UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

Método de navegación basado en aprendizaje y repetición autónoma para vehículos aéreos no tripulados

Tesis presentada para optar al título de
Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Lic. Matías Nitsche

Directora de Tesis: Dra. Marta E. Mejail

Director Asistente: Dr. Miroslav Kulich

Consejera de Estudios: Dra. Marta E. Mejail

Lugar de Trabajo: Laboratorio de Robótica y Sistemas Embebidos (LRSE), Departamento de Computación

Buenos Aires, 20 de Febrero de 2016

Método de navegación basado en aprendizaje y repetición autónoma para vehículos aéreos no tripulados

En esta tesis se presenta un método basado en la técnica de *Aprendizaje y Repetición* (*teach & repeat* o TnR) para la navegación autónoma de Vehículos Aéreos No Tripulados (VANTs). Bajo esta técnica se distinguen dos fases: una de aprendizaje (*teach*) y otra de navegación autónoma (repeat). Durante la etapa de aprendizaje, el VANT es guiado manualmente a través del entorno, definiendo así un camino a repetir. Luego, el VANT puede ser ubicado en cualquier punto del camino (generalmente, al comienzo del mismo) e iniciar la etapa de navegación autónoma. En esta segunda fase el sistema opera a lazo cerrado controlando el VANT con el objetivo de repetir en forma precisa y robusta el camino previamente aprendido. Como principal sensado se utiliza un sistema de visión monocular, en conjunción con sensores que permitan estimar a corto plazo el desplazamiento del robot respecto del entorno, tales como sensores inerciales y de flujo óptico.

El principal objetivo de este trabajo es el de proponer un método de navegación tipo T&R que pueda ser ejecutado en tiempo real y a bordo del mismo vehículo, sin depender de una estación terrena a la cual se delegue parte del procesamiento o de un sistema de localización externa (como por ejemplo GPS, en ambientes exteriores) o de captura de movimiento (como por ejemplo ViCon, en ambientes interiores). En otras palabras, se busca un sistema completamente autónomo. Para ello, se propone el uso de un enfoque basado en apariencias (o *appearance-based*, del inglés), que permite resolver el problema de la localización del vehículo respecto del mapa en forma cualitativa y que es computacionalmente eficiente, lo que permite su ejecución en hardware disponible a bordo del vehículo.

La solución propuesta se diferencia del típico esquema de Localización y Mapeo Simultáneo (*Simultaneous Localization and Mapping* o SLAM) mediante el cual se estima la pose del robot (y la de los objetos del entorno) en forma absoluta con respecto a un sistema de coordenadas global. Bajo dicho esquema, como consecuencia, el error en la estimación de la pose se acumula en forma no acotada, limitando su utilización en el contexto de una navegación a largo plazo, a menos que se realicen correcciones globales en forma periódica (detección y cierre de ciclos). Esto impone el uso de técnicas computacionalmente costosas, lo cual dificulta su ejecución en tiempo real sobre hardware que pueda ser llevado a bordo de un robot aéreo.

En contraste, bajo el enfoque propuesto en esta tesis, la localización se resuelve en forma relativa a un sistema de coordenadas local cercano al vehículo (es decir, una referencia sobre el camino) que es determinado mediante un esquema de filtro de partículas (Localización de Monte Carlo). Esto se logra comparando la apariencia del entorno entre la etapa de aprendizaje y la de navegación, mediante el empleo de características visuales salientes (*image features*) detectadas en dichas etapas. Finalmente, utilizando una ley de control simple, se logra guiar al robot sobre el camino, a partir de reducir la diferencia entre la posición aparente de los objetos del entorno entre ambas etapas, producto del desplazamiento y diferencias de orientacion del robot respecto del mismo.

Como parte del desarrollo del trabajo de tesis, se presenta tanto la formulación y descripción del método como el diseño y construcción de una plataforma VANT, sobre la cual se realizaron los experimentos de navegación. Asimismo, se exhiben experimentos tanto con plataformas aéreas en entornos simulados como sobre plataformas terrestres, dado que el método es aplicable también a los mismos. Con los resultados obtenidos se demuestra la factibilidad y precisión de los métodos de localización y navegación propuestos, ejecutando en hardware a bordo de un robot aéreo en tiempo-real. Así, con esta tesis se presenta un aporte al estado del arte en lo que refiere a temas de navegación autónoma basada en visión, particularmente (pero no exclusivamente) para el caso de robots aéreos.

Palabras clave: robótica, navegación, VANT, autonomía, visión

Appearance-based teach and repeat navigation method for unmanned aerial vehicles

This thesis presents a method based on the Teach & Repeat (TnR) technique for the autonomous navigation of Unmanned Aerial Vehicles (UAVs). With this technique, two phases can be distinguished: a learning phase (teach) and an autonomous navigation (repeat) phase. During the learning stage, the UAV is manually guided through the environment, thus defining a path to repeat. Then, the UAV can be positioned in any point of the path (usually at the beginning of it) and the autonomous navigation phase can be started. In this second phase the system operates in closed-loop controlling the UAV in order to accurately and robustly repeat the previously learned path. Monocular vision is used as the main sensing system in conjunction with other sensors used to obtained short-term estimates of the movement of the robot with respect to the environment, such as inertial and optical flow sensors.

The main goal of this work is to propose a T&R navigation method that can be run in real-time and on-board the same vehicle, without relying on a ground control-station to which part of the processing is delegated or on external localization (for example GPS, in outdoor environments) or motion capture (such as VICON, for indoor settings) systems. In other words, a completely autonomous system is sought. To this end, an appearance-based approach is employed, which allows to solve the localization problem with respect to the map qualitatively and computationally efficient, which in turns allows its execution on hardware available on-board the vehicle.

The proposed solution scheme differs from the typical Simultaneous Localization and Mapping (SLAM) approach under which the pose of the robot (and the objects in the environment) are estimated in absolute terms with respect to a global coordinate system. Under SLAM, therefore, the error in the pose estimation accumulates in unbounded form, limiting its use in the context of long-term navigation unless global corrections are made periodically (loop detection and closure). This imposes the use of computationally expensive techniques, which hinders its execution in real time on hardware that can be carried on-board an aerial robot.

In contrast, under the approach proposed in this thesis, the localization is solved relative to a local coordinate system near the vehicle (i.e. a reference over the path) which is determined by a particle-filter scheme (Monte Carlo Localization or MCL). This is accomplished by comparing the appearance of the environment between the learning and navigation stages through the use of salient visual features detected in said steps. Finally, using a simple control law, the robot is effectively guided over the path, by means of reducing the difference in the apparent position of objects in the environment between the two stages, caused by the vehicle's displacement with respect to the path.

As part of the development of the thesis, both the formulation and description of the method, and the design and construction of a UAV platform, on which navigation experiments were conducted, are presented. Moreover, experiments using aerial platforms in simulated environments and using terrestrial platforms are also presented, given that the proposed method is also applicable to the latter. With the obtained results the feasibility and precision of the proposed localization and navigation methods are demonstrated, while executing on-board an aerial robot and in real-time. Thus, this thesis represents a contribution to the state of the art when it comes to the problem of vision-based autonomous navigation, particularly (but not exclusively) for the case of aerial robots.

Key-words: robotics, navigation, UAV, autonomy, vision

A Oma

Agradecimientos

Quiero agradecer a la Profesora Marta Mejail por haberme ofrecido un gran apoyo a lo largo de estos años, no solo viniendo desde el ámbito académico sino también desde lo personal.

Quiero agradecer enormemente a mis compañeros en el LRSE, ya que sin su guía, ayuda y buena onda, afrontar el trabajo que conllevó esta tesis no hubiera sido lo mismo.

Agradezco las discusiones con Taihú Pire, quien siempre logró que tenga una mirada distinta sobre lo que daba por hecho.

Agradezco a Thomas Fischer, por tener siempre la mejor onda y ganas de ayudar en lo que fuera.

Agradezco a Facundo Pessacg por entusiasmarse con cualquiera de los problemas que le planteara, quien me ayudo a bajar a tierra muchas ideas que tenía en el aire y por compartir el gusto por los "fierros".

Agradezco a Pablo De Cristóforis, con quien crecí enormemente, quien fue una guía constante y quien, junto a Javier Caccavelli y Sol Pedre, me abrió las puertas al Laboratorio con la mejor onda, haciéndome sentir parte del grupo desde el primer día.

Estoy agradecido con todos los miembros del grupo en general, con quienes nos animamos a hacer crecer un grupo de investigación casi desde sus cimientos, trabajando incontables horas y haciendo los mejores de nuestros esfuerzos.

Agradezco a todos aquellos que ponen su granito de arena en desarrollar el software libre en el que casi totalmente se basó mi trabajo de tesis y agradezco la posibilidad de hacer una devolución también contribuyendo mis desarrollos.

Agradezco la posibilidad que me ofreció la Educación Pública, y a quienes la hicieron y hacen posible, permitiendo mi formación y desarrollo personal en el país.

Agradezco a mis padres, quienes hicieron posible que me dedicara a algo que me apasiona y que va más allá de un trabajo.

Agradezco a mi hermana por siempre entenderme y apoyarme.

Agradezco a Oma, a quien esta tesis está dedicada, por haber estado presente en mi defensa de Tesis de Licenciatura.

Por último y no menor, estoy enormemente agradecido con Ale, por compartir la pasión por la fotografía y la ciencia, con quien siento que crecí en todo sentido, quien me dio coraje y fuerza tanto para afrontar mis miedos como para alcanzar mis anhelos, con quien soñamos juntos, quien me bancó durante mis sesiones maratónicas de escritura de tesis, y por infinidad de razones que no entrarían en esta tesis.

A todos y a los que no nombré pero debería haber nombrado,

Gracias.

Contents

Chapter 1

Introduction

In the field of mobile robotics, the problem of autonomous navigation can be described as the process of finding and following a suitable and safe path between starting and goal locations. As navigation is one of the basic capabilities to be expected of a mobile robot, great research efforts have been placed on finding robust and efficient methods for this task. In particular, the focus is generally placed on autonomous navigation over unknown environments. In other words, the robot does not have any information of its surroundings in advance (i.e. a map) nor of its position relative to some reference frame. Thus, in addressing the autonomous navigation problem, many proposed solutions consist in first solving the localization and mapping tasks first.

By far, the most widely studied problem in the recent years corresponds to SLAM, for *Simultaneous Localization and Mapping*, where the goal is to solve both tasks at the same time by incrementally building a map while estimating the motion of the robot within it, iteratively. Many successful SLAM solutions have been proposed over the years, such as MonoSLAM[1], PTAM[2] and, more recently, LSD-SLAM[**?**], all of which use vision as the primary sensor. Under the SLAM based approach to the navigation problem, both a precise estimation of the robot pose and a globally consistent map of the environment are sought in order to be able to plan and follow a global path leading to the desired location.

For the purpose of map building and pose estimation, different sensors have been used throughout the history of mobile robotics. Initially, basic sensors such as infra-red or sonar range-finders where used to asses distances to obstacles and progressively build a map. More recently, the field of mobile robotics has greatly evolved thanks to the development of new sensing technologies. While current laser range-finders have been increasingly been used due to their extremely high precision, they are generally costly solutions and have considerably high power requirements. Nowadays, the most widely used sensor is the camera, due to the emergence in recent years of affordable and high-quality compact imaging sensors. Images, in contrast to range readings, have the great potential of not only be used for recovering structure but also appearance of the surrounding environments and objects within in with which to interact. Moreover, unlike range-finders, cameras are passive sensors and do not pose any interference problems even when several are employed in the same environment. The downside, in this case, is that to harness such sensing power, the challenge is then placed on finding appropriate image-processing algorithms which are able not only to deal with real-time constraints but also with the limitations of current imaging technologies (low frame-rate, dynamic-range, resolution). Moreover, these algorithms need to tackle the ambiguities inherent to such a sensor which translates a 3D world into a 2D image, thereby loosing valuable information. For these reasons, current visual SLAM-based approaches are attempting to not only solve issues such as changing illumination or weather conditions in outdoor environments or reconstructing environment geometry from texture-less regions, but also to find suitable mathematical frameworks which are able to both precisely and efficiently recover the 3D structure of the environments and the motion of the robot, in the context of uncertainties, sensor noise and numerical issues which exist when dealing with the real world.

