Robotic navigation based on servo task and sensory memory

Eder Rodriguez Martinez

Thèse de Doctorat

Mention ……………. Sciences pour l'Ingénieur
Spécialité ……………. Vision par Ordinateur

présentée à ***l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)***

de l'Université de Picardie Jules Verne

par

Eder Alejandro RODRIGUEZ MARTINEZ

pour obtenir le grade de Docteur de l'Université de Picardie Jules Verne

Navigation Robotique Basée Tâches d'Asservissement et Mémoire Sensorielle

Soutenue le 9 Juillet 2020 de après avis des rapporteurs, devant le jury d'examen :

M. David FILLIAT, Professeur, ENSTA Paris	**Rapporteur**
M. Youcef MEZOUAR, Professeur, Institut Pascal	**Rapporteur**
M. Noury BOURAQADI, Professeur, IMT Lille Douai	**Examinateur**
M. Rogelio LOZANO, Directeur en Recherche, UTC	**Président du jury**
M. Andrew I. COMPORT, Chargé des recherches	**Examinateur**
M. Guillaume CARON, Maître de Conférences, UPJV	**Examinateur**
M. Claude PÉGARD, Professeur, UPJV	**Directeur de thèse**
M. David LARA-ALABAZARES, Professeur Associé	**Co-encadrant**

Page left intentionally blank.

Preamble

Author – This thesis proposes a solution to the *sensor-based navigation problem* for the cases where:

- the scene is static,
- a model of it is provided,
- no localization method is available.

Examples of these cases are denied GPS and indoor navigation.

Reader – Wait, what do you mean by the "sensor-based navigation problem"?

Author – This problem consists into controlling the navigator, from a start position to a final one, using an onboard sensor to navigate.

Reader – As easy as that?

Author – Well, if the start and final pose are "close" enough, the navigation can be addressed as a single *servo problem.* Otherwise, you can apply a *topological approach* to it.

Reader – I know this problem, its aim is to design a controller which makes the ouput of the system converge to a reference signal.

Author – That is right! Within the sensor-based navigation context, the output corresponds to the measurements of the onboard sensor and the reference signal to the final pose measurement.

Remerciements

Je tiens à exprimer toute ma reconnaissance à mes encadrants Guillaume CARON, Claude PÉGARD et David LARA-ALABAZARES. Je les remercie de m'avoir encadré, orienté, aidé et conseillé. Mes études au laboratoire ont été possibles grâce au conseil mexicain pour la science et la technologie (CONACYT).

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, Fidel Alejandro RODRIGUEZ SALAS et Beatriz MARTINEZ SIERRA, qui ont toujours été là pour moi. Je remercie mon frère César Alejandro RODRIGUEZ MARTINEZ pour m'encourager. Enfin, je remercie mes amis Tania, Samantha, Aya, Farouk, Nathan, Anahïs, Alex et aux groupes Basuras, Satiga et Discours Horizontaux qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Contents

List of Figures

List of Tables

Part I

Introduction and Foundations

Chapter 1

General Introduction

This thesis presents a general framework of a *goal-oriented navigation system* (GONS) and two Vision-Based Navigation Systems (VBNS). The first of them is an instantiation of the framework and the other was inspired from the Visual Teach & Repeat technique (VT&R). The main characteristic that features the framework and the systems is their autonomy to generate a sensory or virtual memory. This memory serves as reference during the navigation.

The framework, named Servo Navigation Architecture (SNA), belongs to a specific classification of robotic navigation systems which *plans a path to reach the goal*,i.e.

GONS. In order to plan the path, the systems that belong to this classification require a previous knowledge of the environment and, at least, the start and end poses [Stella et al. (1995)]. SNA was originated from the Photometric Navigation System (PNS) as a generalization of it; therefore, PNS follows the structure of SNA. Both of them are covered in Chapters 4 to Chapter 7 and were published in [Rodriguez et al. (2020)].

The study of robotics comprises an abundant variety of concepts, definitions, classifications and approaches that had been developed since the first patented robot in 1961 [Matica and Kovendi (2011)] and keeps evolving until nowadays. Therefore, the purpose of the *General Introduction* is to provide a solid background of the topics that encompass the general framework. The latter is situated at the end of a succession of topics illustrated in Fig. 1.1.

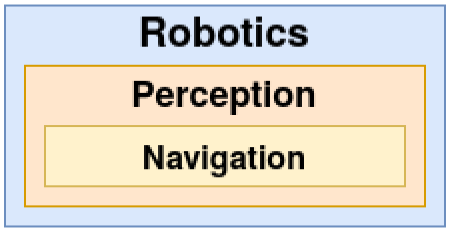


Figure 1.1: Robotics, perception and navigation.

PNS and the Photometric Teach & Repeat (PT&R) are considered as model-based approach systems since they rely on a preobtained model of the scene. Nevertheless, they perform a memory-based navigation as the appearance-based system do. Fig. 1.2 displays the classification of PT&R and PNS.

The next part of the Chapter briefly addresses the topics related to robotics, perception and navigation, and the last part is dedicated to the structure of the thesis.

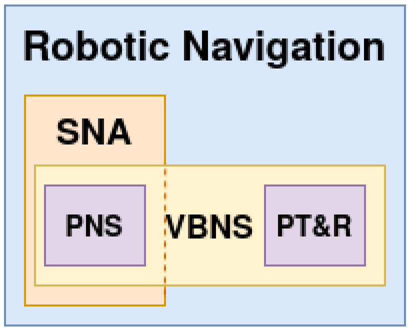


Figure 1.2: Hierarchical classification of SNA and PNS.

1.1 Introduction to Robotics

In 1921, the term *robot* (Figure 1.3) was employed for the first time to refer to artificial people or androids in the science fiction play ”Rossum’s Universal Robots” by Karel Ĉapek.

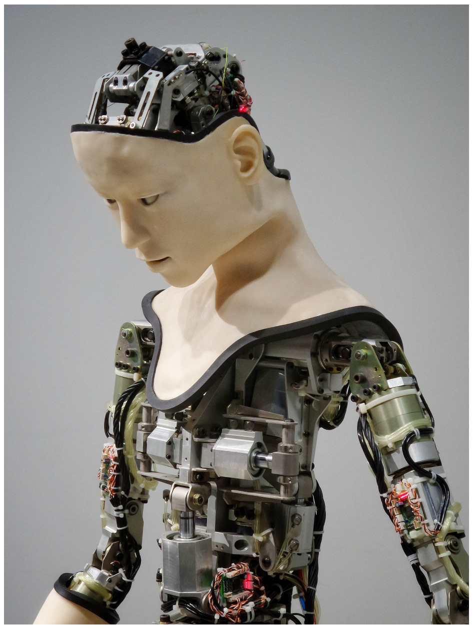


Figure 1.3: Humanoid robot.

Then, Isaac Asimov’s wrote many books composed by short stories about the robot-human interaction from 1950 to 1985. Indeed, this term has different meanings depending on the context and the discipline where it is applied. In science and for the rest of the thesis, the robot is defined as follows [Corke (2017)]:

”A goal oriented machine that can *sense*, *plan* and *act*”.

Hence, a robot is an *agent* capable of sensing its environment and to use the information retrieved by sensors, together with a predefined goal, to plan the action for which the robot was built for. In the case of SNA and PNS, for instance, the action is to navigate. Some of the most important events regarding to the history of robots are recalled next.

The first patent of robot was issued by C. Devol in 1961. The patent consists on a mechanical arm with a gripper which encoded the motion as a magnetic pattern and stores it on a rotating drum. In the same year, the company Unimation installed the first arm-type robot in the industry.

Then, in the 1990s the *field robotics*, *human augmentation* and *service robotics* (as the robot that appears in Fig. 1.4) expanded the research in *robotics*. The latter new science, defined in the 1980s, studies the *intelligent connection* between *perception* and *action*. According with this definition, the action is related to *locomotion* and *manipulation*; the perception is the interpretation of the information acquired by the *sensors* and their designs; the intelligent connection is related to *programming*, *planning* and *control* processes that exploit *models* of the environment and of the robot itself [Siciliano and Khatib (2016)].

Figure 1.4: Vacuum robot for civil purposes.

The following sections relate the fundamentals of robotics in order to settle the main contribution of the thesis.

1.2 Perception

The robotic perception consists on fitting sensor data to a mathematical model that includes the robot, the environment, the sensor and the interaction between them. Along with some estimation process, the perception relies on those sensors that recover the state of the robot itself (*propriocetives*) and on those that recover the state of the external environment (*exteroceptives*) in order to interpret the physical quantities [Siciliano and Khatib (2016)].

Depending on how the sensors interact with the environment, they are classified in two categories: *active* and *passive*. Active sensors emit energy to the environment and measure the response based on a model of it. Passive sensors measure energy “naturally” emitted by the environment. Although in some cases active sensors are more robust to perceive the environment than the passive ones (since they exert some control over the measured signal) some issues as the absorption, scattering and interference can reduce the performance or the accuracy of them. Fig: 1.5 shows an image of an Unmanned Aerial Robot (UAV) equipped with a camera, which is a passive sensor.

Figure 1.5: Conventional camera mounted on an UAV.

1.3 Environment Representation

Appart from control, the sensor information is used to estimate the state of the robot and the environment. Both, the estimation method and the representation of the environment are chosen according to the application. A common model of the environment maps the sensory data, from a local reference, to a location in the world or robot reference frame. These models are categorized into four general classes [Siciliano and Khatib (2016)]:

- **Raw sensor data models**: This class of model is used in feedback control [Astrom and Wittenmark (1995)] where the same sensory data is also used on the final application (e.g. following a trajectory, mapping). This model is the easiest to construct when the sensor modality belongs to a single modality. One example of these maps is the one built from laser range scans in [Gutmann and Schlegel (1996)].

- **Grid-based models**: The world is represented by a number of cells in such model. Each cell contains a feature of the environment (e.g. temperature, force, distribution). The tessellation can be either uniform or tree-based. This last method is well suited for homogeneous and large-scale datasets. This class of model is often used since a while in mobile robotics [Elfes (1987) & Elfes (1989)] and medical imaging [Stytz et al. (1991)].

- **Feature-based models**: This class is usually used when the interesting features (e.g. points, lines, planes, curves) are extracted, from the sensory data, mapped into the space of the model using an estimation process and stored in the memory. In [Dong et al. (2007)], a model built from laser measurements is implemented using a Simultaneous Localization And Mapping technique to describe the static part of a dynamic environment. These laser measurements are obtained from Light Detection And Ranging technology [Priestnall et al. (2000)]. Digital Elevation Models, implemented in [Zhang et al. (2011)], and Digital Surface Models are built from Light Detection And Ranging technology, and used for navigation. A Dense Surface Model is built in [Costante et al. (2018)] from an UAV while exploring and navigating through the scene. The difference between Digital Evaluation Models and Dense Surface Models is that the former model only includes earth relieves whereas the latter one may include artificial objects, e.g. buildings and power lines, or vegetation.

- **Symbolic or graphical models**: This model is adapted when a recognition process (e.g. object, place, landmark, road) of the environment is available. In this class, the environment is often represented by a graph where the recognized features are associated with the edges of a given graph according to their semantic. Examples of these representations are semantic 3D map [Mahe et al. (2019)] and polymaps [Dichtl et al. (2019)]. They are built using Simultaneous Localization And Mapping aimed to autonomous navigation.

1.4 Navigation and Path-Planning

Here, we understand by *robotic navigation* as the aspect of cognition related to robot robust mobility. It combines the sense of perception, some knowledge on the environment and a set of goal positions to reliably control the robot during a mission that involves displacement.

As stated in [Siegwart et al. (2011)], two main components are required for robotic navigation: *path planning* and *obstacle avoidance*.

The remainder of this Section delves on the definition and structure of path planning as it is the main competence exploited by the general framework of this thesis.

Path planning is a strategic problem-solving competence which objective is to find a set of waypoints, relying on the knowledge of the environment, so the robot can reach a set of goal locations. While planning is a long-term competence, *reacting* is the complementary competence which objective is to control the robot over the environment despite the dissimilarities of the *ideal* planned path and the *feasible* one.

[Siegwart et al. (2011)] formulates the navigation problem and the path planning as follows:

- Let R be a robot and $\boldsymbol{\xi}_f = [X_f, Y_f, Z_f, \Theta_f, \Phi_f, \Psi_f]^\top$ be the final pose that R has to reach before $n \in \mathbb{R}$ timesteps. Then, R has completed the navigation mission if the **temporal constraint** $loc_g(R) = \boldsymbol{\xi}_f; (g \leq n)$ is satisfied, where $loc_g(\cdot)$ is the location function at timestep g and n is the maximum number of timesteps allowed.
- Planning a path $\mathcal{P}$ consists in transforming the start belief state b_s into the final belief state b_f, by taking into account the initial world state W_i and by perceiving b_s at $loc_s(R)$.

In a practical situation, any belief state b_j might not be consistent with its location ($loc_j(R)$) or W_i may neither be accurate enough nor consistent with a dynamic world. Thus, the role of *reacting* results essential for the success of navigation as it is responsible to adapt the behavior of the robot despite unexpected circumstances.

1.5 Goal - Oriented Navigation Systems

The main goal of this subsection is not to delve into the vast literature of mobile robotics and goal-oriented navigation, but rather to give a general overview of their diversity and evolution through the time. A secondary goal is to compare SNA with the current literature.

The research on mobile robotics began in the late sixties at the Stanford Research Institute with the development of SHAKEY [Rosen and Nilsson (1967) and Nilsson and Wahlstrom (1968)]. The objective of this project was to study the interaction of the robot with a complex environment.

In 1983 the project HILARE performed the first robotic navigation and path planning, to our knowledge [Giralt et al. (1990)]. The wheeled robot was equipped with an ultrasonic perception module, a vision system and positioning referencing system. The latter was based on infrared sensor and fixed beacons.

The work in [Ryu and Yang (1999)] presents a complete architecture for mapping and navigation through indoor environments. Such architecture is composed by three independent subsystems: mapping (human-aided), perception and control. The map represents the environment using a metric-topological approach and it serves to plan a path and to estimate the pose from it. The perception subsystem performs obstacle avoidance and uses dead-reckoning and distinctive landmarks to localize itself in the map. The control subsystem uses a behavior-based approach to determine the actions of the robot located at a certain position in the map.

One common and challenging tasks, when a complete knowledge of the environment is available, is the path planning. A common strategy to navigate through the environment that is not exactly as the pre-computed model is to use a global (offline) path planner and then a local (online) path planner. The global planner searches for the optimized (to some criteria) path over the pre-computed model without the presence of the robot. However, sometimes the accuracy of the global planner is not sufficient enough or the environment had changed since the computation of the global path. In such situations, a local planner will modify the path from the current location so the robot reaches the goal. The following methods present at least one of these planners. A clear example of this strategy is presented in Huh et al. (2002) where the space is tessellated by the Cell Decomposition method and the Dijkstra's algorithm finds a the shortest path for the global planner. Then, the local planner uses Potential Field method to reach the goal and a Fuzzy controller to avoid obstacles.

The global planner presented in [Li and Duan (2012)] is designed for an UAV application. The planning stage consists in finding a 2D path that reaches the goal while avoiding static convex areas which may put in danger the vehicle. For this mean, they propose to modify the Gravitational Search Algorithm [Rashedi et al. (2009)] by adding the memory and social information of Particle Swarm Optimization [Kennedy (2010)], a dynamic weight function and a *survival of the fittest* strategy that are usually implemented on the Differential Evolution algorithms [Storn and Price (1997)].

Instead of using a heuristic method, as the planner aforementioned, a deterministic approach is preferred in the global planner presented in [Fu et al. (2019)]. The latter is designed to overcome the difficulties that an Autonomous Underwater Vehicle faces as floating obstacles and uneven surface, for example. They propose an improved version of the Rapidly-exploring Random Tree [LaValle and M (1998)]. This version uses a Goal-Biased Gaussian Distribution Sampling Strategy which expands the tree in the direction of the goal and prevents to expand it on undesirable directions.

Another deterministic global planner is presented in [Zagradjanin et al. (2019)] for a multi-robot navigation system that navigates through an environment with obstacles and crowds. The global planner applies D* Lite [Koenig and Likhachev (2002)] to quickly find a path that avoids obstacles. Later, the local planner modifies the path using the Multi-Criteria Decision Making and the Full Consistency Method [Pamučar et al. (2018)] so the robot avoids the crowds. A recent contribution to space tessellation for global path planning is proposed in [Wahdan and Elgazzar (2019)]. Using a method based on *fences* and homotopy classes, a 2D environment with obstacles is represented in a form of a graph as the path planning proceeds applying a graph-search algorithm.

In most GONS, including the systems mentioned above, the robot is the *protagonist* of the navigation. This fact makes sense as it is itself the one that reaches the goal from a start location. However, nothing prevents that the environment itself plays the most important role in path planning. Indeed, cognitive navigation is conceived like this for the case of rats [O'Keefe and Dostrovsky (1971)]. By means of reinforcement learning and mutli-scale representation, an action (e.g. heading direction or forward motion) is computed depending on the location of the bot, i.e. the agent in the simulation, over the virtual environment in [Llofriu et al. (2015)]. In this case, not only the environment is modeled but also the robot is. Once the strategy is satisfactory on the simulation, the same multi-scale representation is applied to the real environment causing the wheeled robot to follow a similar path.

So far, some of the most relevants GONS using global path planners had been covered. The main problem to solve is the path planning regardless of the sensor, in most cases. The next section covers two main approaches that VBNS use to tackle the navigation problem using a visual sensor to navigate.

1.6 Vision-Based Navigation Systems

Vision-based autonomous navigation is the process of making the robot able to reach an end pose, which is out of sensing range from an inital pose, using vision only. Some representative works are described in [Rahmani et al. (2015), DeSouza and Kak (2002), Bonin-Font et al. (2008) & Chatterjee et al. (2012)].

A VBNS is able to plan a path and navigate through the scene. Therefore, the representation of the environment plays an important role in this type of system.

Depending on how a VBNS describes the environment, it is classified as a *model-based* or *appearance-based* navigation system [DeSouza and Kak (2002)]. The following paragraphs briefly recall some of the most relevant works regarding these classifications.

1.6.1 Model-Based Navigation Systems

In the model-based approach, the environment is described by a model. Generally speaking, this type of system compares the global model (offline) with a local measurement (online) either to localize itself or to navigate. On the one hand, a short path can be found by if a complete knowledge of the environment is available *a priori* [Chatterjee et al. (2012)]. On the other one, the succes of such approach depends on the accuracy of the model [Šegvić et al. (2007)].

A navigation system that integrates the data of visual and inertial sensors is presented in [Jones and Soatto (2011)]. The novelty of this system is that it performs ego-motion estimation, localization and mapping from uncalibrated sensors. The 3D model is built from the Simultaneous Localization And Mapping [Bresson et al. (2017)] technique that tracks point features whose descriptor is the average of the gradient over the time [Lee and Soatto (2010)]. The system presented in [Costante et al. (2018)] updates a map during the exploration stage. Its planner leverages on the geometric and photometric information of the map in order to propose a path that reduces the pose uncertainty while heading towards the goal. When the map is obtained by other means, the exploration stage can be avoided. Such is the case in [Kosaka and Pan (1995)] and [Gasteratos et al. (2002)], where the system compares the visual information acquired during navigation, with a Computer-Aided Design (CAD) model.

1.6.2 Appearance-Based Navigation Systems

The second type is the appearance-based navigation approach which describes the environment in a *topological approach.* The environment is represented by a graph, where the nodes correspond to a measurement of the environment and the edges to the navigability between the nodes. On the one hand, this approach is convenient when a metric model of the environment is unavailable. On the other one, the content of the topological map is less precise than metric models [Tomatis (2001)].

Two main strategies are used to create such a graph. With the first strategy, an image database completely describes the environment. Then a group of images is selected to form a connected graph [Goedemé et al. (2004), Fontanelli et al. (2009), Remazeilles and Chaumette (2007) & Mezouar et al. (2002)]. A localization can be performed relative to the closest vertex on the graph [Goedemé et al. (2007)]. This method also allows path planning over the graph. With the second strategy, Visual Teach & Repeat [Furgale and Barfoot (2010)], an expert proposes a navigable path and operates a robot through it [Courbon et al. (2008, 2010); Nguyen et al. (2014); Santosh et al. (2008); Diosi et al. (2011)]. The work presented in [Warren et al. (2018)] adapts VT&R technique [Furgale and Barfoot (2010)] to an autonomous *safety-return* aerial navigation system. In the teaching stage, the system performs Visual Odometry by inserting visual observations into a map of relative poses and scene structure. By comparing the map with the environment, the UAV localizes itself and reproduces the learnt trajectory backward. Meanwhile, a sequence of images is collected from which a group is selected to compose the visual memory of the path. Then, at the navigation stage, the robot trajectory is controlled through the key images on the sensor space, by comparing each key image with current visual measurements.

1.7 Problem Statement and Proposed Solutions

From the state-of-the-art of robotic navigation and model-based navigation, it is possible to spot two *main problems*:

- *Localization:* Some GONS require to be localized at each time the action is computed (e.g. Fast-Simultaneous Localization And Mapping [Montemerlo et al. (2002)] for the latter case) while the autonomous navigation is being carried out. If the localization is inaccurate, the autonomous navigation is likely to fail.
- *Model's accuracy:* As stated in [Segvic et al. (2007)], the success of the model-based navigation approach depends on the accuracy of the model. The problem persists in some geometric model and may cause error propagation when estimating the pose while navigating [Remazeilles and Chaumette (2007)]. Moreover, some of these models were not generated explicity for navigation; such is the case of the CAD model used in [Gasteratos et al. (2002)] where some zone lack of texture and may lead to navigation failure.

By discreting the representation of the scene into the navigation space, the aforementioned error propagation is avoided by the appearance-based navigation systems.

Moreover, the navigation space contains texture since it is intended for autonomous vision-based navigation; thus, it is suitable for navigation. Nevertheless, such representation is usually generated *in situ* and requires the presence of the robot, either by the exploration or teaching stages. Two *main problems* are related to these stages:

- *Exploration:* In this stage, the robot is controlled through the scene in onder to collect sensor measurements. Since a complete representation of the scene is available from this stage, some of the measurements can be autonomously selected for a given task. Nevertheless, the exploration might be time expensive depending of the size of the scene.

- *Teaching:* In order to avoid a complete exploration of the scene, the navigation space can be generated from a prior navigation over the scene. The trajectory of this navigation is proposed by a human, as in the Visual Teach & Repeat techniques: therefore, this stage is non-autonomous. Moreover, the navigability of the navigation space depends on the operator skills [Courbon et al. (2010)] and some paths may remain unexplored [Raj et al. (2016)].

This thesis addresses the aforementioned problems by combining the model-based and appearance-based approaches. The *proposed solutions*, which are related with the contributions, are depicted next:

- In all the contributions (Photometric Teach & Repeat, Photometric Navigation System and Servo Navigation Architecture), only one coarse localization is required to initialize the autonomous navigation, i.e. to place the robot near to the start pose. This step can be avoid by tessellating the navigation space into a set of poses to be attained. Then, the transition from one pose to the next one, analogously to localization, is infered from the convergence of the algorithm that governs the motion of the robot during the autonomous navigation.

- In PT&R and PNS, the scene is represented in the sensor space of the navigation from which their visual memory is generated. Therefore, the comparison between the latter and the onboard images is efficient since no texture is lost in this process.

- In all the contributions, the exploration can be avoided by defining the goal poses in the model of the environment, if the latter is available. In PT&R and PNS, a model of the scene can be textured from a single snapshot *in situ* or by an aerial image.

- In PT&R, the teaching stage is carried out using the model of the scene rather than *in situ*. Moreover, the navigability of the visual memory, generated from this stage, relies on the content of the model rather than on the operator skills. Morever, in PNS and SNA, the teaching stage is replaced by a path-planning that exploits the navigation space in more than one dimension. In other words, it is not restricted to straight paths.

To summarize, the next Section presents the contributions that this thesis offers to the fiels of robotic navigation and vision-based navigation.

1.8 Contributions

This thesis contributes with the fields of robotic navigation and vision-based navigation by considering perception within the context of path-planning. Moreover, the sensory memories can be generated by leveraging on a preobtained model of the scene. The three contributions, PT&R, PNS and SNA, and their main characteristics are briefly presented next.

PT&R is the first attempt of VBNS that combines the model-based and appearance-based approaches. The system is inspired on the VT&R techniques but carrying out the teaching stage using the model of the scene rather than moving the robot over the scene. Its main characteristics are:

- The navigation space is defined in *straight line* and so the visual memory.
- It is implemented on an *aerial robot* equipped with a *camera.*

PNS outperforms PT&R by exploiting an autonomous path-planning over a tessellation of the navigation space in more than one dimension. Moreover, the length of the visual path, related to such memory, is minimal while being navigable. The system features the following characteristics:

- The navigation space is tessellated in *one, two* or *three dimensions*; thus, the autonomous navigation is not restricted to the straight line.
- It is implemented on a *robotic arm* equipped with a *camera.*

SNA is a generalization of PNS as framework for robotic navigation, where:

- The navigation space is tessellated in *one, two* or *three dimensions.*

- It is suitable to be implemented in various robots, sensors, environments and their corresponding models.
- The autonomous navigation is ensured for *1* to *6* DoF, analogously to PNS.

The contributions are formally depicted in their corresponding Chapters in Part II.

1.9 Conclusions

This Chapter begins by covering the fundamental concepts of robotic navigation and some of the most important works on the state-of-the-art. Then, the vision-based navigation systems are presented in order to compare the model-based and the appearance-based approaches. The comparison leads to spot four of the main problems that exist in the current literature in vision-based navigation. Lastly, four solutions are proposed for these main problems as a result of combining both approaches. The solutions presented in Chapter 1.7 are the basis of the contributions.

The next Chapter focuses on the mathematical concepts required to develop PT&R, PNS and SNA.

Chapter 2

Theoretical Framework

This Chapter presents the topics related to the development of the Photometric Teach & Repeat (PT&R), the Servo Navigation Architecture (SNA) and, by consequence, the Photometric Navigation System (PNS).

The first Section (Section 2.1) covers the mathematical representation of the pose and velocity, and the generation of an image from a perspective camera (PT&R and PNS). The second section (Section 2.2) includes the topics related with the motion of the robot and the control laws. The third Section (Section 2.3) is composed by the topics related to the metric-topological approach used for navigation and path-planning. The last Section (Section 2.4) concludes the Chapter by providing a recap of the chosen mathematical concepts implemented on PT&R, SNA and PNS.

2.1 Modeling

In this thesis, the motion of the robot is considered as a **rigid object**, i.e. which constituent points maintain a constant relative position. Therefore, the notions of pose and velocity are fundamentals for the architecture. Moreover, the sensor function of PT& and PNS, i.e. the perspective camera projection, is included at the end of this Section. These topics are largely inspired from [Corke (2017)].

2.1.1 Pose and Velocity Representations

The pose can be represented in the 3D space by the vector $\boldsymbol{\xi} = [\mathbf{X}^\top, \boldsymbol{\Theta}^\top]^\top \in \mathbb{R}^6$, where $\mathbf{X} = [X, Y, Z]^\top \in \mathbb{R}^3$ is its location and $\boldsymbol{\Theta} = [\Theta, \Phi, \Psi]^\top \in \mathbb{R}^3$ its orientation.

Similarly, the velocity of a rigid object can be represented by $\dot{\boldsymbol{\xi}} = [\mathbf{v}^\top, \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6$, where $\mathbf{v} = [v_x, v_y, v_z]^\top \in \mathbb{R}^3$ and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^3$ are the linear and angular velocities, respectively.

Consider two frames ${}^A\mathfrak{F}$, ${}^B\mathfrak{F}$ and a pose ${}^A\boldsymbol{\xi}$ expressed relative to ${}^A\mathfrak{F}$; then, the pose relative to ${}^B\mathfrak{F}$ can be expressed by:

$$ {}^B\boldsymbol{\xi} = {}^B\mathbf{M}_A {}^A\boldsymbol{\xi}, \tag{2.1} $$

where

$$ {}^B\mathbf{M}_A = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.2} $$

is the homogeneous transformation matrix from ${}^A\mathfrak{F}$ to ${}^B\mathfrak{F}$, $\mathbf{t} = [t_x, t_y, t_z]^\top \in \mathbb{R}^3$ is the translation vector and $\mathbf{R} \in SO(3) \subset \mathbb{R}^{3\times3}$ is the orthonormal rotation matrix computed from the Rodrigues' rotation formula:

$$ \mathbf{R} = \mathbf{I}_{3\times3} + \sin\alpha \mathbf{S}(\mathbf{r}) + (1 - \cos\alpha)(\mathbf{r}\mathbf{r}^\top - \mathbf{I}_{3\times3}), \tag{2.3} $$

where $\mathbf{r} \in \mathbb{R}^3$ is the vector around which the rotation occurs, α is the amount of rotation and $\mathbf{S}(\mathbf{r})$ is the skew matrix of $\mathbf{r}$

$$ \mathbf{S}(\mathbf{r}) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{2.4} $$

Analogously, the velocity of a rigid object ${}^{A}\dot{\boldsymbol{\xi}}$ expressed relative to ${}^{A}\mathfrak{F}$ can be expressed relative to ${}^{B}\mathfrak{F}$ by:

$$ {}^{B}\dot{\boldsymbol{\xi}} = {}^{B}\mathbf{V}_{A}{}^{A}\dot{\boldsymbol{\xi}}, \tag{2.5} $$

where

$$ {}^{B}\mathbf{V}_{A} = \mathbf{V}({}^{A}\mathbf{M}_{B}) = \begin{bmatrix} \mathbf{R}^{\top} & (\mathbf{S}(\mathbf{t})\mathbf{R})^{\top} \\ \mathbf{0} & \mathbf{R}^{\top} \end{bmatrix}, \tag{2.6} $$

is the twist transformation matrix and

$$ \mathbf{S}(\mathbf{t}) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{2.7} $$

is the skew symmetric matrix of $\mathbf{t}$.

2.1.2 Jacobian

In robotics, it is common to relate the velocity of an object, perceived by the onboard sensor, with the velocity of the robot. Let $\dot{\boldsymbol{\xi}}$ be the velocity of the robot and $\dot{\mathbf{q}} \in \mathbb{R}^N$ the velocity of the object in the sensor space. Then, their relation is expressed as:

$$ \dot{\boldsymbol{\xi}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.8} $$

where $\mathbf{J}(\mathbf{q})$ is matrix of $6 \times N$ known as the geometric Jacobian of $\mathbf{q}$.

Formally speaking, the Jacobian matrix $\mathbf{J}$ is the matrix equivalent to the derivative of a vector-valued function with respect to another vector. If $\mathbf{y} = F(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$, $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{y} = [y_0, y_1, \ldots, y_{m-1}]$, then:

$$ \mathbf{J} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_0}{\partial x_0} & \cdots & \frac{\partial y_0}{\partial x_{n-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{m-1}}{\partial x_0} & \cdots & \frac{\partial y_{m-1}}{\partial x_{n-1}} \end{bmatrix}. \tag{2.9} $$

2.1.3 Perspective Camera Projection

Consider a world point located at ${}^{C}\mathbf{X} = [{}^{C}X, {}^{C}Y, {}^{C}Z]^{\top}$, expressed in the camera's frame ${}^{C}\mathfrak{F}$, that is projected at the pixel coordinates $\mathbf{x} = [u, v]^{\top}$, by the central perspective camera projection:

$$ \tilde{\mathbf{x}} = \mathbf{K}{}^{C}\tilde{\mathbf{X}}, \tag{2.10} $$

where $\tilde{\mathbf{x}} = [u', v', w']^\top$ is the homogeneous coordinate of $\mathbf{x}$ in pixel coordinates with $u' = uw'$ and $v' = vw'$, ${}^C\tilde{\mathbf{X}} = [{}^CX, {}^CY, {}^CZ, 1]^\top$ is ${}^C\mathbf{X}$ expressed in homogeneous coordinates and:

$$\mathbf{K} = \begin{bmatrix} f/\rho_w & 0 & u_0 & 0 \\ 0 & f/\rho_h & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.11}$$

is the matrix of the intrinsic parameters of the camera, f is the focal lenght, ρ_w and ρ_h are the pixel's width and height and $\mathbf{x}_0 = [u_0, v_0]^\top$ is the principal point in the image.

The projection can also be written in functional form as:

$$\mathbf{x} = \mathcal{P}(\mathbf{X}, \mathbf{K}, \mathbf{M}_C), \tag{2.12}$$

where $\mathbf{M}_C$ is the homogeneous matrix that relates ${}^C\mathfrak{F}$ with the reference frame of $\mathbf{X}$:

$$\tilde{\mathbf{X}} = \mathbf{M}_C {}^C\tilde{\mathbf{X}}. \tag{2.13}$$

2.2 Servo Task

The motion of the robot is generated by performing servo tasks. The Photometric Visual Servoing (PVS) [Collewet and Marchand (2011)] is an instantiation of the servo tasks when the vector of the pixel's intensity of an image is used as feature. The motion in PT&R and PNS is computed from PVS scheme whereas that of SNA is computed from the servo scheme. The concepts regarding the servo task, the optimization methods and PVS, are formally depicted hereafter.

The servo task (also called Sensor-Based Control in [Magassouba et al. (2018)]) is formalized in [Chaumette and Hutchinson (2008)] by considering a feature vector $\mathbf{s}$ as the input of a control loop system (Fig. 2.1). Being $\mathbf{u}$ the velocity input vector and $\dot{\mathbf{s}}$ the feature variation, their relation is expressed by the interaction matrix $\mathbf{L_s} \in \mathbb{R}^{p\times q}$, where p is the dimension of the feature vector $\mathbf{s}$ and q the degrees of freedom (DoF) of the control input, by:

$$\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{u}. \tag{2.14}$$

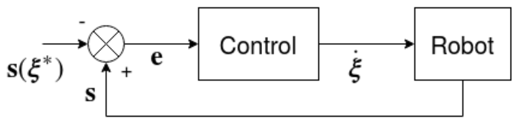


Figure 2.1: Control loop system representation of a servo task.

The goal of the control loop system is to minimize the cost $C(\boldsymbol{\xi})$, depending on the sensor pose $\boldsymbol{\xi}$:

$$C(\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{s}(\boldsymbol{\xi}) - \mathbf{s}^*\|, \tag{2.15}$$

and regulate the error $\mathbf{e}(\boldsymbol{\xi}) = \mathbf{s}(\boldsymbol{\xi}) - \mathbf{s}^*$ to zero, where $\mathbf{s}^* = \mathbf{s}(\boldsymbol{\xi}^*)$ is the desired feature vector, using the following optimization methods.

2.2.1 Optimization Methods

This subsection covers some of the most used control laws in the literature of computer vision and robotics. For more information, consult [Espiau et al. (1992), Collewet et al. (2008) and Malis (2004)].

2.2.1.1 Gradient Method

Being the direction of descent equal to the gradient of the cost function $\nabla C(\boldsymbol{\xi})$, the control law using the gradient method is:

$$\mathbf{u} = -\lambda \mathbf{L}_{\mathbf{s}}^{\top}\mathbf{e}, \tag{2.16}$$

where $\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \boldsymbol{\xi}}\dot{\boldsymbol{\xi}} = \mathbf{L}_{\mathbf{s}}\mathbf{u}$ and $\mathbf{L}_{\mathbf{s}}$ is the interaction matrix of $\mathbf{s}$.

