbol{d}' \in \mathcal{M}(\boldsymbol{\theta})} \Omega(\boldsymbol{d}, \boldsymbol{d}') \tag{4.9}$$

being $\Omega(\cdot, \cdot)$ a suitable distance function. Now a CS can be defined as

$$S(\boldsymbol{\theta}) \equiv \{\boldsymbol{d} \in D : e_{\mathcal{M}}(\boldsymbol{d}; \boldsymbol{\theta}) \leq \delta\} \tag{4.10}$$

where δ is a threshold value that has to be set specifically on each problem.

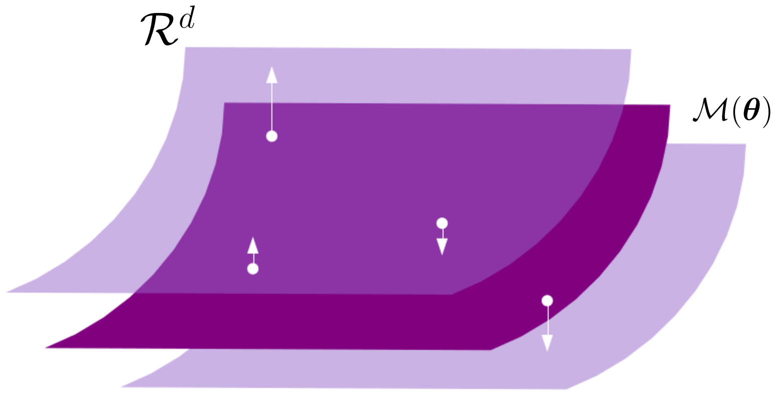


Figure 4.3: *RANSAC theoretical aspects.* Representation of the model space $\mathcal{M}$ in dark magenta. Light magenta surfaces represent the boundaries for the inliers. All datum between those surfaces are considered part of S. White dots represent the inliers and their arrows measure their relative distance to the model space.

Procedure

A MSS $s^{(h)}$ is randomly selected from D. Secondly, the parameter vector $\boldsymbol{\theta}^{(h)}$ is computed using uniquely the MSS datum. Thirdly, RANSAC measures the consistency of the complete dataset D based on $\boldsymbol{\theta}^(h)$ and, if succeeds, it updates the current best CS referenced as S^*. If the probability of finding a better CS turns below a threshold, the algorithm finalizes. There exists a termination criterion based on the maximum number of iterations for the algorithm to recognize S^*.

Determining the number of iterations

The best estimation of the model parameters will be generated picking up a MSS s containing solely inliers. The probability of achieving this situation is q. Therefore, there is a probability $1 - q$ of picking at least one outlier. The more number of iterations h, the less probability of obtaining all MSS contaminated by at least one outlier $(1 - q)^h$. h must be chosen so that the probability $(1 - q)^h$ is lower than a certain value ϵ.

$$(1-q)^h \leq \epsilon \Longrightarrow h \geq \left[\frac{\log \epsilon}{\log\left(1-q\right)}\right] \tag{4.11a}$$

$$\hat{T}_{iter} = \left[\frac{\log \epsilon}{\log(1-q)}\right] \tag{4.11b}$$

Creating the MSSs

The probability of choosing a MSS composed with all the inliers (all elements with the same probability of being chosen) is defined as

$$q = \frac{\begin{pmatrix} N_I \\ k \end{pmatrix}}{\begin{pmatrix} N \\ k \end{pmatrix}} = \frac{N_I!(N-k)!}{N!(N_I-k)!} = \prod_{i=0}^{k-1} \frac{N_I - i}{N - i} \tag{4.12}$$

where k represents the *cardinality* of the MSS (the smallest sufficient to determine the parameter vector) and N_I is the total number of inliers. Because k is much lower than the number of inliers and even much lower than the total population.

$$q \approx \left(\frac{N_I}{N}\right)^k \tag{4.13}$$

This is equivalent to say that the probability of choosing the *true* MSS is approximately the same as picking consecutively k elements from the total datum population.

Experimental values

The experimental values used for plane extraction using the PCL library are stated in Table 4.2.

Parameter	Value
N	≈ 87286
δ	0.005 m
ϵ	0.99
$\hat{T}_{iter}^{MAX}$	1000

Table 4.2: Experimental values for the plane extraction

In Figure 4.4 the experimental result of the RANSAC algorithm has been represented for one plane candidate. Firstly, the color environment is displayed on the left (for understanding purposes). Lastly, the result of the algorithm in a 3D point cloud is represented on the right, where datum has been differentiate between inliers in green and outliers in blue.

However, as stated before, part of the problem comes when more than one plane is found by the algorithm as shown in Figure 4.5, where a large environment point cloud is represented and several planes can be found.

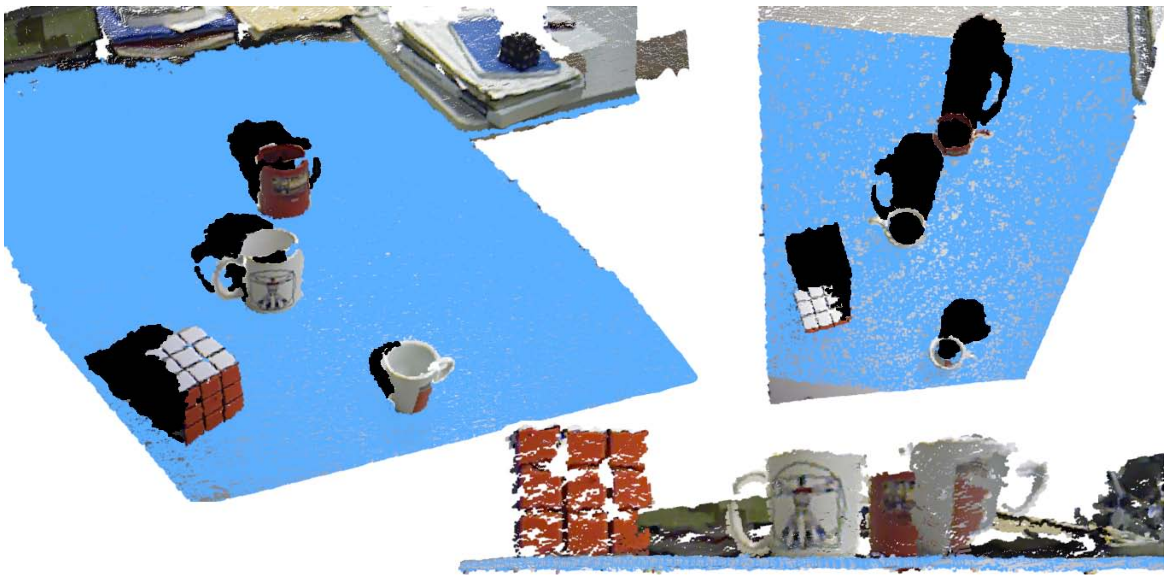


Figure 4.4: *Supporting plane estimation.* (a) Global-view of the scenario with the table colored in blue. Notice the amount of noise provided by the background items; (b) Top-view of the scenario. 3D shadows are projected in the supporting plane; (c) Front-view of the scene with some details of the objects lying on the table.

4.2 Evolutionary Plane Fitting

As a contribution for this work, a plane fitting comparison has been performed in order to measure the robustness of RANSAC based on the quantity of objects lying on the supporting table. That is, the higher the number of outliers, the larger the error must become during the plane fitting. The classic iterative method has been compared with an evolutionary algorithm based on Differential Evolution (see Appendix A for more information about DE) in order to determine if multi-modal functions are more adaptable and versatile as the the number of points grows. Considering a set of N points $D = \{d_1, d_2, ..., d_N\} \subset \mathbb{R}^3$, the estimation of the best plane (model) that fits such points is desired.

Taking Equation 4.1into account, there exist an error function that represents the consistency of the model for $\mathbb{R}^3$ such as

$$e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta}) = \frac{|\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4|}{\sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2}} \tag{4.14}$$

For this experiment, the proposed fitness function is represented in Equation

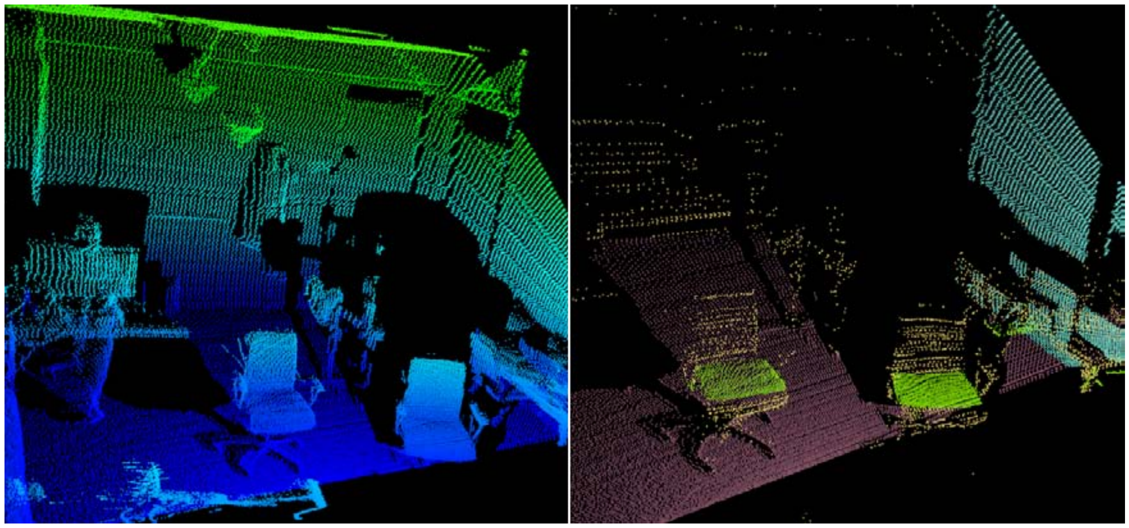


Figure 4.5: *Several planes in a large environment.* Left frame shows a dense point cloud of an office where multiple planes can be found. In right frame several planes have been found and have been represented in different colors. Not all of them are valid candidates to become a supporting plane.

4.15. It considers the distance between an evolutionary plane formed by $\boldsymbol{\theta}$ and the list of elements in the point cloud.

$$\underset{\boldsymbol{\theta}}{\arg\min} \sum_{N} e_{\mathcal{M}}(\boldsymbol{d};\boldsymbol{\theta}) \tag{4.15}$$

The stochastic search of the matching pose is done using the DE method for global optimization problems over continuous spaces. The evolutionary plane is formed by four elements on each iteration k such as

$$pop^k = (\theta_1^k, \theta_2^k, \theta_3^k, \theta_4^k) \tag{4.16}$$

where pop_i^k represents the candidate supporting plane at iteration k. The initial population might be chosen randomly at first glance because there is any fact or information of the pose of the robot and its relative position with the table. If a global map is provided based on global localization techniques such as SLAM, the initialization of the candidates would lead to a faster convergence. However, the physical constraints of the robot can reasonably determine an initial value for the population taking into account that the camera is positioned at 1.6 m. and that the supporting plane must be located at a height of 1 m approximately for the robot to reach the object. For this reason, the initial values of the algorithm could be empirically established.

The convergence of the algorithm has been established with a maximum fixed number of 100 iterations or no changes in the population member cost functions (global convergence). Other stop alternatives such as maximum error distance or threshold timing have been excluded because the error depends on several parameters that make it unfeasible to determine an adaptive factor and time may vary substantially. A few of those uncontrolled parameters are the number of objects, occupancy of those clusters in the scene or even the distance between the robot and the supporting plane.

4.2.1 Stabilization of Parameters

The first experiment evaluates the consistency and repeatability of $\boldsymbol{\theta}$ for both alternatives. This is performed calculating the plane equation in a tidy scenario where only the table is presented opposite to the robot. The errors for both estimators are represented in Figure 4.6 for a set of 100 estimations. The static error has been computed as the error distribution of the fourth parameter of the plane estimator θ_4 that represents the euclidean distance between the table and the pose of the sensor (camera height). The difference between the estimated parameter and the experimental value $\hat{\theta_4}$ has been calculated

$$\xi = 100 \cdot \frac{|\hat{\theta_4} - \theta_4|}{\hat{\theta_4}} \, (\%) \tag{4.17}$$

Both estimator errors ξ_{RANSAC} and ξ_{DE} behave as two Gaussian distributions with the following parameters:

$$\begin{aligned} \xi_{RANSAC}(\%) &= \mathcal{N}(3.62722, 0.09262) \\ \xi_{DE}(\%) &= \mathcal{N}(3.35306, 0.3261) \end{aligned} \tag{4.18}$$

Thus, the standard deviation of the evolutionary estimator is wider than ξ_{RANSAC} but the average error is smaller. That is, DE offers a better performance execution with a small average error nonetheless with a higher variation degree. RANSAC error range moves in the range [3.53459, 3.71984], while DE moves in the range [3.20458, 3.85678], meaning that probabilistically the evolutionary estimator has a better performance than RANSAC in average though more diffused.

Futhermore, the importance of the algorithm does not only resides on this stability but on the global estimation for cluttered scenarios. A large portion of this

4% of static error can be attached to the physical sensing device, lighting conditions and surface material.

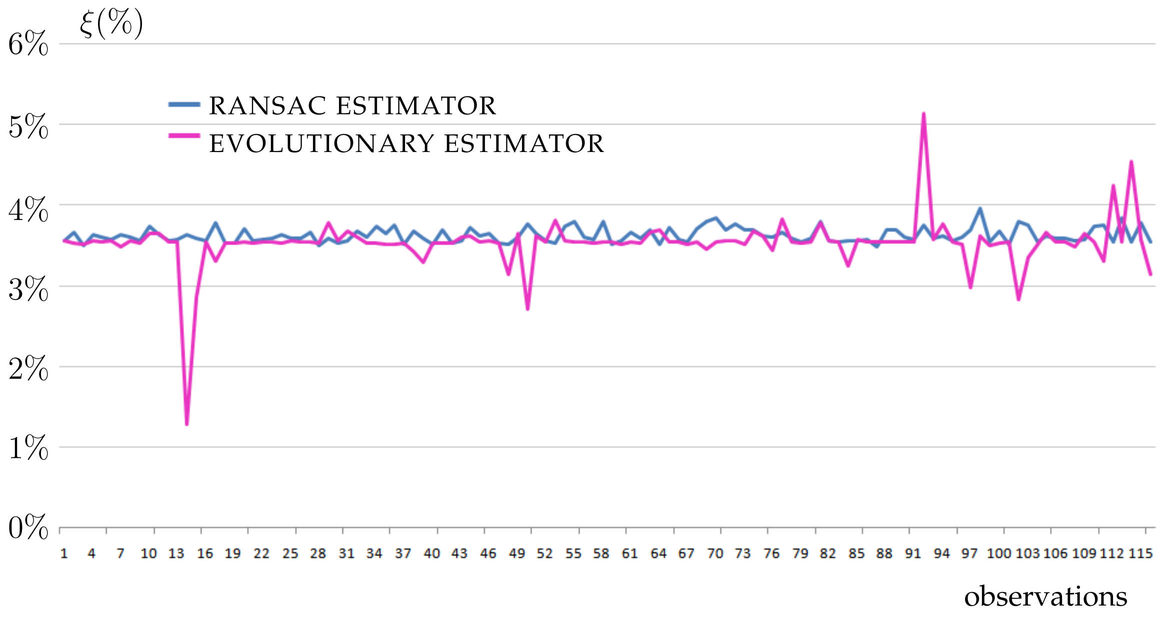


Figure 4.6: *Error comparison.* Comparison of relative errors for RANSAC and the proposed evolutionary plane estimator for an experiment with 100 measures.

4.2.2 Estimator Robustness

The goal is to compare both plane fitting strategies in terms of robustness against noise and time to extract the plane. Thus, a set of scenarios are prepared in the laboratory where on each case, the number of objects lying on the table is increased. Each scene contains a different portion of inliers/outliers that will be increased until obtaining a full cluttered scenario as shown in Figure 4.7. Given the equation of the supporting plane without elements on it, it becomes straightforward to measure the deviation of the parametric equation parameters for both models and determine the robustness against objects shadowing the supporting plane.

Figure 4.8 explains the behavior of RANSAC and evolutionary plane fitters with the increase of objects laying in the scenario. As can be noticed, the abruptly changes of RANSAC estimation are exaggerated after observation 137 in the experiment. Contrary to that, the DE based algorithm behaves steady, stable and nonfluctuating. This happens because the proper nature of RANSAC is based on

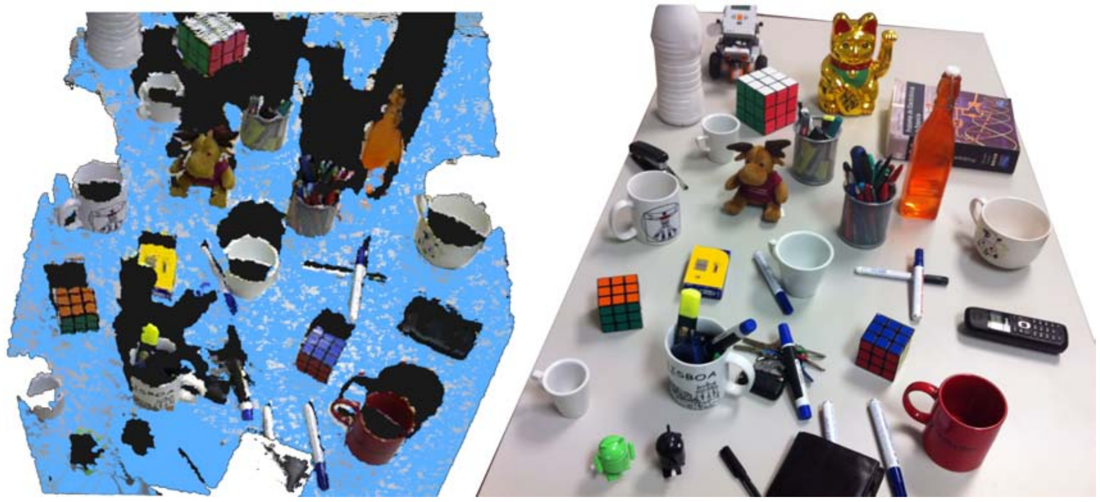


Figure 4.7: *Example of a cluttered scenario*. The number of outliers treble the number of points forming the supporting plane. Left image highlight in blue the plane equation in the 3D point cloud while left image shows the real scenario viewed from the color camera.

an iterative process with random selection, while the genetic algorithm improves the inlier selection on each iteration. The last observation corresponds to the worst scenario evaluated and it is displayed in Figure 4.7 for a better understanding purposes. Using the evolutionary estimator it is not necessary to track the supporting plane with solutions such as Kalman or Particle Filters [45] for these types of cluttered scenarios.

However, to get a better idea of what is happening internally in both estimators, it is necessary to track not only the error but the number of inliers processed on each observation. Table 4.3 contains the parameters of the fitting planes for both estimators at several points of the experiment. Figure 4.9 contains the number of inliers with respect to the ouliers for each observation. As can be noticed, as long as the error increases, the number of estimated inliers decreases instantly and vice-versa. The experiment has been performed with a total number of 164761 points. However, this number may vary as the point cloud is filtered and spurious measures are removed.

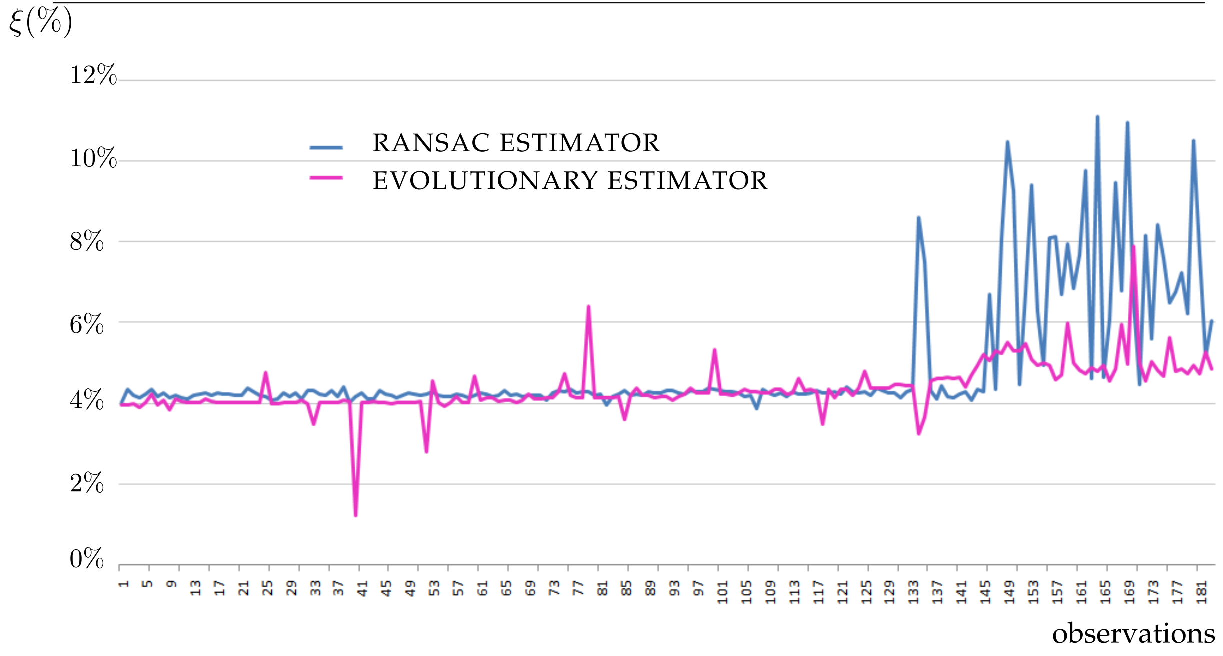


Figure 4.8: *Comparison of relative errors.* Comparative between RANSAC and the proposed evolutionary plane estimator for an scenario where a large list of objects are positioned on the table inducing to misalign the original plane equation.

It is important to highlight that both estimators extract almost the same proportion of inliers during the experiments until a certain moment when the RANSAC estimator falls down. This happens when the number of inliers sums below 72%. After that happens, the estimations start failing and being unstable. Contrary to that, the evolutionary plane fitting estimator reduces the inliers gradually, becoming more stable and steady.

4.2.3 Time Analysis

As it has been stated in the literature multiple times, one of the most significant drawbacks of *Genetic Algorithms* (GA), and more precisely in *Evolutionary Algorithms* (EA), is the large amount of time that those algorithms require to converge into a reasonable solution. This happens because of the proper nature of the evolution, as the algorithm needs to improve previous records in parallel, which requires a high computational cost.

The elapsed time for both estimators is quite dissimilar. RANSAC spends as much as 49,409 ms per observation, while the evolutionary estimator requires

PLANE FITTING PARAMETERS ESTIMATION							
ICP parameters				Evolutionary parameters			
A	B	C	D	A	B	C	D
0.0670087	-0.781455	-0.620353	0.674016	0.0639165	-0.781623	-0.620468	0.673949
0.0802332	-0.778951	-0.621930	0.675195	0.0716981	-0.780098	-0.621536	0.674658
0.0835140	-0.778814	-0.621670	0.675022	0.0774038	-0.780234	-0.620680	0.673850
0.0826786	-0.780034	-0.620251	0.673722	0.0772123	-0.780661	-0.620167	0.673975
0.0826191	-0.779979	-0.620327	0.673695	0.0776226	-0.780232	-0.620655	0.673782
0.0822349	-0.778656	-0.622038	0.675423	0.0781921	-0.784379	-0.615334	0.668163
0.0826251	-0.779832	-0.620512	0.673967	0.0772753	-0.780413	-0.620470	0.673678
0.0830126	-0.779504	-0.620873	0.674231	0.0771343	-0.780211	-0.620742	0.673889
0.0821487	-0.779545	-0.620936	0.674250	0.0772315	-0.780186	-0.620762	0.673881
0.0832413	-0.778810	-0.621712	0.674860	0.0774314	-0.780218	-0.620696	0.673840
0.0830310	-0.778514	-0.622111	0.675265	0.0770501	-0.780217	-0.620745	0.673865
0.0825714	-0.779821	-0.620534	0.673916	0.0633838	-0.779597	-0.623066	0.671071

Table 4.3: Plane parameters for both estimators at different observations of the experiment: 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110. Most parameters varies after the third digit.

1742,495 ms to do the same job, one order of magnitude more. However, contrary to the belief that RANSAC can be the best choice as it requires less time, manipulation tasks demand good quality models as they are decisive for grasping tasks.

Figure 4.10 summarizes the problem of a bad and noisy plane estimation. One of the items of the supporting plane is analyzed at several observations during the experiment applying the plane parameters given by RANSAC. As can be seen, as far as the number of objects increases, the bottom part of the object is avoided reducing the total number of faces of the mesh up to 8.95%. However, this number can be easily increased if the object has a strong horizontal arrangement such as boxes or plates and more perceptible in small objects as pens or markers. This value falls down to 2.47% if the evolutionary plane estimator is adopted.

4.3 Geometric and Outlier Filtering

The ultimate goal of the process is not only to extract the plane model $\mathcal{M}$ but to ensure that it corresponds to a valid supporting plane, meaning that $\boldsymbol{\theta}$ represents a horizontal orientation plane. This can be done with the dot product of the normal associated to the plane $\overrightarrow{n_\theta}'$ and the $y-$axis $\overrightarrow{n_y}$.

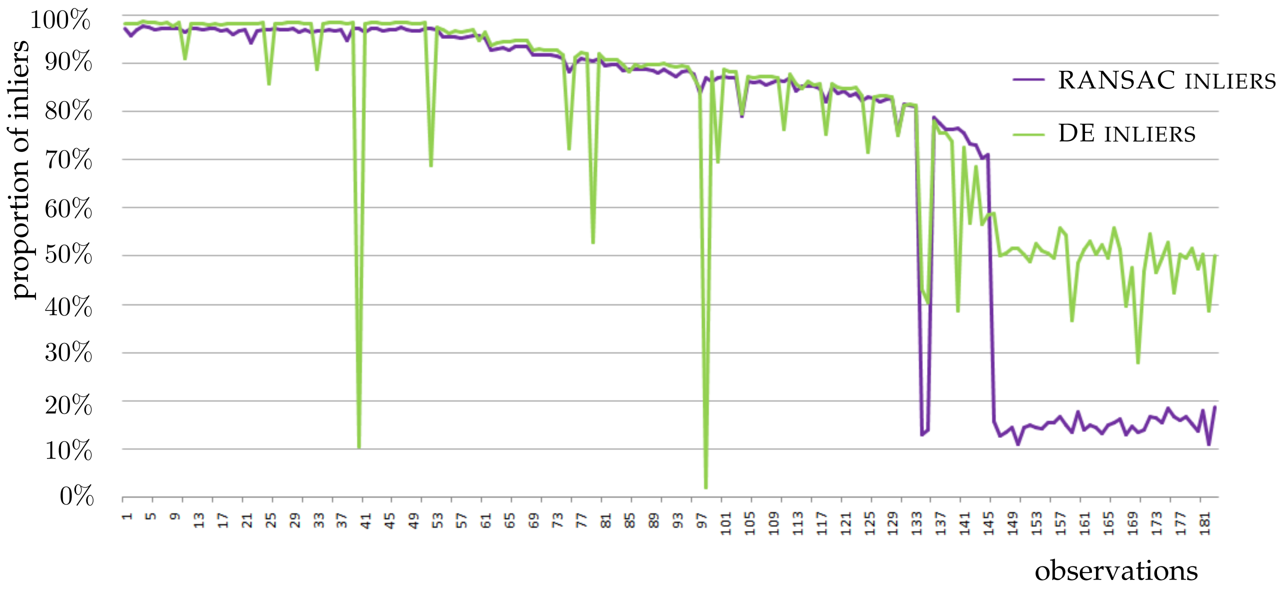


Figure 4.9: *Comparison of relative errors.* Comparison of relative errors for RANSAC and the proposed evolutionary plane estimator for a scenario where a large list of objects are positioned on the table inducing to misalign the original plane equation.

Notice that before applying this condition, it is required to take into account the view-point transformation. Camera global rotation and translation T_{cam} have to be taken into account by projecting the point cloud using the parametric model generated beforehand

$$T_{cam} = (R_{cam}|t_{cam}) \tag{4.19}$$

so $\overrightarrow{n_\theta} = T_{cam} \cdot \overrightarrow{n_\theta}'$. Therefore, any plane can be accepted as a supporting plane if

$$|\overrightarrow{n_\theta} \cdot \overrightarrow{n_y}| = \frac{|\theta_2|}{\sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2}} \leq \tau_y \tag{4.20}$$

Values of τ_y can vary from 0.8 to 0.95 for most of the cases. If the plane model does not reach the orientation condition, the plane is rejected.

4.3.1 Selecting the Most Dense Plane

Even after filtering those candidates which are not horizontal, it is possible to deal with more than one candidate. There must exist only one principal supporting plane. To do that, planes are sorted by number of inliers and the most dense one is chosen only if the ratio between its number of inliers in comparison with the

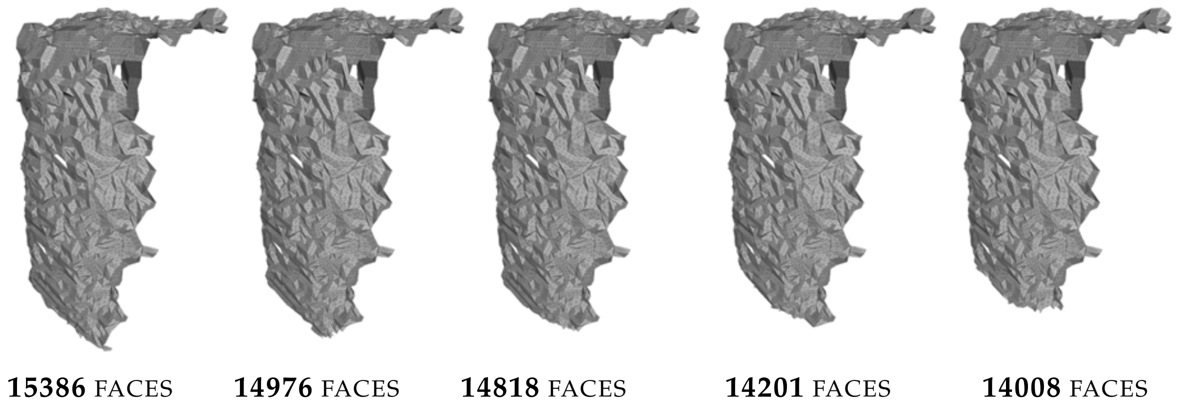


Figure 4.10: *Errors due to wrong plane fitting.* Comparison of the same object extracted using different plane estimations. The amount of information relative to the object is reduced with the addition of noise elements into the depth map for RANSAC plane fitting.

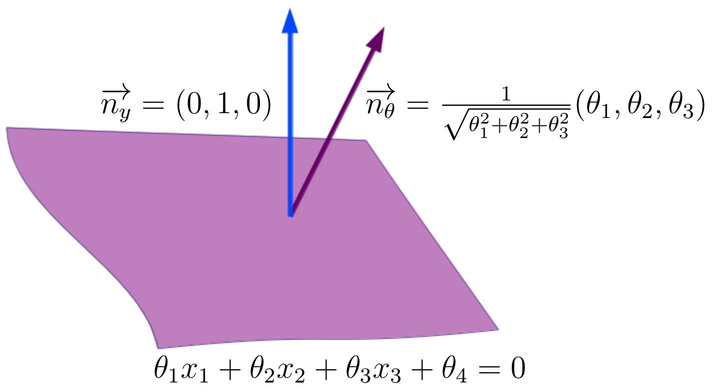

$\overrightarrow{n_y} = (0,1,0)$
$\overrightarrow{n_\theta} = \frac{1}{\sqrt{\theta_1^2+\theta_2^2+\theta_3^2}}(\theta_1,\theta_2,\theta_3)$
$\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 = 0$

Figure 4.11: *Orientation filtering.* Result after applying an orientation filter in order to select a valid supporting plane

number of inliers of the following candidate is larger than a certain value φ. So, if the list of p candidates is $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_P\}$, where P is the number of candidates and they are sorted by $N_I(p)$, it yields

$$\boldsymbol{\theta}^{sel} = \begin{cases} \boldsymbol{\theta}_1 & \text{if} \quad N_I(1) > \varphi \cdot N_I(2) \\ null & \text{if} \quad N_I(1) \leq \varphi \cdot N_I(2) \end{cases} \tag{4.21}$$

Threshold parameter φ can be discussed further, but a fixed value between 1.6 and 2.5 is enough for most of the cases (meaning that the first candidate contains between 40% to 60% more inliers than the second one). This decision makes sense assuming that the robot is adjacent to the workspace. That is, the principal supporting plane becomes $\boldsymbol{\theta}^{sel}$.

Chapter 5

Clustering point clouds

Once the supporting plane is detected, next step to be carried out is to find datum near to the surface and classify them into independent clusters. This procedure must be performed in an unsupervised way and the result must be a list of subsets that could represent objects that will need to be recognized afterwards. There exist multiple techniques to segment clusters: based uniquely on relative distance between datum, gather them based on normals, training with predefined models [46], mixing color and texture with geometric constraints [47], using graph-based [48] or even based on parametric models [49]. Clustering techniques are quite interesting to reduce computation times if patterns have to be found or point clouds have to be analyzed simply because they segment interesting datum from background (or uninteresting) information.

5.1 *k*-means distance clustering

The k-means algorithm is one of the most famous unsupervised clustering algorithms. The procedure follows a simple strategy to classify a given data set of n observations $(p_1, p_2, ..., p_n)$ through a fixed number of k clusters $\boldsymbol{C} = \{C_1, C_2, ..., C_k\}$. The main idea is to define k centroids c_k, one for each cluster. These centroids should be placed wisely since varying its initial location may cause unexpected results. Afterwards, each single observation is associated to the nearest centroid using a custom distance function. When every point is classified, an iteration is performed. At this point, each c_k is recomputed and updated based on the actual clustering. If this process is repeated a sufficient number of times, centroids might converge to a stable position. The formulation for such minimization is called within-cluster sum of squares (WCSS) and it is defined as

$$\arg\min_{\boldsymbol{C}} \sum_{i=1}^{k} \sum_{p_j \in C_i} \|p_j - c_i\|^2 \tag{5.1}$$

where a norm L-2 distance measurement between any single datum and the cluster center is considered. The summation gathers the distance of the N data points from their respective cluster centers. The algorithm is stochastic and is composed of four steps:

1. *Initialization.*- Place k points into the space represented by the objects that are being clustered. These points represent initial group centroids.

2. *Assignment.*- Assign each object to the group that has the closest centroid.

3. *Recalculation.*- When all objects have been assigned, recalculate the positions of the k centroids.

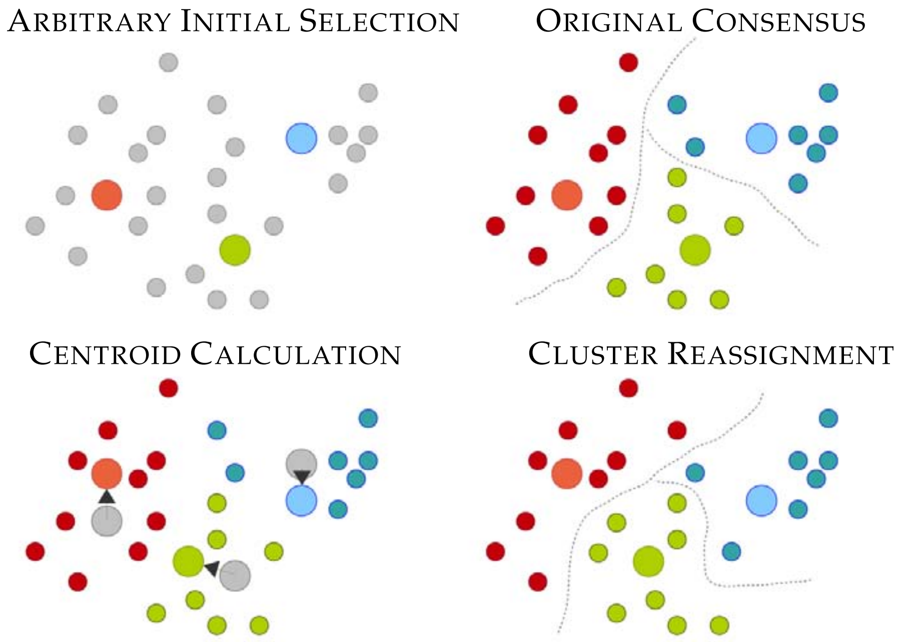


Figure 5.1: *k-means algorithm.* Graphical explanation of *k*-means algorithm and the evolution of three clusters. Last two steps are repeated until centroids converge to a stable position

4. *Convergence.*- Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although this solution will always converge, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers as the initial centroids are stochastically selected, giving different solutions at each run. The k-means algorithm can be run multiple times to reduce this effect. Nevertheless, Yuan *et al.* [50] and [51] introduces an improvement in the initial centroids assignment by evaluating the distances between every pair of data-points and then trying to gather those similar pairs. Figure 5.1 represents the evolution of the algorithm and the results on each iteration. JianYu [52] compares the k-means method with Otsu image thresholding method as they are both based on a same criterion minimizing the within-class variance.

In terms of speed, Na *et al.* [53] proposes a summation buffer that reduces the number of times calculating the distance between each data element and all cluster centers in each iteration reducing one half the execution time. Others such as Hong-tao *et al.* [54] designed a GPU (Graphics Processing Unit) alternative of the algorithm that parallelizes the full process using CUDA, obtaining 40 times of the CPU-based version.

5.2 Estimating Surface Normals and Curvature

Surface normals are very useful to understand the geometry of the surface and also to reconstruct and understand the point cloud. Normal extraction means estimating the normal of a plane tangent to the surface. This process can be as simply as computing the cross product between a pair of nearest points for any query point. This technique is not recommended for noisy point clouds because results are inconsistent and extremely variable [55].

Literature [56] [57] recommends extracting normals as a least-square plane fitting estimation problem. With the analysis of the eigenvectors and eigenvalues of a covariance matrix C created from the nearest neighbors to the query point, it is possible to determine the surface normal

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T \tag{5.2}$$

where

$$C \cdot \vec{v_j} = \lambda_j \cdot \vec{v_j},\ j \in \{0, 1, 2\} \tag{5.3}$$

where λ_j represents the $j - th$ eigenvalue and $\vec{v_j}$ is the $j - th$ eigenvector of C. Principal Component Analysis (PCA) is intended to find the directions in the data with the most variation, *i.e.* the eigenvectors corresponding to the largest eigenvalues of the covariance matrix, and project the data onto these directions discarding important non-second order information provided by the covariance matrix. If eigenvectors are sorted such that $\lambda_0 \geq \lambda_1 \geq \lambda_2$, then the primary surface direction or plane normal might be assumed to be given by the third eigenvector

$$\vec{n_i} = \pm\vec{n_3} \tag{5.4}$$

Obviously the best estimation is yielded when the ratio $\lambda_0 \gg \lambda_1$. Therefore, the problem can be solved in five steps:

1. Construct a projection matrix for the tangent plane given by the normal of the query point.

2. Project all normals from a k-neighborhood (surface patch) onto the tangent plane.
3. Compute a centroid in that projected space, and a covariance matrix.
4. Perform eigen decomposition to obtain the principal directions.
5. Choose the largest eigenvalue and select its corresponding eigenvector as plane normal.

Normals orientation is solved applying the viewpoint constraint given by

$$\vec{n_i} \cdot (v_p - p_i) > 0 \tag{5.5}$$

where v_p is the global position of the viewpoint or sensor coordinates.

An example of this algorithm is displayed in Figure 5.2 where Equation 5.4 produces normal directions to point accordingly to the view point. The map representing the normals of each of the points of the point cloud $\mathcal{P}$ is represented as $\Xi(\mathcal{M})$.

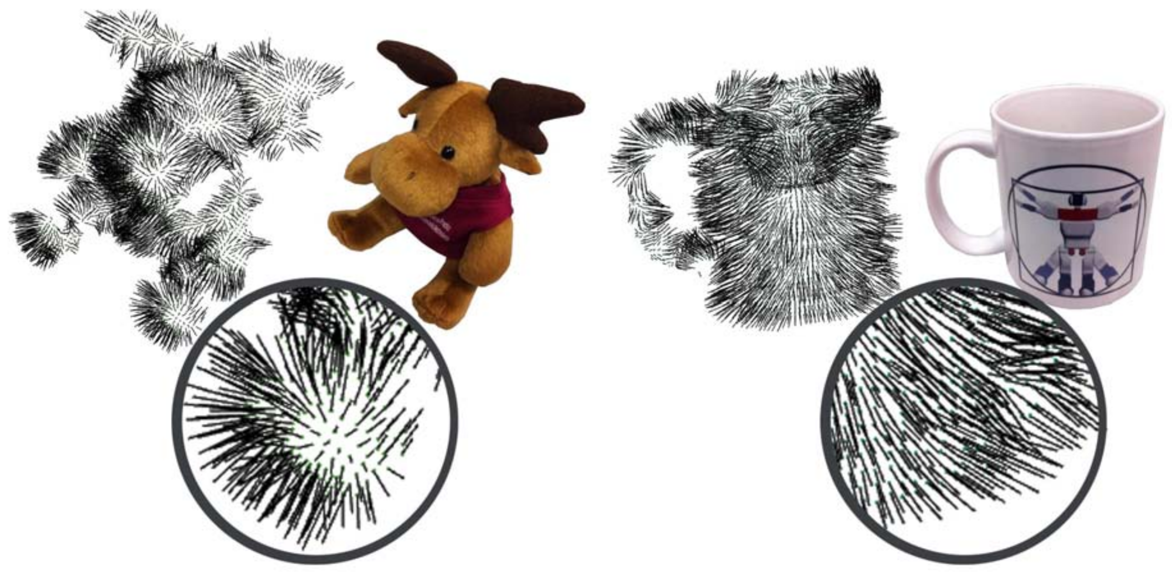


Figure 5.2: *Surface normals estimation*. Normal direction is computed taking into account the view point coordinates for each point. Circles contain a zoomed area of the objects for a better visualization.

5.3 *k*-dimensional trees

Dealing with 3D point clouds is a hard challenge from the computational point of view. 3D cameras work at 30 frames per second with resolution of 640×480 pixels per frame and with XYZ plus RGB information enclosed for each individual point. That makes a sum of 55,296.000 values processed every second by the perception system.

There exist several alternatives to reduce point clouds such as spatial partitioning, and search operations with Octrees or compressing the point cloud data. All those methods concern about reducing the amount of memory resources. The most extensively adopted procedure for optimal searching of group of points with common attributes within a point cloud was presented by Freidman *et.al* [58] and is based on the use of trees.

A tree is a widely-used data structure that imitates a hierarchical tree structure with a group of linked nodes. In mathematical terms, a tree is a synonym of an *arborescence* or tree view: an acyclic connected graph where each node has zero or more children nodes and parent node on the top. Additionally, the children of each node have a specific order that fits together with the search condition imposed for the tree. All the implicit searches carried out in this work are performed using *k-d* trees diminishing the execution times largely.

A *k-d* tree is explained as a binary tree in which every node represents a k-dimensional point where nodes are grouped into two classes: leaf or non-leaf elements. Every non-leaf element creates a new hyperplane splitting the space into two subspaces. Left subtree contains all the points whose location is on the left part of the hyperplane attending to the dimension that has been chosen (3D coordinates, RGB values, curvature, normal direction, etc.).

For this explanation Cartesian coordinates will be established so that each hyperplane attached perpendicularly to the tree generates a new subtree. As an example, if $p_n\left(x_n, y_n, z_n\right)$ is non-leaf node associated with the $y-axis$ in a set of points $p_i\left(x_i, y_i, z_i\right)$, the condition for the division is determined as

$$p_n\left(x_n, y_n, z_n\right)\left\{\begin{array}{l} left : \forall p_i \ni \left(y_i < y_n\right) \\ right : \forall p_i \ni \left(y_i \geq y_n\right) \end{array}\right\} \tag{5.6}$$

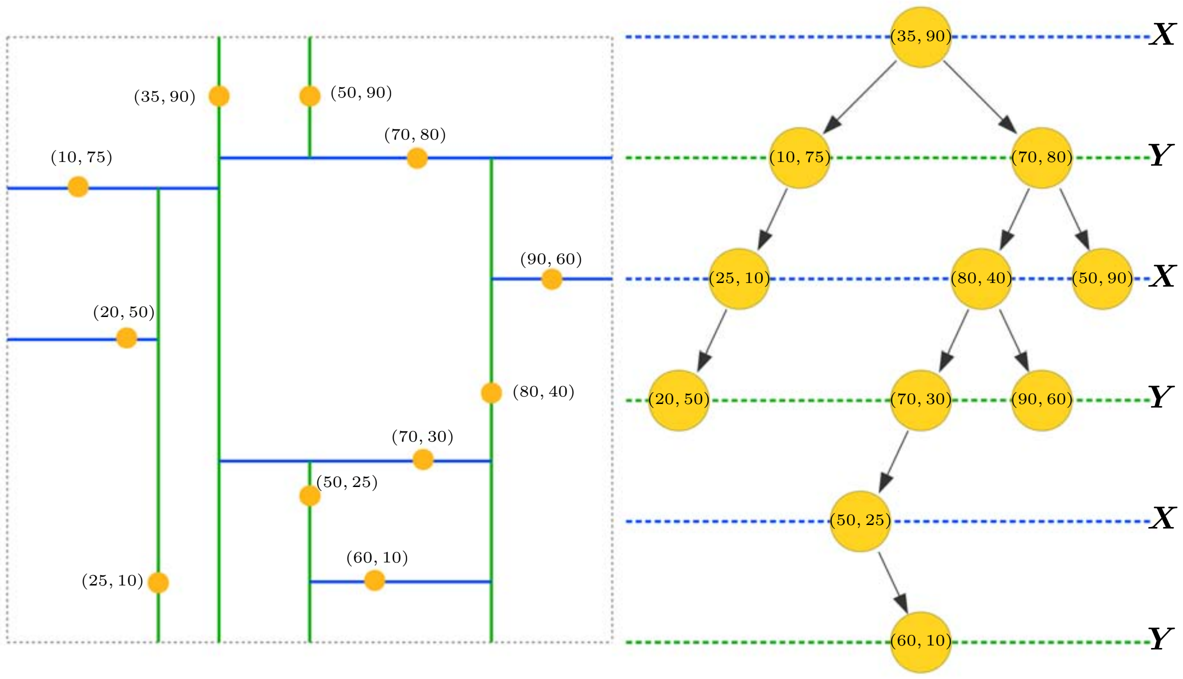


Figure 5.3: *2D k-d tree*. Representation of the tree as a hyperplanes in 2D space on the left. Graph view on the right where blue color represents x split axis, while green depicts y split axis.

There exist multiple ways of cutting the tree. However, as a general rule, median according to the axis of the hyperplane is used to create new subspaces. Splitting planes are cyclically shifted, so each level of the three represents one dimension (*e.g.* in a 3D *k-d* tree the first dimension is x, then in the next division turns to y, afterwards z and finally cutting axis accomplishes again into x.) An example of a 2D *k-d* tree is shown in Figure 5.3, where blue color represents x axis while green depicts y axis.

There is a time complexity associated to a *k-d* tree of $O(n \log^2 n)$ for a point cloud set of n points if a $O(n \log n)$ sort algorithm is used to compute the median. There exist alternatives to reduce complexity such as Quicksort [59] or the one proposed by Thomas *et al.* [60] which diminishes the time complexity to $O(n \log n)$.

Another example of *k-d* tree is shown in Figure 5.4 where a 3D tree space has been plotted. Each subregion is determined by a hyperplane of a different color. The cell is divided on 8 subregions.

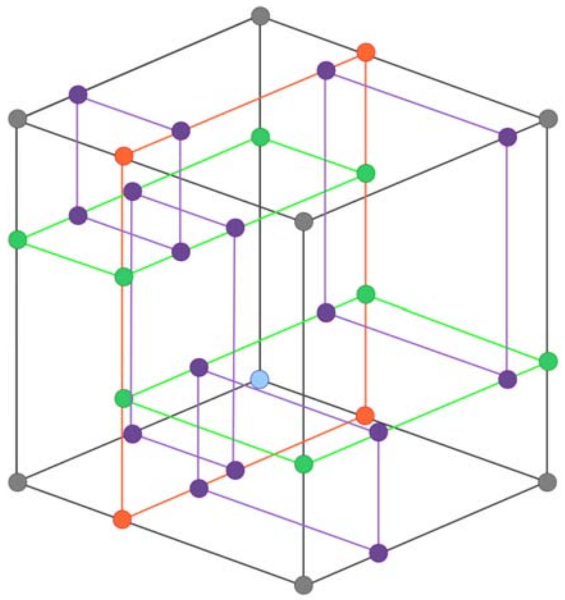


Figure 5.4: *3D k-d tree*. Original cell (gray) is splitted into two subregions (orange). Afterwards both cells are divided into four subcells (green). Lastly, eight different regions are formed in the last split (purple).

5.3.1 *k*NN searchs

As mentioned before, the very first intention of *k-d* trees is to boost the speed for searching in a large number of point clouds. For this type of searches, an interesting election Nearest Neighbors (NN). Whose aim is to find the set of N points closer to a custom query point. The speed up happens because the tree rejects in each level a large amount of candidates, reducing the search space and therefore making the process extremely fast.

However, the original proposal [58] works adequately for exact nearest neighbor search in low-dimensional data but quickly loses its effectiveness as dimensionality increases. This effect is named *curse of dimensionality* or Hughes effect, as discussed by [61]. Arya *et al.* [62] proposed an approximate matching named ϵ*-approximate* NN that imposes a fixed bound on the accuracy. Therefore, a point $p \in X$ is an ϵ*-approximate* nearest neighbor of a query point $q \in X$, if

$$d(p, q) \leq (1 + \epsilon) \cdot d(p^*, q) \tag{5.7}$$

where p^* is the true nearest neighbor and $d(x, y)$ represents the distance based on any norm. This improves the search speed at the cost of the algorithm not always returning the exact nearest neighbors.

Other improvements based on approximate NN have been proposed since then, *Fast Approximate Nearest Neighbors* (FLANN) with Automatic Algorithm Configuration, presented by Muja and Lowe [58], is nowadays the most commonly used. FLANN is a method that reduces the classical *k*NN searching process one order of magnitude by applying priority search on hierarchical k-means trees and selecting the fastest approximate nearest-neighbor algorithm for a given set of data. That is, FLANN chooses the best candidate from two approximate nearest neighbor algorithms:

- *Randomized kd-tree algorithm.-* The original *k-d* tree algorithm splits the data into two halves at each level of the tree in the dimension for which the data exhibits the greatest variation. Randomized *k-d* trees are built by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance.
- *Hierarchical k-means tree algorithm.-* Splits the data points at each level into K distinct regions using a k-means clustering, and then applying the same method recursively to the points in each region. It stops when the region is smaller than K.

To perform this selection, FLANN automatically analyzes a subset of the dataset (as a general rule a tenth of the whole population is randomly chosen) and looks for correlations between features. With this information the algorithm optimizes the parameters required for both alternatives: number of randomized trees to use in the case of kd-trees and both the branching factor and the number of iterations in the case of the hierarchical k-means tree.

5.3.2 Octrees

An octree is a tree-based system for managing distributed 3D point clouds proposed for the first time by Meagher [63]. Octrees subdivide the space into eight cubes called *octants* recursively. The root node describes a cubic bounding box which encapsulates all points. At every tree level, this space becomes subdivided by a factor of two, which results in an increased voxel resolution. Octrees are able to deal with 3D point clouds with a high performance because they permit to analyze the occupancy of a region of the point cloud. Octants are internally linked, permitting custom searches such as Neighbors within Voxel Search, k-Nearest Neighbor Search or Neighbors within *Radius* Search.

Figure 5.5 represents the octrees for a mug at different depths. As long as the resolution is increased, the size of the voxels is reduced. Note that the algorithm prioritizes the divisions on those zones where the density is higher, focusing on those zones that contain more information. As illustrated in Figure 5.5, the number of voxels can even reach the total number of elements in the point cloud as an extreme instance.

5.4 Clustering using Normals and Region Growing

This method is based on point-wise normals and the distance between the individuals. From any input data $P = \{\boldsymbol{d}_1, ..., \boldsymbol{d}_N\}$, normals are computed associated to each member $N = \{n_1, n_2, ..., n_\}$ and their curvature.

Points are gathered into groups based on their normals and curvature with the definition of a custom threshold that establishes that two clusters are different. As a flood method, the first decision is where is the optimal starting point. A good initialization for these seeds are those zones where points have the slightest variations of curvature. This strategy demands an initial classification of points based on their relative curvature but in return the number of clusters is smaller and more consistent. The algorithm steps are:

1. Sort datum based on curvature. Pick up the member with minimum curvature value (called initial seed) and find out its neighbors.
2. For each neighbor measure the angle between its normal and the initial seed. If it is below a certain threshold γ_{thr}, add the neighbor to the actual cluster.
3. If neighbor, not only fulfills the normal condition but also has a curvature below a certain threshold ϕ_{thr}, add the neighbor to the seed list.
4. Remove seed and loop until all datum has been labeled.

5.5 Euclidean Cluster Extraction

Taking advantage of octrees representation, it is straightforward to classify the workspace in occupied and unoccupied voxels. That is, Euclidean Cluster Extraction (ECE) is based on occupancy grids. As a flood-fill algorithm, ECE tries to gather groups of occupied voxels, examining if they are connected or not, as explained in Algorithm 1. ECE is unsupervised but, contrary to the k-means algorithm, the number of clusters is not an input requirement.