1.1 Autonomous Navigation of Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs) have tremendous potential for solving tasks such as search & rescue, remote sensing, among others. Recently, embedded processing and sensing technologies have become widespread, allowing the development of unmanned aerial vehicles available nowadays from various companies in the form of closed commercial products, either for entertainment or business purposes such as photography or filming. In particular, there's a widespread interest that in the future UAVs become more important and will be used for tasks such as transporting small loads. This has motivated the research of autonomous navigation methods that can not only solve these tasks, but do so efficiently and safely.

(a) Medium sized UAV, carrying multiple sensor for autonomous navigation.

(b) Micro UAV, featuring a forward camera which streams images wirelessly.

Figure 1.1: Example aerial platforms.

Navigation systems found in current UAVs consist of hardware units, called *autopilots*, which integrate various sensors such as Inertial Measurement Units (IMUs) and GPS (Global Positioning System) receivers. Autonomous navigation is nowadays addressed in aerial platforms using solutions strongly based on GPS for position estimation. While this generally allows for fairly precise navigation there is an important limitations to this technology. GPS requires a solid lock simultaneously to several satellite signals, otherwise the position estimate will quickly degrade. This happens fairly commonly, for example in urban canyons. Moreover, when attempting to perform autonomous navigation in indoor scenarios, GPS signal is simply not available.

These limitations have driven mobile robotics researchers into finding navigation methods based on other sensing systems, such as vision. However, for the case of UAVs, the limited payload capabilities of a typical aerial platform further difficult the use of navigation methods which rely on visual SLAM approaches for pose estimation, due to size and weight of the processing hardware usually required. While the capabilities of embedded hardware rapidly advance every day, further opening the possibilities of executing computationally expensive algorithms on-board an aerial platform, state-of-the-art SLAM methods still require a great deal of available computational power. This, in turn, increases power requirements, negatively impacting over the platform weight (which further restricts flight-time) and size (where smaller UAVs would be safer for human interaction). For these reasons the study of computationally less demanding localization and navigation techniques becomes attractive for the case of UAVs.

1.2 Teach & Replay Navigation

When focusing on particular application scenarios, the general problem of autonomous navigation can be dealt with by employing alternative strategies to the usual SLAM-based estimation methods. For example,

in applications such as repetitive remote sensing or inspection, autonomous package delivery, autonomous robotic tours, or even sample-return missions in space exploration, there is a need to repeat a given path during which the particular task to be solved is performed (acquiring sensory data, transporting a load, etc.).

For this type of applications, while it is certainly possible to employ typical navigation strategies where a globally consistent map of the environment is built and the robot is precisely localized within it, there are alternative techniques which exploit the benefits underlying the definition of this problem. One such technique is referred to as *teach & replay* (TnR) *navigation*, where a path to be repeated is first taught (usually under manual operation) and later repeated autonomously.

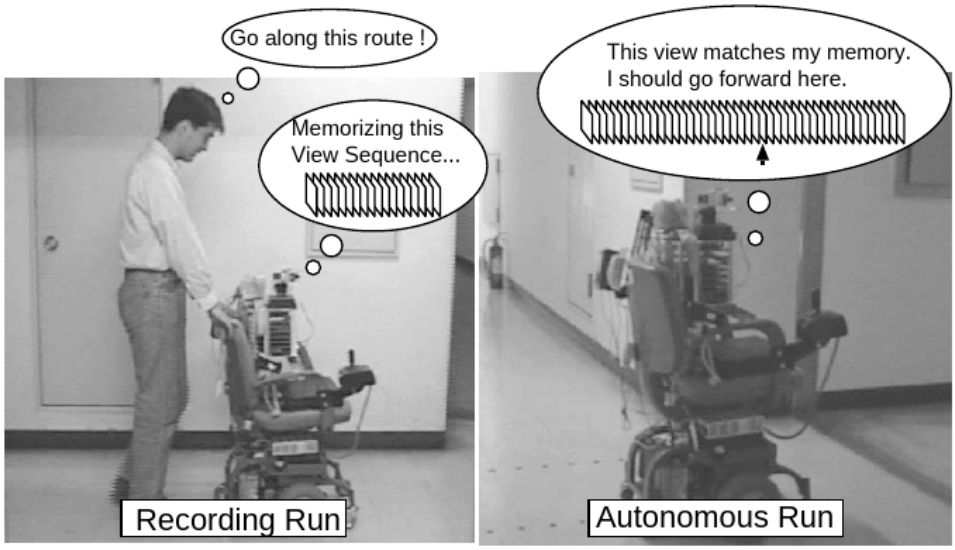


Figure 1.2: Illustration of early Teach & Replay method (image from [3]): a human operator first manually guides a robotic wheelchair while acquiring camera images and then the path can be repeated autonomously by the robot.

The difference in how the navigation problem is addressed with TnR navigation lies in the fact that having already learned a particular path to be followed allows for a relative representation of the robot's position with respect to said path. This is in contrast to the standard approach where both the environment landmarks and the robots pose are described, possibly with six degrees of freedom, with respect to a single fixed global coordinate frame. By localizing with respect to a local coordinate frame, the map (which describes the learned path) does not have to be globally consistent, but only locally. Thus, instead of comparing two absolute poses (that of the robot and that of the reference along the path) in order to perform path-following, the problem can be decomposed in two steps: determining the portion of the map closest to the robot and obtaining the relative pose with respect to this local coordinate frame. To follow the path, the robot can then be autonomously controlled to reduce this relative error to zero.

This formulation of the navigation problem eliminates the reliance on loop detection mechanisms (i.e. whether the robot is re-visiting a previously mapped location), generally required when aiming for a globally consistent representation and used for the purpose of performing global map optimizations which keeps localization error bounded. Moreover, the computational cost of these expensive algorithms is also avoided, which often scale linearly (at best) to the map size.

1.2.1 Metric versus Appearance-Based

The vast majority of SLAM-based methods usually describe the robot's and landmark's position estimates in metric representation. That is, each element of this estimation is described by means of a coordinate

, measured in real-world units, with respect to a fixed reference frame. While with the TnR navigation approach a relative formulation for these coordinates can be taken, a distinction can be made based on whether the approach is either metric or appearance-based.

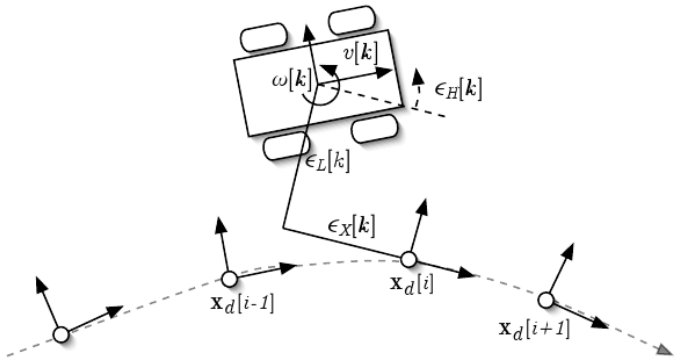


Figure 1.3: Metric-relative localization to local sub-maps: robot state is represented by means of the relative pose $\epsilon_L, \epsilon_X, \epsilon_H$ (for lateral, longitudinal translations and orientation) to a nearby sub-map $\mathbf{x}_d[i]$ (image from [4]).

With a metric approach to TnR navigation, the location of the robot within the learned path can be, for example, described using a six degree-of-freedom (6DoF) pose with respect to a nearby portion of the map, i.e. a *sub-map* (see Figure 1.3 on page 18). In this case, global consistency of the map is not required for navigation since it is assumed that a nearby sub-map will always be chosen as the reference frame. This type of metric TnR approach, though, still generally requires a sensing modality which allows for retrieving the relative pose robustly and precisely, such as laser range-finders or stereo-cameras.

On the other hand, appearance-based approaches avoids the need for determining a metric-relative pose altogether by directly employing sensor readings to perform a so-called *qualitative* localization. That is, localization is reduced to the problem of only retrieving the required information which allows to recognize the overall location w.r.t. the learned map and to perform path-following, for example employing simple image-based visual-servoing control strategies. In this way, a metric-relative pose is not required, which allows to use simpler sensing systems such as monocular vision, with which salient visual features can be extracted and compared to those stored in the map, allowing for localization and navigation.

The issue of monocular versus stereo-vision for landmark and robot self-localization, lies in the fact that the latter has the advantage of enabling the computation of the depth (distance to camera) of perceived elements directly, whereas with the former two views from different positions are required. This *Structure from Motion* (SfM) approach has been widely studied, however it presents considerable challenges arising from the need of extra sensory information (to estimate camera displacement) and robust filtering strategies to deal with sensor noise and ill-conditioned inputs. For these reasons, metric approaches generally employ stereo-vision or even precise laser range-finders. On the other hand, appearance-based approaches become attractive when dealing with monocular vision, which is particularly important when aiming for low-weight and computationally inexpensive solutions for the case of aerial vehicle navigation.

1.3 Image Features for Robot Navigation

Visual navigation methods have to deal with extracting relevant information from camera images. While only recently *direct* methods have appeared, where image pixels are used directly for pose estimation and map building, in recent years the focus was put into extracting salient visual features as a way of reducing the computational load of dealing with the complete image and also of robustly identifying elements which could be tracked and recognized over time, becoming landmarks for localization, map building and navigation. Different kinds of salient image features have been proposed, such as edge-, segment- and point-based,

where the majority of current approaches are focused on the latter.

The difficulty of detecting point-based features lies in the fact that these points should be robustly detected under varying conditions such as scene illumination, image noise, different camera pose or considering even harder aspects such as robustness to occlusions or seasonal differences. At the same time, an important consideration is the computational efficiency of these algorithms, since they are usually employed as a small component in a larger real-time system.

A second challenge after the detection stage, is to be able to uniquely identify a given feature in two images obtained in different conditions. To do so, features generally have an associated *descriptor*, a numerical vector which can be compared under a given distance metric in order to establish *correspondences*. Again, algorithms for descriptor computation are expected to be robust to situations such as those previously described. In the context of mobile robot navigation, image features are generally extracted and tracked over time (matching to previous image frames) or space (matching to others stored in the map). Thus, another consideration is how to perform this matching efficiently. When dealing with large numbers of features, comparing multiple descriptors can easily become computationally demanding.

Figure 1.4: Feature matching: features detected in two different images are matched, despite considerable change in viewpoint of the object.

A great deal of feature extraction and description algorithms have been proposed in the last years. One of the early and most robust algorithms is known as SURF[5] or Speeded Up Robust Features, a faster version of SIFT[6], or Scale-Invariant Feature Transform algorithm, and with invariance to more complex transformations. With SURF, image-features are detected by finding local *maxima* and *minima* over up- and down-scaled versions of the image, using Gaussian smoothing. As these operation can become quite expensive, SURF approximates these by means of box filters, which can be efficiently computed using the so-called *integral image*. Descriptors for each key-point are computed from their neighboring pixels by first determining the orientation of the image-patch and later building a code by measuring the Haar wavelet response of smaller sub-patches. The SURF descriptor thus consists in a real-valued vector which attempts to uniquely encode the appearance of the feature and with invariance to scaling and rotation. In order to establish a correspondence between features extracted in two different images of the same scene, descriptors are matched by nearest-neighbor search in descriptor space. Thus, efficient implementations of k-nearest neighbor (KNN) algorithms are usually employed to also speedup this phase.

While SURF presents itself as a faster alternative to SIFT, the extraction and description algorithms are still very computationally expensive. Furthermore, matching real-valued descriptors involves many floating-point operations which can also represent a limitation for computation on embedded processors, such as those found on-board of relatively small mobile robots. For this reason, recently the use of binary descriptors has been proposed, such as BRIEF[7] or BRISK[8], which aim for the same level of robustness as could be obtained with SURF but with a much lower computational cost, since descriptors can be matched

very efficiently with only a few XOR operations.

Nowadays, thanks to the existence of publicly available implementations of the main feature-processing algorithms, it is possible to not only practically evaluate their efficiency and robustness in real working conditions (as opposed to in artificial benchmarks), but also to try new combinations of extraction-description pairs.

1.4 State of the Art

The necessity of global consistency for accurate path planning and following was already challenged in the early work by R. Brooks[9]. The idea of autonomous navigation without global consistency was proposed, while assuming only local consistency. Here, the map takes the form of a manifold: an n-dimensional topological space where each point has a locally Euclidean neighborhood. This representation allows the environment to be described by means of a series of interconnected sub-maps $\{s_i\}$ [10], taking the form of a graph. Connections specify the relative transformation $H_{i,j}$ between sub-maps s_i, s_j, which are not expected to preserve global consistency (i.e. chaining these transformations leads to a consistent map only within a certain neighborhood).