2.2.1.2 Gauss-Newton

Being $\dot{\mathbf{e}}$ the time variation of $\mathbf{e}$ and $\lambda \in \mathbb{R}_+^*$ the gain, the control law becomes:

$$\mathbf{u} = -\lambda \mathbf{L}_{\mathbf{s}}^{+}\mathbf{e}, \tag{2.17}$$

where $\mathbf{L}_{\mathbf{s}}^{+} \in \mathbb{R}^{q \times p}$ is the Moore-Penrose of $\mathbf{J}_{\mathbf{s}}$ when the latter is of full rank pseudoinverse. However, if $\mathbf{L}_{\mathbf{s}}^{+}$ cannot be directly known, an approximation of it $(\widehat{\mathbf{L}}_{\mathbf{s}}^{+})$ is considered. Stability analysis is proven [Chaumette and Hutchinson (2008)] and [Slotine et al. (1991)] if:

$$\mathbf{L}_{\mathbf{s}}\widehat{\mathbf{L}}_{\mathbf{s}}^{+} > 0. \tag{2.18}$$

By assuming that $\boldsymbol{\xi}$ is close enough to $\boldsymbol{\xi}^*$, it is possible to replace $\widehat{\mathbf{L}}_{\mathbf{s}}^{+}$ for its equivalent with respect to $\mathbf{s}^*$ (so, $\widehat{\mathbf{L}}_{\mathbf{s}^*}^{+}$) in Eq. (2.17). This assumption, demonstrated in [Chaumette and Hutchinson (2006)], allows to compute $\widehat{\mathbf{L}}_{\mathbf{s}^*}^{+}$ once instead of computing $\widehat{\mathbf{L}}_{\mathbf{s}}^{+}$ each iteration.

Finally, the input velocity vector is computed from:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}_{\mathbf{s}^*}^{+}(\mathbf{s}(t) - \mathbf{s}(\boldsymbol{\xi}^*)). \tag{2.19}$$

2.2.1.3 Newton

This method is equivalent to Gauss-Newton's when $\boldsymbol{\xi}$ lies in the neighborhood of $\boldsymbol{\xi}^*$. However, the difference of this method with respect to Gauss-Newton's and the gradient method is the use of the second order Taylor series expansion for the approximation of $\nabla C(\boldsymbol{\xi})$ around $\boldsymbol{\xi}$ instead of the first order.

2.2.1.4 Levenberg-Marquardt

This method also considers a second order expansion of the cost function. Its control law is:

$$\mathbf{u} = -\lambda(\mathbf{H} + \mu \text{diag}(\mathbf{H}))^{-1}\mathbf{L}_{\mathbf{s}}^{\top}\mathbf{e}, \tag{2.20}$$

where $\mathbf{H} = \mathbf{L}_{\mathbf{s}}^{\top}\mathbf{L}_{\mathbf{s}}$ is an approximation of the Hessian matrix and $\mu \in \mathbb{R}_{+}^{*}$. When the parameter μ is very low and $\boldsymbol{\xi}$ lies in the neighborhood of $\boldsymbol{\xi}^*$, Eq. (2.20) behaves as Eq. (2.19).

2.2.2 Photometric Visual Servoing

This scheme considers the vector $\mathbf{I}(\boldsymbol{\xi})$ of image intensities $I(\mathbf{x})$, acquired at the pose $\boldsymbol{\xi}$, for all $\mathbf{x} = (x, y)$ belonging to the image domain:

$$\mathbf{I}(\boldsymbol{\xi}) = (\mathbf{I}_{1\bullet}, \mathbf{I}_{2\bullet}, \ldots, \mathbf{I}_{N\bullet})^{\top} \tag{2.21}$$

where $\mathbf{I}_{i\bullet} \in \mathbb{N}^{1\times M}$ is the i-th line of the image. Here, N is the width and M the height of the image.

The Photometric Visual Servoing is formulated as an optimization process where its goal is to minimize:

$$C(\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{I}(\boldsymbol{\xi}) - \mathbf{I}(\boldsymbol{\xi}^*)\|, \tag{2.22}$$

where $\boldsymbol{\xi}^*$ is the desired pose.

Using the Gauss-Newton optimization (Eq. (2.19)) method and assuming that $\boldsymbol{\xi}(0)$ is close to $\boldsymbol{\xi}^*$, the control law of PVS is:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}_{\mathbf{I}(\boldsymbol{\xi}^*)}^{+}(\mathbf{I}(\boldsymbol{\xi}) - \mathbf{I}(\boldsymbol{\xi}^*)), \tag{2.23}$$

where λ is a positive scalar, $\widehat{\mathbf{L}}^{+}_{\mathbf{I}(\boldsymbol{\xi}^*)}$ is the Moore-Penrose pseudoinverse of the approximation of the interaction matrix $\mathbf{L}_{\mathbf{I}(\boldsymbol{\xi}^*)} = -\nabla\mathbf{I}(\boldsymbol{\xi}^*)^\top\mathbf{L}_\mathbf{x}$ related to $\mathbf{I}(\boldsymbol{\xi}^*)$ when $\mathbf{L}_{\mathbf{I}(\boldsymbol{\xi}^*)}$ is of full rank, $\nabla\mathbf{I}(\boldsymbol{\xi}^*)$ is the image gradient of $\mathbf{I}(\boldsymbol{\xi}^*)$ at location $\mathbf{x}$ and:

$$\mathbf{L}_\mathbf{x} = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}, \tag{2.24}$$

is the interaction matrix that relates the variations of $\mathbf{x}$ to the velocity of the camera.

The interested reader may refer to [Collewet and Marchand (2011), Malis (2004) and Collewet et al. (2008)] for a more detailed mathematical development.

2.3 Graph-Based Navigation

Generally speaking, the sensory memory is generated by navigating on a grid built by tessellating the preobtained model. Then, the nodes of the grid are assigned to the poses where the tessellation occurs. Finally, the edges are connected and weighted so a graph-search algorithm can find a short path.

This section is composed by the specific topics related to the topological-metric approach considered in SNA.

2.3.1 Weighted Kinggraph-m of Poses

Graphs are employed in metric-topological approaches when their nodes are related to poses (**graph of poses**) or sensor measurements, for example. In some cases, e.g. robot navigation, the space is tessellated by a lattice graph [Chen (2014)]. When the lattice graph that tessellates the space is squared and its nodes are simply connected [Khuller et al. (2000)], i.e. as the movements allowed by a king over an infinite chessboard, the graph is called **kinggraph** [Chepoi et al. (2002)]. However, SNA requires simply connected graphs built from linear, squared and cubic lattice graphs. Therefore, this subsections expands the concept of kinggraph for 1, 2 and 3 dimensions.

A **kinggraph-m** $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $n \in \mathcal{N}$ is the set of nodes, $e \in \mathcal{E}$ is the set of edges and $|\mathcal{N}|$ is the number of nodes, is the generalization of a kinggraph in $m = 1, 2, 3$ dimensions. $\mathcal{G}$ is built from a simply connected linear, squared or cubic lattice graph, for $m = 1, 2, 3$ respectively.

A central node n_c is connected to its $3^m - 1$ nearest-neighbor nodes, where $m = 1, 2, 3$ is the dimension of the lattice graph, as shown in Fig. 2.2.

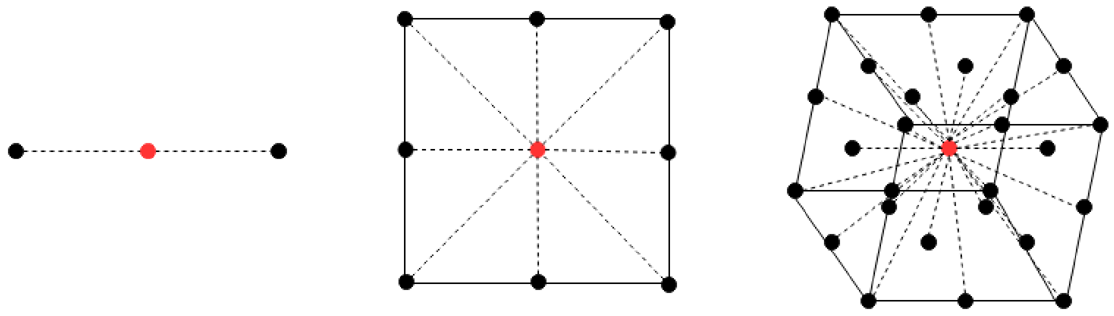


Figure 2.2: Kinggraphs-m, with $m = 1, 2, 3$, with their edges (dotted lines) that connect the central node (red dot) with others nodes (black dots).

Let a set of poses $\boldsymbol{\xi} \in \Xi$ be located at the nodes of a linear, squared or cubic lattice graph. Then, a **weighted kinggraph-m of poses** relates $n \in \mathcal{N}$ to $\boldsymbol{\xi} \in \Xi$ and its edges are weighted by some gamma function $\gamma(\cdot) : \mathbb{R} \mapsto \mathbb{R}$. Being $e_{p,q}$ the edge that connects n_p to n_q, the weight $\gamma(e_{p,q})$ is the Euclidean distance between the poses $\boldsymbol{\xi}_p = [\mathbf{X}_p^\top, \boldsymbol{\Theta}_p^\top]^\top$ and $\boldsymbol{\xi}_q = [\mathbf{X}_q^\top, \boldsymbol{\Theta}_q^\top]^\top$ related to n_p and n_q:

$$\gamma(e_{p,q}) = \|\mathbf{X}_q - \mathbf{X}_p\|. \tag{2.25}$$

2.3.2 Navigating on a 3D Grid

The last two subsections are largely inspired by [LaValle and M (2006)]. Suppose that a robot moves on a 3D grid in which each grid point has integer coordinates of the form (i, j, k). The robot takes discrete steps in one of 26 directions (up, down, left, right, forward, backward and diagonals) by one unit.

Let $s \in S$ be the **state space** expressed in the trio form (i, j, k), in which $i, j, k \in \mathbb{Z}$ and $a \in A(s)$ be the set of **actions** allowed at the state s with $A = \{(-1,-1,-1), (-1,-1,0), \ldots (0,0,-1)\} \cup \{(0,0,1), \ldots, (1,1,1)\}$. Then, the goal is to reach the final state s_f, from the start state s_s, where $s_s, s_f \in S^G$ belongs to the set of goal states $S^G \subset S$. This can be done using the **state transition equation** [LaValle and M (2006)] $f(s, a) = s + a$ once.

Navigating in such grid can be a more complicated problem if some action a is not allowed at given state s, which can be represented by an obstacle in the grid. This issue can be tackled by representing the grid as a kinggraph-m $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which $n \in \mathcal{N}$ is related with $s \in S$ and $e \in \mathcal{E}$ with an allowed action $a \in A(s)$. If $A(s)$

and S^G are known and $|\mathcal{N}|$ is finite, the graph-search algorithm A* [LaValle and M (2006)] can find a **short path** $n^* \in \mathcal{N}^*$.

2.3.3 A*

A* [Hart et al. (1968)] is a graph-search algorithm which is an extension of Dijsktra's algorithm [LaValle and M (2006)]. It incorporates a heuristic estimate of the cost to get to the final state s_f from the current state s that reduces the number of states to be explored compared with Dijsktra's algorithm.

Consider a weigthed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with $\gamma(e) \geq 0$ as the cost to apply to the action a. This cost $\gamma(e)$ can be written using the state-space representation as $f(s, a)$, indicating the cost to execute a at s. The total cost of a plan is the sum of the edge costs over the path, i.e. the node related to s_s (n_s) to the one related to s_f (n_f). The priority queue Q, which stores the states to visit (called **alive** states), is sorted according to a sum which involves the heuristic. This sum $(C^*(s') + \tilde{G}^*(s')$, where $C^*(s')$ is the optimal cost-to-come from s_s to s' and the optimal cost-to-go $\tilde{G}^*(s')$ is the estimate of the lowest possible value of the optimal cost-to-go $G^*(s')$ from s' to $s_f)$ guarantees that the algorithm finds an optimal path $\mathcal{N}^*$ if the cost of $\tilde{G}^*(s)$ is the lowest for all $s \in S$ [Fikes and Nilsson (1971) and Pearl (1984)]. $C^*(s)$ is defined as the lowest cost-to-come $C : S \to [0, \infty]$ and is computed by summing all the edge costs $\gamma(e)$ from all the vertices related to the states from s_s to s for a given path. C is computed incrementally during the execution of A* and $C^*(s_s)$ is set to 0 initially. Then, the algorithm continues by exploring the **unvisited** next states. Each time s' is generated $C(s') = C^*(s) + \gamma(e)$, where e is the edge that connects the vertices related with the states s and s', $C(s')$ is considered as the best cost-to-come known so far until C^* is determined. If s' already exists in Q, then it is possible that a newly discovered path from s_s to s' is more efficient. If this occurs, $C(s')$ must be lowered for s' and Q must be reordered accordingly. When C^* is determined, s is considered **dead** and gets removed from Q. This step is repeated until the only state in Q is s_f.

2.4 Conclusions

This Chapter begins by presenting the fundamentals of the representation of the pose of a rigid object and the perspective camera function. Moreover, these concepts are used to represent the pose of a virtual camera and to render virtual images.

Then, the Chapter covers the motion of the robot computed from the general scheme of the servo tasks and an instantion of it, i.e. PVS. The former one is applied to SNA and the latter to PT&R and PNS. Indeed, PNS is an instantiation of SNA as PVS is to the servo scheme. The servo scheme is used for navigation since it is fast to compute and it does not require localization. These two advantages are suitable for implementing PT&R on an aerial robot and to simulate positioning tasks in SNA context.

The Chapter ends on the graph-based approach, A*, from which SNA and PNS generate their sensory and virtual memories. Such approach is used for path-planning but considering perception within the definition of navigation. In few words, A* proposes short paths but the heuristic might be defined otherwise in order to minimize another cost. The kinggraph-m was created from a generalization of the kinggraph in order to simply connect linear, square or cubic lattice graphs that exploit the search space in 1, 2 or 3 dimensions. Each of these dimensions is related with one DoF of the location of the poses assigned to the nodes of the kinggraph-m. The simple connection of the lattice graph allows the straight line to be proposed by A*. Other algorithms, e.g. rapidly exploring random tree, particle swarm optimization or genetic algorithms, are not contemplated for path planning since the graph is non-dynamic and it has a maximum of 3 dimensions.

To sum up, this Chapter settles down the fundamental concepts used for PT&R, PNS and SNA, and prepares the reader to continue with Part II which is the core of the thesis.

Part II

Contributions

Chapter 3

Photometric Teach & Repeat

The Photometric Teach & Repeat (PT&R) is a Vision-Based Navigation System (VBNS) that combines the characteristics of the model-based and appearance-based approaches. To our knowledge, it is the first attempt to implement the Visual Teach & Repeat (VT&R) technique with a **visual memory** generation from a preobtained model.

This Chapter is structured as follows: First, Section 3.1 provides the scope of combining VT&R techniques that use visual memory with the model-based approach before presenting the foundations in robotic navigation based on servo tasks and sensory memory in Section 3.2. Then, PT&R is presented in Section 3.4. Later, Section 3.5 and Section 3.6 cover the implementation and the experimentation of the aforementioned system, respectively. Lastly, the Chapter concludes with Section 3.7.

3.1 Motivation

As a recall from Chapter 1.6.2, the appearance-based approaches do not rely on a preobtained model of the scene but on a topological representation of it. One advantage of topological models is that they do not have accumulative propagation error pose [Remazeilles and Chaumette (2007)], they are computationally more compact [Friedman et al. (2007)] and more efficient for path planning than metric models [Thrun (1998)]. However, their content is less precise than the latter models [Tomatis (2001)]. A well-known technique that leverages in the former type of representations is the VT&R according with [Yoder et al. (1996), Furgale and Barfoot (2010), Barfoot et al. (2012), Courbon et al. (2009a), Pfrunder et al. (2014) & Vandrish et al. (2012)]. Broadly speaking, this technique tackles the navigation problem by a split into two main *stages*. The first is called *teaching stage*, usually a human expert operator manually controls the robot over a trajectory while the onboard sensors collect measurements. These measurements are stored and sometimes preselected to be used as a reference to follow by the robot in the *repeating stage*. Hence, this technique is called *Teach & Repeat*. The disadvantage of this technique is that it assumes that the taught path is navigable [Courbon et al. (2010)] relying on the human skills [Blanc et al. (2005)]. Indeed, an inaccuracy in the path-following of these references may lead to navigation failure and even to repeat the earlier stage [Raj et al. (2016)]. Theaching a trajectory to follow is a cognitive process managed by the human expert operator in these techniques which makes the navigation not completely autonomous. Furthermore, this is a tedious task that must to be repeated in every scene to navigate and it is limited to the information collected during the first stage [Santosh et al. (2008)]. Thus, the navigation problem is formulated to exploit the advantages of leveraging on a preobtained model while autonomously selecting its minimal representation necessary to navigate. Furthermore, by defining the preobtained model as snapshot (also called instantaneous model [Salichs and Moreno (2000)]) the perspectives proposed in [Courbon et al. (2010)], of automatically generating a visual memory and navigating through paths not taught in the earlier stage, can be tackled at once.

A first attempt to autonomously plan a path from a preobtained model while avoiding to accumulate error (Section 1.7), is to develop a VT&R system based on visual memory [Gaussier et al. (1997), Matsumoto et al. (1996) & Matsumoto et al. (1999)]. Generally speaking, these navigation systems store the topological representation of the scene in form of a connected graph which nodes are related to visual measurements and the edges to the navigability between them [Dame and Marchand (2013)]. The content of the visual memory depends on the trajectories chosen by the expert during the teaching stage but it does not exploit every possibility, as in a metric model. Now the question arises: Which type of visual feature to use? On the one hand, VT&R system based on visual memory that use sparse visual feature can be found at: [Goedemé et al. (2004), Courbon et al. (2010), Courbon et al. (2009b) & Blanc et al. (2005)]. On the other hand, some use a dense visual feature to navigate: [Dame and Marchand (2013), Raj et al. (2016) & Teulière and Marchand (2014)]. Direct approaches consider more information leading to higher accuracy when implemented in positioning tasks [Caron et al. (2013)]. Even for the naked eye of the expert in charge of the *teaching stage*, it is difficult to determine wether or not the system will converge when using any Image-Based Visual Servoing for which only local asymptotic stability can be obtained [Chaumette and Hutchinson (2006)]. Conveniently, such information can be determined from a preobtained model. In consequence, the system developed relies on a dense and direct feature, and a model of the scene.

3.2 Navigation Based on Servo Tasks and Sensory Memory

This section introduces the basis and fundamental concepts of the navigation based on servo tasks used on the contributions, i.e. PT&R, PNS and SNA.

In order to clarify the context of the contributions, the term **virtual** refers to a process that does not take place in the navigation scene; the term **hybrid** refers to a process that involves the sensor, the robot and a virtual reference; the term **real** refers to a process that does not involve any virtual representation, e.g. conventional PVS.

The contributions perform navigation, which belongs to a higher level autonomy class [Huang et al. (2003)] than the low level control of the servo task. Therefore, they assume that a control input $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is instantaneously applied. As it might not always be the case, the closed loop control will correct some of the "error" in the supposed instantaneous application of ${}^{W}\dot{\boldsymbol{\xi}}$. In this thesis, the autonomous navigation is conceived as a concatenation of positioning tasks, either by the servo approach (for SNA) or PVS (for PT&R and PNS). The former one is called **servo navigation** and the latter one **photometric navigation**.

For the sake of simplicity, the contributions suppose that the sensor is mounted on the robot, then the reference frames of the sensor and the robot (${}^{S}\mathfrak{F}$ and ${}^{R}\mathfrak{F}$, respectively) have the origin coincident with the axis aligned in the same direction. If this assumption is not true, e.g. when the architecture is implemented to a specific set-up, the velocity of the robot ${}^{R}\dot{\boldsymbol{\xi}}$ can be computed from the velocity of the sensor ${}^{S}\dot{\boldsymbol{\xi}}$ by:

$$ {}^{R}\dot{\boldsymbol{\xi}} = {}^{R}\mathbf{V}_{S}{}^{S}\dot{\boldsymbol{\xi}}, \tag{3.1} $$

where ${}^{R}\mathbf{V}_{S} \in \mathbb{R}^{6\times 6}$ is the twist transformation matrix that relates ${}^{S}\mathfrak{F}$ and ${}^{R}\mathfrak{F}$ (see Chapter 2.1.1). Therefore, $\boldsymbol{\xi}$ is used indistinguishable for the sensor and robot for the rest of the thesis.

When performing autonomous navigation, the desired elements stored in the memory serve as reference when sequentially accessing to them. The contributions uses two types of memories:

- **Sensory memory:** that stores the **sensory path** $\mathcal{S}^* = \{\mathbf{s}_0^*, \mathbf{s}_1^*, \mathbf{s}_2^*, \dots, \mathbf{s}_{|\mathcal{S}^*|}^*\}$ which is an ordered set of desired feature vectors $\mathbf{s}^*$ (SNA only).
- **Visual memory:** that stores the **visual path** $\mathcal{I}^* = \{\mathbf{I}_0^*, \mathbf{I}_1^*, \mathbf{I}_2^*, \dots, \mathbf{I}_{|\mathcal{I}^*|}^*\}$ which is an ordered set of desired luminance feature $\mathbf{I}^*$ (PT&R and PNS).

Notice that $\mathcal{I}^*$ is an instantiation of $\mathcal{S}^*$ and both of them are virtual, i.e. they are generated using the models of the sensor $\mathcal{M}^{sensor}$ and scene $\mathcal{M}^{scene}$. The latter one corresponds to a 3D model where a pose ${}^{V}\boldsymbol{\xi}$ can be defined. For PT&R and PNS, a satellite or aerial image texture the model as a flat mesh. All the virtual images ${}^{V}I$ are rendered from $\mathcal{M}^{scene}$ and $\mathcal{M}^{sensor}$ using the ViSP libraries [Marchand et al. (2005a)].

3.3 Luminance Feature for Vision-Based Navigation

The luminance feature $\mathbf{I}$ is a **dense** feature more accurate than a **sparse** one as it uses the entire image to perform a positioning task. However, it is not robust to light variation [Raj et al. (2016)]. This variation has even more impact when performing outdoor navigation. Furthermore, the illumination in a textured mesh is unlikely to be the same that the illumination of the scene; therefore, a hybrid positioning task using Photometric Visual Servoing (PVS), between a virtual image ${}^{V}I$ and an onboard one ${}^{W}I$ is more likely to fail. However, a Zero-mean Normalization (ZN) is a transformation applied to the image I able to reduce the impact of the light variation when comparing the virtual ${}^{V}I$ and onboard ${}^{W}I$ images, which is called **hybrid approach**. Moreover, a Gaussian filtering G might also be considered when the robot that performs visual servoing is not precise as shown in this Chapter.

The rest of the Section covers the performance of the point feature in the hybrid approach and the comparison between some similarity measurements.

3.3.1 Point Feature in the Hybrid Approach

Based on the literature [Remazeilles et al. (2006), Courbon et al. (2008) and Nguyen et al. (2014)], the first feature vector to consider is the **point feature**. The performance of the ORB (oriented BRIEF [Calonder et al. (2010)]) features [Rublee et al. (2011)] was tested within the hybrid approach.The virtual image ${}^{V}I$ was rendered from the textured mesh at a given pose in it ${}^{V}\boldsymbol{\xi}$, see Chapter 3.2. The onboard image ${}^{W}I$ was acquired by a conventional camera (Chapter 6.1) in such way that ${}^{W}I \approx {}^{V}I$. Nevertheless, the matching failed using Random Sample Consensus (RANSAC) [Fischler and Bolles (1981)] because some ORB features were not correctly located in both images. Fig. 3.1 displays one example of this situation.

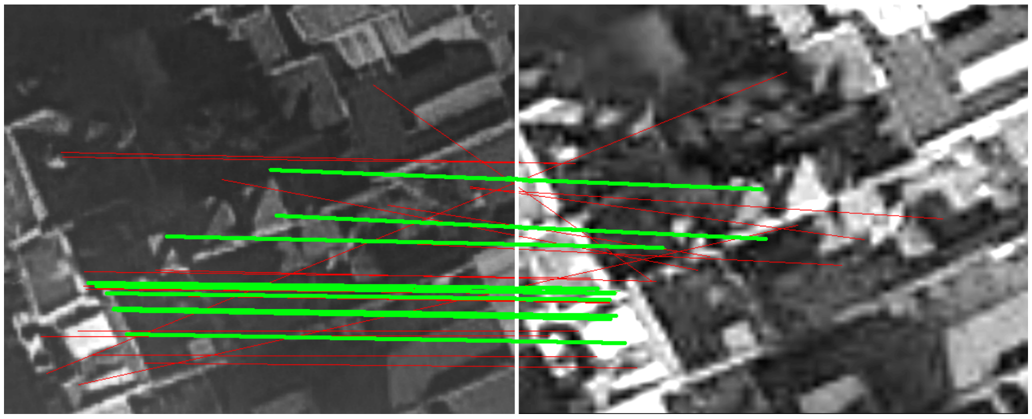


Figure 3.1: Imprecise matching. The image on the right is the onboard image whereas the image on the left is the virtual image (contrast enhanced for display). Relative point locations do no correspond between the two images.

The performance of this method was suboptimal mainly because the light source in the virtual image ${}^{V}I$ and in the onboard one ${}^{W}I$ were different. Indeed, the virtual image ${}^{V}I$ is not a snapshot of the textured scene as the onboard image ${}^{W}I$ is. The hybrid approach is fundamental in any memory-based system that generates the visual memory from a textured mesh. Therefore, another approach should be considered to cope with this problem.

3.3.2 Similarity Measurements for PT&R

To address the illumination problem that the ORB feature could not overcome, two *dense* and *direct* visual criteria were proposed for comparing the virtual image ${}^{V}I$ with the onboard image ${}^{W}I$: the Sum of Squared Differences (SSD) between pixel intensities and Mutual Information (MI) [Dame and Marchand (2011)]. Regarding PT&R context, SSD exhibits three important advantages over MI, when a Zero-mean Normalization is applied to pixel intensities of the images to be compared; thus, leading to ZNSSD.

The first advantage is related to the shape of the cost function $C(\boldsymbol{\xi})$ of ZNSSD of pixel intensities. This cost function is more convex and its minimum is more pronounced than the corresponding maximum of MI [Crombez et al. (2015)]. These characteristics cause a more rapid convergence to the optimum while applying a control law which reduces the computational cost at the *path-planning stage* (Chapter 4.4.1).

The second reason is that the domain of convergence $\mathcal{C}$, of the ZNSSD is larger than the corresponding one of MI. Such a statement is supported by the experiments carried out in Park et al. (2017) between MI and the zero-mean normalized cross correlation (ZNCC), and the equivalence between ZNCC and ZNSSD highlighted by Pan (2011). The wider the convergence domain, the weaker the number of vertices in the topological graph of poses for path-planning (Chapter 4.4.1).

Thirdly, considering ZNSSD rather than SSD in the control law for visual servoing [Collewet and Marchand (2011)] slightly increases the computational cost. This fact is caused by the computation of the zero-mean normalization of pixel intensities before computing the Extended Photometric Visual Servoing (Extended PVS) control law (Eq. (5.3)). However, the computational cost remains far lower than in MI-based visual servoing.

A Gaussian filter G was also considered to smooth the cost function $C(\boldsymbol{\xi})$, either by GSSD or GZNSSD. Indeed, GSSD and GZNSSD produce smooth cost functions as shown in Fig. 3.2. However, GSSD is not robust to illumination changes as ZNSSD, so it was discarded. Moreover, the cost function $C(\boldsymbol{\xi})$ generated by SSD and GSSD do not reach the value of zero at the desired pose ${}^{V}\boldsymbol{\xi}^{*}$ in the textured mesh because the I^{*} is darker than the other virtual images ${}^{V}I$. The costs C that GZNSSD and ZNSSD produce are similar (Fig. 3.2) and robust to illumination changes. The main difference lies in the shape of the extents of their domains of convergence $\mathcal{C}$, larger for GZNSSD than ZNSSD. Which is convenient when a robot that performs a positioning task is not precise. However, by adding a Gaussian filter G to the ZN transformation, the computation time slightly increases.

Fig. 3.3 shows the corresponding desired images $I({}^{V}\boldsymbol{\xi}^{*})$ from which cost functions $C(\boldsymbol{\xi})$ (SSD, GSSD, ZNSSD and GZNSSD) were computed. Finally, by aligning every corresponding pixel of images, PVS is far more precise at convergence than an ORB-based visual servoing could be, considering the rough matching of the latter in the hybrid approach (Fig. 3.1).

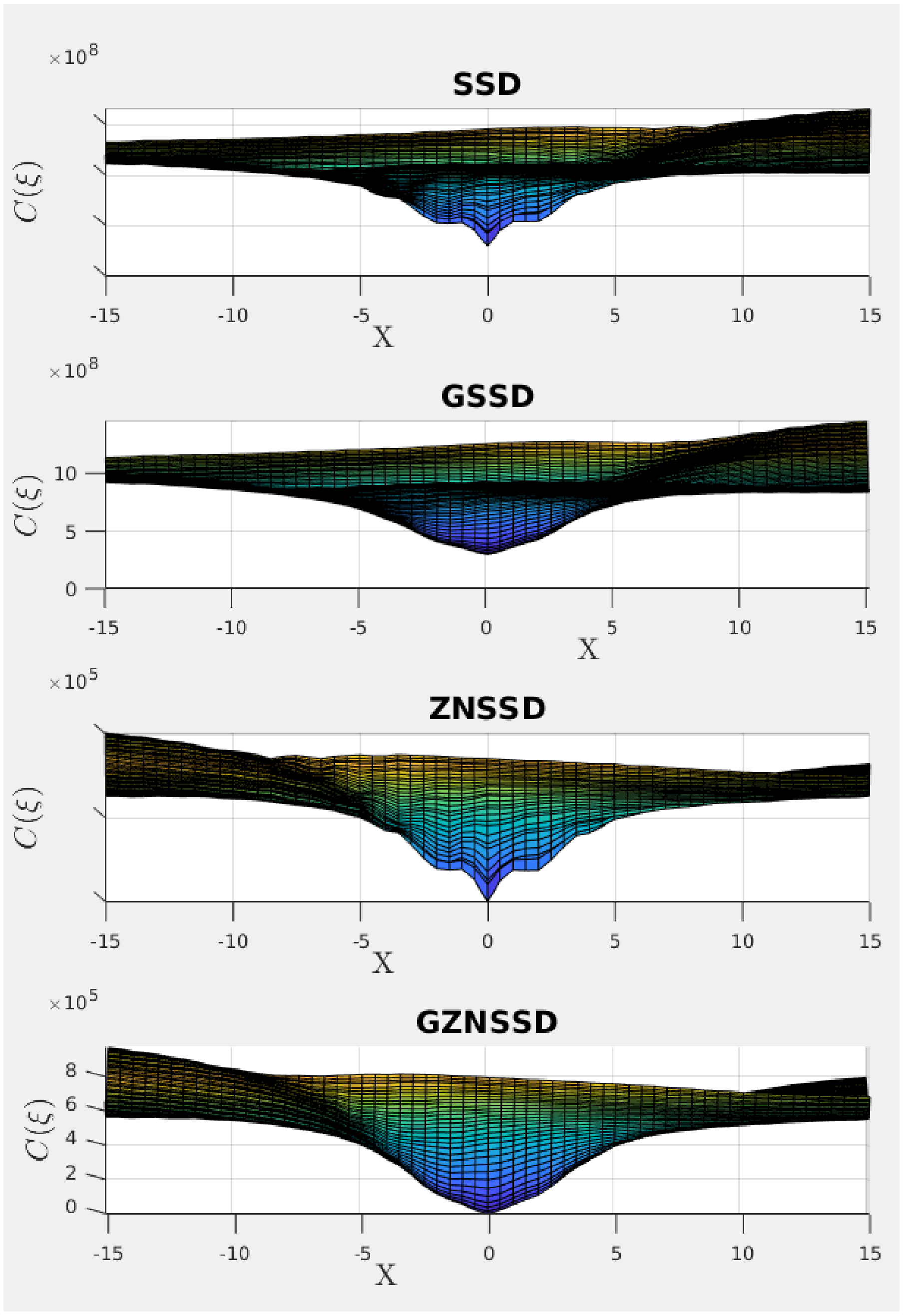


Figure 3.2: Study of the shape of the cost functions for SSD, GSSD, ZNSSD and GZNSSD. Distances from the central locations are shown in mm.

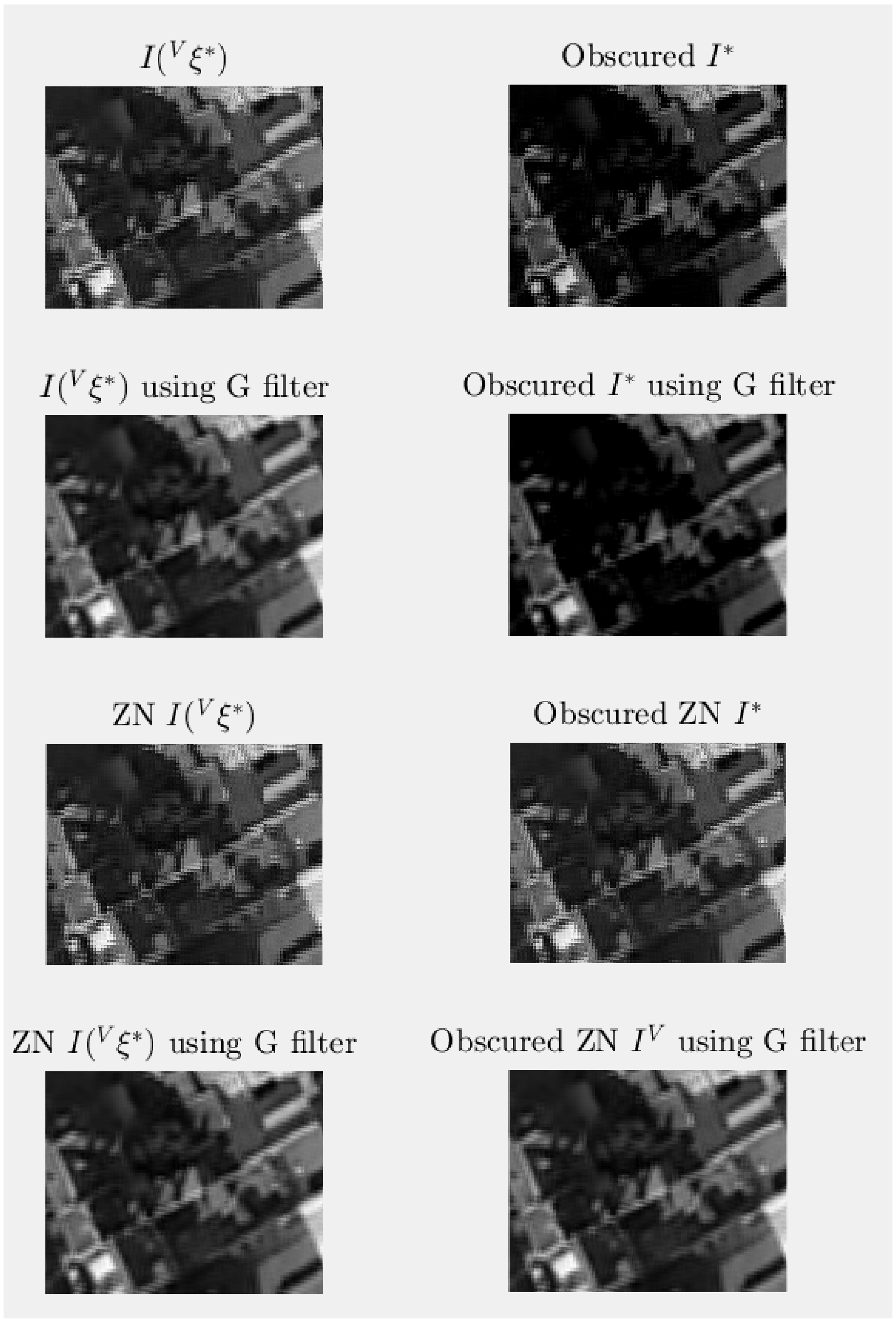

$I(^V\xi^*)$
Obscured I^*
$I(^V\xi^*)$ using G filter
Obscured I^* using G filter
ZN $I(^V\xi^*)$
Obscured ZN I^*
ZN $I(^V\xi^*)$ using G filter
Obscured ZN I^V using G filter

Figure 3.3: Desired images used for the study of the shape of the cost functions.