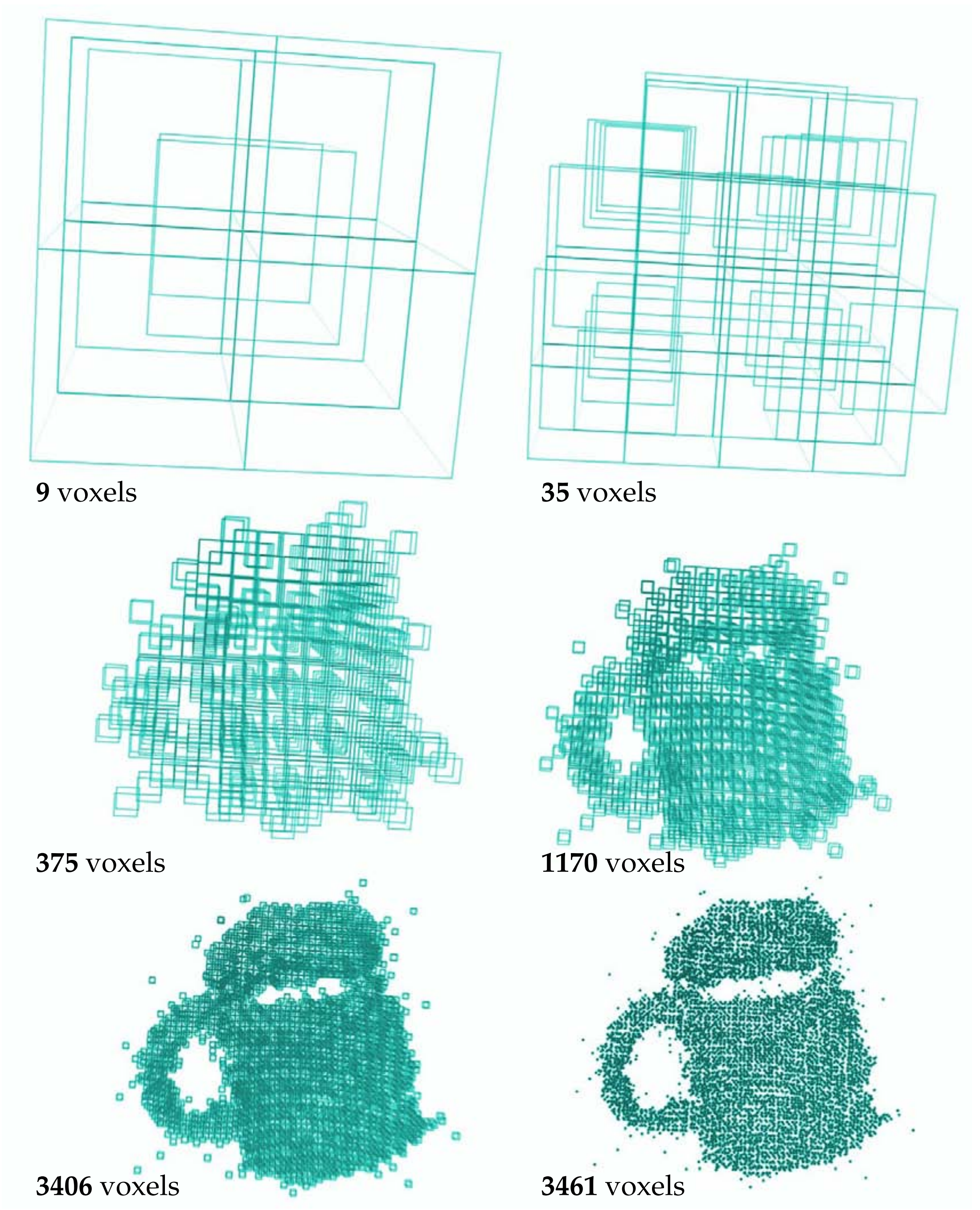


Figure 5.5: *Octree representation*. Incrementing depth of the octree (14 levels) involves better resolution. From top to bottom voxel sizes are increased for a point cloud of a mug formed by 3461 points. Voxels size varies from 81.9 to 0.06 mm.

ECE requires a cluster tolerance that delimits the minimum division between clusters. This value has to be chosen wisely because a large value will connect different clusters and a small number could make an actual object to be seen as multiple clusters. A reasonable value for this radius is 2 cm for small objects segmentation. Figure 5.6.a shows the result of ECE for an office scenario. Each cluster is colored distinctively to discriminate its boundaries. Figure 5.6.b details two clusters extremely close to each other. They might be unexpectedly gathered in the same cluster if they are close enough. This problem might arise when the cluster tolerance value is disgracefully picked.

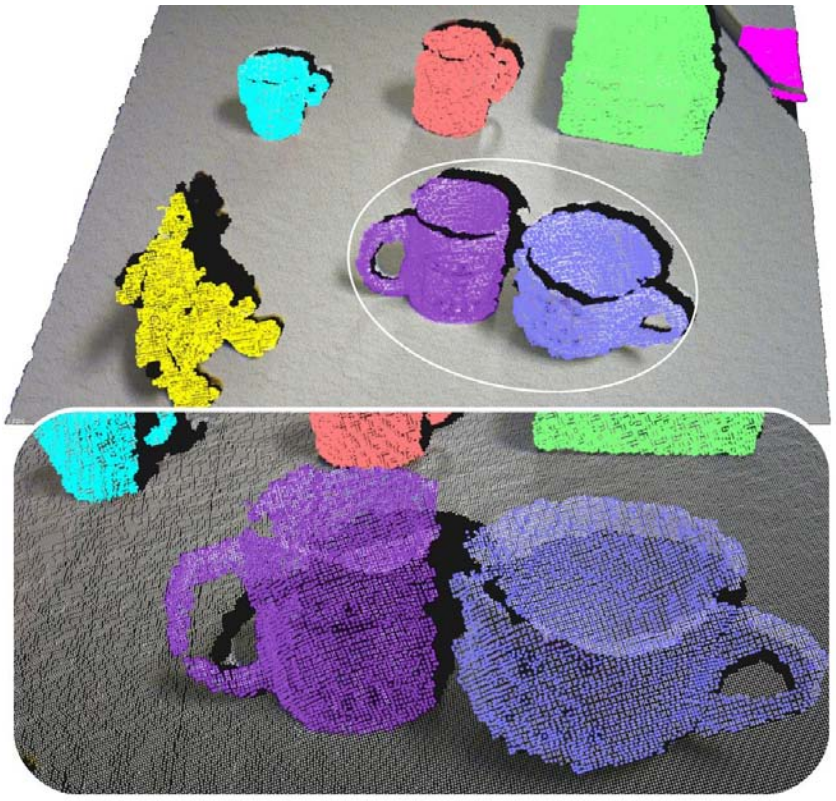


Figure 5.6: *Euclidean Cluster Extraction.* (a) Result of clustering a set of unknown objects lying on a table using ECE. Each color represents an individual cluster; (b) Detail of two clusters at the maximum cluster tolerance (2 cm). The algorithm would mislead the two mugs if a large cluster tolerance is set.

Therefore, this chapter gathers the different techniques that can be used to extract clusters from supporting planes. Once those clusters are enclosed, it is high time to analyze them and examine them to determine if they are familiar to the

Algorithm 1 Euclidean Cluster Extraction. *Computes the ECE for a point cloud P returning the resulting clusters C*

```
Require: Point cloud $P$
Require: list of clusters $C$
Require: list of points $Q$
Require: Cluster tolerance $r$
 1: $K \Leftarrow KD$ tree $(P)$
 2: $C \Leftarrow \{\}$
 3: $Q \Leftarrow \{\}$
 4: $N = $ size($P$)
 5: for $j = 1, 2, ..., N$ do
 6:     $p = P[j]$
 7:     $Q \leftarrow p$
 8:     $M = $ size($Q$)
 9:     for $k = 1, 2, ..., M$ do
10:         $E \leftarrow K.search(p, r)$
11:         $T = $ size($E$)
12:         for $l = 1, 2, ..., T$ do
13:             if $T[l]$is not processed then
14:                 $Q \leftarrow T[l]$
15:             end if
16:         end for
17:     end for
18:     $C \leftarrow Q$
19:     $Q \leftarrow \{\}$
20: end for
21: return $C$
```

robot. Figure 5.7 contains several examples of the clustering algorithm for different configurations and objects.

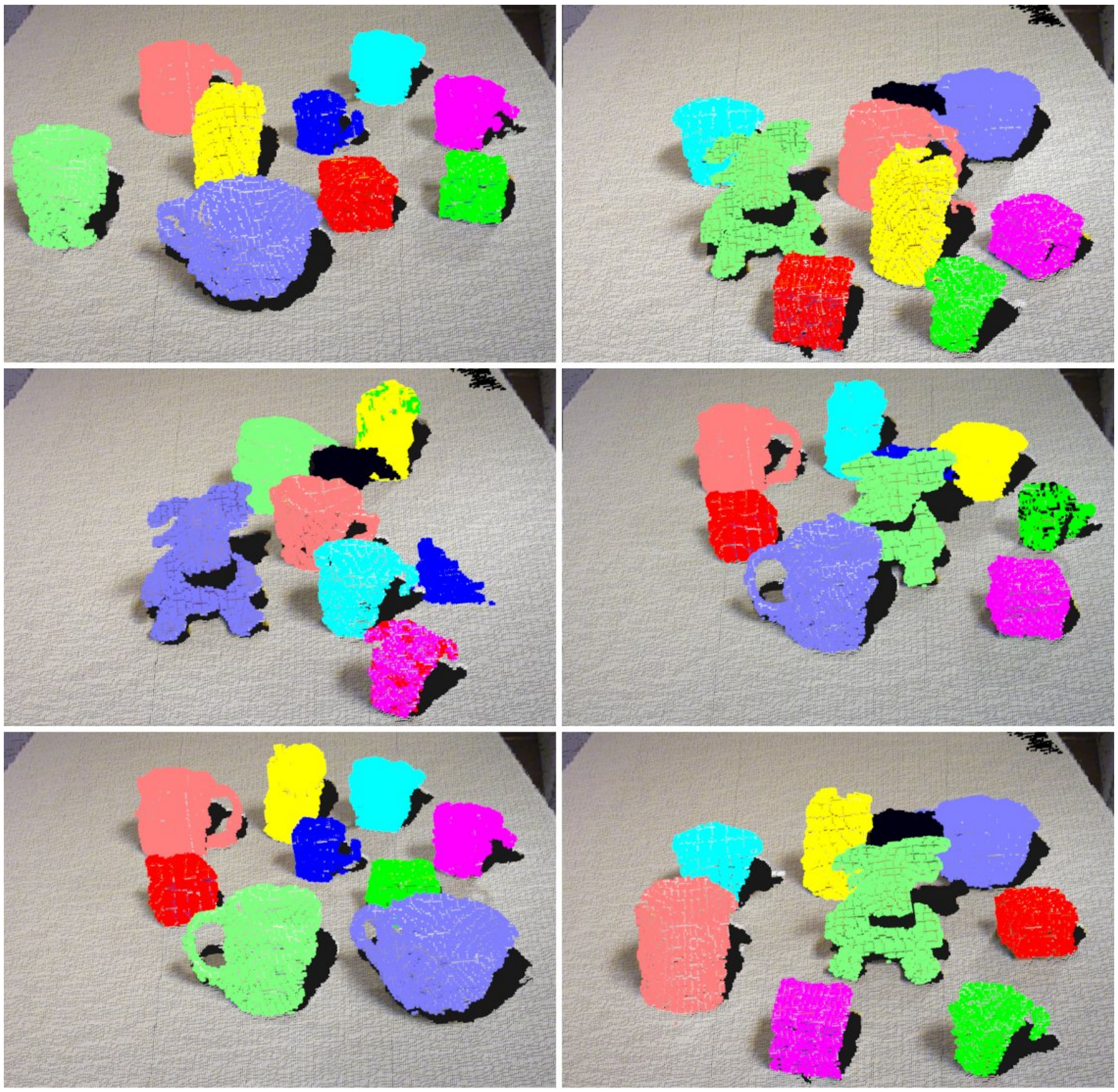


Figure 5.7: *Euclidean Cluster Extraction Examples.* Demonstration of the clustering on the laboratory.

PART II

RECONSTRUCTING NEW MODELS

Chapter 6

Correspondence between point clouds

An autonomous robot must obtain a correct idea of which and where are the objects approaching to its surroundings to place a safe interaction with them. It must be capable of recognizing them, grasping them and even using them to perform some specific tasks such as filling a glass of water by picking a jar. This chapter presents mechanisms for the integration of partial datasets acquired from different views of an scenario or an object into a consistent global model.

Before recognizing a single item, it must be taught how to extract valuable information in order to learn from it. This includes parameters such as its geometric definition, color histograms, textures, shapes, semantic information related with the object such as places where the item can be found, what kind of task the object is designed to perform, any information that might help to localize the robot itself, etc. Furthermore, each item might indicate if it is normally used alone or requires from other objects to become useful, increasing the probability of finding the rest of those necessary objects in the nearby of the query item. There are many areas concerned with 3D reconstruction among which virtual reality applications, digital preservation of cultural heritage, machine vision, medical imaging are the most common.

The first thing to do in order to recognize the object with a 3D camera is to acquire a complete three-dimensional model of each desired object. Furthermore, this comprehensive model may help not only to recognize it from other points of view but also to define a safe strategy to grasp it. A large list of techniques has been proposed for object model acquisition using RGB-D cameras. In [64], classical multi-view stereo is combined with a ToF camera to reconstruct poorly textured regions. Using just a single 3D camera, Cui [65] focuses on providing high quality models with costly super-resolution techniques. Using a different kind of RGB-D camera, Krainin *et al.* [66] established a surface-based technique relying on *surfels* (surface elements). This technique however requires a high depth precision that is not currently provided by existing ToF cameras.

There exist two principal solutions to face the object reconstruction challenge. On the one hand, those methods based on space volume. On the other hand, those based on registering points. The first alternative generates a fixed 3D volume that contains a solid body and it is carved using a ray emitted from the sensor device along the robot point of view. When the sensor perceives a depth distance at any three-dimensional position, the volume is carved until reaching the same depth. Repeating the process along the scenario it is possible to create a carved reconstruction of the surfaces.This is assumed as an efficient tool to provide dense and full 3D reconstructions of objects from multiple views.

The second alternative is known as correspondence between point clouds. Given two consecutive point clouds, the rigid transformation between them is determined and constructing a new point cloud containing the summation of both. This is done iteratively creating a global point cloud containing single views of the scenario. The reconstruction of a scene using 3D sensors or range scanners requires an acquisition phase, a registration of the range scans in the same coordinate system and a data processing for refinement: removal of redundant information and creation of a hole filling model comprised by polygonal facets. Registration algorithms are divided into two groups based on the dynamic of the scenario: rigid objects acquisition and those for non-rigid objects [67]. The range scan registration procedure can be divided into two steps: an initial registration that provides a good initial guess of the alignment transformation and then fine registration that gives the accurate alignment transformation.

6.1 Reconstruction based on Dense Maps

Dense maps reconstruction is also known as multi-view reconstruction. Merging several depth maps in the same space provide of dense and full 3D reconstructions. Observing that ToF sensors are very good at delivering silhouettes of the objects in a scene, the first idea was to reconstruct objects based on the visual hull of the object. Also, based on the surface of the object, space carving reconstruction methods exist that erode a solid body. Finally, there exists a class of methods that optimize the surface integral of a consistency function over the surface shape. Last alternative is more focused on methods that optimize *iso-surfaces*.

6.1.1 Visual Hull Reconstruction

Silhouettes have been used extensively in the literature such as Kutulakos *et al.* [68], Yemez *et al.* [69] and Walck *et al.* [70], but their computation is still problematic using classical cameras. Uniform or easily discriminated backgrounds are usually required, resulting into a less flexible system. However, using depth information, it becomes easy to discriminate an object of interest from the background extracting silhouettes using a depth threshold. Silhouette-based methods are popular for use in multi-camera environments mainly due to their simplicity and computational efficiency. 3D reconstruction based on silhouette information is done by means of the intersection of visual cones, as shown in Figure 6.1.

Franco and Boyer [72] presented a framework for multi-view silhouette cue

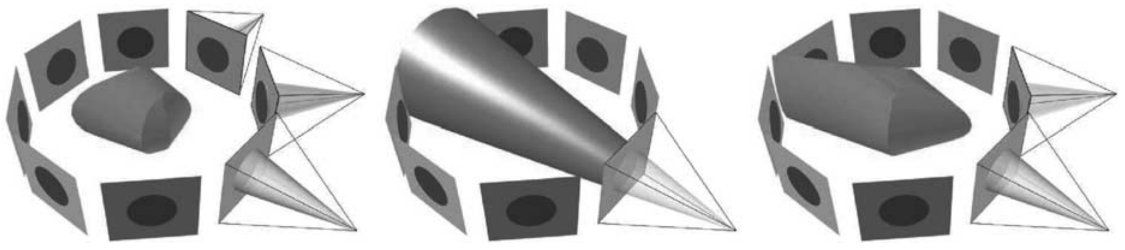


Figure 6.1: *Visual hull object reconstruction.* Carving process by silhouette cones to locate object shape. Courtesy of Yemez *et al.* [71].

fusion where a space occupancy grid is translated into a probabilistic 3D representation of scene contents. The idea behind their work was to consider each camera pixel as a statistical occupancy sensor. All pixel observations are then used jointly to infer where, and how likely, the object is presented in the scenario. Yemez and Wetherilt propose in [69] a volumetric fusion technique that fuses geometrical information acquired from silhouette images and optical triangulation using marching cubes algorithm [73]. A Bayesian approach was proposed by Grauman *et al.* [74], where they model the prior density using a probabilistic principal components analysis-based technique and then estimate a maximum a posteriori reconstruction of multi-view contours.

With the recent revolution of 3D printers, visual hull reconstruction techniques are being widely exploited to reconstruct objects creating 3D scanners. Companies such as makerbot *Inc.* have developed a digitizer[7] (see Figure 6.2) based on this principle, permitting to create models in less that 12 seconds using 720 silhouette scans and texture information. As a general rule, shape-from-silhouette approaches have been a cheap and fast alternative for object reconstruction in offline tasks where the human supervision is taken for granted. Dynamic scenarios are not supported as silhouettes might not match and become inconsistent during the occupancy analysis.

In this aspect, a simple space carving technique based on the depth measurements was designed in the laboratory and explained in [75] and is illustrated on Figure 6.3. It is similar in spirit to the work of Walck *et al.* [70] but without the application of photo consistency. This prevents the integration of small concave details but processing is faster and the obtained models are accurate enough for manipulation tasks.

[7]More info: `http://store.makerbot.com/digitizer.html`

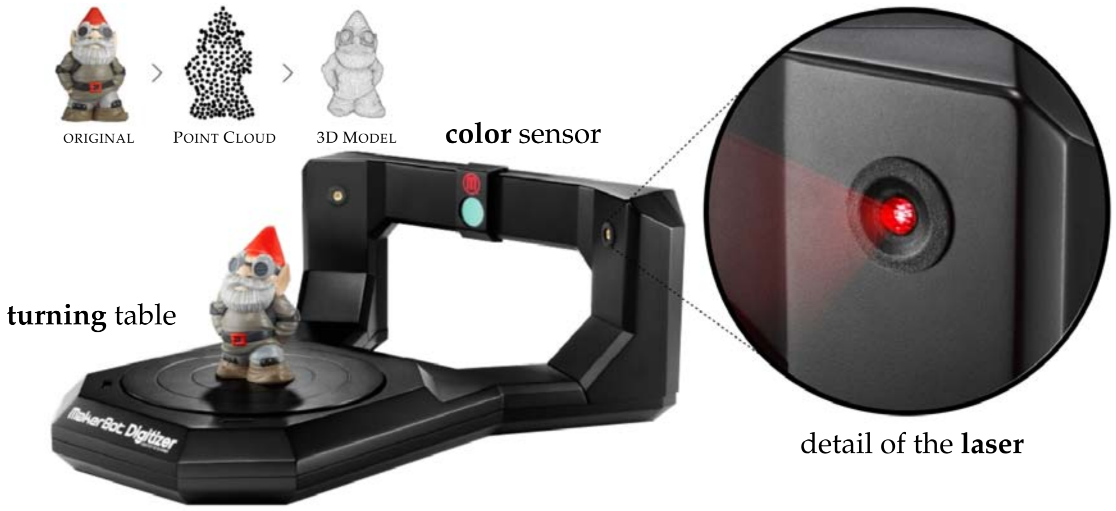


Figure 6.2: *MakerBot Digitizer Desktop 3D Scanner.* Companies are developing fast 3D scanners based on convex hull techniques. They include color sensor devices to map the 3D point clouds.

The algorithm developed iteratively carves a 3D discrete volume. Each voxel is represented as a cube whose size is user-defined in function of the level of details required. For each view taken by the Time of Flight camera, voxels are eliminated according to their depth compatibility with the new view. This is done by projecting each voxel onto the depth image, and comparing its depth d_{voxel} with the measured depth d_{view}. If $d_{view} > d_{voxel}+\delta$, it means that the voxel is not consistent in the scene, and it is discarded. δ is the tolerated margin and depends on the sensor precision. In all the experiments performed it was set to 3 cm to be conservative and avoid removing voxels with the camera. This value is not very sensitive since most of the carving will come for the edges. A small value however enables the reconstruction of concave structures whose depth is greater than δ.

Depending on the voxel size, the projection of one voxel can overlap several pixels on the depth image. The actual overlap is approximated by computing the projected width and height of the voxel and comparing d_{voxel} with the depth measurements in the corresponding neighborhood in the depth image. If at least one d_{view} is compatible with d_{voxel}, the voxel is kept. The output of the algorithm is a rather dense set of cube-shaped voxels. For manipulation tasks, it is more useful to get a surface representation of the object. This can be achieved by first removing all the inside voxels with a simple neighborhood test, and then run a surface reconstruction algorithm such as Poisson [76].

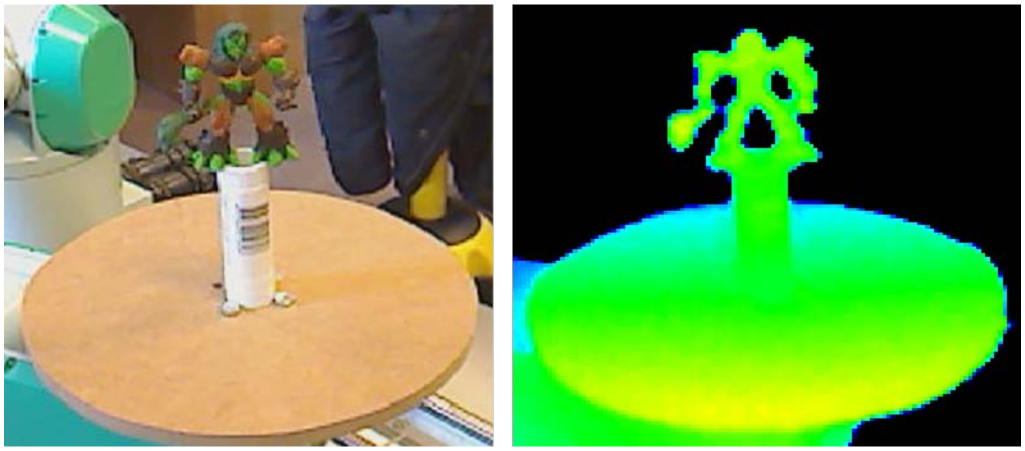


Figure 6.3: *Silhouette extraction.* Using a ToF camera and simple depth thresholding. The depth image is color-encoded.

Some results are given in Figure 6.4. These models were acquired using 35 views captured by rotating the turntable by 10^o steps. The camera was at a distance of 50 cm from the object and the chosen voxel size is 1 mm. Processing time is currently less than 10 s on a 2 Ghz computer for 36 views, and real time performance should be reachable using a careful implementation.

6.1.2 Space Carving Reconstruction

This reconstruction method takes into account the photometric consistency of the surface across the input images creating a consistent model of the scanned object. Starting with a dense space sufficiently enough to house the object, the surface of reconstruction is partially eroded on those areas where consistency is approved. When evolving the surface, the visibility has to be considered because it depends specifically on the view it is observed from. If a voxel is removed, visibility has to be updated because new voxels might then appear. This method is efficiently done using a multi-pass plane-sweep algorithm that repeats the scan until it converges. Furthermore, voxel removing has to be performed with care because if a voxel is removed by error, further voxels can be erroneously removed in a cascade effect, as Kordelas *et al.* pointed in [77].

The voxel-based representation, used in the space carving approaches, disregards the continuity of shape and makes very hard to enforce any kind of spatial

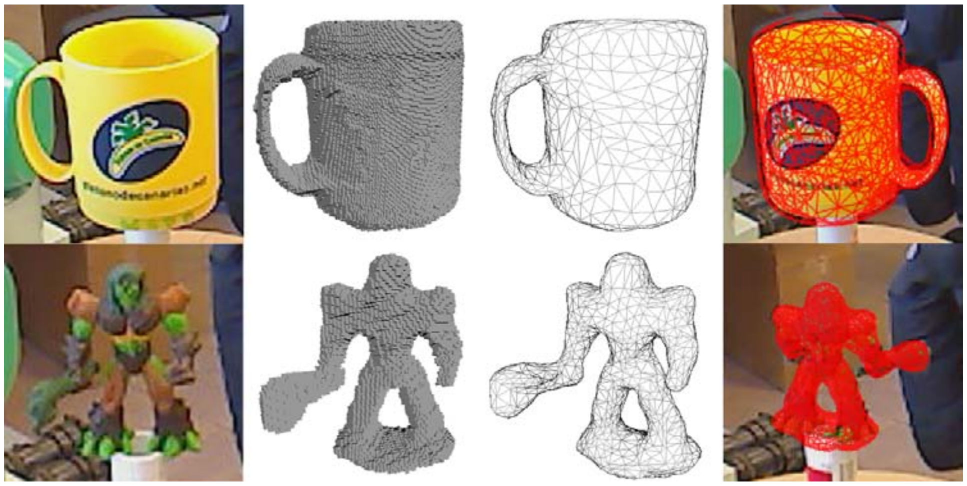


Figure 6.4: *Examples of acquired models*. First the object to scan is represented; then, the carved volume generated; afterwards, the Poisson reconstruction is performed and finally the re-projection on a color image with known pose.

coherence. As a result, space carving is sensitive to noise and outliers and may yield to noisy reconstructions. For this reason, spurious voxels are discarded using a noise filter.

This method brings up several limitations that must be taken into account:

- The photo hull is only guaranteed to be the tightest superset of the true reconstruction. That means that the reconstructed photo hull is only a superset of the true shape but does not guarantee the removal of internal hulls.
- Photo-consistency measure is critical.
- If a voxel is wrongly removed it can lead to the removal of other correct parts of the object.
- Needs calibrated input images.
- Might become problematic for non-Lambertian[8] surfaces.
- Accuracy is limited by voxel resolution. GPU is highly recommended to reduce processing time. A 256^3 voxel model requires 3-4 minutes to be computed even on recent hardware [78], while Zach *et al.* [79] proposed a method that creates a model in 5 seconds.

[8]Lambertian surface is commonly called to any surface with isotropic luminance.

6.1.3 Surface Integral Minimization Reconstruction

The last class of methods for object reconstruction is focused on the surface shape of the object. A cost function defines the level of similarity between a consistency function and the surface shape. The idea is similar to drop a slim and lightweight sheet over the object and analyze which is the best mathematical representation of the sheet's final shape. First proposals [80], [81] used gradient descent method to converge the minimization problem, handling complicated topology and deformations as well as noisy or highly non-uniform data sets. Duan *et al.* [82] developed a mathematical framework capable of discovering not only the underlying topological structure of the object but also its geometric boundaries using partial differential equations and deformable surfaces.

Jin*et al.* proposes in [83] a surface integral minimization algorithm which starts with a cube and evolves it to approximate it to the object by numerically integrating systems of partial equations. The main restriction of this system is because of the surface evolution, which assumes a closed and smooth surface. This assumption rejects sharp or noisy objects such as outdoor or polyhedric objects. In [84], a multi-resolution approach is presented. It starts with coarse settings, and then it is refined in those zones of interest.

Radial Basis Functions (RBFs) are used by Carr *et al.* [85] to generate surfaces based on a polyhedric representation. This functions permit to reconstruct manifold surfaces from point-cloud data smoothly and also to repair incomplete meshes. Because RBFs are scale-independent, this alternative is well-suited to reconstructing surfaces from non-uniformly sampled data. Figure 6.5 shows a sample of an object reconstruction using RBFs functions. The number of centers of those RBS functions determine the complexity and accuracy of the final reconstruction. In this case the 544,000 point cloud is represented by 80,000 centers to a relative accuracy of 5×10^{-4} in the final frame. Pons *et al.* [86] proposed a method for multi-view stereo vision that minimizes the prediction error using a global image-based matching score handling projective distortion and partial occlusions.

Indeed, there exists a second class of surface minimization based on graph-cuts. Those methods such as Yu*et al.* [87] and Boykov *et al.* [88] transform the surface fitting problem into a graph optimization. By means of *Surface Distance Grid* (SDG), Yu *et al.* [87] minimize a cost function over the object surface where two types of biases are found: the minimal surface bias and the discretization bias. These biases make it difficult to recover surface extrusions and they are turned into controllable degree of surface smoothness. This method requires an initial estimation

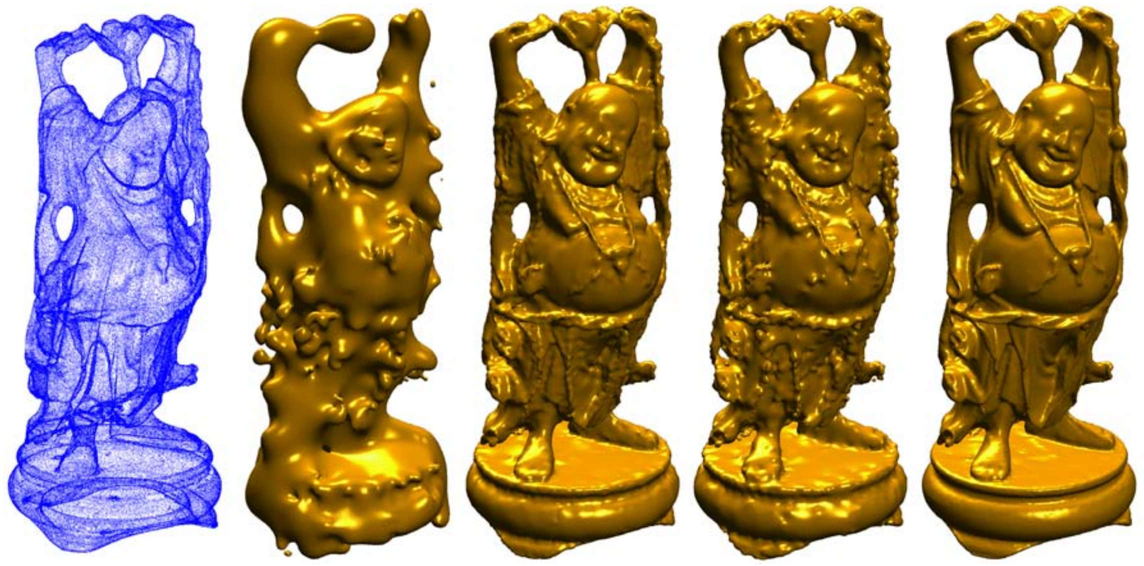


Figure 6.5: *Surface Integral Minimization Reconstruction using RBFs.* Using a greedy algorithm iteratively, a RBF is fitted in the point cloud reducing the number of centers required to represent the whole surface. Courtesy of [85].

close enough to the final model in order to converge successfully. Hornung *et al.* [89] uses an octahedral graph structure that creates a well delimited connection between the photo-consistency of a voxel and the edge weights of an embedded octahedral subgraph. This specific graph design supports a hierarchical surface extraction, which allows processing even high volumetric resolutions and a large number of input images efficiently. Shi *et al.* [90] integrate a curvature-based variational model and Delaunay-based tetrahedral mesh framework succeeding in the reconstruction of surfaces with important features such as sharp edges and corners.

6.2 Correspondence between Point Clouds

The challenge of consistently aligning various 3D point cloud data views into a complete model (*world*) is known as *registration*. Registration transforms multiple 3D datasets into the same coordinate system so as to align overlapping components of these sets. As a result of the restrictions of 3D scanning technology, more than one datasets must be grabbed from different view-points, each scan is associated with a different coordinate system. Original reconstructions were used to align laser scanner technology, nowadays those systems are used also in 3D sensors using Time of Flight or light coding technology. The basic advantages of the methods that use this technology are: speed, accuracy and resolution of the reconstruction. This alternative can be used not only for small and reduced scenarios but also to large scale outdoor scenes where the number of points becomes enormous.

The challenge during the 3D reconstruction consists on register each point cloud and determine the rigid transformation between consecutive scans. As a general rule, the registering process is a challenging task due to several drawbacks:

- Point clouds could contain not only systematic noise due to the sensor but floating artifacts in borders or zones with significant depth changes.
- Bright and shiny surfaces affect the reflexion of the infrared pattern altering the resulting point cloud elements.
- Errors are depth dependent. That means that the precision varies with distance: the further the object, the most inaccurate the measure.
- Uncertain shadows and black gaps behind the objects, due to a view-point dependency, might make the correlation difficult.
- Large rotation changes of the camera entail into dissimilar scenarios.
- Point clouds might not coincide from a feature representation point of view or in space due to slight changes in borders or corners.

The variety of applications is worth while, such as reverse engineering and mold fabrication in the manufacturing process, artifact reproduction and 3D modeling of carving pieces and sculptures both with applications in the souvenir industry and virtual museums, and many others such as augmented reality in graphics and map building in robotics. Figure 6.2 illustrates the correspondence problem: a set of point clouds of the same face are disposed from different poses which are incipiently unknown. The intention of gathering all those samples and constructing a global face model requires from observed information to be examined and processed.

There exists a considerable number of ways to perform the correspondence, range data alignment proposals are widely extended in the literature: Bienert and Maas propose in [91] a registration method for forest stands using a terrestrial laser scanner. Three detection features are used: artificial key-points, using the tree axes

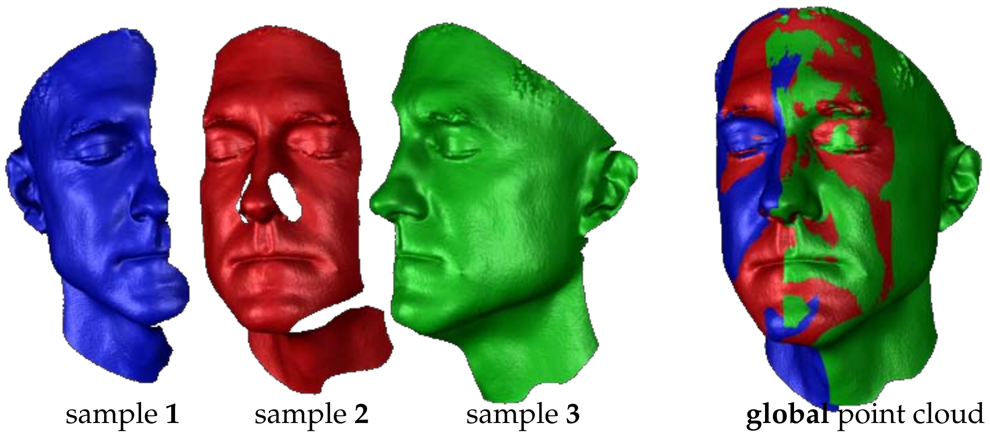


Figure 6.6: *Correspondence problem.* A set of samples from different point-views of the same face are merged into a global point cloud

and finally avoiding artificial tie points. Others such as [92] mixtures multi-view geometry with automated registration of 3D range scans to produce photo-realistic models with minimal human interaction. Feature-based registration algorithms are based on angular features, as Chen *et al.* [93] or Jiang *et al.* [94] pointed out. Most of these methods are based on finding correspondences between distinctive features that may be present in the overlapping area. The basic procedure involves the identification of features, assignment of feature correspondences and then computing an alignment based on these correspondences. There exist many different features that can be explored: edge maps [95], lines and planes [96], bitangent curves [97], surface curvatures introduced by Yamany *et al.* [98], [99], surface orientation explained in [100] by Johnson, and invariant features, such as moments and curvature detailed by Sharp *et al.* [101].

In some circumstances, markers are used as in Qian *et al.* [102], Bienert *et al.* [91] or Akca *et al.* [103], making it simple to extract the features, but with a common drawback: the object to be scanned has to be physically prepared or, in case the interest is focused on a scenario, the placement has to be previously adapted.

In [104], Chua and Jarvis presented the Point Signature, a new form of point representation for describing 3D free-form surfaces that is invariant to rotation and translation. The point signature can be used directly to hypothesize the correspondence to model points with similar signatures. Furthermore, surface registration can be planted as a high dimensional optimization problem solved using genetic algorithms such as Chow *et al.* [105], where a novel fitness function is presented. A common issue during the correspondence is to deal with coarse point clouds.

Mian *et al.* [106] presented a new feature matching algorithm that uses a tensor representation which acts for a semi-local 3D surface patches of a range image by third order tensors. This novel representation becomes robust for dense-limited point clouds.

Any registration algorithm must take into account several challenges including extensive structural changes, large viewpoint differences, repetitive structure, illumination differences and flat regions on every scan. For this reason, the usage of textures, and in particular the frequency analysis of these textures, may lead to better results. Makadia *et al.* present in [107] a scan alignment based on the correlation of two Extended Gaussian Images (EGIs) created by [108] in 1993. EGIs are useful in poor overlapping scans with large displacements between consecutive scans. On those, the EGIs correlation is done in the Fourier domain and therefore the rotational alignment is obtained by the alignment of constellation images generated from the EGIs.

Additionally, intensity features can be combined with range information to boost registration methods such as [109], where local structures and color of object surfaces are extracted as shape and chromaticity patterns so it is possible to determine the pose increment between scans. Bendels *et al.* [110] extract 2D *Scale Invariant Features Transform* (SIFT) descriptors from each frame to estimate the rigid transformation based on texture and then re-project the position onto the depth map. In [111], *Speeded Up Robust Features* (SURF) features of color images are extracted and then RANSAC algorithm is employed to remove large amount of outliers.

As mentioned before, 3D reconstruction based on registration is divided into two main parts: an initial registration that provides a good initial guess of the alignment transformation and then fine registration that gives the accurate alignment transformation. The later part is in charge of improving the initial alignment transformation. The better initial estimation, the faster the convergence of the refinement. In the literature, the *Iterative Closest Point* (ICP) is a very popular method for the fine registration of 3D data sets.

6.2.1 Iterative Closest Point

ICP was originally presented by Zhang [112] and proposed for point matching for free-form curves and surfaces, but it can be easily extended to three-dimensional problems or any $N-$dimensional system. ICP is in charge of refining the 3D point cloud focusing on the position of each point. It iteratively revises the rigid transformation (translation $\boldsymbol{R}$, rotation $\boldsymbol{t}$) required to minimize the distance between the points of corresponding point sets $\boldsymbol{A} = \{a_0, a_1, ..., a_N\}$ and $\boldsymbol{B} = \{b_0, a_1, ..., b_N\}$, such that a error function $e(\boldsymbol{R}, \boldsymbol{t})$ can be defined as

$$e(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} w_{i,j} \| a_i - (\boldsymbol{R} \cdot b_i + \boldsymbol{t}) \|^2 \tag{6.1}$$

where $w_{i,j}$ is equal to 1 if i and j are corresponding points. It means that they describe the same point in the space. If they are not corresponding points, $w_{i,j}$ is 0.

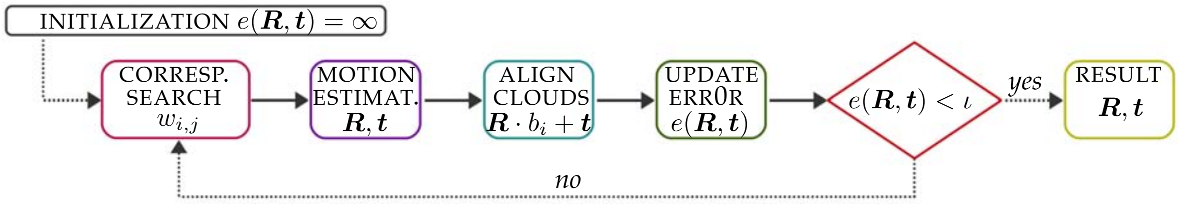

INITIALIZATION $e(\boldsymbol{R}, \boldsymbol{t}) = \infty$
CORRESP. SEARCH $w_{i,j}$
MOTION ESTIMAT. $\boldsymbol{R}, \boldsymbol{t}$
ALIGN CLOUDS $\boldsymbol{R} \cdot b_i + \boldsymbol{t}$
UPDATE ERR0R $e(\boldsymbol{R}, \boldsymbol{t})$
$e(\boldsymbol{R}, \boldsymbol{t}) < \iota$
yes
RESULT $\boldsymbol{R}, \boldsymbol{t}$
no

Figure 6.7: *Iterative Closest Point flow chart.* Representation of the steps followed by ICP iteratively to estimate the transformation between two point clouds. The algorithm converges when error $e(\boldsymbol{R}, \boldsymbol{t})$ is lower than the threshold ι

The original algorithm is presented in Algorithm 2 (notice that Nearest Neighbor Problem was introduced as an improvement proposed in 2001 by Greenspan and Godin [113]). Its complexity is low enough to work in real time even with large 3D cloud points. Despite the fact that the results provided by authors were very good, ICP usually presents problems of convergence, lots of iterations are required, and in some cases the algorithm converges to a local minimum due to large variations between scans. The key concept of the standard ICP algorithm can be summarized in two steps:

1. Compute correspondences between the two scans.

2. Compute a transformation which minimizes distance between corresponding points.

The flow chart of the ICP process has been displayed in Figure 6.7. For this work, a modified version of the algorithm proposed by Besl *et al.* [114], called *Generalized ICP* has been used. This method is based on attaching a probabilistic model to the minimization step on line 11 of Algorithm 2. This option does extract

Algorithm 2 Iterative Closest Point pseudo-code

```
Require: Two point sets: $A = \{a_0, a_1, ..., a_N\}$ and $B = \{b_0, a_1, ..., b_N\}$
Require: Initial transformation : $T_0$
Require: KNN search radii $r$
Ensure: Rigid transformation Matrix $\boldsymbol{t}$ [4x4] which aligns $A$ and $B$
 1: $T \leftarrow T_0$
 2: $K \Leftarrow KD$ tree($A$)
 3: while not converged do
 4:     for $i \leftarrow 1$ to $N$ do
 5:         $m_i \leftarrow K.search(T \cdot b_i, r)$
 6:         if $||m_i - T \cdot b_i|| \leq d_{max}$ then
 7:             $w_i \leftarrow 1$
 8:             $w_i \leftarrow 0$
 9:         end if
10:         $T \leftarrow \underset{T}{\arg\min} \{\sum_i w_i \cdot ||T \cdot b_i - m_i||^2\}$
11:     end for
12: end while
```

planes from both clouds, doing a plane matching instead of point matching. This probabilistic approach mix the simplicity of the original proposal over the advantages of other fully probabilistic techniques: speed and simplicity. Furthermore, other works such as Godin *et al.* [115] included color and curvature as matching constraints but they did not regard on luminance variations or occlusions.

Trucco *et al.* [116] implemented the RICP (*Robust* ICP) method making use of the Least Median of Squares approach to boost the quality of the method, increasing the robustness of ICP substantially. The method is based on executing the registration with simply a portion of random points (m points), evaluating this operation a sufficient number of times with the aim of finding a registration without outliers. The Monte Carlo algorithm was used to estimate the number of executions. Once all the potential registrations were computed, the one that minimizes the median of the residuals is chosen as the solution. Finally, the correspondences with a residual larger that 2.5σ were removed and the transformation between both views was computed using only the remaining inliers.

Nevertheless, during the last years, the inclusion of faster processors and sensors has provided with new techniques for 3D correspondence mixing both volume carving and point cloud correspondences. Others such as Klein and Murray [117] created a parallel tracking and mapping system called PTAM. Newcombe *et al.* presented a dense tracking and mapping algorithm called DTAM in [118].

As a drawback, notice that ICP refinement requires from small transformations between frames in order to succeed and also from a reasonable good initial estimation. Otherwise, ICP may fail if the relative rotation between consecutive scans or the displacement is abrupt. As Byung*et al.* [119] conclude, the variance of the ICP registration error is directly proportional to the variance of the additive noise and inversely proportional to the proposed reliability and the number of data points. Byung recommends to increment the number of control point around the axis during rotation to increase reliability.

6.2.2 Kinect Fusion *a.k.a.* Kinfu

Kinect Fusion was originally presented by Newcombe and Izadi *et al.* in [120] and also in [121] during the last trimester in 2011. Microsoft Research Cambridge in conjunction with the Imperial College London (UK) proposed an innovative and revolutionary algorithm for scan matching based on a quite smart clue. They migrated the tedious point-wise correspondence process to the GPU, boosting the complete scan matching process several times faster.

Kinfu pipeline

The Kinect Fusion technique reconstructs a sole dense surface model with smooth surfaces by combining the depth data from Kinect continuously from several viewpoints. The camera viewpoint pose is tracked (both translation and rotation) as it is moved. With every single frame pose estimation and the relative position of the following and last, it is possible to fuse all depth maps from multiple viewpoints together into a unique reconstruction voxel volume.

For this fusion, an empty reconstruction volume is initially created (usually a cube with 3 – 4 m^3 depending on the available GPU memory). Depth data are integrated on each iteration over the reconstruction volume as long as new scans are positioned inside its limits. This integration is also called volume carving because it *carves* on each voxel the distance determined by the sensor.

The algorithm is divided into four parts:

1. *Depth map conversion.-* Transform the raw point cloud captured by the Kinect sensor into real world 3D floating point depth values based on the camera coordinate axis and compute the normals for the actual depth map.

2. *World pose estimation.-* Integrate the actual depth map into the global coordinate system based on the relative transformation between the scan and the initial starting frame. An implementation of a parallelized ICP scan matching algorithm is used reaching more than 20 fps.

3. *World volumetric fusion.-* According to the pose estimation result, the depth map is introduced into the global reconstruction volume. This step is done introducing a Bayesian occupancy estimation for each voxel so that dynamic objects can be altered over time from the global scenario. Furthermore, this process is fastened using a TSDF cloud. The TSDF value is the distance to the nearest isosurface for each voxel of the scan. That is, TSDF is zero in an isosurface, positive between the sensor and the isosurface, and negative otherwise.

4. *Raycast rendering.-* Scenario is ray-casted from the current Kinect sensor pose so the global reconstruction volume is shaded for a rendered visible image of the complete 3D scenario. This last step is exclusively used for representation purposes and could be ignored.

However, Kinfu is not a real SLAM algorithm as it does not explicitly close large-scale loops and will inevitably incur drift over time. Rather, it can be considered a 6D visual odometry approach which tracks relative camera motion. Of course the significant additional benefit beyond visual odometry alone is that a map of local environment surfaces is also always available.

Moving Volume KinectFusion

The most determining problem of Kinect Fusion is that the global reconstruction volume is fixed to a certain value. This might be an issue if large scenarios are demanded to be scanned. To solve this, Roth *et al.* [122] proposed moving volume Kinfu. Figure 6.9 represents a large scenario where several moving volumes have been used.

Using a fixed volume of 3m^3, Kinfu is performed as usual. When the sensor starts capturing data outside itself, the actual volume is moved from GPU to CPU and a new fixed moving volume is created. Taking the last transformation on each volume, it is possible to recreate in CPU the summation of the individual global reconstruction volumes as represented in Figure 6.8.

Definitely, correspondence algorithms are used thoroughly in robotics to create global maps of information in the vicinity of robots and they have become a powerful tool not only to perceive the surroundings but also to model any object that

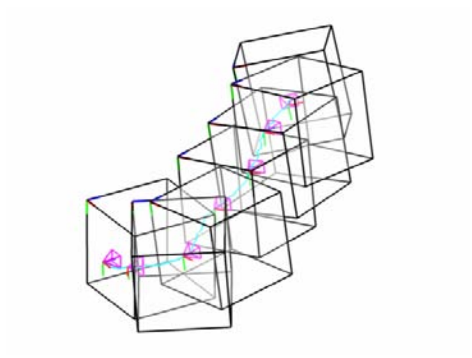
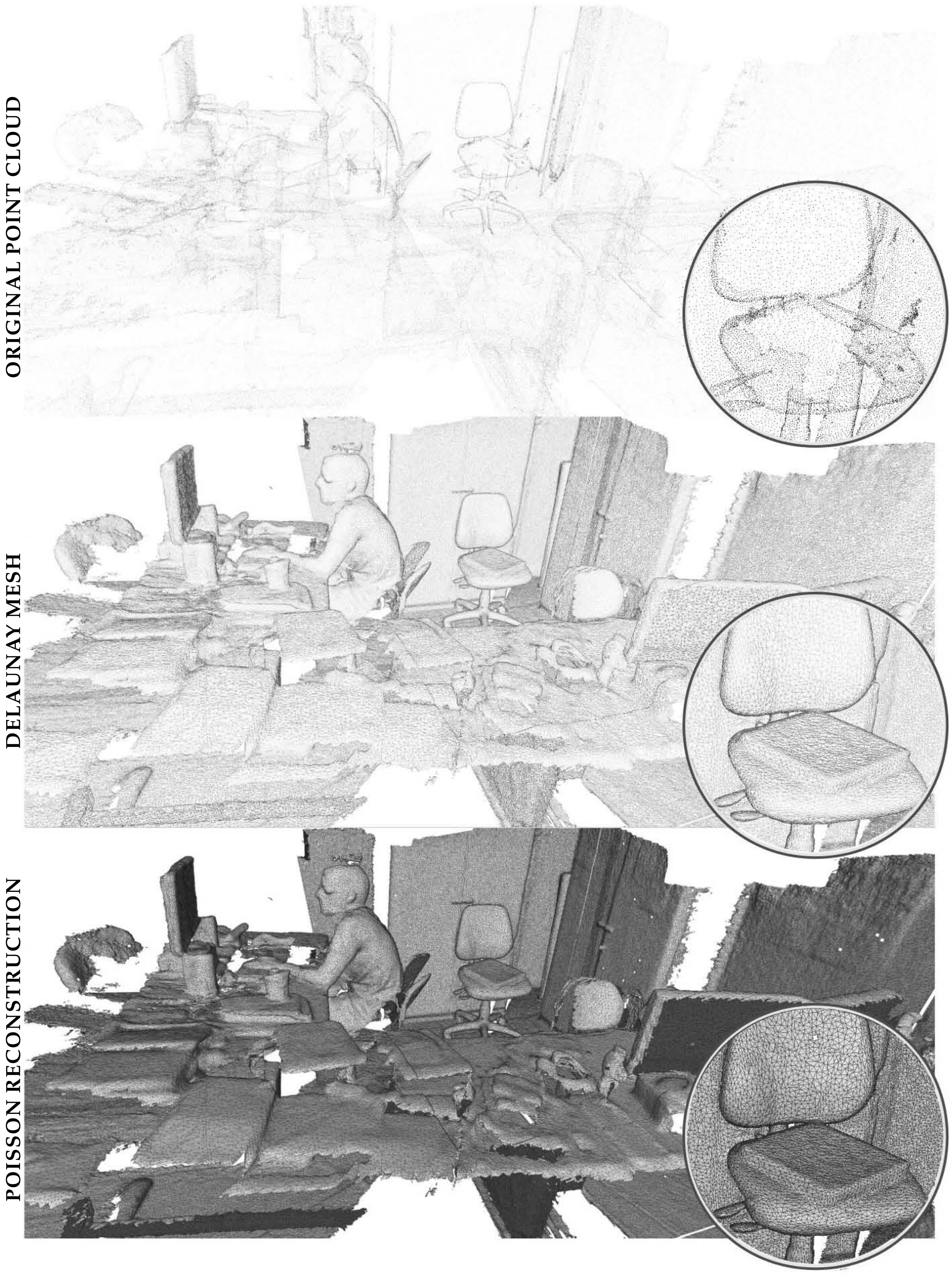


Figure 6.8: *Moving volume schema.* Representation of a subset of the moving volumes used in [123] to reconstruct a climb up two staircases in a hallway ($\sim$ 11.3 m).

might be interesting for any reason. In this thesis, the correspondence problem has been applied for object reconstruction as will be introduced in the following chapter. A novel ICP algorithm that makes use of an evolutionary algorithm will be presented to generate complete 3D models of objects with limited point cloud information.


ORIGINAL POINT CLOUD
DELAUNAY MESH
POISSON RECONSTRUCTION

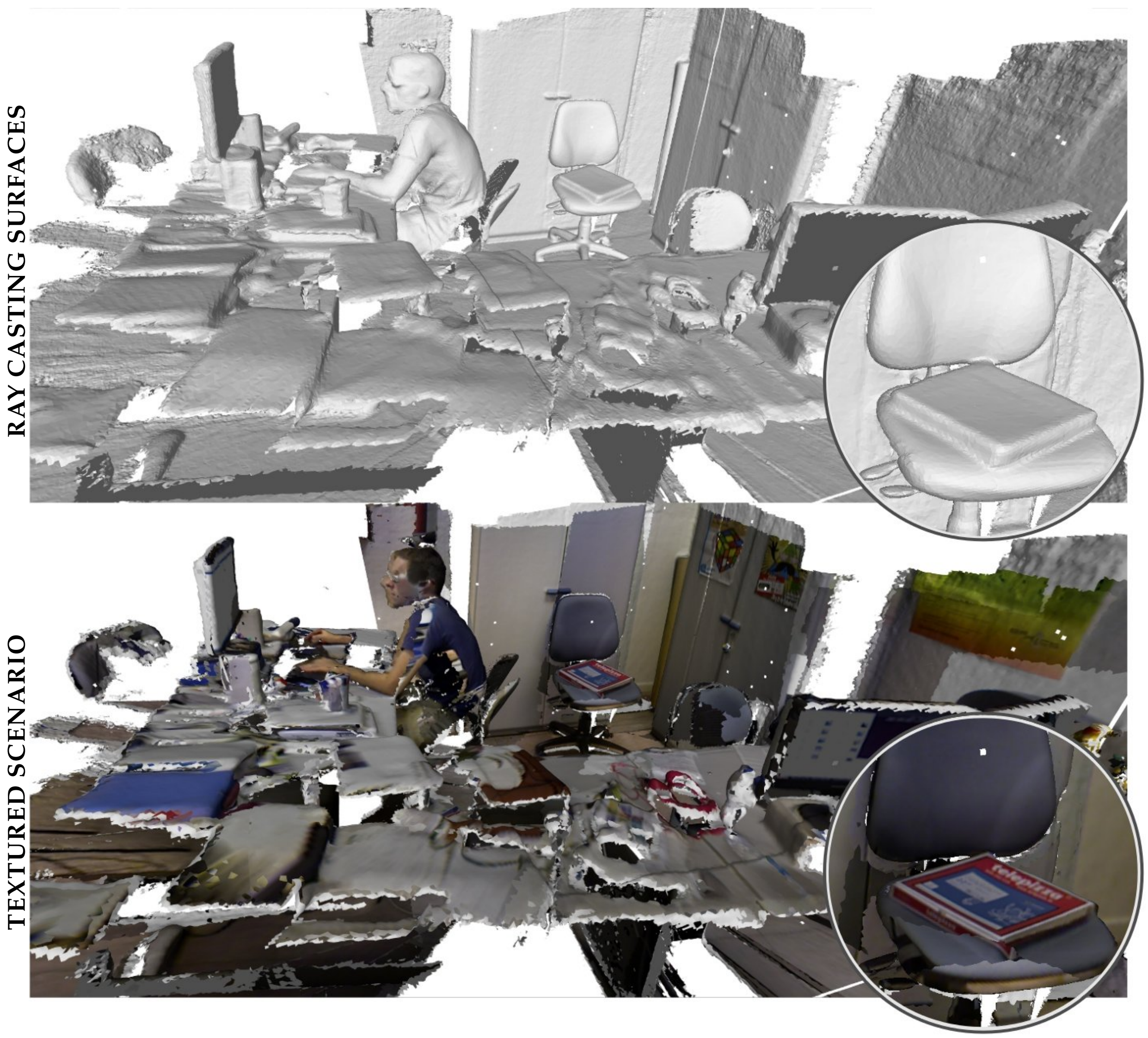


Figure 6.9: *Moving volume Kinect Fusion.* Representation of the laboratory using Kinect Fusion algorithm with moving volume. Global reconstruction volume is updated on demand so Kinfu is extended to larger environments. Respectively: original point cloud, Delaunay mesh triangulation [123], Poisson surface interpolation [76] and texture mapping. The office chair has been zoomed to highlight the definition level of the algorithm.