A manifold map enables a relative representation also of the robot pose, as a pair (s_i, H_p) where s_i identifies the i-th sub-map and H_p a relative transformation to a local coordinate frame defined by s_i. As a consequence, the robot pose has no unique representation, since s_i could be arbitrarily chosen. However, a nearby sub-map is generally used, where local consistency (which impacts the accuracy of H) can be assumed. Furthermore, this map representation has the unique advantage of constant-time loop closure[9], since a loop event can be simply represented as a new connection between two submaps. This connection only serves the purpose of correctly describing the topology of the environment and does not require any global error back-propagation to ensure consistency. Navigation using this representation is thus generally decomposed in two levels: planning over the topological map and planning over a locally metric neighborhood.

Most TnR approaches were based on the same principle of local consistency for autonomous navigation. In some works[11, 12, 13] laser range-finders were employed to build metric submaps during an initial teaching phase, and later localize against said map. Such methods have allowed for precise navigation (lateral and longitudinal errors less than $10\,\mathrm{cm}$) in challenging environments such as underground mines[12] for the task of repetitive hauling and dumping of loads using large trucks. However, laser-based methods cannot reliably be used in outdoor unstructured environments, where no distinguishable object may lie inside the scanner's range. On the other hand, vision-based systems do not posses this limitations and provide a richer (although more complex) set of information which can be exploited for navigation.

Numerous Visual Teach-and-Replay (VTnR) methods have been proposed over time. Based on the chosen map representation, they can be categorized[14] as purely metric, metric/topological and purely topological. Furthermore, appearance-based approaches can be used to localize the robot and navigate the environment without an explicit and consistent metric reconstruction. Thus, appearance-based approaches generally employ topological maps as the environment representation.

In early VTnR methods, due to the limitations of available computational power at the time, different strategies were employed to simplify the processing of images acquired by the camera. In [15] , artificial landmarks were used to localize the robot against the metric map built during the initial training phase. Alternatively, appearance-based approaches were also proposed with which remarkable results were obtained, in terms of navigation and localization precision and robustness. One such work[3] proposed a View-Sequenced Route Representation (VSRR) to describe the learned path using a sequence of images, which are memorized along with relational information in the form of high-level motion primitives (move forward, turn left or right). To perform autonomous navigation using the VSRR, a localization module first finds a target image using template matching and then a simple control module steers the robot (while moving forward at constant speed) to minimize the horizontal shift of the template.

This kind of path-following controller can be found in most of the appearance-based approaches based on monocular vision. This lies in the fact that, without a metric approach, it is not possible (without extra information) to distinguish lateral pose errors from heading errors. However, under an appearance-based approach, the same control output (steering towards the path) is correct in both cases without separately recovering the individual errors. This type of navigation is usually referred to as *bearing-only*, since only the relative bearing of the perceived elements, produced as a consequence of the displacement of the robot w.r.t the reference pose, is employed for localization and/or navigation. This is in contrast to metric approaches which estimate the metric position of perceived elements (bearing plus distance, under a polar representation) and with it, the robot's relative pose to the reference.

As a continuation to the VSRR approach, variations of the original formulation of the method were proposed both by the original authors[16] as well as by others[17, 18]. One of the proposed additions was the use of line-based image features, present in most structured scenarios, as natural landmarks. Also, as standard perspective cameras have generally a restricted field-of-view which imposes a limitation for appearance-based bearing-only VTnR methods, omnidirectional cameras were employed in these works.

Further works focused on the use of point-based image features as natural landmarks for VTnR navigation. One such work [19] proposed tracking point-features using the KLT[20] (Kanade-Lucas-Tomasi) algorithm, over images acquired using an omnidirectional camera. This panoramic view of the environment was exploited to obtain a switched control-law for holonomic robots. In contrast to perspective cameras, convergence to a reference pose in the presence of large translations is here possible due to the extra visual information which allows to distinguish translation from orientation errors. While omnidirectional cameras present several benefits for VTnR navigation approaches, many works have continued the search for robust methods based on standard perspective cameras. Likely reasons for this decision are that omnidirectional cameras are generally of low-resolution, more expensive and the captured images are highly distorted.

One notable series of works employing a single perspective camera for VTnR navigation was the one of Chen and Birchfield[21, 22, 23], the first to introduce the term "qualitative navigation". The proposed approach consists of a robot with a forward-looking camera which is used to build an appearance-based topological map during the teaching phase. The qualitative aspect of the approach lies in the use of feature-matching between the current view and a corresponding view in the map for both localization and control, without recovering any metric information. To do so, the map is built from features that can be succesfully tracked (using the KLT algorithm) between pairs of so-called *milestone images*, periodically captured during training. To localize the robot, features are then matched between the current view and the expected milestone (the robot is assumed to start from a previously known location). Then, a switching control law is employed, following the typical bearing-only navigation strategy. The authors arrive at the proposed control-law based on the analysis of the constraints imposed by the matching features on the valid motions that allow reaching the goal. They distinguish the area where the correct motion is to simply move forward, the *funnel-lane*, from the areas where turning towards the path is required. According to the authors, with the proposed switching control-law, the motion of the robot would thus resemble a liquid being poured down a funnel, where it would first bounce on the sides and finally be channeled towards the spout (i.e. the target pose for the robot).

Based on the successful VSRR approach[3], large-scale navigation in outdoor environments using an appearance-based VTnR approach was later demonstrated in [24], where a robot navigated a total of $18\,\mathrm{km}$ under varying conditions, such as lighting variations and partial occlusions. The appearance-based approach in this case was based on an omnidirectional camera and the use of a cross-correlation algorithm applied to the captured images for the purpose of along-route localization and relative bearing to the reference pose. It should be noted here that even when using panoramic vision, the proposed control law still follows the bearing-only strategy which does not distinguish between lateral offset and orientation errors. In fact, authors state that only the forward facing portion of the image is employed and also a weighting function is used to de-emphasize features on the sides, where distance to landmarks is generally too small. The need for distant landmarks to guide navigation is said to be required in the convergence analysis presented by the authors. The reason for this is that under bearing-only control, the effect of lateral offset on the relative bearing to features is assumed to be homogeneous, which is a valid assumption only when their

distance to the camera is large enough or, alternatively, when these distances are mostly the same from the camera's perspective.

In contrast to previous approaches where along-route localization is reduced to an image matching task, in [24] this problem is solved using a Markov localization filter. As the bearing-only control law does not require explicitly recovering the full relative pose information, the expected output of the localization is only the distance along the learned route. Thus, the state space of the Markov filter is effectively one-dimensional. The localization approach consists in a fine discretization of the learned route into small localization states ($7\,\mathrm{cm}$ long, for the presented experiments), an odometry-based prediction model and an observation likelyhood function based on the image cross-correlation scores of the hypothesis. The highest scoring hypothesis is taken as the localization estimate and used as a the reference pose for the path-following controller. It should be noted that solving the global localization problem (i.e. localizing without a prior) is not attempted here and thus computation can be restricted to a small set of neighboring hypotheses instead of requiring the processing of the whole state-space.

In the following years, other purely metrical [25, 26, 27] and metrical/topological[28] VTnR methods were proposed. The work by Royer *et al.*[25] actually has great similarities to standard monocular SLAM-based approaches, since it maintains a globally consistent metric path, interlinked as a pose-graph. To ensure global consistency, Bundle-Adjustment is used to minimize global pose errors. Also, a careful key-frame selection is necessary to avoid spoiling the approach when degenerate solutions are encountered (e.g. in the presence of pure rotational motion or bad landmark association). Localization over the map during the navigation phase is solved only initially, since authors note this process takes a few seconds. No odometry information is used in this approach, as the method also targets hand-held camera situations. Given that monocular vision is used, the missing scale factor is roughly estimated by manual input of the total map length. Localization precision is measured to be quite close to that of the RTK/GPS system used as ground-truth information. One particular aspect is that the inevitable pose drift visible after learning a closed-loop path is presented as the authors as being sufficiently small ($1.5\,\mathrm{m}$ error for a $100\,\mathrm{m}$ trajectory) allowing longer paths to be traversed before this is deemed significant. However, this claim was not experimentally proven for longer (of several kilometers) trajectories, where pose drift could be high enough so that tracking (which assumes global map consistency) is no longer possible.

On the other hand, in the metrical/topological framework proposed by Segvic *et al.*[28] global consistency is not assumed and localization (i.e. path-tracking) is performed over a graph-based topological map. Here, metric information is employed but only to assist the topological localization. Unexpectedly, in this work the proposed control-law again follows the bearing-only strategy of other purely appearance-based methods. One possible reason is that scale is not recovered in any way thus not allowing other type of control based on metric information.

Accounting for the difficulties of working using omnidirectional cameras and the limitation of monocular vision, the work by Furgale and Barfoot[29] distinguishes itself from the previously proposed methods in the fact that stereo vision is employed. This approach follows the strategy of a manifold representation of the environment, by means of a series of interconnected metric maps, built from the stereo triangulation of image-features. Global localization is not fully solved explicitly since the robot is assumed to start at a known location. However, a three-dimensional visual-odometry method based on robust feature-matching (using RANSAC) and pose-estimation from epipolar geometric models is employed to precisely track the current sub-map. To perform route-following the three-dimensional pose is down projected to two dimensions and then both lateral offset and orientation error are fed into a control-law based on an unicycle locomotion model [12]. As a result, the authors achieve impressive results by repeating $32\,\mathrm{km}$ long paths with lateral errors under $0.3\,\mathrm{m}$, under challenging environmental conditions.

The work by Furgale and Barfoot, albeit quite successful compared to previous VTnR approaches, includes some aspects that are worth pointing. First, the use of stereo vision implies the reliance on a heavier and more complex sensor than other approaches based on a single camera. Second, as other authors already noted, for a metric approach based on a three-dimensional reconstruction of the environment to be successful, a rich enough set of landmarks needs to be used, which also implies that bad feature associations (outliers) can quickly hinder the pose estimation. In this work, the robust SURF[5] feature detection

and description algorithm was employed. However, this algorithm can be computationally expensive when aiming for real-time execution. To tackle this issue, the authors employed a GPU-based implementation of this algorithm and of the feature matching stage. While for a ground-robot this may not be a limiting aspect, for aerial robots (as is the focus of this thesis), this type of sensing and processing hardware cannot be taken for granted.

One notable case of an appearance-based VTnR method based solely on a monocular camera and a basic form of odometry-based dead-reckoning is the one of Krajnik *et al*[30]. In contrast to previous approaches, along route localization is solely based on odometry-based dead-reckoning. On the other hand, heading is corrected using the typical bearing-only control, based on the relative bearings of matching image-features across views. The learned map is segment based, and thus the control strategy during the repeat phase switches between forward motion at constant speed, for straight segments while heading is controlled from visual information, and a pure rotational motion between segments, based on odometric estimation.

This formulation of the navigation method, while appearing overly simplistic, allowed for robustly repeating several kilometer long paths, as shown in the presented experiments with a ground-robot in an unstructured outdoor environment. The reason for its successful performance is presented in the convergence principle made by the authors. The underlying idea is that, while the use of dead-reckoning for along route localization implies an unbounded increase in pose uncertainty in the forward direction, on the other hand, the heading correction based on visual information reduces position uncertainty in the direction perpendicular to the path. Thus, whenever the robot performs a turn between segments (in the best scenario, perpendicular to the previous one), these directions alternate and now the heading correction will start to reduce the uncertainty in the direction to where it is larger. As the linear motion estimation obtained from an odometer is generally quite precise (in contrast to the rotational motion estimation, which involves considerable wheel skidding), the new growth in uncertainty in the new forward direction (where uncertainty was relatively small so far) is quite low. As a result, under the premise of a learned path which involves periodical turning, the uncertainty in position remains bounded allowing for a successful repetition of the path. On the other hand, as would be expected, this implies that if relatively long segments would like to be learned, the user would have to periodically introduce some turns. This would be necessary in order to restrict the growth in uncertainty in the forward direction, which could have the effect of over/under-estimating the segment length and thus performing a turn to the following one before or after time, possibly spoiling the approach (since image-feature matches would be unlikely). Moreover, the proposed method does not deal with curved paths which greatly limits its applicability to real situations.