3.4 System Overview

PT&R is a Vision-Based Navigation System (VBNS) that autonomously generates its visual memory from a model of the scene. The visual memory contains a **visual path** $\check{\mathcal{I}}^*$ that indirectly embodies poses over the scene. The visual path $\check{\mathcal{I}}^*$ of PT&R contains a set of desired features called Gaussian (G) Zero-mean Normalized (ZN) luminance $\check{\mathbf{I}}$ (Section 3.4.1). As in the VT&R, the PT&R comprises of identical main *stages*, the *teach* and *repeat*. Here, in the first *stage* the visual memory is generated (Section 3.4.2) whereas in the next *stage* the navigation occurs (Section 3.4.3). The term "teach" is used since this *stage* mimics a human operator that proposes a path in straight line over the preobtained model, which is a **textured mesh**.

The rest of the Section presents the feature that the system uses and the main processes of the two *stages*.

3.4.1 GZN Luminance Feature

The GZN luminance feature $\check{\mathbf{I}}$ is selected over the luminance feature $\mathbf{I}$ since the former one is more robust to illumation variations (inherited from ZNSSD) and the shape of the cost C is more convex (Section 3.3.2).

The GZN luminance feature $\check{\mathbf{I}}$ is obtained from the pixels intensities of the image I after a Gaussian filtering with the following kernel:

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.2}$$

where x and y are the distance from the origin to their corresponding axes, standard deviation σ, and a Zero-mean Normalization; thus, becoming:

$$\check{\mathbf{I}}(\boldsymbol{\xi}) = (\check{\mathbf{I}}_{1\bullet}, \check{\mathbf{I}}_{2\bullet}, \dots, \check{\mathbf{I}}_{N\bullet})^\top \tag{3.3}$$

where $\check{\mathbf{I}}_{i\bullet} \in \mathbb{N}^{1\times M}$ is the i-th line of the GZN image $\check{I}$. Similar to Eq. (2.21), N is the width and M the height of the image.

Then, its cost function is defined as:

$$C(\boldsymbol{\xi}) = \frac{1}{2}\|\check{\mathbf{I}}(\boldsymbol{\xi}) - \check{\mathbf{I}}(\boldsymbol{\xi}^*)\|^2 \tag{3.4}$$

which has a smoother shape than the one of ZN luminance that causes a quicker convergence.

The chosen control law that handles the motion of the robot is the Gauss-Newton method:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^{+}_{\check{\mathbf{I}}(\boldsymbol{\xi}^*)} (\check{\mathbf{I}}(\boldsymbol{\xi}) - \check{\mathbf{I}}(\boldsymbol{\xi}^*)), \tag{3.5}$$

where $\mathbf{u} = \dot{\boldsymbol{\xi}}$ is the control input $\widehat{\mathbf{L}}^{+}_{\check{\mathbf{I}}(\boldsymbol{\xi}^*)}$ is the Moore-Penrose pseudoinverse of the approximation of the interaction matrix $\mathbf{L}_{\check{\mathbf{I}}(\boldsymbol{\xi}^*)} = -\nabla \check{\mathbf{I}}(\boldsymbol{\xi}^*)^\top \mathbf{L}_{\mathbf{x}}$ related to $\check{\mathbf{I}}(\boldsymbol{\xi}^*)$ when $\mathbf{L}_{\check{\mathbf{I}}(\boldsymbol{\xi}^*)}$ is of full rank and $\nabla \check{\mathbf{I}}(\boldsymbol{\xi}^*)$ is the gradient of $\check{\mathbf{I}}(\boldsymbol{\xi}^*)$.

3.4.2 Visual Memory Generation

The visual memory is generated during the *teaching stage* from a preobtained map, called **textured mesh**. The first *stage* begins by defining a set of world goal poses ${}^{W}\boldsymbol{\Xi}^{G}$ over the scene. These poses shares the same orientation ${}^{W}\boldsymbol{\Theta}^{G}$. Then, this set is represented in the textured mesh by ${}^{V}\boldsymbol{\Theta}^{G}$. Let ${}^{V}\boldsymbol{\xi}^{G}_{i}$ and ${}^{V}\boldsymbol{\xi}^{G}_{i+1}$ be a group of two consecutive goal poses $\{{}^{V}\boldsymbol{\xi}^{G}_{i}, {}^{W}\boldsymbol{\xi}^{G}_{i+1}\} \subset {}^{V}\boldsymbol{\Xi}^{G}$ defined in the mesh. Then, a trajectory in straight line is proposed from ${}^{V}\boldsymbol{\xi}^{G}_{i+1}$ to ${}^{V}\boldsymbol{\xi}^{G}_{i}$. The trajectory is sampled in candidates poses ${}^{V}\boldsymbol{\xi}^{C}$ every step Δ in such way that ${}^{V}\boldsymbol{\xi}^{C}_{0} = {}^{V}\boldsymbol{\xi}^{G}_{g-1}$, which ${}^{V}\boldsymbol{\xi}^{G}_{g-1}$ is the last goal pose defined in the mesh. Fig 3.4 illustrate the sampling process. Beginning from the last candidate pose ${}^{V}\boldsymbol{\xi}^{C}$ to the first one, a virtual image is rendered using the inverse of the perspective camera function (Eq. (2.1.3)) and transformed into $\check{I}$. The current GZN luminance feature $\check{\mathbf{I}}({}^{V}\boldsymbol{\xi}^{C})$ is stored in the memory and is considered desired feature $\check{\mathbf{I}}^*$ if one of the two following conditions is satisfied:

- The current candidate pose ${}^{V}\boldsymbol{\xi}^{C}$ is also a goal pose ${}^{V}\boldsymbol{\xi}^{G}$ defined in the mesh.
- Being $\check{\mathbf{I}}^*$ the last selected desired GZN luminance feature, the value of the numerical differentiation of the cost $C = \|\check{\mathbf{I}} - \check{\mathbf{I}}^*\|$ falls below a threshold ϵ_d.

This is known as the **selection criteria**.

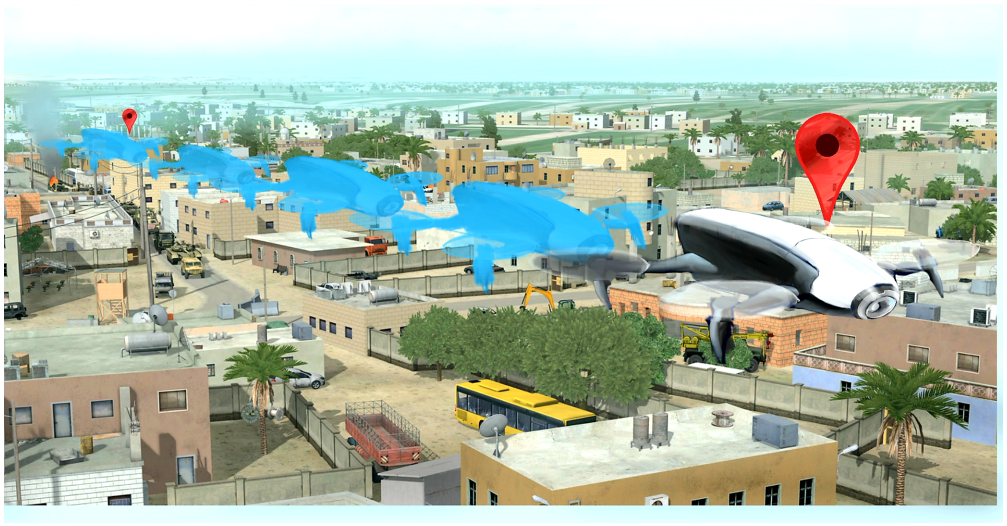


Figure 3.4: Sampling process of PT&R.

3.4.3 Hybrid Servo Navigation

During the *repeat stage*, the navigation is conceived as a concatenation of positioning servo tasks. Due the nature of comparing a virtual reference (a desired feature in the visual path $\check{\mathcal{I}}^*$) with a real feature $\check{\mathcal{I}}({}^W\boldsymbol{\xi})$ (obtained from the onboard sensor), such concatenation will be called **hybrid servo navigation** for the rest of this document. Beginning from the first element $\check{\mathbf{I}}_0^* = \check{\mathbf{I}}^*$ in the visual path $\check{\mathcal{I}}^*$, the aforementioned navigation occurs by applying to the robot the **hybrid control law for the GZN scheme**:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^+_{\check{\mathbf{I}}(\boldsymbol{\xi}^*)}(\check{\mathbf{I}}({}^W\boldsymbol{\xi}) - \check{\mathbf{I}}(\boldsymbol{\xi}^*)), \tag{3.6}$$

where $\mathbf{u} = {}^W\dot{\boldsymbol{\xi}}$. The desired feature $\check{\mathbf{I}}^*$ in the visual path $\check{\mathcal{I}}^*$ will change to the next one, if one of the following conditions is satisfied:

- If the current desired feature $\check{\mathbf{I}}^*$ is not the last element in the visual path $\check{\mathcal{I}}^*$.
- If the value of the GZN error $\check{\mathbf{e}}$ falls below a threshold ϵ_e.

This is known as the **update criteria**.

3.5 Implementation

The experimental setup (Fig. 3.5) is composed by a Bebop 2, which is a micro aerial vehicle (MAV) equipped with a camera, and the optitrack system, to track and record the pose of the robot and a HP ZBook 15, as workstation.

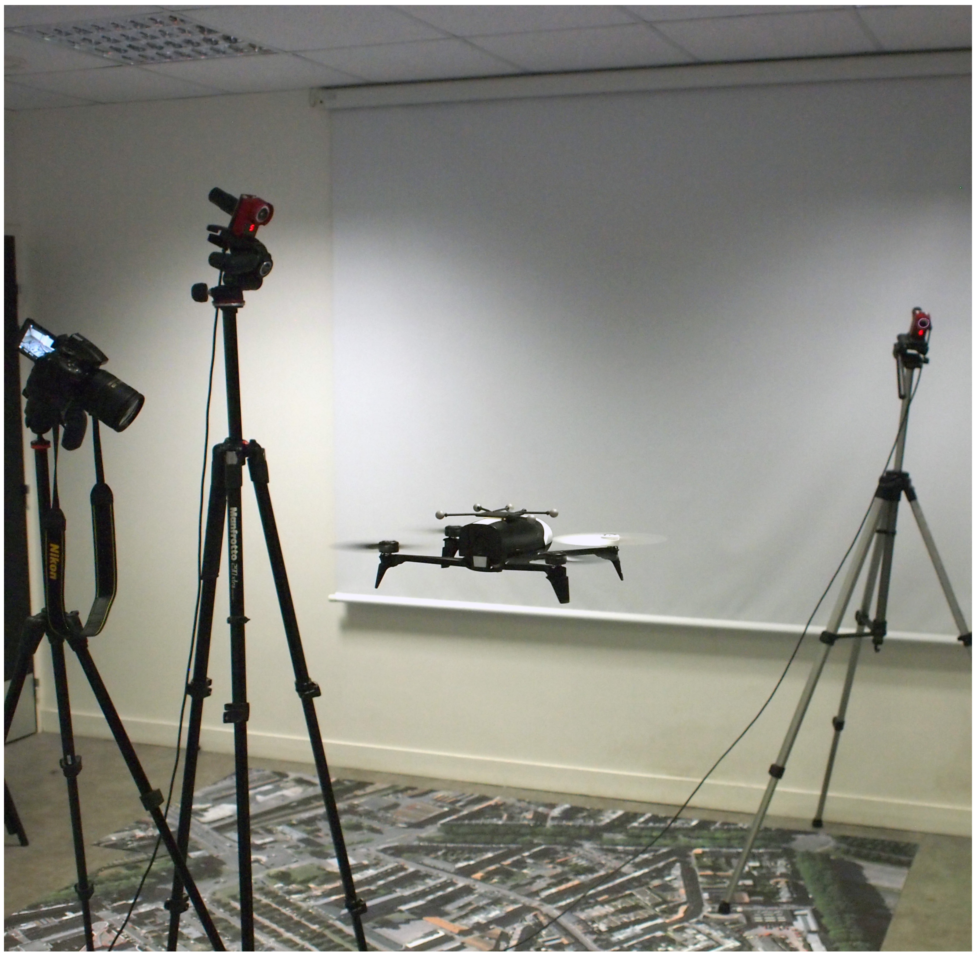


Figure 3.5: Setup for indoor experimentation. The Bebop 2 flies over the scene with the onboard camera virtually pointing downwards. The Optitrack system tracks records the pose of the rigid frame of markers attached to the robot. Additionally, a Nikon D5200 records video of the experiment.

The Parrot Bebop 2 is a commercial aerial robot classified as quadrotor because it is lifted and propolled with the thrust four motors. It belongs to the mini Rotorcraft Unmanned Aircraft Vehicle (RUAV) class [Kendoul (2012)] because its weight is barely 500 g. The RUAV is equipped with two cameras, a GPS module, an Inertial Measurement Unit (IMU), a barometer, a dual processor with quadcore Graphics Processing Unit (GPU) and a Wi-Fi module[1]. The communication is usually established through Wi-Fi from the robot to a ground station, e.g. a computer, a tablet or a

[1]for more details, see the manufacturer website`https://www.parrot.com/us/drones/parrot-bebop-2-fpv`

smartphone. When transmitted through Wi-Fi, the inertial measurements update at 5 Hz [Monajjemi (2018)]. Such measurements are provided by the Visual-Odometry (VO) system composed by a down-looking camera and the inertial sensors [Monajjemi (2018)]. The frontal camera features a digitally stabilized image that can be controlled by software as a conventional pan-tilt camera. In pilot mode, the robot is controlled in 4 DoF by a control input $\widehat{\mathbf{u}} = [{}^{W}\Phi, {}^{W}\Theta, {}^{W}v_Z, {}^{W}\omega_Z]^\top$. The reference frame of the Bebop 2 appears in Fig. 3.6. Parrot provides an official Software Development Kit (SDK) from which the Robot Operating System (ROS) bebop_autonomy package[2] is based.

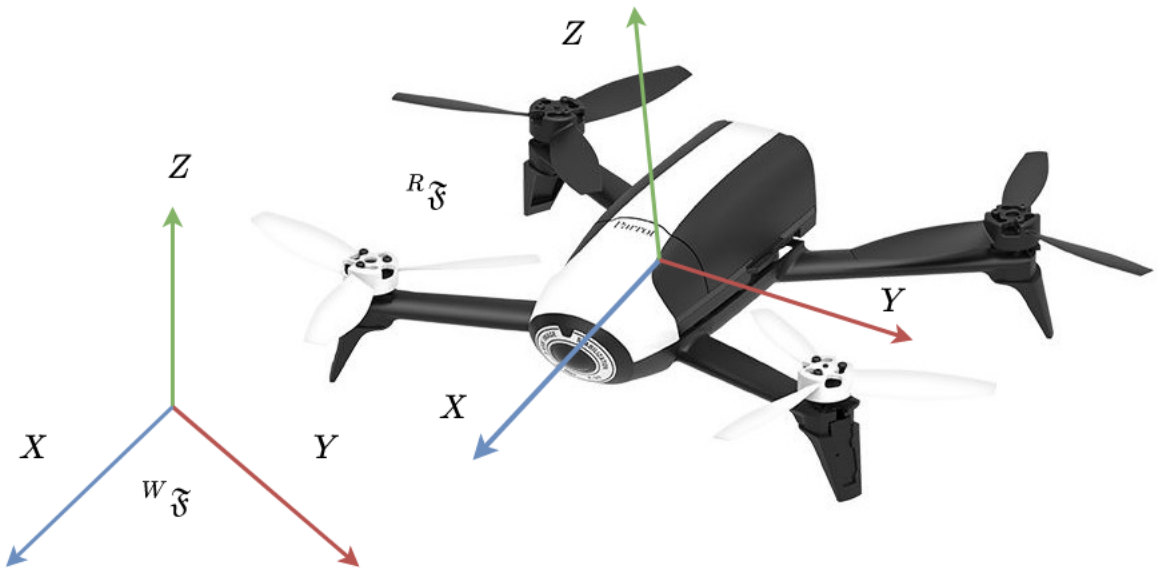


Figure 3.6: Reference frame of the robot and the world.

In the following experiments, the Bebop 2 is wireless connected to a HP ZBook 15 which serves as workstation. It features 16 GB of RAM and an Intel Core i7 processor with 6 cores and runs Ubuntu 16.04 LTS and ROS kinetic to communicate with the Bebop 2 using the bebop_autonomy package. A brief introduction to ROS is provided in the Annex.

The *teaching stage* is executed in MATLAB (offline) and the *repeat stage* is coded in C++ (online), as a ROS node, using the Visual Servoing Platform (ViSP) [Marchand et al. (2005b)]. The onboard image ${}^{W}I$ is acquired by the Bebop 2 and sent to the workstation. Then, the latter replies with a control input $\widehat{\mathbf{u}}$ adapted to the Bebop 2.

[2]see the website of the package `https://bebop-autonomy.readthedocs.io/en/latest/`

Indeed, a **proportional controller** was used to approximate $\widehat{\mathbf{u}}$ to $\mathbf{u}$ in 4 DoF. The controller is applied only in ${}^{W}\Theta$ and ${}^{W}\Phi$, such that $[c_0^{-1}\,{}^{W}\Phi, c_1^{-1}\,{}^{W}\Theta, {}^{W}v_Z, {}^{W}\omega_Z]^\top \approx [{}^{W}v_x, {}^{W}v_y, {}^{W}v_Z, {}^{W}\omega_Z]^\top$. The latter was computed from a velocity characterization of second order and the dynamic model of the quadrotor proposed in [Luukkonen (2011)] which considers air resistance:

$$\begin{bmatrix} {}^{W}v_x(t) \\ {}^{W}v_y(t) \end{bmatrix} = \frac{g}{m} \begin{bmatrix} c_0 & 0 \\ 0 & c_1 \end{bmatrix} \begin{bmatrix} tan({}^{W}\Phi)(1 - e^{-t\frac{c_0}{m}}) \\ tan({}^{W}\Theta)(1 - e^{-t\frac{c_1}{m}}) \end{bmatrix} \tag{3.7}$$

where the acceleration caused by the gravity $g = 9.8m \cdot s^{-2}$, the coefficients of the air in both axes are $c_0 = c_1 = 0.35$, and the mass of the quadrotor $m = 0.497$ kg. A PID controller [Salih et al. (2010)] would be prefered over a P controller but it was not possible to design since the readings of the IMU are retrieved at 5 Hz.

Furthermore, an **image latency**, i.e. from an event in the real world to its corresponding change in the image, was measured with a mean of 183.9 ms and a standard deviation of 11.26 ms. The latency was measured using an executable, coded in C++, that triggers a LED through a mbed[3] microcontroller and starts a chronometer in the executable. Then, the chronometer stops when a pixel in the center of the image detects a high change of light. In order to mitigate the impact of the latency during the hybrid servo navigation, a null control input $\widehat{\mathbf{u}} = [0, 0, 0, 0]^\top$ is sent to the robot each visual memory update during 20 iterations at 30 Hz.

In order to measure the trajectories of the flying robot, the experimental setup features an Optitrack motion capture system. It uses infrared light and infrared cameras. The setup used to compute the pose of the Bebop 2 ${}^{W}\boldsymbol{\xi}$ is composed by cameras Flex 3 (Fig. 3.5) and a rigid frame (Fig. 3.5) of markers on the top of the drone. The cameras are connected to the OptiSync through USB 2.0 and the latter is connected to a PC running Windows 10, which receives the information from the OptiSync and process it using the Motive software. Notice that this is not the workstation that controls the Bebop 2. Then, the tracked pose is stored in a file to be plotted, e.g. using MATLAB.

The next Section details the experiments of PT&R carried out with this setup.

3.6 Experiments

Two types of experiments evaluate the performance of the PT&R system: **positioning** and **navigation tasks**. Each one of them are performed **indoor** and

[3]Mbed website `https://os.mbed.com/`

outdoor. The size of the onboard and virtual images is set to 240 pixels. For the indoor experiments, a satellite image of 2952 $\times$ 3052 pixels from Amiens, France at a height of 1449 m is provided by Google Earth [4] from which the textured mesh is created. The scene corresponds to the mesh printed on a tarpaulin of approximately 4 m per side. In these experiments, the robot flies at a height of 90 cm.

Regarding outdoor experiments, two meshes and scenes are considered. The mesh used in the positioning experiment is also generated from a satellite image of Google Earth. In this experiment, the drone should fly at 10 m above a crossroad that corresponds to the scene and appears in the textured mesh. Then, the mesh used in the outdoor navigation, called **parking experiment**, is obtained from a snapshot that the drone takes when flying at 30 m above the ground. However, the hybrid servo navigation is carried out at a height of 13 m.

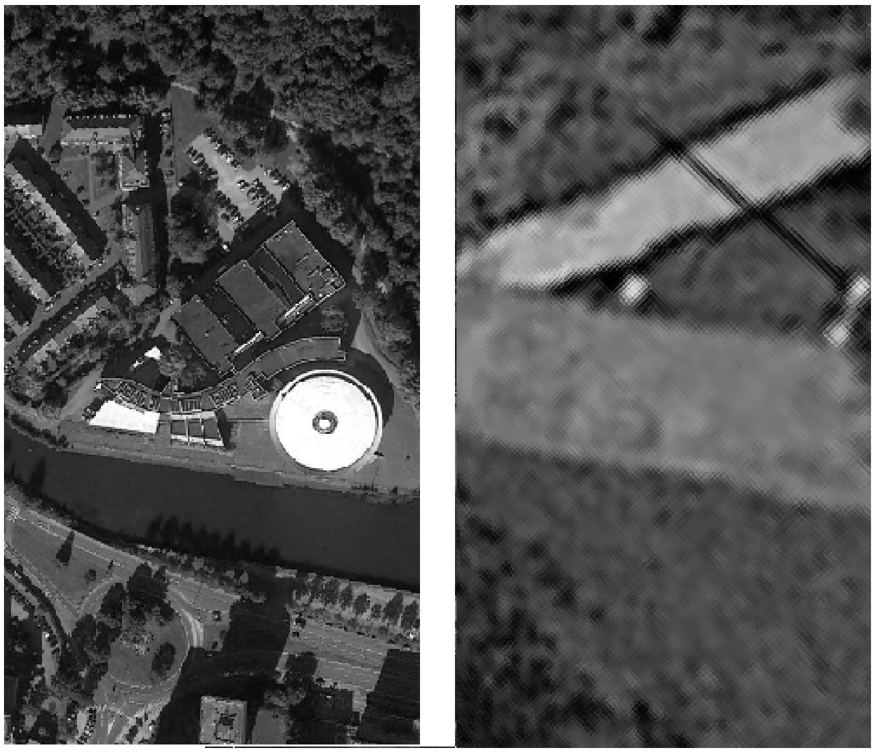


Figure 3.7: Desired virtual images concerning PT&R positioning tasks. Image used for indoor positioning (left) and image used for outdoor positioning (right)

[4]Google Earth at `https://www.google.com/earth/`

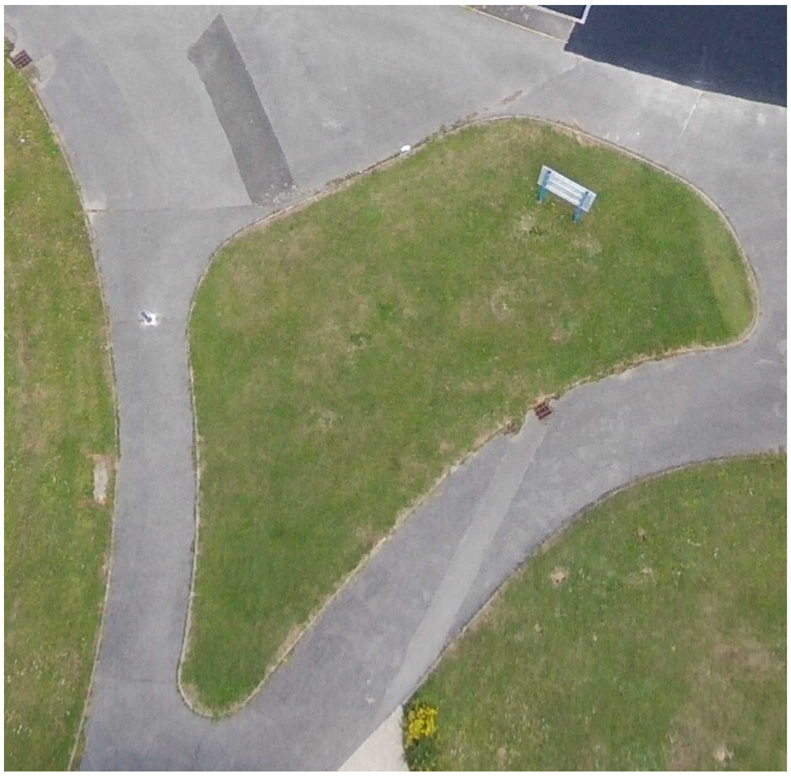


Figure 3.8: Textured mesh used for the parking experiment.

3.6.1 Positioning Task

Three positioning tasks (two indoor and one outdoor) are proposed to test the performance of the GZN luminance $\check{\mathbf{I}}$ and the experimental setup before applying PT&R. The desired virtual images ${}^{V}I^{*}$ for the two groups of experiments are shown in Fig. 3.7. If the value of cost C (Eq. (3.4)) lies between two thresholds $\epsilon_{min} < C < \epsilon_{max}$, the input velocity $\bar{\mathbf{u}}$ computed from the hybrid control law for the GZN scheme (Eq. (3.7)) is applied, this is known as "servoing mode". Otherwise, null velocities are sent $\widehat{\mathbf{u}} = [0, 0, 0, 0]^{\top}$ so the Bebop 2 enters in "hovering mode" (managed by the internal VO system). If the value of the cost $C > \epsilon_{max}$, the task is considered as failed. It is worth to say that these thresholds were experimentally set to $\epsilon_{min} = 1.4 \times 10^{5}$ and $\epsilon_{max} = 2 \times 10^{5}$.

Indoor Hybrid Servoing: This experiment consists in servoing a virtual GZN desired image ${}^{V}\check{\mathbf{I}}$. It is important to notice that when the robot obeys null velocities $\widehat{\mathbf{u}} = [0, 0, 0, 0]^{\top}$ its VO is not able to compensate the dragging error and it eventually diverges. While applying the control law (Eq. (3.6)), the robot minimizes the cost $\mathcal{C}$ (Eq. (3.4)). Then, the experiment finished when the memory that stores the data of the experiment was filled up in about 5 minutes. The cost C and control input evolutions are displayed in Figure

Indoor Gusty: A fan is actuated 2 m behind the robot, in oscillation mode, to cause perturbation while the robot minimize the GZN the cost $\mathcal{C}$ (Eq. (3.4)). In this experiment, if the hovering mode is activated, the drone diverges quicker than in the indoor servoing experiment. However, the experiment lasted 163 s when the robot entered in servoing mode.

Outdoor Crossroad: This experiment took place on a crossroad during the day but with presence of clouds. The robot flies at 10 m above the ground while being perturbed by wind at low speed. It remained in place without any important perturbance for the entire task that lasted 240 s. Indeed, the robot hovered for the most part of the experiment except at the beginning and at $t = 100$ s when a pedestrian caused an increment in the value of the cost C and triggered the servoing mode.

This group of experiments showed that the hybrid servoing is performed at scale (concerning image variation and camera displacement) since Z is not determined from this scheme. Therefore, perturbations that occur at lower altitude had a greater impact in these tasks since they triggered the servoing mode several times whereas perturbations in the image at higher altitude were less impacting. The perturbation caused by the pedestrian showed that the GZN hybrid servoing is robust to partial occlusions. Unfortunately, the GPS was not available at the moment of the experiment, so no ground truth data was retrieved. Although it prevents the quantitative evaluation of this experiment, it also shows the interest of not relying only on GNSS for robot positioning and navigation.

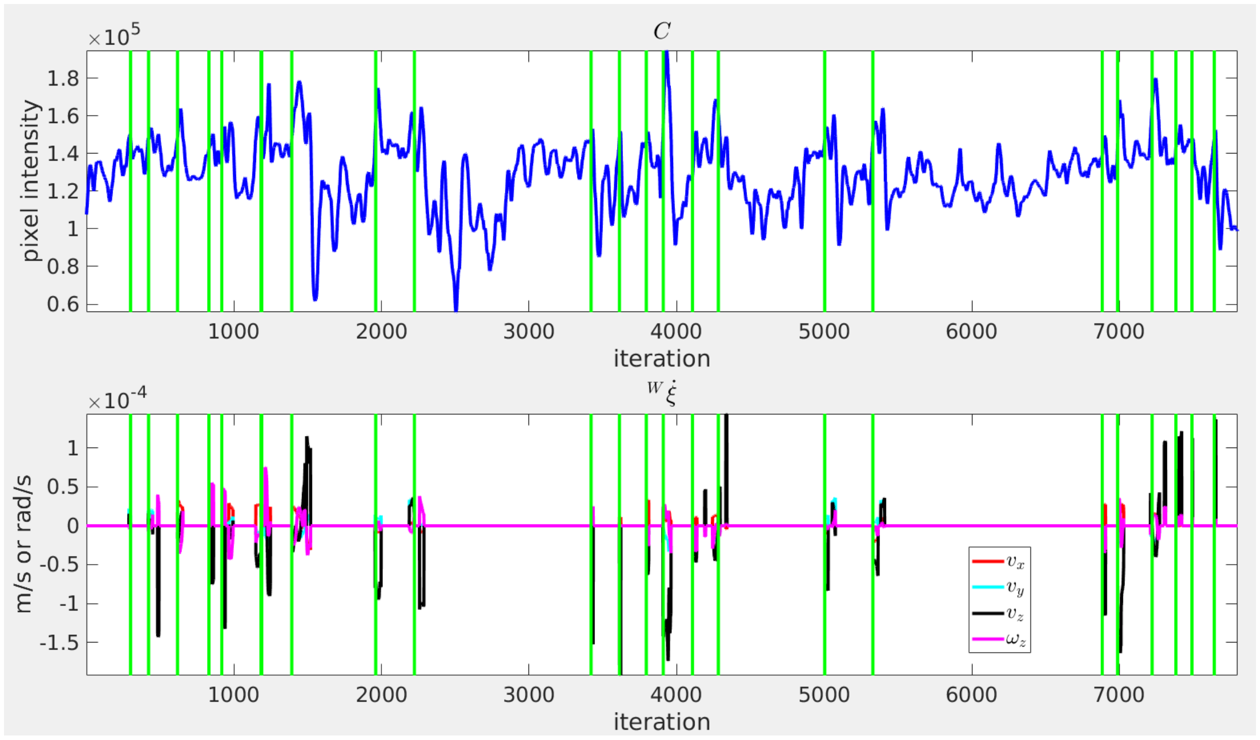


Figure 3.9: Cost $\mathcal{C}$ and control input $\bar{\mathbf{u}}$ evolution of the indoor hybrid servoing. Vertical lines indicate the memory update.

3.6.2 Navigation Task

Two navigation tasks test the performance of PT&R implemented on the experimental setup. All of these tasks are defined as **roundtrips** to evaluate the visual path $\check{\mathcal{I}}^*$.

Indoor Navigation: In this experiment, the robot is intended to travel a path of 55 cm in straight line. This experiment is displayed in Fig. 3.5. During the *teaching stage*, a visual path $\check{\mathcal{I}}^*$ of $|\check{\mathcal{I}}^*| = 11$ desired GZN luminance features $\check{\mathbf{I}}^*$ is stored in the visual memory. The hybrid servo navigation attained 54 desired GZN luminance features $\check{\mathbf{I}}^*$ before diverging. Fig. 3.10 shows the trajectory of a round trip and Fig. 3.9 display the cost C and the velocity input evolution $\bar{\mathbf{u}}$ of this experiment.

Parking Experiment: This experiment takes place in the parking of the University of Picardie Jules Verne Campus Amiens, France. The *mission* is defined by two goal poses ${}^{W}\boldsymbol{\xi}_0^G$ and ${}^{W}\boldsymbol{\xi}_1^G$ within a distance of 8 m. PT&R generated a visual memory with a visual path $\check{\mathcal{I}}^*$ that contained $|\check{\mathcal{I}}^*| = 5$ desired GZN luminance features $\check{\mathbf{I}}^*$ during the *teaching stage*. Then, the robot traveled the roundtrip using the visual path $\check{\mathcal{I}}^*$ but it diverged when it tried to restart from $\check{\mathbf{I}}_0^*$. Unfortunately, the GPS was not available during this experiment neither. Fig. 3.12 shows an excerpt of the images related to this experiment and Fig. 3.11 displays the cost C and the control input evolution $\bar{\mathbf{u}}$.

From this group of experiments, it is fair to state that PT&R completed the navigation tasks but it diverged when an important perturbation in the image or in the robot occurred.

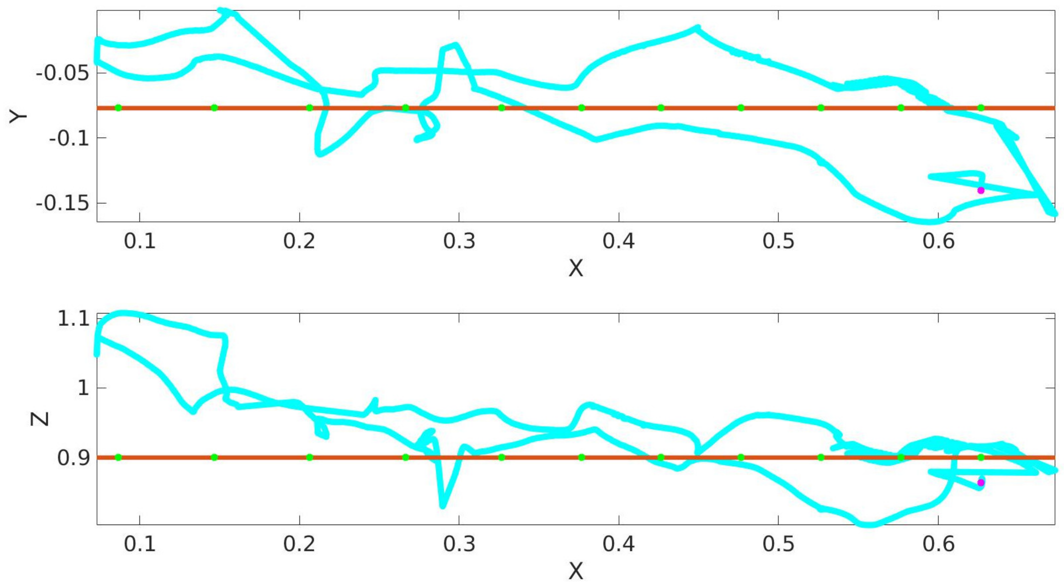


Figure 3.10: Trajectory of the indoor navigation. Green dots indicate the locations of the poses from where the visual path $\bar{\mathcal{I}}^*$ was obtained and purple dots indicate the location where the drone diverged.