Chapter 7

Evolutionary model reconstructor

The aim of this chapter is to formulate the registration problem as a high dimensional optimization problem which will be solved using a evolutionary algorithm (EA). In particular, DE optimizer is proposed with a precise fitness function that takes merges favorably several characteristics from the point cloud.

The majority of the alternatives to reconstruct 3D models based on registration are divided into two strategies: point-wise comparison and feature matching. The first choice has been explained in detail during the preceding sections in algorithms such as ICP. This process can be boosted parallelizing the proper algorithm achieving large frame rates. On the other side, the second alternative is the usage of matches based on correspondences such as in Yamany *et al.* [124], who proposed a novel surface signature that can be compared using template matching. In [125], Schutz *et al.* analyze a multi-feature ICP matching algorithm that includes the surface color and the surface orientation information.

The problem of ICP falls in the enormous sensitivity to rotations. While the algorithm responds correctly to point cloud correspondences when the object is linearly shifted, it breaks when rotations are abrupt. Furthermore, its performance is greatly affected by noise and occlusion, specially in multiple range image registration. Thus, in cases where there are a few views of the object, or situations where the robot moves at the same time as the object does, as a general rule ICP does not converge successfully to a global minima. A robust method for dense clouds was proposed by Masuda and Yokoza[126], where random sampling and the *Least Median of Squares* (LMS) estimator were integrated in the original ICP algorithm. In this way, they firstly segmented input data points in four types: inliers and three categories of outliers (occluded, unpaired or outlier). The least squares estimator minimizes the sum of squared residuals while the LMS estimator minimizes the median of squared residuals. As the LMS estimator yields better results than the standard least squares, outliers might be better tolerated up to a presence of 50%. There exist then, three drawbacks of ICP that must be overcame in order to succeed: sensitivity to rotations, delicacy to noise and data occlusion and finally, its initial estimation dependence.

To overthrow these particular situations, global optimization can turn into a powerful tool. If the problem cannot be solved locally due to large movements or steep transitions, multi-modal algorithms, in particular evolutionary-based ones, can help. In this way, this chapter brings a new approach in point cloud correspondence alternatives with an evolutionary-based method. The approach reaches the following aspects:

- *Robustness against abrupt rotations.*- The alternative has to improve ICP and converge models with large rotations. It must be ready to deal with changes of any quantity. For this reason, a multi-modal optimizer is a good idea, it will check and try on random poses covering all the possibilities.
- *Multi-dimensional optimization* .- Due to the geometrical restrictions of the supporting plane, the most noticeable change will be performed in the yaw angle, but the three Euler angles must be tuned in each transformation and therefore the six DOF have to be evaluated and optimized.
- *Feature extensible* .- The more information processed, the better guess about the relative pose achieved. For this reason, it becomes extremely useful to include extra data in the matching algorithm such as apart from the 3D point cloud, geometrical features, color, textures or any other custom features that might be added posteriorly.
- *Compatibility* .- The result of the algorithm has to be consistent and compatible with other alternatives. The most reasonable output is a 6 DOF matrix containing the displacement and Euler angles of the relative movement.
- *Uncertainty models* .- The lack of information about the different views of the same model has to be dealt with properly. ICP demands a lot of points and poses to create the complete model. However, this large quantity of views is not always available. The proposed method has to be able to converge with a minimum amount of views to complete the full model view.
- *Initialization* .- While the ICP-based algorithm yields good results, a good initial guess is indispensable. If the initial solution is far from the actual solution, the most probable thing is that the algorithm does not converge favorably.

Let us say there exist two point clouds: the first one is the *model* $\mathcal{M}$ and the second one is known as *view* $\mathcal{V}$. This model is the reference and it will never be altered or moved. Matching is performed modifying always the view point cloud. Therefore, rigid transformations, constraints and in general any kind of point-wise transformation T will be applied in $\mathcal{V}$.

Brunnstrom and Stoddart addressed in [127] the problem of free-form surface matching without an initial guess using a genetic algorithm. They defined a fitness function designed to be invariant to translation or rotation of either the scene or model becoming robust in the presence of clutter as well. They perform optimization in the correspondence space rather than the transformation space. Assuming that two points p, q are chosen, there exist two absolute positions $\vec{r}_p, \vec{r}_q$ and two normals associated to each point $\vec{n}_p, \vec{n}_q$. In order to make the distance invariant to rotation or translations, relative measurements between points are taken

$$\begin{array}{rcl} \|\vec{v}_{pq}\| & = & \|\vec{r}_p - \vec{r}_q\| \\ \cos(\theta_{pq}) & = & \vec{n}_p \cdot \vec{v}_{pq} \\ \cos(\theta_{qp}) & = & \vec{n}_q \cdot \vec{v}_{qp} \\ \cos(\beta_{pq}) & = & (\vec{n}_q \cdot \vec{v}_{pq}) \cdot (\vec{n}_p \cdot \vec{v}_{pq}) \end{array} \tag{7.1}$$

Then, if two points p, q are selected from $\mathcal{M}$ and their corresponding from $\mathcal{M}$ are picked r, s, then the pairwise match quality $e(r, s)$ will be the product of two terms, the first related to the distance mismatch

$$e_d(r, s) = exp\left\{-\frac{(\|\vec{v}_{rs}\| - \|\vec{v}_{pq}\|)^2}{2\sigma^2}\right\} \tag{7.2}$$

and the second representing the angular quality

$$e_n(r, s) = exp\left\{-\frac{(\theta_{rs} - \theta_{pq})^2 + (\theta_{sr-\theta_{qp}})^2 + (\beta_{rs} - \beta_{pq})^2}{2\mu^2}\right\} \tag{7.3}$$

where σ and μ are related to measurement noise and sampling density. Then, the whole quality error is defined by

$$e(r, s) = e_d(r, s) \cdot e_n(r, s) \tag{7.4}$$

that equals 1 when there exist a perfect matching and decreases to zero as fast as matching dissociates. If C stands for the total number of correspondences of index c between two point clouds, a pairwise relation for that match can be defined as

$$e(r, s) = \sum_{c \in C} e(\mathcal{V}_c, \mathcal{M}_c) = \sum_{c \in C} e_d(\mathcal{V}_c, \mathcal{M}_c) \cdot e_n(\mathcal{V}_c, \mathcal{M}_c) \tag{7.5}$$

However, as they suggested in [127], the algorithm they presented was no suitable for matching objects composed of *simple* surfaces such as planes, cylinders or spheres. They recommended the usage of other methods that extract such features and perform matching based on them. Their results are, as they conclude, not accurate and valid to determine fine poses. Furthermore, their experiments were presented using scans with more than a 50% overlap with a genetic population of 50 individuals, a crossover probability of 90% and a mutation rate of one percent, the number of operations needed to evaluate it growing quadratically with the number of points. Yamany *et al.* [98] proposed a registration solved by a genetic algorithm minimizing the mean-squared error considering uniquely the points that overlapped the two data sets. Salomon*et al.* [128] presented a DE-based registration that works directly on the parameter vector to be optimized. The weak point is that the cost function exploits exclusively 3D views created using high precision models being unaware of imprecise scans. Furthermore, the rotation boundaries were too much moderate, reaching relative rotations of $\pm 20^o$. Nevertheless, their work has been meticulous studied and taken as a model to improve in this thesis. Robertson *et al.* [129] proposed a parallel evolutionary registration approach based on GA. This alternative avoided becoming trapped in a local minimum in most cases reducing computational time but reducing the precision of the registration.

A quite interesting research was followed by Silva *et al.* in [130], where they suggested a hybrid genetic algorithm technique, including hill-climbing and parallel-migration, combined with a robust evaluation metric based on surface interpenetration [9]. They defined a measure of surface alignment that was called Surface Interprenetation Measure (SIM), an improvement of the classical mean-squared error used in ICP.

Chow *et al.* [105] proposed to minimize the median of the residuals instead of the Euclidean distances among all correspondence pairs. This improves the robustness but makes the method inapplicable to data overlaps below 50%. This alternative makes sense when the overlapping is sufficient. For this reason, the median of the residuals might be favorable during the last iterations of the optimization process. Olague *et al.* [132] presented a hybrid evolutionary ridge regression approach for the problem of corner modeling. Image data is modeled using multi-parameter corner models named *L*-corners. This is not a registration problem as they use parametric models but in essence tries to solve the same problem but for corner modeling. Xu and Dony [133] presented an improvement of Powell's direction set method (PDSM) for image registration by using DE for initialization. The most noticeable shortcoming of PDSM method is a local search which is not guaranteed to find the global optimum, making its performance sensitive to the initial conditions. In that way, Xu and Dony proposed a multi-resolution scheme to supplement the initial statement. The results yielded are valid but it does not converge in some cases and suffers from inefficiency when the dimension of the problem is too large.

[9]Dalley and Flynn [131] proposed that a good pair-wise registration should exhibit a large *splotchy* surface, which is the visual consequence of two surfaces represented in a contrasted color crossing over each other repeatedly. This effect can be particularized as the interpenetration of the two surfaces.

7.1 Reconstructor Architecture

A 3D scan matching algorithm has been implemented that fulfills the conditions mentioned previously for object reconstruction. Registration is based on the DE optimizer, a particle-based evolutionary algorithm that evolves with the time to the solution that yields the cost function minimum value. If the cost function is strategically thought, it becomes feasible to determine the solution for the scan matching problem applying this method. The high accuracy and computational efficiency of the proposed alternative have been demonstrated with experimental results.

The management of uncertainty models due to a small number of views is also addressed by this reconstructor. An algorithm has been developed that performs a global estimation before the point cloud matching starts, with the aim of creating a robust model with a reduced quantity of observations. Furthermore, noise points and spurious measurements are threated separately in order to differentiate the valuable information of the point cloud by means of a smoothing process on each addition to the full model. Figure 7.1 resumes the correspondence process followed in this research and the complete pipeline of how each view is attached into the full model using an evolutionary optimizer.

The system works as follows:

1. *Features extractor.*- For each view of the object, three characteristics are extracted: an intensity map based on texture histograms, a normal map based on local PCA fit planning and a distance *kd*-tree to reduce posterior computation time.

2. *Curvature extension.*- Reconstructor may deal with live captures from a structured light sensor. As they suffer from inconsistency and extensive variance in borders, holes and fuzzy boundaries may appear. Furthermore, the number of views are supposed to be restricted, with almost no overlapping between point clouds. In order to improve the optimization process, a curvature extension will be performed in those areas where holes or discontinuities appear. Those synthetic points included will be regarded in a different way during the surface alignment.

3. *Yaw initialization.*- As the number of views is limited, a first transformation will be held based on a yaw angle rotation. Taking in mind the fact that objects are lying on a supporting plane, the most probable rigid transformation between scans will be a yaw angle alteration. Evaluating a top projection of

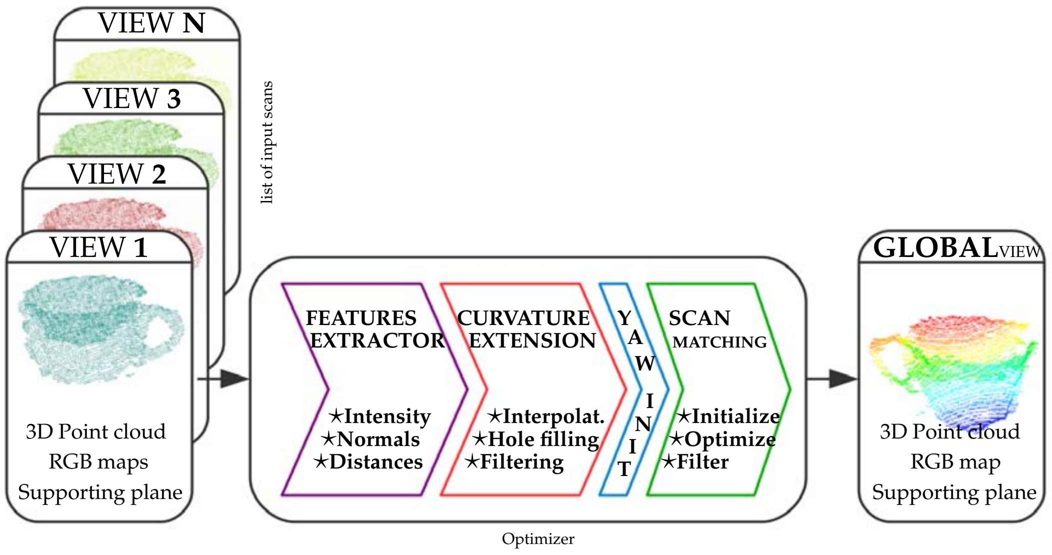


Figure 7.1: *Reconstructor schema.* Diagram of the different parts of the proposed reconstructor. Features are extracted from each scan, a hole filling is done using a curvature hypothesis. A first alignment is initialized based on yaw using brute force and then a scan matching based on DE optimizer is done. Finally, a filter is passed to clean noise points.

the point cloud at different yaw angles, it is possible to determine an initial transformation of the view in relation with the previous scan.

4. *Scan matching.*- Reconstruction of the set of point clouds using a DE-based algorithm that merges all views into a global object performing a fine registration. After the registration, a radius filtering process is accomplished to remove noise.

7.2 Curvature Extension and Hole Filling

With the purpose of increasing the robustness of the correspondences, a probabilistic model in particular areas has been introduced with the aim of improving the matching performance. According to the curvature distribution of each object, an extension of the point cloud is performed in those areas where discontinuities appear or abrupt changes are distinguished. Thus, the curvature extension algorithm will fill those regions close to the borders of the view, introducing a set of synthetic points called *guess* points that will create an extension of the *observable* point cloud. Those *guess* points will be introduced according to a probabilistic

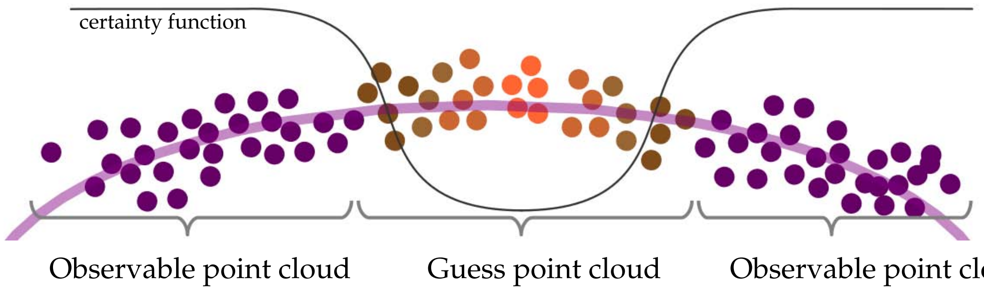


Figure 7.2: *Curvature extension.* Graphical explanation of the curvature hypothesis to fill holes in areas where discontinuities appear or abrupt changes are distinguished. Certainty function will give a different weight to guess points according to a exponential function.

model so the importance of guess points is related to their distance to the closest observable element in the *observable* point cloud. In Figure 7.2 a visual explanation of the hypothesis is performed. As long as the certainty decreases, points become brighter for a better understanding. Certainty function returns a single unit within the observable point cloud and decreases exponentially as long as it covers the hole or discontinuity.

In such a way, the view is splitted into several slices similar to a level set method. For each slice, the curvature distribution is examined and an extra collection of new points is adhered if necessary on the frontiers. Figure 7.3 shows a point cloud and a set of crossing planes. In first place, the point cloud is sliced every $\sigma_{slice} = 2mm$ from the ground upward along the $Y-$ axis. Curvature is then tracked for each slice, detecting those areas where the sensor has not acquired any information and, if necessary, it is extended with the result of the curvature estimation as explained in Algorithm 3 and represented in Figure 7.2.

Figure 7.4 represents the set of slices that are created for each model. The set of slices $\mathcal{S}$ is defined as

$$\mathcal{S} = \{\varsigma_j : \varsigma_j \in \mathcal{V}\} \tag{7.6}$$

where ς_j represents each slice. Mathematically speaking, a slice will be generated such as

$$\varsigma_j = \left\{p_i \in \mathcal{R}^3 \wedge d(p_i, \pi_j) \leq \sigma_{slice}\right\} \tag{7.7}$$

Algorithm 3 Curvature expansion. *Generates a point cloud where holes are filled with extra points to improve the matching process*

```
Require: Point cloud $P$                                   ▷ Original Point cloud
Require: New extra point cloud $N$
Require: Threshold for sliding $\sigma_{slice}$
 1: $S \Leftarrow \{\}$                                    ▷ Clear the list of slices
 2: $(p_{min}, p_{max}) \leftarrow$ getMaxMin($P$)
 3: height = $p_{min}(y) <$
 4: while height $< p_{max}(y)$ do
 5:     $s \leftarrow$ getClosestPoints($P$, height, $\sigma_{slice}$)
 6:     height$\leftarrow$ $height + \sigma_{slice}$
 7:     if size($s$)$> 0$ then
 8:         $c \leftarrow$ getCurvature($s$)               ▷ Curvature distribution
 9:         $\tilde{s} \leftarrow$ interpolateCurvature($c$)    ▷ New slice interpolation
10:         $S \leftarrow S + \tilde{s}$
11:     end if
12: end while
13: $N \leftarrow$ reconstructSlices($S$,$p_{min}(y) <$, $p_{max}(y) <$, $\sigma_{slice}$)    ▷ Recreate point cloud
14: return $N$
```

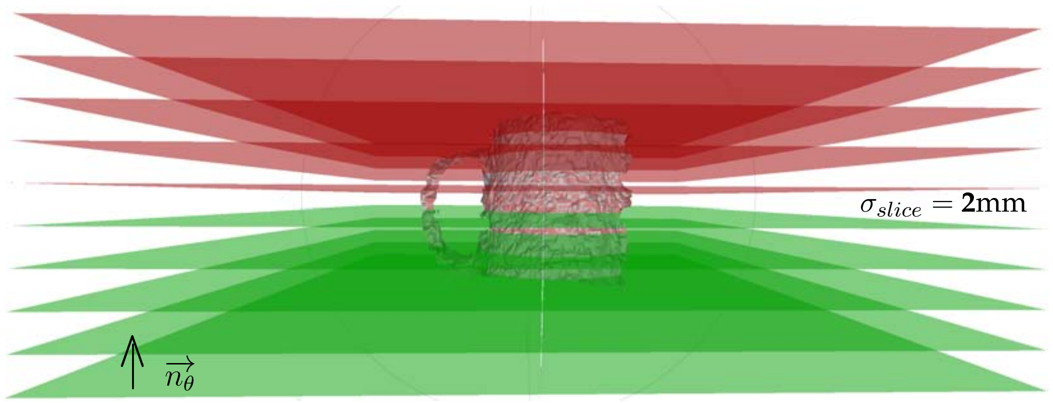


Figure 7.3: *Curvature extension.* A set of 10 planes cross the cluster and extract the distribution curvature on each stage. If necessary, extra points will be attached into the borders for a better matching estimation. If the number of points close to a single slice is lower than a certain quantity, the slice will be discarded.

Taking Figure 4.11 from Chapter 4 in mind, $\overrightarrow{n_\theta}$ symbolizes the normal direction of the supporting plane. Thus, the set of planes Π are determined as

$$\Pi = \{\pi_i : \overrightarrow{n_{\pi_i}} = (\pi^{(j)}, \pi^{(j)}, \pi^{(j)}), \overrightarrow{n_{\pi_i}} \parallel \overrightarrow{n_\theta}\} \tag{7.8}$$

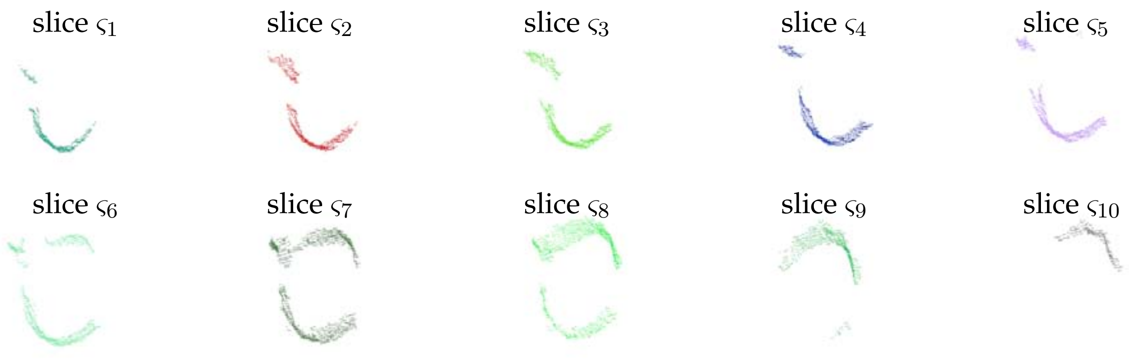


Figure 7.4: *Curvature extension.* Display of the ten slices of the model for the previous figure. Each slice is then processed independently, analyzing its curvature with respect to its barycenter and filled with new points if necessary.

Once the model is sliced, each section is processed independently following the steps explained in the previous Algorithm 3: curvature is estimated for each slice, then curvature is interpolated and new points are added according to the new members. Then the point cloud is updated and lately a filter is passed so any spurious point is removed. The demanded condition to introduce new points in the model is that new members have to follow the same curvature distribution as their neighbors. In order to do so, curvature distribution will be computed near to those areas of discontinuity and then new elements will be inserted. Finally, a measure of certainly will be attached to each element according to their closest distance to any observed member.

7.2.1 Synthetic Slice Initialization

The first thing to do is to create an empty point cloud with 360 points representing each circumference degree. Replicate those points included in the slice filling the positions as appropriate. Some positions will be filled while others will remain empty. Figure 7.5 represents the array of the projected points inscribed in a circumference and the histogram representation of the radius of curvature with respect to the barycenter of the set. It can be noticed that those empty areas are exactly the uncertain holes that the algorithm is interested in filling with new points.

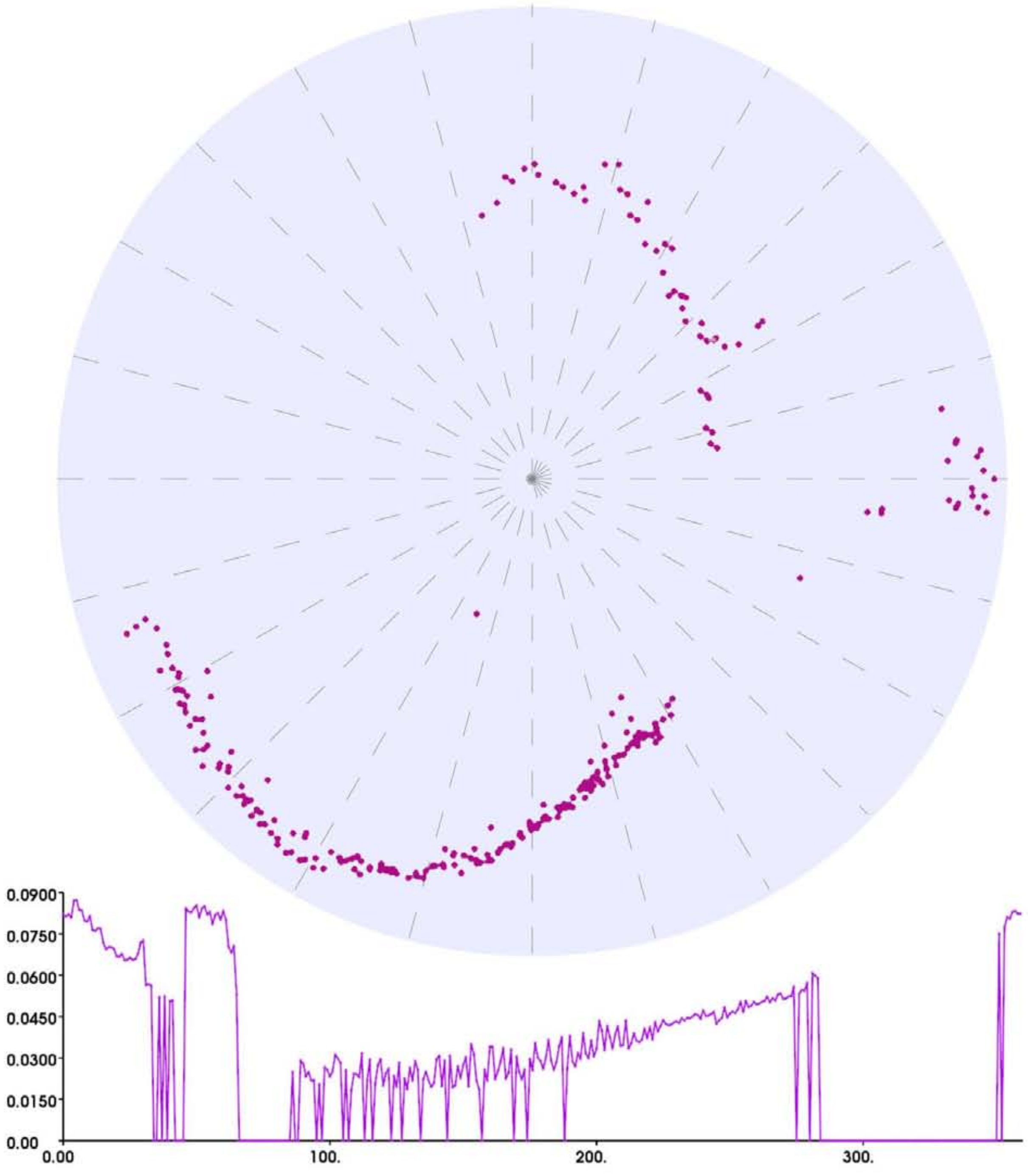


Figure 7.5: *Synthetic slice initialization.* Representation of a 360-length slice and its radius of curvature histogram

7.2.2 Curvature Estimation

The next step consists on generating a curvature histogram. This histogram will address the curvature for each element of the slice. Two-dimensional curvature for a certain point p_i has been established as the angle between two vectors: on the one hand, the vector joining the barycenter $\mathcal{M}_{CoG}$ and each point p_i and, on the other hand, the linear regression formed by the H nearest of p_i. Figure 7.6 explains the geometric meaning of the proposed curvature.

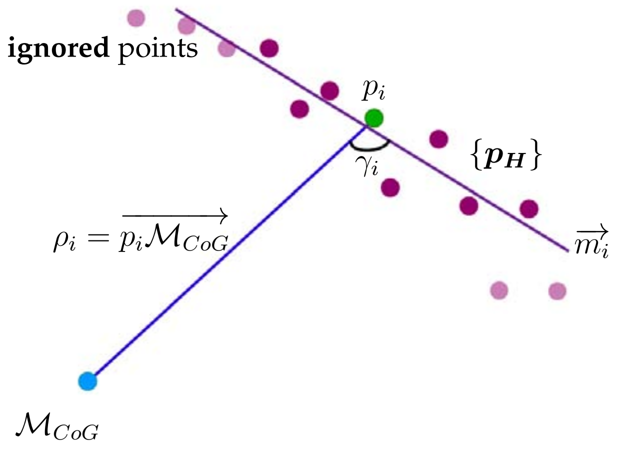


Figure 7.6: *Computation of curvature.* Curvature for two dimensions has been defined as the angle between the vector that goes from the barycenter to each point (so called ρ_i) and the linear regression formed by the nearest neighbors of that point (so called m_i)

The curvature is therefore the angle formed by the two vectors γ_i. This is computed as

$$\gamma_i = \cos^{-1}\left(\frac{\overrightarrow{p_i\mathcal{M}_{CoG}} \cdot \overrightarrow{m_i}}{\|\overrightarrow{p_i\mathcal{M}_{CoG}}\| \cdot \|\overrightarrow{m_i}\|}\right) \tag{7.9}$$

and vector m_i has been computed by least squares using the H nearest neighbors of p_i so that

$$\overrightarrow{m_i} = \beta_i \cdot p_i(x) + \tau_i \tag{7.10}$$

being β_i the slope associated to the linear regression of the $H-$ nearest neighbors $\{\boldsymbol{p_H}\} = \{p_1(x_1, y_1, z_1), p_2(x_2, y_2, z_2), \dots, p_H(x_H, y_H, z_H)\}$ and τ_i its intercept. They are defined as

$$\beta_i = \frac{H \cdot (\sum_{h=1}^{H} x_h \cdot z_h) - (\sum_{h=1}^{H} x_h)(\sum_{h=1}^{H} z_h)}{H \cdot (\sum_{h=1}^{H} x_h^2) - (\sum_{h=1}^{H} x_h)^2} \tag{7.11}$$

and the intercept τ_i becomes

$$\tau_i = \frac{(\sum_{h=1}^{H} z_h) \cdot (\sum_{h=1}^{H} x_h^2) - (\sum_{h=1}^{H} x_h) \cdot (\sum_{h=1}^{H} x_h \cdot z_h)}{H \cdot (\sum_{h=1}^{H} x_h^2) - (\sum_{h=1}^{H} x_h)^2} \tag{7.12}$$

However, to perform a noise reduction and obtain a smoother curvature estimation, a Moving Average Filter is performed over the γ_i element with the identical number of neighbors H by means of the following operation [10]:

$$\widetilde{\gamma_i} = \frac{1}{H} \sum_{h=-H/2}^{+H/2} \gamma_{i+h}, i = \{\frac{H}{2}, \frac{H}{2}+1, \frac{H}{2}+2, \ldots, 360 - \frac{H}{2}\} \tag{7.13}$$

Therefore, the set of filtered curvatures $\widetilde{\gamma}$ is an array of 360 pair elements with a curvature $\widetilde{\gamma_i}$ and a radius ρ_i associated to each angle:

$\widetilde{\gamma_1}$	$\widetilde{\gamma_2}$	$\widetilde{\gamma_3}$	$\ldots$	$\widetilde{\gamma_{358}}$	$\widetilde{\gamma_{359}}$	$\widetilde{\gamma_{360}}$
ρ_1	ρ_2	ρ_3	$\ldots$	ρ_{358}	ρ_{359}	ρ_{360}

7.2.3 Hole Filling using Side by Side Interpolation

Once the list of curvatures has been computed, it can be represented in a histogram. The reader can note that there might exist certain holes in the histogram desirable to be filled. To do that, a linear interpolation has been chosen at each side of each hole. Therefore, analyzing the trend of the curvature near the hole frontiers, it is possible to extrapolate to new points. In this experiment, the number of supplement points has a maximum of 10 units on each side. Using a linear interpolation between the opposite ends of the hole, new curvatures are attached, as it is represented in Figure 7.7. Red elements represents the original $\widetilde{\gamma}$ array while blue elements symbolizes the extrapolated curvature elements.

[10]Notice that the first and last $H/2$ elements of the filtered curvature array have been filled with the original curvature values.

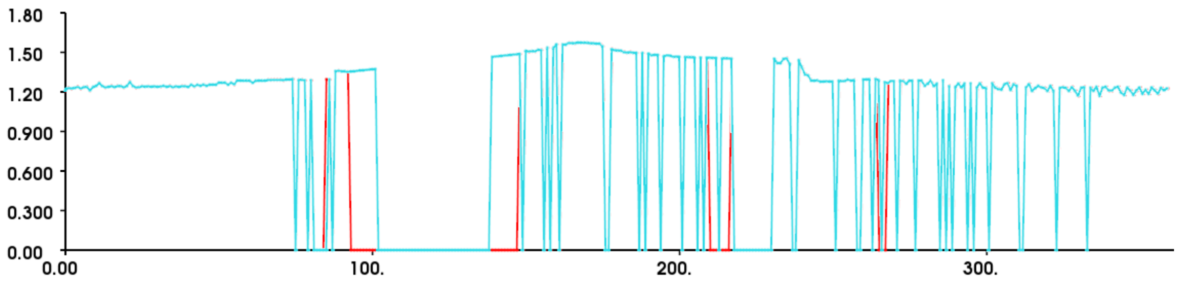


Figure 7.7: *Curvature extrapolation.* Representation of the original curvature histogram in red and the extrapolation near the empty areas in blue. Note that the filter only attacks to the empty areas growing at once from both hole sides.

Once the curvature distribution is filled with new points, it is necessary to introduce the respective position of those new points inside the radius distribution. To do that, a simple linear transformation is performed so that the new ρ_i values are interpolated. Suppose the following situation

$\dots$	$\widetilde{\gamma}_{p-1}$	$\widetilde{\gamma}_{p}$	$\widetilde{\gamma}_{p+1}$	$\dots$	$\widetilde{\gamma}_{m}$	$\dots$	$\widetilde{\gamma}_{q-1}$	$\widetilde{\gamma}_{q}$	$\widetilde{\gamma}_{q+1}$	$\dots$
$\dots$	ρ_{p-1}	ρ_p	0	0	ρ_m	0	0	ρ_q	ρ_{q+1}	$\dots$

The addition has be performed using the following equation

$$\rho_m = \rho_p + \frac{(\rho_q - \rho_p)}{(\widetilde{\gamma}_q - \widetilde{\gamma}_p)} \cdot (\widetilde{\gamma}_m - \widetilde{\gamma}_p) \tag{7.14}$$

Now the model is formed by two point clouds: the original observed model and the estimation as it is displayed in Figure 7.8.

7.3 Yaw Initialization

Matching two point clouds might demand a lot of time. For this reason, the initialization process for global optimization functions is always appropriated. Solutions to this initialization problem depend on the nature of the process and also on the information addressed on each case, such as in Smith *et al.* [134], who extract SIFT features to perform the initial matching problem. Evolutionary algorithms have been applied to improve the domain of convergence of ICP, but the initialization process will accelerate the complete process.

This research addresses the problem exploiting the geometric conditions of the object. Taking in mind that the object is lying on the supporting plane, the restrictions affect uniquely the rotating angles. That is, pitch and roll angles are restricted

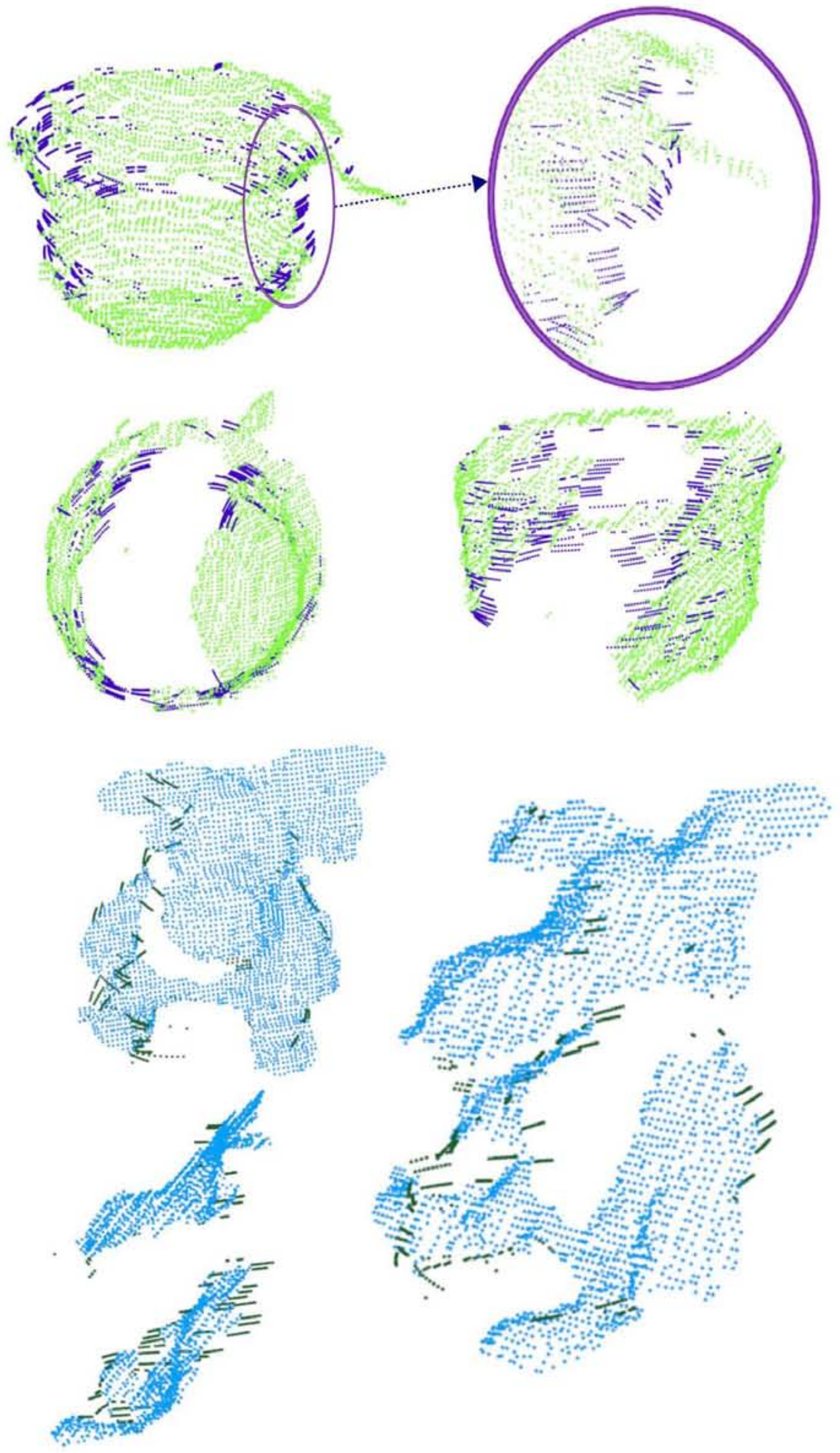


Figure 7.8: *Curvature extension.* Representation of two objects, a mug and a teddy, from different perspectives. Light points belong to the original point clouds while dark points represent the additional point clouds in charge of improving the scan matching.

and only yaw can be altered as far as the object rotates in the scenario. For this reason, and giving importance to the fact that the individual views of the object are rotated randomly, a yaw initialization process is performed in a few steps.

1. Project the model and view in the supporting plane.

2. Extract the concave hull representation of both model and view.

3. Rotate the view over the yaw angle and calculate the matching distance for each angle.

4. Apply a reduction factor for each angle giving more importance to those angles near the origin.

5. Extract the minimum matching error angle and set the initial transformation to this value.

This procedure has to be performed once for each view. However, it can be easily improved saving the last iteration point clouds in order to reduce the processing intervals. Figure 7.9 represents the initialization process and the resulting histogram of distances for each angle. The histogram presented accumulates the matching error for each angle by an iterative comparison. Furthermore, this histogram is convoluted with a triangular weight function giving more importance to those angles near the origin (it is supposed that the consecutive views have suffered the minimum variance) with a 30% decay. After all, the minimum angle in the processed histogram is then selected (49^o) as the best initial yaw angle with a top-view matching error 0.0087788.

As a general rule, the triangular filter will return the same result as the original min-function. However, this filter might reduce the number of false positives in those situations where the object is highly rotated but a pronounced peak exists. Therefore, with this improvement the matching process increments its performance. This reduction factor does not necessarily needs to be a linear function and can be extended to exponential or polynomial functions easily. In order to reduce processing time, the weight values are stored in a look-up table. Figure 7.10 represents the stages of the initialization and the result obtained. The two point clouds represent the model and the view in the original position and after the initialization process.

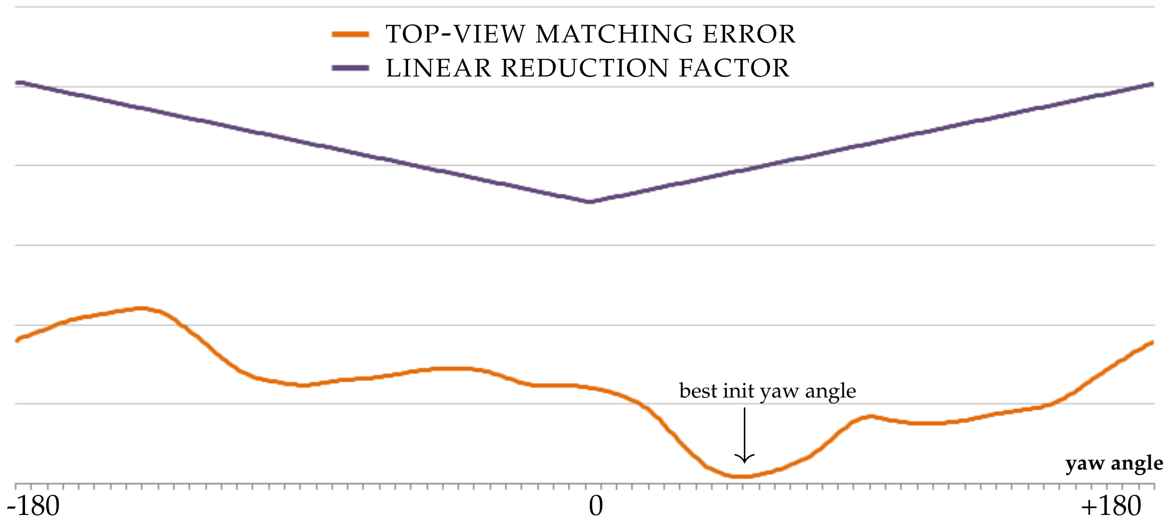


Figure 7.9: *Registration initialization.* Taking in mind that objects are lying on a horizontal plane, roll and pitch angles can be restricted to get a fast rough initialization for the matching procedure.

7.4 Evolutionary Optimizator in 3D Data Registration

One of the fundamental parts of this chapter focuses on the design of an application of evolutionary optimization in 3D data registration. Considering the problem of Euclidean alignment of two arbitrarily oriented, partially overlapping surfaces represented by measured point sets contaminated by noise and outliers, the proposed solution must converge into a global minimum overlapping the set of views as much as possible. However, this work is focused on small datasets that contain a limited number of views of the object (most of the times the number of views is less than a dozen). That means that classical registration algorithms are worthless and in most cases ineffective due to their ICP-based nature. Furthermore, attending to the necessities explained at the beginning of the chapter, the proposed system will exploit the view features such as color information and spatial distribution of the point cloud among others.

Thus, in this thesis the registration problem has been addressed to reconstruct point clouds extracted from a structured light sensor (as explained in Chapter 2) by means of an evolutionary optimization technique. The usage of evolutionary optimization techniques turn out to be a valid decision in those problems that require global optimization such as multi modal functions with several local minimums. Evolutionary optimizers are probabilistic, avoiding derivatives to estimate the best

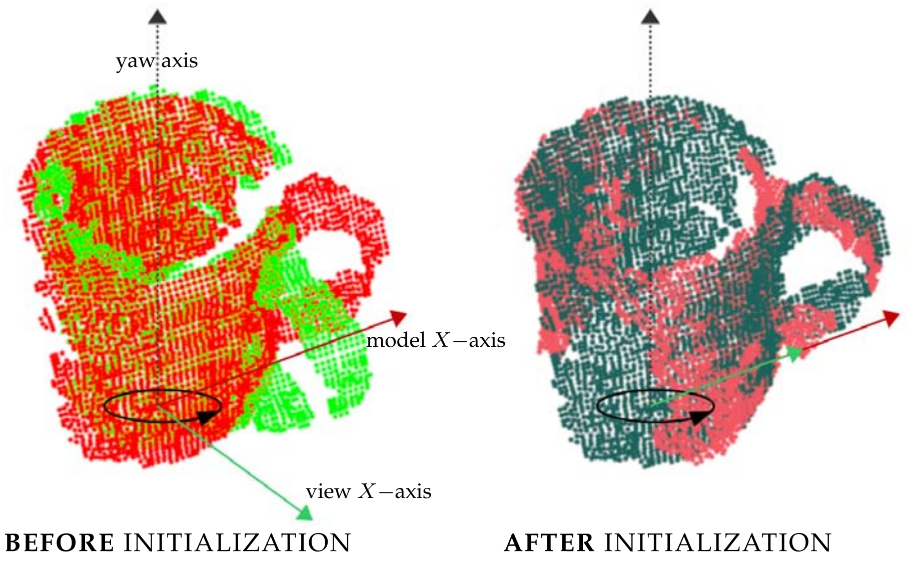


Figure 7.10: *Registration initialization result.* On the left the model (red) and the view (green) point clouds are displayed superposed. After the initialization histogram, the view point cloud is rotated over the yaw axis 49^o and the result is displayed on the right. This step helps the scan matching algorithm to evolve and converge much faster.

solution to the correspondence problem. The usage of evolutionary algorithms for global optimization problems has been addressed regularly in 3D location problem such as Martín-Monar *et al.* [135] or Moreno *et al.* [136], where a SLAM filter is designed using a non linear evolutionary filter called *Evolutive Localization Filter* (ELF) that searches stochastically along the state space for the best robot pose estimate. Others such as Vahdat *et al.* [137] compare the genetic proposal with particle swarm optimization obtaining encouraging results. Following the same line, Martín-Monar [34] presented a DE-based Scan Matching and Feature-based Loop Detection. He developed a mapping algorithm that recreated maps from local views. This section illustrates the optimizer and their most representative features.

Every particle-based evolutionary algorithm evolves through time to the lower cost value minimizing a custom fitness function. The design of the fitness function consumes the largest efforts as it mostly determines the performance of the whole optimizer and it will be explained in the subsequent section. In any case, the fitness function represents the matching error between two scans. The method is population-based where each population member represents a rigid transformation between the two point clouds. Any geometric relation (Euclidean rigid transformation) between two surfaces can be interpreted by six parameters or DOF. Each set of parameters is settled as a population member. Each parameter then corresponds to one of the genes in the candidate. They can be enumerated as shown in Table 7.1.

POPULATION FORMULATION			
Symbol	Description	Symbol	Description
T_x	Translation of $x-$axis	R_x	Rotation about $x-$axis
T_y	Translation of $y-$axis	R_y	Rotation about $y-$axis
T_z	Translation of $z-$axis	R_z	Rotation about $z-$axis

Table 7.1: Population definition formulation for the fitness function

Therefore, T_x, T_y and T_z represent the translation genes while the rotation matrices R_x, R_y and R_z are formed by the genes α, β, γ, so that a rigid transformation can be defined as

$$T = R_x \cdot R_y \cdot R_z \cdot S \tag{7.15}$$

where

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.16}$$

$$R_y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.17}$$

$$R_z(\gamma) = \begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.18}$$

and the translation matrix corresponds to

$$S(T_x, T_y, T_z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix} \tag{7.19}$$

Therefore, the population set will evolve through time to the optimal rigid transformation that minimizes the fitness function. The stochastic search of the robot's coordinates is done using the DE method proposed by Storn and Price [138] for global optimization problems over continuous spaces. In this section the basics of this algorithm will be explained. If the reader considers interesting to get further into the problem, an elaborated demonstration of Differential Evolution technique can be found in Appendix A. Algorithm 4 explains the details of the method.

In the first place, a description of the environment and its highlights has to be performed. Each view of a specific object corresponds to a point cloud that contains between 1000 and 5000 points. Obviously its density depends on the actual size of the observed object. Each point contains a three-dimensional position in meters and a texture value RGB encoded. As it has been explained already, in order to determine the correspondence between the two point clouds, six coordinates must be estimated, defining a state space with six DOF ($x-$ displacement, $y-$ displacement, $z-$ displacement, row, pitch and yaw).

Search starts with a population of N_P candidates or candidates where each one represents a possible solution. As stated before, any rigid transformation has 6 DOF if is applied in a 3D space. Thus, a candidate pop_i^k $D-$dimensional is represented as:

$$pop_i^k = (x_i^k, y_i^k, z_i^k, \alpha_i^k, \beta_i^k, \gamma_i^k) \tag{7.20}$$

where sub-index i represents the element number and super-index k the iteration. Initial population can be set stochastically around the initial position or forced to a certain value if there exist any clue about its value (Line #2). In this case, the yaw initialization angle estimated in section 7.3 will be introduced as a first guess. With no doubt, the selection of a good transformation involves in a better and faster convergence as the number of iterations increases. For each initial candidate, a cost value is associated by evaluating the candidate within the fitness function (Line #3). The full description of the fitness function will be presented in the following section. The main loop starts in Line #5 and it is replicated until the maximum number of iterations $max\_iterations$ is reached. However, this loop can be broken

Algorithm 4 Differential Evolution. *Computes the rigid transformation between two point clouds using evolutionary optimization*

```
Require: Model point cloud $\mathcal{M}$
Require: View point cloud $\mathcal{V}$
 1: for $(i = 1 : N_P)$ do                                              ▷ Population Initialization
 2:     $pop_i^1 \Leftarrow init\_population(data\_initial\_pose)$     ▷ First population generation
 3:     $error^0[i] \Leftarrow fitness(\mathcal{M}, \mathcal{V}, pop_i^1)$     ▷ First cost computation
 4: end for
 5: for $(k = 1 : max\_iterations)$ do
 6:     for $(i = 1 : N_P)$ do
 7:         $v_i^k \Leftarrow pop_a^k + F \cdot (pop_b^k - pop_c^k)$     ▷ Mutation
 8:         for $(j = 1 : D)$ do                                        ▷ $D$ = candidate dimension
 9:             $u_{i,j}^k \Leftarrow v_{i,j}^k, \forall p_{i,j}^k < Crossover$     ▷ Crossover
10:             $u_{i,j}^k \Leftarrow pop_{i,j}^k, \forall p_{i,j}^k \geq Crossover$
11:         end for
12:         $error^k[i] \Leftarrow fitness(\mathcal{M}, \mathcal{V}, pop_i^k)$     ▷ Calculate cost
13:         if $(e^k[i] < e^{k-1}[i] \cdot \tau)$ then                  ▷ Selection with thresholding $\tau$
14:             $pop_i^{k+1} \Leftarrow u_{i,j}^k$
15:         end if
16:     end for
17:     $index_{best} \Leftarrow minimum(e^k)$                          ▷ Find best candidate
18:     $best_{energy} \Leftarrow pop^k[index_{best}]$                  ▷ Update best candidate
19:     if $(convergence(best_{energy})$ is $true)$ then                ▷ Convergence condition
20:         return $(best_{energy}, k)$                                 ▷ Return the optimal candidate
21:     end if
22: end for
23: return $(best_{energy}, k)$                                         ▷ Return the best candidate found
```

if the convergence function of Line #20 is accomplished.

For each iteration k a search is performed. This evolutionary search starts in Line #6 and generates a new population for the upcoming iteration. In order to find the optimal solution, candidates are perturbed producing a variation v_i for each candidate using the following formula:

$$v_i^k = pop_a^k + F \cdot (pop_b^k - pop_c^k) \tag{7.21}$$

where pop_a^k, pop_b^k and pop_c^k are three randomly selected candidates at k iteration. The indexes a, b and c are always different to the actual index i. The constant F

is a real number. It represents an amplification factor that determines the intensification of the differential variations $(pop_b^k - pop_c^k)$ and, as a consequence, the population evolution rate. It has an empirical range of value $(0, 1^+)$ as it has not been demonstrated yet that $F > 1$ achieves the optimal solution of the optimization problem. It is feasible to solve optimization problems with $F > 1$, but every case that is successfully solved always obtains a more suitable performance with $F < 1$. Furthermore, a value of $F = 1$ does not make sense as there might exist infinite combinations of vectors with the same value. Depending on the three elements selected during the perturbation process, different results are achieved [11].

One of the most valuable features of evolutionary strategies is the simplicity to control and change the behavior of the evolution process. At this point, DE-based algorithms enlarge the diversity of the population by introducing a crossover Cr factor. Trial vector is represented as

$$u_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,D}^k) \tag{7.22}$$

where each parameter depends on the crossover probability given by

$$u_{i,j}^k = \begin{cases} v_{i,j}^k & \text{if} p_{i,j}^k < Cr \\ pop_{i,j}^k & \text{otherwise} \end{cases} \tag{7.23}$$

A random value from the interval $[0, 1]$ is set on $p_{i,j}^k$ for each parameter j of the candidate i at k iteration and it is updated for each trial vector i. Thus, any new candidate $u_{i,j}^k$ is compared to pop_i^k (Line #13) to settle which element is better and then will become a member of the next generation $k+1$. In case that the evaluation of the element $u_{i,j}^k$ on the fitness function returns a better result than the actual pop_i^k, it is replaced by $u_{i,j}^k$ (Line #14). Otherwise the actual member is preserved until the successive generation. Finally, the best index and energy values are updated until the next iteration (Lines #17-18). If the best energy satisfies the convergence conditions, the loop is finished. Otherwise it continues forward to the next iteration.

As a result, the algorithm returns the best candidate found and also the number of iterations k required to converge. It yields the most suitable transformation T that approaches both point clouds $\mathcal{M}$ and $\mathcal{V}$ (Line #23).

7.4.1 Fitness Function

With the idea of converging into an optimal global minima avoiding local sinks, the fitness function takes an important role in the whole correspondence process.

[11] Appendix A explains in deep the behavior of DE

The function here presented is based on the original idea expressed in Equation 6.1. That expression can be analyzed separately in two parts: the weights $w_{i,j}$ in charge of the quality of the matching and the proper comparison for this iteration.

$$e(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \underbrace{w_{i,j}}_{quality} \overbrace{\| a_i - (\boldsymbol{R} \cdot b_i + \boldsymbol{t}) \|^2}^{comparison} \tag{7.24}$$

The function given in the Equation 7.24 has been boosted with several improvements based on the requisites and exigencies of the faced problem: models with a high degree of symmetry, partially occluded, resting on a common supporting plane with a few number of views. Tunning the quality term in the right way will give out better capabilities and efficacy. The mentioned cost function can be rewritten excluding all the non-corresponding points such as

$$e(\boldsymbol{R}, \boldsymbol{t}) = \sum_{c \in C} d(a_c, b_c) \tag{7.25}$$

being C the total number of correspondences and c an index covering each of them. All the values a_c and b_c represent each of those corresponding values and $d(\cdot, \cdot)$ a custom distance function that can hold any desired feature according to any of both point clouds. This function error represents a generic error assigning a specific value for each pair of correspondences.