In a more recent work[31], the need for a VTnR method similar to the one by Furgale and Barfoot[29], but based on monocular instead of stereo vision (as is the case for many existing and widely used robots), was identified. Thus, a variant of the original approach was presented, based on a single perspective camera. Due to the ever-present problem of the scale ambiguity under monocular vision, the proposed method requires the knowledge of the camera pose with respect to the ground, namely its tilt and ground-distance. Moreover, while the stereo-based approach used cameras tilted towards the floor in order to mostly observe nearby features because of the challenging desert-like experimental environment, in the monocular version this becomes a necessity. Here the ground plane needs to be clearly distinguished for the approach to be valid and, as the authors conclude, this can quickly become a limitation in the prescense of untextured floors or highlights from overhead lamps, quite often found in indoor environments.

In the analysis of other similar existing methods[29], Furgale and Barfoot point out that most appearance-based VTnR approaches assume a planar motion of the robot (i.e. they only deal with a relative bearing angle to the reference and a distance along a planar path). On the other hand, they also point out other metric or metrical/topological approaches [27, 26] where even though metric information is estimated, again a planar motion is assumed. At this point, it could be argued that, for a ground-robot, this assumption would be still valid for a great deal of structured and unstructured environments. On the other hand, in the case of aerial robots, where planar motion cannot be assumed, this could indeed be a limiting factor. However, several VTnR methods have effectively demonstrated that the simple formulation of appearance-based methods can be applied to aerial robots without significant differences. Furthermore, in this thesis, an appearance-based method, originally proposed for ground-robots[30], is extended for aerial robot navigation without

requiring neither a two or three dimensional metric reconstruction.

One example of a VTnR approach for aerial robots is the one by Courbon *et al*[32], which is presented as an extension of the original work targeting ground robots[33]. The proposed method is based on a hybrid metrical/topological representation, where a graph of key-frame images is built during the learning phase and a metric relative pose is estimated (w.r.t. to these key-frames) during the repeat phase. Localization is hierarchical: a set of candidate key-frames is retrieved by a global matching step based on a global image descriptor, and from these the best candidate is selected by a typical feature-based matching approach (detecting Harris[34] corners and comparing image-patches using Zero-mean Normalized Cross Correlation). The shortcoming of this approach, common to most metric VTnR systems based on monocular vision, is the missing scale. The authors indicate, without any further description on the particular process, that this parameter is manually added to the system.

The VTnR method by Furgale and Barfoot[29], later applied to monocular vision[31], was also adapted to the case of aerial robot navigation[35]. In this work, monocular vision is also employed. However, a fully downward looking camera was used. The proposed method is based on the assumption of motion over a flat floor. After extracting features, the distance to the ground plane is directly obtained from the sonar rangefinder (used by the robot to estimate and control vertical motion). This sensor thus allows to overcome the scale ambiguity and recover the horizontal velocity w.r.t. the ground plane. This system is then used as a visual odometer to estimate the motion with respect to the current sub-map, along with the same feature-based tracking of sub-maps employed in the original stereo-based formulation. In the experimental evaluation, the training phase is not performed by manual flight but by the manual motion of the robot while attached to a rolling structure. This guarantees constant altitude and, more significantly, a parallel orientation with respect to the ground. A short trajectory is repeated in a controlled environment, successfully demonstrating the validity of the approach. The authors point out two main shortcomings of the method. First, the limited resolution of the camera difficult the extraction and matching of image features when flying at low altitude. Second, the method depends on the presence of a strongly textured floor for sufficient features to be detected. These limitations, arising from the use of downward looking cameras for localization and navigation, motivate the use of using forward looking cameras, as it can be found in other works. Finally, the proposed method is still based on the GPU-based processing pipeline and thus all computation is performed off-board.

Following the qualitative navigation approach of Chen and Birchfield[21, 22, 23], the work by Nguyen, Mann, and Gosine [36] proposes a simple extension of the original approach, for the case of aerial robot navigation. Here, the funnel lane concept is simply extended to the vertical dimension, allowing for the robot not only to control the heading but its vertical velocity. As in the original method, for the horizontal dimension the relative bearing to the matching feature is employed. For the vertical dimension, the elevation of the feature (computed from the y pixel coordinates of the matching pair) is used. Again, a simple switched control law is used to direct the aerial robot towards the goal. Furthermore, the orientation and altitude, as estimated by means of the on-board magnetometer and sonar range-finder, are fused (using a complementary filter) with the corresponding vertical and horizontal deviation, estimated from visual information. It should be pointed that, while the magnetometer allows detecting an orientation error w.r.t to the reference pose, the horizontal deviation detected from visual information could be a result of both an orientation error and a lateral offset. Thus, fusing both measurements is not theoretically correct. In any case, lateral offset can generally be assumed to be small enough so that this inconsistency has no real impact. The authors not in their conclusions that the along-route localization, based on a simple ratio of matched to total features, would deserve an improved solution. Again, even though in this case the computational requirements of the method do not appear to be a limiting aspect, on-board execution is not demonstrated.

More recently, the work by Martinez-Carranza *et al*[37] presented a metric approach for the auto-retrieval of an aerial robot, a particular case of TnR navigation where after a long manual flight the robot is to be returned autonomously to its starting point. In contrast to prior approaches, computation is performed on-board. However, the proposed method is highly dependent on depth information for the initialization of 3D points. For this purpose, experiments were demonstrated using two aerial platforms, one with an RGB-D sensor and the other with a stereo camera.

Among the more recent literature, the work by Do *et al*[38] proposes an interesting strategy which incorporates concepts from both metric and appearance-based approaches. Most works employing metric pose estimation are based on the estimation of the camera pose from epipolar geometry, which is prone to fail if insufficient baseline is present (as in the case of pure rotation w.r.t. to the reference) or if most landmarks are too far away (where their perceived motion as an effect of camera translation is not detectable). On the other hand, appearance-based approaches mostly assume that enough distant features will be visible or that most features are within a similar range of distances. In this case, even while a translation cannot be distinguished from a rotation, a pure rotation can be assumed for the purpose of a bearing-only control while still resulting in successful path-following in most situations. In this sense, Do *et al* test for both approaches simultaneously by performing five- and two-point RANSAC estimations in parallel. The former allows to individually recover relative translation and rotation, but only in the presence of sufficient baseline and landmarks at different depths. On the other hand, the latter assumes a pure rotation. The distinction on whether the relative pose to the reference is of short or large baseline (i.e. pure rotation or rotation-translation) can be thus made by comparing the number of outliers obtained from each approach. This further allows to employ a switching controller which will only control the robot's heading in the presence of pure rotation and both heading and translation, in the other case. As another important aspect, the proposed method is able to run fully on-board a small aerial robot.

The aforementioned approach is based on an off-line reconstruction of the path (learning phase), which is described by a sequence of images acquired by a hand-held camera. The limitation here is that this reconstruction is said by the authors to take several minutes. This has the implication that, if during the repeat phase the operator wishes to extend the the path by switching again to learning phase, under this approach it would seem unfeasible. In contrast, this use case is specifically addressed in the present thesis. Another aspect of the work by Do *et al* is that the absence during the learning phase of other sensory data (such as the downward looking optical-flow sensor or the vehicle's orientation estimation, both available in the platforms used for experiments) signifies that if visual tracking should fail (for example, if insufficient image features are detected and/or matched) the method would also fail. Finally, since global localization is reduced to an image matching problem (by matching FREAK[39] features extracted from the current image to those stored during the learning phase, using a Vocabulary Tree structure[40]), if tracking fails the proposed strategy is to blindly repeat the last control action during a brief period of time and to periodically attempt re-localization. Since this could result in a collision, the aerial platform includes a series of sonar range finders which are used to avoid hitting obstacles.

1.5 Thesis Contributions

This thesis presents a novel appearance-based navigation method for Unmanned Aerial Vehicles using monocular vision. As the proposed method follows the teach & replay technique, it is suitable for scenarios where a given path needs to be periodically repeated to gather sensory information, for example for remote inspection or search & rescue operations. The proposed approach presents some similarities to other methods found in the literature. In particular, the underlying principle used for navigation and localization is based on the original approach by Krajnik *et al[30]*. Initially, a very simple adaptation of the control law was presented[41], however the navigation method was mostly left unchanged. A second work was then presented[42] which proposed the use of vision-based along-route localization as a replacement for pure dead-reckoning. In the present thesis, the aforementioned work was completed by also presenting the navigation method under the new localization scheme.

Compared to previous existing works on the topic of VTnR navigation applied to aerial robots, this thesis presents the following contributions:

- A robust VTnR method based on the use of a single forward-looking perspective camera, using appearance cues for navigation, effectively avoiding the scale-ambiguity problem. As no metric information is reconstructed, the method does not suffer from typical problems arising from insufficient matching landmarks, typically leading to ill-conditioned systems of equations for the problem of pose

estimation.

- A robust appearance-based localization scheme, based on the Monte Carlo Localization[43] particle filter approach, over a manifold map representation, using a topological graph which allows repeating arbitrarily long paths without being affected by pose-drift. More importantly, this localization system extends the original approach of Krajnik *et al[30]* to the case of aerial robots, whose motion estimation systems is generally not precise enough for a purely dead-reckoning based solution. Finally, this method is not based on image snapshots or *key-frames* but on continuous feature-tracking and matching, thus avoiding most problems common to such approaches.
- A generalized solution of the classical TnR problem formulation, which not only allows repeating a given path but to easily extend the map on-demand, by teaching outward branching paths. During navigation, with this approach, any sub-path can be followed by first finding a topological path leading to the goal and then following only the required portions of the complete map. In this manner, the learned map is no longer simply a sequential path but a tree-shaped graph. This extension to the classical TnR navigation problem is a first step towards a fully general solution which allows building maps as arbitrary graphs involving cycles, which could be detected using any appropriate algorithm. Loop closure in this case would be solved in constant time by simply adding a new connection to the graph.
- A fully on-board implementation comprising the complete navigation and localization system, running on lightweight hardware, and based on efficient image-feature extraction and binary-descriptor matching algorithms. The localization method is of constant complexity with respect to map size, and can be tuned to balance localization quality over computational cost.
- An open-source ROS (Robot Operating System) based implementation of the proposed approach, which allows other researchers to easily compare similar but alternative VTnR methods. Since none of the most established VTnR methods have been publicly released, this point is a considerable contribution to the mobile robot research community.
- A VTnR method which takes advantage of the extra sensory information found on aerial robots, most specifically an Inertial Measurement Unit, to compensate for the particular motion of aerial vehicles, but which can still be applied to the case of ground-robots since no particular assumptions are made regarding the platform.

Furthermore, this thesis also includes:

- The design and construction of an experimental aerial platform for performing vision-based navigation experiments
- An experimental evaluation of the proposed navigation method on aerial vehicles and ground robots under controlled and uncontrolled conditions

1.6 Thesis Outline

The remainder of this thesis is organized as follows. First, in Chapter 2, the underlying principles of appearance-based navigation are presented. Then, in Chapter 3, the complete method is presented in detail. Chapter 4 describes the experimental aerial platform used to evaluate the performance of the proposed method. The results obtained with these experiments are shown in Chapter 5. Finally, in Chapter 6 conclusions and future work are given.

Capítulo 1

Introducción

En el campo de la robótica móvil, el problema de la navegación autónoma se puede describir como el proceso de encontrar y seguir un camino adecuado y seguro entre las ubicaciones de partida y la meta. Como la navegación es una de las capacidades básicas que se espera de un robot móvil, grandes esfuerzos de investigación se han colocado en la búsqueda de métodos robustos y eficientes para esta tarea. En particular, el foco se coloca generalmente en la navegación autónoma sobre entornos desconocidos. En otras palabras, el robot no tiene ninguna información de su entorno por adelantado (es decir, un mapa) ni de su posición con relación a algún marco de referencia. Por lo tanto, para abordar el problema de navegación autónoma, muchas soluciones propuestas consisten primero en la solución de las tareas de localización y mapeo.