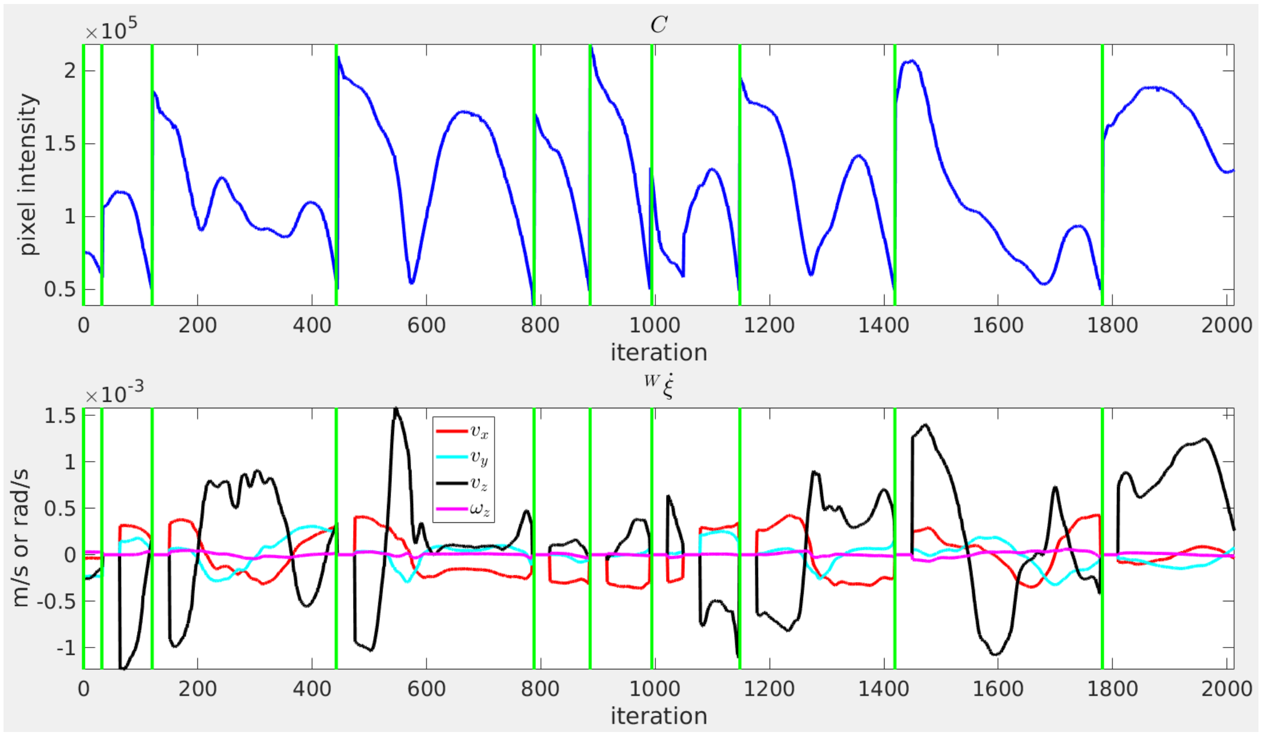


Figure 3.11: Cost $\mathcal{C}$ and control input $\bar{\mathbf{u}}$ evolution of the parking experiment. Vertical lines indicate the memory update.

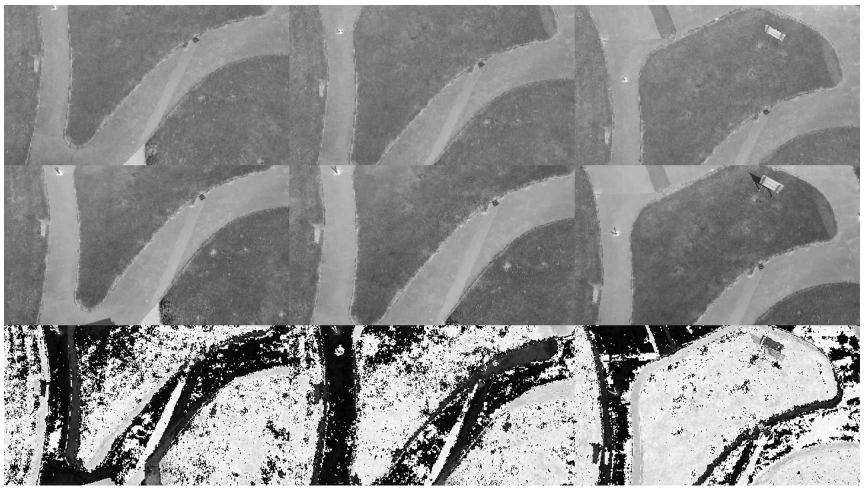


Figure 3.12: Excerpt of some images from the parking experiment. Desired virtual images appear at the top row, onboard images appear at the central row and images of difference appear at the bottom row.

3.7 Conclusions

PT&R is the first attempt to tackle the problem of memory-based navigation while relying on a textured mesh. The system results represent an immediate solution of the current state-of-the-art limitations regarding *ex situ* visual memory generation. The pros and cons exhibited by PT&R are summarized next.

Pros:

- The textured mesh can be an instantaneous model (also called snapshot [Salichs and Moreno (2000)]) generated from a single aerial image taken by an UAV.
- At teaching, the robot is not required *in situ* as in other VT&R systems.
- The system overcomes small perturbations in the image and in the robot.
- The digital image stabilization of the Bebop 2 filters high frequency perturbations in the image.

Cons:

- As it relies on a static mesh, it is not adapted to dynamic environments [Güzel (2013)].

- The Bebop 2 is not a well adequated testbed because it presents some flaws, e.g. a lack of velocity low level control (since the control input $\bar{\mathbf{u}}$ is not correctly applied), poor GPS signal (important for ground truth data acquisition) and unstable hovering mode (less important). Nevertheless, hovering can be achieved by means of PVS, e.g. indoor positioning experiments.

- The autonomous navigation is ensured for 1 Degree of Freedom (DoF) when the navigation space is correctly represented by its model.

- Not all possible trajectories are explored [Raj et al. (2016)]; just the straight line.

- The dissimilarities between the model of the scene and the scene itself are not considered for generating the visual memory.

Similar to others VT&R systems, only one trajectory is explored during the *teaching stage*; therefore, some navigables paths are not considered during the *repeating stage*. Furthermore, the selection criteria only guarantee the convergence of the system if the latter is actuated in one DoF since the numerical derivative of the cost C is computed by translation along one axis. In a practical scenario for Bebop 2, this is not the case since its flight is susceptible to perturbations.

To summarize, PT&R represents a contribution to vision-based navigation since it is the pioneer to generate its visual memory from a preobtained mesh. Moreover, it was implemented on an aerial robot overcoming most difficulties. The visual memories were generated backwards and validated in roundtrip to evaluate the robustness of the virtual Teaching stage. Lastly, this system inspired the research for another VBNS which can solve the aforementioned flaws; the Photometric Navigation System (PNS), Chapter 5.

Chapter 4

Servo Navigation Architecture

The Servo Navigation Architecture (SNA) is the first of the two main contributions presented in this thesis. The main difference between SNA and other Goal-Oriented Navigation Systems (including teach & repeat) is the process to autonomously generate a navigable path while relying on a preobtained model of the scene.

The content of this Chapter is theoretical. The architecture is implemented on a vision-based navigation system in Chapter 5.

The rest of the Chapter follows the next structure: a general idea of SNA is provided in Section 4.1, Section 4.2 is dedicated to study the models and sensors suitable for SNA, two types of servo tasks are defined in Section 4.3 and the entire process of SNA is detailed in Chapter 4.4. Finally, the Chapter concludes in Section 4.5.

4.1 Architecture Overview

Briefly speaking, SNA reformulates the navigation problem, i.e. to reach a list of **world goal poses**, as a sequence of **servo tasks**. These tasks consist in minimizing the **error**, i.e. the difference between the **current measurement** and the **reference**, by means of a control law [Magassouba et al. (2018)].

In the context of SNA, the current measurement is perceived by the navigation sensor mounted on the robot, called **onboard sensor**, from which a **feature vector** is obtained. The reference is generated before the robotic navigation takes place and it contains a list of **desired feature vectors**. When the robot obeys a control law that regulates the error to zero, i.e. by comparing the feature vectors from the sensor measurement and those in the reference, one of three situations may occur:

- the robot will converge towards the reference from an initial pose;
- the robot will diverge from the reference, i.e. it will move far away from the latter;
- the robot will not move.

Thus, SNA exploits the implicit embodiment of a pose in the feature vector, relating the servo and navigation problems. The difficulty lies in how to determine the reference in such way that the robot converge towards the poses implicetely embodied in the reference. The architecture tackles this problem by considering a **sensory memory** as reference. This memory embodies a list of intermediate poses that the robot must successively reach. SNA uses a metric-topological approach to generate the memory from a preobtained model of the scene.

SNA ensures a successful navigation when the scene is static and a model of it is provided. This architecture does not require any localization method nor a human expert to propose or impose a trajectory to be followed. Moreover, SNA is suitable for any mobile robot (or even arm-type robots) and many combinations of sensors and models of the scene (Chapter 4.2).

The goal of the *mission* is to sequentially reach the list of world goal poses implicitely defined by the sensory memory. It is composed by two stages: the *path-planning stage*, which goal is to generate a sensory memory and the *navigation stage*, which goal is to navigate using the memory.

During the *path-planning stage*, SNA searches a navigable path running A* over a kinggraph-m, called the **navigation graph**, and simulated servo tasks are executed in order to validate it. The sensory memory is created from such navigable path using the model of the scene. Similarly, during the *navigation stage*, the system repeats the completed servo tasks by comparing the feature vectors in the onboard measurements with those in the sensory memory.

4.2 Which navigation systems are suitable for SNA?

In order to answer such question, the elements of a navigation system and its representation in the context of SNA have to be defined. A system that belongs to SNA is composed by a **robot** equipped with an exteroceptive onboard **sensor** that takes measurements of the **scene** from which it navigates. Then, the system can be represented by the rigid motion of the sensor described by a pose ${}^{V}\boldsymbol{\xi}$, defined in the model of the scene $\mathcal{M}^{scene}$, and the models of the scene and sensor, $\mathcal{M}^{scene}$ and $\mathcal{M}^{sensor}$ respectively. This representation has a great impact in the mission's performance, explained hereafter. As long as the robot's dynamics can be accurately represented by its model, any robot can be considered suitable for SNA.

Now, the question can be reformulated as follows: which aspects have to be considered to determine if the sensor's model $\mathcal{M}^{sensor}$ and scene's model $\mathcal{M}^{s}cene$ are suitable for SNA? Indeed, there are three main aspects to consider. *Firstly*, similar to model-based systems [Güzel (2013)], SNA relies on the accuracy of the model $\mathcal{M}$ w.r.t. the scene to extract a sequence of landmarks expected to be found over the scene when navigating. More precisely, the feature vectors $\mathbf{s}$ should be similarly located in the onboard sensor's measurements and in the rendered ones. Therefore, three conditions have to be met: the sensor should be calibrated to extract enough $\mathbf{s}$ from the scene, the model of the sensor $\mathcal{M}^{sensor}$ should be known, and the scene should be similar to its model $\mathcal{M}^{scene}$ regarding the sensor's space. Accordingly to this reason, Dense Surface Models, Digital Elevation Models, Computer-Aided Design (CAD) models and occupancy maps are considered suitable for SNA. *Secondly*, the scene and its model $\mathcal{M}^{scene}$ are required to be static. The main reason is that a global planner, in the sense of Goal Oriented Navigation Systems (GONS), is considered at the path-planning stage and no local planner acts during the navigation stage. Therefore, the navigation is planned once and no re-planning is considered. For this reason, odor maps, and by consequence odor sensors, are considered not suitable for SNA as the distribution of the gas, which describes the scene in the odor sensor's space, varies in time [Marjovi and Marques (2014)]. *Thirdly*, exteroceptive sensors that meet the condition of stability of sensor-based control, in Eq. (2.18), are considered to be suitable for SNA. Typical exteroceptive sensors used for navigation are cameras

[Bonin-Font et al. (2008)], microphones [Magassouba et al. (2018)], and range-based sensors [Urmson et al. (2003) & Soares et al. (2013)], e.g. ultrasonic sensors and Light Detection and Ranging sensors.

Summarizing, some suitable systems for SNA are: ground and aerial vehicles equipped with cameras that navigate over an outdoor environment represented by a Dense Surface Model; underwater robots equipped with ultrasonic sensors that perceive the bottom of the sea represented by a Digital Elevation Model; ground robots equipped with range-based sensors or cameras that navigate in indoor environment represented by a CAD model or occupancy maps (range-based sensors only); ground robots equipped with microphones that navigate through an environment with different sound sources (although theoretically because the scene is difficult to model due to the presence of noise and reverberating [Magassouba et al. (2018)]); or even arm-type robots equipped with a conventional camera that navigate over a planar textured surface represented by an image.

4.3 Positioning Tasks

SNA conceives the navigation as a concatenation of positioning tasks. Whithin SNA context, there are two types of positioning tasks: **hybrid positioning task** and **virtual positioning task**.

The hybrid positioning task is performed by the robot. The main difference with a traditional (here named **real**) positioning task is that the desired feature vector $\mathbf{s}^*$ is extracted and generated from the models of the scene and sensor, $\mathcal{M}^{scene}$ and $\mathcal{M}^{sensor}$ respectively. The virtual positioning task is a simulated positioning task using $\mathcal{M}^{scene}$ and $\mathcal{M}^{sensor}$.

Several control laws had been used to compute the input velocity $\mathbf{u} = \dot{\boldsymbol{\xi}}$ of a servo task in the literature (Chapter 2.2.1). Although any control law can be used in SNA to compute $\dot{\boldsymbol{\xi}}$, the Gauss-Newton optimization method is chosen to regulate the error $\mathbf{e}$ to zero. The main reason of this choice is that this method requires the computation of the interaction matrix of the desired feature vector $\mathbf{L}_{\mathbf{s}^*}$ once, instead of the computation of $\mathbf{L}_{\mathbf{s}}$ at each iteration (Chapter 2.2.1). Saving computation is recommended as several virtual positioning tasks are sometime expected, depending on the *mission*.

4.3.1 Hybrid Positioning Task

The **hybrid positioning task** is used to calculate the number of nodes $|\mathcal{N}|$ in the navigation graph used for the *planning initialization module* (Chapter 4.4.1) and to navigate during the *hybrid servo navigation module* (Chapter 4.4.2).

Let ${}^W\boldsymbol{\xi} = [{}^W\mathbf{X}^\top, {}^W\boldsymbol{\Theta}^\top]^\top$, where ${}^W\mathbf{X} = [{}^WX, {}^WY, {}^WZ]^\top$ is the position and ${}^W\boldsymbol{\Theta} = [{}^W\Theta, {}^W\Phi, {}^W\Psi]^\top$ the orientation, and ${}^V\boldsymbol{\xi}^* = [({}^V\mathbf{X}^*)^\top, ({}^V\boldsymbol{\Theta}^*)^\top]^\top$, where ${}^V\mathbf{X}^* = [{}^VX^*, {}^VY^*, {}^VZ^*]^\top$ is the position and ${}^V\boldsymbol{\Theta}^* = [{}^V\Theta^*, {}^V\Phi^*, {}^V\Psi^*]^\top$ the orientation, be a desired pose of the sensor in the model's virtual frame ${}^V\mathfrak{F}$. Then, beginning at an initial pose ${}^W\boldsymbol{\xi}(0)$, the input velocity $\mathbf{u} = {}^W\dot{\boldsymbol{\xi}}$ applied to the robot is computed from:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^+_{\mathbf{s}({}^V\boldsymbol{\xi}^*)}(\mathbf{s}({}^W\boldsymbol{\xi}(t)) - \mathbf{s}({}^V\boldsymbol{\xi}^*)). \tag{4.1}$$

During the *convergence domain experimental evaluation* (Chapter 4.4.1), the task is considered completed if the **hybrid evaluation criteria** are met: $\|{}^W\mathbf{X}(t) - {}^V\mathbf{X}^*\| < \epsilon_d$, $|{}^W\Theta(t) - {}^V\Theta^*| < \epsilon_o$, $|{}^W\Phi(t) - {}^V\Phi^*| < \epsilon_o$, $|{}^W\Psi(t) - {}^V\Psi^*| < \epsilon_o$ and $t < \epsilon_t$, where ϵ_d, ϵ_o and ϵ_t are the thresholds on the distance, orientation and time-steps, respectively. However, since no localization system is suppossed to be available during the *navigation stage*, the task is considered completed if the **navigability cirteria** are met: $\|{}^W\dot{\boldsymbol{\xi}}\| < \epsilon_v$ and $t < \epsilon_t$, where ϵ_v is the threshold on the norm of the input velocity. In both criteria, the condition $t < \epsilon_t$ implies that the pose of the robot ${}^W\boldsymbol{\xi}$ should converge to a desired pose ${}^W\boldsymbol{\xi}^* \approx {}^V\boldsymbol{\xi}^*$, implicitely embodied in $\mathbf{s}({}^V\boldsymbol{\xi}^*)$, before ϵ_t timesteps. If the criteria are not met, the robot is considered to had diverged from ${}^W\boldsymbol{\xi}^*$.

4.3.2 Virtual Positioning Task

A **virtual positioning task** is used to evaluate an edge of the navigation graph during the *navigability evaluation module* (Chapter 4.4.1). This process consists into updating the pose of the robot in the virtual model frame ${}^V\boldsymbol{\xi} = [{}^V\mathbf{X}^\top, {}^V\boldsymbol{\Theta}^\top]^\top$, where ${}^V\mathbf{X} = [{}^VX, {}^VY, {}^VZ]^\top$ is its position and ${}^V\boldsymbol{\Theta} = [{}^V\Theta, {}^V\Phi, {}^V\Psi]^\top$ its orientation, from an initial pose ${}^V\boldsymbol{\xi}(0)$, by computing its time derivative from:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^+_{\mathbf{s}({}^V\boldsymbol{\xi}^*)}(\mathbf{s}({}^V\boldsymbol{\xi}(t)) - \mathbf{s}({}^V\boldsymbol{\xi}^*)), \tag{4.2}$$

where $\mathbf{u} = {}^V\dot{\boldsymbol{\xi}}$.

A virtual positioning task is considered completed if the **virtual evaluation criteria** are met: $\|{}^V\mathbf{X}(t) - {}^V\mathbf{X}^*\| < \epsilon_d$, $|{}^V\Theta(t) - {}^V\Theta^*| < \epsilon_o$, $|{}^V\Phi(t) - {}^V\Phi^*| < \epsilon_o$, $|{}^V\Psi(t) - {}^V\Psi^*| < \epsilon_o$ and $t < \epsilon_t$. Notice that the this criteria is similar to the corresponding one in Chapter 4.3.1 but using ${}^V\boldsymbol{\xi}$ instead of ${}^W\boldsymbol{\xi}$.

4.4 Mission of SNA

The *mission* starts with the set of world goal poses ${}^{W}\boldsymbol{\xi}^{G} \in {}^{W}\Xi^{G}$, which the robot must reach, defined in the world frame ${}^{W}\mathfrak{F}$, i.e. over the scene. Figure 4.1 displays one example of a *mission* where ${}^{W}\Xi^{G} = \{{}^{W}\boldsymbol{\xi}_0^{G}, {}^{W}\boldsymbol{\xi}_1^{G}, {}^{W}\boldsymbol{\xi}_2^{G}\}$ is defined over the scene.

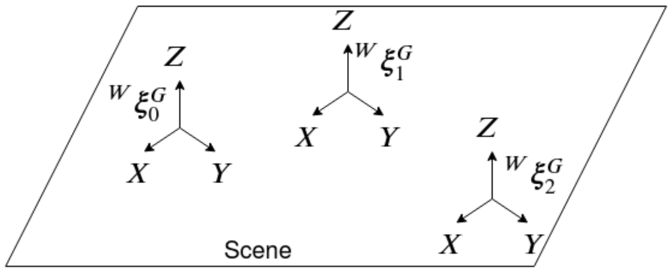

Z
${}^{W}\boldsymbol{\xi}_0^{G}$
X
Y
Z
${}^{W}\boldsymbol{\xi}_1^{G}$
X
Y
Z
${}^{W}\boldsymbol{\xi}_2^{G}$
X
Y
Scene

Figure 4.1: Example of a mission defined by three world goal poses over the scene.

The goal of a *mission* is to reach the set of world goal poses ${}^{W}\boldsymbol{\xi}^{G} \in {}^{W}\Xi^{G}$ by only relying on the sensory path $\mathcal{S}^*$. Such path is stored in the memory which is generated from the preobtained model of the scene $\mathcal{M}$. Fig. 4.2 shows the control decomposition of SNA.

The first *stage* of the *mission* is composed by six *modules*: *planning initialization*, *path search*, *virtual sensory memory generation*, *navigability evaluation*, *graph update* and *sensory memory generation*. The second *stage* is composed by two *modules*: *sensory memory access* and *hybrid servo navigation*.

The rest of the Chapter describes the process of the *path-planning* and *navigation stages*.

4.4.1 Path-Planning Stage

The *path-planning stage* uses the model of the scene $\mathcal{M}$, the sensor function$\mathcal{F}$ and a pose ${}^{V}\boldsymbol{\xi}$ in a topological approach to evaluate the navigability of a path $\mathcal{N}^*$ proposed by A* (Chapter 4.4.1). Roughly speaking, when $\mathcal{N}^*$ is considered navigable, its sensory path $\mathcal{S}^*$ is stored in the sensory memory. The *planning initialization module* is composed by three *phases*. Firstly, a *convergence domain experimental evaluation* allows the determination of the initial number of nodes $|\mathcal{N}|$ in the navigation graph $\mathcal{G}$ for each pair of consecutive world goal poses $\{{}^{W}\boldsymbol{\xi}^{G}{}_{i}, {}^{W}\boldsymbol{\xi}^{G}{}_{i+1}\} \subset {}^{W}\Xi^{G}$. Secondly, the *space tessellation* consists in creating a discrete set of search poses in the virtual model ${}^{V}\Xi^{S}$ from a continuous space ${}^{V}\mathbb{X}^{S}$ called the virtual search space. Thirdly,

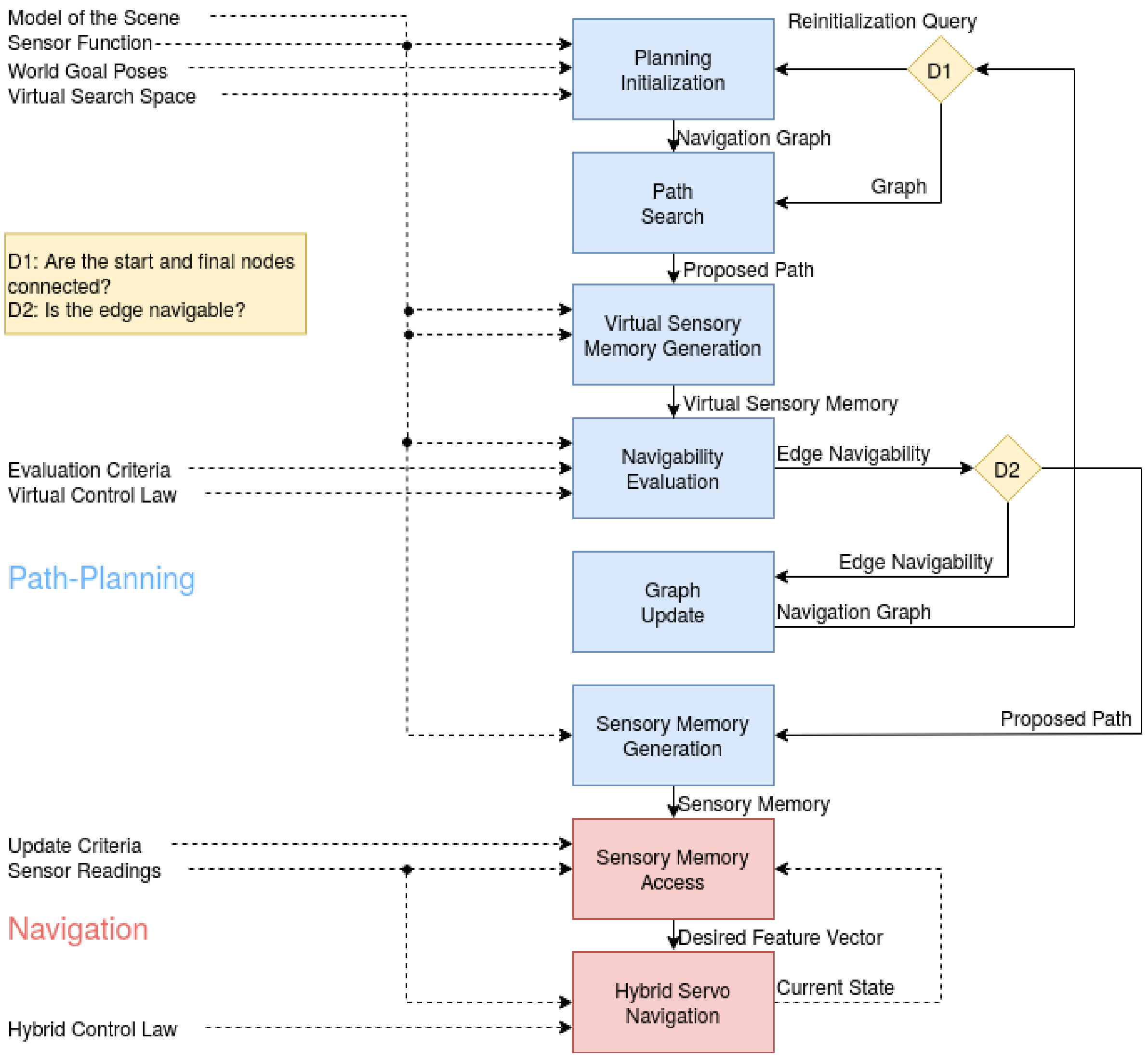


Figure 4.2: Control decomposition of the mission. The letter “D” in the diamond blocks stands for decision.

the poses ${}^{V}\boldsymbol{\xi}^{S} \in {}^{V}\Xi^{S}$ are assigned to the nodes $n \in \mathcal{N}$ in the navigation graph $\mathcal{G}$ during the *graph initialization*, with the start pose ${}^{V}\boldsymbol{\xi}^{G}{}_{s} = {}^{W}\boldsymbol{\xi}^{G}{}_{i}$ in the model's virtual frame ${}^{V}\mathfrak{F}$ related to the start node n_s and the final pose ${}^{V}\boldsymbol{\xi}^{G}{}_{f} = {}^{W}\boldsymbol{\xi}^{G}{}_{i+1}$ related to the final node n_f. From the navigation graph $\mathcal{G}$, A* proposes the path $\mathcal{N}^*$ and the navigability of the proposed edge $e^{P}_{j,j+1}$, related to each two consecutive nodes in the path $\{n^*_j, n^*_{j+1}\} \subset \mathcal{N}^*$, is evaluated. The latter process is repeated until a navigable path is found or there are no more paths that connect the start node n_s with the final one n_f. If such case happens, $|\mathcal{N}|$ is increased by one in each dimension of the navigable graph, the set of virtual search poses ${}^{V}\Xi^{S}$ is updated, and the navigable path-search process is repeated. The *path-planning stage* finishes when a navigable path is found for each two consecutive goal poses $\{{}^{W}\boldsymbol{\xi}^{G}{}_{i}, {}^{W}\boldsymbol{\xi}^{G}{}_{i+1}\} \subset {}^{W}\Xi^{G}$ and the sensory path $\mathcal{S}^*$ is generated from the poses related to the path that goes over ${}^{W}\Xi^{G}$. The possibility of increasing the density of the navigation graph $\mathcal{G}$ and the virtual search space ${}^{V}\mathbb{X}^{S}$ allows to find a solution, even if the initial $|\mathcal{N}|$ was determined too optimitically. Such optimism is, however, considered to directly short for the planning process with the expected adpted density of the navigation graph $\mathcal{G}$ and the set of virtual search poses ${}^{V}\Xi^{S}$, to save time.

The *modules* that compose this stage and the generation of the sensory memory are depicted hereafter.

Planning Initialization Module

The goal of this *module* is to initialize the navigation graph $\mathcal{G}$ for each pair of consecutive world goal poses $\{{}^{W}\boldsymbol{\xi}^{G}{}_{i}, {}^{W}\boldsymbol{\xi}^{G}{}_{i+1}\} \subset {}^{W}\Xi^{G}$; either by means of the convergence domain experimental evaluation or by the request for a graph reinitialization (**Decision 1**, Fig. 4.2). This *module* is composed by three *phases* whereas just the first one, i.e. *convergence domain experimental evaluation*, is required to be carried out only once per consecutive pair of two world goal poses $\{{}^{W}\boldsymbol{\xi}^{G}{}_{i}, {}^{W}\boldsymbol{\xi}^{G}{}_{i+1}\} \subset {}^{W}\Xi^{G}$.

Convergence Domain Experimental Evaluation Phase: The convergence domain experimental evaluation estimates the largest hybrid domain of convergence ${}^{H}\mathcal{C}$ that surrounds the central ${}^{W}\boldsymbol{\xi}^{C} = [{}^{W}\boldsymbol{\xi}^{G}_{i} + {}^{W}\boldsymbol{\xi}^{G}_{i+1}]/2$, so an initial tessellation of the navigation graph $\mathcal{G}$ can be carried out. ${}^{W}\boldsymbol{\xi}^{C}$ is chosen so the number of nodes $|\mathcal{N}|$ in the navigation graph $\mathcal{G}$ can be optmimally determined from this evaluation. Nevertheless, nothing guarantees that the extent of ${}^{H}\mathcal{C}$ around ${}^{W}\boldsymbol{\xi}^{C}$ is similar anywhere else over the scene. However, for the case where $|\mathcal{N}|$ is underestimated from this evaluation, Desicion 1 will request to increase the density in the navigation graph $\mathcal{G}$. Analogously to the determination of the borders of the domain of convergence

$\mathcal{C}$ of any kernel-based method [Chesi and Hashimoto (2010)], the borders of ${}^{H}\mathcal{C}$ are experimentally determined from hybrid positioning tasks. In these tasks, the input velocity ${}^{W}\dot{\boldsymbol{\xi}}$ is computed from Eq. (4.1), with $\mathbf{s}({}^{V}\boldsymbol{\xi}^{*}) = \mathbf{s}({}^{V}\boldsymbol{\xi}^{C})$ and ${}^{V}\boldsymbol{\xi}^{C}$ equivalent to ${}^{W}\boldsymbol{\xi}^{C}$ but defined over the model $\mathcal{M}$. Then, for each direction $j \in [\![0, 3^{m} - 1]\!]$, where $m = 1, 2, 3$ is the dimension of the navigation graph $\mathcal{G}$ (Chapter 2.3.1), regularly surrounding the central pose ${}^{W}\boldsymbol{\xi}^{C}$, hybrid positioning tasks are executed using the same DoF as those used at the navigation stage. The evaluation consists in running the hybrid positioning tasks from initial poses $({}^{W}\boldsymbol{\xi}_{j})$ farther and farther from ${}^{W}\boldsymbol{\xi}^{C}$, with step $\delta \in \mathbb{R}_{+}^{*}$, along direction j, until it fails to converge. Poses ${}^{W}\boldsymbol{\xi}_{j}$ furthest from ${}^{W}\boldsymbol{\xi}^{C}$, in every direction, determine the borders of ${}^{H}\mathcal{C}$. Distances ${}^{H}d_{j}$, from ${}^{W}\boldsymbol{\xi}^{C}$ to the borders of ${}^{H}\mathcal{C}$, are computed for every direction j by ${}^{H}d_{j} = \|{}^{W}\mathbf{X}_{j} - {}^{W}\mathbf{X}_{c}\|$, ${}^{W}\mathbf{X}_{j}$ being the position of the furthest pose ${}^{W}\boldsymbol{\xi}_{j}$ from ${}^{W}\boldsymbol{\xi}^{C}$. Gathering distances ${}^{H}d_{j}$ in set ${}^{H}\mathcal{D}$, ${}^{H}d_{min}$ is defined as the minimum element of ${}^{H}\mathcal{D}$.

Space Tessellation Phase: Once the hybrid minimum distance ${}^{H}d_{min}$ is computed, the space tessellation of the virtual search space ${}^{V}\mathbb{X}^{S}$ is executed. ${}^{V}\mathbb{X}^{S} \subset \mathbb{R}^{3}$ is tessellated in $m = 1, 2, 3$ dimensions. This tessellation discretizes ${}^{V}\mathbb{X}^{S}$ into a set of 3D locations whose, along with a constant orientation vector ${}^{V}\boldsymbol{\Theta}^{S}$, compose the set of search poses ${}^{V}\boldsymbol{\Xi}^{S}$ in the virtual model of the scene. Let $\mathcal{G} = (\mathcal{G}, \mathcal{E})$ be a linear, squared or cubic lattice graph depending on the choice of $m = 1, 2, 3$. Then, for a given pair of two consecutive world goal poses ${}^{W}\boldsymbol{\xi}_{i}^{G} = [({}^{W}\mathbf{X}_{i}^{G})^{\top}, ({}^{W}\boldsymbol{\Theta}_{i}^{G})^{\top}]^{\top}$ and ${}^{W}\boldsymbol{\xi}_{i+1}^{G} = [({}^{W}\mathbf{X}_{i+1}^{G})^{\top}, ({}^{W}\boldsymbol{\Theta}_{i+1}^{G})^{\top}]^{\top}$, the number of nodes in $\mathcal{G}$ is computed from:

$$|\mathcal{N}| = \left(\left\lceil \frac{\|{}^{W}\mathbf{X}_{i}^{G} - {}^{W}\mathbf{X}_{i+1}^{G}\|}{{}^{H}d_{min}} \right\rceil + 1 \right)^{m}. \tag{4.3}$$

If the navigation graph $\mathcal{G}$ requires to be reinitialized (Decision 1, 4.2), the number of vertices per sides increases by one in $\mathcal{G}$.