With this in mind, in terms of view $\mathcal{V} = \{v_i\}_{i=0}^{N_\mathcal{V}}$ and model $\mathcal{M} = \{m_j\}_{j=0}^{N_\mathcal{M}}$, Equation 7.25 representing the error can be rewritten as

$$e(\boldsymbol{R}, \boldsymbol{t}) = \sum_{c \in C} d(\mathcal{V}_c, \mathcal{M}_c) \tag{7.26}$$

where

$$\mathcal{V}_c = \{\ v_i\} \,/\, \| v_i - (\boldsymbol{R} \cdot m_j + \boldsymbol{t}) \|^2 < \tau \tag{7.27}$$

$$\mathcal{M}_c = \{\ m_j\} \,/\, \| v_i - (\boldsymbol{R} \cdot m_j + \boldsymbol{t}) \|^2 < \tau \tag{7.28}$$

with $0 < i < N_\mathcal{V}$ and $0 < j < N_\mathcal{M}$. As stated before, the previous equation represents the summation of a set of distance errors for features concerning to both point clouds.

Local distance error

The local distance error represents the *n*-dimensional distance between the query point $\mathcal{V}_i$ and the average location of the H nearest neighbors in a radius $\aleph$ in a *k-d* tree of $\mathcal{M}$. Applying the same concept as in Silva *et al.* [130], the local distance error for one element is defined as

$$d(v_i, m_j)^2 = \frac{1}{H}\sum_{h=1}^{H} \rho(r_h) \tag{7.29}$$

where

$$\rho(r_h) = \begin{cases} r_h & \text{if } r_h < \aleph \\ \aleph & \text{otherwise} \end{cases} \tag{7.30}$$

and r_h represents the distance

$$r_h = \|m_j - (\boldsymbol{R} \cdot v_h + \boldsymbol{t})\|^2 \tag{7.31}$$

being v_h the closest point in $\mathcal{V}$ to the point m_j. For this work, H has been established in a maximum of 10 neighbors using norm-2 distance. This operation does not represent a high computational cost because the *k-d* tree returns immediately the set of H points in the desired error space. Nevertheless, measuring the distance to an average of H elements smooths the distance error returning a more confident value of the local error. Therefore, the local distance error will gather the individual errors such as

$$d_{LDE}(\mathcal{V}_c, \mathcal{M}_c)^2 = \sum_{c \in C} d(v_c, m_c)^2 \tag{7.32}$$

Based on the proposals by Chetverikov in [139] and [140], trimmed ICP (TrICP) has been implemented on the local distance error as it is applicable to overlaps under 50% and robust to erroneous and incomplete measurements. The concept is straightforward in essence: sortws ascending local distances and only selects the first N_{po} minimizing the trimmed mean-squared error. Therefore,

$$\left\{d_{LDE}(\mathcal{V}_c, \mathcal{M}_c)^2\right\}_1^{N_{po}} : d_{LDE}(\mathcal{V}_1, \mathcal{M}_1)^2 \leq d_{LDE}(\mathcal{V}_2, \mathcal{M}_2)^2 \leq \ldots \leq d_{LDE}(\mathcal{V}_{N_{po}}, \mathcal{M}_{N_{po}})^2 \tag{7.33}$$

$$d^*_{LDE}(\mathcal{V}_c, \mathcal{M}_c) = \frac{1}{N_{po}}\sum_{i=1}^{N_{po}} d_{LDE}(\mathcal{V}_c, \mathcal{M}_c)^2 \tag{7.34}$$

However, the selection of N_{po} is not immediate since it directly affects the overlap, and in this study will be set to one half of the whole data set.

Global distance error

Up to now, the designed fitness function is intended to attend to local disparity between point cloud elements but also a global distance representation might be desired. Since the local distance error is focused on individuals, the global distance error is meant to measure the behavior of the transformation from a collection point of view. Therefore, the application of statistical tools such as median or average make sense.

Zhang *et al.* [112] extended ICP to include robust statistics and adaptive thresholding to handle outliers and partial occlusions. Masuda and Yokoya [126] used ICP with random sampling and a least median square error measurement that is robust to a partially overlapping scene. Chen and Medioni [141] separately refined an approach similar to ICP, which minimizes the sum of squared distance between scene points and a local planar approximation of the model. Correspondences are formed by projecting the scene points onto the model in the direction of their normal vectors rather than selecting the closest point. Dorai *et al.* extended the method of Chen and Medioni to an optimal weighted least squares framework in [142] by deriving a minimum variance estimator (MVE) for computing the view transformation parameters accurately from two point clouds. Furthermore, Chow *et al.* [105] proposes to minimize the median of the transformation instead of the Euclidean distances among all correspondence pairs

$$\begin{matrix}\min \\ T\end{matrix} \sum_{c \in C} \text{Median} \left\| T(\mathcal{V}_c) - \mathcal{M}_c \right\|^2 \tag{7.35}$$

where T is the 6 DOF transformation such that $T(a) = \boldsymbol{R} \cdot a_i + \boldsymbol{t}$. Since the median error function is non-linear, it is therefore non-differentiable in general and must be optimized using a GA.

Normal alignment error

One of the extracted features of point clouds are normals associated to each element. The usage of normals improve the stability of the algorithm as it represents the principal direction of the surface on each point. Extracting the surface normals as explained in Section 5.2 it is possible to compare point-wise the direction of the query point and the model candidate. A shape descriptor, called *surfel*[12] presented by Wahl *et al.* [143] will be used as a reference. Furthermore, their descriptor has been demonstrated to obtain robust recognition rates using Kullback-Leibler and

[12]A *surfel* is commonly named to any feature that describe a local surface curvature

likelihood matching. This surflet definition is widely used in keypoints extraction, addressed initially by Drost *et al.* in [144] and used afterwards in Wahl *et al.* [143] and Rusu *et al.* [145] and more recently in Torsten *et al.* [146], where they use it for an entropy-based interest operator that selects distinctive points on surfaces.

If $\Xi(\mathcal{V}) = \{n_i^{\mathcal{V}}\}_{i=0}^{N_{\mathcal{V}}}$,represents the normal map for the view $\mathcal{V}$, the expression $\Xi(\mathcal{M}) = \{n_j^{\mathcal{M}}\}_{j=0}^{N_{\mathcal{M}}}$ represents the equivalent in $\mathcal{M}$. Let us assume that a correspondence c is found between m_c which has a normal associated $n_j^{\mathcal{M}}$ and v_c with a normal associated $n_i^{\mathcal{V}}$. A reference frame is created so

$$\begin{aligned} u &= n_i^{\mathcal{V}} \\ v &= \frac{d \times u}{\|d \times u\|} \\ w &= u \times v \end{aligned} \tag{7.36}$$

where $d = m_c - v_c$. The relative angles are then

$$\omega = \tan^{-1}\left(\frac{w \cdot n_j^{\mathcal{M}}}{u \cdot n_j^{\mathcal{M}}}\right) \tag{7.37}$$

$$\psi = v \cdot n_j^{\mathcal{M}} \tag{7.38}$$

In order to describe curvature in the local vicinity of an correspondence point, a histogram of surfel pair relations from neighboring surfels is created. A histogram of 90 values is created for each correspondence point (see Figure 7.11). The first half will contain the information relative to ω, while the last half will comprehend the values of ψ. As ω and ψ values variate between 0 and 360^o in the sexagesimal numeric system, a bin will correspond to an interval of 8^o for both variables. For each correspondence, a surface histogram will be extracted according to their surrounding normal values. A surfel histogram $\mathcal{G}$ associated to a certain correspondence $c \in C$ for the model point cloud can be computed as

$$\mathcal{G}(\Xi(v_c)) = \left\{ \underbrace{g_{v_c}^{(1)}, g_{v_c}^{(2)}, \ldots, g_{v_c}^{(45)}}_{corresponds\, to\, \omega}, \underbrace{g_{v_c}^{(46)}, \ldots, g_{v_c}^{(90)}}_{corresponds\, to\, \psi} \right\} \tag{7.39}$$

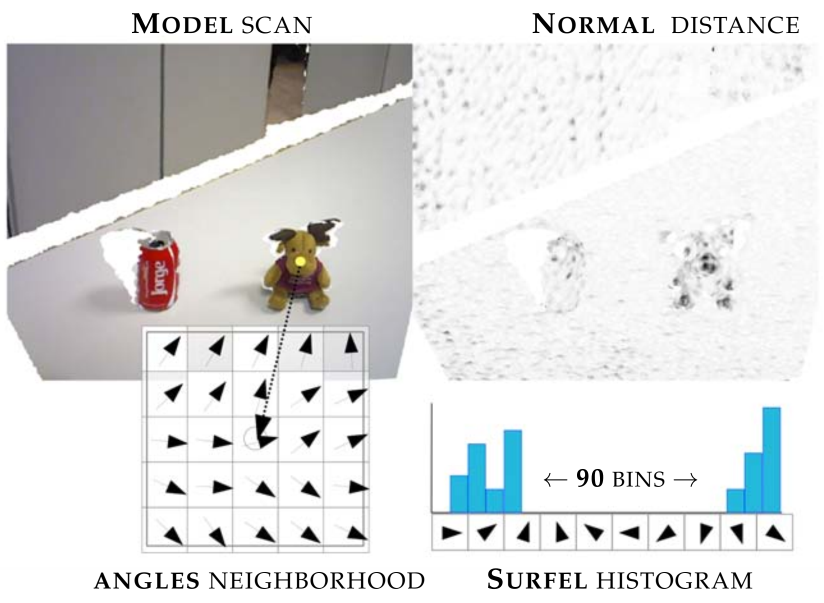


Figure 7.11: *Normal distance* Display of the normal distance (on the right) based on the input 3D map (on the left). Each normal is extracted and a surfel histogram is computed for each pixel looking at its neighbors. Search is performed using Kullback-Leibler divergence as distance metric.

and for the same correspondence in the view point cloud will be

$$\mathcal{G}(\Xi(m_c)) = \left\{ \underbrace{g_{m_c}^{(1)}, g_{m_c}^{(2)}, \ldots, g_{m_c}^{(45)}}_{corresponds\,to\,\omega}, \underbrace{g_{m_c}^{(46)}, \ldots, g_{m_c}^{(90)}}_{corresponds\,to\,\psi} \right\} \tag{7.40}$$

where $g_{m_c}^{(1)}$ corresponds to the first bin of the intensity histogram associated to m_c. The summation of bins for both distributions will be equal to one, so

$$\sum_{k=0}^{90} g_{m_c}^{(k)} = \sum_{k=0}^{90} g_{v_c}^{(k)} = 1 \tag{7.41}$$

In order to compare two surfels, Kullback-Leibler divergence method presented in [147] will be applied as recommended in [146]. According to that, the distance between two distributions P, Q is

$$D_{PQ} = \sum_{i=1}^{N} p_i \cdot \log \frac{p_i}{q_i} \tag{7.42}$$

It is important to pay attention to those cases where $p_i = 0$ or $q_i = 0$ because it yields discontinuities. In these cases, the addend will be discarded. With this in mind, the distance between to correspondences will be

$$d(v_c, m_c) = \sum_{k=1}^{90} \left(g_{m_c}^{(k)} \cdot \log \frac{g_{m_c}^{(k)}}{g_{v_c}^{(k)}} \right) \forall g_{m_c}^{(k)}, g_{v_c}^{(k)} \neq 0 \tag{7.43}$$

and for the whole list of correspondences C

$$d_{NAE}(\mathcal{V}_i, \mathcal{M}) = \sum_{c \in C} d(v_c, m_c) \tag{7.44}$$

that according to 7.43 corresponds to

$$d_{NAE}(\mathcal{V}_i, \mathcal{M}) = \sum_{c \in C} \sum_{k=1}^{90} \left(g_{m_c}^{(k)} \cdot \log \frac{g_{m_c}^{(k)}}{g_{v_c}^{(k)}} \right) \forall g_{m_c}^{(k)}, g_{v_c}^{(k)} \neq 0 \tag{7.45}$$

Intensity correlation error

The last addend is related to the texture disparity of the point clouds being matched. This feature attends to detect the similarity in those objects with noticeable changes in texture or intensity along their surfaces. There exists a large number of metrics to determine the closeness of two pixels or colors. The most basic way to measure two color pixels is using Euclidean distance over their RGB. This is insufficient for most cases due to luminance is implicit in the RGB representation of the color. This means that RGB space is very sensitive to light changes and correlates weakly with human color discrimination performance and therefore it is a powerless color descriptor.

Ma *et al.* [148] searches for a dominant color profile using $CIE\ L^*a^*b$ color space and then compares each with each neighbor creating a *Color Distance Histogram* (CDH) descriptor. Once again, the problem searches for an Euclidean-distance based homogeneity criteria with no explicit proof of its performance.

A good color descriptor should allow interest points to be matched despite illumination changes as Kyriakoulis *et al.* stands in [149]. Taking as a reference the work of Torsten *et al.* [146] a hue and saturation histograms in an inner volume are extracted for each point in a HSV color space. The usage of histograms makes correspondences rotation invariance, boosting the matching performance. Each histogram is formed by 24 bins for hue and one last bin for unsaturated regions. Each entry to a hue bin is weighted with its saturation value s. The gray bin will receive a value of $1 - s$ and therefore colorless regions will be perceived as well.

Euclidean distance is not suitable for color comparisons even in HSV space due to its variation and irregularity in unsteady scenarios. For such a reason, a generalization of the original Kullback Leibler divergence [147] is proposed as distance metric in this work. Histograms have to be normalized before computing the histogram distance so that the area of the histogram equals one.

Therefore, for each correspondence a color histogram will be extracted according to their surrounding texture values. Considering that the intensity distribution of the view $\mathcal{V}$ is given by $\Lambda(\mathcal{V}) = \{\Lambda(v_c)_{i=0}^{N_\mathcal{V}}\}$, a hue histogram $\mathcal{H}$ associated to a certain correspondence $c \in C$ can be computed as

$$\mathcal{H}(\Lambda(v_c)) = \{h_{v_c}^{(1)}, h_{v_c}^{(2)}, \ldots, h_{v_c}^{(24)}\} \tag{7.46}$$

Accordingly, model distribution $\mathcal{M}$ is given by $\Lambda(\mathcal{M}) = \{\Lambda(m_j)_{j=0}^{N_\mathcal{M}}\}$ and its histogram for each correspondence will be

$$\mathcal{H}(\Lambda(m_c)) = \{h_{m_c}^{(1)}, h_{m_c}^{(2)}, \ldots, h_{m_c}^{(24)}\} \tag{7.47}$$

where $h_{m_c}^{(1)}$ corresponds to the first bin of the intensity histogram associated to m_c. The summation of bins for both distributions will be equal to one, so

$$\sum_{k=0}^{24} h_{m_c}^{(k)} = \sum_{k=0}^{24} h_{v_c}^{(k)} = 1 \tag{7.48}$$

To compute the difference between two histograms, the Jensen-Shannon divergence has been applied. Its fundamentals are based on the Kullback-Leibler divergence, with two main differences: divergence always results in a finite value and it is also symmetric. This divergence and its properties are explained in detail in Appendix B. The base is, according to Endres and Schindelin in [150], the distance between two distributions P, Q:

$$D_{PQ}^2 = \sum_{i=1}^{N} \left(p_i \cdot \log \frac{2p_i}{p_i + q_i} + q_i \cdot \log \frac{2q_i}{p_i + q_i} \right) \tag{7.49}$$

For a single correspondence c, the distance function will be

$$d(v_c, m_c)^2 = \sum_{k=1}^{24} \left(h_{v_c}^{(k)} \cdot \log \frac{2h_{v_c}^{(k)}}{h_{v_c}^{(k)} + h_{m_c}^{(k)}} + h_{m_c}^{(k)} \cdot \log \frac{2h_{m_c}^{(k)}}{h_{v_c}^{(k)} + q_i} \right) \tag{7.50}$$

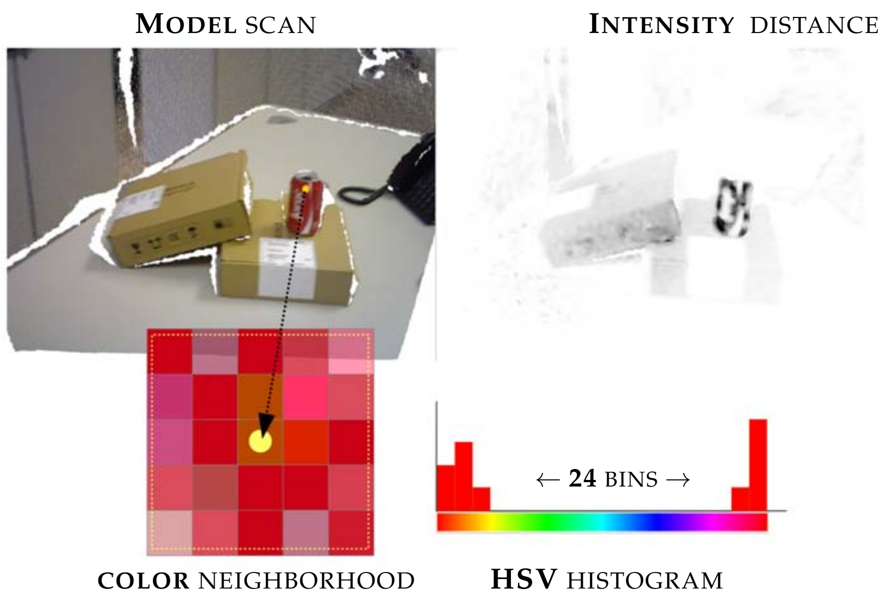


Figure 7.12: *Color distance* Representation of the color distance for an scenario. The input color map (on the right) is analyzed and a HSV histogram is extracted for each pixel looking at its neighbors. Then a global matching search is done using Jensen-Shannon divergence as distance metric.

and for the whole list of correspondences C

$$d_{ICE}(\mathcal{V}_c, \mathcal{M}_c)^2 = \sum_{c \in C} d(v_c, m_c)^2 \tag{7.51}$$

Replacing 7.50 in the last equation yields the color distance function

$$d_{ICE}(\mathcal{V}_c, \mathcal{M}_c)^2 = \sum_{k=1}^{24} \left(h_{v_c}^{(k)} \cdot \log \frac{2h_{v_c}^{(k)}}{h_{v_c}^{(k)} + h_{m_c}^{(k)}} + h_{m_c}^{(k)} \cdot \log \frac{2h_{m_c}^{(k)}}{h_{v_c}^{(k)} + q_i} \right) \tag{7.52}$$

Figure 7.12 shows an example of the color distance for a scenario. There are two boxes and a can lying on a table. If the yellow point belonging to the can is chosen, its 5x5 neighborhood is analyzed and a 24-bin hue histogram is created for that particular element. Then, with that information, color distance function with respect to the rest of the point cloud is applied.

For those scenarios whose objects are similar in saturation, a weight function can be passed. Using an exponential filter with lower and upper boundaries $[\epsilon_{min}, \epsilon_{max}]$, distances are magnified near the extremes. Looking at the distance function gray scale representation displayed in Figure 7.12, nearest distances are amplified following the weight function

$$y(x) = e^{b \cdot x} - a \tag{7.53}$$

For the boundaries ϵ_{min}, ϵ_{max} and taking into account that pixels are represented in gray scale between 0 and 255, the filter stands as

$$d^*(v_c, m_c)^2 = \exp\left\{ \frac{\log(1 + (\epsilon_{max} - \epsilon_{min})) \cdot d(v_c, m_c)^2}{255} \right\} - (1 - \epsilon_{min}) \tag{7.54}$$

Overall fitness function

Several assumptions have to be done at this point. Firstly, since the global distance makes sense during the initial alignment, it will have more importance during the first iterations. The local distance error has an interest on those areas where curvature information is irrelevant, such as flat surfaces or extremely noisy surfaces. Thereby, normals alignment error take an important role in order to match parts of the object with flat surfaces such as the handle of a mug, increasing the convergence speed and discarding local minima. In order to make the fitness function more efficient, two improvements will be held. On the one hand, the number of elements will be reduced using a down-sampling filter using a voxelized grid approach. VoxelGrid creates a 3D voxel grid over the input point cloud data. Then, in each voxel all the points present will be approximated with their centroid. On the other hand, the second optimization process is the generation of $k - d$ trees for each point cloud, reducing search times.

It is also expected that the local distance is always lower than the global distance: this is true if the objects do not contain spurious measurements or floating points. For this to happen, a smooth and outliers removal filter will be applied as will be explained in the following section. Furthermore, intensity values are extracted so the intensity becomes valuable when luminance and texture is rich of information. Therefore, the fitness function assumes all those error sources and merge them into a single cost. Equation 7.26 can be substituted with a more specific version integrating all the errors in the fitness function.

$$e(\boldsymbol{R},\boldsymbol{t}) = \sum_{c=1}^{C} \overbrace{d_{LDE}(\mathcal{V}_c,\mathcal{M})}^{local\,distance} + \underbrace{d_{GDE}(\mathcal{V}_c,\mathcal{M})}_{global\,distance} + \underbrace{d_{NAE}(\Xi(\mathcal{V})_c,\Xi(\mathcal{M}))}_{normal\,distance} + \overbrace{d_{ICE}(\Lambda(\mathcal{V})_c,\Lambda(\mathcal{M}))}^{intensity\,distance} \quad (7.55)$$

So, those distances are exclusively computed to those correspondences. For this case, a correspondence c will be assigned to the pair (v_i, m_j) assuming that the view point cloud has been transformed according to the corresponding transformation. The overall registration process is explained in Algorithm 5.

Algorithm 5 Registration process. *Explanation of each step performed to compute the registration of a set of point clouds*

```
Require: Model point cloud $\mathcal{M}$
Require: List of $N_L$ View point clouds $\hat{\mathcal{V}} = \{\mathcal{V}^1, \mathcal{V}^2, \dots, \mathcal{V}^{N_L}\}$
Require: Global Model point cloud $\mathcal{G}$
 1: $\mathcal{M} \Leftarrow demean(\mathcal{M})$                                  ▷ Initialization of the model
 2: $features_{model} \Leftarrow extract\_features(\mathcal{M})$               ▷ Features of the model
 3: $T_{global} \Leftarrow I$                                              ▷ Initialization with unitary matrix
 4: $\mathcal{G} \Leftarrow \{\}$                                             ▷ Clear the global model
 5: for $(i = 1 : N_L)$ do                                       ▷ Go through the list of views
 6:     $\mathcal{V}^{(i)} \Leftarrow demean(\mathcal{V}^{(i)})$                          ▷ Initialization of the view
 7:     $features^{(i)}_{view} \Leftarrow extract\_features(\mathcal{V}^{(i)})$         ▷ Get all the features of the view
 8:     $\mathcal{V}^{(i)} \Leftarrow T_{global}(\mathcal{V}^{(i)})$                       ▷ Update the initial pose of the view
 9:     $T^{(i)}_{initial} \Leftarrow yaw\_estimation(\mathcal{M}, \mathcal{V}^{(i)})$          ▷ Estimate initial yaw value
10:     $(T^{(i)}_{optimal}, best_{energy}) \Leftarrow DE(\mathcal{M}, \mathcal{V}^{(i)}, T^{(i)}_{initial}, features^{(i)}_{view}, features_{model})$
11:     if $(best_{energy} < \mu)$ then                              ▷ Check false transformations
12:         $T_{global} \Leftarrow T_{global} \cdot T^{(i)}_{optimal}$                  ▷ Update the global transformation
13:         $\mathcal{G} \Leftarrow \mathcal{G} + T(\mathcal{V}^{(i)})$                                 ▷ Update the model
14:     end if
15: end for
16: $\mathcal{G} \Leftarrow filter(\mathcal{G})$                                           ▷ Clean the model
17: return $\mathcal{G}$                                                ▷ Return the global model
```

The process requires from a list of view point clouds $\hat{\mathcal{V}}$ that corresponds to a set of scans made by the robot during the learning process. Then a model point cloud

$\mathcal{M}$ that could be the first in the view list is also required. The model will represent the initial view from which the complete object will be reconstructed. Firstly a demean function will move to the origin the whole point cloud as it is not necessary for the correspondence process (Line #1). Two features maps are extracted for the model that corresponds to the normal map $\Xi(\mathcal{M}) = \{n_j^{\mathcal{M}}\}_{j=0}^{N_{\mathcal{M}}}$ and an intensity map $\Lambda(\mathcal{M}) = \{m_j^{\mathcal{M}}\}_{j=0}^{N_{\mathcal{M}}}$ (Line #2). Then, for the whole list of views, the registration process starts. Primarily, the same process as that for the model is done, removing the spatial offset by demeaning the point cloud and extracting their features. Then, a first transformation is performed and stored in T_{global}. The first time this transformation is null but as long as the register process is being executed, it stores the accumulation of all the transformations. Then, in Line #9 the yaw initialization process is performed to estimate the initial configuration of the view. Then the global evolutionary optimizator is launched taking into account all the list of features for both point clouds and also the initial transformation (Line #10). Then in the next line, a check routine will determine if the solution received is correct or not. This verification will avoid false local minima and accumulated errors. In case the $best_{energy}$ is sufficiently small, the transformation will be assumed as valid and the result $T_{optimal}^{(i)}$ will be incorporated in T_{global}. Furthermore, the global model $\mathcal{G}$ will be updated with a new transformed point cloud. After the whole list of views is analyzed, the global model will be filtered to remove noise points and spurious elements and returned.

7.4.2 Filtering the Point Cloud

Last step consists on removing and filtering all the empty and spurious points from the new point cloud and aligning the registered point clouds. For this process, an outlier removal has been implemented using a radius filter. It operates searching for the nearest neighbors for each element of the point cloud in an sphere using L2 euclidean norm. If a certain number of points fall inside the sphere, those points are saved. Otherwise they are removed. Furthermore, it is possible to find garbage points in the origin due to the geometrical operations performed previously. For this reason, an extra inspection is done to avoid incoherences and dismiss any incorrect point.

The following Figure 7.13 represents an example of how the radius filter removal responds. If the number of neighbor points in the filter is set to one, only the blue point will be removed. However, if the required number of points is equal to two, both the blue and the red points will be removed. This approximation can be extended to 3D with no effort. The only parameter to be chosen here is the radius value. For this experiments, taking into account the size of the analyzed

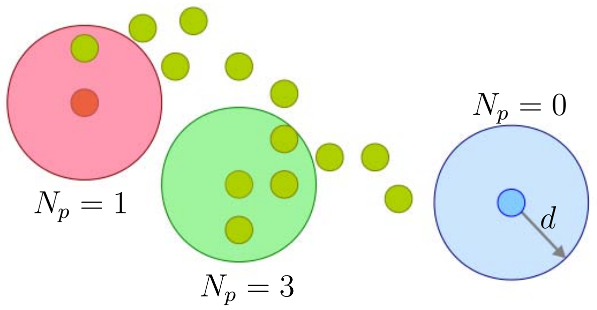


Figure 7.13: *Outlier filtering by radius search.* Each element of the point cloud is accepted if there exists at least a certain number of neighbors in its nearby.

objects and the precision of the 3D sensor, a reasonable value for this distance is 0.02 m.

However, the problem of data irregularities yielded by the registration process can be extremely decisive during the reconstruction phase. The addition of systematic errors in conjunction with the estimation errors induce to the addition of noisy floating points, *double walls* effects or any other artifacts as a consequence of a poor transformation estimation. The number of views is quite limited for some cases and therefore the intrinsic information of the object. A solution is to use a re-sampling algorithm, which attempts to recreate the missing parts of the surface by higher order polynomial interpolations between the surrounding data points. An example of this filter is represented in Figure 7.14 for a complete 3D model of an object. As can be seen, the blue point cloud has a most structured shape.

7.5 Experimental Results

In this section several experiments will be performed in order to corroborate the novel object reconstructor. The experimental results have been divided into three different sections. Firstly, the initialization process, which has been studied in order to determine its contribution to the fitness function during the initial iterations. Secondly, the hole filling process will be measured in order to determine the enrichment of the system after the scan matching. Finally, a comparison between the classical ICP and the different versions of the fitness function will be discussed. All the experiments have been developed using real models acquired in the laboratory. Some examples of the resulting models can be seen at the end of the chapter.

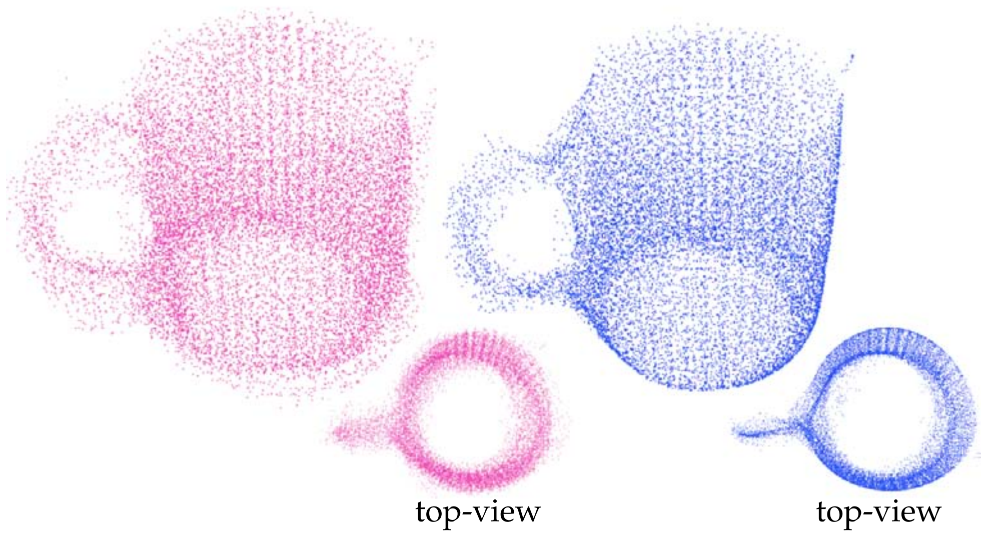


Figure 7.14: *Normal alignment filter.* Result of filtering the global 3D point cloud of a mug using a Moving Least Squares surface reconstruction. The surface becomes smoother and noisy data is avoided (right).

7.5.1 Yaw Initialization Response

The first experiment evaluates the effect of the yaw initialization with the convergence of the scan matching optimizer. Therefore, two views of the same object have been used in the whole experiment. In order to measure the convergence speed, the angle resolution of the initialization process has been altered gradually. Therefore, for a set of two views, the angle increment has been incremented. For all the experiments the convergence condition is to finish by the number of iterations, limited to 150. The weight function will be the same for all cases and will follow the values plotted in 7.9.

Figure 7.15 represents the fitness function error during the first 150 iterations of the evolutionary reconstructor. Each distribution corresponds to the average of the experiment repeated 10 times at different yaw resolutions. As can be seen, the lowest is the angle resolution, the fastest is the convergence during the first third range of iterations.

At iteration number 10 the finest yaw resolution (in purple) gets stacked during 10 iterations while the rest of the series continue decreasing. However, at iteration number 37, the fastest convergence is performed by the finest initialization, 1^o. At iteration 40 the results are almost the same: error values are sorted by yaw resolution. However, as long as the reconstructor iterates, the values for each resolution

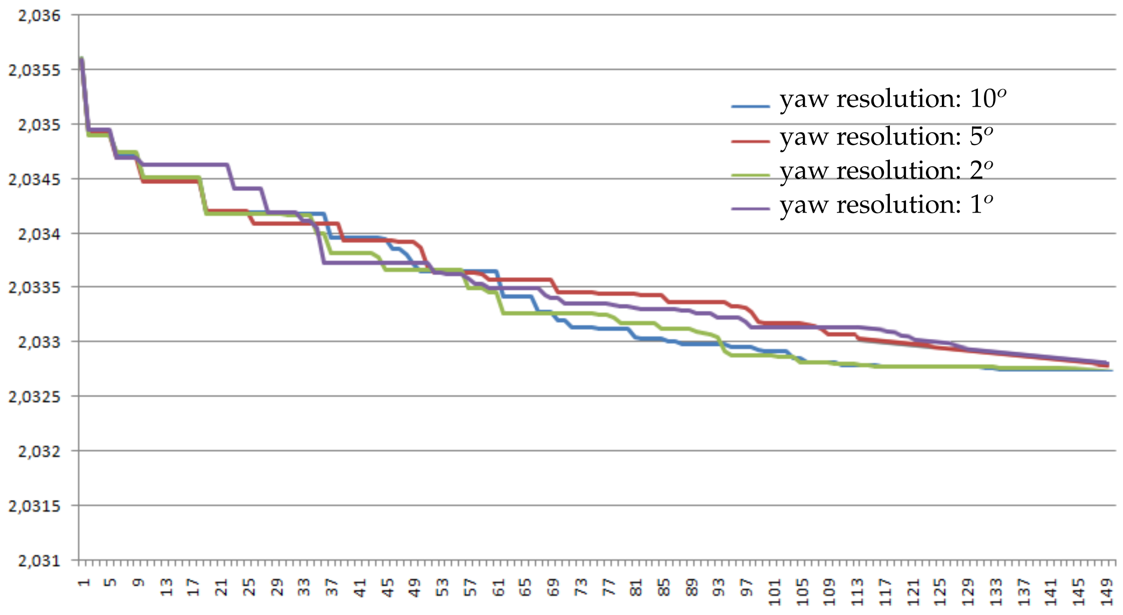


Figure 7.15: *Yaw initialization response.* Average of fitness function $e(\boldsymbol{R}, \boldsymbol{t})$ with different angle resolutions during the yaw initialization process. Optimization stops after 150 generations.

start changing again.

At iteration 150, the results are almost the same for the four categories. Analyzing the whole evolution and taking into account the elapsed time of the initialization and the convergence, the best resolution is the second one (measure errors every two degrees) as it requires less time than the third and the fourth in most of the cases. Table 7.2 contains the final transformation for each case.

TRANSFORMATION RESULTS							
Yaw_{res}	MSE	t_x[mm]	t_y[mm]	t_z[mm]	r_x[rad]	r_y[rad]	r_z[rad]
1 deg	1.02427	-0.31612	-0.19762	-0.82514	0.00264	-0.00349	-0.36444
2 deg	1.13281	-0.31622	-0.22152	-0.81529	0.00263	-0.00325	-0.35231
5 deg	1.17487	-0.32411	-0.21582	-0.74519	0.00261	-0.00355	-0.37107
10deg	1.48654	-0.36624	-0.24168	-0.46125	0.00261	-0.00337	-0.46669

Table 7.2: Average of the final transformation and MSE for different initializations at generation 150

7.5.2 Hole Filling Enrichment

One of the most important challenges of structured light 3D sensors are the glows of the objects. If the sensor has to deal with dark objects or extremely shining and gloss surfaces, the depth estimation will decay abruptly. Technology advances in this topic are providing with more and more precise sensors that reduce the systematic errors. In the mean time, algorithms like the one proposed in this dissertation helps to introduce estimated data inside the observed model.

In this case, a Gaussian distribution will be used as weight function along those estimated points introduced in uncertain areas of the observation such as holes and boundaries. Therefore, elements with such lack of information in these areas will yield better results during the matching process. A standard deviation of $1mm$ will be applied in the neighborhood of the holes. A $k-$NN tree will be created in these areas to reduce times. Curvature estimation length parameter will be changed. It represents the density of new elements that have to be introduced in these areas. For this experiment three different values will be picked: 3 points/mm, 5 points/mm and 10 points/mm.

In Table 7.3, Table 7.4 and Table 7.5 the result of introducing this feature is compared with the original observations. As can be seen, the number of points increases with the increase of density and the MSE gets its best result in 5 mm/point curvature estimation length. If the parameter exceeds this value, the number of points is overestimated and the MSE results are incorrect. And on the other hand, if the curvature estimation length parameter is too short, the MSE is almost the same.

EXPERIMENTAL RESULTS					
object name	$N_{original}$	$N_{extended}$	% increase	$MSE_{original}$	$MSE_{extended}$
`mug-cow`	949	1891	+99.26%	1.23321	1.64223
`mug-r-red`	715	1422	+98.88%	1.16515	1.4239
`mug-r-white`	775	1541	+98.83%	1.38950	1.61609
`mug-coffee-tiny`	342	665	+94.27%	0.98732	1.14324

Table 7.3: Results of error measurements for different objects with a curvature estimation length of 10 points/mm

EXPERIMENTAL RESULTS					
object name	$N_{original}$	$N_{extended}$	% increase	$MSE_{original}$	$MSE_{extended}$
`mug-cow`	949	1274	+34.24%	1.23321	1.13475
`mug-r-red`	715	1007	+40.83%	1.16515	1.12471
`mug-r-white`	775	1073	+38.45%	1.38950	1.27589
`mug-coffee-tiny`	342	427	+24.85%	0.98732	0.89740

Table 7.4: Results of error measurements for different objects with a curvature estimation length of 5 points/mm

EXPERIMENTAL RESULTS					
object name	$N_{original}$	$N_{extended}$	% increase	$MSE_{original}$	$MSE_{extended}$
`mug-cow`	949	1047	+10.32%	1.23321	1.23214
`mug-r-red`	715	814	13.84%	1.16515	1.16411
`mug-r-white`	775	955	+10.32%	1.38950	1.39474
`mug-coffee-tiny`	342	390	+14.03%	0.98732	0.98428

Table 7.5: Results of error measurements for different objects with a curvature estimation length of 3 points/mm

7.5.3 ICP versus DE-ICP

The last experiment illustrates the comparison between the classical ICP and the reconstruction yielded by the evolutionary reconstructor. Table 7.6 contains the MSE errors for different image pairs with the four fitness functions: only local distance (DE1), local distance plus global distance (DE2), distances and normal alignment (DE3) and finally the complete system with color matching (DE4).

SCAN MATCHING COMPARISON					
object name	MSE(ICP)	MSE(DE1)	MSE(DE2)	MSE(DE3)	MSE(DE4)
`mug-cow`	1.31414	1.30478	1.20179	1.17415	**1.13475**
`mug-r-red`	1.8725	1.34787	1.24381	1.15474	**1.12471**
`mug-r-white`	1.3999	1.4874	1.36544	**1.25748**	1.27589
`mug-coffee-tiny`	0.97584	0.92147	0.90558	**0.88475**	0.89740

Table 7.6: Comparison between the results given by classic ICP scan matching and the evolutionary reconstructor in all its variants. In blue the best result configuration.

Surprisingly, the color function does not always work as expected. This happens on texture-less objects such as the `mug-r-white` or `mug-coffee-tiny`, where the color information may confuse the optimization algorithm. Nevertheless, the results are quite promising, improving the ICP algorithm in almost all the cases. The best transformations (blue configurations in the previous Table 7.6) have been represented in Figure 7.16. The model is in blue and the evolved view is in red.

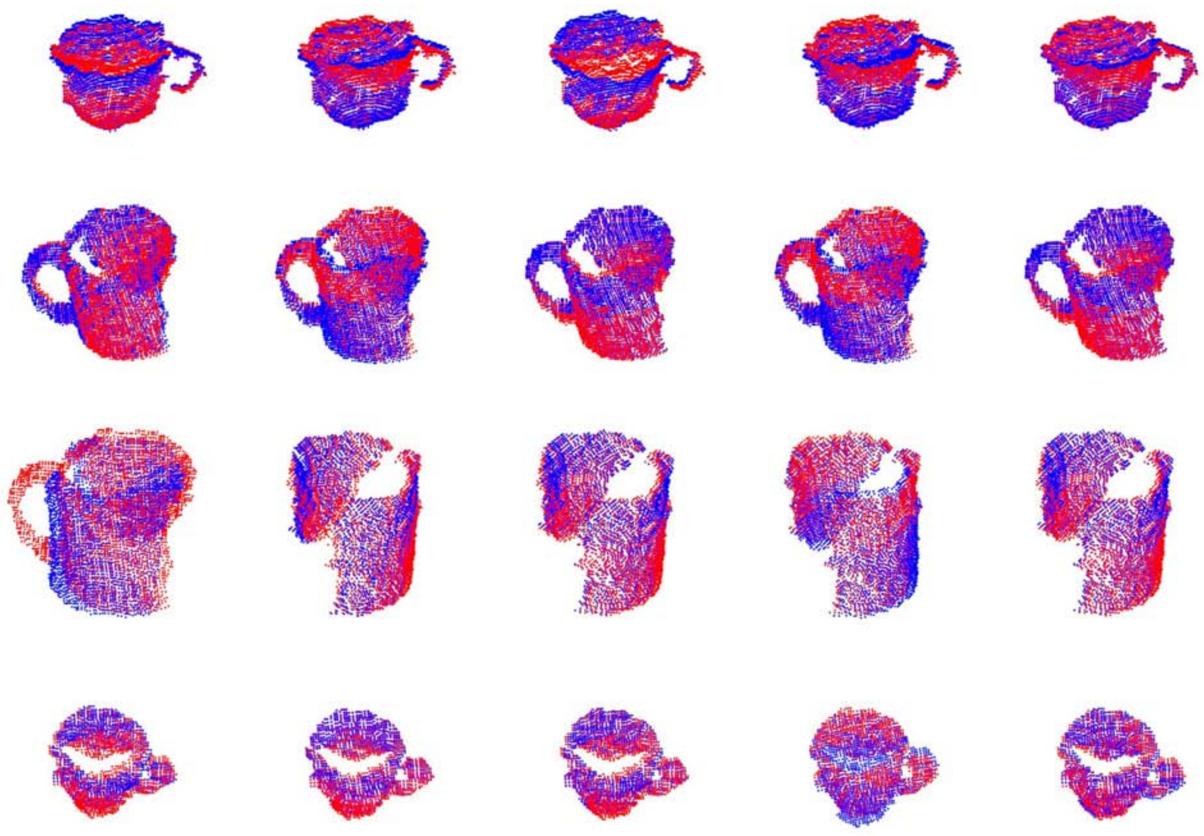


Figure 7.16: *Reconstruction results.* Representation of the best transformation achieved for each algorithm. Each row represents a different object. Columns are the transformation using ICP, DE1, DE2, DE3 and DE4 functions.

Finally, some images of the output reconstructor are displayed in Figure 7.17. Those images are obtained after the application of the radius and the normal alignment filters explained earlier. The models are perfectly ready to be introduced in grasping point analysis in order to determine the optimal grasping strategy. This work goes beyond this thesis but it may address promising results.

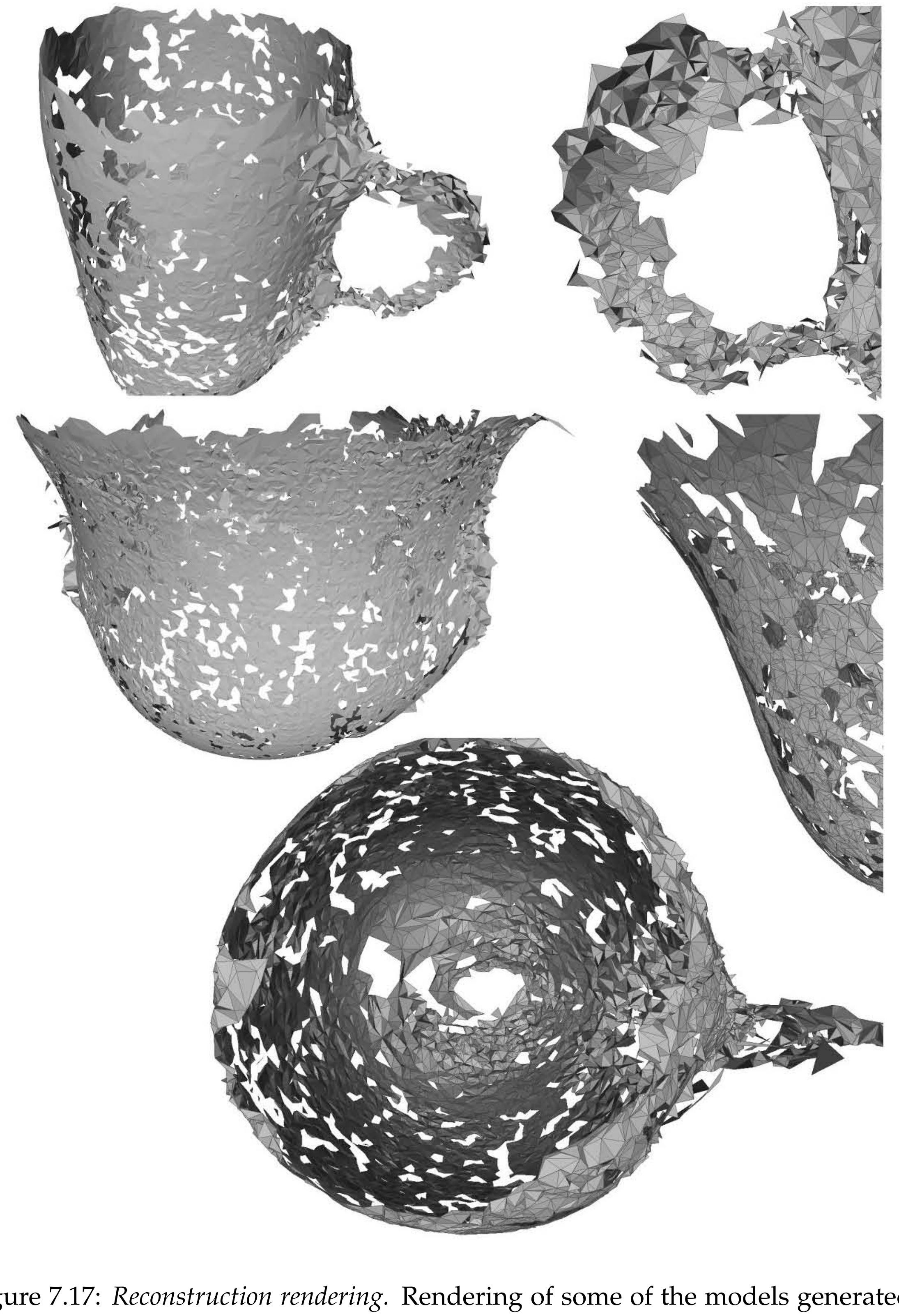


Figure 7.17: *Reconstruction rendering.* Rendering of some of the models generated using the evolutionary reconstructor.

PART III

RECOGNIZING MODELS

Chapter 8

3D Features

During the last years, object recognition has turned into a prior research line in robotics. The idea of designing robots capable of detecting and recognizing objects in their vicinity make more natural and prosperous their integration in humanlike scenarios such as offices, kitchens [151] or living rooms [44] and [152]. With the aim of detecting and interacting with these objects, robot perception abilities have to be mostly enhanced. Time-Of-Flight technology and structured-light have become decisive in perception sensorial devices such as Kinect camera or Asus Xtion Pro Live. Those instruments are capable of capturing RGB-D point clouds at high frame rates allowing robots to interpret and analyze with clarity what is coming about [153] and consequently react to these perturbations.

8.1 Object Recognition

As Beserra *et al.* [154] reported, early object recognition systems acquired data from expensive and rarely available range sensors, such as laser scanners [155],[156] and structured light patterns [157]. Ashbrook *et al.* [155] describe an object recognition system that relies on similarities between geometric histograms extracted from the 3D data and the Hough Transform [158]. Johnson and Hebert popularized the Spin Images descriptor [156] or [157] which was used as the basis to an object recognition algorithm that groups correspondences of Spin Images extracted in a given query model and those extracted in the scene data that share a similar rigid transformation between the model and the scene [156]. Data from 3D scanners and also from synthetic CAD 3D models are employed in the work of Mian *et al.* [159]. Until recently, 3D object recognition systems processed data mostly in an off-line fashion, due to long computing times involved [160]. This paradigm has started to shift as algorithms have been proposed in the Robotics community such as Rusu [161] or Aldoma *et al.* [162], to enable real-time manipulation and grasping for robotic manipulators.

In fact, algorithms designed to describe 3D surfaces through histograms of various local geometric traits evaluated on point clouds became a major trend in the last years [161], [163], [164], [165]. Consequently, faster and more accurate 3D object recognition systems based on keypoint matching and descriptors extracted in the scene and in the sought object point clouds were developed. After being established, point correspondences are grouped by hypotheses sharing a common transformation, which is estimated by voting as Chen *et al.* [166] and afterwards Tombari *et al.* in [167], multi-dimensional clustering [168], [169] or RANSAC [170]. The presence of the object of interest is then inferred if certain conditions are met, such as the number of votes, cluster size, or the number of RANSAC inliers.

With the wider availability of consumer-grade depth sensors such as the Microsoft Kinect, several works on 3D object recognition are proposed employing this class of sensor [162], [171],[172], [173]. Aldoma *et al.* [162] proposed the global feature coined Clustered Viewpoint Feature Histogram (CVFH) to improve performance of object recognition for robotics. Machine learning-based approaches such as Lai *et al.* in [174] were formulated to perform 3D object recognition making heavy use of depth information, without any computation on point clouds involved. There exist also 3D object classification/categorization systems, as in the works by Wohlkinger *et al.* [175] and by Lai *et al.* [174]. In this latter class of systems, every chair in a scene should be labeled as the object of type "chair", whereas in object recognition only the specific chair being sought should be retrieved from the scene. Other features such as low-level features are used, *e.g.* histograms of gradients, as well as those describing the saliency[13] and exploited in works such as Palmer *et al.* [176].

Objects are intended to be detected mostly for grasping. This fact leads to a consequent challenge: items have to be not only successfully detected but also favorably located in order to determine the path to reach and grasp them [24]. This requirement demands two different goals: object pose by means of its 6 DOF and object labeling based on its inherent features such as color, texture, shape, 3D key-points, geometry, center of gravity, etc. For the first purpose, there exist multiple strategies such as 3D recognition based on correspondence grouping using SHOT32 features, ICP-like algorithms such as Rusinkiewicz *et al.* [177] or 3D matching Garcá *et al.* [44] or the latter in [178],[179]. For the last, 3D features are necessary. Some of the perception features used nowadays are SIFT 3D presented by Scovanner in [178], or its optimized counterpart SURF 3D [179] and [180]. NARF descriptor proposed by [181] will be explained below and other such as FPFH, presented by Rusu in [145]. Without exception, every descriptor is based on consistent features from a 3D point cloud cluster such as edges, normal variations on the surfaces or intrinsic geometric models such as lines, planes or spheres.

Most 3D features are extracted in areas where the surface is stable but has substantial change in the vicinity. Furthermore, interest points are always intended to be robust against noise and displacements so that they can be applied in partial views due to occlusions to create a collection of views correctly aligned.

[13]Saliency measures the level of unusualness in a specific region of the image

8.2 Normal Aligned Range Value Patches

The first version of this feature named NARVP was proposed by Steder and Grisetti in [182]. They proposed the extraction of point features from range images that are computed from the point-clouds. Applying a Harris detector [183] on the range image (see Figure 8.1 as an example), a list of interest points is extracted.

Then, for each of those point features p_i a descriptor vector f_i is calculated in the following manner:

1. Find the nearest neighbors $N(p_i)$ of p_i.
2. Extract the normal $\boldsymbol{n_i}$ of $N(p_i)$ applying a least-square plane fitting estimation (read section 5.2).
3. Select a point v_i along the line which passes through p_i and is oriented according to $\boldsymbol{n_i}$.
4. Move the observer point of view at v_i and its viewing direction according to vector $-\boldsymbol{n_i}$.
5. A descriptor vector f_i is generated by creating a square grid standing on the previous plane. The distances of the surrounding points projected onto the plane determine the value for every grid cell.

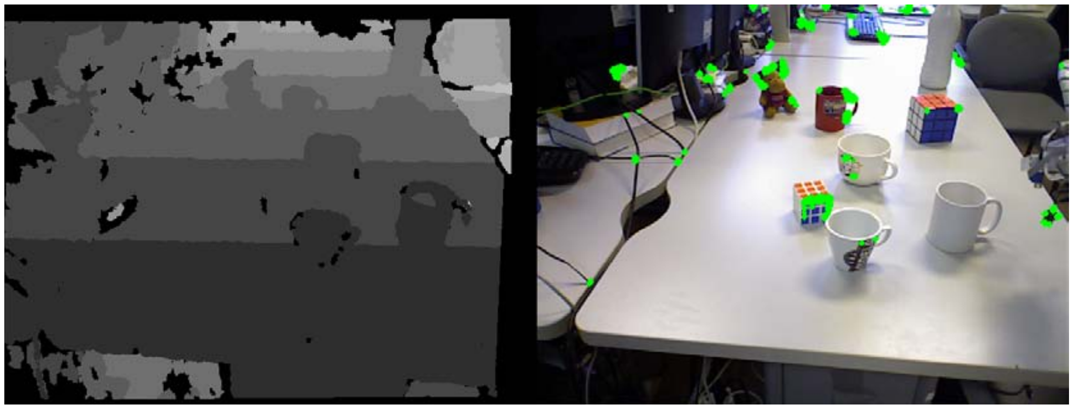


Figure 8.1: *Harris feature detector.* Example of a gray scale range image (brighter pixels represent further positions) on the left. Harris features are superimposed on the color frame on the right.

There exist three DOF for each feature. Two of them are restrained by the selection of the viewpoint v_i along the normal vector of the surface. The last degree is fixed by orienting every patch according to the $z-$axis in the world. This restriction is valid for objects lying on a supporting plane such as tables, chairs, etc., but it is still a constraint that will be fixed below.

An object is labeled by extracting its features from different poses and gather them in a *feature list* according to that item. While grabbing the scenario, the corresponding range image for this scan is calculated. Afterwards, features are extracted for the image and compared with the bag of *feature lists*. Correspondences are extracted by means of GOODSAC [184] and a list of possible transformations is created. This list contains a set of triples[14] based on the previous correspondences. With several filters and geometrical restrictions, the best triple is chosen representing the best fit.