Por lejos, el problema más ampliamente estudiado en los últimos años corresponde a SLAM, o Localización y Mapeo Simultáneo, donde el objetivo es resolver las dos tareas al mismo tiempo ampliando gradualmente un mapa, mientras se estima el movimiento del robot con respecto al mismo, de manera iterativa . Muchas soluciones SLAM exitosas se han propuesto en los últimos años, tales como MonoSLAM [1], PTAM[2] y, más recientemente, LSD-SLAM [**?**], todos los cuales utilizan la visión como el sensado primario. Bajo el enfoque basado SLAM al problema de la navegación, tanto una estimación precisa de la pose del robot y un mapa globalmente consistente coherente son necesarios con el fin de ser capaz de planificar y seguir un camino global que lleve hacia la ubicación deseada.

A los efectos de la estimación de la pose y la construcción del mapa, diferentes sensores se han utilizado a lo largo de la historia de la robótica móvil. Inicialmente, los sensores básicos, tales como los sensores infrarrojos o los sonares, fueron utilizados para la obtención de las distancias a los obstáculos, los cuales se representaban progresivamente en un mapa. Más recientemente, el campo de la robótica móvil ha evolucionado considerablemente gracias al desarrollo de nuevas tecnologías de detección. Mientras los actuales telémetros láser son cada vez más han utilizado debido a su extremadamente alta precisión, en general son soluciones costosas y tienen requisitos considerablemente altos de potencia. Hoy en día, el sensor más ampliamente utilizado es la cámara, debido a la aparición en los últimos años de sensores de imagen compactos, de alta calidad y asequible. Las imágenes, en contraste con las mediciones de distancia, tienen el enorme potencial de no sólo poder recuperar la estructura del entorno sino también su apariencia y la de los objetos circundantes, con los cuales se busca interactuar. Por otra parte, a diferencia de los telémetros, las cámaras son sensores pasivos y no suponen ningún problema de interferencia, incluso cuando se emplean varios en el mismo entorno. La desventaja, en este caso, es que para aprovechar ese poder de detección, el desafío se coloca entonces en la búsqueda de algoritmos de procesamiento de imágenes apropiados que sean capaces no sólo de hacer frente a restricciones de tiempo real, sino también a las limitaciones de las tecnologías de imagen actuales (baja cantidad de cuadros por segundo, limitado rango dinámico y resolución). Por otra parte, estos algoritmos también necesitan hacer frente a las ambigüedades inherentes a un sensor de este tipo que traduce un mundo 3D en una imagen 2D, lo que implica la pérdida de información valiosa. Por estas razones, los enfoques basados en SLAM visuales actuales están tratando de resolver no sólo problemas como el cambio de la iluminación o las condiciones climáticas de ambientes al aire libre o

la reconstrucción de la geometría en regiones de poca textura, sino también se buscan los modelos matemáticos adecuados que sean capaces de recuperar de manera eficiente y precisa la estructura tridimensional del entorno y del movimiento del robot, en el contexto de incertezas, ruido de los sensores y problemas numéricos que existen cuando se trata con el mundo real.

1.1 Navegación autónoma de vehículos aéreos no tripulados

Los vehículos aéreos no tripulados (VANT) tienen un enorme potencial para la solución de problemas tales como la búsqueda y rescate, el sensado remoto, entre otros. Recientemente, los procesadores integrados y las tecnologías de sensado se han visto fuertemente desarrolladas, lo que permitió el desarrollo de vehículos aéreos no tripulados hoy disponibles como productos comerciales cerrados, ya sea para fines de entretenimiento o de tareas tales como la fotografía o filmación. En particular, existe un interés generalizado de que en el futuro los vehículos aéreos no tripulados sean utilizados también para tareas tales como el transporte de cargas pequeñas. Esto ha motivado la investigación de métodos de navegación autónoma que no sólo puedan resolver estas tareas, sino que también lo hagan de forma eficiente y segura.

(a) Medium sized UAV, carrying multiple sensor for autonomous navigation.

(b) Micro UAV, featuring a forward camera which streams images wirelessly.

Figura 1.1: Example aerial platforms.

Los sistemas de navegación que se encuentran comúnmente en los vehículos aéreos no tripulados actuales consisten en unidades de *hardware*, llamados *auto-pilotos*, que integran varios sensores, tales como unidades inerciales (IMU) y receptores de GPS (Sistema de Posicionamiento Global). La navegación autónoma hoy en día se logra utilizando soluciones fuertemente basados en GPS. Aunque esto permite generalmente para una navegación bastante precisa, una importante limitación de esta tecnología es el requerimiento de una señal sólida en forma simultánea a varios satélites, de lo contrario la estimación de la posición se degradará rápidamente. Esto sucede bastante comúnmente, por ejemplo en cañones urbanos. Por otra parte, al realizar la navegación autónoma en escenarios interiores, la señal GPS simplemente no está disponible.

Estas limitaciones han llevado a los investigadores en robótica móvil a la búsqueda de métodos de navegación basados en otros sistemas de detección, tales como la visión. Sin embargo, para el caso de vehículos aéreos no tripulados, las capacidades de carga útil limitada de una plataforma aérea típica dificultan el uso de métodos de navegación que se basan en soluciones de SLAM visual para la estimación, debido al tamaño y peso del hardware de procesamiento generalmente requerido. Mientras que las capacidades de hardware integrado avanzan rápidamente todos los días, abriendo aún más las posibilidades de ejecución de algoritmos computacionalmente costosos a bordo de una plataforma aérea, en lo que refiere al estado del arte de la técnica SLAM todavía se requiere una gran potencia de cálculo. Esto, a su vez, aumenta los requisitos de energía, impactando negativamente sobre el peso de la plataforma (que restringe aún más el tiempo de vuelo) y el tamaño (donde los vehículos aéreos no tripulados más pequeños resultarían en una opción más

seguras para la interacción humana). Por estas razones, el estudio de técnicas de localización y de navegación computacionalmente menos exigentes se convierte en un gran atractivo para el caso de vehículos aéreos no tripulados.

1.2 Navegación de aprendizaje y repetición

Al centrarse en casos de aplicación en particular, el problema general de la navegación autónoma puede ser tratado mediante el empleo de estrategias alternativas a los métodos habituales de estimación basados en SLAM. Por ejemplo, en aplicaciones tales como de sensado remoto en forma repetitiva, de entrega de paquetes autónoma, de visitas guiadas robotizadas, o incluso de misiones de retorno de muestras en la exploración espacial, existe una necesidad de repetir una trayectoria determinada durante la cual la tarea en particular a ser resuelta es realizada.

Para este tipo de aplicaciones, aunque sin duda es posible emplear estrategias de navegación típicas que utilizan un mapa globalmente consistente del entorno y una localización precisa del robot en el mismo, existen otras técnicas que explotan los beneficios subyacentes a la definición de este problema. Una de tales técnicas se conoce como de aprendizaje y repetición (*teach and repeat* o TnR), en donde un camino que se repite se enseña primero (por lo general bajo la operación manual) y luego repite de forma autónoma.

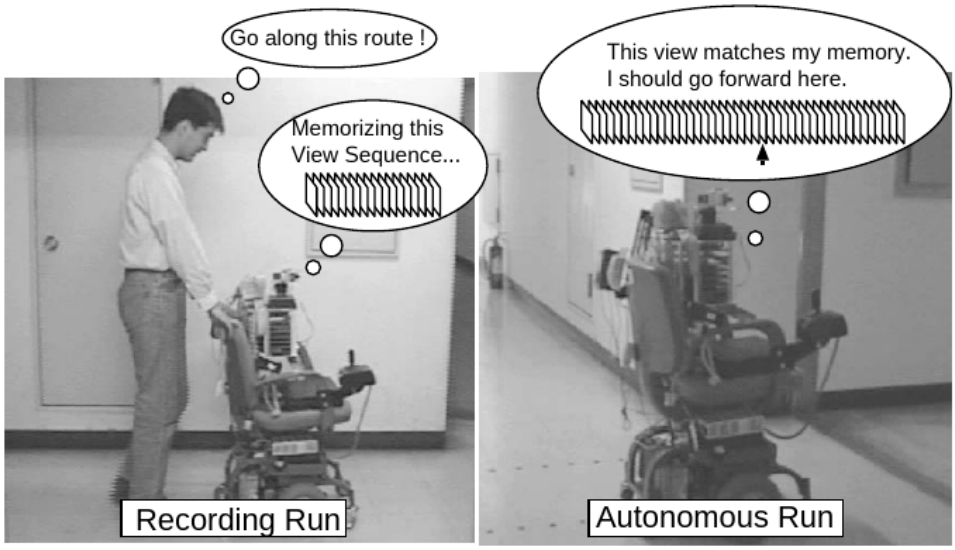


Figura 1.2: Illustration of early Teach & Replay method (image from [3]): a human operator first manually guides a robotic wheelchair while acquiring camera images and then the path can be repeated autonomously by the robot.

La diferencia en cómo el problema de navegación se aborda con la navegación TnR radica en el hecho de que haber ya aprendido un camino particular a seguir permite una representación relativa de la posición del robot con respecto a dicha trayectoria. Esto está en contraste con el enfoque estándar en el que se describen tanto los puntos de referencia del entorno y como el robot, posiblemente con seis grados de libertad, con respecto a un solo marco global de coordenadas fijo. Al localizarse con respecto a un marco de coordenadas local, el mapa (que describe la trayectoria aprendida) no tiene que ser globalmente consistente, sino sólo a nivel local. Así, en lugar de comparar dos poses absolutas (la del robot y la de referencia sobre el camino) con el fin de realizar el seguimiento de camino, el problema se puede descomponer en dos pasos: la determinación de la porción del mapa más cercana al robot y la obtención de la pose relativa con respecto a este sistema de coordenadas local. Para seguir el camino, a continuación, el robot puede ser controlado de forma autónoma para reducir este error relativo a cero.

Esta formulación del problema de navegación elimina la dependencia respecto de los mecanismos de detección de ciclos (es decir, si el robot ha vuelto a visitar un lugar previamente aprendido), lo que generalmente se requiere cuando el objetivo es una representación globalmente consistente y que es utilizada con el propósito de realizar optimizaciones globales al mapa lo que mantiene acotado el error de localización. Por otra parte, el costo computacional de estos algoritmos también se evita, lo cuales a menudo escalan linealmente (en el mejor caso) con respecto al tamaño del mapa.

1.2.1 Métrico versus basado en apariencias

La gran mayoría de los métodos de navegación basados en SLAM por lo general describen la posición del robot y de las referencias mediante una representación métrica. Es decir, cada elemento de esta estimación se describe por medio de una coordenada, medida en unidades del mundo real, con respecto a un marco de referencia fijo. Mientras que con la navegación TnR es posible utilizar una formulación relativa para dichas coordenadas, aún así pueden distinguirse dos enfoques según si la representación es métrica o es basada en apariencias.

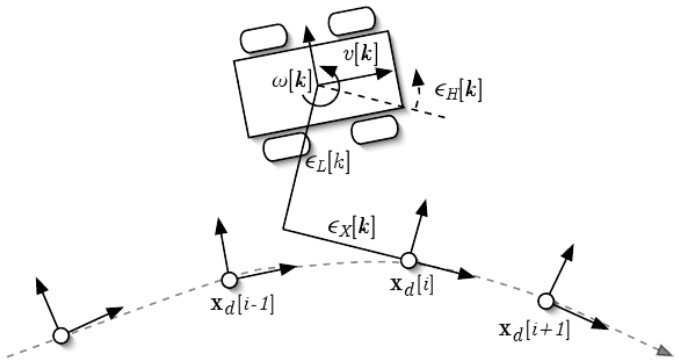


Figura 1.3: Metric-relative localization to local sub-maps: robot state is represented by means of the relative pose $\epsilon_L, \epsilon_X, \epsilon_H$ (for lateral, longitudinal translations and orientation) to a nearby sub-map $\mathbf{x}_d[i]$ (image from [4]).

Con un enfoque métrica a la navegación TnR, la ubicación del robot con respecto a la trayectoria aprendida puede ser, por ejemplo, descrita en un seis grados de libertad, con respecto a una porción cercana del mapa, es decir, una sub-mapa (ver Figure 1.3 on page 30). En este caso, no se requiere la consistencia global del mapa para la navegación ya que se supone que una sub-mapa cercano siempre será escogido como marco de referencia. Este tipo de enfoque TnR métrico, sin embargo, todavía requiere generalmente de una modalidad de sensado que permite la recuperación de la pose relativa en forma robusta y precisa, como es posible mediante telémetros láser o cámaras estéreo.