Navigation Graph Initialization Phase: The lattice graph $\mathcal{G}$ is *transformed* into a graph of poses by assigning the set of search poses ${}^{V}\boldsymbol{\Xi}^{S}$ to the set of nodes $\mathcal{N}$. In other words, a pose ${}^{V}\boldsymbol{\xi}^{S} \in \mathbb{R}^{6}$ is assigned to each node of a linear, squared or cubic lattice graph where the location ${}^{V}\mathbf{X}^{S}$ of the pose maps the node. The start node n_{s} is related to ${}^{V}\boldsymbol{\xi}_{i}^{G}$ and the final node n_{f} to ${}^{V}\boldsymbol{\xi}_{i+1}^{G}$, where ${}^{V}\boldsymbol{\xi}_{i}^{G}$ and ${}^{V}\boldsymbol{\xi}_{i+1}^{G}$ are the equivalent of ${}^{W}\boldsymbol{\xi}_{i}^{G}$ and ${}^{W}\boldsymbol{\xi}_{i+1}^{G}$ but defined in the virtual model. Then, $n \in \mathcal{N}$ is simply connected, transforming $\mathcal{G}$ into a kinggraph-m of poses. Lastly, the weight of an edge is determined by the gamma function $\gamma(\cdot)$, transforming $\mathcal{G}$ into a weighted kinggraph-m of poses. Let two connected nodes n_{p} and n_{q} be related to the search poses ${}^{V}\boldsymbol{\xi}_{p}^{S} = [({}^{V}\mathbf{X}_{p}^{S})^{\top}, ({}^{V}\boldsymbol{\Theta}_{p}^{S})^{\top}]^{\top}$ and ${}^{V}\boldsymbol{\xi}_{q}^{S} = [({}^{V}\mathbf{X}_{q}^{S})^{\top}, ({}^{V}\boldsymbol{\Theta}_{q}^{S})^{\top}]^{\top}$, respectively. Then, the weight γ of $e_{p,q}$ is computed from the Euclidean distance:

$$\gamma(e_{p,q}) = \|{}^{V}\mathbf{X}_q^S - {}^{V}\mathbf{X}_p^S\|. \tag{4.4}$$

Path-Search Module

At this point, the *path-search module* uses A* as graph-search algorithm to propose the path $\mathcal{N}^* = \{n_0^*, n_1^*, \ldots, n_{|\mathcal{N}^*|-1}^*\}$ of nodes that connect $n_s = n_0^*$ with $n_f = n_{|\mathcal{N}^*|-1}^*$, where $|\mathcal{N}^*|$ is the number of nodes in the set and $\mathcal{N}^* \subset \mathcal{N}$. This algorithm is prefered over Dijkstra [Dijkstra (1959)] due to the drastic reduction of the vertex expansion explained in Chapter 2.3.3. A* uses the gama function $\gamma(\cdot)$ as the **heuristic** to propose $\mathcal{N}^*$. Then, the naviagability of $\mathcal{N}^*$ is evaluated.

Virtual Sensory Memory Generation Module

Let ${}^{V}\boldsymbol{\xi}^P$ be a proposed pose in the virtual model $\mathcal{M}$ that belongs to the set ${}^{V}\Xi^P \subset {}^{V}\Xi^S$ related to the path $\mathcal{N}^*$ proposed in the last *module*. In such path, ${}^{V}\boldsymbol{\xi}_i^G = {}^{V}\boldsymbol{\xi}_0^S = {}^{V}\boldsymbol{\xi}_0^P$ is assigned to n_0^* and ${}^{V}\boldsymbol{\xi}_{i+1}^G = {}^{V}\boldsymbol{\xi}_{|\mathcal{N}|-1}^S = {}^{V}\boldsymbol{\xi}_{|\mathcal{N}^*|-1}^P$ to $n_{|\mathcal{N}^*-1|}^*$, where $\{{}^{V}\boldsymbol{\xi}_i^G, {}^{V}\boldsymbol{\xi}_{i+1}^G\} \subset {}^{V}\Xi^G$. Then, a **virtual sensory memory** is generated from the sensory path $\mathcal{S}^* = \{\mathbf{s}({}^{V}\boldsymbol{\xi}_0^P), \mathbf{s}({}^{V}\boldsymbol{\xi}_1^P), \ldots, \mathbf{s}({}^{V}\boldsymbol{\xi}_{|\mathcal{S}^*-1|}^P)\}$, where $|\mathcal{S}^*| = |\mathcal{N}^*|$. The virtual sensory memory is used to evaluate the navigability of the proposed path $\mathcal{N}^*$ in the next *module*.

Navigability Evaluation Module

The navigability evaluation of $\mathcal{N}^*$, related to the set of proposed poses ${}^{V}\Xi^P$ in the virtual model, is evaluated by a group of successive virtual positioning tasks using the virtual sensory memory as reference (Chapter 4.3.2). Being ${}^{V}\boldsymbol{\xi}_k^P$ and ${}^{V}\boldsymbol{\xi}_{k+1}^P$ a pair of two consecutive proposed poses $\{{}^{V}\boldsymbol{\xi}_k^P, {}^{V}\boldsymbol{\xi}_{k+1}^P\} \subset {}^{V}\Xi^P$ defined in the virtual $\mathcal{M}$, the virtual positioning task related to the proposed edge $e_{j,j+1}^P$ is executed by updating ${}^{V}\boldsymbol{\xi}$, from an initial pose ${}^{V}\boldsymbol{\xi}(0) = {}^{V}\boldsymbol{\xi}_k^P$, from Eq. (4.2), with ${}^{V}\boldsymbol{\xi}^* = {}^{V}\boldsymbol{\xi}_{k+1}^P$. If a navigable path had been found for the entire set of goal poses in the virtual model ${}^{V}\Xi^G$ (**Decision 2**, Fig. 4.2), continue to the *sensory memory generation module*. If the virtual positioning task fails, go to the *graph update module*; otherwise, continue to the next pair $\{{}^{V}\boldsymbol{\xi}_{k+1}^P, {}^{V}\boldsymbol{\xi}_{k+2}^P\} \subset {}^{V}\Xi^P$, if any.

Graph Update Module

If any proposed edge e^P is evaluated as non-navigable, disconnect it from $\mathcal{G}$. Then, if not a single path connects n_s with n_f in $\mathcal{G}$, a **graph reinitialization query** is sent to the *path-planning initialization module* (Chapter 4.4.1).

Sensory Memory Generation Module

At this point, $\mathcal{S}^*$ that composes the virtual sensory memory produced a successfully virtual navigation from ${}^V\boldsymbol{\xi}_i^G$ to ${}^V\boldsymbol{\xi}_{i+1}^G$ during the *navigability evaluation module*. If ${}^V\boldsymbol{\xi}_i^G = {}^V\boldsymbol{\xi}_0^G$, store the entire $\mathcal{S}^*$ in the sensory memory. Otherwise, enqueu $\mathcal{S}^*$ except for the first element, i.e. $\{\mathbf{s}_1^*, \mathbf{s}_2^*, \ldots, \mathbf{s}_{|\mathcal{S}^*-1|}^*\}$, in the memory. This measure has to be considered in order to avoid the overlaping of desired feature vectors $\mathbf{s}^*$ caused by ${}^V\boldsymbol{\xi}_{i+1}^G$ when ${}^V\boldsymbol{\Xi}^G$ contains more than 2 poses. The sensory memory is ready to be used during the *navigation stage*.

4.4.2 Navigation Stage

The goal of this stage is to successively reach each one of the world goal poses ${}^W\boldsymbol{\xi}^G \in {}^W\boldsymbol{\Xi}^G$ by performing hybrid positioning tasks, aided by the sensory path $\mathcal{S}^*$ stored in the memory.

Figure 4.3 displays the navigation stage as a control loop system.

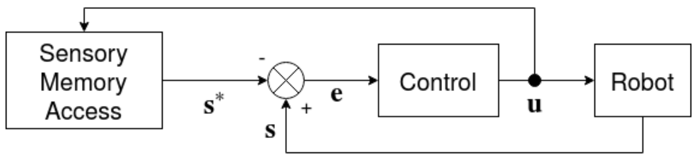


Figure 4.3: Control loop system representation of the navigation stage.

Sensory Memory Access Module

The *sensory memory access* module determines the element in the memory $\mathbf{s}^* \in \mathcal{S}^*$ to be compared with $\mathbf{s}({}^W\boldsymbol{\xi}(t))$.

Beginning with $\mathbf{s}_0^*$, this module updates the l-th element in the memory $\mathbf{s}_l^*$ to $\mathbf{s}_{l+1}^*$, when the **update criterion** is satisfied:

$$\|{}^{W}\dot{\boldsymbol{\xi}}\| < \epsilon_v, \tag{4.5}$$

where ϵ_v is the same threshold used for the **navigability criteria** (Chapter 4.3.1), as long as $l < |\mathcal{S}^*| - 1$. If the current element in the memory is the last one, i.e. $l = |\mathcal{S}^*| - 1$, and Eq (4.5) is satisfied, the *navigation stage* and *mission* are considered completed.

Hybrid Servo Navigation Module

The term **hybrid servo navigation** refers to the concatenation of successive hybrid positioning tasks that cause the robot to navigate over the scene. The *hybrid servo navigation module* is initialized by placing the robot in such way that $\mathbf{s}({}^{W}\boldsymbol{\xi}) \approx \mathbf{s}_0^*$. Then, the navigation proceeds as a concatenation of hybrid servo tasks (Chapter 4.3.1) considering $\mathbf{s}({}^{V}\boldsymbol{\xi}^*) = \mathbf{s}_l^*$, where l is the index of the desired feature vector in $\mathcal{S}^*$, in Eq (4.1). The input velocity $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is sent to the *sensory memory access module* (Chapter 4.4.2) to determine if the *mission* is completed or the memory should be updated. If the latter is true, $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is sent to the robot to navigate.

4.5 Conclusions

This Chapter presents a framework where the concept of navigability is strongly related with perception. Such framework is suitable for various robots, sensors, environments and models, depending on the definition of the task.

The mission of SNA is composed by two stages. The first stage is in charge of generating a sensory memory by executing virtual navigations over the model of the scene. These virtual navigations are related to a path proposed by A* and executed over the navigation graph. Since the goal poses are known, A* is prefered over Dijsktra's algorithm to save computation. Once a virtual navigation is considered navigable, the sensory memory is generated from it. The second stage is in charge of the hybrid navigation. It begins by placing the robot, e.g. manually, near to the first goal pose. Then, the elements in the sensory memory are sequentially compared with the onboard measurements.

If the model of the scene is accurate, the hybrid navigation is ensured. Nevertheless, one drawback of SNA is that the speed is reduced almost to zero each time that the control law converges. Although this characteristic is common and desired for positioning tasks, it is not ideal for autonomous navigation. Such problem remains open for study.

The next Chapter presents the instantiation of SNA for vision-based navigation in order to evaluate the performance of the framework. Moreover, its implementation favors the study of SNA by reducing the number of constraints, e.g. image latency.

Chapter 5

Photometric Navigation System

This Chapter focuses on the development of an instantiation of the Servo Navigation Architecture (SNA), called Photometric Navigation System (PNS). PNS is a Vision-Based Navigation System (VBNS) because it uses a conventional camera to navigate. Within the VBNS' classes, PNS is considered as appearance and model-based navigation system as it uses both to generate its memory. Once the memory is generated, the PNS employs it in appearance-based navigation through the scene. Such system is the second contribution presented in the thesis.

The rest of the Chapter is structured as follows: PNS, the concepts that it involves and its main characteristics are introduced in Section 5.1; the feature implemented in PNS is presented in Section 3.3 as well as its advantages w.r.t. other features and similarity measurements in the state-of-the-art; a modification of the Photometric Visual Servoing (PVS) implemented in the system is presented in Section 5.2 the structure of the mission of PNS is explained in Section 5.3 based on the corresponding of SNA; the chapter concludes with a recap in Section 5.4

5.1 System Overview

PNS inherits its name from the Photometric Visual Servoing (PVS) (Chapter 2.2.2) as it uses a modification of this technique, called Extended PVS (Chapter 5.2), to perform servo tasks. In this system, a **textured mesh** (instantiation of the model of the scene $\mathcal{M}$) represents the scene where the navigation is carried out. The mesh can be generated from the data of a sensor different to the one used to navigate. Furthermore, as the **visual path** $\tilde{\mathcal{I}}^*$ (instantiation of the **sensory path** $\mathcal{S}^*$) is computed offline, the illumination in the scene and in the textured mesh may be different during the *path-planning stage* and the *navigation stage*. Therefore, the system requires a feature able to cope with these situations. The Zero-mean Normalization (ZN) of the pixel intensities has been used in the literature to deal with light variations (Chapter 3.3.2). Therefore, the Extended PVS considers a transformation of the luminance feature $\mathbf{I}$, called Zero-mean Normalized (ZN) luminance feature $\tilde{\mathbf{I}}$. Thus, the **visual path** is composed by an ordered set of features: $\tilde{\mathcal{I}}^* = \{\tilde{\mathbf{I}}^*_0, \tilde{\mathbf{I}}^*_1, \ldots, \tilde{\mathbf{I}}^*_{|\mathcal{I}|-1}\}$. PNS is developed according to SNA; therefore, its *mission* proceeds similarly to the one of SNA but some components and techniques employ vision for robotic perception.

Consider the following scenario of a mission in SNA context:

"The *mission* starts by defining a set of world goal poses ${}^{W}\mathbf{\Xi}$ over the **scene**.

During the *path-planning stage*, a **sensory path** $\mathcal{S}^*$ is stored in the **sensory memory**. Such path $\mathcal{S}^*$ is determined by a sequence of **virtual servo positioning tasks**. These tasks are executed in a metric-topological approach using the **preobtained model** $\mathcal{M}$ and the **sensor function** $\mathcal{F}(\cdot)$.

During the *navigation stage*, the motion of the robot is computed from a control law which compares the **features s** in the **onboard measurements** with those in the **virtual measurements** stored in the memory."

An equivalent scenario for PNS is stated as follows:

"The *mission* starts by defining a set of world goal poses ${}^{W}\mathbf{\Xi}$ over a **textured scene**.

During the *path-planning stage*, a **visual path** $\tilde{\mathcal{I}}^*$ is stored in the **visual memory**. Such path $\tilde{\mathcal{I}}^*$ is determined by a sequence of **virtual extended photometric visual servoing positioning tasks**. These tasks are executed in a metric-topological approach using the **textured mesh** and the **perspective camera function** $\mathcal{P}(\cdot)$.

During the *navigation stage*, the motion of the robot is computed from a control law which compares the **ZN luminance features** $\tilde{\mathbf{I}}$ of the **onboard images** I^O with those of the **virtual images** I^V stored in the memory."

Notice that PNS is not the only possible implementetion of VBNS following SNA. For example, a 3D textured scene, a Dense Surface Model and an omnidirectional camera (Fig. 5.1) could be used instead of the textured scene, the textured mesh and the conventional camera.

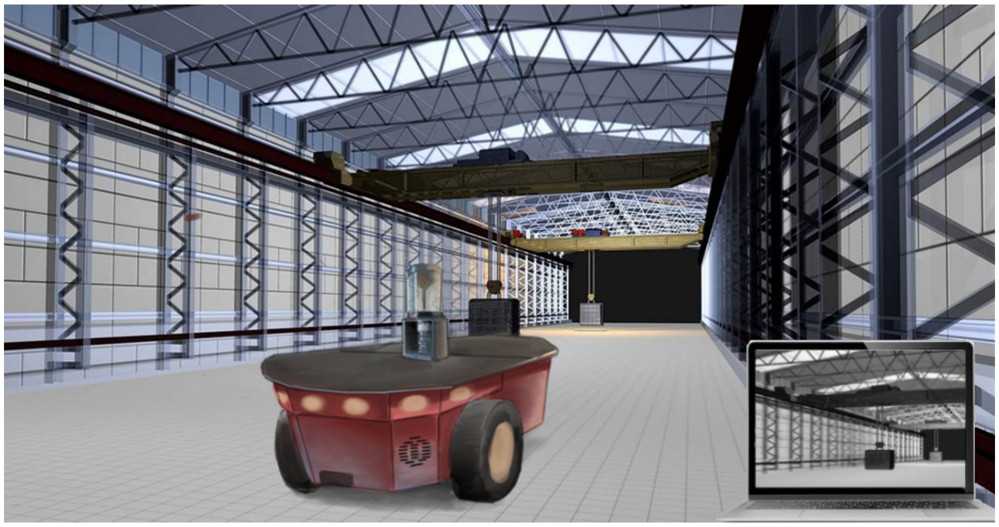


Figure 5.1: Example of another VBNS in SNA context. A terrestrial robot is navigating through a 3D textured scene using an omnidirectional camera.

However, the choice of the latter set-up is related to its practical implementation (Chapter 7).

Compared with other model-based navigation systems, PNS does not require to be localized in the scene while navigating. Indeed, the *navigation stage* is similar to the corresponding one of appearance-based navigation systems that uses a visual memory to navigate [Courbon et al. (2008) & Nguyen et al. (2014)]. The main difference w.r.t. the latter systems is that they adopt the Visual Teach & Repeat [Furgale and Barfoot (2010)] methodology, where a human expert proposes a path [Courbon et al. (2008)]. PNS autonomously generates its visual memory whitout relying in human experts. By not relying in Global Navigation Satellite System (GNSS) nor human experts, PNS is more autonomous than many VBNS in the literature.

Therefore, the system is suitable for applications where the GNSS fails (or it is too innacuarate). Moreover, by not relying on a human expert to find a navigable path *in situ*, the system is able to generate offline visual memories for missions with several world goal poses ${}^W\boldsymbol{\xi} \in {}^W\boldsymbol{\Xi}$.

The following Section is dedicated to the motivation of using the ZN luminance feature $\tilde{\mathbf{I}}$ rather than other features or similarities measurements already used in VBNS.

5.2 Extended Photometric Visual Servoing

Basically, the Extended PVS considers the ZN luminance feature $\tilde{\mathbf{I}}$ instead of $\mathbf{I}$:

$$\tilde{\mathbf{I}}(\boldsymbol{\xi}) = (\tilde{\mathbf{I}}_{1\bullet}, \tilde{\mathbf{I}}_{2\bullet}, \dots, \tilde{\mathbf{I}}_{N\bullet})^\top \tag{5.1}$$

where $\tilde{\mathbf{I}}_{i\bullet} \in \mathbb{N}^{1\times M}$ is the i-th line of the image, N is the width and M the height of the image, similarly to Eq. (2.21).

Therefore, the cost function is defined as:

$$C(\boldsymbol{\xi}) = \frac{1}{2}\|\tilde{\mathbf{e}}\|, \tag{5.2}$$

where $\tilde{\mathbf{e}}(\boldsymbol{\xi}) = \tilde{\mathbf{I}}(\boldsymbol{\xi}) - \tilde{\mathbf{I}}(\boldsymbol{\xi}^*)$ is the extended photometric error and $\tilde{\mathbf{I}}(\boldsymbol{\xi}^*) = \tilde{\mathbf{I}}^*$ is the desired ZN luminance feature.

Analogously to Eq. (2.17), the Gauss-Newton contol law that computes the input velocity $\mathbf{u} = \dot{\boldsymbol{\xi}}$ for PNS is:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^+_{\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)} \tilde{\mathbf{e}}, \tag{5.3}$$

where $\widehat{\mathbf{L}}^+_{\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)}$ is the Moore-Penrose pseudoinverse of the approximation of the interaction matrix $\mathbf{L}_{\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)} = -\nabla\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)^\top \mathbf{L}_{\mathbf{x}}$ related to $\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)$ when $\mathbf{L}_{\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)}$ is of full rank and $\nabla\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)$ is the gradient of $\tilde{\mathbf{I}}(\boldsymbol{\xi}^*)$.

Eq. (5.3) is chosen to be the general control law used in PNS.

5.3 Mission of PNS

The structure of the *missions* in PNS and SNA are similar. However, some elements are specific in PNS. This Section offers a description of the structure of the *mission* of PNS emphasizing such elements.

The *mission* starts by defining the set of world goal poses ${}^{W}\Xi^{G}$ in the and textured scene.

The goal of the *path-planning stage* is to generate a navigable visual path $\tilde{\mathcal{I}}^{*}$ and store it in the visual memory. During the *planning initialization module*, the navigation graph $\mathcal{G}$ is created for two consecutive world goal poses $\{{}^{W}\boldsymbol{\xi}_{i}^{G}, {}^{W}\boldsymbol{\xi}_{i+1}^{G}\} \in {}^{W}\Xi^{G}$. Firstly, the hybrid minimum distance ${}^{H}d_{min}$ is computed from the *convergence domain experimental evaluation phase* by performing **hybrid Extended PVS positioning tasks** over the textured scene. For this purpose, the central virtual image $I^{V} = I({}^{V}\boldsymbol{\xi}^{C})$ is rendered at the central location ${}^{V}\boldsymbol{\xi}^{C} = ({}^{V}\boldsymbol{\xi}_{i}^{G} + {}^{V}\boldsymbol{\xi}_{i+1}^{G})/2$ defined in the textured mesh. Analogously to SNA, a clever idea to determine the initial number of nodes $|\mathcal{N}|$ in the navigation graph $\mathcal{G}$ is to assume that the extent of the hybrid domain of convergence ${}^{H}\mathcal{C}$ at the central location ${}^{V}\boldsymbol{\xi}^{C}$ is similar than the corresponding of any pose ${}^{V}\boldsymbol{\xi}$ in the textured mesh. In other words, that the navigability is homogeneous within the textured mesh. However, if the initial number of nodes $|\mathcal{N}|$ in the navigation graph $\mathcal{G}$ is not enough, i.e. there is not a path that connects the nodes n_0 with $n_{|\mathcal{N}|-1}$, a reinitialization query of the navigation graph $\mathcal{G}$ is sent from the *graph update module* to the *planning initialization module*. Then, the motion of the robot is computed from the **hybrid control law for the Extended PVS**:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^{+}_{\tilde{\mathbf{I}}({}^{V}\boldsymbol{\xi}^{*})}(\tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}(t)) - \tilde{\mathbf{I}}({}^{V}\boldsymbol{\xi}^{*})) \tag{5.4}$$

and the extent of the hybrid domain of convergence ${}^{H}\mathcal{C}$ is determined by the hybrid evaluation criteria (Chapter 4.3.1) for the $3^{m}-1$ dimensions, where $m = 1, 2, 3$ is the dimension of the kinggraph-m. Secondly, the tessellation of the virtual search space ${}^{V}\mathbb{X}^{S}$ and the generation of the navigation graph $\mathcal{G}$ proceed exactly as the last two phases in Chapter 4.4.1. The *path-search module* proceeds exactly as in Chapter 4.4.1. Later, a visual path $\tilde{\mathcal{I}}^{*}$ related to the proposed path $\mathcal{N}^{*}$ is generated during the *virtual sensory memory generation module*. The visual path $\tilde{\mathcal{I}}^{*}$ is generated by rendering the virtual image I^{V} at the proposed pose ${}^{V}\boldsymbol{\xi}^{P}$ in the textured mesh related to a node in such path $n^{*} \in \mathcal{N}^{*}$. Then, a Zero-mean Normalization is applied to the luminance feature $\mathbf{I}$ obtained from the virtual image I^{V}.
The next *module* evaluates the navigability of the visual path $\tilde{\mathcal{I}}^{*}$ by executing a sequence of **virtual Extended PVS positioning tasks** for each two consecutive poses $\{{}^{V}\boldsymbol{\xi}_{j}^{P}, {}^{V}\boldsymbol{\xi}_{j+1}^{P}\}$ related to the nodes that compose the proposed path $n^{*} \in \mathcal{N}^{*}$ until one of these tasks fail. The input velocity $\mathbf{u} = {}^{V}\dot{\boldsymbol{\xi}}$ of the pose of the virtual camera is computed from the **virtual control law for the Extended PVS**:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^{+}_{\tilde{\mathbf{I}}({}^{V}\boldsymbol{\xi}^{*})}(\tilde{\mathbf{I}}({}^{V}\boldsymbol{\xi}(t)) - \tilde{\mathbf{I}}({}^{V}\boldsymbol{\xi}^{*})), \tag{5.5}$$

and the navigability of the proposed edge $e_{j,j+1}^{P}$ that connects two nodes in the proposed path $\{n_{j}^{*}, n_{j+1}^{*}\} \subset \mathcal{N}^{*}$ is evaluated from the **virtual evaluation criteria**

(Chapter 4.3.2). If any of these tasks fail, proceed to the *graph update module* (Chapter 4.4.1). Finally, store the naviable visual path $\tilde{\mathcal{I}}^*$ analogously to the storage of the sensory path in the *sensory memory generation module*.

Using the visual memory, the robot must successively reach each world goal pose ${}^{W}\boldsymbol{\xi}^{G}$ during the *navigation stage*. This stage starts by placing the robot near the first world global pose ${}^{W}\boldsymbol{\xi}_0^{G}$. Then, the elements stored in the visual memory $\tilde{\mathbf{I}}^* \in \tilde{\mathcal{I}}^*$ are updated by the *sensory memory access module*. Lastly, the robot performs hybrid servo navigation similarly to the last *module* of this *stage* by computing the input velocity $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ from Eq. (5.4).

At this point, PNS can generate a navigable visual path $\tilde{\mathcal{I}}^*$, from a textured mesh, given a certain *mission*. However, the *mission* is considered complete only if the PNS is controlled through the set of world goal poses ${}^{W}\boldsymbol{\Xi}^{G}$ using the visual path $\tilde{\mathcal{I}}^*$ stored in the visual memory and the onboard camera to navigate.

5.4 Conclusion

PNS is the improvement of the Photometric Teach & Repeat that inspired SNA as a generalization of it. In PNS and SNA, the autonomous navigation is ensured for *1* to *6 DoF* when the navigation space is correctly represented by its model. This represents an advantage over PT&R since the latter only ensures the autonomous navigation in *1 DoF*. Moreover, PNS is more robust to dissimilarities between the model and the scene since these are considered for generate the visual memory. This VBNS follows the structure of a mission defined in SNA context. In order to correctly study its behaviour and to avoid the problems that a commercial testbed may cause, e.g. image latency, it is recommendable to be implemented on a high precision robot. A static scene is also recommended for similar reasons before performing outdoor tasks on a mobile robot, which remains as a perspective for future research.

Now that PNS had been detailed, the next Chapter covers the experiments that test the performance of the system and some modifications of it.

Part III

PNS Experimentation

Chapter 6

Implementation

This Chapter is dedicated to evaluate the behavior of positioning tasks, mainly in term of convergence domain, to serve the Photometric Navigation System (PNS).

The first Section presents the experimental setup in which the Servo Navigation Architecture (SNA) is implemented as the Photometric Navigation System (PNS). The second Section experimentally evaluates the domains of convergence $\mathcal{C}$ in all three **configurations**, i.e. hybrid, real and virtual. The results of the latter evaluation are interpreted in the last Section.

6.1 Experimental Setup

The experimental setup (Fig. 6.1) is composed by a Stäubli TX2-60, which is a 6-axis robotic arm, an ISD uEye camera in an eye-in-hand configuration and a HP ZBook 15 computer as the workstation. The range of the robotic arm is 67 cm. This setup offers great precision (with a repeatability of ±0.02 mm) regarding to control, ground-truth and data acquisition. Thus, it is used for the determination of the extents of domains of convergence $\mathcal{C}$ and the PNS navigation trajectories evaluation (Chapter 7).

The Stäubli TX2-60 is a robotic arm designed for industrial applications. The end-effector can be controlled in 6 Degrees of Freedom (DoF) and a camera can be mounted on it. One of the applications for which it was designed is precise path-follow guided by a camera [1], making the robot ideal for the implementation of PNS. The robot can be manually controlled from its user's interface. However, it is wired through a cable for TCP/IP communication for the experiments. The camera's frame ${}^{C}\mathfrak{F}$ is selected from the user's interface so the input velocity $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is directly applied.

The IDS uEye UI-1250LE is a lightweight camera. Its frame rate is 17.6 fps and it produces an image of 1600×1200 pixels. The camera is connected to the computer through USB 2.0 port. This camera is mounted in the end-effector of the robotic arm (Fig. 6.1).

The ground station for this experiment is implemented in the HP ZBook 15 computer described in Chapter 3.5. It runs Ubuntu 16.04.6 LTS where the executable, that receives the onboard image ${}^{W}I$ and sends the input velocity $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ to the robotic arm, runs. The executable was coded in C++ and it uses the libraries of the Visual Servoing Platform (ViSP) [Marchand et al. (2005b)] and the driver[2] of the camera. Furthermore, the size of all the images are reduced to 640×512 pixels.

[1]User's guide at `https://www.staubli.com/en-us/file/21193.show`

[2]Driver at `https://en.ids-imaging.com/download-details/AB00353.html`

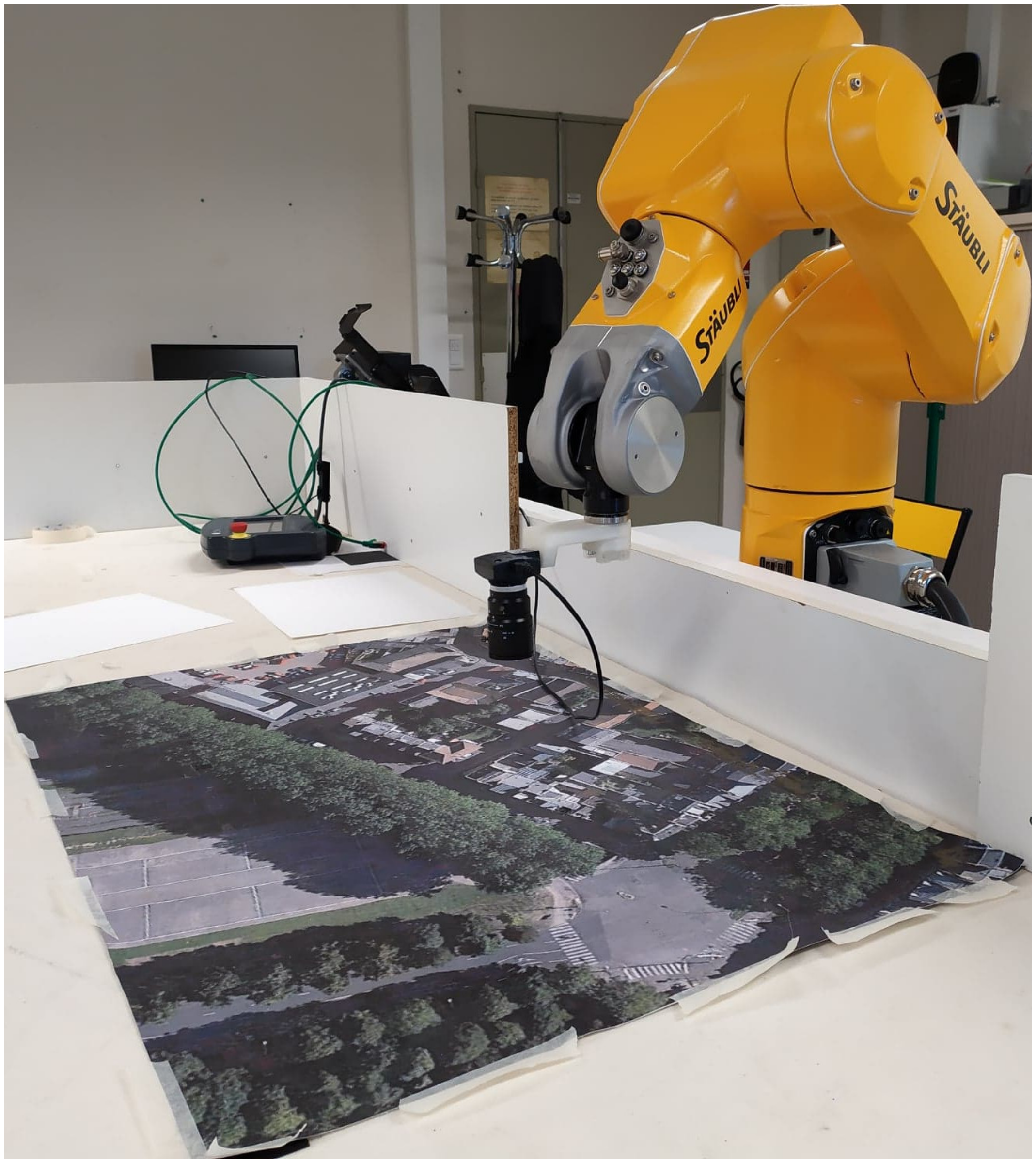


Figure 6.1: Experimental setup. The uEye camera is mounted on the end-effector of the robot. The latter travels through the planar textured scene.

6.2 Experimental Evaluation of the Domains of Convergence

A set of experiments determine the extent of the domain of convergence $\mathcal{C}$. Analogously to the *convergence domain experimental evaluation phase* of SNA (Chapter 4.4.1), the evaluation is carried out in all the three **configurations**: hybrid, real and virtual. These distances define the maximum tolerable intial position error that allows to converge.

Depending on the evaluation, it is carried out over the textured scene and using the onboard central image ${}^{W}I^{*}$ (real), executed over a textured mesh and using the virtual central image ${}^{V}I^{*}$ (virtual) or carried out over the textured scene but using the virtual central image ${}^{V}I^{*}$ computed from the textured mesh (hybrid). Such mesh (Fig. 6.2) and scene are both planar because it is a satellite image (provided by Google Earth [3]) the first of them is provided by Google Earth and the second one is printed on a tarpaulin (Fig. 6.1). Therefore, the hybrid servo navigation is carried out mainly in 2 DoF ($\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}} = [v_x, v_y]^\top$) and the navigation graph $\mathcal{G}$ can be defined as a kinggraph-m (Chapter 2.3.1) with $m = 2$. Thus, the evaluation is carried out or executed in $3^m - 1 = 3^2 - 1 = 8$ equally spaced directions (Chapter 4.4.1). Furthermore, the virtual evaluation is also carried out in 4 and 6 DoF to study the performance of ZN luminance when implemented in PNS (Chapter 7.1.1).