With the aim of scoring the candidates, a comparison of the depth values of the original range scenario $D^s = \{d_1^s, d_2^s, \dots, d_n^s\}$ and validation points of the object transformation $D^v = \{d_1^v, d_2^v, \dots, d_n^v\}$ is performed penalizing those points in the scene which should have been occluded by the matched object

$$s(d_i^v, d_i^s) = \begin{cases} d_i^v - d_i^s & < & -\epsilon & : & 0 \\ d_i^v - d_i^s & < & \epsilon & : & \frac{|d_i^v - d_i^s|}{\epsilon} \\ d_i^v - d_i^s & > & \epsilon & : & -p \end{cases} \tag{8.1}$$

and the score of the full transformation is the summation of each individual comparison

$$S(D^s, D^v) = \frac{\max(0, \sum\limits_i s(d_i^v, d_i^s))}{n} \tag{8.2}$$

With this implementation, Steder and Grisetti in [182] proposed a robust 3D keypoints implementation that is able to run in real time. However, it was definitely improved several years later by Steder in [185], in a new keypoint descriptor named NARF (Normal Aligned Radial Feature).

8.2.1 Normal Aligned Radial Features

The goals for this 3D range data point extractor are mainly two. In the one hand, the algorithm aspire to ensure a robust estimation of the normal while in the other hand these keypoints are prepared to focus on object borders (large changes in

[14]A triple is formed by three correspondences and represent a valid 3D rigid transformation.

depth). To achieve this independence, dominant directions of the surface changes in the area are obtained with the following points:

1. Find borders in the range image measuring the distance gradient along the view-point axis.
2. Score each image point local neighborhood surface changes determining the dominant directions for this change.
3. Measure the changes on those dominant directions.
4. Remove noise performing non-maximum suppression and smoothing the interest values.

Therefore, as Steder states in [185], for each point, a normal aligned range value patch is calculated taking into account the principal components of the normals locally around the point involved. With an image patch size of `10x10` pixels, a Gauss kernel is passed to smooth the range image in this area. The star shaped pattern represented in Figure 8.2 is projected onto the point with an angular resolution that has been fixed to 10^o so the descriptor is represented by a total of 36 values in a histogram. As Skiff explains in [186], NARF features can be exploited in RGBD-SLAM researches albeit there exist noisy and nonuniform point clouds, which can be suitably filtered and segmented to generate basic blueprints of a certain space. Grzonka [187]*et al.* proposed a NARF feature extraction to detect objects such as chairs and tables from a small-sized quadrotor.

In the example displayed, the patch is colored in pink in the point cloud with its normal in yellow. On the right side, the patch of the upper border of the chair is printed in gray with the pattern superposed in pink. The dominant orientation is represented with a blue arrow on the patch. The intuition says that the more the structure below the beam changes, the higher the corresponding descriptor value.

Figure 8.3 represents a cluttered scenario with NARF keypoints represented. As can be seen, borders and areas with large depth changes are more likely to be interest points, contrary to the flat surfaces where there are no candidates at all.

8.3 3D Scale Invariant Feature Transform

This algorithm was proposed by Lowe [188] as a revolutionary solution to achieve a large quantity of important points from bustling scenarios. This method has been used not only for points extraction but also to correlate frames of the same scenario

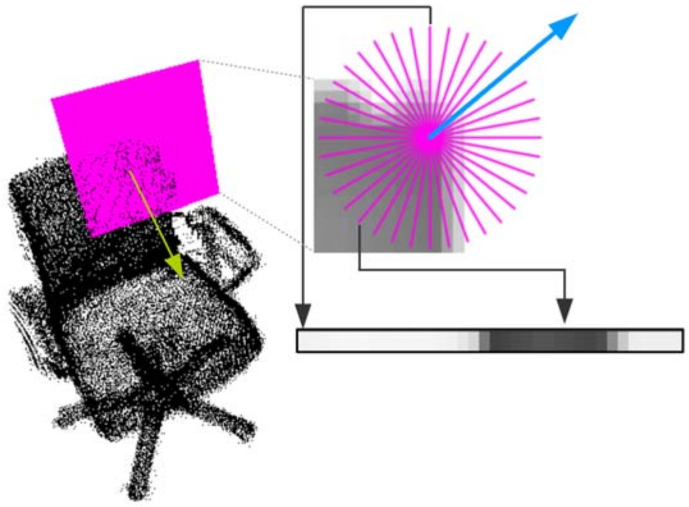


Figure 8.2: *NARF keypoints.* Representation of a point cloud with a sample patch with its descriptor and the pattern used to extract it.

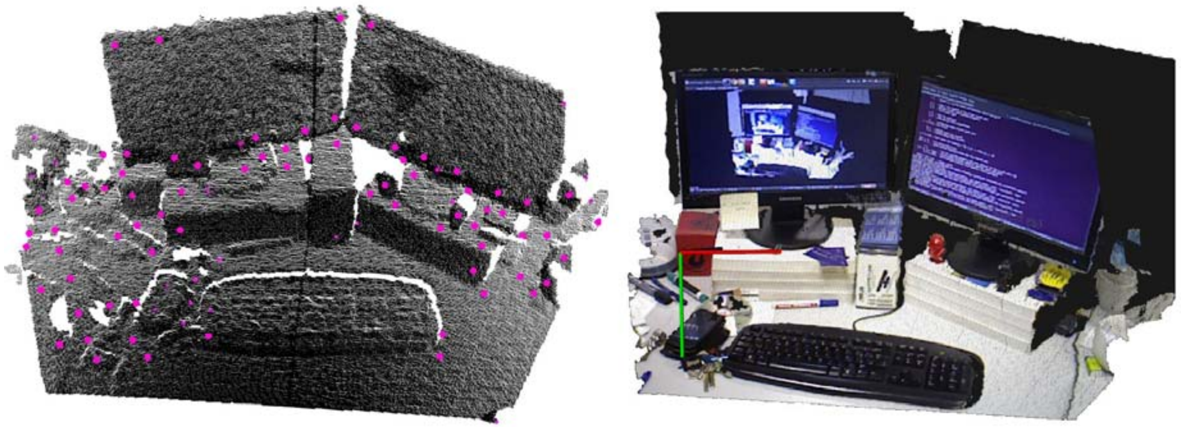


Figure 8.3: *NARF keypoints.* Example of a cluttered environment point cloud (on the left) and the NARF keypoints generated by the NARF feature extractor in pink (on the right).

extracted from different points of view. The correct correlation of several clouds of points might be used for building a complete 3D scene from a batch of 2D samples, as Zhang*et al.* proposed in [189] or Schleicher in [190].

SIFT is divided mainly into four steps, each one extracts a different feature from the image giving at the end a list of points with a complete description representing the most important data fields for each frame [191]:

1. Maxima and minima detection in the space-scale: First step consists on the search of possible candidates to represent *keypoints*. This search is done using Gaussian Functions differences in order to identify points invariant to scale and orientation.
2. Keypoints's localization: After localizing candidates, scale and localization is computed. Afterwards, those keypoints which are more stable are selected.
3. Orientation assignment: To each selected point, its orientation (or principal orientations) is selected. Those orientations are defined based on local gradients around the keypoint.
4. Keypoints's descriptors: Local gradients are measured and transformed into a descriptor vector. This representation allows to describe the distortion levels around keypoints and changes in illumination.

As it is going to be explained afterwards, the number of keypoints extracted from any image will be dependent on the number of objects, textures and edges of those objects. These keypoints can become very useful to have an idea of the important regions of the image and then to pay more attention to find out objects in these areas. Next sections will cover the four steps in detail, to get a global idea of how this method works.

8.3.1 Maxima and Minima Detection in the Space-scale

As commented before, the first stage of SIFT algorithm corresponds to the detection of possible keypoints. To do that, stable features are searched along the frame on different scales. Once this requirement is fulfilled, a Gaussian function is in charge of space-scale changes.

Indeed, $L(x, y, \sigma)$ is defined as a space-scale function in an image. It corresponds to a convolution between a Gaussian of a scalar variable $G(x, y, \sigma)$ and the original image $I(x, y)$ in this way:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{8.3}$$

where Gaussian function is defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{8.4}$$

To perform an efficient calculation of stable keypoints, in scale and space, the usage of maxima and minima of $D(x, y, \sigma)$ function is proposed, that corresponds to the difference of Gaussians convoluted with the image. Difference of Gaussians can be obtained from two contiguous scales separated by a multiplicative constant k as

$$\begin{aligned} D(x, y, \sigma) &= [G(x, y, k \cdot \sigma) - G(x, y, \sigma)] * I(x, y) \\ &= L(x, y, k \cdot \sigma) - L(x, y, \sigma) \end{aligned} \tag{8.5}$$

This approximation is quite fast and easy to compute in a computer. Furthermore, it represents a good representation of the normalized Laplacian $\sigma^2\nabla^2 G$. That is important since σ^2 factor makes the transformation scale invariant. It is also demonstrated that maxima and minima of the normalized Laplacian function are more stable than gradient, Hessian function or even Harris corners.

The relation between D and $\sigma^2\nabla^2 G$ can be compared with the heat diffusion equation, where temporal parameter is $\sigma^2(t = \sigma^2)$

$$\frac{\partial G}{\partial \sigma} = \sigma^2\nabla^2 G \tag{8.6}$$

where in case of close scales $k\sigma$ and σ it can be approximated

$$\sigma^2\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k \cdot \sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{8.7}$$

therefore,

$$G(x, y, k \cdot \sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G \tag{8.8}$$

This result demonstrates that DoG function differs from normalized function just by factor $(k - 1)$. Because this factor affects on every scale, it does not change the maximums and minimums location. Next Figure 8.4 displays an optimal way to construct $D(x, y, \sigma)$ function.

Initial image is increasingly convoluted with Gaussians to produce separated images by a constant scale factor k (left row). Each initial image forms an octave. Each octave is divided a number of intervals s due to $k = 2^{1/s}$. Afterwards it has

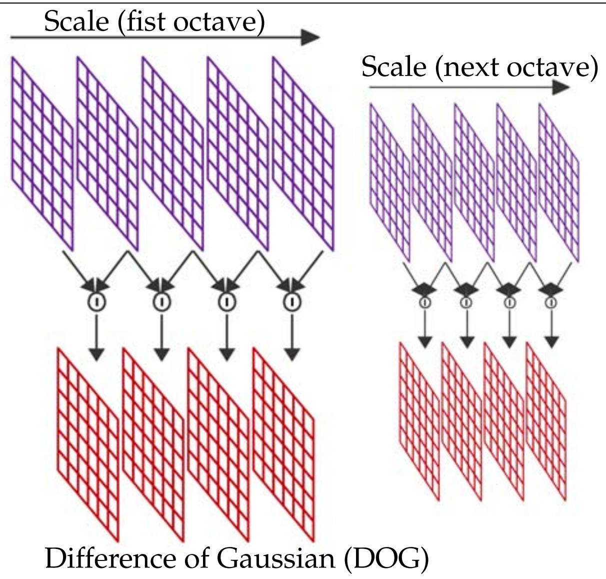


Figure 8.4: *Difference of Gaussians.* Each octave generates several DoG images. On each iteration, image is reduced and blurred again

to be produced $s + 3$ images by octave in the group of fuzzy images so detection of maximums and minimums covers a complete octave. Finally adjacent images are subtracted to obtain a DoG image. After finishing the process for one octave, images have to be sampled again, but this time with a σ value double to initial value, taken the second pixel for each column and row. Precision is reduced in each iteration. Figure 8.5 shows some Gaussians of the same octave and how they get blurred.

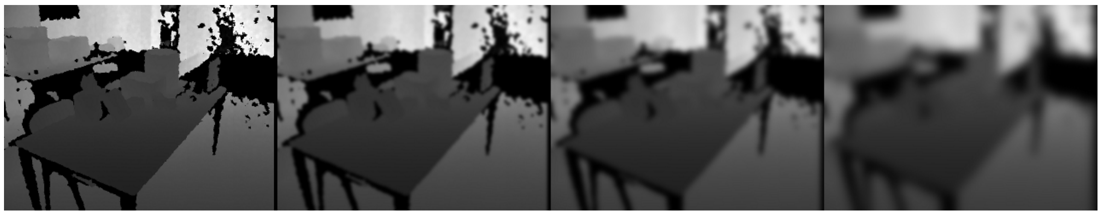


Figure 8.5: *Difference of Gaussians* Original depth map and three Gaussians of the same octave get blurred as long as the process iterates

To obtain the local maximums and minimums, not only the greater and closer pixels from the neighbors pixels are good candidates, but they also those pixels that are maximums and minimums on previous and next images in the same column as Figure 8.6 displays. The computational cost of this maximum and minimum obtaining is small due to most of the points are removed during the initial checking.

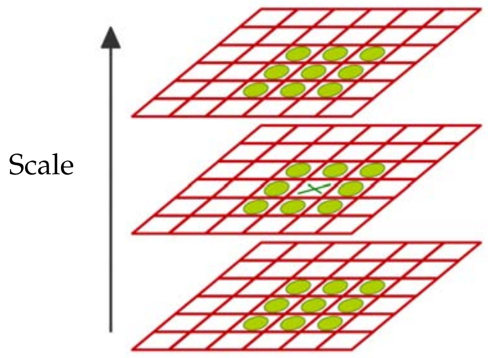


Figure 8.6: *Difference of Gaussians* Finding maxima and minima not only in the same scale but also in upper and bottom scales [188]

A small spatial sampling does not ensure a large number of stable points. The number of scales per octave has to be chosen experimentally. If a large number of scales is chosen, a lot of unstable points would appear being less repetitive.

8.3.2 Keypoints Localization

Once the keypoints are found, next step is to adjust their location, scale and brightness in the neighborhood. This information can reject points due to a low contrast or poor localization near an edge. Again, another approximation has been done to improve calculus and better computational times. In this case, DoG function has been approximated to a Taylor series as

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \tag{8.9}$$

where D and its derivatives are evaluated in the sampling point and $x = (x, y, \sigma)^T$ represents the offset in that point. The location of the critical point $\widehat{x}$ yields by equaling Equation 8.9 to zero. Is obtained that

$$\widehat{x} = \frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \tag{8.10}$$

and it can be demonstrated that Hessian and derivative are approximated by the use of difference of points around near samples. If the offset $\widehat{x}$ is greater than 0.5 in any dimension, that means that there is a critic point very close, and a simple interpolation between both points is performed. Using Equations 8.9 and 8.10 the critical point $D(\widehat{x})$ is obtained as:

$$D(\widehat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\widehat{x} \tag{8.11}$$

According to [188], those points where $\|D(\widehat{x})\| < 0.03$ can be discarded for normalized pixels between 0 and 1. And that removes all those points with a low contrast.

Nonetheless, there is another condition required for keypoints to became real candidates: those points with a low location along an edge, as DoG posses a high response along the edges. The principal curvature of a point can be processed from a 2x2 Hessian matrix evaluated in the keypoint

$$H = \begin{pmatrix} D_{x,x} & D_{x,y} \\ D_{y,x} & D_{y,y} \end{pmatrix} \tag{8.12}$$

The required derivatives to obtain H matrix have been computed taking differences with neighbors in the sampling point. The H eigenvalues are proportional to principal curvatures of D. Taking α as the greatest eigenvalue and β as the smallest, the summation of eigenvalues can be done from the trace as

$$Tr(H) = D_{x,x} + D_{y,y} = \alpha + \beta \tag{8.13}$$

and the determinant as the product

$$Det(H) = D_{x,x} \cdot D_{y,y} - (D_{x,y})^2 = \alpha \cdot \beta \tag{8.14}$$

Taking r as the ratio between those eigenvalues ($\alpha = r\beta$), yields

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha \cdot \beta} = \frac{(r+1)^2}{r} \tag{8.15}$$

that only depends on the relation between eigenvalues. So, according to [188], a reasonable threshold is $r = 10$ for Equation 8.16 to reject those instable points because of a poor location in an edge.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{8.16}$$

8.3.3 Orientation Assignment

The orientation of keypoints is quite important. A good assignment can make the point invariant to rotation. The approach here exposed is based on local image properties. This has a disadvantage, the number of descriptors selected is reduced, cropping areas of the image.

The selection of the orientation is based on local gradients around the keypoints. For this, image is blurred with the highest Gaussian in that point. In this way, calculus are done over information invariant to scale. For each sampling image, $L(x, y)$, gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are calculated using pixel's differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{8.17}$$

$$\theta(x,y) = arctg\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{8.18}$$

A histogram of a keypoint is formed by the orientation of the sampling point's gradients around the keypoint's neighborhood. The orientation histogram is formed by 36 divisions that covers 360^o. Each sample added to the histogram has a weight equivalent to the gradient magnitude multiplied by a Gaussian mask with a 1.5 times the value of the keypoint. Strong directions in the orientation histogram correspond to dominant directions in local gradients. The maximum in the histogram is recorded and compared with the second maximum. If there exist maxima above 80% of the largest one, those will be used to create new keypoints with different orientations, creating a keypoint set with the same location but different orientation.

8.3.4 Keypoints Descriptors

Once all keypoints are found, processed and segmented, keypoint's neighborhood is divided into 4×4 regions of 4×4 pixels (See Figure 8.7). Then a gradient orientation histogram is generated for each region with a weight Gaussian function with $\sigma = 4$ pixels. To decrease conflicts made by small displacements, each pixel's contribution is multiplied by a weight $1 - d$, where d represents the distance to the neighborhood's center. Since orientation histograms for each region are divided into 8 columns, for each neighborhood a three dimensional histogram of $4 \times 4 \times 8(128)$ values is computed.

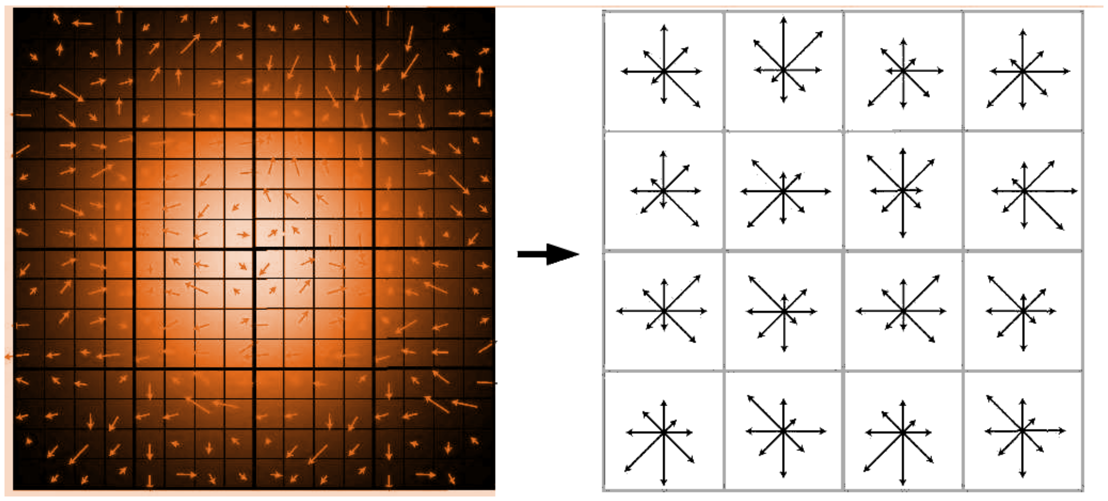


Figure 8.7: *Keypoints orientation* Divisions performed to create a gradient orientation histogram [188]

8.3.5 3D Approach

SIFT can be extended to three dimensional space in two different ways. Triggered by the success of SIFT in 2D computer vision, there have been several attempts to extend the algorithm to three dimensions. N-SIFT [192] and Volume-SIFT (VSIFT) [193] are extensions of SIFT for 3D volumetric (medical) image data, but also extensions of the original SIFT algorithm to work on 3D surface data have been proposed. In [194] the SIFT-algorithm is adapted to range images but is not suitable for full 3D surface representations such as point clouds or meshes. The developed algorithm is hence called Z-SIFT or 2.5D SIFT. Maes *et al.* [195] proposed mesh-SIFT, an alternative that allows reliable detection of scale space extrema as local feature locations.

The first and most direct way is applying keypoint extraction to 2D representations of the 3D depth map. This method is called Z-SIFT and was proposed by He *et al.* [196]. The main idea is to take the intensity gradient of an object and extract its keypoints for that view. This process can be made iteratively while rotating the model and therefore acquiring multiple features for the same object.

8.4 Point Feature Histogram

Point Feature Histogram (PFH) was initially presented by Rusu *et al.* in [57]. It consist on a histogram of values that represents with a single histogram the alterations and most remarkable variations for a specific surface. FPFH histogram might be used to search similarities among different objects with scale invariability. This feature becomes an issue when databases are composed by scaled versions of similar objects at different scales (this circumstance is quite common for kitchen items such as spoons, mugs, glasses or dishes). Classifiers are always supported by a distance metric in charge of comparing candidates with the observed guest. The selection of a correct metric can affect directly to the recognition rates as discussed by Clemente in [7].

There exist multiple alternatives to represent and interpret surfaces. As a general rule those techniques analyze point cloud curvatures and normals along the surfaces to reduce a high dimensional problem into something more manageable. Most scenes will contain 3D clusters that may represent dissimilar objects or not, making it challenging to create a wide representation for 3D point clouds into a single feature list. As mentioned before, Rusu *et al.* proposed in [145] and [57] a novel point features representation named Point Feature Histogram, based on the curvature and normal orientation between neighbors in a certain point cloud. Since surface normals and curvature estimations are able to capture precise details of the geometry around a point, most algorithms base their intention focusing on them to deal with false correspondences.

In order to enhance recognition approaches, there exist many different feature representations for a surface. In general, it would be ideal to represent each point with an information label that contains the geometry class it belongs to: edge point, spherical surface point, etc.

Following the above, there is a need of finding a multi-dimensional feature space which separates surfaces in different categories. In terms of point-wise analysis, the concept of a dual-ring neighborhood is introduced for any point $p_i \in \wp$ as

$$(\exists) r_1, r_2 \in \mathfrak{R}, r_1 < r_2 \begin{cases} r_1 \Rightarrow \wp^{k_1} \\ r_2 \Rightarrow \wp^{k_2} \end{cases} \tag{8.19}$$

with $0 < k_1 < k_2$. $\wp$ represents the complete set of 3D points. Both radii r_1, r_2 have a specific target. The value r_1 represents the surface normal at the query point p_i obtained from the Principal Component Analysis formed by the neighborhood patch $\wp^{k_1}$. The radius r_2 delimits the PFH representation itself.

As it has been stated before, the main goal of the PFH formulation is to encode the $\wp^{k_2}$ neighborhood's geometrical properties by generalizing the mean curvature

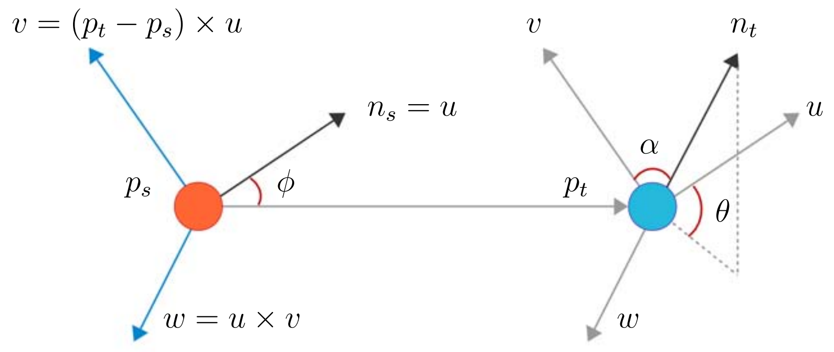


Figure 8.8: *Point Feature Histograms.* Darboux coordinates frame between source and target points when computing PFH

around p_i using a multi-dimensional histogram of values. Assuming that normal values of neighbors of p_i have been already computed, it is possible to state that having two different points p_i and p_j the relative difference between them can be defined as follows:

$$\vec{p_{ij}} = p_i - p_j, \vec{p_{ji}} = p_j - p_i \tag{8.20}$$

$$\begin{cases} if \cos^{-1}(\vec{n_i} \cdot \vec{p_{ij}}) \leq \cos^{-1}(\vec{n_j} \cdot \vec{p_{ji}}) & \Longrightarrow \begin{cases} p_s = p_i, n_s = n_i \\ p_t = p_j, n_t = n_j \end{cases} \\ otherwise & \Longrightarrow \begin{cases} p_s = p_j, n_s = n_j \\ p_t = p_i, n_t = n_i \end{cases} \end{cases} \tag{8.21}$$

being p_s the source and p_t the target. As the above condition takes place, p_s is chosen such that the angle between its associated normal and the line connecting the two points is minimal. It is then defined a Darboux coordinates frame is then defined at one of the points as shown in Figure 8.8.

The following mathematical statements are disposed:

$$u = n_s \tag{8.22}$$

$$v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \tag{8.23}$$

$$w = u \times v \tag{8.24}$$

With this in mind, the difference between the two normals n_s and n_t is defined as

$$\alpha = v \cdot n_t \tag{8.25}$$

$$d = \|p_t - p_s\|_2 \tag{8.26}$$

$$\phi = u \cdot \frac{(p_t - p_s)}{d} \tag{8.27}$$

$$\theta = \tan^{-1}\left(\frac{w \cdot n_t}{u \cdot n_t}\right) \tag{8.28}$$

The quadruplet $< \alpha, \phi, \theta, d >$ reduces itself from 12 initial values for both points to 4. The influence region diagram for a query point p_q can be represented as in Figure 8.9, where k represents the neighbors of p_q in a 3D sphere of radius r_1 and they are displayed as p_{k_i}.

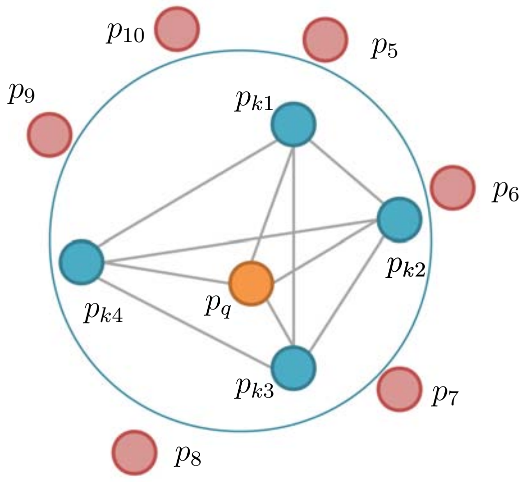


Figure 8.9: *Point Feature Histograms.* Influence region for a query point p_q when computing the PFH.

In order to encode the quadruplet a histogram has been created. The quadruplet values are divided into bins forming a histogram, each bin including the number of coincidences on this interval. Since three of the four values are angles, it is straightforward to divide the values into *equi-spaced* bins.

8.5 Fast Point Feature Histogram

A first improvement of PFH consists on taking into account the histogram of the neighbors and set for each point a histogram averaged with the surrounding points, so

- For each point p_q a set of tuples $< \alpha, \phi, \theta >$ are computed between a query point and its neighbors using the equations described above. This is called SPFH (Simplified PFH).
- Then, for each point p_q its histogram is recomputed taking into account a weighted value of neighbors SPFH

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=0}^{k}\frac{1}{w_i} \cdot SPFH(p_i) \tag{8.29}$$

where w_i represents a distance between the query point p_q and a neighbor point p_k in some given metric space (for this study Euclidean distance metric will be computed). FPFH includes additional point pairs outside the r_1 radius sphere with a re-weighting strategy that smooths local discontinuities and defines a better geometry around the query point.

8.6 Viewpoint Feature Histogram

Last enhancement applied to FPFH concerns to the view point of the object. As explained previously, one of the problems of FPFH is the view point dependence. To make the feature estimator view point independent, it has to be removed from the quadruplet necessarily. [161] proposes to maintain viewpoint variance while retaining invariance to scale computing additional statistics between the viewpoint direction and the normals estimated at each point creating a new descriptor named Viewpoint Feature Histogram (VFH).

The best way to remove this dependence is straight mixing the viewpoint direction directly into the relative normal angle calculation in the FPFH. A histogram of the angles that the viewpoint direction makes with each normal is computed. This is done from the viewpoint direction at the central point and each of the normals on the surface in order to maintain the feature scale invariant. The histogram then appears as in Figure 8.10.

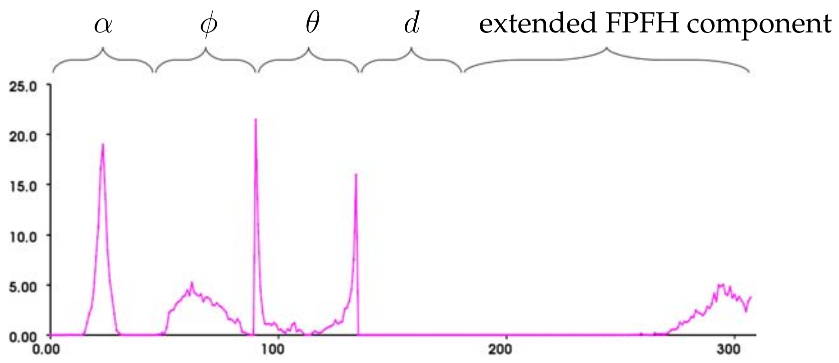


Figure 8.10: *Viewpoint Feature Histograms.* Representation of a Feature Histogram for a mug. It consists of 45 binning subdivisions for each of the 3 extended FPFH values, plus another 45 binning subdivisions for the distances between each point and the centroid and 128 bins for the viewpoint component.

As a conclusion, the usage of 3D descriptors based on keypoints are extremely useful to recognize and classify objects viewed from different poses. In this thesis, VFH descriptors are used in the following chapter to create a complete classifier that detects confusable objects. The geometrical description of the object pose will be determined by its VFH sign, the global object size and the texture information.

Chapter 9

3D Object meta-classifier

In this chapter, a meta-classifier to identify and distinguish between objects with confusable geometry, similar in appearance, colors and dimensions is proposed. The primary task of the proposed approach is to perform 3D scene labeling, which is to assign a label to every cluster point in a 3D scene. The suggested classifier has to be able to differentiate successfully between 3D point clouds with a high degree of similarity extracting parameters such as 3D features from the surface of the cluster, absolute geometrical boundaries and also color distributions. Objects are until now detected based on its Viewpoint Feature Histograms (VFH) with the downside of being scale independent, making the segmentation of items with similar geometry difficult. Furthermore, this thesis discusses several distance metrics to choose which one fits better with VFH descriptors based on the previous work by Bueno *et al.* [197]. Several experiments that validate the model and corroborate the meta-classifier are proposed.

As the quantity of new knowledge generated in the world quickly increments, proficient search in collections of structured data is required. 3D objects are a significant sort of multimedia information with many purposes. Latest advancements in procedures for modeling, digitizing and visualizing 3D models have led to an spread in the number of accessible 3D models in the Internet. For visualization, 3D models are represented as a surface, in particular as polygonal meshes. Many researchers have investigated the specific problems of content based on 3D shape retrieval and in the related fields of computer vision, object recognition, geometric modeling and obviously in robotics for manipulation and Human-Robot interactions.

Similitude seek in 3D models is becoming a notable mechanism in multimedia retrieval with many feasible applications such as CAD medicine, molecular biology or entertainment [198]. These utilizations lead to noteworthy diversity in the quantity of models growing in the past decade [199]. Utilization of the current models can be helpful in many events, which demands adequate methods for classifying or fetching the models in a large database. Searching and cataloging objects from a database are conventionally performed by using notes with the well studied keywords matching algorithm. It turns into a extremely difficult task to depict by words shapes that are not in well known shape or semantic categories. It is thus essential to have content-based search and retrieval systems for 3D models that are based on the geometric shapes features and color descriptors, as Sundar *et al.* states in [200].

Any classification process has to accomplish two basic rules to deal with 3D

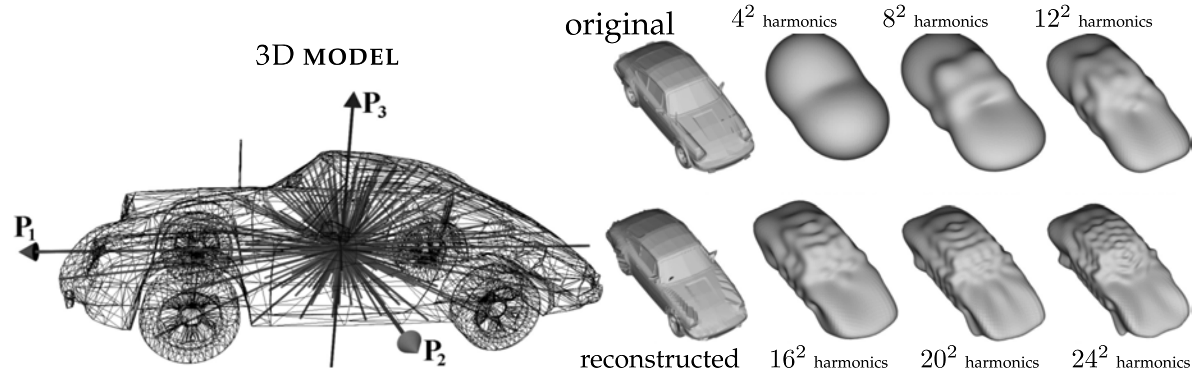


Figure 9.1: *Ray-based feature vector.* Back-transform of the ray-based samples from frequency to spatial domain at several frequencies

models. It has to interpret each model based on features or any possible representation and it has to be able to compare two different models.

- *Model representations.-* A custom or generic representation of an object based on its proper features must be set. These features may contain accurate information about any desired feature that characterizes the object and make it discriminative. Most literature base those features in 3D features, as explained in Chapter 8: VFH, SIFT-3D, NARF, etc. Those features can be mixed up forming a feature container for each object that contains color information, shape, texture, dimensions, surfaces, weight, usage, location, etc. A model can even be represented by a frequency Fourier transform of the surface and be back-transformed to spatial domain (see Figure 9.1).

- *Similarity measuring.-* The second requirement of a classifier is the measurement tool. Since it has to discern between objects stored in a database, it has to be able to compare their similarity. If the recorded features are arrays of numbers, a classical way to measure their similarity is to compute the distance between pairs of descriptors. Then, a custom matching function will determine a score indicating the degree of their resemblance.

Several methods are appropriate for similarity search requirement. The approaches should be invariant to alterations in the orientation, translation, reflection and scale of 3D models in their reference coordinate frame. They should also be robust with respect to variations of the level-of-detail and to small aberrations of the topology and geometry of the models. As Ronneberger *et al.* exposes in [201], the invariance and robust attributes are convenient for those practices that bear in mind relative objects properties or that combine a similarity measurement over the space of transformations. Otherwise these properties can be approximated by a preprocessing normalization, which transforms the objects so that they are represented in a canonical coordinates frame. In this form, directions and distances are comparable between different models inside the classifier. Therefore, taking in mind the two basic rules that any classifier has to interact with, this thesis will address a feature representation of the objects based on 3D shape, dimensions and a color descriptor. Furthermore, a list of distance metric candidates will be described in order to determine which metric yields better results during the similarity measuring.

9.1 Model Matching Methods

There exist two main alternatives for matching models. The first alternative are *feature vector*-based (FV-based)methods and the second alternative are graph based methods.

9.1.1 Feature Vectors

In order to measure the similarity of one item with respect to another, the 3D model is changed somehow to acquire a mathematical representation for indexing and retrieval, which is refered to as *feature vector* (FV). The primary intention is to select a list of numeric values that characterize the model under an unambiguous geometric form, and to find the similarity of the models from the distance of these feature vectors in some vector space. The consequent representation of a feature is considered as a descriptor.

Features are displayed by vectors with real-valued elements, and such descriptors are regarded as FV. Figure 9.2 shows the principle of a feature-based similarity search system (extracted from [202]). Descriptors should be defined in such a way that similar 3D models are attributed FVs that are close in search space. Apart from the 3D features that have been explained before, other characteristics might be used to extract a valid feature vector from any 3D point cloud. The similarity measure of two 3D objects is determined by a nonnegative real number, and usually, queries to the database are divided into two: *range* queries, which reports all objects from the database that are within distance lower than some tolerance value, and $k-$ nearest neighbors queries, which assume that the distance between the query and any other element is lower than a certain value.

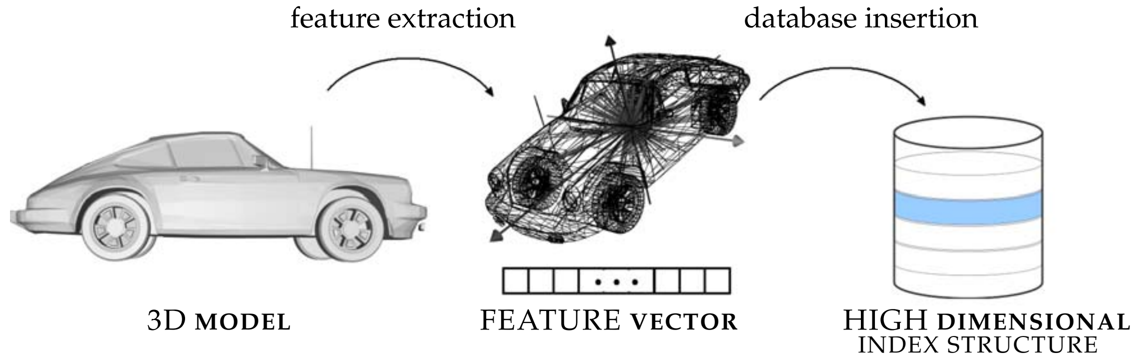


Figure 9.2: *Feature-based similarity search.* Each object is analyzed and a high dimensional feature vector is extracted. This vector is then inserted in a high dimensional index structure with the rest of the database members.

Silhouette Feature Vector method

Silhouette represents the region of a 2D image of an object which contains the projections of the visible points of that object. A silhouette can also be defined as an outline of a solid object, and a contour as a collection of boundary points of a silhouette. A silhouette feature vector proposed by Heczko *et al.* [203] designates 3D objects in terms of their silhouettes that are obtained from canonical renderings. The objects are earliest PCA-normalized and scaled into a unit cube that is axis-parallel to the principal axis. After that, parallel projections onto three planes, each orthogonal to one of the principal axes, are obtained. The algorithm suggests to recover a feature vector by concatenating Fourier estimation of the three resulting contours. To obtain such nearness, a silhouette is sampled by placing a predetermined quantity of uniformly spaced succeeding points on the silhouette, and computing the Euclidean distance among the image center and the consecutive outline points as the sampling evaluations.

Experimental results on the retrieval effectiveness of this descriptor were published in Vranice and Saupe [204], while Song and Golshani [205] tackled the usage of projected images for 3D retrieval. The authors suggested a structure to render object images from manifold directions and to employ various distance functions on resulting image pairs, for instance, based on circularity measures from the projections or distances between vectors of magnitudes after Fourier transform. There are much more works on image-based retrieval methods reported in Ansary *et al.* [206].

Depth buffer feature vector method

Heczko *et al.* [203] proposed another image-based descriptor called *depth-buffer* descriptor. It is also scaled into a unit cube and oriented as so. Six grey-scale images are extracted (two for each of the principal axes). Each pixel encodes in an 8-bit grey value the distance from the viewing plane of the object [15]. The six images are transformed using the standard 2D discrete Fourier transform and the magnitudes of certain k first low-frequency coefficients of each image contribute to the depth buffer feature vector of dimensionality $6k$.

Surface Voxelization based method

Saupe *et al.* [207] proposed a FV method based on the rasterization of a model into a voxel grid structure and then represent the feature in either spatial or frequency domain. The bounding cube addressing the object is subdivided in $n \times n \times n$ equally sized voxels. Distance metric for these descriptors is obtained applying a 3D Fourier-Transform in the frequency domain. Extracting the first k frequency coefficients, it is possible to perform a multi-resolution search.

9.1.2 Non-feature Vector Matching Techniques

As it has been explained, FV is focused on object characteristics such as spatial extent, surface curvature, 2D projections, etcetera. In contrast, Bustos *et al.* [202] determine that graph-based methods endeavor to obtain a geometric definition from a 3D shape using a graph to expose that the shape basics are associated to each other.

Model graph based similarity

Hilaga *et al.* [208] presented an approach to particularize the topology of 3D objects by a graph assemblage and showed how to apply it for matching and retrieval. The algorithm is based on an assembling so-called *Reeb* graphs from the models, which can be interpreted as information about the skeletal structure of an item. The fundamental concept is to divide the object into linked pieces by examining a function μ that is bounded over the entire object's surface. The *Reeb* graph created from a 3D object is made up of nodes that signify divisions of the object for which μ assumes values ranging in specific value intervals. Parent-child relationships between nodes represent adjacent intervals of these function values for the contained object parts. Similarity among objects is determined after the comparison of the topology of the respective *Reeb* graph for both items. The selection of

[15]These images correspond to the concept of $z-$ or $depth-$ buffers in computer graphics

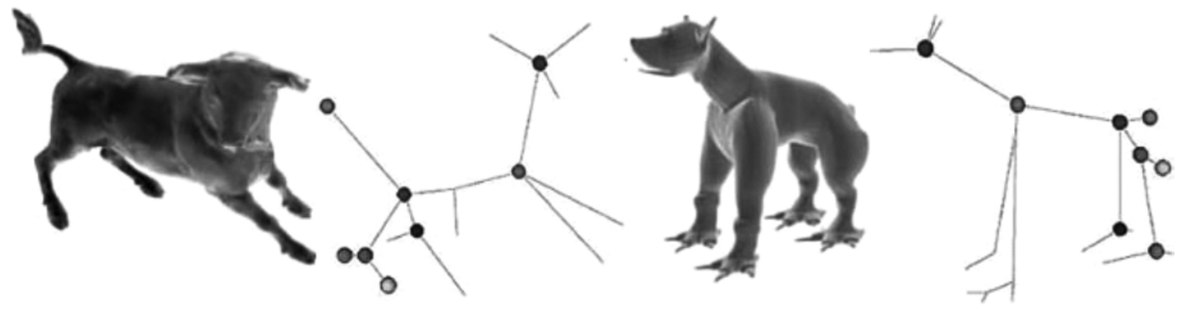


Figure 9.3: *Skeleton based similarity.* Two objects and their corresponding skeleton [200]

μ function is critical to the construction of graphs suited for object analysis and matching.

Skeleton based similarity

Skeletons acquired from solid objects can be noticed as instinctive object descriptions. Sundar *et al.* [200] used a skeletal graph shape descriptor encoding geometric and topological information at the same time. To obtain a thin skeleton, the authors recommended to apply a thinning algorithm on the voxelization of a solid object. With that method, the model voxels are reduced to only those voxels that are crucial for object reconstruction. This is determined by a heuristic that associates the distance transform value of each voxel with the mean of the distance transform values of the voxels among its 26 neighbors [209]. In a second step, the remaining voxels are clustered, and a minimum spanning tree is constructed connecting the voxel clusters. Figure 9.3 represents the final skeleton of two objects and its corresponding volumetric representation.

9.2 Architecture of the Classifier

In this section, a novel classifier for confusable objects is presented. Its main purpose is to permit the robot to categorize uncertain objects lying on a table and also to distinguish all of them. As it will be shown, most of the objects are, on purpose, extremely similar in shape. One of the challenges faced in this thesis is the implementation of a classifier that attends not only to one specific feature of the object but to the combination of several others. In order to classify correctly objects with similar surfaces but different scales, it has been proposed a meta-classifier that not only takes into account the VFH histogram but also the external geometry of the object (height, width and depth dimensions) and the color texture to make the robot recognize most objects in order to grasp them. Therefore, this research is focused on the extraction of the optimal result from the database in order to order the robot, once the object has been recognized, to plan a safe path with the arm and then grasp it.

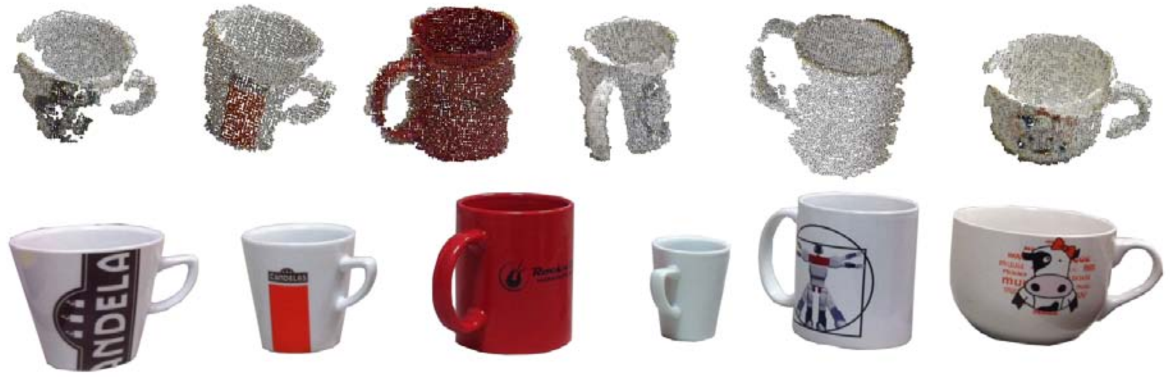


Figure 9.4: *Database created by the author.* Representation of 2D and 3D views for several objects acquired during the classification process. Notice that most of the items are geometrically similar.

For this study, a 3D household items database has been created. It has been developed by the author and contains a list of 15 objects. Each object is formed by 20 views from different point views, bringing the total of 300 individual point cloud views. The main feature of this database is that all the objects are quite similar in terms of shape. Figure 9.4 shows a representation of some of the items included in the author's database created specifically for this study. As can be noticed, there exist objects with similar surface shape but different size or color. All the objects are labeled and ground truth poses are also given. All of them will be used in these experiments as the final intention of the presented meta-classifier is to distinguish specifically between confusable objects.

Geometry features contain absolute measures in the three axes giving more significance to the height value. That condition is imposed to maintain the viewpoint invariance since width and depth depend on the rotation of the object, while height maintains constant relative to the supporting plane and detached from yaw angle. The following subsections explain each part of the filter and the overall meta-classifier.

As a setup stage, the list of object views have to be analyzed. For each view of a certain object, all the involved features are extracted. The first one is the VFH

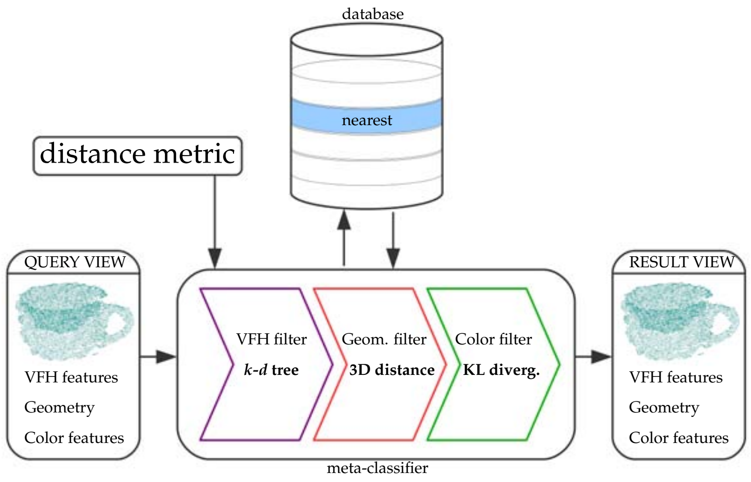


Figure 9.5: *Meta-Classifier schema.* Diagram of the classifier proposed for this study. Distance metric can be changed before filtering is done. The meta-classifier takes into account 3D surface descriptors, color and geometry for each query and returns the closest candidate in the database.

descriptor explained in the later chapter. This descriptor will determine the form of the surface in a local manner. Furthermore, this feature is scale invariant, representing a problem for grasping as it may let the robot to get confused if two similar objects of different scale are presented to it. For this reason, a second filter will determine the global dimensions of the view, allowing the robot to distinguish between similar objects at different scales. Finally, those objects that are most of the time similar in size and shape but not in color (examples of this could be the salt and pepper recipients or an orange versus apple, etc.) have to be considered. Thus, in order to classify those last cases, a color classification is processed using the same technique as the explained in the evolutionary reconstructor (see section 7.4.1 for more information). The color distance for every pixel of the cluster will be computed and the classifier will discern the most probable one.

However, it may happen that an object is not yet in the database or needs to be re-learned for some reason: it has changed its color, it is not rigid and its original pose has been altered or it has been simply folded. In order to deal with these situations, the classifier has been provided with a learning routine that will introduce a new object in the database. In order to do this, the robot will ask for a bunch of views of the concerned object. The classifier will memorize the first view's descriptor and it will remain acquiring information from new views until the original view is shown again. In that moment the system will ask for an object's name and the new object will be attached to the database.

9.3 Distance Metric

As stated before, there exist two main metrics for feature vectors. The similarity measurement of two 3D objects is determined by a real number that represents the feature distance in a high level search space. Generally, a similarity measure is, therefore, a function of the form

$$\delta \,:\, Obj \times Obj \Longrightarrow \mathbb{R}_0^+ \tag{9.1}$$

where Obj is a suitable space of 3D objects. Similarity is then measured in terms of δ. A small value of δ represents strong similarity (small distance) and, equivalently, a high value of δ denotes a dissimilarity.

Let $\mathbb{U}$ represent the 2D object database and let q be the query 3D object in question. There exist basically two types of similarities.

- *Range queries.-* A range query (q, r) for a tolerance value $r \in \mathbb{R}^+$ reports a list of objects from the database that are within a distance r to q; that means that $(q, r) = \{u \in \mathbb{U}, \delta(u, q) \leq r\}$.
- $k - NN$ *queries.-* It returns a list of the k objects from $\mathbb{U}$ closest to q. It means that it yields a set $C \in \mathbb{U}$ such as $|C| = k$ and $\forall u \in C, v \in \{\mathbb{U} - C\}$, $\delta(u, q) \leq \delta(v, q)$.

It has to be considered a $d-$dimensional FV to perform similarity comparisons. Classical systems would report $\delta(u, v)$ as a metric distance $L(\vec{x}, \vec{y})$ in the $d-$dimensional space of FVs where $\vec{x}$, $\vec{y}$ represent the FVs of u and v, respectively. The most popular family of similarity functions in vector spaces is Minkowski L_s family of distances

$$L_s(\vec{x}, \vec{y}) = \left(\sum_{1 \leq i \leq d} |x_i - y_i|^s \right)^{\frac{1}{s}}, \; \vec{x},\, \vec{y} \in \mathbb{R}^d,\; s \geq 1 \tag{9.2}$$

Most common variations of this family are *Manhattan* distance L_1, Euclidean distance L_2 and the maximum distance L_∞, $L_\infty = \max_{1 \leq i \leq d} |x_i - y_i|$.

If the feature components correspond to histogram data, as it is proposed in this thesis, several further extensions to the standard Minkowski distance can be featured. In the context of image similarity search, color histograms are often used.

The descriptors then consist of histogram bins, and cross-similarities can be used to reflect natural neighborhood similarities among different bins. In order to measure distances between distributions, several distance metrics have been proposed.

One of the aims of this research is to determine the degree of reliability for several variations of the Minkowski family and also the Kullback-Leibler divergence, already mentioned in this research. The main purpose of this study is to determine the performance of each alternative and also to select the most accurate distance metric to distinguish between different VFH histograms and color histograms. The list of functions and their equations can be found in Table 9.1. It will be assumed that all the histograms are normalized in order to satisfy the metrics conditions.

METRIC FUNCTIONS	
Distance metric	Distance function
Euclidean	$d = \sqrt{\sum_{i=1}^{N} (p_i - q_i)^2}$
Manhattan	$d = \sum_{i=1}^{N} \lvert p_i - q_i \rvert$
Bhattacharyya	$d = -ln \sum_{i=1}^{N} \sqrt{p_i - q_i}$
Kullback-Leibler	$d = \sum_{i=1}^{N} p_i \ln \frac{p_i}{q_i}$
Chi-Square	$d = \sum_{i=1}^{N} \frac{(p_i - q_i)^2}{(p_i + q_i)}$
Histogram Intersection Kernel	$d = \sum_{i=1}^{N} \min(p_i, q_i)$

Table 9.1: Distance metrics used for this study. Most of them are Minkowski family functions and divergence functions

9.4 Surface Shape Filter

The earlier filter in the classifier extracts the VFH descriptor for the observed cluster and compares it with the descriptors listed in the database for each view and for each object. Because this operation might spend a lot of time, a *k*-NN tree is previously built with the catalog of objects already added to the database. This advancement achieves faster execution times. Therefore, the VFH histogram will be compared with each member of the database. This comparison requires a distance metric. The metrics already mentioned will be evaluated in order to extract the most sensible distance function.

The aim of this first procedure is to sort the database in order of 3D shape similarity. From the whole list, only the ten most similar candidate views will be taken into account. Those elements are supposed to be the majority of objects with the same surface shape. Furthermore, those aspirants could have different scales or textures but the same surfaces. Figure 9.6 represents a list of the five nearest neighbors for a single candidate that is also shown in the upper part. The list is sorted from left to right in order of similarity using Euclidean distance as the classification metric. Surprisingly, all the resulting views do not belong to the same candidate. This happens because of the nature of VFH. As this descriptor is exclusively based on geometrical shapes and the selected database contains quite similar and confusable objects, this results become predictable. The mission of the meta-classifier is to refine the results and extract the most suitable result based on the similarity.

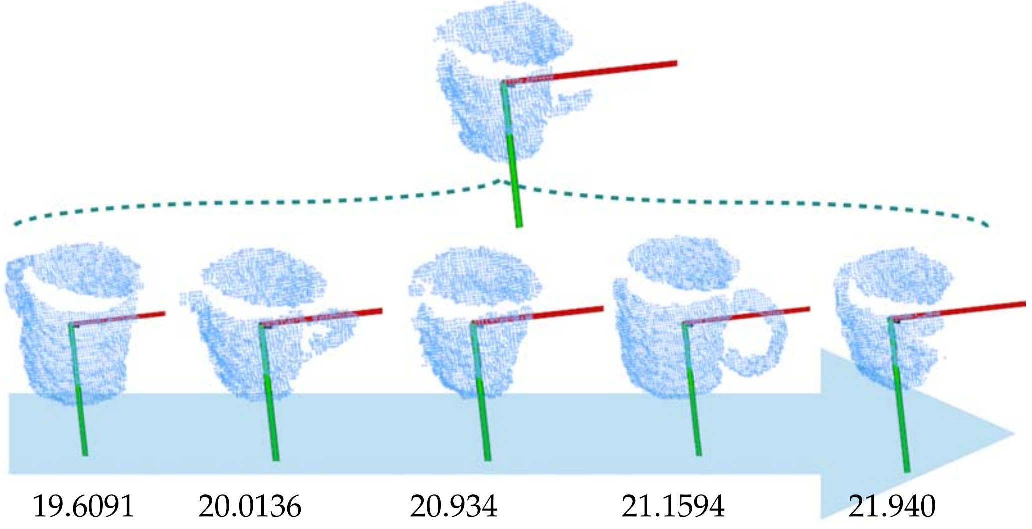


Figure 9.6: *k-NN distance results for a given VFH.* Result of searching for similar objects in the database. The top item is the query and the above candidates are the most similar sorted by VFH distance.