Por otro lado, los enfoques basados en apariencias evitan la necesidad de determinar una pose relativa en forma métrica por completo empleando directamente lecturas de los sensores para realizar una denominada "localización cualitativa". Es decir, la localización se reduce al problema de recuperar solamente la información necesaria que permita reconocer la ubicación del robot a grandes rasgos sobre el mapa y que permita además realizar el seguimiento del camino, por ejemplo, utilizando estrategias de seguimiento de caminos en forma visual (*visual homing*). De esta manera, no se requiere una pose métrica, lo que permite utilizar sistemas de sensado más simples tales como lo es la visión monocular, con la cual es posible extraer características visuales salientes de las imágenes y comparar estas con las almacenados en el mapa, lo que permite la localización y la navegación.

El tema de visión monocular versus visión estéreo para la localización de referencias y del robot en forma robusta, radica en el hecho de que la última posee la ventaja de permitir el cálculo de la profundidad (distancia con respecto a la cámara) de elementos percibidos en forma directa, mientras que con la primera

se requiere de dos vistas de la escena, adquiridas desde dos posiciones distintas. Esta enfoque es conocido como "estructura a partir del movimiento" (*Structure from Motion* o SfM) y si bien ha sido ampliamente estudiado, presenta retos considerables que surgen de la necesidad de información sensorial extra (para poder estimar el desplazamiento de la cámara) y de las estrategias de filtrado robusto requeridas para poder lidiar con el ruido del sensor y cálculos numéricamente mal condicionadas. Por estas razones, los enfoques métricas generalmente emplean visión estéreo o incluso láser precisos. Por otro lado, los enfoques basados en apariencia se vuelven atractivos cuando en lo que refiere a visión monocular, lo que es particularmente importante cuando el objetivo es obtener soluciones de bajo costo computacional para el caso de la navegación de vehículos aéreos de bajo capacidad de carga.

1.3 Características visual para la navegación

Los métodos de navegación visuales tienen que lidiar con la extracción de información relevante a partir de imágenes de la cámara. Si bien recientemente han aparecido métodos *directos*, donde se utilizan directamente los píxeles de la imagen para la estimación de pose y la construcción de mapas, en los últimos años el foco se puso en la extracción de características visuales salientes como una forma de reducir la carga computacional que conllevaría el procesamiento de la imagen completa, y también de identificar con precisión elementos de la escena con el objetivo de realizar su seguimiento a lo largo del tiempo, convirtiéndolos en referencias para la localización, construcción de mapas y la navegación. Se han propuesto diferentes tipos de características salientes de las imágenes, tales como las basadas en segmentos o puntos, donde la mayoría de los enfoques actuales se centran en el segundo caso.

La dificultad de detectar características visuales puntuales radica en el hecho de que estos deben ser detectados en forma robusta bajo condiciones variables tales como la iluminación de la escena, el ruido de imagen, diferentes poses de la cámara o considerando aspectos aún más difíciles tales como la robustez a las oclusiones o a las diferencias estacionales. Al mismo tiempo, una consideración importante es la eficiencia computacional de estos algoritmos, ya que se emplean habitualmente como un pequeño componente en un sistema de tiempo real más grande.

Un segundo reto después de la etapa de detección, es la habilidad de identificar de forma unívoca una determinada característica en dos imágenes obtenidas en diferentes condiciones. Para ello, las características en general tienen un *descriptor* asociado, que es un vector numérico que puede ser comparado bajo una determinada distancia métrica para establecer correspondencias. Una vez más, se espera que los algoritmos para el cálculo descriptor sean robustos a situaciones como las descritas anteriormente. En el contexto de la navegación en robot móvil, las características de la imagen generalmente se extraen y se realiza un seguimiento en el tiempo (que se corresponda a los cuadros de imagen anteriores) o en el espacio (que se corresponda a otros almacenados en el mapa). Por lo tanto, otra consideración es cómo llevar a cabo esta tarea de manera eficiente. Cuando se trata de un gran número de características, la comparación de múltiples descriptores puede convertirse fácilmente en una tarea computacionalmente costosa.

Una gran número de algoritmos de extracción y descripción de características se han propuesto en los últimos años. Uno de los primeros y más robustos algoritmos se conoce como SURF[5] o *Speeded Up Robust Features*, una versión más rápida de SIFT [6], o *Scale-Invariant Feature Transform*, y con invarianza a transformaciones más complejas. Con SURF, las características visuales son detectadas a partir de la extracción de mínimos y máximos locales sobre versiones de mayor o menor escala de la imagen original, utilizando suavizado Gaussiano. Dado que estos operación puede llegar a ser bastante costosa, SURF aproxima a dichos filtros por medio de *Box Filters*, que pueden ser calculados de manera eficiente utilizando la denominada imagen integral.

Los descriptores de cada punto clave son calculados utilizando los píxeles vecinos, determinando primero la orientación de dicho parche y posteriormente construyendo un código que se obtiene a partir de medir la respuesta del wavelet Haar se pequeños sub-parches. Así, el descriptor de SURF consiste de un vector de valores reales que busca codificar de forma unívoca el aspecto de la característica y con invarianza a escala y rotación. Con el fin de establecer una correspondencia entre las características extraídas en dos

imágenes diferentes de la misma escena, los descriptores son correspondidos mediante la búsqueda del vecino más próximo en el espacio de descriptores. Por lo tanto, las implementaciones eficientes de algoritmos de búsqueda del k-ésimo vecino más cercano (k-nearest neighbor o KNN) son utilizadas normalmente para obtener buenos rendimientos también en esta fase.

Mientras SURF se presenta como una alternativa más rápida a SIFT, los algoritmos de extracción y de descripción son todavía muy costosos computacionalmente. Por otra parte, corresponder descriptores con valores reales implica el uso de numerosas operaciones de punto flotante lo que también puede representar una limitación para la capacidad de cálculo de los procesadores embebidos, tales como los que se encuentran a bordo de los robots móviles relativamente pequeños. Por esta razón, recientemente se ha propuesto el uso de descriptores binarios, tales como BRIEF[7] o BRISK [8], cuyo objetivo es obtener el mismo nivel de robustez que se podría obtener con SURF pero con un coste computacional mucho menor, ya que los descriptores pueden ser comparados de manera muy eficiente con sólo unas pocas operaciones XOR. Hoy en día, gracias a la existencia de implementaciones públicas de los principales algoritmos de procesamiento de características, es posible evaluar prácticamente no sólo su eficiencia y robustez en condiciones reales de trabajo, sino también probar nuevas combinaciones de pares de algoritmos de extracción y descripción.

1.4 Contribuciones de la tesis

En esta tesis se presenta un nuevo método para la navegación de vehículos aéreos no tripulados basada en apariencias utilizando visión monocular. Dado que el método propuesto sigue la técnica de aprendizaje y repetición es adecuado para escenarios donde una trayectoria dada debe ser repetida periódicamente para reunir información sensorial, por ejemplo para la inspección remota o operaciones de búsqueda y rescate. El enfoque propuesto presenta algunas similitudes con otros métodos encontrados en la literatura. En particular, el principio subyacente utilizado para la navegación y la localización se basa en el enfoque original de Krajnik et al *[30]*. Inicialmente, una adaptación muy simple de la ley de control fue propuesta[41], sin embargo el método de navegación fue mayormente el mismo. En un segundo trabajo[42] se presentó la continuación del anterior en el cual se propuso la localización a lo largo de la ruta aprendida, basada en visión y como reemplazo a la basada puramente en la integración del movimiento sensado. En la presente tesis, dicha obra fue completada al presentar también la navegación bajo el nuevo esquema de localización.

En comparación con anteriores trabajos existentes sobre el tema de navegación VTnR aplicado a robots aéreos, esta tesis presenta las siguientes contribuciones:

- Un método VTnR robusto basado en el uso de una sola cámara perspectiva de futuro, basado en la apariencia del entorno como la señal de entrada para la navegación, evitando efectivamente el problema de la ambigüedad de la escala. Dado que no se reconstruye ningún tipo de información métrica, el método no sufre de problemas típicos que surgen de la insuficiencia de puntos de referencia lo que en general conduce a sistemas de ecuaciones mal condicionados, para el problema de la estimación de la pose
- Un esquema de localización robusto basado en apariencias y que sigue el enfoque del filtro de partículas conocido como Localización de Monte Carlo[43], sobre un mapa tipo *manifold*, representado mediante el un grafo topológico que permite la repetición de caminos arbitrariamente largos sin ser afectado por la deriva de la pose. Más importante aún, este sistema de localización extiende el enfoque original de Krajnik *et al [30]* para el caso de robots aéreos, cuyos sistemas de estimación de movimiento en general no son lo suficientemente precisos para utilizar una solución puramente basado en estima. Por último, este método no se basa en tomar cuadros clave (*snapshots*), sino en función del seguimiento continuo de referencias visuales, evitando así la mayoría de los problemas comunes a dicho enfoque.
- Una solución generalizada de la formulación del problema TnR clásico, que no sólo permite la repetición de una trayectoria dada, sino que es posible extender fácilmente el mapa bajo demanda, mediante la enseñanza de caminos salientes. Durante la navegación, con este enfoque, cualquier sub-ruta puede ser seguida buscando primero un camino topológico que conduzca a la meta y luego siguiendo

únicamente las partes necesarias del mapa completo. De esta manera, el mapa aprendido ya no es simplemente una trayectoria secuencial, sino un grafo en forma de árbol. Esta extensión al problema TnR es un primer paso hacia una solución totalmente general que permite la construcción de mapas en forma de grafos arbitrarios que incluyen ciclos, los cuales podrían ser detectados utilizando cualquier algoritmo apropiado. El cierre de un ciclo en este caso sería resuelto en tiempo constante, simplemente añadiendo una nueva conexión al grafo.

- Una aplicación totalmente embebida que puede ejecutar a bordo del robot y que comprende el sistema de navegación y localización completo, en base a la extracción de características visuales a partir de las imágenes y de algoritmos eficientes de establecimiento de correspondencias basados en descriptores binarios. El método de localización es de complejidad constante con respecto al tamaño del mapa, y se puede ajustar para equilibrar la calidad de la localización sobre el coste computacional (afectando el número de partículas).

- Una implementación de código abierto basada en ROS (*Robot Operating System*) del enfoque propuesto, que permite a otros investigadores comparar fácilmente métodos VTnR similares pero alternativos. Dado que ninguno de los métodos VTnR más reconocidos han sido liberados públicamente, este punto es una contribución considerable a la comunidad de investigación robot móvil.

- Un método VTnR que aprovecha la información sensorial extra que se encuentra en robots aéreos, más específicamente una Unidad de Medición Inercial, para compensar el movimiento particular que realizan los vehículos aéreos, pero que es aún así aplicable al caso de robots terrestres ya que no se realizan supuestos particulares sobre la plataforma utilizada.

Por otra parte, esta tesis también incluye:

- El diseño y la construcción de una plataforma aérea experimental para la realización de experimentos de navegación basados en la visión

- Una evaluación experimental del método de navegación propuesto sobre vehículos a

1.5 Esquema de la Tesis

El resto de esta tesis se organiza como sigue. En primer lugar, en Chapter 2, se presentan los principios subyacentes de la navegación basada en apariencias. A continuación, en Chapter 3, el método completo se presenta en detalle. El Chapter 4 describe la plataforma aérea experimental utilizada para evaluar el desempeño del método propuesto. Los resultados obtenidos con estos experimentos se muestran en Chapter 5. Por último, en Chapter 6 se presentan las conclusiones y el trabajo futuro que se desprenden de esta tesis.

Chapter 2

Principles of Bearing-only Navigation

In this chapter, the concepts related to *bearing-only* navigation are presented, which allow for a better understanding of the proposed VTnR approach, described in the following chapter.

2.1 Appearance-Based Path-Following

Path-following can be defined as the task of repeating, as closely as possibly, a previously learned path. Many approaches exists to this problem, mostly depending on the information at hand. In broad terms, it is possible to distinguish the typical *metric* approach, which involves the estimation of the relative-pose error in real-world units (generally in six degrees of freedom), with respect to a reference pose in the path, based on the estimated geometric structure of the environment and the robot's pose. On the other hand, under an *appearance-based* approach, the full relative-pose is not completely obtained. Moreover, the geometric structure of the environment is generally not reconstructed. Instead, appearance cues (i.e. obtained from visual information) which describe the relation between current and a reference pose on the path are directly utilized as inputs to a path-following controller. A path-following controller can then be designed, which guides the motion of the robot while minimizing the relative-pose error.