[3]Google Earth website `https://www.google.com/earth/`

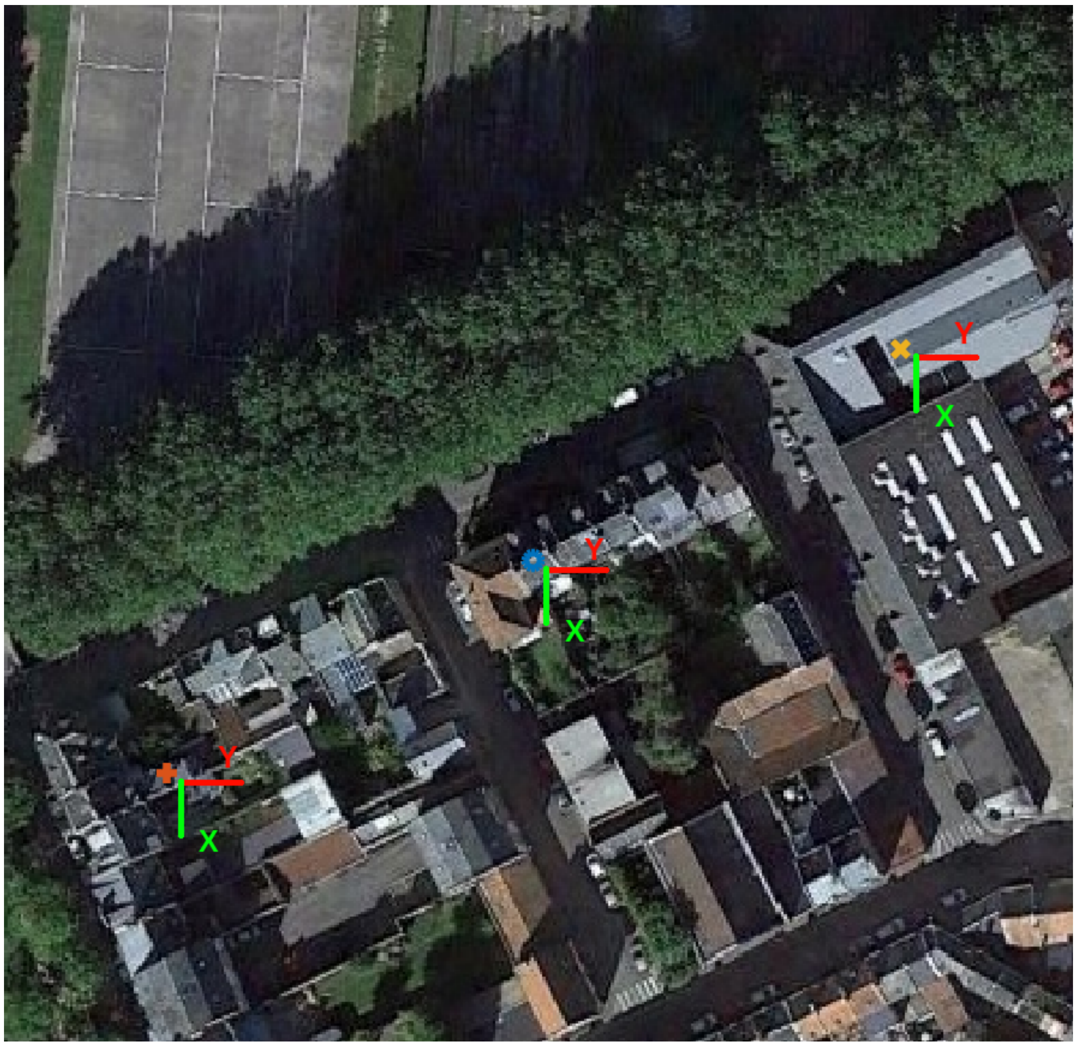


Figure 6.2: First virtual goal pose ${}^{V}\boldsymbol{\xi}_{0}^{G}$ (+), virtual central pose ${}^{V}\boldsymbol{\xi}^{C}$ (o) and second virtual goal pose ${}^{V}\boldsymbol{\xi}_{1}^{G}$ (x) over the planar textured mesh. The reference frames follow the direct convention.

The central pose ${}^{W}\boldsymbol{\xi}^{C}$ is located at the center of the two world goal poses ${}^{W}\boldsymbol{\Xi}^{G} = [{}^{W}\boldsymbol{\xi}_0^{G}, {}^{W}\boldsymbol{\xi}_1^{G}]$, with a distance of 495 mm between them. These poses also define the mission of PNS in Chapter 5.3. Indeed, this distance is bounded by the operational range of the robotic arm, therefore the camera is set close to the scene and a focal lenght of 13 mm is considered to compensate the small distance (compared with the range of a mobile robot). Fig. 6.3 displays the onboard and virtual central images, ${}^{W}I^{C} = I({}^{W}\boldsymbol{\xi}^{C})$ and ${}^{V}I^{C} = I({}^{V}\boldsymbol{\xi}^{C})$, respectively.

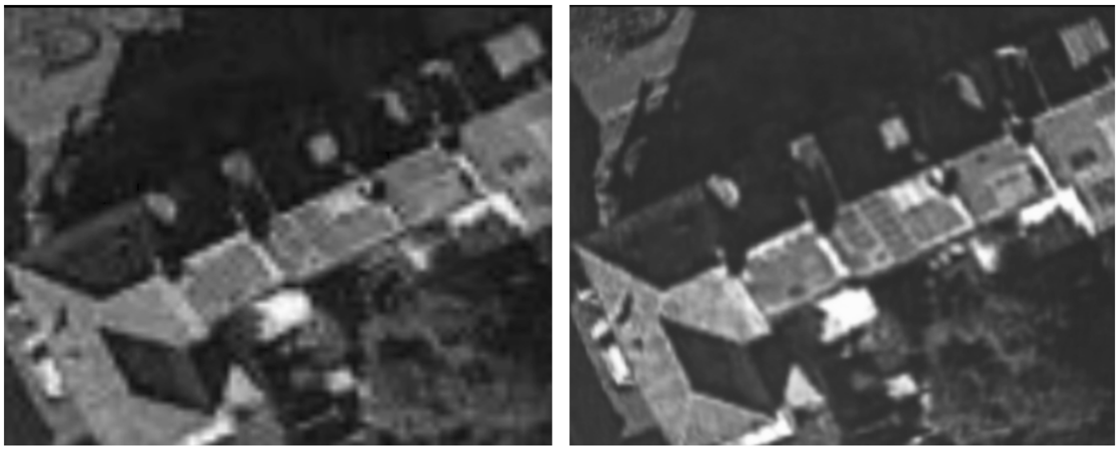


Figure 6.3: Virtual central image ${}^{V}I^{C}$ (left) and onboard central image ${}^{W}I^{C}$ (right).

The success of a positioning task is determined analogously to the hybrid navigability criteria (Chapter 4.3.1) but in the three configurations. The step is experimentally set to $\delta = 2$ mm and the values of the thresholds are experimentally set to: $\epsilon_d = 0.5$ mm, $\epsilon_t = 500$ iterations and $\epsilon_o = 1.74 \times 10^{-6}$ rad.

The DoF of the input velocity $\mathbf{u} = \dot{\boldsymbol{\xi}}$ are set to 2, 4 and 6, depending on the configuration.

6.2.1 Hybrid Configuration

This evaluation is used as the *convergence domain experimental evaluation phase* for the *mission* of PNS (Chapter 4.4.1). The hybrid minimum distance ${}^{H}d_{min}$, obtained here, is used to compute the initial number of nodes $|\mathcal{N}|$ in the navigation graph $\mathcal{G}$. The input velocity $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is computed from the hybrid control law for the Extended PVS (Eq. (5.4)).

6.2.2 Real Configuration

Also considered as "traditional" configuration. The novelty here, is the implementation of ZN luminance for visual servoing. In other words, this is a extent determination of the real domain of convergence ${}^{R}\mathcal{C}$ of the novel Extended Photometric Visual Servoing (PVS). The input velocity of the robot $\mathbf{u} = {}^{W}\dot{\boldsymbol{\xi}}$ is computed from the **real control law for the Extended PVS**:

$$\mathbf{u} = -\lambda \widehat{\mathbf{L}}^{+}_{\tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}^{*})}(\tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}(t)) - \tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}^{*})), \tag{6.1}$$

where $\tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}^{*}) = \tilde{\mathbf{I}}({}^{W}\boldsymbol{\xi}^{C})$ is obtained from the onboard image ${}^{W}I$ taken at the central pose ${}^{W}\boldsymbol{\xi}^{C}$ with the uEye camera (Chapter 6.1).

6.2.3 Virtual Configuration

The results obtained from this evaluation, i.e. the virtual minimum distance ${}^{V}d_{min}$, are used to estimate the hybrid minimum distance ${}^{H}d_{min}$ when needed (Chapter 7.1.1). Notice that, even if this evaluation does not follow exactly the *mission* of SNA, it can be used when the robot can not perform this evaluation *in situ*. The input velocity of the sensor $\mathbf{u} = {}^{V}\dot{\boldsymbol{\xi}}$ in the textured mesh is computed from the virtual control law of the Extended PVS, Eq. (5.5).

6.3 Evaluation Analysis

Table 6.1 shows the distances from the central pose $\boldsymbol{\xi}^{C}$ to the extents of each domain of convergence $\mathcal{C}$ in the $m^3 - 1 = 8$ directions and with an increment of $\delta = 2$ mm at each completed positioning task in the same direction j (Chapter 4.4.1). These extents are reported over the textured mesh and labeled in a counterclockwise direction beginnig ($j = 0$) at the center-right location of the central pose $\boldsymbol{\xi}^{C}$ (Fig. 6.4). The lenght of these distances are determined by the content of the mesh. Moreover, larger distances will intialize the navigation graph $\mathcal{G}$ with a larger number of nodes $|\mathcal{N}|$ during the *space tessellation phase* (Chapter 4.4.1).

Concerning 2 DoF cases, the maximum gap is equal to 10 mm ($|{}^{R}_{2}d_0 - {}^{V}_{2}d_0| = |{}^{H}_{2}d_0 - {}^{V}_{2}d_0|$ in Table 6.1) and the minimum one is equal to 0 ($|{}^{R}_{2}d_0 - {}^{H}_{2}d_0| = |{}^{R}_{2}d_1 - {}^{H}_{2}d_1|$ in Table 6.1) when comparing the positioning test in each direction for these three configurations. The minimum distances are ${}^{R}_{2}d_{min} = {}^{H}_{2}d_{min} =$ 14 mm at $j = 1$ and ${}^{V}_{2}d_{min} = 18$ mm at $j = 1, 5$. From these cases, the ratio between the hybrid and virtual minimum distances in 2 DoF can be computed by:

$$r = {}^{H}_{2}d_{min}/{}^{V}_{2}d_{min}, \tag{6.2}$$

which is used in the PNS *missions* (Chapter 7.1.1) to estimate the hybrid minimum distances in 4 and 6 DOF, ${}^{H}_{4}d_{min}$ and ${}^{H}_{6}d_{min}$ respectively.

Regarding 4 DoF and 6 DoF cases, the minimum distances are ${}^{V}_{4}d_{min} = 16$ mm at $j = 0, 3, 4, 7$ and ${}^{V}_{6}d_{min} = 16$ mm at $j = 3, 4$.

Notice that this evaluation is similar to a *convergence domain experimental evaluation phase* (Chapter 4.4.1) performed by a flying at high altitude (Chapter 7.1).

Now that the extents of the hybrid and virtual domains of convergence, i.e. ${}^{H}\mathcal{C}$ and ${}^{V}\mathcal{C}$, were computed and that ZN luminance $\tilde{\mathbf{I}}$ proved to be a feature adapted to cope with the light variations caused by the hybrid configuration, the feature is ready to be fully implemented on the PNS *missions*.

Table 6.1 summarizes the result of this experiment.

Table 6.1:: Convergence domain extent evaluation (unit: mm)

Direction	0	1	2	3	4	5	6	7	min
Real 2 DoF	16	14	22	46	20	18	24	30	**14**
Hybrid 2 DoF	16	14	18	42	24	20	28	36	**14**
Virtual 2 DoF	26	18	28	48	22	18	26	32	**18**
Virtual 4 DoF	16	18	22	16	16	18	22	16	**16**
Virtual 6 DoF	18	20	22	16	16	20	24	18	**16**

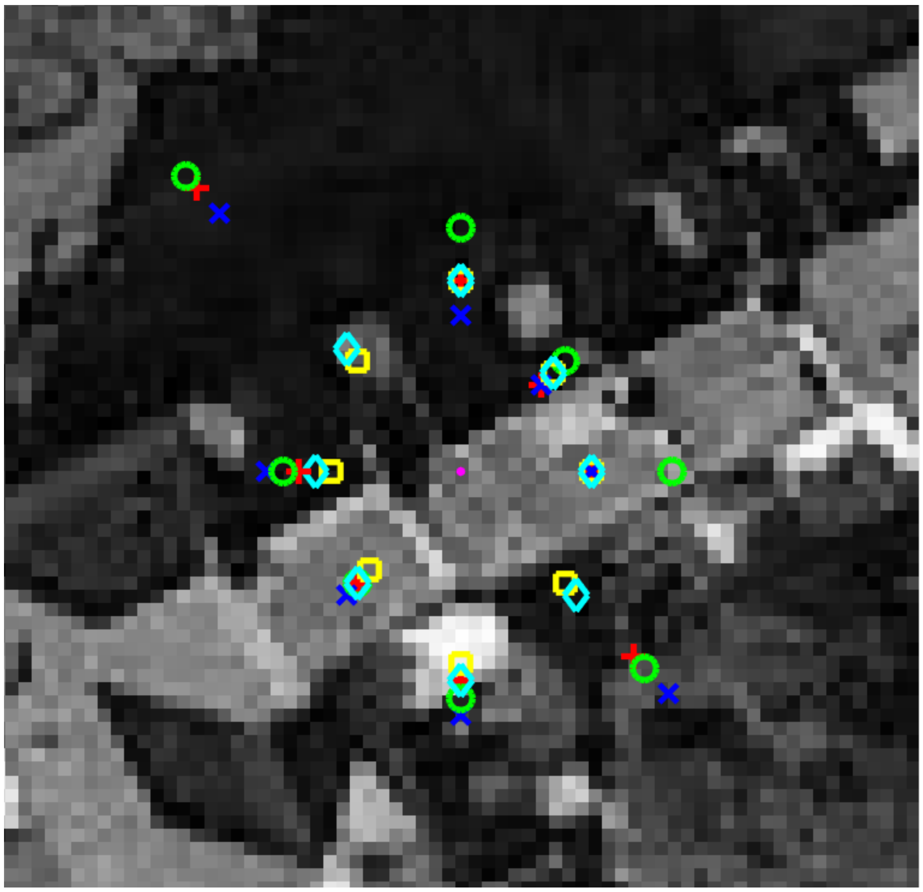


Figure 6.4: Extents of the domain of convergence for ZN luminance around the location of the center pose $\boldsymbol{\xi}^C$ (●) projected over the mesh. Hybrid configuration in 2 DoF (x), real configuration in 2 DoF (+) and Virtual configuration in 2 DoF (o), 4 DoF (□) and 6 DoF (◇).

Chapter 7

Experimental Evaluation

This Chapter presents the main experiments, i.e. the *missions*, that test the performance of the Photometric Navigation Systems (PNS) and, by consecuence, the Servo Navigation Architecture (SNA).

The first Section depicts a hypotethical scenario from which two group of *missions* are based on. Then, the second Section classifies and shows the results from the latter one.

7.1 Missions

Two group of *missions* are carried out using the experimental setup to evaluate the performance of PNS simulating the **hypothetical scenario** presented hereafter:

"A flying robot, equipped with a conventional camera, navigates at high altitude over a city without relying on GNSS. Its *mission* is defined by two world goal poses, ${}^{W}\boldsymbol{\xi}_0^G$ and ${}^{W}\boldsymbol{\xi}_1^G$, and the navigation graph $\mathcal{G}$ is of dimension 2, i.e.

kinggraph-2. The textured scene and the mesh feature only one of the following characteristics:

- The scene and the mesh are fully textured and the hybrid servo navigation is carried out in **differents DoF**.
- The robot is *ex situ* and either a part of the scene and mesh are **partially textureless** or the **outdated mesh** does not accurately represent the scene."

The latter scenario is hypotetical because it is simulated using a robotic arm.

For the sake of simplicy, the world goal poses ${}^{W}\boldsymbol{\xi}_0^G$ and ${}^{W}\boldsymbol{\xi}_1^G$ and the scene are identically used to experimentally evaluate the extents of the domain of convergence $\mathcal{C}$ and to define the *mission* in Chapter 6.2. Thus, the hybrid and virtual minimum distances, ${}^{H}d_{min}$ and ${}^{V}d_{min}$, are obtained or estimated from Table 6.1. Moreover, the straigth line that intersects two consecutive goal poses $\{\boldsymbol{\xi}_i^G, \boldsymbol{\xi}_{i+1}^G\} \subset \Xi^G$ is defined as the **main trajectory** and the world pose ${}^{W}\boldsymbol{\xi}$ embodied by a virtual desired image ${}^{V}I^*$ related to an element of the visual path $\tilde{\mathbf{I}}^*$ is referred as **world desired pose** ${}^{W}\boldsymbol{\xi}^*$.

7.1.1 DoF Actuation

Four experiments test the performance of PNS in 2, 4 and 6 DoF. These experiments are carried out over the textured scene (Fig. 6.1) and using the mesh (Fig. 6.2) to generate the visual memory. The purpose here is to study the performance of PNS when navigating in different DoF rather than study the difficulties generated by the textured scene or mesh, e.g. lack of texture. Furthermore, the rosbustness of the navigation graph $\mathcal{G}$ initialization computed from an **estimation** of the hybrid minimum distance ${}^{H}d_{min}$ is studied. It should be noticed that, even if this estimation saves experimental time, it is not considered by SNA.

The first experiment of this group is called **nominal case**. This is the simplest one of all because its *mission* completely follows SNA (Chapter 4.4). As an approximation to the hypothetical scenario (Chapter 7.1), the hybrid servo navigation is performed in 2 DoF. In this experiment, the robot and the camera were able to perform the *convergence domain experimental evaluation phase* (Chapter 4.4.1) **in situ** and in the same DoF that the later hybrid servo navigation. The hybrid minimum distance in 2 DoF ${}^{H}_{2}d_{min} = 14$ mm is obtained from Table 6.1.

The second experiment is called **nominal ex situ case**. This experiment is similar to the first one except that the robot and the camera were **ex situ** during the *convergence domain experimental evaluation phase* (Chapter 4.4.1). Therefore, the virtual search space ${}^{V}\mathbb{X}^{S}$ was tessellated according to the virtual minimum distance in 2 DoF ${}^{V}_{2}d_{min} = 18$ mm (Table 6.1) instead of the hybrid one (${}^{H}_{2}d_{min}$). The virtual minimum distance ${}^{V}d_{min}$ is computed by simulating the *convergence domain experimental evaluation phase* (Chapter 4.4.1) using only the textured mesh and a group of virtual positioning tasks instead of the hybrid ones. Certainly by simulating the aforementioned *phase* instead of carrying out as in SNA, the experimental time is considerably reduced.

The third and fourth experiments are called **approximated 4 and 6 DoF cases**. The purpose of these experiments is to study the robustness of the visual memory generated by estimating the hybrid minium distance in 4 and 6 DoF by:

$$\begin{cases} {}^{H}_{4}d_{min} \approx r\ {}^{V}_{4}d_{min}, \\ {}^{H}_{6}d_{min} \approx r\ {}^{V}_{6}d_{min}, \end{cases} \tag{7.1}$$

where r is the ratio from Eq. (6.2) and ${}^{H}_{4}d_{min} = {}^{H}_{6}d_{min} \approx 12.44$ mm.

7.1.2 Robustness to Scene and Mesh Content

The two experiments presented here feature two difficulties regarding vision-based navigation. Both of them generate their visual memory in an *ex situ* manner (similarly to the nominal *ex situ* case), i.e. by tessellating the virtual search space ${}^{V}\mathbb{X}^{S}$ from the virtual minimum distance in 2 DoF ${}^{V}_{2}d_{min} = 18$ mm. Afterwards, the hybrid servo navigation is also carried out in 2 DoF.

In the first of these experiments, a textureless zone is superposed roughly in the center of the scene and mesh, thus is called **textureless zone case**. The interest here is to invalidate any visual path $\tilde{\mathcal{I}}^{*}$ related with the main trajectory (Chapter 7.1) because the vision-based navigation would be impossible over scenes with uniform texture [Sanfourche et al. (2012)].

In the **outdated mesh case**, a mesh that does not represent anymore the scene is used to generate the visual memory. This experiment tests one the most difficult circumstances of the model-based navigation systems; when the preobtained model is not accurate [Segvic et al. (2007) & Dichtl et al. (2019)]. Indeed, such a experiment brings PNS to one of its limits by generating a visual memory from a mesh generated 10 years before than the scene but performing the hybrid servo navigation over the latter. Some buildings and shadows, that appear in the desired virtual images related to the visual path, are not present in the scene (Fig. 7.7e).

7.2 Results

The results from the six experiments presented in the preivous Section are compared, hereafter, in the aspects mentioned below:

- **Navigation Graph Initialization:** This aspect refers to the number of nodes $|\mathcal{N}|$ from which the navigation graph $\mathcal{G}$ was initially created. Certainly, the number of nodes $|\mathcal{N}|$ is directly related with the total duration from the *space tessellation phase* to the *navigation graph initialization phase* (Chapter 4.4.1).
- **Visual Path Evaluation:** It concerns to the locations over the mesh from where the visual $\tilde{\mathcal{I}}^*$ path was generated, the number of elements $\tilde{\mathbf{I}}^*$ stored in the visual memory and the total duration from the *space tessellation phase* (Chapter 4.4.1) to the *sensory memory generation module* (Chapter 4.4.1). Notice that the latter duration is strongly related with the content of the mesh and the minimum distance d_{min} from which the naviagtion graph $\mathcal{G}$ was generated.
- **Navigation Performance:** It focuses on the correct element update of the visual memory and if the hybrid servo navigation caused PNS to navigate from the first to the last world goal poses, i.e. ${}^{W}\boldsymbol{\xi}_0^G$ and ${}^{W}\boldsymbol{\xi}_1^G$.

Regarding the navigation graph initialization, the results in terms of duration and number of nodes $|\mathcal{N}|$ can be roughly classified depending on the minimum distance d_{min} used. The navigation graphs $\mathcal{G}$ that were initialized by computed or estimated hybrid minimum distances ${}^{H}d_{min}$ were created in 10.1 s with $|\mathcal{N}| = 1369$ nodes for the nominal case and 21.7 s with $|\mathcal{N}| = 1764$ nodes for the approximated cases in 4 Dof and 6 DoF. The navigation graphs $\mathcal{G}$ that were initialized from the virtual minimum distance in 2 DoF ${}^{V}_{2}d_{min}$ were created with $|\mathcal{N}| = 841$ nodes in 2.3 s. Indeed, the virtual minimum distance in 2 DoF ${}^{V}_{2}d_{min}$ used for the textureless zone was not generated from a simulation of the *convergence domain experimental evaluation phase* because the virtual desired image ${}^{V}I^*$ lacks of texture; instead, this distance is obtained from the Table 6.1.
From this comparison, it is evident that the hybrid minimum distance d_{min} provides an insight of the navigability of the mesh in the hybrid configuration that allows to find a navigable visual path $\tilde{\mathcal{I}}^*$ at the first iteration. Despite generating navigation graphs $\mathcal{G}$ with more nodes, this situation saves computational time during the *visual memory*

generation module (Chapter 4.4.1) by validating only once each virtual positioning task during the *navigability evaluation module* (Chapter 4.4.1).

The results from the visual path evaluation can also be grouped within the same two classes; experiments that used either the hybrid or the virtual minimum distance, ${}^{H}d_{min}$ and ${}^{V}d_{min}$. The navigables visual paths $\tilde{\mathcal{I}}^{*}$ from the first class are reported hereafter: 145.1 s from the nominal case with $|\tilde{\mathcal{I}}^{*}| = 37$ desired ZN luminance features $\tilde{\mathbf{I}}^{*}$, 169.3 s from the approximated 4 DoF case with $|\tilde{\mathcal{I}}^{*}| = 42$ features and 177.5 s from the approximated 6 DoF case with $|\tilde{\mathcal{I}}^{*}| = 42$ features stored in the visual memory. The visual paths $\tilde{\mathcal{I}}^{*}$ from the second class are: 2286.5 s from the nominal *ex situ* case with $|\tilde{\mathcal{I}}^{*}| = 35$ features, in 2245.8 s from the textureless zone case with $|\tilde{\mathcal{I}}^{*}| = 35$ features and in 1831.7 s from the outdated mesh case with $|\tilde{\mathcal{I}}^{*}| = 34$ features stored in the visual memory.
In fact, the experiments that used the hybrid minimum distance ${}^{H}d_{min}$ found a navigable visual path $\tilde{\mathcal{I}}^{*}$ at the first iteration of the *path-search module* (Chapter 4.4.1), i.e. from the locations related to the main trajectory, despite searching for a path over a navigation graph $\mathcal{G}$ with a larger number of nodes $|\mathcal{N}|$ than the ones that used the virtual minimum distance ${}^{V}d_{min}$. The locations related to the visual paths $\tilde{\mathcal{I}}^{*}$ are reported over the textured mesh in Fig. 7.1 and Fig. 7.2. An excerpt of them are displayed in Fig 7.7.

The hybrid servo navigation caused the robot to reach the last world goal pose ${}^{W}\boldsymbol{\xi}_{1}^{G}$ from the first one ${}^{W}\boldsymbol{\xi}_{0}^{G}$ in all the experiments except for the approximated 6 DoF and outdated mesh cases.
In the former case, however, the hybrid control law of the Extended PVS in 6 DoF (Eq. 5.4) leaded to reach a joint limit near the border of the robot's range. This problem is not managed by the control law nor the lower level control of the robot. The elements $\tilde{\mathbf{I}}^{*}$ of the visual memory were correctly updated, i.e. when the world desired poses ${}^{W}\boldsymbol{\xi}^{*}$ are sequentially reached, in all experiment except for the textureless zone and outdated mesh cases. However, the dragging error (Fig. 7.3) occurred in the former one did not prevent the success of its hybrid servo navigation. Despite the dragging error that occurred in the outdated mesh case, the robot traveled a distance of 225 mm in the correct direction before it completely diverged. Furthermore, by implementing PNS in the robotic arm, the discontinous velocities computed at the element update did not affect the performance of the system. The trajectories are reported over the textured mesh in Fig. 7.1 and Fig. 7.2. The evolution of the input velocity $\mathbf{u} = \dot{\boldsymbol{\xi}}$ and the cost C are displayed in Fig.7.4, Fig. 7.5 and Fig. 7.6 with their respective element update.

In conclusion, the experiments that used the hybrid minimum distance ${}^{H}d_{min}$ generated visual paths $\tilde{\mathcal{I}}$ along the main trajectory that resulted in completed *mis-*

sions in less time than the rest and in any DoF. The nominal ex situ and textureless zone cases demonstrated that it is possible to complete the *mission* if the hybrid servo navigation is defined in 2 DoF even if the robot is not able to carried out the *convergence domain experimental evaluation phase* (Chapter 4.4.1). As expected, the *mission* of the outdated mesh case could not be completed due to the difficulty that represents an inaccurate textured mesh. Nevertheless, the robot traveled a distance that barely corresponds to the half of the complete trajectory.

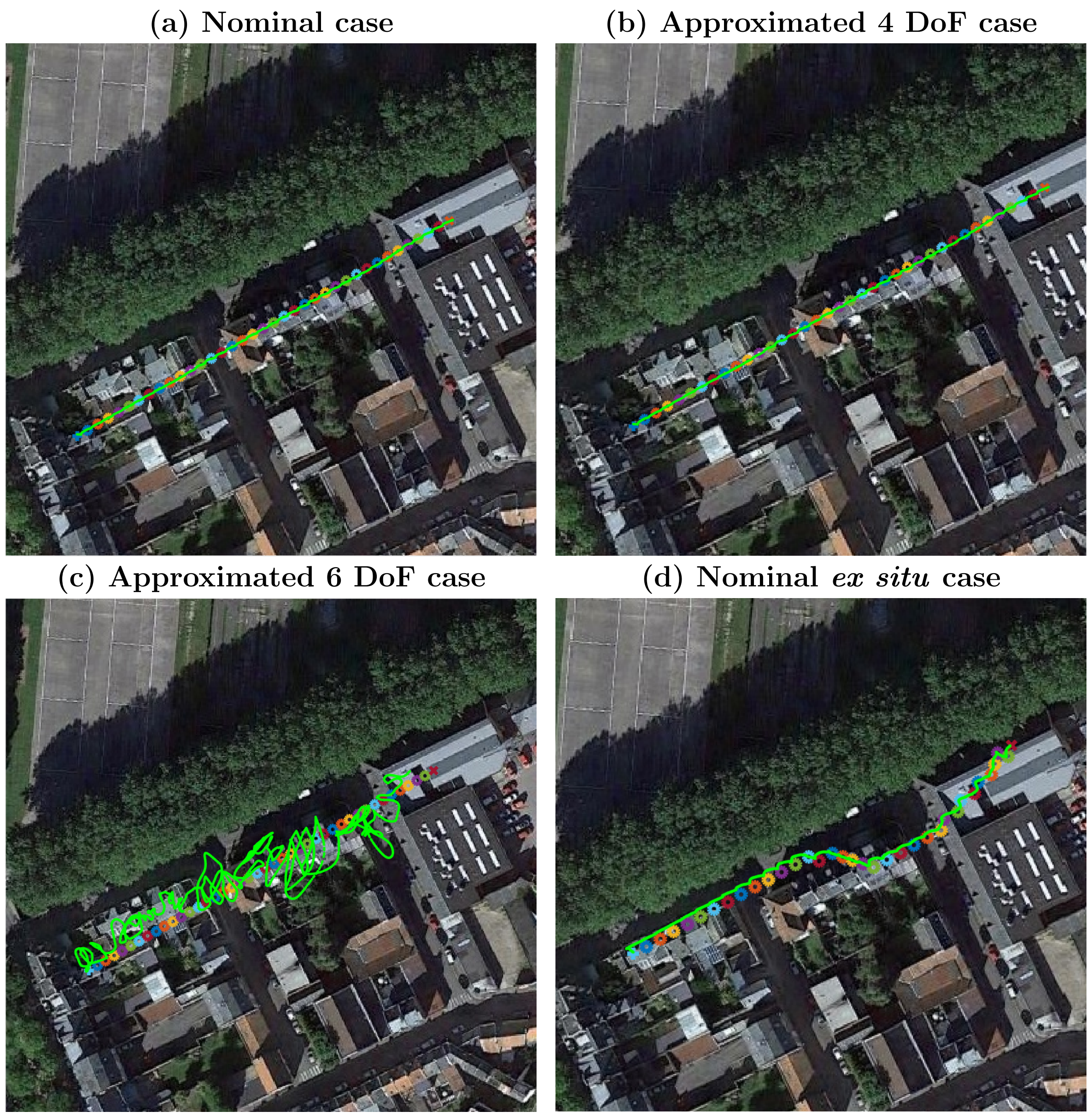

(a) **Nominal case**
(b) **Approximated 4 DoF case**
(c) **Approximated 6 DoF case**
(d) **Nominal *ex situ* case**

Figure 7.1: Trajectory of the robotic arm (green line) of the nominal and approximated cases. Locations of the first ${}^{W}\boldsymbol{\xi}_0^G$ (+) and last world goal pose ${}^{W}\boldsymbol{\xi}_1^G$ (x), other world desired poses ${}^{W}\boldsymbol{\xi}^*$ (*) appear reported over the textured mesh.

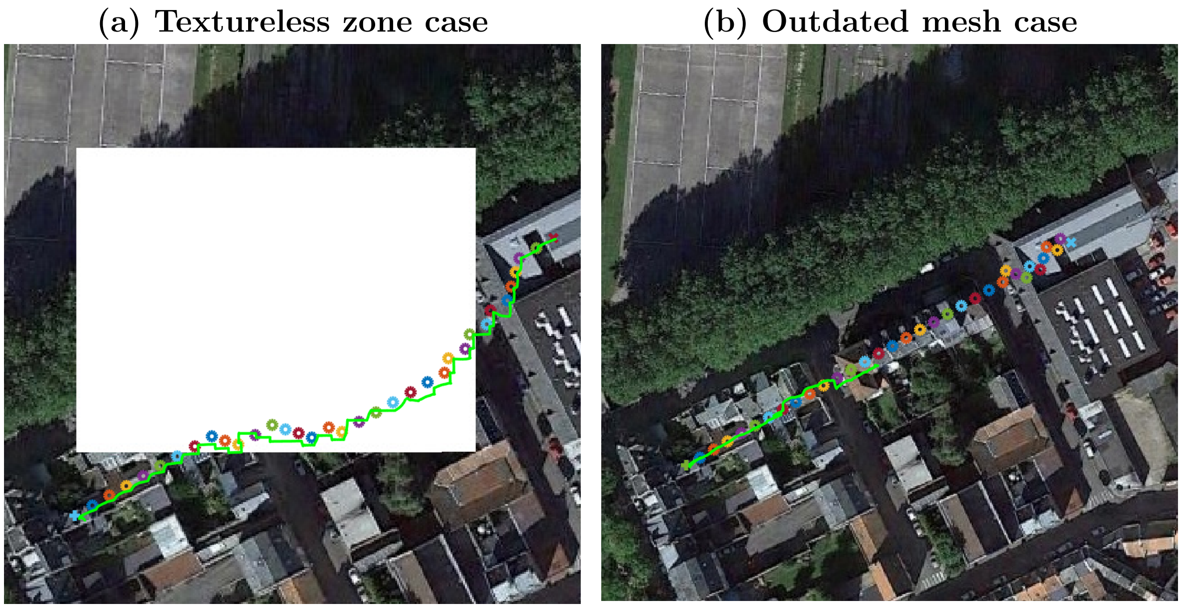


Figure 7.2: Trajectory of the robotic arm (green line) of the textureless zone and outdated mesh cases. Locations of the first ${}^{W}\boldsymbol{\xi}_0^G$ (+) and last world goal pose ${}^{W}\boldsymbol{\xi}_1^G$ (x), other world desired poses ${}^{W}\boldsymbol{\xi}^*$ (*) appear reported over the textured mesh.