The *k*-NN tree modeled contains, for each leaf, the VFH histogram of a single view for a specific object. VFH implementation uses 45 binning subdivisions for each of the three extended FPFH values, plus another 45 binning subdivisions for the distances between each point and the centroid and 128 binning subdivisions for the viewpoint component yielding a total of 308-length vector. Figure 9.7 represents a set of views of the same object and their individual VFH values making it easy to understand the structure of the database.

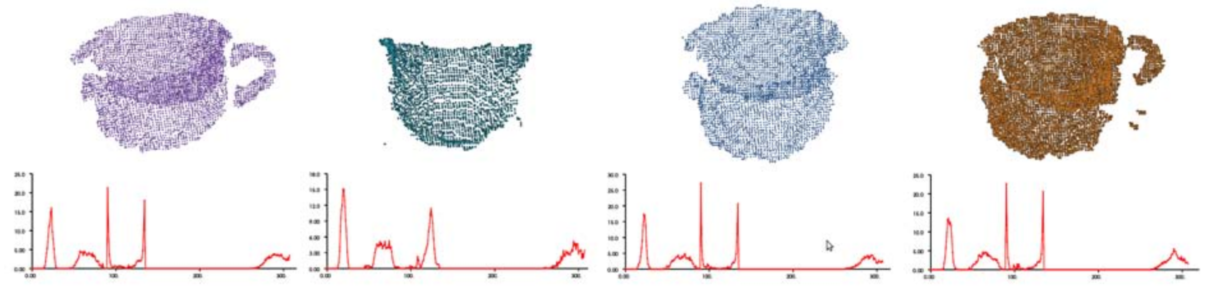


Figure 9.7: *Representation of VFH histograms.* Illustration of several items in 3D and their corresponding View-Point feature Histogram in red. Histograms look quite similar due to their shape similarity.

The small difference among VFH histograms for each view makes it mathematically complex to determine the dissimilarity between them. That is the reason why several distance metrics have been evaluated. In order to transform histograms into distributions, they have been weighted making them unitary on those where it was necessary.

9.5 Geometrical and Color Filtering

As mentioned before, VFH is a scale-independent feature and therefore is not able to distinguish between objects with the same shape but different size or texture. To solve this deficiency, several options have been proposed. Taking into account the real dimensions of the cluster, it is feasible to classify objects at various scales. This dimensional feature is highly dependent on one factor that was mentioned in Chapter 4: the plane fitting estimation. As long as the plane is well fitted on the supporting surface, the computation of the height dimension of the view becomes more and more confident. However, if the plane is poorly estimated, this factor in the classification becomes less advantageous.

Furthermore, color features have been extracted for each view to classify correctly the candidates. Several color space representations (RGB, HSV and SciLab) have been compared in order to determine the best classification feature.

9.5.1 Bounding Box Extraction

The subsequent filter step of the classifier is focused on absolute dimension values of the view: height, depth and width. The most valuable measure of these three is the height, since either width or depth are weak to point of view or object rotation. Height is almost view-point invariant as the higher part of the point cloud will always remain observed from the camera coordinates, and therefore successfully determined. Contrary to that, depth or width can be mistaken due to occlusions or convex surfaces. Furthermore, if the observation errors are taken into account, height value is the most confident value, since the object is always lying on a table. Thus, at least the minimum height of the bounding box will be ensured by the plane estimation, while the other two dimensions are freely modifiable. Table 9.2 shows the variability of a static object lying on a table over 100 measures.

OBJECT DIMENSIONS		
width	height	depth
$\mathcal{N}(0.1244537, 0.079865)$	$\mathcal{N}(0.098416, 0.053207)$	$\mathcal{N}(0.081089, 0, 101128)$

Table 9.2: Bounding box parameters variance. A rigid cube of $0.10 \times 0.10 \times 0.10\, m^3$ is measured 100 times in order to determine the stability of the observed dimensions.

The selected object is a cube of $0.10 \times 0.10 \times 0.10\, m^3$. It has been measured 100 times and the resulting dimensions have been approximated to three normals, one for each direction. As can be seen, the most stable and accurate one is the height of the cube with an average of 9.84 cm. The most unstable measure is the depth of the cube as floating pixels may vary the boundary in this direction. Therefore the height remains as the best qualification for the geometrical filtering.

An arising problem when capturing these three dimensional values becomes while determining the absolute position of the object according to the supporting plane world position. For this reason, it is necessary not only to extract the object cluster but also to project the point cloud to the coordinate system formed by the supporting plane and the point view coordinates. Figure 9.8 represents the cluster point cloud of a teddy with the three views projected based on the parametric equation of the table plane $\pi(\pi_x, \pi_y, \pi_z)$.

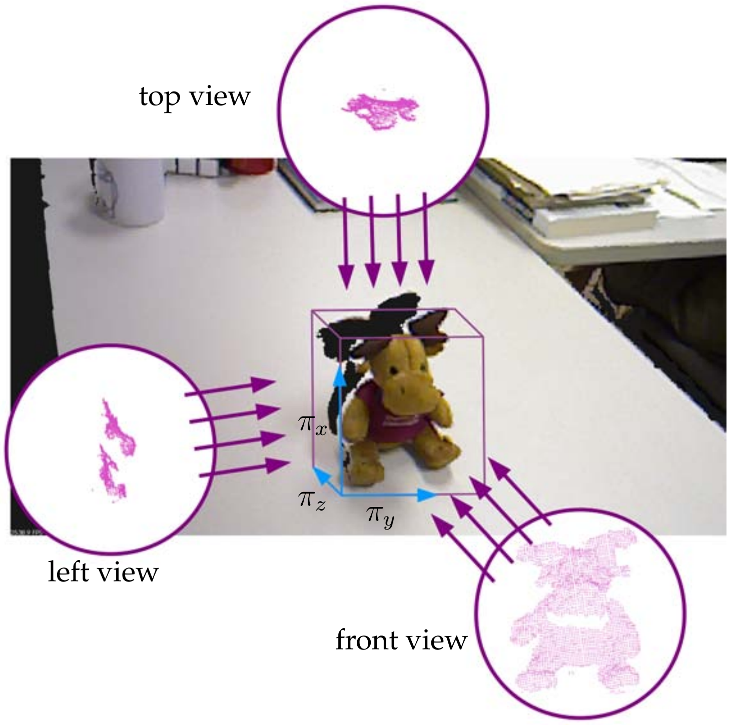


Figure 9.8: *Projected bounding box for a cluster.* Bounding box of a point cloud projected on each axis referenced to the normal of the supporting plane π

9.5.2 Color Filter

Last step of the classifier is focused on determining differences in texture between similar candidates. This last condition makes the complete architecture able to distinguish between objects with approximate surface shapes and identical geometrical sizes but dissimilar colors.

There exist multiple ways of comparing textures among objects. For this research, each color histogram R, G and B will be converted into a HSV histogram and then assumed as an independent distribution. Figure 9.9 shows the RGB distributions of two mugs that might be distinguished clearly. Distributions change slightly for each mug. For the first histogram relative to the redish mug, most of the red distribution is assigned to the higher values of the red channel, while green and blue values are maintained low. Contrary to this, the white mug distribution obtain high values for the three channels assembling most of their values to the tail of the sub-distributions. Note the density dimensions in both distributions. However, this is not enough for many cases since orange and red colors are quite confusable for RGB color space. For this reason, textures are transformed into HSV before being compared.

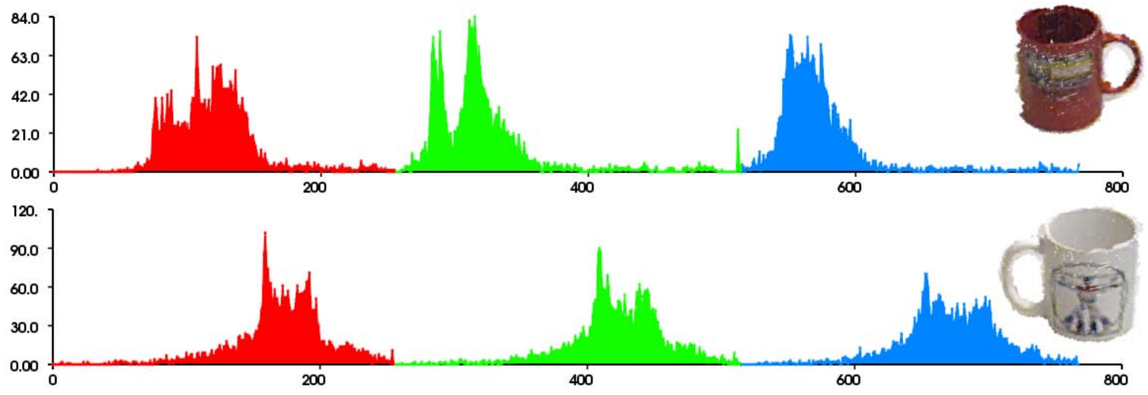


Figure 9.9: *Illustration of color histograms in RGB space.* Representation of the color histograms created by the classifier. Only extremely different textures can be distinguished easily.

The distance metric will be the same as the one proposed for the object reconstructor. The distance metric between objects will be the Jensen-Shannon divergence. Each color distribution will be normalized to remove cluster size dependency. From the three channels, Hue will be used in conjunction with Saturation, while Value will be avoided.

Figure 9.10 a search by color similarity is displayed for a single object of the database using the HSV space color. The upper element is the query and the list below is the result of the color matching using the Jensen-Shannon divergence as distance metric. As it is shown in the image, the color is not homogeneous and the HSV histogram might get contaminated with noisy points and floating pixels. However, the proposed method is robust for this kind of alterations in texture and illumination.

9.6 Experimental Results

The proposed detection-based 3D scene labeling has been evaluated on the RGB-D Object Dataset presented before, which contains everyday objects captured individually in a non-controlled scenario. That means that turntable or similar techniques were not used to acquire the models. All the models have bee acquired

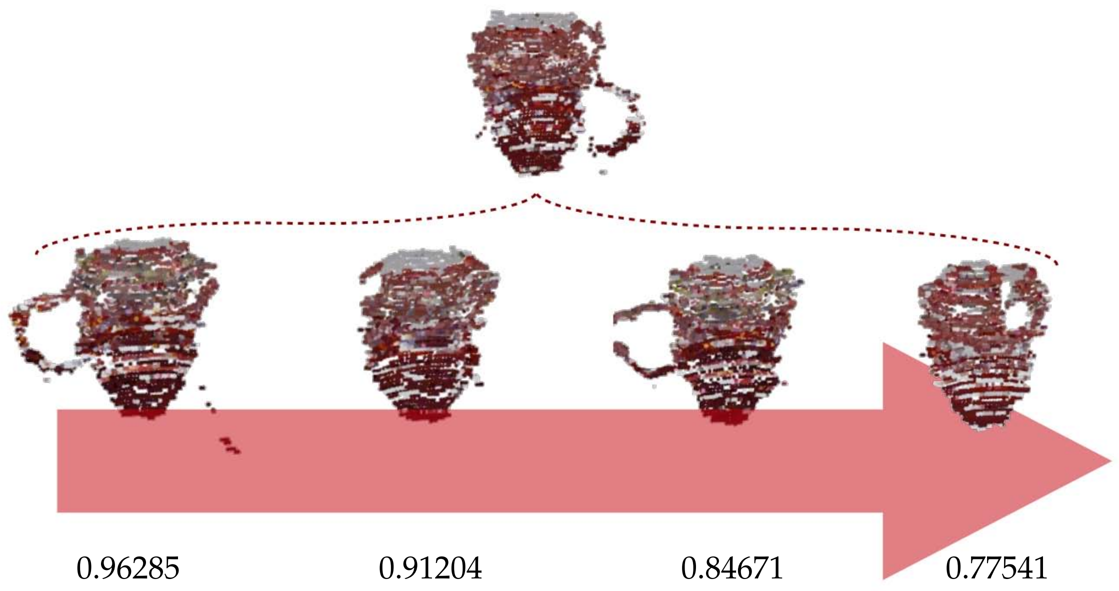


Figure 9.10: *Color distance results for a red mug.* Result of searching for color-like objects in the database on the HSV space. The top item is the query and the above candidates are the most similar sorted by color distance with the corresponding distance on each footer.

using the 3D sensor provided to the robot (details about the sensor are addressed in Chapter 2). The meta-classifier has been trained using captured objects individually and evaluates scene labeling on point clouds reconstructed from the RGB-D scene videos. Furthermore, an extensive discussion of the proposed classifier is done afterwards with a set of confusion matrices determining the accuracy of each part of the classification. Results comparing the proposed technique will be held and finally some examples of 3D labeling in real scenarios will be presented.

A good way for discussing the accuracy of an object recognition algorithm with similar candidates is creating a confusion matrix with them. This type of studies compare all the members and explain the correlation between them, similarities and the degree of segmentation. For this study, several confusion matrices will be shown. The first experiments will cover the dependency of distance matrices when comparing VFH features and RGB histograms. The last investigations will focus on the performance of the meta classifier and its accuracy depending on the filters enabled.

9.6.1 Distance Metric Performance

In this experiment, several distance metrics have been evaluated in order to measure their performance and select the most suitable function to evaluate the distance between feature histograms. As it has been mentioned in the previous section, six classic distance functions will be used: Manhattan distance (L_1), Euclidean (L_2), Bhattacharyya, Kullback-Leibler + Jensen Shannon divergence[16] , Chi-Square distance (χ^2) and Histogram Intersection Kernel.

Their performances have been represented in Table 9.3 with a single candidate `tiny-mug-coffee` from random poses and distances relative to the sensor view point with the aim of obtaining realistic observations. The classifier will perform the three-stage classification for this experiment: shape, dimensionality and color.

SHORT DISTANCES					
Manhattan L_1	Euclidean L_2	Bhattacharyya	K-L+ J-S	χ^2	HIK
63 %	66 %	46 %	**84 %**	74 %	23 %
LARGE DISTANCES					
Manhattan L_1	Euclidean L_2	Bhattacharyya	K-L + J-S	χ^2	HIK
58 %	61 %	45 %	**81 %**	63 %	22 %

Table 9.3: Recognition rates for different distance metrics at short distances (0.4 - 0.5 m) and large distances (0.9 - 1.0 m). The mixture of Kullback-Leibler divergence for VFH descriptor and Jensen Shannon divergence for color histogram presents the most accurate results for VFH comparisons

The above results have been computed using the following equation

$$\text{performance} = 100 \cdot \frac{\sum_{i=1}^{N} 1 - |o_i - r_i|}{\sum_{i=1}^{N} r_i} (\%) \tag{9.3}$$

Results show up two main conclussions:

- The best performance is yielded by using Kullback-Leibler divergence when comparing full feature histograms. The robustness of this metric is quite recommendable for features comparison and therefore, it will be used in the subsequent experiments.

[16]Kullback Leibler divergence for measuring VFH dissimilarity and Jensen-Shannon divergence for color histograms

- Distance affects the recognition rate, since the number of observable points decreases. As the robot interacts in a workspace between 0.4 and 0.8 meters, this dependency does represent a problem that might be taken into account.

9.6.2 Depth Reliance

For a better understanding of the behavior of the meta-classifier with respect to the distance of the objects, a statistical analysis has been performed in these terms. Several objects have been classified independently at different distances in order to determine the relation between depth and the recognition rates. For each item, the classifier has been tested at different depths, the size of the observed point cloud and the results are presented in Table 9.4.

From this table several conclusions are extracted. Firstly, there exists a strong relation between performance rate and the relative distance between the sensor and the object. This dependency basically decreases with the cluster size, since it decreases as long as the object is moved away. Therefore, the amount of information directly affects the classification score.

9.6.3 Confusion Matrices

The next experiment is focused on measuring the improvement of the filter as long as it is refined. That is, by means of confusion matrix it is feasible to compare detection rates for similar candidates and therefore understand the degree of confusion or precision of the filter. Three matrices have been computed. Each one represents a different version of the filter. First version (Table 9.5) includes only the VFH filter, next version includes the geometric filter (Table 9.6) and the last differentiate among colors and textures (Table 9.7).

VFH filter

The average recognition rate for this filter is **50.85%**. Notice that it uniquely takes into account the 3D surface shape of each object, confusing objects with similar surfaces such as `mug-robotics-white` and `mug-robotics-red`, which are highly confusable. The same is happening between `mug-medium` and `mug-big`, where the recognition rates are below 35%.

Depth dependency			
Object name	Depth [m]	Cluster size [points]	Performance
`tiny-mug-coffee`	0.40	1487	84%
	0.60	1190	82%
	0.80	1039	81%
`medium-mug-coffee`	0.40	2004	60%
	0.60	1721	60%
	0.80	1445	57%
`big-mug-coffee`	0.40	2395	96%
	0.60	2137	92%
	0.80	1866	87%
`mug-robotics-white`	0.40	3544	100%
	0.60	2946	98%
	0.80	2434	98%
`mug-robotics-red`	0.40	3444	88%
	0.60	2730	85%
	0.80	2287	80%
`mug-cow`	0.40	4083	100%
	0.60	3411	100%
	0.80	2813	95%
`teddy`	0.40	3303	100%
	0.60	2969	100%
	0.80	2607	98%

Table 9.4: Recognition rates at several distances (0.40, 0.60 and 0.80 m.) There exist a strong relationship between the classifier performance rate and the depth at which the object is positioned.

VFH+Geometry filter

This second experiment takes into account not only the surface features VFH but also the global geometry of the object. With this addition, it is expected to decrease the confusion error between objects with similar shape but different size, such as the `mug-tiny`, `mug-medium` and `mug-big`, where confusion has decreased significantly. Note that `mug-cow` and `mug-big` are now much better segmented due to their distinction. The average recognition rate for this test is **75.85%**.

	mug-tiny	mug-medium	mug-big	mug-r-white	mug-r-red	mug-cow	teddy
mug-tiny	**7**	41	3	19	30	0	0
mug-medium	0	**29**	26	18	27	0	0
mug-big	0	0	**84**	1	8	5	2
mug-r-white	0	0	0	**26**	74	0	0
mug-r-red	0	0	0	49	**51**	0	0
mug-cow	0	0	34	0	0	**59**	7
teddy	0	0	0	0	0	0	**100**

Table 9.5: Confusion matrix for VFH filter. Note the high degree of confusion among scaled versions of the same objects.

	mug-tiny	mug-medium	mug-big	mug-r-white	mug-r-red	mug-cow	teddy
mug-tiny	**49**	29	0	10	12	0	0
mug-medium	0	**40**	11	11	38	0	0
mug-big	0	0	**99**	1	0	0	0
mug-r-white	0	0	0	**77**	23	0	0
mug-r-red	0	0	0	30	**67**	0	3
mug-cow	0	0	1	0	0	**99**	0
teddy	0	0	0	0	0	0	**100**

Table 9.6: Confusion matrix for VFH+Geometry filter. Objects with similar geometry but different textures are still confused.

VFH+Geometry+Color filter

Last filter compares VFH features, geometric features and color for each object. In this case the confusion matrix's diagonal is strictly higher and confusion is well diminished. `mug-robotics-red` and `mug-robotics-white`, which are geometrically exact but with different color, are well segmented and recognized, while the rest of the distinction rates are maintained as expected. The average recognition rate for this last experiment which includes the three filters is **89.71%**.

Therefore, the three-layer classifier presented in this chapter is able to classify with objects lying on a table with the following requirements:

1. *Objects that contain different surface shapes.*- Depending on the shape of the surface of the object, a VFH descriptor is able to distinguish dissimilar contours. Those features are local features of each object that represent the singularities and curvatures of a certain surface.

2. *Objects with similar shape but different scale.*- Since VFH descriptors are scale

	mug-tiny	mug-medium	mug-big	mug-r-white	mug-r-red	mug-cow	teddy
mug-tiny	**84**	12	0	4	0	0	0
mug-medium	20	**60**	20	0	0	0	0
mug-big	3	0	**96**	0	0	0	0
mug-r-white	0	0	0	**100**	0	0	0
mug-r-red	0	0	0	11	**88**	0	1
mug-cow	0	0	0	0	0	**100**	0
teddy	0	0	0	0	0	0	**100**

Table 9.7: Confusion matrix for VFH+Geometry+Color filter. Confusion matrix is highly diagonal, stating the degree of success for the filter.

invariant, last requirement does not fulfill the distinction of objects with similar surface shape but different scale. This second layer of the classifier is in charge of measuring the global dimensions of the view and filter those views with incongruous values.

3. *Objects with equal shape and scale but contrasting textures.*- Last layer of the classifier is in charge of selecting the most suitable candidates based on color features. As the first two stages of the filter are based on 3D information, the last part analyzes the HSV histogram of the view and determines which is the closest view from the remaining aspirants.

In Figure 9.11, the 3D scene labeling results for four complex scenarios are displayed. On the left the original scenarios and on the right the result of the three-layer meta-classifier. Objects are colored by their category according to Table 9.8.

COLOR REFERENCE	
object name	color
mug-coffee-big	🟥
mug-coffee-medium	🟧
mug-coffee-tiny	🟩
mug-cow	🟨
teddy	🟪
mug-robotics-red	🟥
mug-robotics-white	🟩

Table 9.8: Classifier color reference. Relation between objects and color for the classifier.

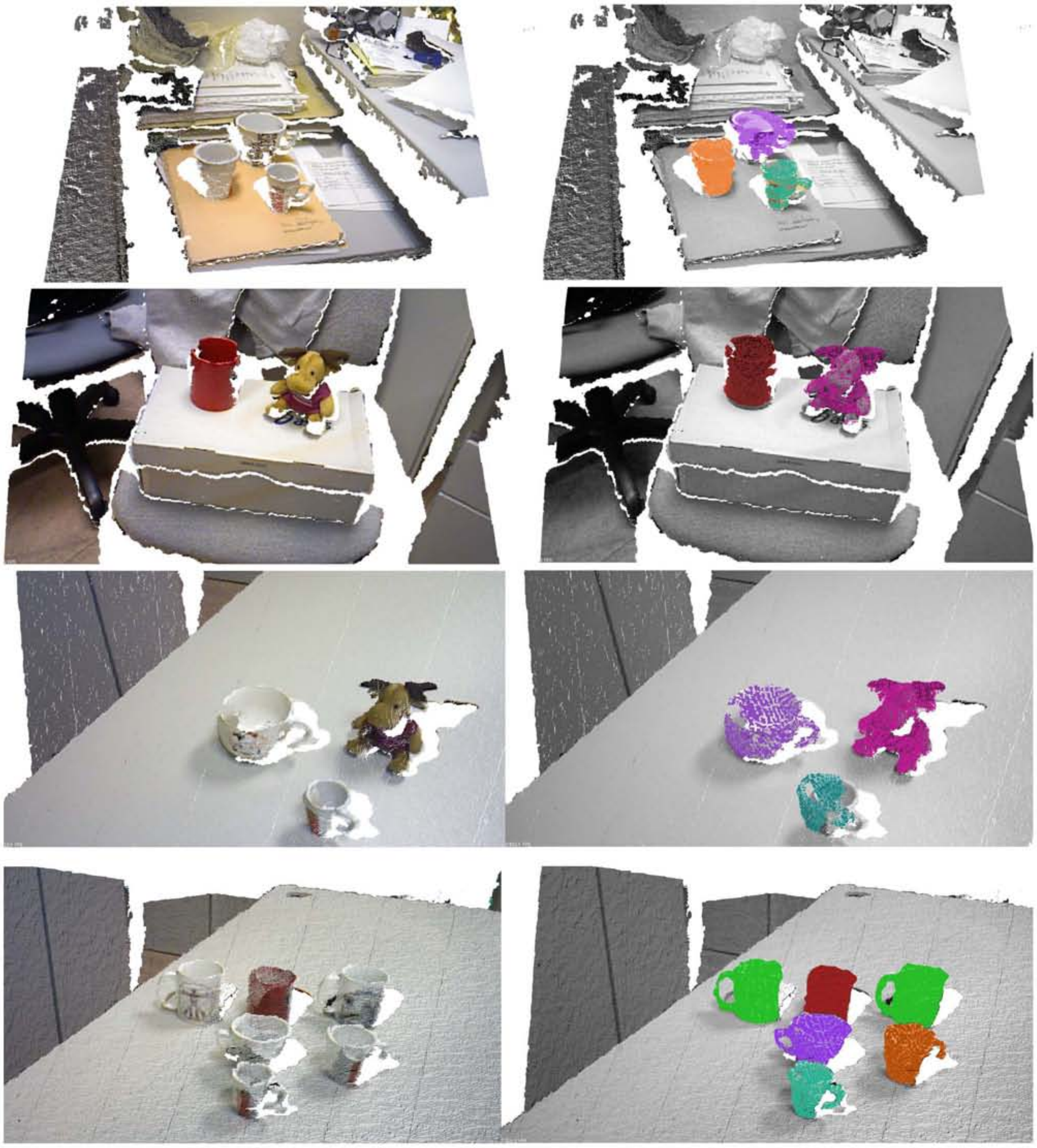


Figure 9.11: *Classifier results for complex scenarios.* Representation of the classifier results. On the left the original scenarios and on the right the result of the three-layer classification. Objects are colored consistently.

9.6.4 Updating the Database

One of the aims of this thesis regards on the dynamism of the classifier. The idea is to make the robot able to learn new objects on the fly. Contrary to other offline systems such as Lai *et al.* [210], the meta-classifier here proposed is able to update its database in real time and let the robot learn new objects with the minimum human intervention.

If the robot detects an unknown object and it is interesting for it to learn it, the classifier activates the learning mode. In this state, the classifier will acquire a list of views $\mathcal{V}_i$ at a constant rate of the unknown object in different poses and store them in a buffer $\mathcal{B}$ such as

$$\mathcal{B} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_N\} \tag{9.4}$$

Each view will be processed separately and a descriptor ψ_i will be associated to that view so that

$$\psi_i = \Psi(\mathcal{V}_i) \tag{9.5}$$

where $\Psi(\Delta)$ is a function that extracts the three-layer descriptor of the view (VFH descriptor, bounding box dimensions and HSV histogram). If the robot considers that it has enough information about the object, it will stop recording data and will ask for a natural name for that object. Then, the algorithm will include the new information stored in $\mathcal{B}$ and it will re-structure the full database with the new object.

In order to make the classifier work autonomously, two things have to be taken into account:

- *Relative comparison.-* The most probable case is that the 3D sensor acquires frames at rates faster than the robot movements. Therefore, there must exist a reject function that controls the number of views introduced since the information may become redundant. For a view i this is done such as

$$\begin{cases} d(\psi_i, \psi_{i-1}) \geq \tau & : \quad \{\mathcal{B}\} \leftarrow \mathcal{V}_i \\ d(\psi_i, \psi_{i-1}) < \tau & : \quad discard(\mathcal{V}_i) \end{cases} \tag{9.6}$$

  That is, if the distance d in the high dimensional descriptor space between two consecutive views is smaller than a certain threshold τ, then the view is discarded. Otherwise, the buffer array is updated with ψ_i.

- *Close loop.-* Again, in order to finish the acquisition stage, the classifier needs a exit condition. Literature recommends to close the loop comparing the actual view with the initial statement. Again, this comparison will be afforded using a custom distance d. According to that, the expression would be

$$\begin{cases} if\, d(\psi_i, \psi_0) \geq \sigma & : \text{continue();} \\ if\, d(\psi_i, \psi_0) < \sigma \,\&\, (i > \epsilon) & : \text{continue();} \\ else & : \text{exit();} \end{cases} \tag{9.7}$$

where a new parameter ϵ has to be introduced that represents the minimum number of views recommended to recognize the object in the future. A good value for this parameter is $\epsilon = 5$. However, the values of τ and σ depend on several things, principally on the distance function selected. Nevertheless, these values have been established empirically for this research, even they could be determined as adaptive variables finding a relation among the parameters or statistically analyzing the behavior of the variables between them.

The distance function d proposed for this part is the multiplication of three unitary distances. As stated before, the best measurement between VFH descriptors is Kullback-Leibler divergence $d_{KL}(\psi_i, \psi_j)$. For color histograms the best metric is the Jensen-Shannon divergence $d_{JS}(\psi_i, \psi_j)$. Finally, in order to compare the dimensions of the views, a unitary vector will be constructed respectively, addressing the three values for each axis $d_{DI}(height, width, depth)$ such that $\|d_{DI} = 1\|$. Therefore, the distance metric will be

$$d(\psi_i, \psi_j) = d_{KL}(\psi_i, \psi_j) \cdot d_{JS}(\psi_i, \psi_j) \cdot d_{DI}(\psi_i, \psi_j) \tag{9.8}$$

Figure 9.12 represents the evolution of each distance through the acquisition of a new object in the database. The first point corresponds to the initial view of the object $\mathcal{V}_0$ and due to the first restriction (Equation 9.7). After overpassing the condition, the classifier will analyze each view as shown in the graph. The next highlight happens at iteration 31 since it is when Equation 9.7 moves to the exit condition. Some snapshots of the experiment have been introduced in Figure 9.13. The number of the iteration has been included to understand better the behavior of the global distance function.

In this chapter, a novel meta-classifier has been proposed. It has been designed to recognize and classify objects lying on a supporting place in an unexplored scenario. Furthermore, the system is able to extend its database by acquiring new

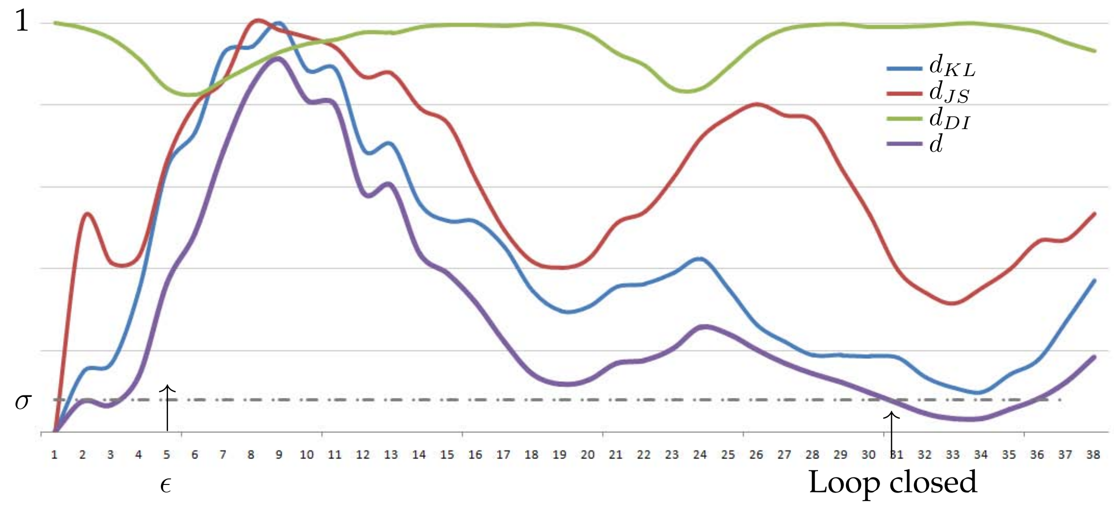


Figure 9.12: *Close loop while learning new object.* Graph showing the evolution of the three parameters d_{KL} (blue), d_{JS} (red), d_{DI} (green) and the result distance d of the complete conjunction .

models. With this feature in mind, MANFRED-2 is ready for interacting with objects in its nearby. The following part of this thesis presents the first perception capabilities of the robot in the laboratory. The context of an experiment will be explained that integrates the work of several researches in order to make MANFRED-2 capable of confirming the existence of a supporting plane, understanding which objects confined in that area are found and finally verifying which of those objects the robot is familiar with. Then, the robot will transfer this knowledge to a path planning module and execute the best solution.

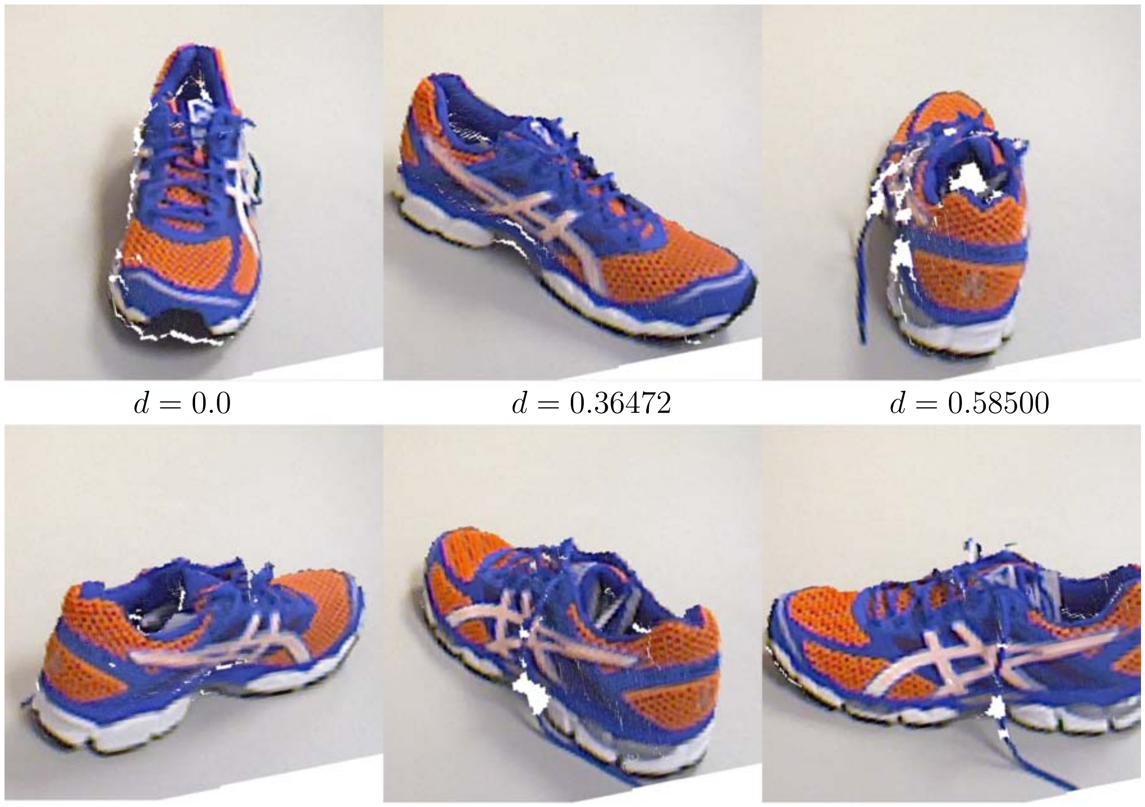


Figure 9.13: *Snapshots of the unknown object during the learning phase.* Representation of several views during the learning process. The number associated to each view is the total distance d for this particular view.

PART IV

APPLICATION

Chapter 10

Affordance experiment

The last chapter of this dissertation aims to integrate in MANFRED-2 manipulator the contributions explained during the previous parts. An experiment that faces the object grasping challenge from a global perspective has been designed, completing the previous integrations of perception challenges in HOAP-3 humanoid made by the author in Bueno *et al.* [211], [212] and Fierro *et al.* [213]. Although the robot is still not qualified to grasp objects, all the previous steps will be executed. That means that the whole perception platform has been integrated in order to make MANFRED-2 ready to grasp familiar objects lying on a table. The platform contains different modules: supporting plane detection, point cloud clustering, features extractor and finally 6-DOF estimator.

However, the experiment goes a step further by integrating the path planner developed by Álvarez *et al.* [214], Gómez *et al.* [215] and Arismendi *et al.* [216] in the same process chain. For the desired object to be grasped, a two-step path planning is presented: during the first phase the anthropomorphic hand reaches the surroundings of the object with no collision and then the 5 fingers perform a precision grasp[17].

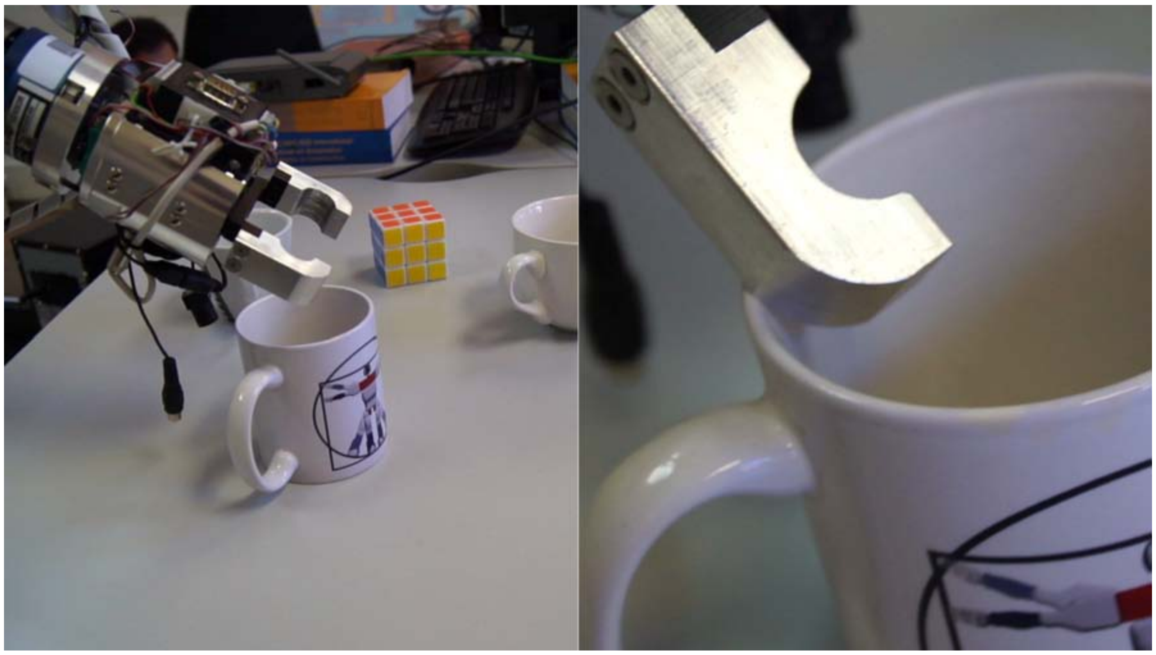


Figure 10.1: *Images of the global experiment.* On the left a wide and close-up of a scene containing a mug and the tool center point

[17]This work has been developed with the invaluable collaboration of David Álvarez

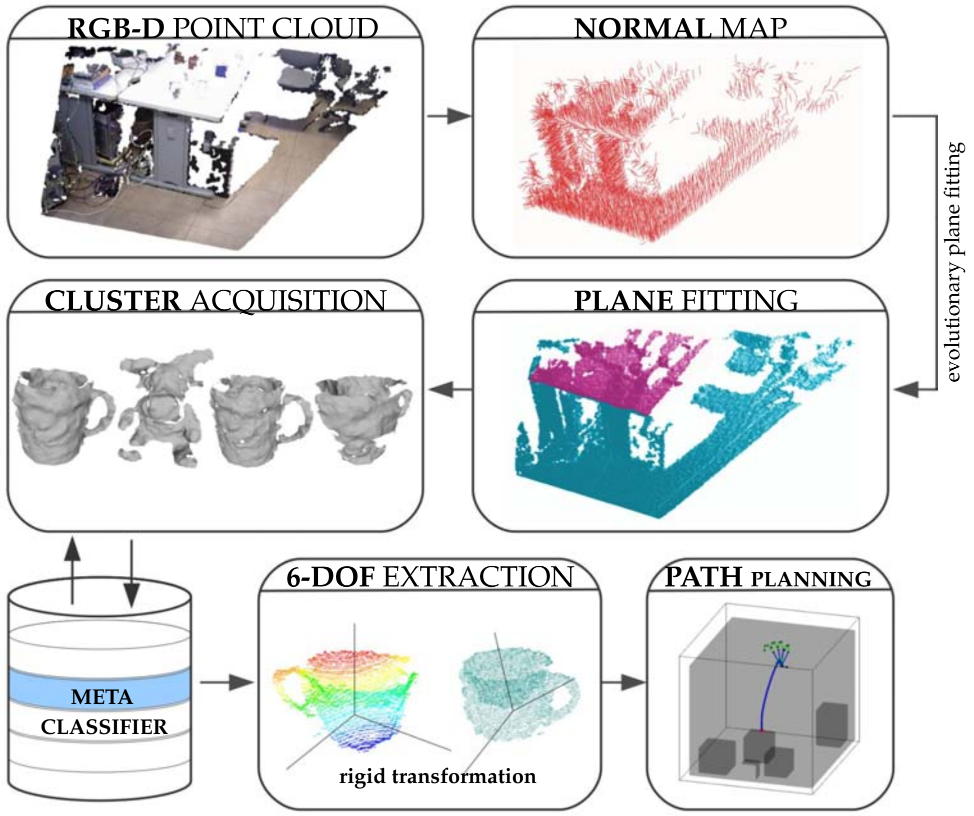


Figure 10.2: *Experiment course.* Pipeline with the main steps followed by the proposed framework to recognize and pick up an object with obstacle avoidance and safe path planning.

10.1 Proposed Architecture

The work proposed here defines a continuous grasping pipeline covering from perception to grasp planning, including visual object recognition for confusable objects. For that purpose, a household environment with several objects is presented in front of the robot. Items are recognized from a database and if one is chosen, the robot will calculate how to grasp it taking into account the kinematic restrictions associated to the anthropomorphic hand and the 3D model for this particular object. Figure 10.2 considers the complete pipeline and the information shared among modules on each part of the process.

For this study, a RGB-D Asus Xtion Pro Live fixed camera has been installed in the upper part of the robot. The role of the camera is to acquire a valid model of the complete environment and substract all the possible objects that could interfere during the path planning for the grasping. Camera output is VGA (640×480 pixels) and output video frame rate may vary between 25 and 30 Hz. Camera flow acquisition is programmed using OpenNI library [217], while cloud operations are done using Point Cloud Library [57] with *ad-hoc* interface for the visualization. All the architecture follows a modular structure to facilitate the addition of new filters or improvements.

The following section will provide a theoretical backup of the experiment. Since safe path planning using Fast Marching is not within the list of objectives of this thesis, part of the demonstrations and explanatory details will be omitted. One of the aims of this experiment is to introduce the grasping problem to the robot. Since it is a preliminary work, several aspects have been assumed. For instance, the location of grasping points for the object is one of the aspects in the rough. Being aware of the enormous state of art about object grasping as in Sahbani *et al.* [218], the experiment here presented only provides with the very first guidelines to make MANFRED-2 qualified for object grasping.

10.2 Fast Marching and Path Planning

In the framework presented in this paper, the Fast Marching Method (FMM) has been chosen as path planner. Fast Marching is a computational algorithm to solve the arrival time of a expanding wave in every point of the space. Conceptually, it can be considered as a continuous version of the Dijkstra's algorithm [219].

The FM2 method arises from the application of the FMM twice over the same map. The first time it is used to create a map of velocities of the environment and the second time computes the time of arrival of the wave for every point. The velocity at which the wave moves is the one computed in the previous step. The FM2 method is very versatile when applied to motion planning problems.

10.2.1 Fast Marching Method

The FMM was proposed by Sethian [220] to approximate the solution of the Eikonal equation [221]. Let us assume a 2D map, where $\mathbf{x} = (x, y)$ is a point on the map with the coordinates in relation to a Cartesian referential, $T(\mathbf{x})$ is the front-wave arrival time function and $F(\mathbf{x})$ is the velocity of the wave propagation.

Let us assume that a wave starts propagating at time $T = 0$ with velocity F, always non-negative. The Eikonal equation (10.1) defines the time of arrival of the front-wave, $T(\mathbf{x})$, at each point $\mathbf{x}$, in which the propagation speed depends on the point, $F(\mathbf{x})$, according to

$$|\nabla T(\mathbf{x})|F(\mathbf{x}) = 1 \tag{10.1}$$

Discretizing the gradient ∇T according to Osher *et al.* [222], it is possible to solve the Eikonal equation at each point $p(x_i, y_j)$, where i and j are the row and column of a grid map, as follows:

$$\begin{aligned} T_1 &= min(T_{i-1,j}, T_{i+1,j}) \ . \\ T_2 &= min(T_{i,j-1}, T_{i,j+1}) \ . \end{aligned} \tag{10.2}$$

$$\left(\frac{T_{i,j} - T_1}{\triangle x}\right)^2 + \left(\frac{T_{i,j} - T_2}{\triangle y}\right)^2 = \frac{1}{F_{i,j}^2} \ . \tag{10.3}$$

The Fast Marching method consists on solving $T_{i,j}$ for every point of the map, starting at the source point of the wave where $T_{i_0,j_0} = 0$. The following iterations solve the value $T(i,j)$ for the neighbors of the points solved in the previous one. Using as an input a binary grid map, the output of the algorithm is a map of distances to obstacles as shown in Figure 10.3. These distances correspond to the time of arrival of the expanding wave at every point of the map. By applying gradient descent from any point of the map of distances, a path will be obtained with the source of the wave as a goal point, this way FMM can be directly used as a path planner algorithm. This is valid only if one wave has been employed to generate the map of distances; otherwise, local minima will appear. The main advantage of this method is that the path obtained is optimal in distance, as the example of Figure 10.3.

10.2.2 Fast Marching Square Method

Although the paths provided by the FMM are optimal in distance terms, they do not accomplish the smoothness and safety constraints that most of robotic applications require, since they run too close to obstacles and have sharp curves. In view of these drawbacks, the FMM algorithm is not a good enough solution in most cases. However, the FM2 algorithm [223] solves these two main disadvantages.

It is based on creating a map of velocities in which the velocity of the expanding wave varies depending on the distance to the closest obstacle. The FMM can be also applied in order to obtain this map of velocities. In this case, all the occupied

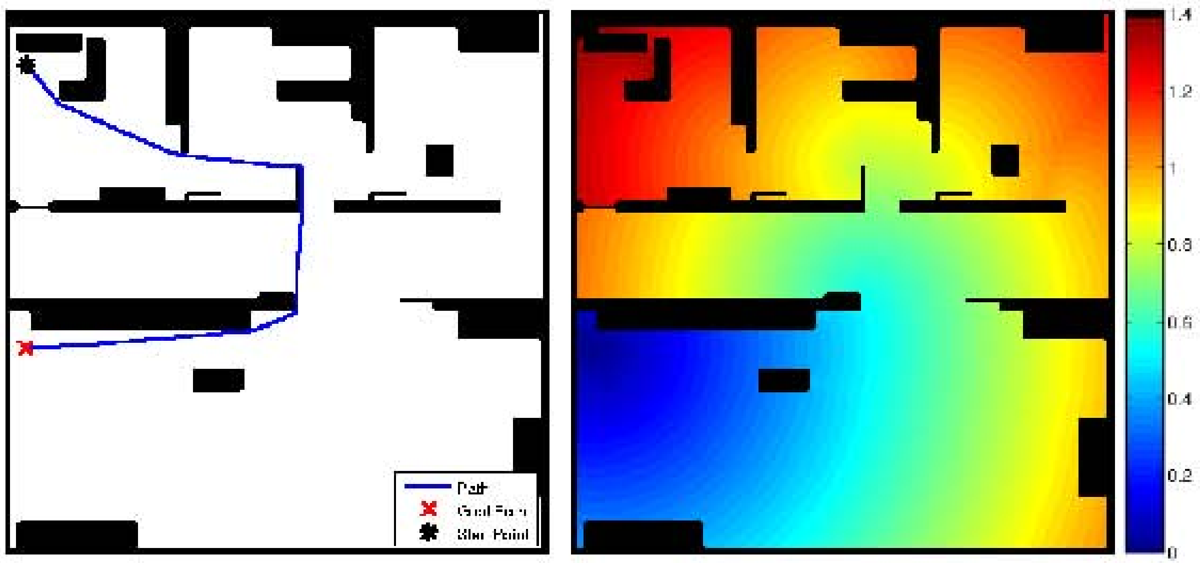


Figure 10.3: *Example of a path obtained with the FMM*. The left side shows the original map and the path calculated. In the right there is the map of distances computed with FMM.

positions in the grid are labeled as wave sources. The result is a map of distances in which those cells in the grid that are farther from the obstacles have a higher value (Figure 10.4 up)).

Once the map of distances is computed, it is normalized and interpreted as relative wave expansion speeds, so that 0 and 1 mean null and full speed of the wave expansion, respectively. Then, the FMM is applied once again with the goal point as wave source. During the expansion, the wave will propagate with the velocities indicated in the map generated previously. The propagation ends once the initial point of the path is reached. When gradient descent is applied to the resulting map of distances, a very smooth path is obtained, as shown in Figure 10.4. In summary, FM2 applies the FMM twice without any mathematical modification: the first step creates a map of velocities, $F(\mathbf{x})$, and the second step computes the time of arrival function, $T(\mathbf{x})$, in which gradient descent is applied to find the path.

In addition to the smoothness and safety, FM2 has other properties worth to mention:

- *No local minima*.- As long as only one wave is employed to generate the map of distances, FMM ensures that there is a single global minimum at the source point of the wave (goal of the path).

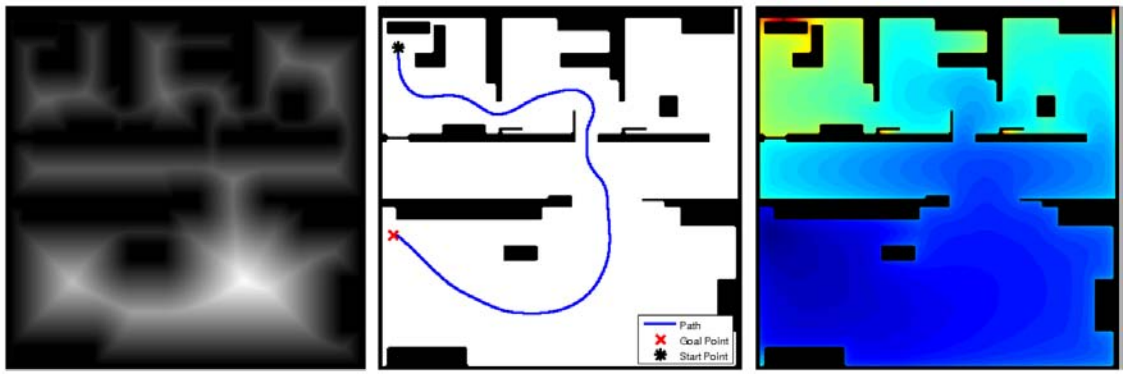


Figure 10.4: *Fast Marching algorithm.* Computation of the distance transform in a 2D scenario after applying FMM over the map of velocities. In the center the result of the FMM. On the right the front-wave representation.

- *Completeness.*- The method finds a path, if it exists, and notifies it in case of no feasible path.
- *Fast response.*- If the environment is static, the map of velocities is calculated only once. Since the FMM can be implemented with a complexity order of $O(n)$ [224], building the map of velocities is a fast process.

10.2.3 3-Dimensional Fast Marching Square

Since the FM2 algorithm is based on the standard FMM, it is extensible to more than 2 dimensions. Due to the fact that a grasping task is carried out in a 3 dimensional space, 3D FM2 algorithm is applied. The algorithm is exactly the same as explained before but, in this case, the front wave becomes a spatial curve.

All the properties of the FM2, such as smoothness or safeness, remain in a $N-$dimensional environment. This is the main fact that leads researchers to use this algorithm as path planner. Figure 10.5 represents an example of a path obtained in 3D for a given environment using FM2.

10.2.4 Geometry of the Hand as a Robot Formation

For the purpose of this experiment, a hand is modeled as a kinematic chain so that simulations can be performed. The model used is based on the kinematic characteristics of Gifu Hand III, which is shown in Figure 10.6. This model has been chosen because it is a five-finger hand and is very similar to a human one. In the structure, the lengths of the links between different joints (palm, finger separation

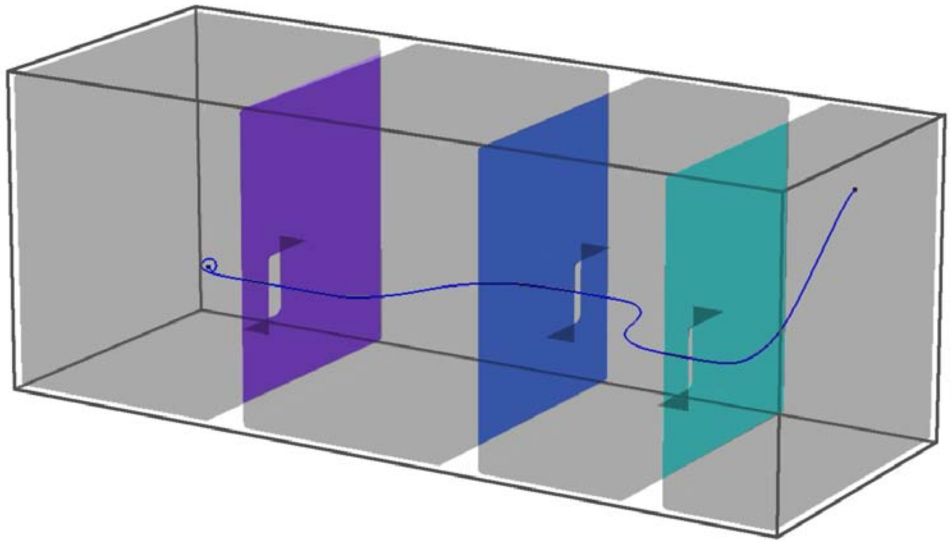


Figure 10.5: *Example of a 3D path result of the 3D FM2 algorithm*. The path is crossed three times passing through small wholes. The path must be considered smooth and safe.

and proximal/inter/distal phalanges) are constant, while the angles between these links (α, β and γ) are variable. These angles are limited by software following the specifications of the Gifu Hand III [35].

It is important to notice that, as in a human hand, the fourth joint of each finger engages with the third one linearly, as Kawasaki proposed in [225]. The same idea is applied for the thumb (although in the Gifu Hand III these joints in the thumb are independent). In order to consider the hand as a robot formation, mobile robots are located at the position of the joints and the fingertips of the hand. The hand is considered as a leader-followers formation in which the followers will change their location while performing the grasping process depending on their position in the environment. For each phase of the algorithm, the configuration of the formation is changed so that the different objectives can be achieved.