One such appearance-based path-following technique is known as *bearing-only navigation*, where only the observed difference in relative bearings to landmarks are utilized in order to guide the robot from its current location to a moving reference over the path. To understand the principle governing this navigation strategy, it is necessary first to introduce the concept of *optical-flow*.

2.1.1 Optical Flow

Let $\mathbf{X_c} = \begin{bmatrix} X & Y & Z \end{bmatrix}^t$ be the coordinates of a point in the world, expressed in the local coordinate system C, corresponding to a camera moving w.r.t. a global fixed coordinate frame, with linear and angular velocities, $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^t$ and $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^t$ respectively, the latter expressed in the camera's own coordinate frame (2.1). Then, the relative motion $\dot{\mathbf{X}}_\mathbf{c}$ of the point perceived by C, or *optical-flow*, can be written as

$$\dot{\mathbf{X}}_\mathbf{c} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \boldsymbol{\omega} \times \mathbf{X_c} + \mathbf{v} \tag{2.1}$$

. The image projection of $\mathbf{X_c}$ into camera C, following the pinhole camera-model and represented in homo-

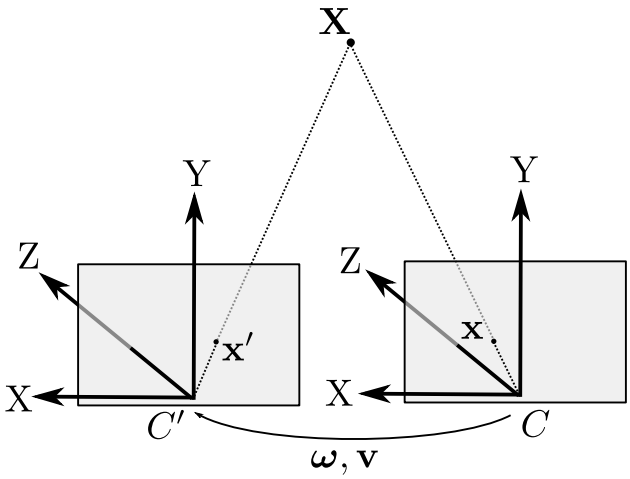


Figure 2.1: Coordinate frames C and C' observing a world point $\mathbf{X}$, with corresponding projections $\mathbf{x}, \mathbf{x'}$.

geneous coordinates, is defined as

$$\mathbf{K}\mathbf{X_c} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X_c} = \begin{bmatrix} f_x X + p_x \\ f_y Y + p_y \\ Z \end{bmatrix}$$

where $\mathbf{K}$ is the intrinsic parameter matrix corresponding to camera C, with f_x, f_y and p_x, p_y being the focal length and principal point, respectively. When dealing with calibrated cameras, this parameters can be recovered. By left-multiplying K^{-1} it is possible to work with *normalized* coordinates, where $f_x, f_y = 1$ and $p_x, p_y = 0$. When transforming back to in-homogeneous coordinates, the image-projection results as

$$\mathbf{x} = \begin{pmatrix} x & y \end{pmatrix} = \begin{pmatrix} \frac{X}{Z} & \frac{Y}{Z} \end{pmatrix} \tag{2.2}$$

. In this way, it is possible to express $\dot{\mathbf{X}}_\mathbf{c}$ in image coordinates as

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \underbrace{\frac{1}{Z} \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}}_{\dot{\mathbf{x}}_\mathbf{v}} + \underbrace{\begin{pmatrix} x \cdot y & -(1+x^2) & y \\ 1+y^2 & -x \cdot y & -x \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}}_{\dot{\mathbf{x}}_{\boldsymbol{\omega}}} \tag{2.3}$$

. Analyzing (2.3) and (2.1), it can be observed that the total flow $\dot{\mathbf{x}}$ can be written as the sum of two independent terms: a rotational flow $\dot{\mathbf{x}}_{\boldsymbol{\omega}}$ and a translational flow $\dot{\mathbf{x}}_\mathbf{v}$. Thus, (2.3) can be re-written as

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}_\mathbf{v} + \dot{\mathbf{x}}_{\boldsymbol{\omega}} = \frac{1}{Z} A(\mathbf{x})\mathbf{v} + B(\mathbf{x})\boldsymbol{\omega} \tag{2.4}$$

, where $\dot{\mathbf{x}}_\mathbf{v}$ depends on the depth Z of the observed point but $\dot{\mathbf{x}}_{\boldsymbol{\omega}}$ does not.

It should be noted that, although the model (2.3) could be applied to an arbitrary perspective camera, when using a standard lens (i.e. non wide-angle or fish-eye), not every world-point can be projected to the image since some of these will lay outside the camera frustum. As a consequence, x, y values cannot hold arbitrary values. In fact, with a field-of-view of less than $90°$, from (2.2) it results that $x, y \leq 1$. Thus, a first-order approximation can be used for $B(\mathbf{x})$:

$$B(\mathbf{x}) \simeq \begin{pmatrix} 0 & 1 & y \\ 1 & 0 & -x \end{pmatrix} \tag{2.5}$$

. This simplification leads to straightforward relation between a variation in ω_x, ω_y (roll and pitch motion) to a variation in $\dot{y}$ and $\dot{x}$, respectively.

2.2 Going With the Flow: Bearing-Only Navigation

As previously defined, the notion of optical-flow is conceived as the perceived motion of image elements, for a camera moving with given instantaneous linear and angular velocities during a small enough time-interval. In the context of the path-following problem, this equation can also be used to describe the perceived difference between two views of the environment (the current and reference pose over the path), related by a translation and rotation expressed in $\mathbf{v}$ and $\boldsymbol{\omega}$. While (2.1) can only be employed under the "small-angle approximation" (where $R \sim I + [\boldsymbol{\omega}]_\times$), for the same reasons by which (2.5) was proposed, this use of (2.1) remains valid in the context of the path-following problem with a standard perspective camera.

Thus, under an appearance-based approach, the path-following problem can be formulated as finding the appropriate controls for the robot to be driven from its current pose towards a reference pose in the path, while letting $\mathbf{v}$ and $\boldsymbol{\omega}$ converge to zero, based on the optical-flow $\dot{\mathbf{x}}$. As both the world structure and relative transformation are unknown, the actual optical-flow is estimated by the position difference of a pair $\mathbf{x}, \mathbf{x}'$ of corresponding points in both views, which are assumed to be the projections of the same world-point (the landmark). In other words, optical-flow is estimated as:

$$\dot{\mathbf{x}} \sim \mathbf{x}' - \mathbf{x} \tag{2.6}$$

. The process of establishing this correspondence is by means of extracting and matching salient image-features between the current image and that of the reference.

However, by observing (2.4) it becomes evident that from $\dot{\mathbf{x}}$ alone it is not possible to individually recover $\mathbf{v}$, $\boldsymbol{\omega}$ or Z without further information. In the absence of a sensor such as a stereo camera providing range and when not using other more complex methods such as SfM or full SLAM-based approaches, one solution is to effectively work around the inherent limitation of too many unknowns by only observing the total flow $\dot{\mathbf{x}}$, without ever explicitly distinguishing $\dot{\mathbf{x}}_{\mathbf{v}}$ from $\dot{\mathbf{x}}_{\mathbf{w}}$.

2.2.1 Ground-robot Case

To illustrate the approach, let us assume a robot positioned along a previously learned path (Figure 2.2 on page 38), slightly off its expected position. In this case, the reference pose over the path is assumed to be always known. Finding this reference is a task handled by a localization module, which will be later described in Section 3.4. Furthermore, the reference will always be a pose for which longitudinal error is negligible. Thus, the relative pose with respect to the reference, in the two-dimensional ground-robot case, only involves a lateral error v_x and an orientation difference ω_y (the relative yaw angle). Finally, let us assumed that a given landmark $\mathbf{X}_c$ can be detected from both the reference pose, related to the viewpoint of camera C, and the current robot's pose C'. The change in coordinates from C to C' follows (2.1). If, at this point, the horizontal component $\dot{x}^{(x)}$ of the projected optical-flow is considered, its value will be the result of both the rotational flow $\dot{x}_\omega^{(x)}$ induced by ω_y and the translational flow $\dot{x}_v^{(x)}$ induced by v_x. In fact, based on (2.3) and (2.5), in this scenario it results that

$$\dot{x} = \underbrace{\frac{v_x}{Z}}_{\dot{x}_v^{(x)}} + \underbrace{\omega_y}_{\dot{x}_\omega^{(x)}}$$

. Now, consider a simple proportional controller which drives $\dot{x} \to 0$ by

$$\begin{array}{rcl} w_t & \leftarrow & -k\dot{x} \\ z_t & \leftarrow & c \end{array} \tag{2.7}$$

, where z_t, w_t are the (forward) linear and angular velocities of the robot at time t, and k, c are constants. As a result, while the robot drives forward at constant speed it will turn inversely with the magnitude of the horizontal image-flow $\dot{x}$.

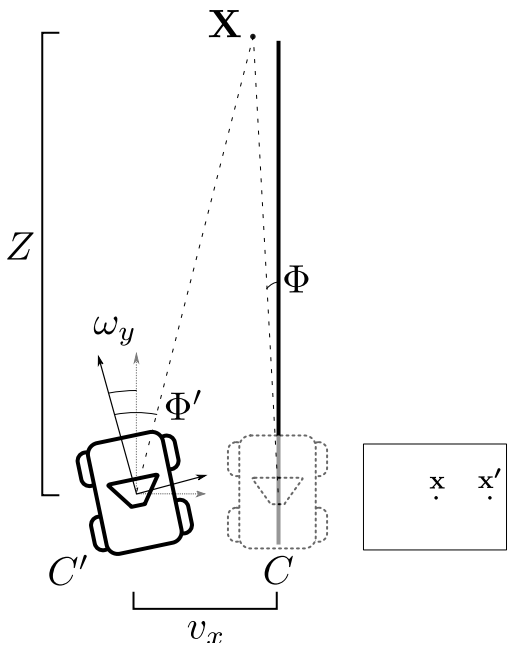


Figure 2.2: Ground-robot path-following problem in relation to bearing-only control. The robot is at pose C', displaced by lateral translation v_x and with an orientation difference ω_y w.r.t. a reference pose C' in the path to be followed. The robot observes a landmark X at distance Z, for which the relative bearing angle in both the current and reference views is Φ' and Φ respectively. From the projections $\mathbf{x}, \dot{\mathbf{x}}$ of X from both poses, optical-flow can be estimated to and used as input to a bearing-only controller.

2.2.2 Analysis of the Bearing-only Controller

The effect of this controller can be analyzed when considering different initial scenarios and those encountered during the convergence to the path (Figure 2.3 on page 39).

If $v_x = 0$ and $\omega_y \neq 0$, the robot is at the expected position but differs in orientation (case A in the figure), then the bearing-only controller will set an angular velocity of opposite sign, thus turning in the direction which drives ω_y to zero. On the other hand, if the robot starts with $v_x \neq 0$ and $\omega_y = 0$ (it is laterally displaced but with same orientation, case B in the figure), it will turn in the direction of the reference (e.g. if the robot is to the left of the reference it will turn right, since $v_x > 0$ and thus $w_t < 0$). By doing so, an orientation error is introduced w.r.t. the reference (i.e. ω_y will be non-zero, case C in the figure), thus increasing the magnitude of the rotational component $\dot{x}_\omega$, but now with opposite sign to v_x. As a result, during the rotation, the total flow $\dot{x}$ will be gradually reduced to zero (independently of the sign of v_x) as the value of $\dot{x}_\omega$ begins to compensate the value of $\dot{x}_v$. Now, as the robot moves forward, since the robot points towards the path, lateral error v_x is reduced and with it, the translational flow component $\dot{x}_v$ (case D in the figure). Now, a behavior opposite to the previously described is generated. If the magnitude of $\dot{x}_v$ becomes smaller, since up to this point $\dot{x}_v = \dot{x}_\omega$, the magnitude of $\dot{x}$ will be larger. Thus, in this case, the robot will turn to the direction opposite of the path until the balance is again restored (case E in the figure). In other words, while the robot keeps advancing towards the path and continuing the reduction of the lateral pose error v_x, the orientation error will also be gradually reduced reaching the point where $v_x = 0$ and $\omega_y = 0$. It should be noted that z_t is set to be relatively small compared to w_t so that during the convergence of $\dot{x}$ to zero it can be assumed that v_x will not change (as a result of approaching the path during forward motion).