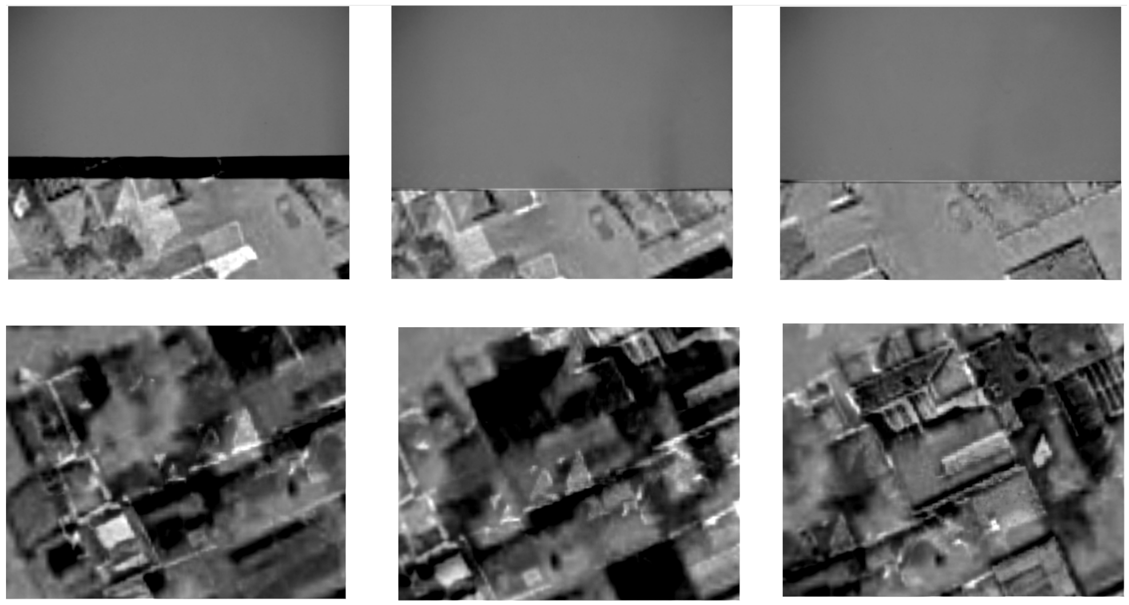
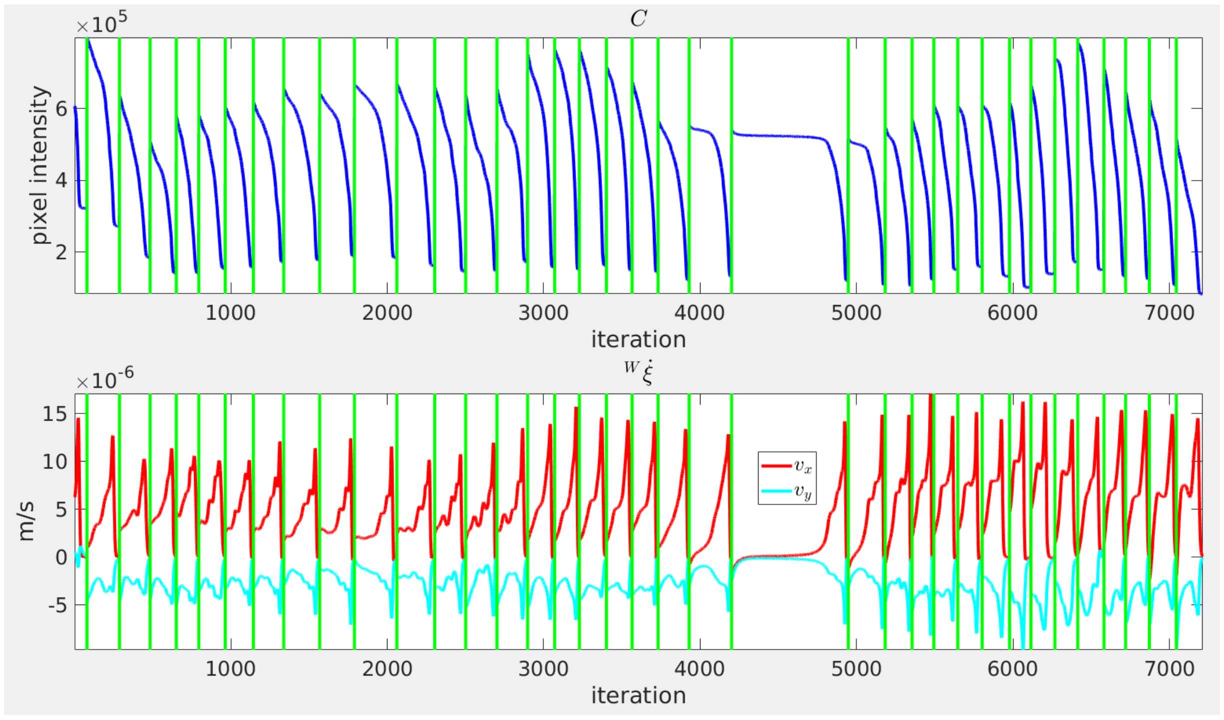


Figure 7.3: Dragging error corrected. Two rows of images of difference, between the onboard image ${}^{W}I$ and the desired virtual image ${}^{V}I^*$, show the issue where the dragging error began (left column), was carried (central column) and corrected (right column).

(a) **Nominal case**



(b) **Nominal *ex situ* case**

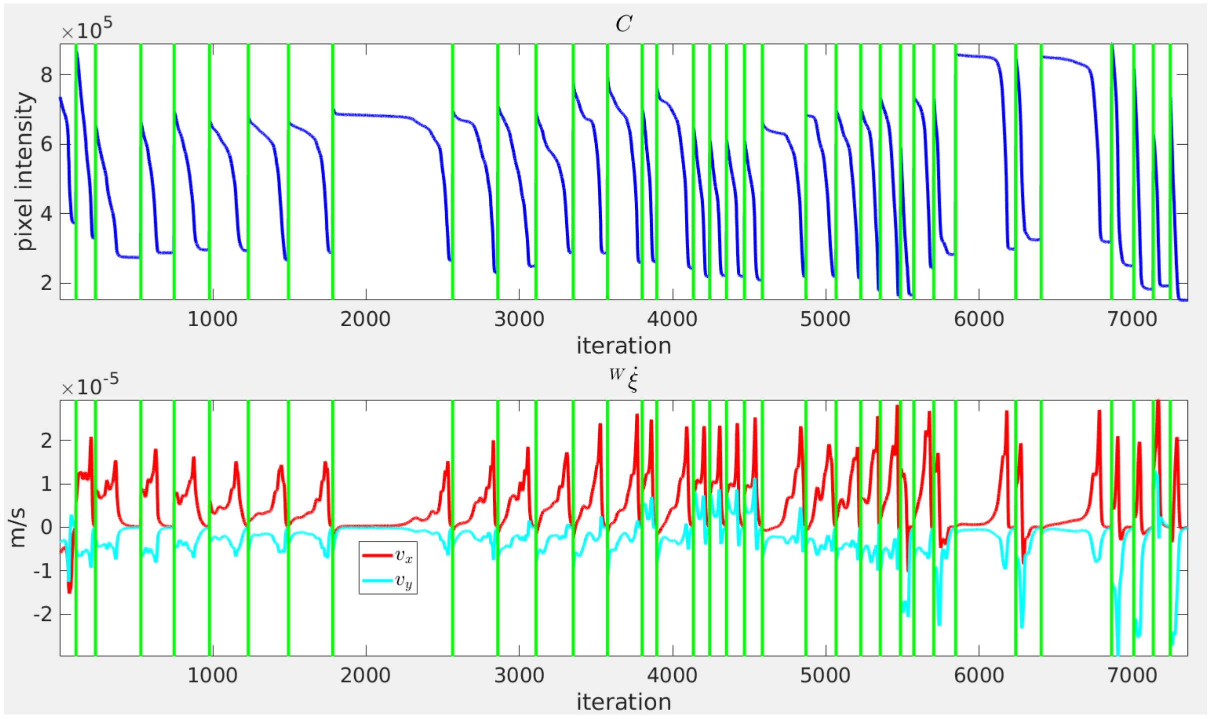
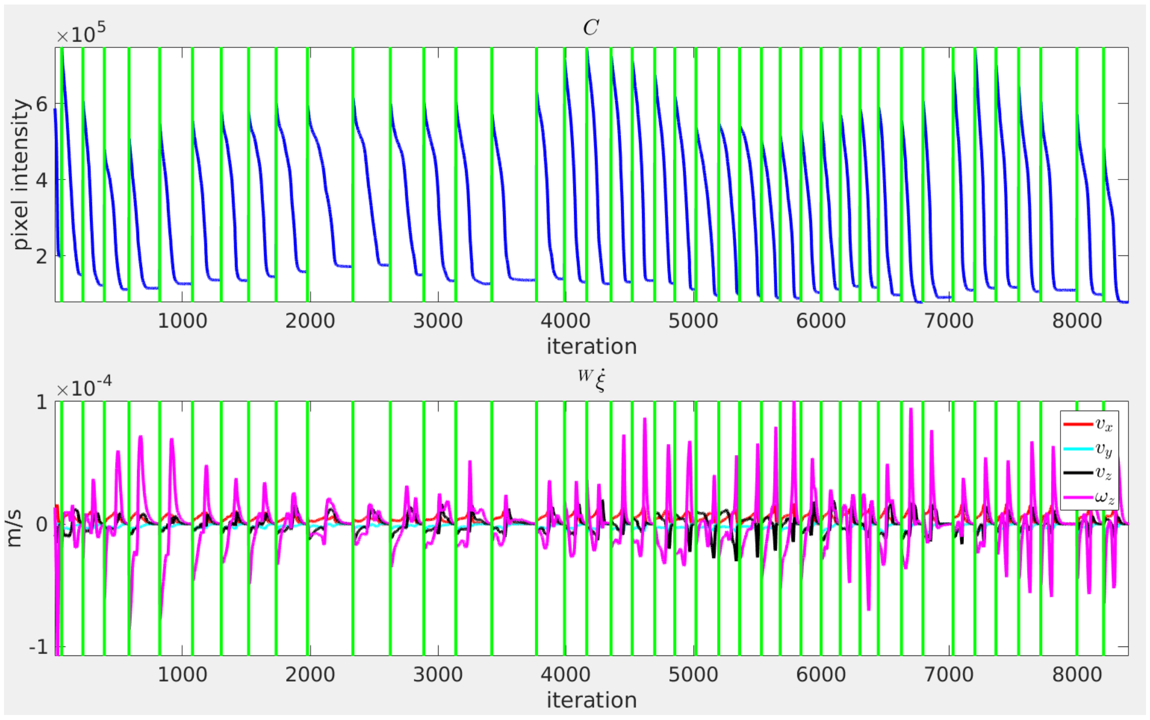


Figure 7.4: Cost C (top) and input velocity $\mathbf{u} = {}^W\dot{\boldsymbol{\xi}}$ (bottom) evolution of the nominal cases. Element update appears in green.

(a) Approximated 4 DoF case



(b) Approximated 6 DoF case

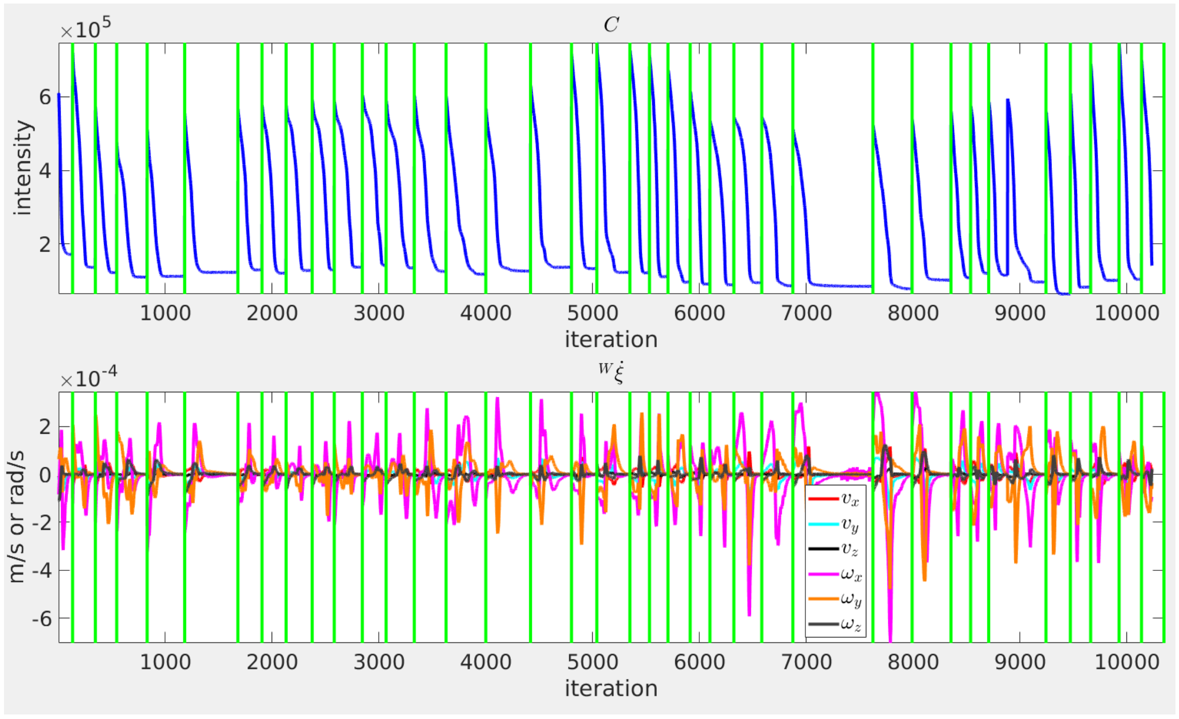
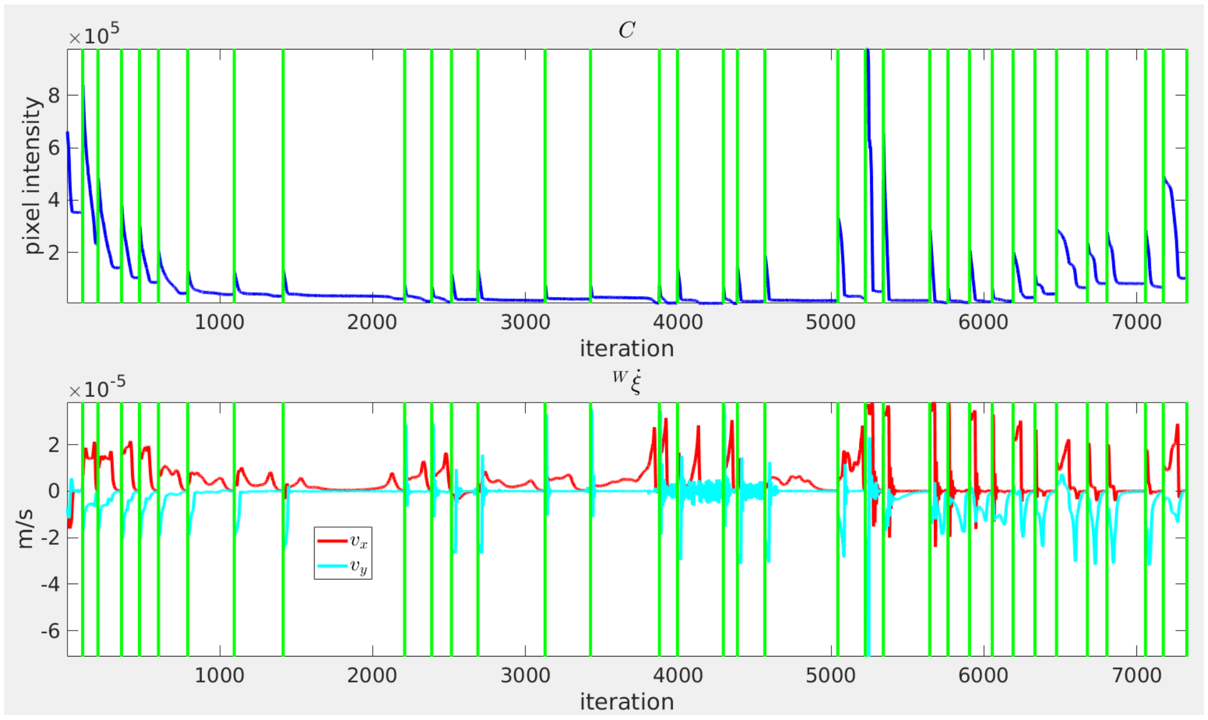


Figure 7.5: Cost C (top) and input velocity $\mathbf{u} = {}^W\dot{\boldsymbol{\xi}}$ (bottom) evolution of the approximated cases. Element update appears in green.

(a) Textureless zone case



(b) Outdated mesh case

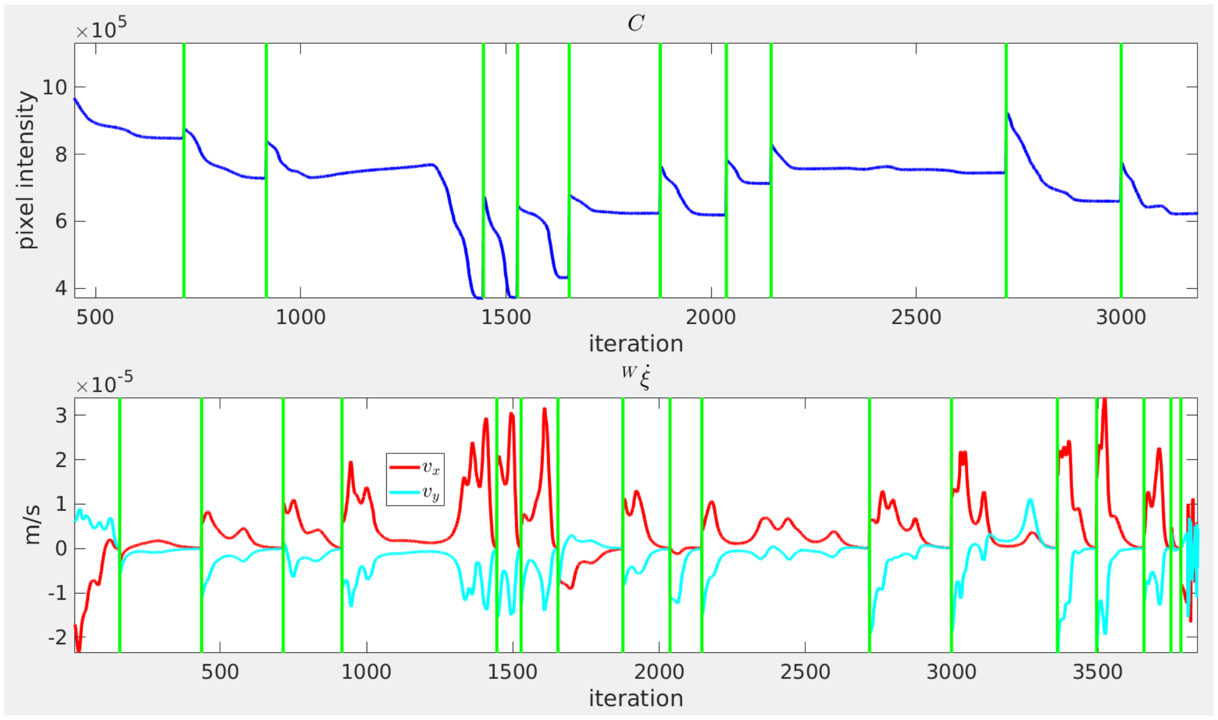
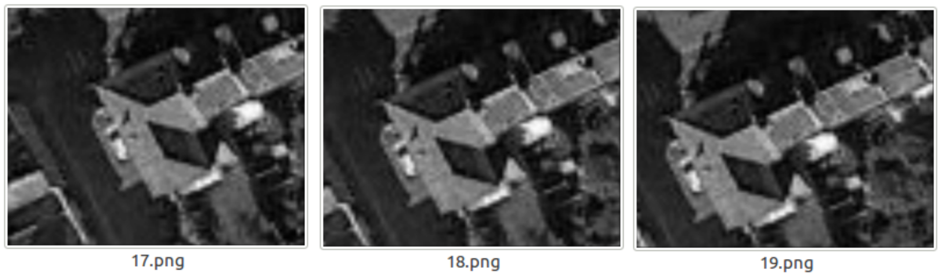
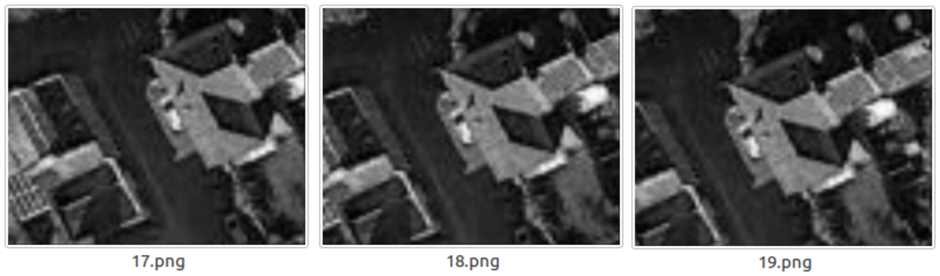


Figure 7.6: Cost C (top) and input velocity $\mathbf{u} = {}^W\dot{\boldsymbol{\xi}}$ (bottom) evolution of the textureless zone and outdated mesh cases. Element update appears in green.

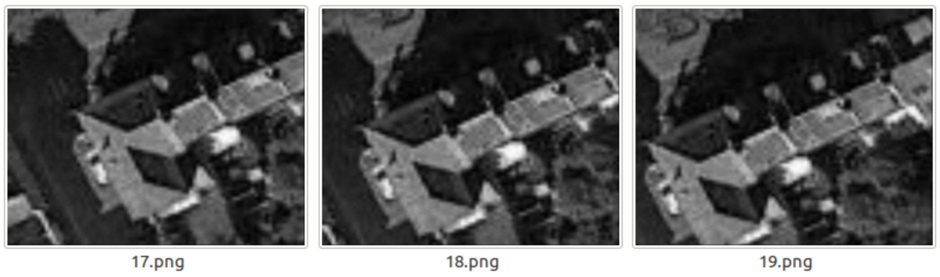
(a) Nominal case



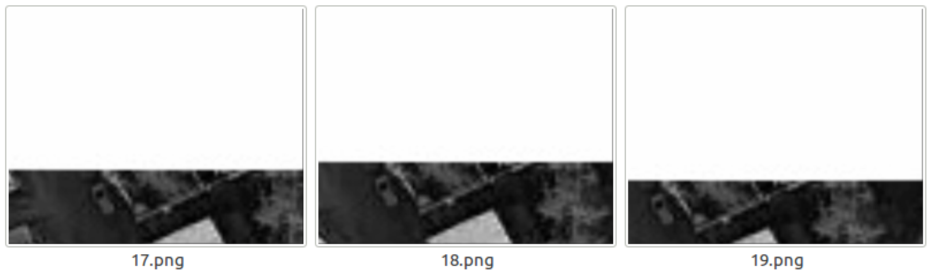
(b) Approximated 4 and 6 DoF case



(c) Nominal *ex situ* case



(d) Textureless zone case



(e) Outdated mesh case

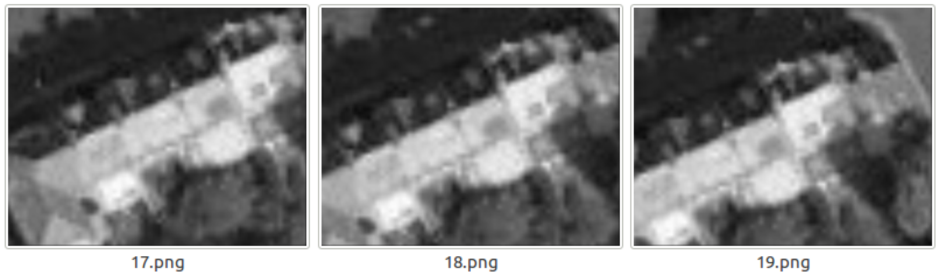


Figure 7.7: Excerpt of three consecutive virtual desired images ${}^{V}I^{*}_{17}$, ${}^{V}I^{*}_{18}$ and ${}^{V}I^{*}_{19}$ related to the visual paths $\tilde{\mathcal{I}}^{*}$.

7.3 Conclusions

This Chapter evaluates the performance of PNS when implementing it to a robotic-arm equipped with a perspective camera. Such configuration simulates an aerial robot flying at high altitude so the scene might be considered flat. PNS represents an improvement of the Photometric Teach & Repeat (PT&R) in all the aspects except for two. The latter one was implemented in outdoor aerial navigation which is a difficult task. One advantage of PT&R over the Servo Navigation Architecture (SNA) is that the memory of the former one might be computed *ex situ* whereas that one of the latter requires a *convergence domain experimental evaluation phase* which is *in situ*. Therefore, a complete *ex situ path-planning stage* is considered where the virtual minimum distance ${}^{V}d_{min}$ is used instead of the hybrid one ${}^{H}d_{min}$, even if it does not follow SNA.

By finding a navigable path over the straigth line, the hybrid minimum distance ${}^{H}d_{min}$ exhibited that is was correctly estimated, while minimizing the distance to travel. This distance proved to be robust enough so an estimation of it produced navigable paths when more DoF were enabled. Nevertheless, when the robot is not available to carry out the *convergence domain experimental evaluation phase* (Section 4.4.1), the virtual minimum distance ${}^{V}d_{min}$ can be used instead to initialize the navigation graph $\mathcal{G}$. Due that the hybrid approach produces more dissimilarities between the textured mesh and scene than comparing the mesh with itself, the extents of the virtual domain of convergence ${}^{V}\mathcal{C}$ are expected to be larger than those of the hybrid one ${}^{H}\mathcal{C}$. This situation generates a virtual minimum distance ${}^{V}d_{min}$ larger than the hybrid one ${}^{H}d_{min}$, and by consequence, a navigation graph $\mathcal{G}$ with a fewer number of nodes $|\mathcal{N}|$. As the search poses ${}^{V}\boldsymbol{\xi}^{P}$ in the mesh are further from each others, the virtual positioning tasks are more likely to fail. Therefore, the *path-planning stage* usually takes more time before finding a navigable visual path $\tilde{\mathcal{I}}^{*}$. Indeed, this process is almost 16 times slower than the corresponding one when using ${}^{H}d_{min}$ instead. In other words, the closer the search poses ${}^{V}\boldsymbol{\xi}^{P}$ in the mesh wrt to each other, the faster to validate them by means of virtual positioning tasks during the *navigability evaluation module* (Section 4.4.1). The advantages of using the latter distance ${}^{V}d_{min}$ are: the reduction in the experimental time, since a virtual *convergence domain experimental evaluation phase* is faster than the hybrid one and can be executed without the presence of the robot *in situ*, and less elements in the visual path $\tilde{\mathcal{I}}^{*}$.

In the *textureless zone case*, ${}^{V}d_{min}$ is computed using the model without the textureless zone. Notice that neither ${}^{V}d_{min}$ nor ${}^{H}d_{min}$ are possible to compute. As future work, a group of positioning tests that completely cover the search space might be carried out in order to compute the density of the navigation graph. This improvement allows to completely determine the navigability of the search space ${}^{V}\mathbb{X}^{S}$ instead of assuming that it is homogeneous everywhere.

To summarize, all the missions in the experimentation part succeeded except for the last one where the model is different from the scene. This result was expected since this issue is inherited from the model-based approach. So far, all the contributions had been presented; moreover, two of them had been experimentally evaluated. This Chapter closes Part II and prepares the general conclusions provided in the next Chapter.

Part IV

Closure

Chapter 8

General Conclusions

This thesis proposes one way to bridge the gaps between the appeareance-based and model-based approaches for vision-based robotic navigation. The former ones leverage on the characteristics of the scene instead on relying on a preobtained model. They are suitable for navigation through unstructured environments when the models are not available. In previously existing methods, a desired path can be created either through human guided navigation or from an exploration stage. This path, represented in the sensor space, is stored in a memory and serves as reference to the robot during the autonomous navigation. Conversely, the human operator is not required to propose a desired path when the model is available *a priori*, as in the model-based approach. In this approach, the path-planning process is executed autonomously. Moreover, it can optmize certain criteria, e.g. distance or memory.

Three main contributions to robotic navigation and vision-based navigation are presented: the Photometric Teach & Repeat (PT&R), the Photometric Navigation System (PNS) and the Servo Navigation Architecture (SNA). All of the contributions have in common a memory-based navigation autonomously generated from a preobtained model of the scene.

PT&R was the first attempt to address the robotic navigation problem from a memory-based perspective. Its main difference w.r.t. others Visual Teach & Repeat (VT&R) systems, is that its visual memory is generated from a model of the scene rather than from the scene itself. This contribution is able to autonomously generate a visual memory from a straight trajectory defined in the textured mesh. Then, the robot performs the hybrid servo navigation using the visual path stored in the visual memory. The system does not requires the presence of the robot to carry out the *teaching stage*. Nevertheless, the system does not exploit all the advantages of relying on a textured mesh of the scene to generate the visual memory, i.e. others paths than the straight line are not considered.

PNS was created in order to exploit the advantages of a memory-based navigation autonomously generated from a textured mesh. During the *path-planning stage*, the search space in the textured mesh is tessellated accordingly with the hybrid minimum distance. This distance is computed from an experimental determination of the extents of the hybrid domain of convergence and gives an insight of the navigability at the center of the mesh. Such evaluation requires the robot to be once in a small area of the environment. This evaluation is recommended since some differences between the model and the scene are expected which hinder the hybrid servo navigation. Nehertheless, it can be simulated to determine the virtual minimum distance; although, the success of the hybrid servo navigation will be ensured only if the model accurately represents the scene. Then, the navigability of some simulated navigations is evaluated, following a topological approach, in order to generate the visual memory. This process is an improvement of the *learning stage* of PT&R whose convergence is evaluated from a cost along a single DoF. Later, the hybrid servo navigation is carried out analogously to PT&R but with another update criteria. The experiments demonstrated the effectiveness of PNS and inspired to develop a general framework for robotic navigation based on sensory path planning; SNA. In other words, SNA is an architecture that explores the paths that can be traveled using a representation of the sensor and scene. The hybrid servo navigation, based on a sensory memory, is flexible to a variety of sensors and environments.

Motivated by the current state of SNA and PNS the future works are mentioned below:

- Further research on SNA includes its implementation on mobile robots, different scenes, sensors and models other than those explored by PNS.
- One important limitation of SNA is that it requires a static obstacle-free scene to navigate. So, one perspective is to extend SNA to avoid static obstacles since the *path-planning stage*. However, this perspective does not consider to cope with dynamic obstacles during the autonomous navigation.
- The virtual servo navigation that evaluates the navigability of a proposed path might be replaced by a stability analysis which might save computation. Moreover, it is suitable for generating sensory memories for aerial robots even with latency and complex dynamics.
- Another switching criteria can be employed depending on the application.

- A group of positioning tests, that completely cover the search space, should be carried out in order to compute the density of the navigation graphs. Their densities should be related with the navigability of their corresponding search spaces that they encompass.

- SNA considers a short and navigable path as optimal which is related to the heuristic of A*. However, the latter might be defined in such way that it minimizes another criterion, e.g. energy or memory.

- The definition of the update criteria avoids the step of localization during the hybrid navigation. However, it causes an undesired behaviour; the speed of the robot is reduced almost to zero in order to determine that the intermediate pose was attained. Indeed, this is an assumption since the algorithm converged to a minimum (global or local). Another update criteria might be proposed so the hybrid navigation carries out with a constant, or smoother, speed. This problem remains open.

- Additional modeling to explicitly take into account the difference between the actual scene and a model of it can be implemented, e.g. robust estimation [Okutomi et al. (2002)], brightness distribution based methods [Vidyadharan and Thampi (2015)] or interval analysis [Atiya and Hager (1993)], in order to replace the experimental evaluation of the extents of the hybrid domain of convergence. Indeed, such modelling can determine the initial tessellation of the navigation graph instead of performing the *convergence domain experimental evaluation phase*. This modification may save experimental time and even be considered in the *navigability evaluation module*.

- The virtual servo navigation process that evaluates the path proposed by A*, might be simultaneously evaluated using parallel computing. Such improvement might reduce in great measure the computational cost to generate the sensory path.

- PNS was implemented in order to simulate an UAV flying at high altitude. Therefore, a future work is to implement PNS on an UAV and exploit the kinggraph-3.

- Another SNA implementation, in vision-based navigation, is to use a depth sensor and a wheeled robot to perform indoor navigation. In this system, the model of the scene can be obtained from a CAD model or by a Polymap.

Annex A: ROS

The Robotic Operating System, which logo appears in Fig. A.1, is a middleware used as framework for robotic applications and software. ROS provides libraries and tools to help developers and roboticists to collaborate between them in robotic projects from different research institutions [Quigley et al. (2009)].

ROS

Figure A.1: ROS logo.

The software organization unit of ROS is the **package**, described by a **manifest**, that contains libraries, executables, scripts, etc. Furthermore, the middleware is organized in the **ROS environment** that follows the structure of a **catkin workspace**. Birefly speaking, a catkin workspace is composed for up to four **spaces**: the **source space**, that contains the source code; the **built space**, from where CMake is invoked to build the projet; the **development space**, where targets are placed before being installed; the **install space**, where the targets are installed.

Consider the next scenario that briefly illustrates two types of communication paradigms at the ROS **Computation Graph Level**: many-to-many, two or more **nodes** that **publish** / **subscribe** to a common **topic**, and one-way, one or more nodes that **request** for a **service** and one that **replies**.

"The **master** and the **parameter server** are initialized and four nodes (n_0, n_1, n_2 and n_3) are launched with their respective parameters. n_0 publishes a **message** to a topic. n_1 subscribes to this topic and stores the content of the message in a **bag**. n_2 is a **service node**, also called **server**, that receives and process the data requested by n_3, called the **client node**."

In the experimentation with the Bebop 2, the driver is launched from a node in the bebop_autonomy package. Then, the connection is established with the server created by the robot. Finally, the node that controls the robot publishes two message to the topics whose the driver node is subscribed. These topics are: `cmd_vel` to control the motion and `image_raw` to recive the onboard image ${}^{W}I$.

Bibliography

Astrom, K. J. and Wittenmark, B. (1995). A survey of adaptive control applications. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 1, pages 649–654. IEEE.

Atiya, S. and Hager, G. D. (1993). Real-time vision-based robot localization. *IEEE transactions on robotics and automation*, 9(6):785–800.

Barfoot, T. D., Stenning, B., Furgale, P., and McManus, C. (2012). Exploiting reusable paths in mobile robotics: Benefits and challenges for long-term autonomy. In *2012 Ninth Conference on Computer and Robot Vision*, pages 388–395. IEEE.

Blanc, G., Mezouar, Y., and Martinet, P. (2005). Indoor navigation of a wheeled mobile robot along visual routes. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 3354–3359. IEEE.

Bonin-Font, F., Ortiz, A., and Oliver, G. (2008). Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53.

Bresson, G., Alsayed, Z., Yu, L., and Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 20:1–1.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer.

Caron, G., Marchand, E., and Mouaddib, E. M. (2013). Photometric visual servoing for omnidirectional cameras. *Autonomous Robots*, 35(2-3):177–193.

Chatterjee, A., Rakshit, A., and Singh, N. N. (2012). *Vision based autonomous robot navigation: algorithms and implementations*, volume 455. Springer.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90.

Chaumette, F. and Hutchinson, S. (2008). Visual servoing and visual tracking.

Chen, L. M. (2014). *Digital and Discrete Geometry: Theory and Algorithms*. Springer.

Chepoi, V., Dragan, F., and Vaxès, Y. (2002). Center and diameter problems in plane triangulations and quadrangulations. In *Symposium on Discrete Algorithms*, volume 2, pages 346–355.

Chesi, G. and Hashimoto, K. (2010). *Visual servoing via advanced numerical methods*, volume 401. Springer.

Collewet, C. and Marchand, E. (2011). Photometric visual servoing. *IEEE Transactions on Robotics*, 27(4):828–834.

Collewet, C., Marchand, E., and Chaumette, F. (2008). Visual servoing set free from image processing. In *International Conference on Robotics and Automation*. IEEE.

Corke, P. (2017). *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer.

Costante, G., Delmerico, J., Werlberger, M., Valigi, P., and Scaramuzza, D. (2018). Exploiting photometric information for planning under uncertainty. pages 107–124.

Courbon, J., Mezouar, Y., Guenard, N., and Martinet, P. (2009a). Visual navigation of a quadrotor aerial vehicle. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5315–5320. IEEE.

Courbon, J., Mezouar, Y., Guénard, N., and Martinet, P. (2010). Vision-based navigation of unmanned aerial vehicles. *Control Engineering Practice*, 18(7):789–799.

Courbon, J., Mezouar, Y., and Martinet, P. (2008). Indoor navigation of a non-holonomic mobile robot using a visual memory. *Autonomous Robots*, 25(3):253–266.

Courbon, J., Mezouar, Y., and Martinet, P. (2009b). Autonomous navigation of vehicles from a visual memory using a generic camera model. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):392–402.

Crombez, N., Caron, G., and Mouaddib, E. (2015). 3D point cloud model colorization by dense registration of digital images. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):123.

Dame, A. and Marchand, E. (2011). Mutual information-based visual servoing. *IEEE Transactions on Robotics*, 27(5):958–969.

Dame, A. and Marchand, E. (2013). Using mutual information for appearance-based visual path following. *Robotics and Autonomous Systems*, 61(3):259–270.

DeSouza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267.

Dichtl, J., Le, X. S., Lozenguez, G., Fabresse, L., and Bouraqadi, N. (2019). Robot navigation with polymap, a polygon-based map format. In *Proceedings of SAI Intelligent Systems Conference*, pages 1138–1152. Springer.