10.2.5 Robot Formation Control with Fast Marching Square

In order to perform a precision grasp the problem is divided into two phases. In the first one, a path towards the object to be grasped is computed and covered until a position from where the grasping points can be reached. In the second one, a path for each fingertip towards the corresponding grasping point is computed

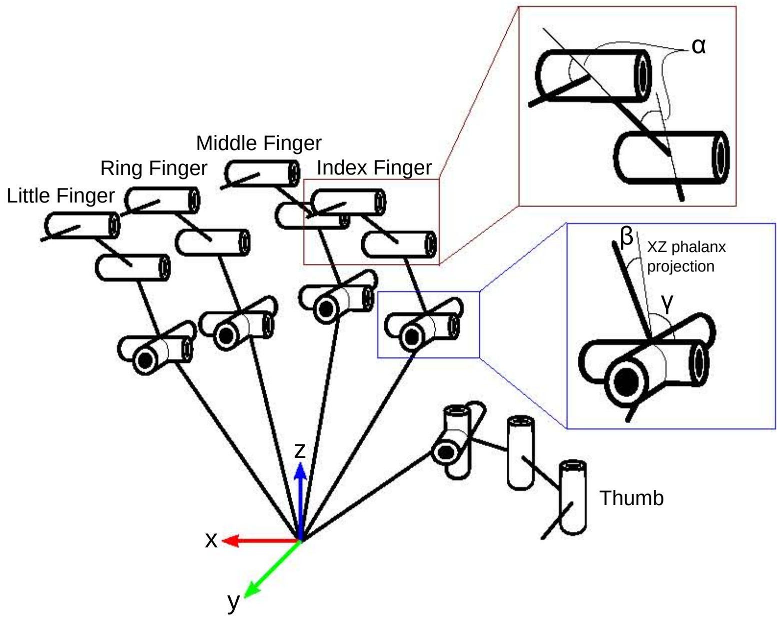


Figure 10.6: *Gifu Hand III*. Kinematic model of the Gifu Hand III.

and covered until contact is detected. In both phases the concept of robot formation control based on FM2 explained in Garrido *et al.* [226], Gómez *et al.* [227] is applied in order control the configuration of the hand in the different stages. By applying this concept, a reduction on the complexity of the problem is reached, since the configuration of a complex kinematic chain with a high number of degrees of freedom, like the hand presented before, is controlled using the information computed for the FM2 path planning algorithm, as presented by Álvarez *et al.* in [214].

While the performing the path, the configuration of the hand changes to make the next phase easier. For this purpose the hand is first opened to ensure that the grasping can be done and later the configuration closes slowly so that the grasping process is shorter. In the second phase, the given precision grasp points must be reached by the fingertips of the hand. In this case a path for every fingertip must be calculated. These paths are covered just moving the fingers of the hand and the palm remains in the same pose. In this experiment an algorithm for each of these phases is presented, both of them based on FM2 path planning method and its application to the control of a robot formation in a 3D environment.

In both phases leader-followers formations are created to apply this concept. In the first part of the algorithm the purpose is to get the hand close to the object that is going to be grasped. In this phase, a leader for the formation is placed in the center of the palm of the hand. While covering the path, the configuration of the hand evolves in order to ensure the grasping of the object. As an addition to the algorithm in [214], the orientation of the hand is also controlled.

For this purpose, the concept of the Frenet frame applied to the analysis of curves in Euclidean space is used. In a three-dimensional space, this analysis provides a coordinate system at each point of the path composed of the normal, tangent of curvature and bi-normal vectors [228]. Since the framework presented in this work applies to on the table scenes, most grasping configurations are limited to overhead grasps. When an overhead grasping is going to be performed, the normal vector of the palm of the hand has the opposite direction than the normal vector of table. At the same time, the approximation movement of the hand to the object is done above the table, which leads to a path in which the tangent vector is opposite to the normal vector of the table. Therefore, the leader of the formation follows the orientation of the tangent vector at every point of the path in order to have the right orientation to execute the grasping.

10.3 Experimental Results

In the following section the results of the complete architecture will be presented. The experiment was performed in the labs of Robotics Lab in Madrid. MANFRED-2 was positioned inside an unexplored scenario (see Figure 10.7). The robot was primaryly asked to move forward until a supporting plane was found within a distance below 60 cm. to the 3D sensor location in the $z-$axis. The robot was equipped with a navigation module that ensures that there is no collision of the robot with the environment while moving the base. This is done by tracking the odometry and running a SLAM algorithm in the background. In case the robot found an obstacle during the base movement, it would try to keep away from it by finding other alternatives to move in forward direction.

The speed of the wheels is adjusted dynamically by the PMAC but maximum speed and maximum torque factors can be set before the path is executed. The 3D sensing device begins the evolutionary plane fitting algorithm from the beginning. As it would find several planes in random directions, including the walls and the floor, an extra condition has been introduced in the perception module to control the false positives. In this case, a distance condition has been applied in such a way that any plane whose distance to the origin (3D sensor) overpasses 50cm will be rejected. In this way, most of the plane candidates get rejected straightforwardly. In addition to these conditions, the geometric and outlier filtering explained in 4.3

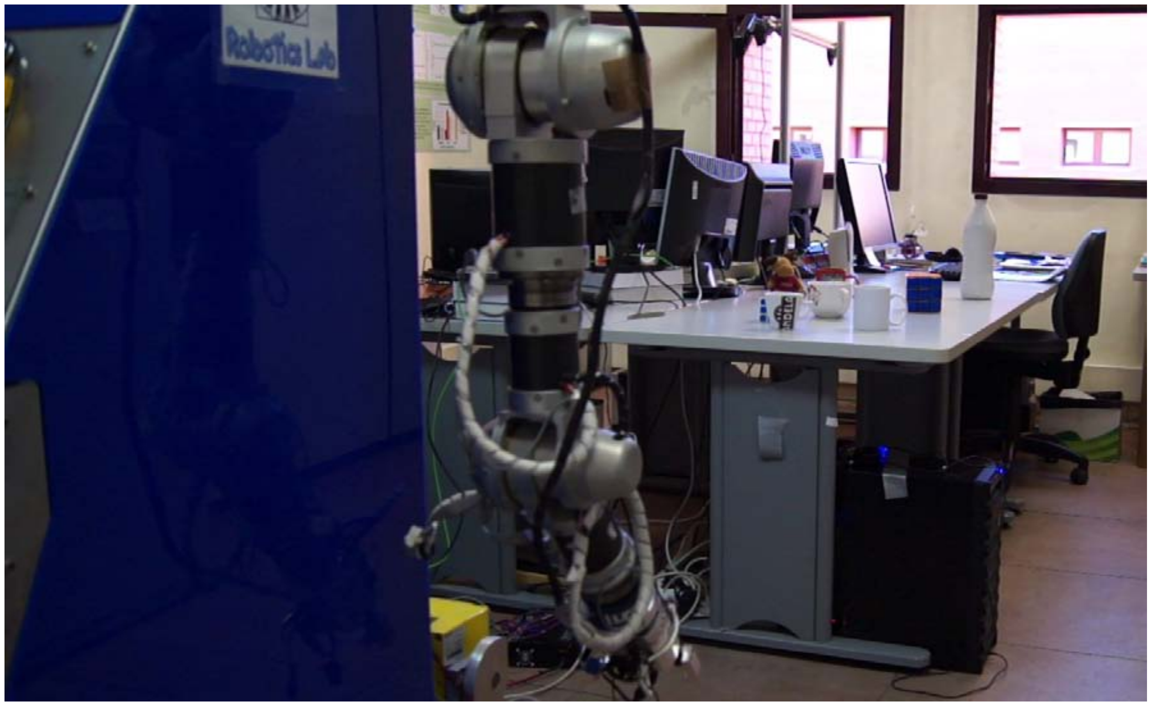


Figure 10.7: *Images of the global experiment.* The robot was positioned in an unexplored scenario and was asked to move forward until a supporting plane was found within a distance below 60 cm.

is still applied. This state is maintained until a horizontal plane is correctly fitted.

As soon as the robot locates a valid supporting plane, it will stop immediately. In that moment, the plane fitting process will halt and the Euclidean Clustering algorithm will start analyzing the observed 3D point cloud as shown in Figure 10.8. A maximum cluster tolerance of 2 cm has been establish for this experiment. Each cluster is then represented in a different color. As a default rule, the robot will show interest in the object that is located in the rightmost. In this case, the object is `mug-robotics-white`.

Once the bounding box is processed for each clusters (and obstacles) lying on the table, the classification algorithm will start up. The first stage consists on extracting the three features required by the meta-classifier to operate: the VFH descriptor of the observed surface, the boundary dimensions of the object and finally the HSV histogram. This information is stored and normalized as needed (HSV

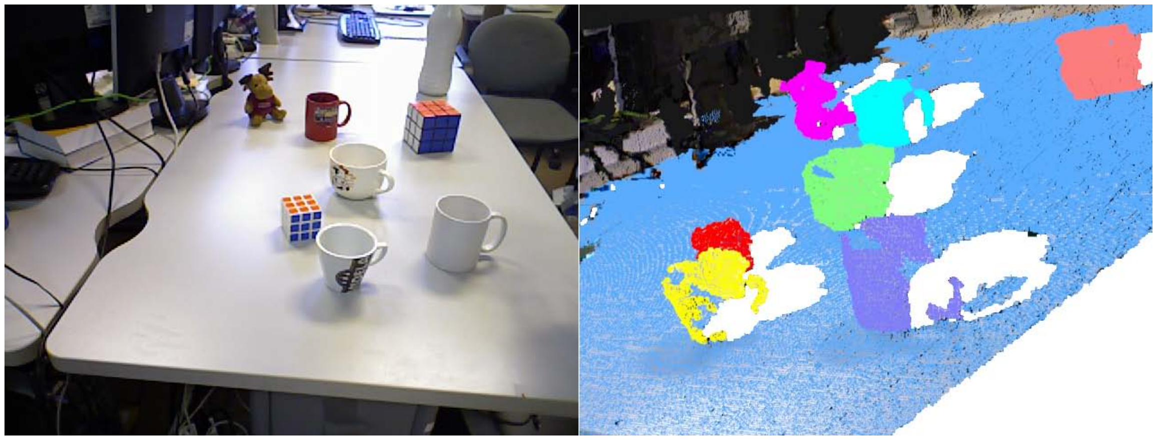


Figure 10.8: *Result of the Euclidean Clustering algorithm.* After the extraction of the supporting plane. Each object is marked with a different colour. For this project, red mug will be chosen as grasping target.

histogram and VFH descriptors are normalized as they will be classified by using a divergence metric). The size of the database is the same as the previous experiments: 300 individual point cloud views formed by 15 different objects. The selected target (see Figure 10.9) is rendered and meshed using Poisson reconstruction to be processed by the upcoming module: the path planner.

Rough approximation

After the clusters are labeled as target or obstacles, a voxelization process is performed before the path planner is executed. Within this process, each pixel in the space will turn into one of the three possible states: occupied, unoccupied and target. As mentioned before, the last objective of MANFRED-2 is to grasp objects with the Gifu Hand III anthropomorphic hand. Since it has not been installed yet in the robotic arm, path planner will choose the barycenter of the target as the final destination of the arm. Then, once the arm reaches the object barycenter, a simulation of the finger affordance using a robot formation in a 3D environment will be performed.

Since the first step of the path planning is rough and the target is simply to approach the robot end tool to the target's barycenter, each voxel of the workspace is defined with a 3D space resolution of 1 cm^3. The complete workspace is then enclosed in an empty cube of 1 m^3, so it is possible to expand a wave in the 3D

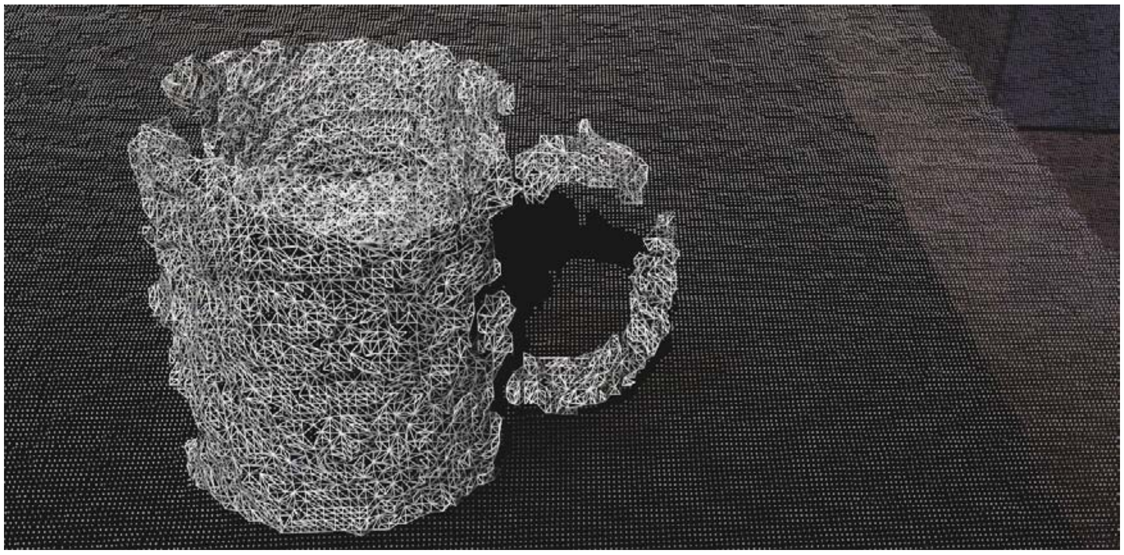


Figure 10.9: *3D Mesh.* Render and mesh generation using Poisson reconstruction for a single point cloud view.

space. Figure 10.10 represents the path planning calculation stage. All the bounding boxes of the objects in the environment are labeled as explained and represented as dark tetrahedrons lying on the supporting plane. The initial position of the hand has been already fixed. As the path planning is computed from the hand position, a 3D transformation is executed between the 3D sensor device and the hand.

The frames shown in Figure 10.10 represent consecutive stages of the path planner algorithm. The real movements of the robotics arm are on the right and central column while on the left side the computed path planning estimation is displayed. Furthermore, the path for the approach stage is represented in blue color and it turns progressively into red as soon as the hand reaches it.

The center of the hand palm is set as the leader of the formation for covering the complete path. Note that the 16-DOF of the hand are managed with the same path planning strategy, so the computation complexity is highly reduced for the experiment.

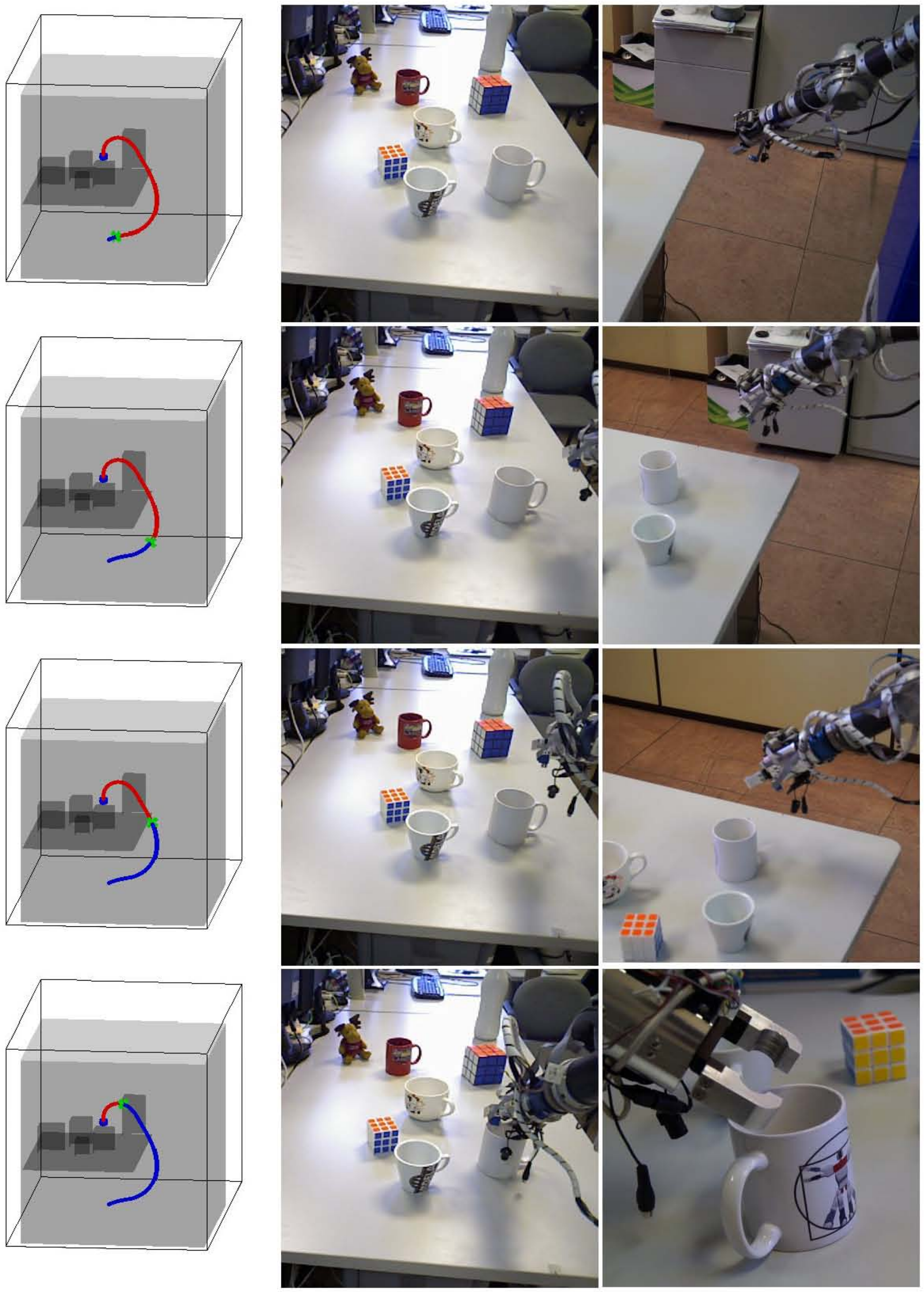


Figure 10.10: *Rough hand approximation.* Representation of the voxelized environment with the bounding boxes of each obstacle. Left column shows the path planning and the hand during the approaching phase. Red path means portion of the path covered by the hand up to now. Center and right columns show the experiment from the robot point of view and from an external observer.

Fine approximation

Once the hand is positioned close enough so that the grasping points are inside the finger's workspace, the second path planning phase is started. Figure 10.11 represents the path for each finger in red to achieve the grasping point as it is in the database. For the second path planning phase, the resolution is increased 64 times the previous one so each voxel has a 3D space resolution of 2.5 mm^3. The execution time for both path calculations are presented in Table 10.1. It contains the time spent by FM2 for the fingers at different resolutions.

FINE PATH PLANNING	
Resolution	time [s]
1 cm^3	0.419789
	0.431052
	0.418900
	0.408478
	0.443023
64 cm^3	5.628340
	5.618100
	5.550387
	5.568807
	5.561760

Table 10.1: Time intervals for the 5 fingers path planning at different voxel resolutions. As long as resolution is increased, the elapsed time to compute the fine path planning increases exponentially.

As it is stated, the elapsed time of the complete pipeline increases exponentially as long as the resolution is increased during the fine approximation. For the fingers path planning the high resolution alternative will be selected in order to obtain a good geometrical representation of the target object. This will lead to a better performance for the precision grasp. Table 10.2 contains the time intervals consumed by each part of the pipeline, including perception steps and path planning.

Analyzing the previous results, the most expendable time consuming is made by the fingers path planning of the grasping step. This is not directly caused by FM2 but by of the high resolution required during the finger grasping. For a better understanding of the algorithm an example of its performance has been published online [18].

[18]Visit `http://www.youtube.com/watch?v=GPWQh6pb6u0`

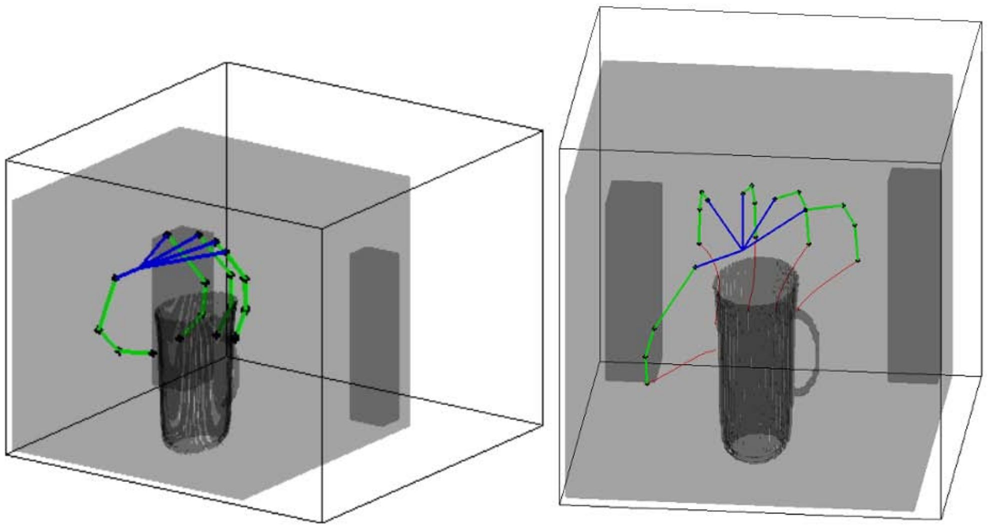


Figure 10.11: *Fine fingers approximation.* Evolution of the finger's path planning moving to the grasping point positions for the target object.

Table 10.2: Time intervals for each part of the pipeline

FINE PATH PLANNING	
Pipeline stage	elapsed time [ms]
Normal Estimation	24
Supporting Plane	67
Euclidean Clustering	52
FPFH Extraction	327
Matching + Pose est.	46
Velocities map approximation	42
Path planning approximation	1272
Velocities map fingers	116
Path planning fingers	27925
total execution time	**29871**

10.3.1 Finger Simulation

To conclude the experiment, a final simulation of the fine grasping has been performed using Graspit!, a software developed by Department of Computer Science in Columbia University, New York. [229]. This tool is quite common in grasping research presented by Miller *et al.* in [230]. It has been used in several researches such as Rusu *et al.* in [231] where it was applied to model an object as a set of shape primitives, such as spheres, cylinders, cones and boxes; It has been updated since its first release in 2003 [232]. Figure 10.12 represents the affordance planning of the GIFU Hand model grasping the reconstructed model of the target object `mug-robotics-white`.

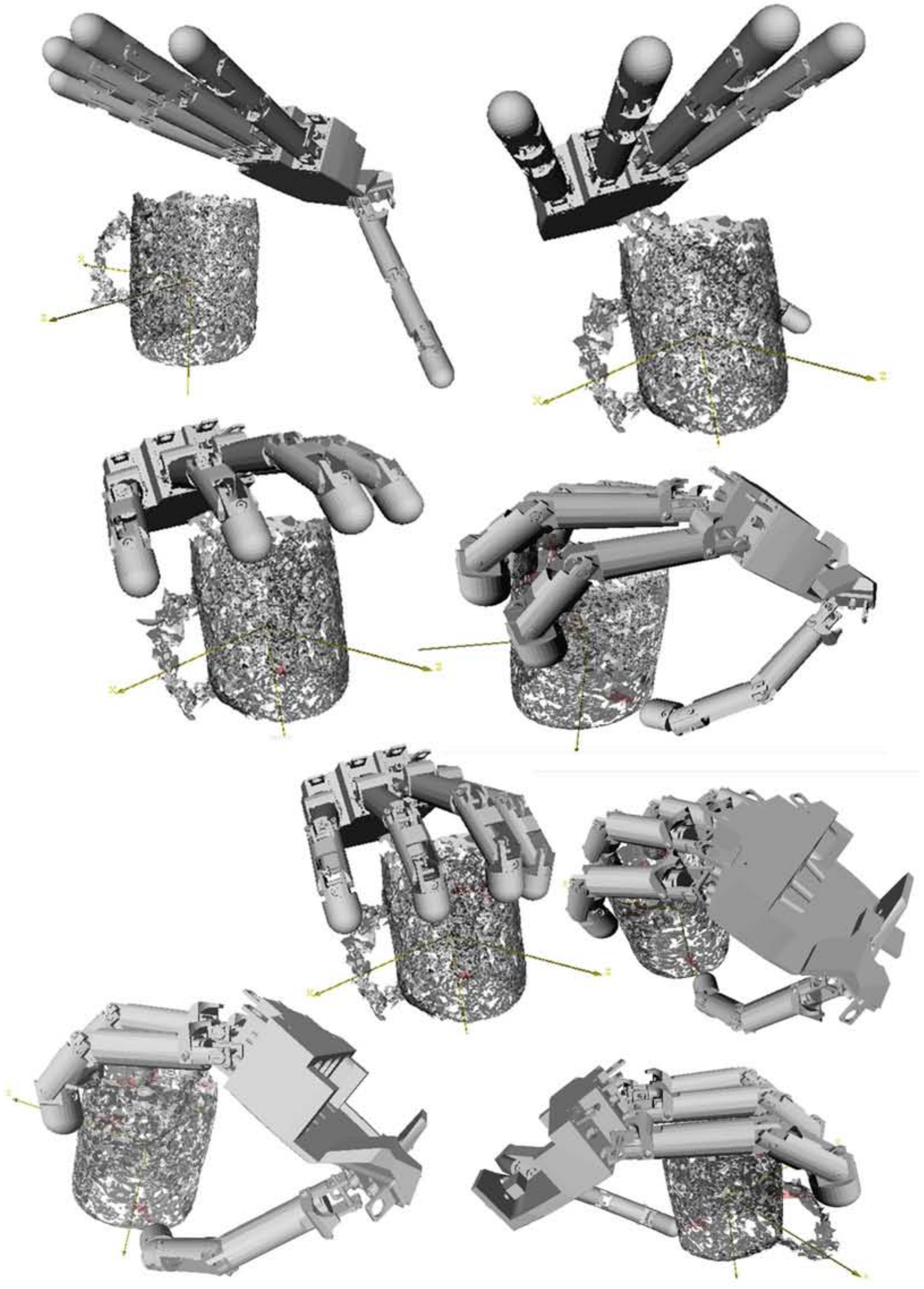


Figure 10.12: *Fine affordance for the 5 fingers of the anthropomorphic hand.* Different perspectives of the affordance simulation using Graspit! simulator.

Chapter 11

Conclusions and future developments

Three different contributions related to the 3D perception challenges in mobile manipulators have been addressed in this dissertation. These algorithms solve the following difficulties: plane fitting, object reconstruction and object recognition. The foremost highlight conclusions and the particular results of each part are described in this chapter. Finally, a list of future developments and extension lines will be provided for a short and long term researches.

11.1 Supporting Plane Fitting

A novel plane fitting algorithm for 3D scenarios based on evolutionary algorithms has been presented in this document. It has been designed to segment and detect supporting planes in unexplored scenarios yielding the four parameters that constitute the general equation of that particular plane. It is based on DE, which is a particle-based evolutionary algorithm that evolves with time to the solution that returns the lowest cost function value. The proficiency of this method to strongly accomplish the expected task is corroborated and the comparison with the state of the art algorithm RANSAC is also considered.

As it has been exposed by the results obtained in the experiments, DE offers a better performance execution with a small average error nonetheless with a higher variation degree when it is compared with RANSAC iterative method. However, the most interesting results obtained during these researches arise when the observed plane is contaminated with multiple objects. The enlargement of noise in the measurements entails with a lack of information about the inliers, preventing the fitting process to yield the same accuracy. In these terms, it has been demonstrated that DE fitting algorithm proposed in this thesis becomes more robust than RANSAC for cluttered scenarios. Since the plane fitting model proposed grants a reasonable estimation error, it not necessary to introduce a tracking component in order to follow the plane during the time.

As the final intention of this perception challenge is to provide the manipulator robot with grasping dexterity, the clustering output based on the proposed plane equations for both estimators has been discussed. Point cloud clusters populated from the evolutionary plane fitting solution presented richer information at the base of the clusters.

11.2 Evolutionary Model Reconstructor

A DE-based model reconstructor has been implemented for 3D objects. Taking advance of the optimization vantage of evolutionary algorithms, a 6 DOF scan matching process has been carried out satisfying six requirements: robustness against abrupt rotations, multi-dimensional optimization, feature extensibility, compatible with other scan matching techniques, management of uncertain information and an initialization process.

The reconstructor presented in this thesis has been designed to deal with 3D point clouds and the local accuracy is suitable enough to be used in manipulation tasks. The reconstructor is divided into four stages: while the first phase extracts relevant information from the point cloud, such as 3D descriptors and normal maps, the second stage makes use of a novel probabilistic curvature extension for hole filling. This extension has been proved to increase the correspondence level during the registration procedure. The third step is a yaw initialization process that has been demonstrated to increase the convergence speed of the scan matching optimizing the initial transformation of the point cloud.

The last contribution of this evolutionary model reconstructor is focused on the design of the cost function of the DE optimizer for the registration procedure. It has been detached into four interconnected errors that cover the local vicinity error, a global distance error that measures the transformation from a collection point of view, a normal alignment error that improves the stability of the algorithm, since it represents the principal direction of the surfaces, and lastly an intensity correlation error that includes information of the color and texture of the cluster.

11.3 3D Object Meta-classifier

An object classifier that measures the similarity of a single point cloud with respect to a database of point cloud views has been developed. The suggested classifier has to be able to differentiate successfully between 3D point clouds with a high degree of similarity (confusable objects), extracting parameters such as 3D features from the surface of the cluster, absolute geometrical boundaries and also color distributions.

The classifier has been designed to detect and populate the most similar object from the database in three steps. Each step represents a filter with a very precise commitment. The first one is focused on the surface shape of the object, therefore it will select from the database a narrow list of candidates whose surface form is similar to the query. However, since this filter is based on VFH descriptors, which are scale invariant, candidate objects may have similar shape but different size. Therefore, the second stage of the classifier is a dimensional filter that will compare the candidates dimensions with the query ones. At this stage, the classifier will return objects with similar surface shape and also with similar dimensional measures. However, confusable objects may have similar size and shape but dissimilar textures. For this reason, the last filter finally segments the most probable object keeping in mind also the color.

In order to compare between different descriptors, a distance metric comparison has been carried out. Kullback-Leibler divergence has been demonstrated to be an efficient metric, while VFH descriptor matching and Jensen Shannon divergence yield good performance while comparing HSV histograms for color correspondence. Another conclusion has been extracted from the study of the classifier performance with respect to the object depth. As the cluster size is reduced, the information is lessened and the global performance is affected.

A deep study of the object recognition rates has been developed with a representation of the confusion matrices. The classifier has been analyzed gradually with the addition of the already mentioned filters. It has been demonstrated that VFH-based filters are not accurate enough to distinguish between confusable objects. The introduction of a dimensional filter and a color filter boost the global performance in conjunction with the metrics obtained in the earlier paragraph.

Finally, with the aim of making the classifier as autonomous and dynamic as possible, an automatic learning program has been introduced that allows the robot to acquire new models on the fly. For this purpose, a cost function that compares iteratively observations has been designed. The algorithm has been prepared to discard redundant information and it detects a close loop when the object is sufficiently observed. The reliably and robustness of the system is enough to make the robot push the detected object with no human intervention.

The last part of the thesis gathers a complete perception architecture interconnected with a path planner that has been implemented in the mobile manipulator platform. A real scenario has been considered, where the robot was asked to find a table in the scenario. Then the robot detected and segment the different objects lying on the table and recognized them individually. Afterwards one of the objects was commanded to be grasped and the robot computed successfully a safe path planning for the robotic arm based on the coordinates given by the perception module.

11.4 Future Developments

This thesis commends some challenges to be achieved in the near future:

- Improve the plane fitting performance by finding new strategies to reduce time consumption and computational costs during the optimization process, convergence speed and outlier filtering.
- Develop a deeper study of the state of the art iterative processes for parametric model estimators and compare them to the DE-based proposal taking into account internal parameters such as crossover probability, selection method and population initialization.
- Determine the best distance metric for the cost function of the evolutionary plane fitting. Evaluate other cost functions, norms and problem formulation.
- Improve the actual Euclidean cluster extraction in order to segment touching clusters by means of statistical densities, local surface analysis or graph cuts optimization.
- Although the hole filling proposed improves the scan matching, a more efficient alternatives based on statistical patterns might be considered to be introduced.
- An extensive research on the fitness function weights and its adaptivity may be granted. The performance of the matching process and the convergence speed are open to fresh proposals.
- The introduction of machine learning algorithms such as SVM or Bayesian statistics must be examined for the classification problem. Although the proposed meta-classifier yields promising results, the addition of new models makes it necessary to introduce other alternatives.
- The versatility of the classifier allows a further analysis of the distance metrics during the descriptor comparison.
- The rapid release of new 3D sensing devices makes it interesting to introduce new sensors and compare their performance in terms of accuracy, speed and stability.

Appendix A

Differential Evolution

Differential Evolution is a practical approach to global numerical optimization problems that involve constrained functions having many local minima and mixed-type variables. The foremost advance of DE is that *multi-modal* problems are successfully affronted because of its genetic algorithm nature. Differential Evolution was developed in 1995 by Keneth Price [233] and it is assorted as a *metaheuristic* method in the way that it makes few or none assumptions about the problem it is dealing with but as a result it does not certify an optimal solution.

Optimization can be defined as the minimization of the $negative$ parameters (or not desired) of a system at the same time that $positive$ parameters (or desired) are maximized. As a general rule, there exist D parameters in a function $f(x) = f(x_0, x_1, x_2, \ldots, x_{D-1})$ that alters its behavior during its minimization. That means that somehow, in order to minimize f it must be necessary to understand and interpret the rule of each variable within the global system.

A.1 Optimization for Monomodal Functions

Figure A.1 represents a simple quadratic function $f(x, y) = \Omega - e^{-(\psi x^2 + \xi^2)}$. Those functions with a unique minimum are named unimodal or *monomodal*. That is, if for some value $\boldsymbol{m} = (m_x, m_y)$ it is monotonically increasing for $f(x, y) \leq \boldsymbol{m}$ and monotonically decreasing for $f(x, y) \geq \boldsymbol{m}$. In that case, the maximum value of $f(x, y)$ is $f(m_x, m_y)$ and there are no other local maxima.

Monomodal functions contain exclusively a minimum identical to the global solution and therefore the optimal solution, since there exist a unique solution. This value $\boldsymbol{m}$ can be resolved using traditional methods such as the simplest gradient-based method, secant, Gauss-Newton, brute force search or even randomly.

A.1.1 Derivative-Based Optimization

Classic methods used for finding a minimum are based on derivative procedures. A $D-$imensional parameter vector can be defined as

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{D-1} \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & \ldots & x_{D-1} \end{pmatrix}^T \tag{A.1}$$

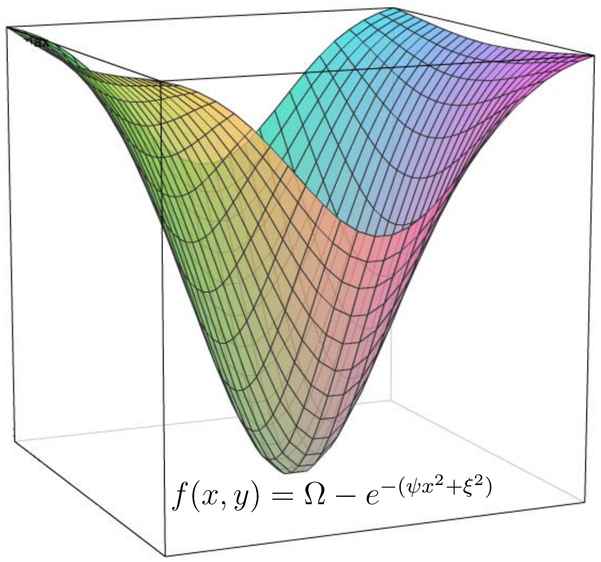


Figure A.1: *Example of a monomodal or unimodal function.* It contains only a local minimum $m(m_x, m_y)$.

Nabla operator is represented as ∇ and is defined as

$$\nabla = \begin{pmatrix} \partial/\partial x_0 \\ \partial/\partial x_1 \\ \vdots \\ \partial/\partial x_{D-1} \end{pmatrix} \tag{A.2}$$

and therefore, gradient vector is defined as

$$g(\boldsymbol{x}) = \nabla \cdot f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_0} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_{D-1}} \end{pmatrix} \tag{A.3}$$

Hessian matrix can be defined as

$$g(\boldsymbol{x}) = \nabla^2 \cdot f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_0 \partial x_0} & \frac{\partial f(\boldsymbol{x})}{\partial x_0 \partial x_1} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_0 \partial x_{D-1}} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_0} & \frac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_1} & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_{D-1} \partial x_0} & \cdots & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_{D-1} \partial x_{D-1}} \end{pmatrix} \tag{A.4}$$

The Taylor series for the objective function turns out to

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \frac{\nabla f(\boldsymbol{x}_0)}{1!}(\boldsymbol{x} - \boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^T \cdot \frac{\nabla^2 f(\boldsymbol{x}_0)}{2!}(\boldsymbol{x} - \boldsymbol{x}_0) + \ldots = \\ f(\boldsymbol{x}_0) + g(\boldsymbol{x}_0) \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^T \cdot \frac{1}{2} g(\boldsymbol{x}_0) \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + \ldots \tag{A.5}$$

being $\boldsymbol{x}_0$ the point around which the function $f(\boldsymbol{x})$ is evaluated. m will be a minimum if and only if all partial derivatives at $\boldsymbol{x} = \boldsymbol{m}$ are equal to zero.

$$g(\boldsymbol{m}) = 0 \tag{A.6}$$

Taking the first three terms of the Taylor expression in Equation A.5 and differentiating them makes the gradient to be declared near the point $\boldsymbol{x}_0$ as

$$\nabla f(\boldsymbol{m}) = g(\boldsymbol{x}_0) + G(\boldsymbol{x}_0) \cdot (\boldsymbol{m} - \boldsymbol{x}_0) = 0 \tag{A.7}$$

that can be resumed as

$$\boldsymbol{m} = -g(\boldsymbol{x}_0) + G^{-1}(\boldsymbol{x}_0) + \boldsymbol{x}_0 \tag{A.8}$$

and the path followed by the optimizer is represented for a second order monomodal equation in Figure A.2, where the starting point is $\boldsymbol{x}_0$ and the end is $\boldsymbol{m}$.

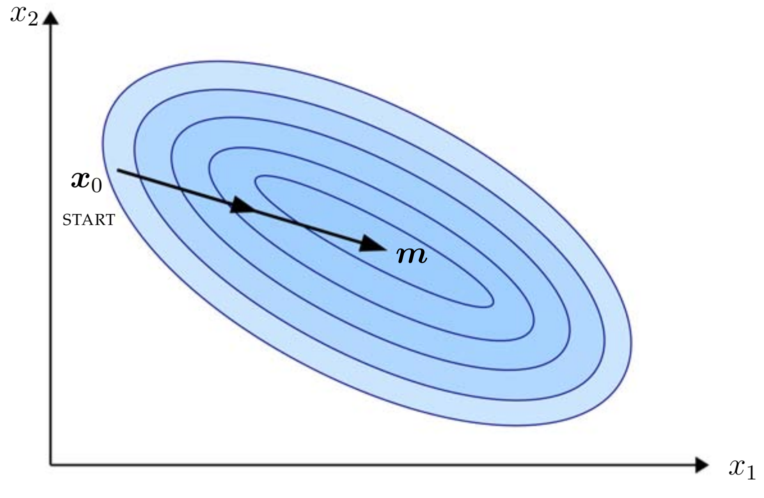


Figure A.2: *Derivative-based optimization*. Representation of the path $\boldsymbol{x}_0 \rightarrow \boldsymbol{m}$ (global minimum) if the objective function is quadratic and differentiable.

The simplest gradient-based method is named *steepest descent*. It turns into a very useful technique for monomodal and differentiable functions. Taking in mind

Equation A.8 and supposing that $G^{-1}(\boldsymbol{x}_0) = I$, where I represents the identity matrix, this turns into

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - g(\boldsymbol{x}_0) \tag{A.9}$$

The gradient will move at any step into a point closer to the minimum as long as the descent is not too large. For this, a step size γ is provided as a measure of control. Therefore, Equation A.9 turns into

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_0 - \gamma \cdot g(\boldsymbol{x}_n) \tag{A.10}$$

Figure A.3 displays an example of this method to find out the minima $\boldsymbol{m}$. As can be noticed, there exist a problem regarding the step size γ that has to be solved depending on each function performance.

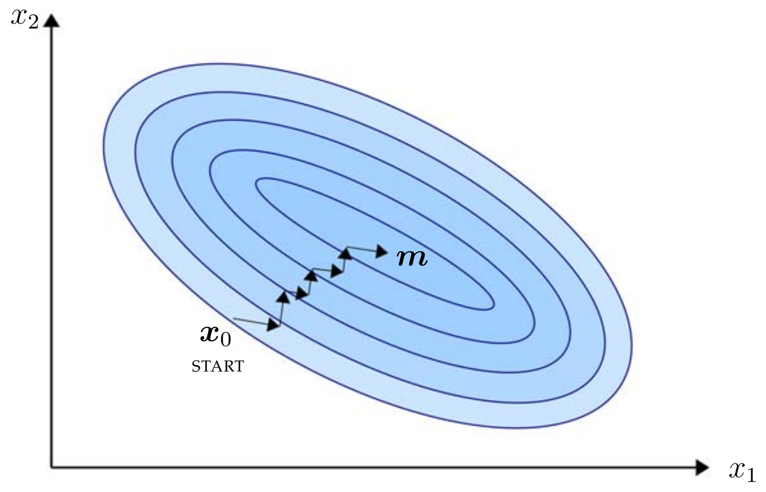


Figure A.3: *Derivative-based optimization*. Representation of the steepest descend gradient path for an objective function that is quadratic and differentiable.

A.1.2 Brute Force Search

Brute search will visit all grid points in a bounded region while keeping the best candidate in memory. Again, as it occurs in steepest descend gradient, there is a problem of resolution or step size. Depending on the step size, the minimum $\boldsymbol{m}$ will be found or a near point will be misled.

Furthermore, a *curse of dimensionality* happen: if the grid becomes too small because a grid with N points in D dimensions will have to evaluate N^P candidates. Figure A.4 contains a grid example to cover the whole space at a fixed step size γ

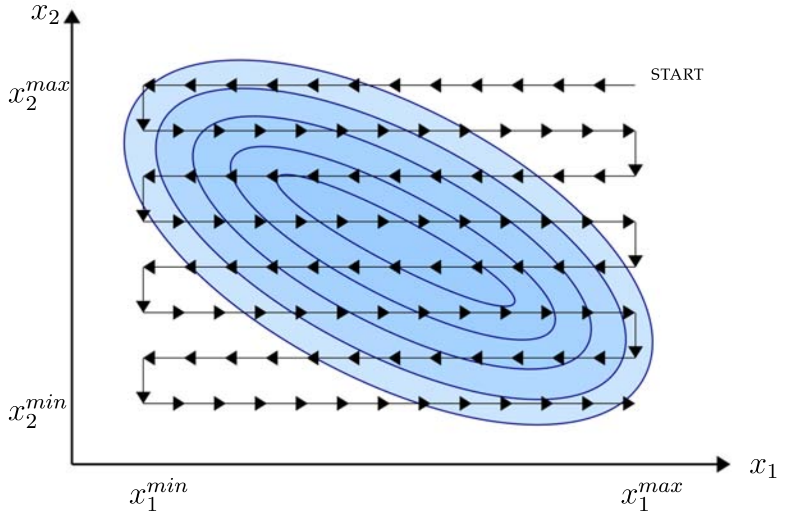


Figure A.4: *Derivative-based optimization*. The lack of knowledge about the objective function makes brute force search useless for most cases.

A.1.3 Random Walk

One chance to avoid the curse of dimensionality implicit in the brute force search is to evaluate candidates stochastically in the objective function. New candidates are created adding a random deviation $\Delta \boldsymbol{x}$ to a given base point $\boldsymbol{x}_0$ using a Gaussian distribution as

$$p(\Delta x_i) = \frac{1}{\sigma_i \cdot \sqrt{2\pi}} e^{-\frac{(\Delta x_i - \mu_i)^2}{2\sigma_i^2}} \tag{A.11}$$

with σ_i and μ_i the corresponding standard deviation and mean value for coordinate i respectively. This method converges to a minimum if the selection of the next candidate is done according to the condition

$$f(\boldsymbol{x}_0 + \Delta \boldsymbol{x}) \leq f(\boldsymbol{x}_0) \tag{A.12}$$

If this is true, $\boldsymbol{x}_0 + \Delta \boldsymbol{x}$ becomes the new base point. Figure A.5 represents an example of the behavior of the random walk over a monomodal function. As in classical and brute force methods, the step size (standard deviations) is in this case a problem that can not be easily solved.

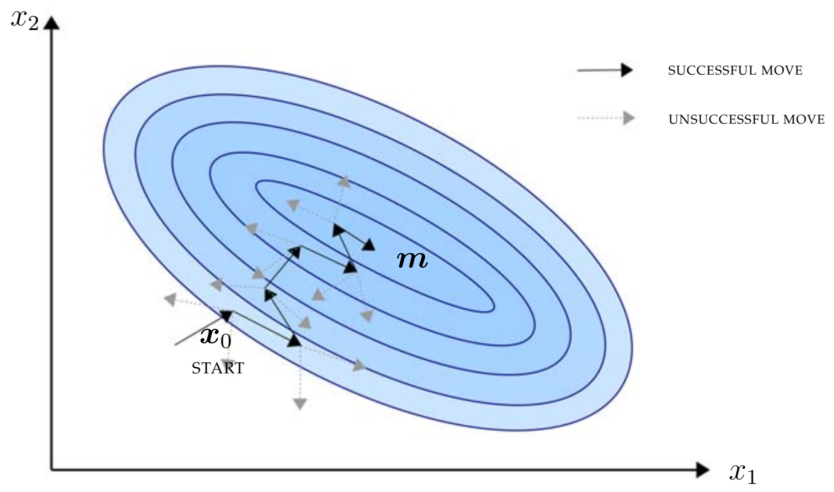


Figure A.5: *Random Walk.* A Gaussian distribution selects the candidates and the one which falls closer to the minimum turns into the next candidate.

There exist other options to face the step size problem as the one proposed by Hooke and Jeeves [234] that tries to create a pattern search exploring each coordinate axis at every iteration, tunning the step size according to the results. If none of the trial points enhance the objective function, the step size is reduced.

This method is much more effective than random walk or brute force but it is still inefficient: if steps are smaller and smaller due to a valley in the objective function, once the valley is overtaken the expansion is extremely ineffective. However, if the optimization process starts near the valley, the results will be adequate.

A.2 Optimization for Multimodal Functions

As a general rule, non-lineal systems bring up problems where the internal mechanisms are completely unknown. In those cases, it is pretty easy to face problems that admit more than one exclusive (local) solution, conceiving a new problem: local minima convergence. Figure A.6 represents the Schwefel's function $f(x,y) = -x \cdot \sin\sqrt{|x|} - y \cdot \sin\sqrt{|y|}$. It is a multimodal function with several minima and only a global solution.

As the reader might notice, if there was an initialization problem for the monomodal

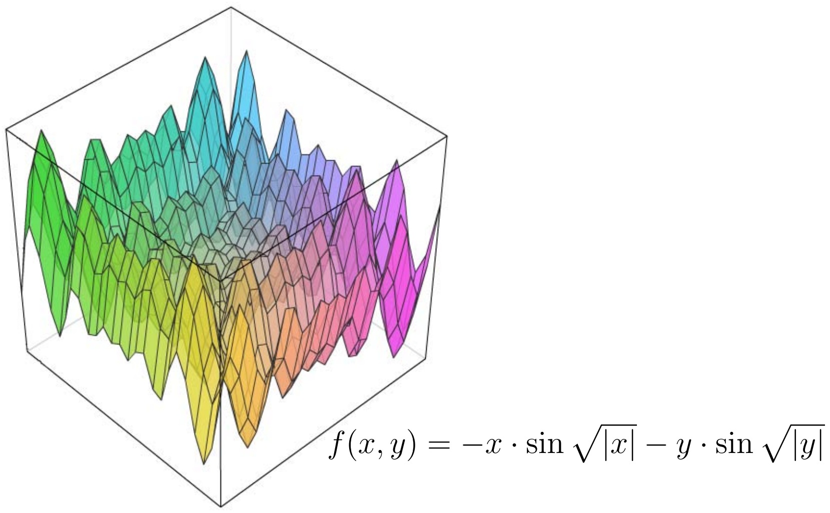


Figure A.6: *Representation of Schwefel's multimodal function*. It contains several local minima and only a global solution.

functions, the problem gets accentuated in a function where there exist local minima. The algorithm performing the search must be ready for getting in and out the valleys while finding the optimal solution. To do this, the optimization functions are evaluated in multiple points, creating global searches, comparing the costs in the whole surface and selecting the most representative (best) candidate for the next iteration. This kind of algorithm requires high computational costs since the solution demands multiple evaluations in order to converge. Some of the typical functions used for benchmarking optimizers are found in [235].

Price *et al.* [233] outline several methods to solve the optimization problem in multimodal functions carried out before genetic algorithms were created. Those solutions are simpler but less potent and less precise:

- *Simulated Annealing.-* A heuristic search is performed evaluating the nearest points on each iteration. Probabilistically, on each iteration it is decided if the algorithm moves to a new state s' or otherwise the system is maintained in the original state s expecting a lower energy direction. This process is performed until the energy is below a certain value established at the beginning of the optimization method.

- *Multi-Point, Derivative-Based methods.-* Several candidates are evaluated in parallel in the objective function. Local search is not necessarily derivative-based and each aspirant might use any direct local search method to evaluate its own path to the solution. The number of points is the principal disadvantage for this method. The quantity of candidates can be estimated using a clustering method.

- *Multi-Point, Derivative-free methods.-* Mimicking the Darwinian evolution, those methods attempt to determine the solution combining and mutating the candidates using the so-called *evolutionary algorithms*. Contrary to multi-start approaches where each member is isolated, evolution strategies are influenced among them and new candidates spring up from previous ones, introducing a new concept of improvement based on a global vision.

A.3 Differential Evolution

DE method is a population-based optimizer that solves the problem of initial point location by means of several stochastic samplings in the objective function. Domain must be preset beforehand with the values x_m^{min}, x_m^{max} and N_p vectors will be generated which are indexed with a number from $0 \rightarrow Np - 1$ (see Figure A.7).a. DE generates new points which are perturbations of existing points. This perturbation is performed with the scaled difference of two randomly selected population vectors (see Figure A.7.b). To produce the trial vector $\boldsymbol{u}_0$, DE adds the scaled randomly selected difference to a third randomly selected population vector, as shown in Figure A.7.c.

The trial vector already created competes against the population vector of the same index (in Figure A.7.d the index is 2). The comparison between objective function value returns the future member for the next generation that might be the original or the trial vector, depending on which is lower. In Figure A.7.e the trial vector loses, while in Figure A.7.f another new population vector is randomly generated with index 3. Finally, in Figure A.7.g the new trial vector will replace the actual population vector for index 3.

A.3.1 Population Composition

There exist three population vectors that will be changing in each iteration as will be explained and defined next. The current population vector $P_{\boldsymbol{x},g}$ contains N_p $D-$dimensional vectors $\boldsymbol{x}_{i,g}$ of real-valued parameters that have been found to

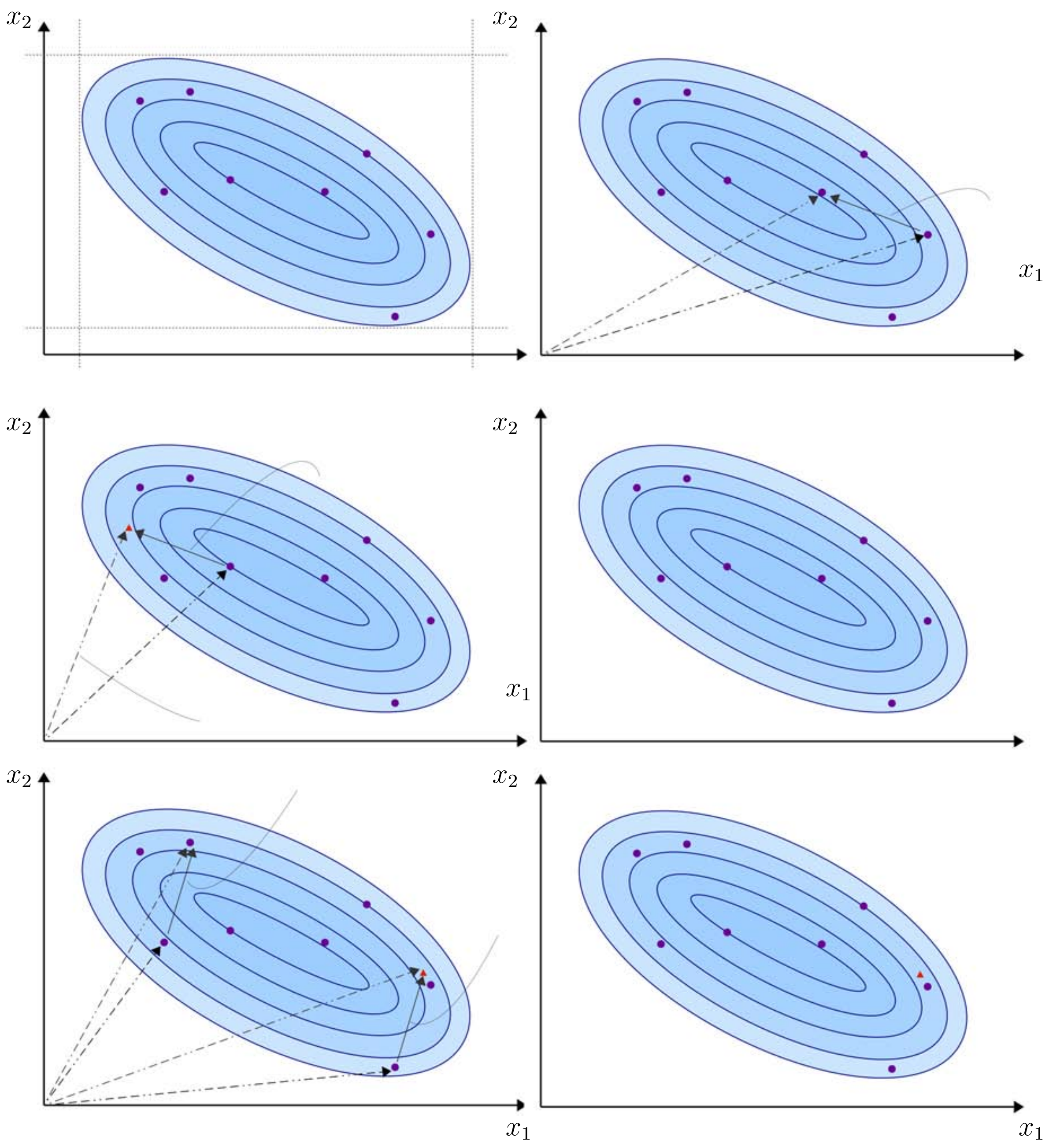


Figure A.7: *DE procedure*. Result of all the steps followed by DE in each iteration.

be acceptable either as initial points or by comparison with other vectors

$$\begin{aligned} P_{\boldsymbol{x},g} &= \{\boldsymbol{x}_{i,g}\} \quad i = 0, 1, \ldots, N_p - 1,\ g = 0, 1, \ldots, g_{max} \\ \boldsymbol{x}_{i,g} &= \{x_{i,j,g}\} \quad j = 0, 1, \ldots, D - 1 \end{aligned} \tag{A.13}$$

where the index g represents the generation to which the vector belongs, i indicates the population element index and j the element in the population vector itself. After the initialization of the algorithm, DE mutates generating an intermediately population vector $P_{\boldsymbol{v}_g}$ composed by mutant vectors $\boldsymbol{v}_{i,g}$ such that

$$\begin{aligned} P_{\boldsymbol{v},g} &= \{\boldsymbol{v}_{i,g}\} \quad i = 0, 1, \ldots, N_p - 1,\ g = 0, 1, \ldots, g_{max} \\ \boldsymbol{v}_{i,g} &= \{v_{i,j,g}\} \quad j = 0, 1, \ldots, D - 1 \end{aligned} \tag{A.14}$$

Then a recombination is produced so a vector $P_{\boldsymbol{u}_g}$ of N_p trial vectors $\boldsymbol{u}_{i,g}$ is created

$$\begin{aligned} P_{\boldsymbol{u},g} &= \{\boldsymbol{u}_{i,g}\} \quad i = 0, 1, \ldots, N_p - 1,\ g = 0, 1, \ldots, g_{max} \\ \boldsymbol{u}_{i,g} &= \{u_{i,j,g}\} \quad j = 0, 1, \ldots, D - 1 \end{aligned} \tag{A.15}$$

A.3.2 Initialization

In order to create a first list of parameters and introduce them in a vector, it is indispensable to define the boundaries of the search function. That is crucial as those limits will focus each evolution iteration on a specific region of the solutions space. Upper and lower bounds can be collected in two $D-$dimensional initialization vectors $\boldsymbol{b}_{\mathcal{U}}$ and $\boldsymbol{b}_{\mathcal{L}}$ so that a random number generator assigns each parameter of every vector a value within the set range. The first generation ($g = 0$) of the $j-$th parameter of the $i-$th vector is

$$x_{i,j,0} = \boldsymbol{b}_{j,\mathcal{L}} + \text{random}_j(0, 1) \cdot (\boldsymbol{b}_{j,\mathcal{U}} - \boldsymbol{b}_{j,\mathcal{L}}) \tag{A.16}$$

The random function $\text{random}_j(0, 1)$ represents a stochastic uniform number generator within the range $0 \leq \text{random}_j(0, 1) < 1$ and a new random value for each parameter j is created.

A.3.3 Mutation

DE has to mutate and recombine the population to produce a new population of N_p trial vectors. This mutation is done by mixing a scaled randomly selected difference of two vectors and summing it to a third one. The combination of those three vectors is a mutant vector $\boldsymbol{v}_{i,g}$ such as

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{r0,g} + F \cdot ((\boldsymbol{x}_{r1,g}) - \boldsymbol{x}_{r2,g}) \tag{A.17}$$

where the scale factor $F \in (0, 1^+)$ represents a positive real number in charge of controlling the population evolution rate. The are three vectors: one is the base vector r_0 and the other two are the difference vectors r_1 and r_2. The three of them are chosen randomly and r_0 must be always different from the target vector with index i.

A.3.4 Crossover

A uniform crossover completes the DE search strategy. The discrete recombination of two vectors creates a trial vector $\boldsymbol{u}_{i,g}$ that is crossed with a mutant vector

$$\boldsymbol{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & if(\text{random}_j(0,1) \leq Cr) \,\text{or}\, j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{A.18}$$

where the crossover factor $0 \leq Cr \leq 1$ controls the proportion of parameter values that will be taken from the mutant vector.

A.3.5 Selection

Finally, in case the trial vector $\boldsymbol{u}_{i,g}$ yields a lower energy than its target vector $\boldsymbol{x}_{i,g}$, it is replaced in the next iteration. Otherwise, the target vector is maintained for one extra generation. The comparison of those two vectors represents the selection step as

$$\boldsymbol{x}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g} & if\ fitness(\boldsymbol{u}_{i,g}) \leq fitness(\boldsymbol{x}_{i,g}) \\ \boldsymbol{x}_{i,g} & \text{otherwise} \end{cases} \tag{A.19}$$

A.3.6 Convergence

The ideal condition that makes the optimization process finalize is when all the restrictions have been satisfied; but this not the general case since in most cases the requirements are not accomplished at the same time. Furthermore, it is not easy to assign a different significance to each condition of the optimization problem. The most common finalizing criteria are listed below.

- *Target reached.*- This criteria is useful in cases were the maximum and minimum of the fitness function are known *a priori*. This is not quite common as these values might vary for each case and the fitness function does not require to be enclosed.

- *Number of iterations.*- As the bounds of the fitness function does not necessarily have to be known, the optimization process may finish after a certain number of iterations are reached. The selection of the threshold has to be determined previously using a trial and error procedure since most of the times it is an empirical value. The presence of *death periods* in DE is more common than in other evolutionary algorithms and for this reason the maximum number of iterations may not have an excessive low value.

- *Population statistics.*- Another choice is to finish the process with the compliance of any significant condition inside the population such as stop when the difference between the best and the lowest population members are below a certain value. This method can be ineffective since it can lead to a fast convergence and conditions must be chosen accordingly.

- *Time limit.*- In some occasions time becomes decisive and therefore the limit may be temporal rather than followed by quality terms. This is usual in real-time applications such as production, manufacturing, navigation, space or biomedicine.

Appendix B

Jensen-Shannon divergence

Metric definition[19]

Let X be the discrete random variable which can have N different values $\in \Omega_N = \{\omega_1, \omega_2 \ldots, \omega_N\}$. An independent and identically distributed sample $\tilde{X}$ is drawn, where each observation is drawn from one of two known distributions, P and Q. Each of those is used with the same probability but it is not possible to recognize which one is chosen. The incentive is to find the coding strategy that returns the shortest average code length for the data representation. That is, the aim is to find the most $efficient$ distribution R.

The mentioned code is named κ. The code lengths are $\kappa_i = -\log r_i$, where $i = \{1, \ldots, N\}$ and r_i represents the probability of $X = \omega_i$ under R. If $\epsilon(\kappa, P)$ represents the expectation of κ with respect to P, the average code length $< \kappa >$ is then $\frac{1}{2}\epsilon(\kappa, P) + \frac{1}{2}\epsilon(\kappa, Q)$. By the very definition of entropy, the $minimum$ $< \kappa >$ is obtained by setting $R = \frac{1}{2}(P + Q)$, that is, $< \kappa >= H(R)$.

An ideal observer, i.e, one who knows which distribution has been used to generate individual data could reach an even shorter average code length $\frac{1}{2}H(P) + \frac{1}{2}H(Q)$. Hence the redundancy of κ is $H(R) - \frac{1}{2}H(P) - \frac{1}{2}H(Q)$. The distance measure Jensen-Shannon is twice this redundancy:

$$\begin{aligned} D^2_{PQ} &= 2 \cdot H(R) - H(P) - H(Q) \\ &= D(P\|R) + D(Q\|R) \qquad \text{(B.1)} \\ &= \sum_{i=1}^{N} \left(p_i \cdot \log \frac{2 \cdot p_i}{p_i + q_i} + q_i \cdot \log \frac{2 \cdot q_i}{p_i + q_i} \right) \qquad \text{(B.2)} \end{aligned}$$

Since the Kullback-Leibler divergence presented in [147] can be interpreted as the inefficiency of assuming that the true distribution is R, when it is really is P, D^2_{PQ} can be understood as a minimum inefficiency distance.

The original publication of this distance metric for distributions was made by Topsøe [236], introduced from an information-transmission point of view. The proof of the metric properties of D^2_{PQ} is listed in the following section.

[19] This demonstration has been fully extracted from Endres and Schindelin publication in [150].

Proof of metric properties

In the following, $\mathbb{R}^+$ includes 0.

Definition I.- Let the function $L(p,q) : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ be defined by

$$L(p,q) := p \cdot \log \frac{2 \cdot p}{p+q} + q \cdot \log \frac{2 \cdot q}{p+q} \tag{B.3}$$

This function can be taken to be any one of the summands of $D_{PQ}2$ (see Equation B.2). By standard inequalities, it is assumed that $L(p,q) \geq 0$, with equality only for $p = q$.

Theorem I uses some properties of the partial derivative of $L(p,q)$ and, to show these, we introduce the function $g : \mathbb{R}^+ \backslash \{1\} \to \mathbb{R}$ defined by

$$g(x) := \frac{\log \frac{2}{1+x}}{\sqrt{L(x,1)}} \tag{B.4}$$

Lemma I.- Let g be defined as above. Then

1. $\lim_{x \to 1^{\mp}} g(x) = \pm 1$ i.e., g jumps from +1 to -1 at $x = 1$.
2. The derivative $\frac{d}{dx} g$ is positive for $x \in \mathbb{R}^+ \backslash \{1\}$.

A consequence of this lemma is that $|g(x) \leq 1|$, with equality only at $x = 1$. Also, it is easy to see that $|g|$ is continuous, but not g.

Proof: First note that g changes sign at $x = 1$. A straightforward application of L'Hôspital rule (differentiate twice) yields $\lim_{x \to 1} g^2(x) = \pm 1$.

By differentiation, one finds that $\frac{d}{dx} g$ is positive if and only if $f < 0$, where f is given by

$$f(x) = \log \frac{2}{1+x} + \log \frac{2 \cdot x}{1+x} \tag{B.5}$$

Straightforward differentiation shows up that $f(1) = f'(1) = 0$ and that

$$f''(x) = \frac{1}{x^2(1+x)} \left(\log \frac{2}{1+x} + x^2 \log \frac{2 \cdot x}{1+x} \right) \tag{B.6}$$

Using the standard inequality $\log a \geq 1 - \frac{1}{a}$, it is found that $f'' < 0$, hence f is concave. Combined with the first found facts, $f < 0$ for $x \neq 1$.

Theorem I.- Let $\mathbb{F}_N$ be the set of all discrete probability distributions over $\Omega_N, N \in \mathbb{N}$. The function $D_{PQ}^{\cdot}\mathbb{F}_N \times \mathbb{F}_N \to \mathbb{R}^+$ is a metric.

Proof.- Recall that $D(P\|Q)$ equals 0 for $P = Q$ and strictly positive otherwise (Demonstrated by [237]). Furthermore, D^2_{PQ} is symmetric in P, Q and so is D_{PQ}. Therefore, only the triangle inequality hold has to be demonstrated.

Lemma II.- Let $p, q, r \in \mathbb{R}^+$. Then

$$\sqrt{L(p,q)} \leq \sqrt{L(p,r)} + \sqrt{L(r,q)} \tag{B.7}$$

Proof: It is easy to see that this holds if any p, q, r are zero. Assuming that $p \leq q$, denote by **rhs** the right-hand side as a function of r, and show that

1. **rhs** has two minima: namely one at $r = p$ and one at $r = q$.
2. Only one maximum somewhere between p and q.

This is shown using the derivative

$$\frac{\partial \boldsymbol{rhs}}{\partial r} = \frac{\log \frac{2\cdot r}{p+r}}{2 \cdot \sqrt{L(p,r)}} + \frac{\log \frac{2\cdot r}{p+r}}{2 \cdot \sqrt{L(1,r)}} \tag{B.8}$$

With g as in *Lemma I* and $x := \frac{p}{q}$ and $\beta \cdot x := \frac{q}{r}$ $(\beta > 1)$, it yields that

$$2 \cdot \sqrt{r} \cdot \frac{\partial \boldsymbol{rhs}}{\partial r} = g(x) + g(\beta \cdot x) \tag{B.9}$$

With $|g(x)| \leq 1$ with equality only at $x = 1$ and the fact that g jumps from +1 to -1 at $x = 1$, as demonstrated in *Lemma I*, the derivative $\frac{\partial \boldsymbol{rhs}}{\partial r}$ indeed changes sign at $r = p$, because then $x = 1$ and $|g(x)| > |g(\beta \cdot x)|$ is monotonic increasing and, as a consequence, has at most one sign change.

Applying Minkowski's inequality to the square root of the sum which defines D_{PQ}, the triangle inequality is fulfilled. Therefore, D_{PQ} is a metric.

The generalization of this result to continuous random variables is straightforward. Let P and Q be the probability measures defined on a measurable space (Ω, A) and let $p = \frac{dP}{d\mu}$, $q = \frac{dQ}{d\mu}$ be their Radon-Nikodym derivatives with relation to a dominating $\sigma-$finite measure μ. Then

$$D_{PQ} = \sqrt{\int_\Omega \left(p \cdot \log \frac{2 \cdot p}{p+q} + q \cdot \log \frac{2 \cdot q}{p+q} \right) d\mu} \tag{B.10}$$

is considered a metric as well. An alternative proof could be constructed using the results presented in [238]. The maxima and minima of D_{PQ} are analyzed. The minimum is found in $P = Q$, as it yields $D_{PQ} = 0$. To find its maximum, Equation B.3 is rewritten as

$$L(p,q) = \underbrace{(p+q)log2}_{\geq 0} + \underbrace{p \cdot \log \frac{p}{p+q}}_{\leq 0} + \underbrace{q \cdot \log \frac{q}{p+q}}_{\leq 0} \tag{B.11}$$

It follows that when P and Q are two distinct deterministic distributions, D_{PQ} assumes its maximum value $\sqrt{2 \cdot \log 2}$.

Bibliography

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989.

[2] R. Lange and P. Seitz, "Solid-state Time-of-Flight Range Camera," *IEEE Journal of Quantum Electronics*, vol. 37, pp. 390–397, Mar. 2001.

[3] M. Gong and Y. Yee-Hong, "Fast stereo matching using reliability-based dynamic programming and consistency constraints," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 610–617, IEEE, 2003.

[4] H. Wu-Chih, "Adaptive template block-based block matching for object tracking," in *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, vol. 1, pp. 61–64, IEEE, 2008.

[5] J. Davis, D. Nehab, R. Ramamoorthi, and S. Rusinkiewicz, "Spacetime stereo: a unifying framework for depth from triangulation.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 2, pp. 296–302, 2005.

[6] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly USA, 2008.

[7] A. Martín Clemente, "Generación de Mapas de Disparidad usando CUDA," Master's thesis, Universidad Carlos III de Madrid, 2009.

[8] J. Liu, Y. Xu, R. Klette, H. Chen, and T. Vaudrey, "Disparity Map Computation on a Cell Processor," *Architecture*, p. 2009.

[9] C. B. Boyer, "Early Estimates of the Velocity of Light," *The University Of Chicago Press On Behalf Of The History Of Science*, vol. 33, no. 1, pp. 24–40, 1941.

[10] A. Kolb, E. Barth, and R. Koch, "ToF-sensors: New dimensions for realism and interactivity," *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–6, June 2008.

[11] R. Lange, *3D Time-of-Flight Distance Measurement with Custom Solid-State Image Sensors in CMOS/CCD-Technology*. PhD thesis, University of Siegen, 2000.

[12] S. Hussmann and T. Liepert, "Robot Vision System based on a 3D-TOF Camera," *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*, pp. 1–5, May 2007.

[13] M. Böhme, M. Haker, T. Martinetz, and E. Barth, "Shading constraint improves accuracy of time-of-flight measurements," *Computer vision and image understanding*, vol. 114, no. 12, pp. 1329–1335, 2010.

[14] S. Hussmann, T. Ringbeck, and B. Hagebeuker, "A Performance Review of 3D TOF Vision Systems in Comparison to Stereo Vision Systems," *Review Literature And Arts Of The Americas*, no. November, 2008.

[15] I. Moring, T. Heikkinen, R. Myllyla, and A. Kilpela, "Acquisition of three- dimensional image data by a scanning laser range finder," *Optical Engineering*, vol. 28, no. 8, pp. 897–902, 1989.

[16] G. Beheim and K. Fritsch, "Range finding using frequency-modulated laser diode.," *Applied optics*, vol. 25, p. 1439, May 1986.

[17] S. Hussmann, T. Ringbeck, and B. Hagebeuker, "A performance review of 3D ToF vision systems in comparison to stereo vision systems," *Stereo Vision*, pp. 103–120, 2008.

[18] R. Chung, "Correspondence stereo vision under general stereo camera configuration," in *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*, vol. 1, pp. 405–410, IEEE, 2003.

[19] T. Kahlmann, F. Remondino, and H. Ingensand, "Calibration for increased accuracy of the range imaging camera swissrangertm," *ISPRS Commission V Symposium Image Engineering and Vision Metrology*, no. 4, pp. 136–141, 2006.

[20] A. Lindner, Marvin and Kolb, "Lateral and Depth Calibration of PMD-Distance Sensors," in *Advances in Visual Computing*, vol. 4292, pp. 524–533, Springer Berlin Heidelberg, 2006.

[21] S. M. Fuchs and Stefan, "Calibration and Registration for Precise Surface Reconstruction with ToF Cameras," tech. rep., DLR, Institute of Robotics and Mechatronics, 2007.

[22] M. Lindner, A. Kolb, and T. Ringbeck, "New insights into the calibration of ToF-sensors," *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–5, June 2008.

[23] S. A. Gudmundsson, H. Aanaes, and R. Larsen, "Environmental Effects on Measurement Uncertainties of Time-of-Flight Cameras," in *2007 International Symposium on Signals, Circuits and Systems*, vol. 1, pp. 1–4, IEEE, July 2007.

[24] J. G. Bueno, P. J. Slupska, N. Burrus, and L. Moreno, "Textureless Object Recognition and Arm Planning for a Mobile Manipulator," in *53rd International Symposium ELMAR-2011*, (Zadar, Croatia), pp. 59 – 62, 2011.

[25] N. Burrus, J. G. Bueno, L. Moreno, and M. Abderrahim, "3D Object Model Acquisition and Recognition with a Time-Of-Flight Camera," in *7th Wokshop Robocity2030. Vision in robotics*, pp. 77–91, 2001.

[26] S. Jamtsho and D. Lichti, "Modelling scattering distortion in 3D range camera," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 5, pp. 299–304, 2010.

[27] L. Zhang, B. Curless, and S. M. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming," in *3D Data Processing Visualization and Transmission, 2002. First International Symposium on*, pp. 24–36, IEEE, 2002.

[28] L. Cruz, D. Lucio, and L. Velho, "Kinect and RGB-D images: Challenges and applications," in *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on*, pp. 36–49, IEEE, 2012.

[29] J. Kramer, N. Burrus, F. Echtler, H. Daniel, and M. Parker, *Hacking the Kinect*. Apress, 2012.

[30] G. Gerig, "Structured Lighting," *Carnegie Mellon University. 3D Computer Vision subject*, 2012.

[31] B. Pan, Q. Guan, X. Wang, and S. Y. Chen, "Strategies of Real-Time 3D Reconstruction by Structured Light," in *2010 Chinese Conference on Pattern Recognition (CCPR)*, pp. 1–5, IEEE, Oct. 2010.

[32] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of Kinect depth data for indoor mapping applications," *Sensors (Basel, Switzerland)*, vol. 12, pp. 1437–1454, Jan. 2012.

[33] J. F. C. Rascon and L. M. Lorente, *Localización Global 2D de Robots Móviles usando Evolución Diferencial dentro del Framework ROS*. PhD thesis, Universidad Carlos III de Madrid, 2012.

[34] F. M. Monar, *Evolutionary-based global localization and mapping of three dimensional environments*. PhD thesis, Universidad Carlos III de Madrid, 2012.

[35] T. Mouri, H. Kawasaki, K. Yoshikawa, J. Takai, and S. Ito, "Anthropomorphic Robot Hand: Gifu Hand III," *International Conference on Control, Automation and Systems*, pp. 1288–1293, 2002.

[36] R. Dillmann, "Stereo-based 6D object localization for grasping with humanoid robot systems," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 919–924, Oct. 2007.

[37] R. B. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz, "Perception for mobile manipulation and grasping using active stereo," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 632–638, IEEE, Dec. 2009.

[38] M. Zuliani, "RANSAC for Dummies," 2012.

[39] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 International Conference on Computer Vision*, pp. 858–865, IEEE, Nov. 2011.

[40] A. Cherian, V. Morellas, and N. Papanikolopoulos, "Accurate 3D ground plane estimation from a single image," *2009 IEEE International Conference on Robotics and Automation*, pp. 2243–2249, May 2009.

[41] A. Milella, G. Reina, J. Underwood, and B. Douillard, "Combining radar and vision for self-supervised ground segmentation in outdoor environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 255–260, IEEE, Sept. 2011.

[42] A. G. Linarth, M. Brucker, and E. Angelopoulou, "Robust ground plane estimation based on particle filters," *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pp. 1–7, Oct. 2009.

[43] B. Enjarini and a. Graser, "Planar segmentation from depth images using gradient of depth feature," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4668–4674, Oct. 2012.

[44] J. G. Bueno, P. J. Slupska, N. Burrus, L. Moreno, and M. Abderrahim, "Robust Pedestrian Detection using a Time-Of-Flight Camera," in *8th Wokshop Robocity2030. Robots de exteriores*, pp. 60–76, 2010.

[45] G. Klein and D. W. Murray, "Full-3D Edge Tracking with a Particle Filter," *Procedings of the British Machine Vision Conference 2006*, pp. 1119–1128, 2006.

[46] E. Wahl and G. Hirzinger, "Cluster-based point cloud analysis for rapid scene interpretation," *Pattern Recognition*, pp. 160–167, 2005.

[47] J. Stuckler and S. Behnke, "Combining depth and color cues for scale- and viewpoint-invariant object segmentation and recognition using Random Forests," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4566–4571, Oct. 2010.

[48] J. Strom, A. Richardson, and E. Olson, "Graph-based segmentation for colored 3d laser point clouds," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 2131–2136, IEEE, 2010.

[49] A. Golovinskiy and T. Funkhouser, "Min-cut based segmentation of point clouds," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 39–46, IEEE, 2009.

[50] Fang Yuan, Zeng-Hui Meng, Hong-Xia Zhang, and Chun-Ru Dong, "A new algorithm to get the initial centroids," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 2, pp. 1191–1193, IEEE, 2004.

[51] D. Napoleon and P. G. Lakshmi, "An efficient K-Means clustering algorithm for reducing time complexity using uniform distribution data points," *Trendz in Information Sciences & Computing(TISC2010)*, pp. 42–45, Dec. 2010.

[52] D. Liu and J. Yu, "Otsu Method and K-means," *2009 Ninth International Conference on Hybrid Intelligent Systems*, no. 2, pp. 344–349, 2009.

[53] S. Na, L. Xumin, and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 63–67, Apr. 2010.

[54] B. Hong-tao, H. Li-li, O. Dan-tong, L. Zhan-shan, and L. He, "K-Means on Commodity GPUs with CUDA," *2009 WRI World Congress on Computer Science and Information Engineering*, pp. 651–655, 2009.

[55] T. K. Dey and J. A. Levine, "Delaunay Meshing of Isosurfaces," *IEEE International Conference on Shape Modeling and Applications 2007 (SMI '07)*, pp. 241–250, June 2007.

[56] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Proceedings of the 19th annual conference on Computer graphics and interactive techniques - SIGGRAPH '92*, pp. 71–78, 1992.

[57] R. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," *KI-Künstliche Intelligenz*, vol. 4, no. 24, pp. 345–348, 2010.

[58] D. G. Muja, Marius and Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, pp. 331—-340, 2009.

[59] C. A. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962.

[60] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen, *Introduction to algorithms*. The MIT press, 2001.

[61] R. E. Bellman, *Dynamic Programming*. Princeton: Princeton University Press, 2010.

[62] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[63] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 1, pp. 129–147, 1982.

[64] Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun, "Multi-view image and ToF sensor fusion for dense 3D reconstruction," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 1542–1549, IEEE, Sept. 2009.

[65] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt, "3D shape scanning with a Time-of-Flight camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1173–1180, IEEE, June 2010.

[66] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3D object modeling," *The International Journal of Robotics Research*, vol. 30, pp. 1311–1327, July 2011.

[67] G. Tam, C. Zhi-Quan, L. Yu-Kun, L. Frank, L. Yonghuai, A. Marshall, R. Martin, X. Sun, and P. Rosin, "Registration of 3D Point Clouds and Meshes: A Survey From Rigid to Non-Rigid," *IEEE transactions on Visualization and Computer Graphics*, vol. 19, pp. 1199–1217, July 2013.

[68] K. N. Kutulakos and S. M. Seitz, "A Theory of Shape by Space Carving," *International Journal of Computer Vision*, vol. 38, pp. 199–218, July 2000.

[69] Y. Yemez and C. Wetherilt, "A volumetric fusion technique for surface reconstruction from silhouettes and range data," *Computer Vision and Image Understanding*, vol. 105, pp. 30–41, Jan. 2007.

[70] G. Walck and M. Drouin, "Progressive 3D reconstruction of unknown objects using one eye-in-hand camera," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics*, pp. 971–976, IEEE Press, Dec. 2009.

[71] Y. Yemez and F. Schmitt, "3D reconstruction of real objects with high resolution shape and texture," *Image and Vision Computing*, vol. 22, no. 13, pp. 1137–1153, 2004.

[72] J.-S. Franco and E. Boyer, "Fusion of multiview silhouette cues using a space occupancy grid," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1747–1753 Vol. 2, IEEE, 2005.

[73] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 163–169, Aug. 1987.

[74] K. Grauman, G. Shakhnarovich, and T. Darrell, "A Bayesian approach to image-based visual hull reconstruction," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, pp. 187–194, IEEE Comput. Soc, 2003.

[75] N. Burrus, M. Abderrahim, J. Garcia, and L. Moreno, "Object reconstruction and recognition leveraging an RGB-D camera," in *12th IAPR Conference on Machine Vision Applications*, (Nara, Japan), pp. 132–135, 2011.

[76] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Eurographics Association*, pp. 61–70, June 2006.

[77] G. Kordelas, J. D. Agapito, J. M. H. Vegas, and P. Daras, "State-of-the-art Algorithms for Complete 3D Model Reconstruction," *Summer School Engage. University of Geneva*, 2010.

[78] M. Sainz, N. Bagherzadeh, and A. Susin, "Hardware Accelerated Voxel Carving," in *1st Iberoamerican Symposium in Computer Graphics*, pp. 289–297, 2002.

[79] C. Zach, K. Karner, B. Reitinger, and H. Bischof, "Space carving on 3D graphics hardware," *Algorithms*, vol. 14, p. 7, 2004.

[80] H. Zhao, S. Oshery, and R. Fedkiwz, "Fast surface reconstruction using the level set method," in *Variational and Level Set Methods in Computer Vision, 2001. IEEE Workshop on*, pp. 194–201, 2001.

[81] S. Daniel and G. Taubin, *An Energy Minimization Approach to Surface Reconstruction*. PhD thesis, Brown Computer Science, 2009.

[82] Y. Duan, L. Yang, H. Qin, and D. Samaras, "Shape Reconstruction from 3D and 2D Data Using PDE-Based Deformable Surfaces," in *ECCV 2004 Lecture Notes in Computer Science*, vol. 3023, pp. 238–251, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[83] H. Jin, S. Soatto, and A. J. Yezzi, "Multi-View Stereo Reconstruction of Dense Shape and Complex Appearance," *International Journal of Computer Vision*, vol. 63, pp. 175–189, Apr. 2005.

[84] G. Slabaugh, R. Schafer, and M. Hans, "Multi-resolution space carving using level set methods," in *International Conference on Image Processing*, vol. 2, pp. 545–548, IEEE, 2002.

[85] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01*, (New York, USA), pp. 67–76, ACM Press, Aug. 2001.

[86] J. P. Pons, R. Keriven, and O. Faugeras, "Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score," *International Journal of Computer Vision*, vol. 72, pp. 179–193, July 2006.

[87] N. Ahuja, "SDG Cut: 3D Reconstruction of Non-lambertian Objects Using Graph Cuts on Surface Distance Grid," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognitio CVPR'06*, vol. 2, pp. 2269–2276, IEEE, 2006.

[88] Y. Boykov, "Computing geodesics and minimal surfaces via graph cuts," in *International Conference on Computer Vision*, pp. 26–33, 2003.

[89] A. Hornung, E. Hornung, and L. Kobbelt, "Hierarchical Volumetric Multi-view Stereo Reconstruction of Manifold Surfaces Based on Dual Graph Embedding," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 503–510, 2006.

[90] J. Shi, M. Wan, X.-C. Tai, and D. Wang, *Curvature Minimization for Surface Reconstruction with Features*, vol. 6667 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[91] A. Bienert and H.-G. Maas, "Methods for the automatic geometric registration of terrestrial laser scanner point clouds in forest stands," in *Laser Scanning*, vol. 38, (Paris), pp. 93–98, 2009.

[92] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai, "Integrating Automated Range Registration with Multiview Geometry for the Photorealistic Modeling of Large-Scale Scenes," *International Journal of Computer Vision*, vol. 78, pp. 237–260, Nov. 2007.

[93] Chen Chao and I. Stamos, "Semi-Automatic Range to Range Registration: A Feature-Based Method," in *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, pp. 254–261, IEEE, 2005.

[94] J. Jiang, J. Cheng, and X. Chen, "Registration for 3D point cloud using angular-invariant feature," *Neurocomputing*, vol. 72, no. 16, pp. 3839–3844, 2009.

[95] A. Sappa, A. Restrepo-Specht, and M. Devy, "Range Image Registration by using an Edge-based Representation," in *9th International Symposium on Intelligent Robotic Systems SIRS*, pp. 167–176, 2001.

[96] O. Faugeras and M. Hebert, "The Representation, Recognition, and Locating of 3D Objects," *The International Journal of Robotics Research*, vol. 5, pp. 27–52, Sept. 1986.

[97] J. Vanden Wyngaerd and L. Van Gool, "Automatic Crude Patch Registration: Toward Automatic 3D Model Building," *Computer Vision and Image Understanding*, vol. 87, no. 1, pp. 8–26, 2002.

[98] S. Yamany, M. N. Ahmed, and A. A. Farag, "A New Genetic-Based Technique for Matching 3D Curves and Surfaces," *Pattern Recognition*, vol. 32, pp. 1817–1820, 1999.

[99] C. S. Chua and R. Jarvis, "3d free-form surface registration and object recognition," *International Journal of Computer Vision*, vol. 17, no. 1, pp. 77–99, 1996.

[100] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *Pattern Analysis and Machine Intelligence*, vol. 21, pp. 433–449, May 1999.

[101] G. Sharp, S. Lee, and D. Wehe, "ICP registration using invariant features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 90–102, 2002.

[102] X. Qian, I. Horváth, T. Kim, Y. Seo, S. Lee, Z. Yang, and M. Chang, "Simultaneous registration of multiple views with markers," *Computer-Aided Design*, vol. 41, no. 4, pp. 231–239, 2009.

[103] D. Akca, "Full automatic registration of laser scanner point clouds," *ETH, Swiss Federal Institute of Technology Zurich, Institute of Geodesy and Photogrammetry*, pp. 1–8, 2003.

[104] C. Chua and R. Jarvis, "Point signatures: A new representation for 3D object recognition," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 63–85, 1997.

[105] C. K. Chow, H. T. Tsui, and T. Lee, "Surface registration using a dynamic genetic algorithm," *Pattern Recognition*, vol. 37, pp. 105–117, Jan. 2004.

[106] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, pp. 1584–1601, Oct. 2006.

[107] A. Makadia, A. Patterson, and K. Daniilidis, "Fully Automatic Registration of 3D Point Clouds," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 1297–1304, IEEE, June 2006.

[108] S. B. Kang and I. Katsushi, "The Complex EGI: A New Representation for 3D Pose Determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 7, pp. 707–721, 1993.

[109] I. Okatani and A. Sugimoto, "Registration of range images that preserves local surface structures and color," in *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT '04)*, pp. 789–796, IEEE, 2004.

[110] G. H. Bendels, P. Degener, R. Wahl, M. Körtgen, and R. Klein, "Image-Based Registration of 3D-Range Data Using Feature Surface Elements," in *5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*, pp. 115–124, 2004.

[111] K. Qian, X. Ma, F. Fang, and H. Yang, "3D environmental mapping of mobile robot using a low-cost depth camera," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, pp. 507–512, 2013.

[112] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, pp. 119–152, 1994.

[113] M. Greenspan and G. Godin, "A nearest neighbor method for efficient ICP," in *3D Digital Imaging and Modeling, Third International Conference on*, pp. 161–168, IEEE Comput. Soc, 2001.

[114] P. J. Besl, N. D. McKay, and H. McKay, "A method for registration of 3D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[115] G. Godin, D. Laurendeau, and R. Bergevin, "A Method for the Registration of Attributed Range Images," in *Third International Conference on 3D Digital Imaging and Modeling 3DIM*, pp. 179–186, 2001.

[116] E. Trucco, A. Fusiello, and V. Roberto, "Robust motion and correspondence of noisy 3D point sets with missing data," *Pattern Recognition Letters*, vol. 20, pp. 889–898, Sept. 1999.

[117] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, IEEE, Nov. 2007.

[118] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *International Conference on Computer Vision*, pp. 2320–2327, IEEE, Nov. 2011.

[119] Byung-Uk Lee, Chul-Min Kim, and Rae-Hong Park, "An orientation reliability matrix for the iterative closest point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1205–1208, 2000.

[120] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," *10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, Oct. 2011.

[121] S. Izadi, A. Davison, A. Fitzgibbon, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, and D. Freeman, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth

Camera," in *24th annual ACM symposium on User interface software and technology (UIST '11)*, (New York, USA), p. 559, ACM Press, Oct. 2011.

[122] H. Roth and M. Vona, "Moving volume kinectfusion," in *Proceedings of the British Machine Vision Conference*, pp. 112.1–112.11, BMVA Press, 2012.

[123] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1, pp. 21–74, 2002.

[124] S. Yamany and A. Farag, "Free-form surface registration using surface signatures," in *Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1098–1104 vol.2, IEEE, 1999.

[125] C. Schutz, T. Jost, and H. Hugli, "Multi-feature matching algorithm for free-form 3D surface registration," in *Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, vol. 2, pp. 982–984, IEEE Computer Society, 1998.

[126] T. Masuda and N. Yokoya, "A Robust Method for Registration and Segmentation of Multiple Range Images," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 295–307, 1995.

[127] K. Brunnstrom and A. Stoddart, "Genetic algorithms for free-form surface matching," in *13th International Conference on Pattern Recognition*, vol. 4, pp. 689–693, IEEE, 1996.

[128] M. Salomon, G.-R. Perrin, and F. Heitz, "Differential Evolution for Medical Image Registration," in *International Conference on Artificial Intelligence*, pp. 201–207, 2001.

[129] C. Robertson and R. B. Fisher, "Parallel Evolutionary Registration of Range Data," *Computer Vision and Image Understanding*, vol. 87, no. 1, pp. 39–50, 2002.

[130] L. Silva, O. R. Bellon, and K. L. Boyer, "Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, pp. 762–776, May 2005.

[131] G. Dalley and P. Flynn, "Range image registration: A software platform and empirical evaluation," in *Third International Conference on 3D Digital Imaging and Modeling*, pp. 246–253, IEEE Computer Society, 2001.

[132] G. Olague, B. Hernandez, and E. Dunn, "Hybrid evolutionary ridge regression approach for high-accurate corner extraction," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 744–749, IEEE Computer Society, 2003.

[133] R. Dony, "Differential Evolution with Powell's direction set method in medical image registration," in *2nd IEEE International Symposium on Biomedical Imaging: Macro to Nano (IEEE Cat No. 04EX821)*, vol. 2, pp. 732–735, IEEE, 2004.

[134] E. R. Smith, B. J. King, C. V. Stewart, and R. J. Radke, "Registration of combined range–intensity scans: Initialization through verification," *Computer Vision and Image Understanding*, vol. 110, pp. 226–244, May 2008.

[135] F. Martín, C. Gonzalez Uzcátegui, L. Moreno, and D. Blanco, "Accelerated Localization in Noisy 3D Environments using Differential Evolution," in *Proceedings of the International Conference on Genetic and Evolutionary Methods*, (Las Vegas, USA), pp. 166–172, 2010.

[136] L. Moreno, S. Garrido, F. Martín, and M. L. Muñoz, "Differential Evolution approach to the grid-based localization and mapping problem," in *International Conference on Intelligent Robots and Systems (IROS'07) IEEE/RSJ*, (San Diego, USA), pp. 3479–3484, 2007.

[137] A. R. Vahdat, N. NourAshrafoddin, and S. S. Ghidary, "Mobile robot global localization using differential evolution and particle swarm optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 1527–1534, IEEE, 2007.

[138] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, Dec. 1997.

[139] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *16th International Conference on Pattern Recognition*, vol. 3, pp. 545–548, IEEE, 2002.

[140] D. Chetverikov, D. Stepanov, and P. Krsek, "Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm," *Image and Vision Computing*, vol. 23, no. 3, pp. 299–309, 2005.

[141] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *IEEE International Conference on Robotics and Automation*, pp. 2724–2729, IEEE Computer Society Press, 1991.

[142] C. Dorai, J. Weng, and A. Jain, "Optimal registration of object views using range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1131–1138, 1997.

[143] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification," in *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM 2003)*, pp. 474–481, IEEE, 2003.

[144] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Conference on Computer Vision and Pattern Recognition*, pp. 998–1005, IEEE, June 2010.

[145] R. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," *Robotics and Automation (ICRA'09). IEEE International Conference on*, pp. 3212–3217, 2009.

[146] T. Fiolka, J. Stückler, D. A. Klein, D. Schulz, and S. Behnke, "SURE: Surface Entropy for Distinctive 3D Features," in *Spatial Cognition VIII*, pp. 74–93, Springer, 2012.

[147] S. Kullback, "Letter to the Editor: The Kullback–Leibler distance," *The American Statistician*, vol. 41, no. 4, pp. 340–341, 1987.

[148] Kai-Kuang Ma and Junxian Wang, "Color distance histogram: a novel descriptor for color image segmentation," in *7th International Conference on Control, Automation, Robotics and Vision (ICARCV '02)*, vol. 3, pp. 1228–1232, 2002.

[149] N. Kyriakoulis and A. Gasteratos, "Light-invariant 3D object's pose estimation using color distance transform," in *International Conference on Imaging Systems and Techniques*, pp. 105–110, IEEE, July 2010.

[150] D. Endres and J. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, pp. 1858–1860, July 2003.

[151] M. Rusu, R. B., Marton, Z. C., Blodow, N., Holzbach, A., & Beetz, "Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments," *In Intelligent Robots and Systems (IROS 2009). IEEE/RSJ International Conference on*, pp. 3601–3608, 2009.

[152] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun., "Using EM to learn 3D models of indoor environments with mobile robots," in *International Conference on Machine Learning (ICML)*, pp. 329–336, IEEE, 2001.

[153] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," *ACM Transactions on Graphics (TOG '02)*, vol. 21, no. 3, pp. 438–446, 2002.

[154] R. Beserra Gomes, B. M. Ferreira da Silva, L. K. Rocha, R. V. Aroca, L. C. Velho, and L. M. Gonçalves, "Efficient 3D object recognition using foveated point clouds," *Computers & Graphics*, vol. 37, pp. 496–508, Aug. 2013.

[155] A. P. Ashbrook, R. B. Fisher, C. Robertson, and N. Werghi, "Finding Surface Correspondance for Object Recognition and Registration Using Pairwise Geometric Histograms," in *Computer Vision-ECCV'98*, pp. 674–686, Springer-Verlag, June 1998.

[156] A. Johnson and M. Hebert, "Surface matching for object recognition in complex three-dimensional scenes," *Image and Vision Computing*, vol. 16, no. 9, pp. 635–651, 1998.

[157] A. E. Johnson, *Spin-images:A representation for 3D surface matching*. PhD thesis, Carnegie Mellon University, 1997.

[158] P. Hough V C, "Pattent: Method and means for recognizing complex patterns," Dec. 1962.

[159] A. S. Mian, M. Bennamoun, and R. A. Owens, "A Novel Representation and Feature Matching Algorithm for Automatic Pairwise Registration of Range Images," *International Journal of Computer Vision*, vol. 66, pp. 19–40, 2006.

[160] J. W. Tangelder and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools and Applications*, vol. 39, pp. 441–471, Dec. 2007.

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

[162] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model recognition and 6DOF pose estimation using 3D cues," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 585–592, IEEE, Nov. 2011.

[163] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision ECCV 2010*, pp. 356–369, Springer, 2010.

[164] G. Hetzel, B. Leibe, P. Levi, and B. Schiele, “3D object recognition from range images using local feature histograms,” in *Computer Vision and Pattern Recognition (CVPR '01)*, vol. 2, pp. 394–399, IEEE Computer Society, 2001.

[165] R. Rusu, N. Blodow, Z. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” in *International Conference on Intelligent Robots and Systems (IROS '08)*, pp. 3384–3391, IEEE, Sept. 2008.

[166] H. Chen and B. Bhanu, “3D free-form object recognition in range images using local surface patches,” *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1252–1262, 2007.

[167] F. Tombari and L. Di Stefano, “Object Recognition in 3D Scenes with Occlusions and Clutter by Hough Voting,” in *Fourth Pacific-Rim Symposium on Image and Video Technology*, pp. 349–355, IEEE, Nov. 2010.

[168] A. Mian, M. Bennamoun, and R. Owens, “On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes,” *International Journal of Computer Vision*, vol. 89, pp. 348–361, Sept. 2009.

[169] Y. Zhong, “Intrinsic shape signatures: A shape descriptor for 3D object recognition,” in *12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 689–696, IEEE, Sept. 2009.

[170] C. Papazov and D. Burschka, “An efficient RANSAC for 3D object recognition in noisy and occluded scenes,” in *Computer Vision (ACCV '10)*, pp. 135–148, Springer, 2011.

[171] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, “Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation,” *IEEE Robotics & Automation Magazine*, vol. 19, pp. 80–91, Sept. 2012.

[172] M. Blum, J. Wulfing, and M. Riedmiller, “A learned feature descriptor for object recognition in RGB-D data,” in *International Conference on Robotics and Automation (ICRA '12)*, pp. 1298–1303, IEEE, May 2012.

[173] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, “3DNet: Large-scale object class recognition from CAD models,” in *International Conference on Robotics and Automation (ICRA '12)*, pp. 5384–5391, IEEE, May 2012.

[174] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *International Conference on Robotics and Automation (ICRA '11)*, pp. 1817–1824, IEEE, May 2011.

[175] W. Wohlkinger and M. Vincze, "Shape-based depth image to 3D model matching and classification with inter-view similarity," in *International Conference on Intelligent Robots and Systems (IROS '11)*, pp. 4865–4870, IEEE, Sept. 2011.

[176] R. Palmer, M. Borck, G. West, and T. Tan, "Intensity and Range Image based Features for Object Detection in Mobile Mapping Data," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, no. September, pp. 315–320, 2012.

[177] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *Third International Conference on 3D Digital Imaging and Modeling*, pp. 145–152, 2001.

[178] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional SIFT descriptor and its application to action recognition," in *15th International Conference on Multimedia*, pp. 357–360, ACM, 2007.

[179] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3D SURF for robust three dimensional classification," in *Computer Vision (ECCV '10)*, pp. 589–602, Springer, 2010.

[180] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Computer Vision (ECCV '06)*, pp. 404–417, 2006.

[181] W. Steder, B., Rusu, R. B., Konolige, K., & Burgard, "NARF: 3D range image features for object recognition," *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS '10)*, vol. 44, 2010.

[182] B. Steder, G. Grisetti, M. Van Loock, and W. Burgard, "Robust on-line model-based object detection from range images," *International Conference on Intelligent Robots and Systems (IROS '09)*, pp. 4739–4744, Oct. 2009.

[183] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, pp. 147–152, 1988.

[184] E. Michaelsen, W. V. Hansen, M. Kirchhof, J. Meidow, and U. Stilla, "Estimatting the Essential Matrix: GOODSAC versus RANSAC," in *Symposium on Photogrammetric Computer vision (PCV '06)*, 2006.

[185] B. Steder, *Feature-Based 3D Perception for Mobile Robots*. PhD thesis, Albert-Ludwigs Freiburg University, 2013.

[186] D. Skiff, *Autonomous Generation, Segmentation, and Categorization of Point Clouds*. PhD thesis, Cornell University, 2012.

[187] S. Grzonka, B. Steder, and W. Burgard, "3D Place Recognition and Object Detection using a Small-sized Quadrotor," in *Robotics: Science and Systems (RSS), Workshop on 3D Exploration, Mapping, and Surveillance with Aerial Robots*, IEEE, 2011.

[188] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Seventh International Conference on Computer Vision (ICCV'99)*, vol. 2, pp. 1150–1157, 1999.

[189] N. Zhang, M. Li, and B. Hong, "Active Mobile Robot Simultaneous Localization and Mapping," in *Robotics and Biomimetics (ROBIO'06). IEEE International Conference on*, pp. 1676–1681, IEEE, Dec. 2006.

[190] D. Schleicher, L. M. Bergasa, R. Barea, E. López, M. Ocaña, and J. Nuevo, "Real-time wide-angle stereo visual SLAM on large environments using SIFT features correction," in *International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 3878–3883, IEEE, Oct. 2007.

[191] A. Alguacil Gómez, "Aplicaciones del operador SIFT al reconocimiento de objetos," Master's thesis, Universidad Carlos III Madrid, 2009.

[192] W. Cheung and G. Hamarneh, "n-SIFT: n-dimensional Scale Invariant Feature Transform.," *IEEE transactions on Image Processing : A Publication of the IEEE Signal Processing Society*, vol. 18, pp. 2012–2021, Sept. 2009.

[193] X. Jiang, "Face Recognition Using SIFT Features," in *16th International Conference on Image Processing (ICIP '09)*, pp. 3313–3316, IEEE, Nov. 2009.

[194] T. R. Lo and J. P. Siebert, "Local Feature Extraction and Matching on Range Images: 2.5D SIFT," *Computer Vision and Image Understanding*, vol. 113, pp. 1235–1250, Dec. 2009.

[195] C. Maes, T. Fabry, J. Keustermans, D. Smeets, P. Suetens, and D. Vandermeulen, "Feature detection on 3D face surfaces for pose normalisation and recognition," in *Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS'10)*, pp. 1–6, IEEE, Sept. 2010.

[196] L. He, S. Wang, and T. N. Pappas, "3D surface registration using Z-SIFT," in *18th IEEE International Conference on Image Processing*, pp. 1985–1988, IEEE, Sept. 2011.

[197] J. G. Bueno, C. M. Alejandro Iván, L. Moreno, and C. Balaguer, "Distinguishing between Similar Objects based on Geometrical Features in 3D Perception," in *RoboCity2030 12th Workshop: Robótica Cognitiva*, pp. 46–58, 2013.

[198] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics*, vol. 21, pp. 807–832, Oct. 2002.

[199] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors," in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP'03)*, pp. 156–164, Eurographics Association, June 2003.

[200] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton Based Shape Matching and Retrieval," in *Shape Modeling International, (SMI '03)*, p. 130, IEEE Computer Society, May 2003.

[201] Y. Makihara, M. Takizawa, Y. Shirai, J. Miura, and N. Shimada, "Object recognition supported by user interaction for service robots," in *16th International Conference on Pattern Recognition*, vol. 3, pp. 561–564, IEEE, 2002.

[202] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranić, "Feature-based similarity search in 3D object databases," *ACM Computing Surveys*, vol. 37, pp. 345–387, Dec. 2005.

[203] M. Heczko, D. Keim, D. Saupe, and D. Vranic, "Methods for similarity search on 3D databases.," *Datenbank-Spektrum*, vol. 2, no. 2, pp. 54 – 63, 2002.

[204] D. Vranic and D. Saupe, "Description of 3D-shape using a Complex Function on the Sphere," in *International Conference on Multimedia and Expo (ICME'02)*, vol. 1, pp. 177–180, IEEE, 2002.

[205] J. Song and F. Golshani, "3D Object Retrieval by Shape Similarity," in *13th International Conference on Database and Expert Systems Applications (DEXA '02)*, pp. 851–860, Springer-Verlag, Sept. 2002.

[206] T. Ansary, J. Vandeborre, S. Mahmoudi, and M. Daoudi, "A Bayesian framework for 3D models retrieval based on characteristic views," in *2nd International Symposium on 3D Data Processing, Visualization and Transmissio (3DPVT '04)*, pp. 139–146, IEEE, 2004.

[207] D. Vranic, D. Saupe, and J. Richter, "Tools for 3D-object Retrieval: Karhunen-Loeve Transform and Spherical Harmonics," in *Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564)*, pp. 293–298, IEEE, 2001.

[208] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3D shapes," in *28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01)*, (New York, USA), pp. 203–212, ACM Press, Aug. 2001.

[209] N. Gagvani and D. Silver, "Parameter-Controlled Volume Thinning," *Graphical Models and Image Processing*, vol. 61, no. 3, pp. 149–164, 1999.

[210] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based Object Labeling in 3D Scenes," in *Robotics and Automation (ICRA '12) IEEE International Conference on*, pp. 1330–1337, IEEE, 2012.

[211] J. G. Bueno, M. G. Fierro, L. Moreno, and C. Balaguer, "Facial Gesture Recognition using Active Appearance Models based on Neural Evolution," in *Conference on Human-Robot Interaction (HRI 2012)*, (Boston, USA), 2012.

[212] J. G. Bueno, M. G. Fierro, L. M. Lorente, and C. Balaguer, "Facial Emotion Recognition and Adaptative Postural Reaction by a Humanoid based on Neural Evolution," *International Journal of Advanced Computer Science*, vol. 3, no. 10, 2013.

[213] M. G. Fierro, J. G. Bueno, C. Balaguer, and L. Moreno, "A Complete 3D Perception and Path Planning Architecture for a Humanoid," in *Robocity2030 11th Workshop: Social Robots*, pp. 98–114, 2013.

[214] D. Álvarez, A. Lumbier, J. V. Gómez, S. Garrido, and L. Moreno, "Precision Grasp Planning Based on Fast Marching Square," in *21st Mediterranean Conference on Control and Automation*, 2013.

[215] J. V. Gómez, D. Álvarez, S. Garrido, and L. Moreno, "Kinesthetic Teaching via Fast Marching Square," in *International Conference on Intelligent Robots and Systems (IROS '12)*, pp. 1305–1310, IEEE, Oct. 2012.

[216] C. Arismendi, D. Álvarez, S. Garrido, and L. Moreno, "Adaptive evolving strategy for dextrous robotic manipulation," *Evolving Systems*, June 2013.

[217] "The largest 3D sensing development framework and community," in *http://www.openni.org*, 2013.

[218] A. Sahbani, S. El-Khoury, and P. Bidaud, "An Overview of 3D Object Grasp Synthesis Algorithms," *Robotics and Autonomous Systems*, vol. 60, pp. 326–336, Mar. 2012.

[219] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Dec. 1959.

[220] D. Adalsteinsson and J. A. Sethian, "A Fast Level Set Method for Propagating Interfaces," *Journal of Computational Physics*, vol. 118, no. 3, pp. 269–277, 1994.

[221] J. A. Sethian, "A Fast Marching level Set Method for Monotonically Advancing Fronts," in *National Academy of Sciences*, vol. 93, pp. 1591–1595, Feb. 1996.

[222] S. Osher and C. Shu, "High-Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations," *SIAM Journal on Numerical Analysis*, vol. 28, pp. 907–922, Aug. 1991.

[223] S. Garrido, L. Moreno, M. Abderrahim, and D. Blanco Rojas, "A Real-time Sensor-based Feedback Controller for Mobile Robots," *International Journal of Robotics and Automation*, vol. 24, no. n. 1, pp. 48–65, 2009.

[224] L. Yatziv, A. Bartesaghi, and G. Sapiro, "O(N) implementation of the fast marching algorithm." 2006.

[225] H. Kawasaki, T. Komats;, M. Suda, and K. Uchiyama, "Development of an Anthropomorphic Robot Hand Driven by Built-in Servo-motors," in *Proceedings of the 3rd Int. Conf. On ICAM*, pp. 215–220, 1998.

[226] S. Garrido, L. Moreno, and P. Lima, "Robot Formation Motion Planning using Fast Marching," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 675–683, 2011.

[227] J. V. Gómez, A. Lumbier, S. Garrido, and L. Moreno, "Planning robot formations with fast marching square including uncertainty conditions," *Robotics and Autonomous Systems*, vol. 61, no. 2, pp. 137–152, 2013.

[228] E. Kreyszig, *Differential Geometry*. New York, NY: Dover Publications, 1991.

[229] A. Miller and P. Allen, "GraspIt!," *IEEE Robotics & Automation Magazine*, vol. 11, pp. 110–122, Dec. 2004.

[230] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *International Conference on Robotics and Automation (ICRA '03)*, vol. 2, pp. 1824–1829, IEEE, 2003.

[231] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environments,” in *International Conference on Intelligent Robots and Systems (IROS '09)*, pp. 1–6, IEEE, Oct. 2009.

[232] A. Miller and H. Christensen, “Implementation of multi-rigid-body Dynamics within a Robotic Grasping Simulator,” in *International Conference on Robotics and Automation (ICRA '03)*, vol. 2, pp. 2262–2268, IEEE, 2003.

[233] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Adaptative Scheme for Global Optimization Over Continuous Spaces,” tech. rep., TR-95-012, 1995.

[234] R. Hooke and T. A. Jeeves, “Direct Search Solution of Numerical and Statistical Problems,” *Journal of the ACM*, vol. 8, pp. 212–229, Apr. 1961.

[235] M. Molga and C. Smutnicki, “Test functions for optimization needs,” tech. rep., Bioinformatics Laboratory. University of Amsterdam., 2005.

[236] F. Topsoe, “Some inequalities for information divergence and related measures of discrimination,” *IEEE Transactions on Information Theory*, vol. 46, pp. 1602–1609, July 2000.

[237] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. 2012.

[238] P. Kafka, F. Österreicher, and I. Vincze, “On powers of f-divergences defining a distance,” *Studia Sci*, vol. 26, pp. 415–422, 1991.