Dijkstra, E. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271.

Diosi, A., Segvic, S., Remazeilles, A., and Chaumette, F. (2011). Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing. *IEEE Transactions on Intelligent Transportation Systems*, 12(3).

Dong, J. F., Wijesoma, S., and Shacklock, A. P. (2007). Extended rao-blackwellised genetic algorithmic filter slam in dynamic environment with raw sensor measurement. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1473–1478. IEEE.

Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265.

Elfes, A. (1989). Occupancy grids: a probabilistic framework for robot perception and navigation [ph. d. thesis].

Espiau, B., Chaumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fontanelli, D., Danesi, A., Belo, F. A., Salaris, P., and Bicchi, A. (2009). Visual servoing in the large. *International Journal of Robotics Research*, 28(6):802–814.

Friedman, S., Pasula, H., and Fox, D. (2007). Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *IJCAI*, volume 7, pages 2109–2114.

Fu, X., Zhang, L., Chen, Z., Wang, H., and Shen, J. (2019). Improved rrt* for fast path planning in underwater 3d environment. In *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*. ACM.

Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560.

Gasteratos, A., Beltran, C., Metta, G., and Sandini, G. (2002). Pronto: a system for mobile robot navigation via cad-model guidance. *Microprocessors and Microsystems*, 26(1):17–26.

Gaussier, P., Joulain, C., Zrehen, S., Banquet, J.-P., and Revel, A. (1997). Visual navigation in an open environment without map. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97*, volume 2, pages 545–550. IEEE.

Giralt, G., Chatila, R., and Vaisset, M. (1990). An integrated navigation and motion control system for autonomous multisensory mobile robots. In *Autonomous robot vehicles*, pages 420–443. Springer.

Goedemé, T., Nuttin, M., Tuytelaars, T., and Van Gool, L. (2004). Vision based intelligent wheel chair control: The role of vision and inertial sensing in topological navigation. *Journal of Robotic Systems*, 21(2):85–94.

Goedemé, T., Nuttin, M., Tuytelaars, T., and Van Gool, L. (2007). Omnidirectional vision based topological navigation. *International Journal of Computer Vision*, 74(3):219–236.

Gutmann, J.-S. and Schlegel, C. (1996). Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT'96)*, pages 61–67. IEEE.

Güzel, M. S. (2013). Autonomous vehicle navigation using vision and mapless strategies: a survey. *Advances in Mechanical Engineering*, 5:234747.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Huang, H.-M., Messina, E., and Albus, J. (2003). Toward a generic model for autonomy levels for unmanned systems (alfus). Technical report, National institue of standards and technology Gaithersburg.

Huh, D.-J., Park, J.-H., Huh, U.-Y., and Kim, H.-I. (2002). Path planning and navigation for autonomous mobile robot. In *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, volume 2, pages 1538–1542. IEEE.

Jones, E. S. and Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378.

Kennedy, J. (2010). Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766.

Khuller, S., Rosenfeld, A., and Wu, A. (2000). Centers of sets of pixels. *Discrete applied mathematics*, 103(1-3):297–306.

Koenig, S. and Likhachev, M. (2002). Dˆ* lite. *Aaai/iaai*, 15.

Kosaka, A. and Pan, J. (1995). Purdue experiments in model-based vision for hallway navigation. In *IROS*, volume 95, pages 87–96. Citeseer.

LaValle and M, S. (1998). Rapidly-exploring random trees: A new tool for path planning.

LaValle and M, S. (2006). *Planning algorithms*. Cambridge university press.

Lee, T. and Soatto, S. (2010). An end-to-end visual recognition system. *Technical Report*.

Li, P. and Duan, H. (2012). Path planning of unmanned aerial vehicle based on improved gravitational search algorithm. *Science China Technological Sciences*, 55(10):2712–2719.

Llofriu, M., Tejera, G., Contreras, M., Pelc, T., Fellous, J.-M., and Weitzenfeld, A. (2015). Goal-oriented robot navigation learning using a multi-scale space representation. *Neural Networks*, 72:62–74.

Luukkonen, T. (2011). Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22.

Magassouba, A., Bertin, N., and Chaumette, F. (2018). Aural servo: sensor-based control from robot audition. *IEEE Transactions on Robotics*, 34(3):572–585.

Mahe, H., Marraud, D., and Comport, A. I. (2019). Real-time rgb-d semantic keyframe slam based on image segmentation learning from industrial cad models.

Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *International Conference on Robotics and Automation*, volume 2, pages 1843–1848. IEEE.

Marchand, É., Spindler, F., and Chaumette, F. (2005a). Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, 12(4):40–52.

Marchand, E., Spindler, F., and Chaumette, F. (2005b). Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52.

Marjovi, A. and Marques, L. (2014). Multi-robot odor distribution mapping in realistic time-variant conditions. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3720–3727. IEEE.

Matica, M. L. and Kovendi, Z. (2011). Structure analysis for an industrial robot. *Journal of Computer Science and Control Systems*, 4(1):89.

Matsumoto, Y., Ikeda, K., Inaba, M., and Inoue, H. (1999). Visual navigation using omnidirectional view sequence. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 1, pages 317–322. IEEE.

Matsumoto, Y., Inaba, M., and Inoue, H. (1996). Visual navigation using view-sequenced route representation. In *Proceedings of IEEE International conference on Robotics and Automation*, volume 1, pages 83–88. IEEE.

Mezouar, Y., Remazeilles, A., Gros, P., and Chaumette, F. (2002). Images interpolation for image-based control under large displacement. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3787–3794. IEEE.

Monajjemi, M. (2018). Bebop autonomy-ros driver for parrot bebop drone.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598.

Nguyen, T., Mann, G. K., and Gosine, R. G. (2014). Vision-based qualitative path-following control of quadrotor aerial vehicle. In *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 412–417. IEEE.

Nilsson, N. J., R. B. and Wahlstrom, S. (1968). Application of intelligent automata to reconnaissance. Technical report, Stanford Research Institute. Project 5953 Interim Report 4 From the Nilsson archives – SHAKEY papers.

O'Keefe, J. and Dostrovsky, J. (1971). The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research.*

Okutomi, M., Nakano, K., Maruyama, J., and Hara, T. (2002). Robust estimation of planar regions for visual navigation using sequential stereo images. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3321–3327. IEEE.

Pamučar, D., Stević, Ž., and Sremac, S. (2018). A new model for determining weight coefficients of criteria in mcdm models: Full consistency method (fucom). *Symmetry*, 10(9):393.

Pan, B. (2011). Recent progress in digital image correlation. *Experimental Mechanics*, 51(7):1223–1235.

Park, S., Schöps, T., and Pollefeys, M. (2017). Illumination change robustness in direct visual slam. In *International conference on robotics and automation (ICRA)*, pages 4523–4530. IEEE.

Pearl, J. (1984). Heuristics addison-wesley. *Reading, MA.*

Pfrunder, A., Schoellig, A. P., and Barfoot, T. D. (2014). A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera. In *2014 Canadian Conference on Computer and Robot Vision*, pages 238–245. IEEE.

Priestnall, G., Jaafar, J., and Duncan, A. (2000). Extracting urban features from lidar digital surface models. *Computers, Environment and Urban Systems*, 24(2):65–78.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

Rahmani, B., Putra, A. E., Harjoko, A., and Priyambodo, T. K. (2015). Review of vision-based robot navigation method. *IAES International Journal of Robotics and Automation*, 4(4).

Raj, S., Giordano, P. R., and Chaumette, F. (2016). Appearance-based indoor navigation by ibvs using mutual information. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–6. IEEE.

Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248.

Remazeilles, A. and Chaumette, F. (2007). Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4).

Remazeilles, A., Chaumette, F., and Gros, P. (2006). 3d navigation based on a visual memory. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE.

Rodriguez, E., Caron, G., Pegard, C., and Lara, A. D. (2020). Photometric path planning for vision-based navigation. In *IEEE International Conference on Robotics and Automation*.

Rosen, C. A. and Nilsson, N. J. (1967). Application of intelligent automata to reconnaissance. Technical report, Stanford Research Institute. Project 5953 Interim Report 3 From the Nilsson archives – SHAKEY papers.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. (2011). ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, volume 11, page 2. IEEE/CVF.

Ryu, B.-S. and Yang, H. S. (1999). Integration of reactive behaviors and enhanced topological map for robust mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 29(5):474–485.

Salichs, M. A. and Moreno, L. (2000). Navigation of mobile robots: open questions. *Robotica*, 18(3):227–234.

Salih, A. L., Moghavvemi, M., Mohamed, H. A., and Gaeid, K. S. (2010). Flight pid controller design for a uav quadrotor. *Scientific research and essays*, 5(23):3660–3667.

Sanfourche, M., Delaune, J., Le Besnerais, G., De Plinval, H., Israel, J., Cornic, P., Treil, A., Watanabe, Y., and Plyer, A. (2012). Perception for uav: Vision-based navigation and environment modeling.

Santosh, D., Achar, S., and Jawahar, C. (2008). Autonomous image-based exploration for mobile robot navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 2717–2722. IEEE.

Šegvić, S., Remazeilles, A., Diosi, A., and Chaumette, F. (2007). A framework for scalable vision-only navigation. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 1–12. Springer.

Segvic, S., Remazeilles, A., Diosi, A., and Chaumette, F. (2007). Large scale vision-based navigation without an accurate global reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Siciliano, B. and Khatib, O. (2016). *Springer handbook of robotics*. Springer.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Slotine, J.-J. E., Li, W., et al. (1991). *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ.

Soares, J. M., Aguiar, A. P., Pascoal, A. M., and Martinoli, A. (2013). Joint asv/auv range-based formation control: Theory and experimental results. In *2013 IEEE International Conference on Robotics and Automation*, pages 5579–5585. IEEE.

Stella, E., Musio, F., Vasanelli, L., and Distante, A. (1995). Goal-oriented mobile robot navigation using an odour sensor. In *Proceedings of the Intelligent Vehicles' 95. Symposium*, pages 147–151. IEEE.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*.

Stytz, M. R., Frieder, G., and Frieder, O. (1991). Three-dimensional medical imaging: algorithms and computer systems. *ACM Computing Surveys (CSUR)*, 23(4):421–499.

Teulière, C. and Marchand, E. (2014). A dense and direct approach to visual servoing using depth maps. *IEEE Transactions on Robotics*, 30(5):1242–1249.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Tomatis, N. (2001). *Hybrid, metric-topological, mobile robot navigation*. PhD thesis.

Urmson, C., Simmons, R., and Nesnas, I. (2003). A generic framework for robotic navigation. In *Proceedings of the IEEE Aerospace Conference*, volume 5, pages 2463–2470.

Vandrish, P., Vardy, A., and King, P. (2012). Towards auv route following using qualitative navigation. In *2012 Ninth Conference on Computer and Robot Vision*, pages 425–432. IEEE.

Vidyadharan, D. S. and Thampi, S. M. (2015). Brightness distribution based image tampering detection. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–5. IEEE.

Wahdan, M. and Elgazzar, M. M. (2019). Homotopy classes and cell decomposition algorithm to path planning for mobile robot navigation.

Warren, M., Greeff, M., Patel, B., Collier, J., Schoellig, A. P., and Barfoot, T. D. (2018). There's no place like home: Visual teach and repeat for emergency return of multirotor uavs during gps failure. *IEEE Robotics and Automation Letters*, 4(1):161–168.

Yoder, J.-D., Baumgartner, E. T., and Skaar, S. B. (1996). Initial results in the development of a guidance system for a powered wheelchair. *IEEE Transactions on Rehabilitation Engineering*, 4(3):143–151.

Zagradjanin, N., Pamucar, D., and Jovanovic, K. (2019). Cloud-based multi-robot path planning in complex and crowded environment with multi-criteria decision making using full consistency method. *Symmetry*.

Zhang, J., Liu, W., and Wu, Y. (2011). Novel technique for vision-based uav navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2731–2741.

Résumé Étendu

Les robots sont capables d'effectuer un vaste groupe de tâches comment la surveillance, la cartographie et la navigation. Cette thèse est principalment consacrée à l'étude de la dernière tâche qui est définie grossièrement comme la capacité d'atteindre un groupe de poses dans l'environnement.

Cette thèse présente trois contributions qui effectuent la navigation robotique, et elles sont présentées ci-dessous :

- La première d'entre elles considère la navigation à l'aide d'une caméra monoculaire ; par conséquent, elle appartient à la classification des systèmes de navigation basée-vision (ou VBNS en anglais). Ce système s'appelle Instruction & Répétition Photométriques (ou PT&R en anglais) car il est basé sur le paradigme *teach-repeat* et l'Asservissement Visuel Photométrique (ou PVS en anglais) pour traiter le problème de la navigation. La nouveauté dans PT&R est qu'il génère de manière autonome sa mémoire visuelle à partir d'un modèle préalable de la scène, qui est un maillage texturé. Cette mémoire est utilisée comme référence pour naviguer et elle est comparée avec les images acquises par la caméra embarquée pour contrôler le robot. Même si les manipulations sur PT&R démontrent qu'un drone est capable de naviguer en utilisant la mémoire visuelle, la génération de cette dernière est limitée à la ligne droite et n'explore pas d'autres chemins.

- La deuxième contribution est un VBNS inspiré par PT&R, appelé système de navigation photométrique (ou PNS en anglais). La principale différence entre PNS et PT&R est que le premier système explore d'autres chemins que la ligne droite en utilisant une approche topologique. De plus, une navigation simulée évalue la navigabilité d'un chemin représenté par la mémoire visuelle. Le succès des expérimentations de PNS a montre son efficacité à planifier de façon autonome un chemin navigable en utilisant le maillage texturé et à naviguer au sein de la scène. PNS est considéré comme une contribution majeure de la thèse.

- La troisième contribution, également considérée comme une contribution majeure, est une généralisation du PNS qui comprend une grande variété de capteurs et de modèles de scène. Cette architecture est appelée Architecture de Navigation par Asservissement (ou SNA en anglais) car elle aborde le problème de navigation en concevant ce problème comme une concaténation de tâches successives de positionnement en utilisant une mémoire sensorielle comme référence.

Avant de détailler PT&R, PNS et SNA, il est important de définir trois concepts importants dans la navigation robotique :

- **Perception** : Consiste à ajuster les mesures du capteur à un modèle mathématique qui comprend le robot, l'environnement, le capteur même et l'interaction entre eux. Au sein d'un processus d'estimation, la perception repose sur les capteurs qui récupèrent l'état du robot lui-même (propriocetifs) et sur ceux qui récupèrent l'état de l'environnement externe (extéroceptifs) afin d'interpréter les quantités physiques.
- **Planification du chemin** : Il s'agit d'une compétence stratégique de résolution de problèmes dont l'objectif est de trouver un ensemble de points de cheminement, en s'appuyant sur la connaissance de l'environnement, afin que le robot puisse atteindre un ensemble de poses désirées.
- **Navigation** : C'est l'aspect cognitif lié à la mobilité robuste du robot. Il combine le sens de la perception, une certaine connaissance de l'environnement et un ensemble de poses désirées pour contrôler de manière fiable le robot lors d'une *mission* qui implique le déplacement.

De plus, les trois contributions s'inspirent de la navigation basée mémoire, utilisée par certains VBNS. Les systèmes robotiques de navigation basée mémoire ont en commun quatre aspects importants avec les contributions susmentionnées :

- **Structure du système** : Aussi connu sous le nom d'architecture de contrôle, cet aspect se réfère à la hiérarchie de processus qu'un système utilise pour générer la mémoire et naviguer. Le plus haut niveau de structure de ces systèmes est généralement composé par une étape de planification et de navigation.
- **Mémoire sensorielle** : Structure topologique, elle contient des données de capteurs utilisées pour naviguer. Le composant le plus important des données du capteur est sa primitive, qui permet de distinguer une mesure d'une autre.
- **Critère de sélection** : Il détermine le contenu de la mémoire sensorielle à partir d'un groupe candidat de données sensorielles. Ces critères peuvent être définis directement lors de la collecte de données de capteurs, p. ex., chronologiquement, ou sélectionnés à partir d'une base de données, p. ex., en fonction d'un coût ou de sa dérivée.
- **Critère de mise à jour** : Il détermine quand la référence actuelle doit être changée à la suivante ou quand la navigation doit être arrêtée. Elle est liée à

l'état actuel du système par rapport à la référence issue de la mémoire sensorielle. Ce critère peut être déterminé par un seuil sur la valeur d'un coût, sa dérivée ou la sortie d'une loi de contrôle.

Maintenant que les concepts et aspects les plus importants concernant la navigation basée mémoire sensorielle sont définis, les contributions sont détaillées ci-après.

Instruction & Répétition Photométriques

Les concepts généraux de PT&R sont présentés dans le paragraphe suivant.

Avant tout, le but de la navigation est d'atteindre consécutivement un groupe de poses désirées. Cettes poses définissent la mission de PT&R. Ce dernier est composé de deux étapes principales : instruire et répéter. En tant que VBNS, la perception de PT&R repose sur une caméra monoculaire et une simulation de celle-ci qui rend des images virtuelles à partir du maillage texturé. La planification du chemin est exécutée pendant l'étape d'instruction. Elle consiste à créer une mémoire visuelle qui décrit un chemin en ligne droite passant par toutes les poses désirées. De manière générale, la mémoire visuelle ne contient qu'un chemin visuel qui est défini par une liste d'images virtuelles. Ensuite, la navigation est effectuée au cours de l'étape de répétition. La navigation compare des primitives des images virtuelles avec celles des images embarquées. Grâce à la nature hybride de la comparaison de ces caractéristiques et en concevant la navigation comme une concaténation des tâches de positionnement par asservissement, ce dernier processus est appelé navigation basée asservissement hybride.

Les *étapes* qui composent la *mission* de PT&R sont détaillées ci-dessous.

- *Étape d'instruction* : Une fois que les poses désirées sont définies dans la scène pour naviguer, elles sont représentées dans le maillage texturé par un autre groupe, ici appelé le groupe des poses virtuelles désirées. Ensuite, en échantillonnant régulièrement le segment reliant les deux poses on obtient un chemin de poses candidates. À partir de la première pose candidate, qui correspond à la dernière pose virtuelle désirées, une image virtuelle est rendue en utilisant la fonction capteur et le maillage texturé. La primitive de cette image est stockée si les critères de sélection sont satisfaits. En effet, cette caractéristique d'interêt est directement obtenue par une transformation de l'image virtuelle. Les critères de sélection sont principalement basés sur la dérivée numérique d'un coût défini par la Somme des Différences Carrées (SSD). Une fois la dernière pose atteinte, le chemin visuel est stocké dans la mémoire visuelle.
- *Étape de répétition* : Elle commence par placer le robot près de la première pose

désirée. Ensuite, la navigation d'asservissement hybride commence en comparant le premier vecteur de primitives visuelles de la mémoire avec celui issu de l'image courante acquise par la camera embarquée. La commande de vitesse est calculée par une loi de d'asservissement visuel. Lorsque les **critères de mise à jour** sont satisfaits la loi de commande considère le vecteur de primitives visuelles suivant du chemin visuel. Ce critère est fondé sur la dérivée du coût et détermine également quand cette *étape*, et par conséquent la *mission*, est terminée.

La performance du PT&R est évaluée pour un drone, qui est un Bebop 2, par un groupe d'expériences. Le Bebop 2 communique avec la station au sol par Wi-Fi. Le drone est contrôlé en 4 degrés de liberté (Ddl), c.-à-d. trois translations plus une rotation. Cependant, un régulateur proportionnel a été mis en place sur le Bebop 2 pour contrôler la vitesse puisque le déplacement horizontal de ce dernier est plutôt contrôlé par ses angles. D'autre part, l'image acquise par la caméra du drone a une latence dont l'impact est atténué par un simple régulateur on-off, c.-à-d. soit en appliquant l'entrée de contrôle ou en entrant en mode stationnaire. Même si la commande de vitesse est calculée par la loi de contrôle à 30 Hz, elle n'est appliquée que lorsque le coût est supérieur à un certain seuil ; autrement, le mode stationnaire du drone est activé. Les expérimentations sont réparties en tâches de **positionnement** et de **navigation**. Les manipulations de chaque tâche sont divisées en intérieur et extérieur :

- **Positionnement à l'intérieur :** Deux manipulations sont effectuées dont le but est d'asservir une image virtuelle désirée, à 90 cm d'hateur, sur une scène qui correspond au maillage texturé imprimée sur une bâche. La texture du maillage est une image satellite issue de *Google Earth.* Le robot n'est pas perturbé dans la première expérience, mais un ventilateur est activé en mode oscillation derrière le drone dans la seconde.

- **Positionnement à l'extérieur :** L'asservissement considère une image virtuelle désirée à 10 m au-dessus du sol, tout en étant perturbé par un vent léger. La scène correspond à un carrefour et l'image virtuelle désirée est rendue à partir d'une image satellite du carrefour issue de*Google Earth.*

- **Navigation à l'intérieure :** Elle consiste à naviguer en aller-retour sur un chemin d'une longueur de 55 cm à 90 cm d'hauteur. Le maillage texturé et la scène utilisée dans l'autre manipulation intérieure sont ici les mêmes.

- **Navigation à l'exterieur** : La manipulation finale a lieu dans un *parking.* Le maillage texturé est acquis par un *snapshot* du *parking* lorsque le robot vol à

30 m au-dessus du sol. Les poses désirées sont séparées de 8 m et son définies à 13 m au-dessus du sol. La navigation hybride par asservissement est également un aller-retour.

Les **résultats** sont fournis ci-après :

- **Tâches de positionnement :** En ce qui concerne la première expérience, le drone est resté en place en minimisant le coût jusqu'au remplissage de la mémoire des données. Cette expérience est concluante car en mode stationnaire natif (interne au Bebop 2) le drone dérive et diverge rapidement dans 60 s. Le coût varie dans une grande mesure, car même de petits déplacements à basse altitude provoquent des changements importants dans les images. Lorsque le ventilateur est actionné, le drone reste en mode d'asservissement potitionnel pendant 163 s jusqu'à ce qu'une rafale de vent fasse diverger le robot. Dans cette expérience, la loi de commande a calculé des vitesses plus élevées pour corriger l'erreur de positionnement. Dans l'expérience de la commande hybride de navigation d'un carrefour à haute altitude (à l'extérieur), la loi de contrôle n'est appliquée qu'au début de la manipulation et lorsqu'un piéton provoque une augmentation du coût. Sinon, le mode stationnaire est activé puisque les perturbations de l'image sont plus faibles lorsque le drone vole à haute altitude.
- **Tâches de navigation :** À l'interieur, le système a utilisé un chemin visuel qui contient 11 images désirées. La navigation hybride par asservissement a réussir pour un trajet en aller-retour. À l'extérieur, le chemin visuel est composé par 5 images désirée et la navigation hybride par asservissement en aller-retour a réussi. Toutefois, lorsque le trajet aller-retour est recommencé, le drone s'est écarté de la trajectoire et la navigation hybride par asservissement a divergé.

En **conclusion** :

- Seul un chemin rectiligne est considéré ; néanmoins, le vol du drone n'est pas limité à la ligne droite, mais actionné en 4 Ddl. Il faudrai donc prendre cela en compte pour limiter l'impact de légères dérives qui peuvent mener à la navigation hybride à diverger, comme le montrent les échecs du second aller-retour, aussi bien à l'extérieur, qu'à l'interieur.
- Une autre raison forte est qu'une zone sans texture peut exister entre deux poses désirées, et indépendamment du nombre d'images désirées, cette zone pourrait être non navigable par la navigation hybride par asservissement. En outre, un chemin visuel avec moins d'images désirées pourrait être trouvé si la ligne droite n'est pas imposée.

Par conséquent, un autre système de navigation basé sur la mémoire devrait être développé afin de surmonter les problèmes mentionnés ci-dessus.

Système de Navigation Photométrique

PNS est similaire à PT&R puisque les deux génèrent de manière autonome la mémoire visuelle à partir d'un maillage texturé et effectuent ensuite une navigation hybride par asservissement à l'aide de la mémoire. Cependant, la principale différence est que PNS génère la mémoire visuelle en utilisant une approche topologique. Par conséquent, les chemins à l'extérieur de la ligne droite peuvent être explorés si nécessaire, p. ex., si la ligne droite est non navigable. PNS est également basé sur PVS mais en utilisant une autre primitive visuel légèrement modifiée de celui-ci. Cette primitive est plus rapide à calculer par rapport à celle correspondante de PT&R, mais toujours robuste à la variation de la lumière. La *mission* est également définie par deux pose désirées et elle est également composée de deux *étapes* : la *planification du chemin* et la *navigation*. Bien que, ils sont nommés différemment car ils ne suivent pas la même structure de système.

Les *étapes* de PNS sont dépeintes ci-après :

- *Étape de planification du chemin* : Elle commence également par définir les poses désirées sur le maillage texturé. Brièvement, une mémoire visuelle virtuelle est générée de façon autonome en simulant la navigation par asservissement, connue sous le nom de navigation par asservissement virtuelle, dans une approche topologique pour chaque paire de deux poses désirées virtuelles consécutives. Par conséquent, pour chaque paire de cettes poses, l'espace virtuel entre eux est uniformément divisé, dans un graphe de poses, appelé graphe de navigation. Cette division est initialement déterminée par une mesure liée à l'étendue du domaine hybride de convergence, i.e. le domaine de convergence dont l'image désirée est virtuelle. Cette étendue est évaluée expérimentalement sur la scène, mais en utilisant une image virtuelle, rendue à partir du maillage texturé. Une fois le graphe des poses est créé, il est simplement connecté. Ensuite, A* cherche un chemin à évaluer en simulant une navigation par asservissement virtuelle et en appliquant le critère d'évaluation virtuels. Ce critère est principalement basé sur l'erreur de positionnement. Si un bord lié à la trajectoire est considéré comme non navigable, il est déconnecté du graphe de navigation. Si cela se produit, le processus de recherche de chemin est relancé jusqu'à ce qu'un chemin navigable soit trouvé ou que le graphe de navigation doive être réinitialisé avec une plus grande densité. À ce stade, si la navigation virtuelle par asservissement avec la mémoire visuelle virtuelle a réussi, son chemin visuel est stocké. Par conséquent, la mémoire visuelle contient toutes les mémoires visuelles virtuelles dont les navigations virtuelles par asservissement ont réussi.

- *Étape de navigation* : Elle est très similaire à l'étape correspondante de PT&R. La principale différence réside dans les critères de mise à jour qui sont basés sur la dérivée de la commande de vitesse appliquée au robot et calculée par la loi de contrôle, au lieu d'être basé sur le coût. Ceci est considéré comme une amélioration puisque les critères sont liés à une convergence de la loi de contrôle vers un *extremum*, idéalement un *minimum*. Notez que ce critère est mieux adapté à une tâche de positionnement hybride puisque le zéro dans le coût n'est pas susceptible d'être atteint lorsque l'image désirée n'est pas obtenue à une pose désirée mais rendue à partir d'un maillage texturé.

Les manipulations qui ont évalué les performances de PNS ont été réalisées à l'aide d'un Stäubli TX2-60 qui est un bras robotique. En effet, ce robot est conçu pour suivre un chemin précis contrôlé par une caméra montée sur l'effecteur d'extrémité. Les manipulations sont regroupées en deux classes de missions : **Ddl Actuation** et **robustesse à la scène et au contenu texturé**. Dans toutes les manipultaions, la *mission* est définie par deux poses désirées et le graphe de navigation est défini en 2D, même si les poses sur le graphe soient définies en 6 Dof. De plus, la navigation hybride par asservissement est définie en 2 Ddl pour toutes les manipulation à l'exception de deux qui appartiennent au premier groupe. Les expériences qui appartiennent au premier groupe étudient la performance de PNS lors de la navigation dans différents Ddl plutôt que d'étudier les difficultés générées par la scène texturée ou le maillage. Cettes manipulations sont décrites ci-après :

- **Cas nominal :** Il s'agit du cas le plus simple de toutes les manipulations puisque la *mission* est exécutée comme on l'a décrit précédemment.
- **Cas ex situ nominal :** Cette manipulation est similaire à la précédente sauf que la division initiale du graphe de navigation est déterminée par une mesure de l'étendue du domaine de convergence virtuel, au lieu du domaine hybride ; par conséquent, une telle évaluation est exécutée en utilisant le maillage texturé et la fonction de capteur. Cette situation est une solution alternative lorsque le robot n'est pas disponible pour effectuer l'évaluation expérimentale du domaine hybride de convergence *in situ*.
- **Cas approximatifs :** Le but de cettes manipulations est d'étudier la robustesse de la mémoire visuelle générée en estimant la mesure liée à l'étendue du domaine hybride de convergence. Cettes mesures sont estimées à étendre le domaine de la convergence en 4 et 6 Ddl. De plus, contrairement au reste des manipulations du PNS, la navigation hybride par asservissement est réalisée en 4 et 6 Ddl.

Le deuxième groupe se concentre sur l'étude de deux difficultés concernant la navigation basée-vision. Les deux manipulations génèrent leur mémoire visuelle de manière *ex situ*. Cettes manipulations sont décrites ci-dessous :

- **Cas de zone sans texture :** Une zone sans texture est superposée grossièrement au centre de la scène et maillée afin d'empêcher la navigation basée-vision sur la ligne droite.
- **Cas de maillage obsolète :** Un maillage texturé qui ne représente plus la scène est utilisé pour générer la mémoire visuelle. Cette manipulation teste l'une des circonstances les plus difficiles des systèmes de navigation basés sur des modèles, lorsque le modèle préalable n'est pas précis. En effet, la manipulation amène le PNS à l'une de ses limites en générant une mémoire visuelle à partir d'un maillage généré 10 ans avant la scène mais en exécutant la navigation hybride par asservissement sur ce dernier. Certains bâtiments et ombres, qui apparaissent dans les images virtuelles désirées liées au chemin visuel, ne sont pas présents dans la scène.

Les **résultats** des expériences sont présentés ci-dessous :

- **Actionnement du Ddl :** Les chemins navigables de toute cette expérience, à l'exception du cas *ex situ*, ont été trouvées le long de la ligne droite pendant l'*étape de planification du chemin*. Plus tard, la navigation hybride par asservissement a toujours réussi, à l'exception du cas approximatif de 6 Ddl. Cependant, la loi de contrôle définie en 6 Ddl a conduit à atteindre la limite de la portée du robot. Ce problème n'est pas géré par la loi de contrôle ni par le contrôle de bas niveau du robot.
- **Robustesse à la scène et au contenu maillé :** Les mémoires visuelles de cettes manipulations ont été générées de manière *ex situ*. Les chemins navigables n'ont pas été trouvés sur la ligne droite. Le navigation hybride par asservissement a réussi dans le cas de la zone texturée, mais il a échoué dans le cas du maillage obsolète. Cependant, le robot a parcouru une distance qui représente presque la moitié de la distance entre les deux poses désirées.

En **conclusion** de l'évaluation du PNS :

- Lorsque la mémoire visuelle est générée *ex situ*, il est peu probable de trouver un chemin sur la ligne droite. Cette situation est liée à une approximation générale, bien qu'insuffisante, à la navigabilité de la scène représentée par le maillage texturé. Cette approximation ne prend pas compte de la différence entre la scène et son maillage puisqu'il n'est exécuté que utilisant ce dernier.

- Lorsque la mémoire visuelle est générée de manière *in situ*, le graohe de navigation a été initialisé de telle sorte que le chemin navigable a été trouvée sur la ligne droite. C'est une situation favorable puisque la mémoire visuelle est calculée plus rapidement, parce que le premier chemin proposé par A* est validé, et ce chemin est le plus court, car il est lié à la ligne droite qui croise les deux poses désirées.

- Sauf pour le cas le plus difficile, c.-à-d. le cas de maillage obsolète, la loi de commande a produit une navigation de hybride par asservissement avec succès. En excluant, certainement, le problème qui s'est produit dans le cas d'environ 6 Ddl lié à l'articulation du robot.

- D'autres chemins que celui lié à la ligne droite sont explorés. Cette situation est pratique lorsqu'aucun des chemins sur la ligne droite n'est navigable ; c'est le cas lorsqu'une zone texturée est intentionnellement superposée au centre du maillage texturé et de la scène.

Architecture de Navigation Robotique

Par conséquent, SNA est créé comme une généralisation du PNS. Similaire au PNS, la *mission* est définie par un groupe de poses désirées dans la scène. De plus, la *mission* est composée des deux mêmes *étapes*. Cependant, il est limité aux scènes statiques car il repose sur un modèle préalable et le succès de la navigation hybride par asservissement dépend de la précision du modèle. SNA comprend une plus grande variété de capteurs, de modèles et de scènes. Voici quelques systèmes adaptés à l'architecture :

- Véhicules terrestres et aériens équipés de caméras qui naviguent dans un environnement extérieur représenté par un modèle de surface dense (ou DSM en anglais).

- Robots sous-marins équipés de capteurs à ultrasons qui perçoivent le fond de la mer représentés par un modèle d'élévation dense (ou DEM en anglais).

- Robots terrestres équipés de capteurs ou de caméras basés sur la portée qui naviguent dans l'environnement intérieur représentés par un modèle de conception assistée par ordinateur (ou CAD en anglais) ou des cartes d'occupation (capteurs basés sur la portée uniquement)

- Robots terrestres équipés de microphones qui naviguent dans un environnement avec différentes sources sonores (bien que théoriquement parce que la scène est difficile à modéliser en raison de la présence de bruit et de réverbération
- Bras robotiques équipés d'une caméra perspective où la surface texturée plane est représentée par une image, ce les cas du PNS.

Mots clés : Navigation robotique, planification de chemin, asservissement visuel photométrique.

Résumé

Cette thèse présente trois contributions à la navigation robotique :

La première d'entre elles considère la navigation à l'aide d'une caméra monoculaire. Ce système s'appelle Instruction & Répétition Photométriques (ou PT&R en anglais) car il est basé sur le paradigme *teach-repeat* et l'Asservissement Visuel Photométrique pour traiter le problème de la navigation.

La deuxième contribution est un système de navigation basé-vision inspiré par PT&R, appelé système de navigation photométrique (ou PNS en anglais), qu' explore d'autres chemins que la ligne droite en utilisant une approche topologique. De plus, une navigation simulée évalue la navigabilité d'un chemin représenté par la mémoire visuelle.

La troisième contributionest appelée Architecture de Navigation par Asservissement (ou SNA en anglais) car elle aborde le problème de navigation en concevant ce problème comme une concaténation de tâches successives de positionnement en utilisant une mémoire sensorielle comme référence.

Mots clés : Navigation robotique, planification de chemin, asservissement visuel photométrique.

Abstract

This thesis presents three contributions to robotic navigation:

The first of them is a system that navigates using a monocular camera. Such system is called Photometric Teach & Repeat (PT&R) since it is based on the *teach-repeat* paradigm and Photometric Visual Servoing to tackle the problem of navigation.

The second contribution is a visual-based navigation system inspired by PT&R, called the Photometric Navigation System (PNS), which explores other paths than the straight line using a topological approach. Furthermore, a simulated navigation evaluates the navigability of the path depicted by the visual memory.

The third contributionis called the Servo Navigation Architecture (SNA) as it tackles the navigation problem by conceiving the latter as a concatenation of successives positioning tasks using a sensory memory as reference.

Keywords: Robotic navigation, path-planning, photometric visual servoing.