2.2.3 Angle-based Formulation

In order to work with more familiar quantities when tuning the aforementioned controller, an equivalent angle-based formulation can be used. Instead of working with $\dot{x}$ in Cartesian coordinates over the imaging

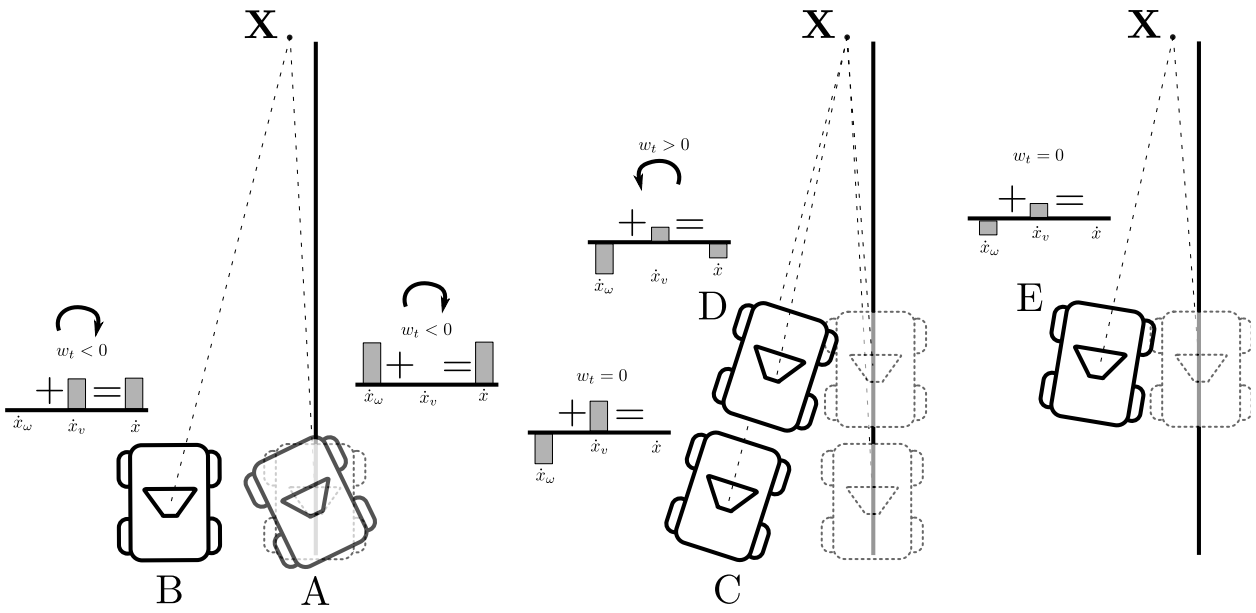


Figure 2.3: Analysis of the bearing-only controller. In (A), the robot is positioned over the path but presents an orientation difference. The perceived rotational flow results in a clockwise turn using the bearing-only controller. In (B), the robot starts both with a lateral and orientation offset. Here, both rotational and translation flow are perceived. As the robot turns towards the path, it will eventually stop turning when both flow contributions are equal, as in case (C). After advancing in the current direction, in case (D) the balance of both flows contributions is broken with the effect of now requiring a turn to the other direction, until balance is again restored as in case (E). This behavior continues until the robot reaches the path, with zero offsets w.r.t. the moving reference.

sensor, it is possible to work with the derivative of *bearing* angles to the corresponding world-point in each view. In the two-dimensional case, the bearing to a point $\mathbf{X_c}$ is equivalent to the *azimuth* angle Φ, which is formed between the optical ray (Z axis) and the projection of $\mathbf{X_c}$ to the XZ plane:

$$\Phi = \frac{\pi}{2} - \text{atan}(\frac{1}{x})$$

. In three dimensions, bearing is decomposed into the azimuth Φ and elevation Θ angles, the latter equivalently defined for the ZY plane:

$$\Theta = \frac{\pi}{2} - \text{atan}(-\frac{1}{y})$$

. Azimuth and elevation derivatives $\dot{\Theta}, \dot{\Phi}$ can be directly estimated from the difference of Φ and Θ, for a pair of feature correspondences in the image as

$$\begin{aligned} \dot{\Theta} = &\quad \text{atan}(\frac{1}{x'}) - \text{atan}(\frac{1}{x}) \\ \dot{\Phi} = &\quad \text{atan}(-\frac{1}{y'}) - \text{atan}(-\frac{1}{y}) \end{aligned}$$

, instead of working with pixel quantities as in (2.6).

Re-writing (2.7) in terms of the azimuth angle results then in:

$$\begin{aligned} w_t &\leftarrow -k\dot{\Phi} \\ z_t &\leftarrow c \end{aligned} \tag{2.8}$$

where k now relates a difference in bearing angles to an angular velocity.

The aforementioned approach is usually known as *bearing-only navigation* (also known as *qualitative navigation* in some works) since it deals with the observation of the bearings to landmarks and their differences between the current and reference pose, in order to guide the robot towards the path.

2.2.4 Aerial-robot Case

So far, for the sake of simplicity, the case of a ground-robot driving over a mostly even terrain was assumed. In this scenario, roll and pitch motions of the camera were deemed negligible which led to a simple solution based on directly using the horizontal image-flow component as the input to the path-following controller. Also, motion in the vertical axis resulting in non-zero v_y was also disregarded. In this section, the general case of motion in three-dimensional space, such as with an aerial robot which is the focus of this thesis, will be considered.

Looking back at (2.3) and (2.5), if $\omega_z = 0$, it results that $\dot{x}$ only depends on v_x and ω_y and $\dot{y}$ depends on v_y and ω_x[1]:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \frac{1}{Z} v_x + \omega_y \\ \frac{1}{Z} v_y + \omega_x \end{pmatrix} \tag{2.9}$$

. On the other hand, if $\omega_z \neq 0$, this is no longer the case. Furthermore, the effect of ω_z on $\dot{x}$ and $\dot{y}$ depends on x and y(see Figure 2.4 on page 40 for a depiction on different flow fields generated by the various individual motions andFigure 2.5 on page 41 for fields generated by more complex motions). However, thanks to the separability of the translational and rotational flow contributions, if knowledge about $\boldsymbol{\omega}$ exists, it is possible to remove the effect of any rotational motion on the total flow by a process known as *flow derotation* (see 3.2.3). As a result, the relation in (2.9) can again be assumed.

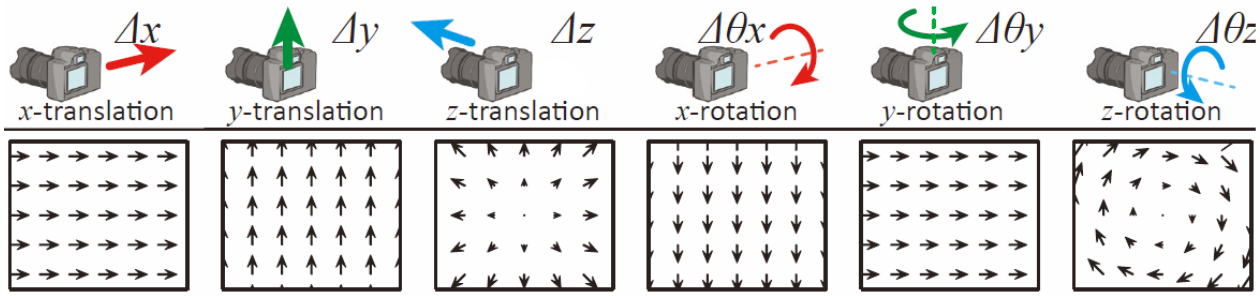


Figure 2.4: Optical-flow fields generated from individual camera motions. Lateral and vertical translations generate similar optical-flow fields to pitch and yaw rotations. However, a longitudinal translation generates a field of converging lines and a roll motion generates a radial flow field.

From (2.9) it can also be seen that the motion in the vertical axis is symmetric to that in the lateral axis. Thus, it would be natural to extend the bearing-only controller presented in (2.8), which reduces lateral pose error by controlling the yaw angle of the robot, by also controlling the pitch angle for the purpose of reducing vertical pose error. However, this is only possible for an aerial robot for which altitude is controlled by pitch changes, which is true for a fixed-wing aircraft but not for a multi-rotor style vehicle, which is the focus of this thesis. Moreover, in the latter case, the aircraft will change pitch as part of the forward velocity control. Thus, for a multi-rotor, pitch needs to be also compensated by derotation. In consequence, ω_x can also be assumed to be zero, resulting in the following straightforward relation:

$$\dot{y} = \frac{1}{Z} v_y$$

. Thus, a possible extension to (2.8) for the three-dimensional case, in which roll and pitch are compensated through flow derotation, is the following:

$$\begin{array}{rcl} w_t & \leftarrow & -k_1 \dot{\Phi} \\ y_t & \leftarrow & -k_2 \dot{\Theta} \\ z_t & \leftarrow & c \end{array} \tag{2.10}$$

where y_t is the vertical velocity of the vehicle and k_1, k_2 are constants.

[1]Remember that, since continuous localization is always assumed, $v_z = 0$.

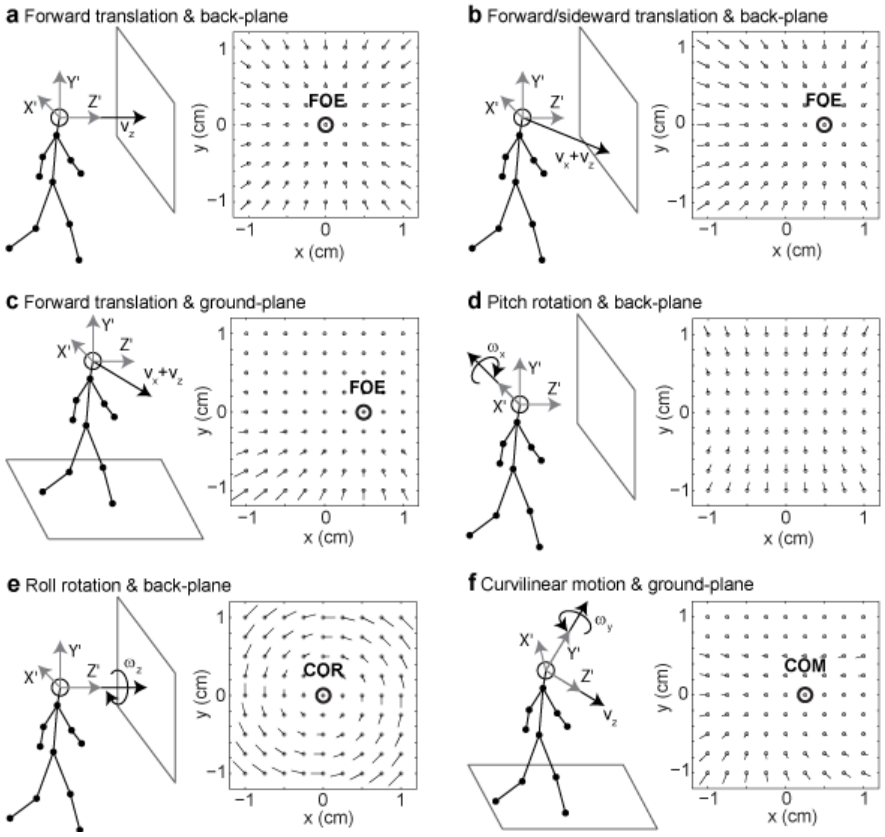


Figure 2.5: Complex optical-flow fields generated from the combination of various simultaneous motions. Note that these examples assume a wide-field of view and therefore approximation (2.5) does not apply, as it can be seen in case **d** where a pitch rotation generates a non-uniform field.

While an aerial robot is not only able to change height, it is also able to move laterally. In fact, an aerial robot (multi-copter type) is holonomic, meaning that it is possible to control all of its degrees of freedom. Thus, it is also worth to analyze the possibility of employing lateral control for holonomic robots, which applies to other platforms such as omnidirectional-wheel based robots or those based on articulated legs.

In principle, with monocular vision, the inherent ambiguity presented in 2.2 does not allow distinguishing lateral translation error (v_x) for a yaw difference (ω_y). However, if the yaw angle w.r.t. the world frame is known, as in the case of a robot with a magnetometer, after flow derotation around ω_y it is possible to obtain a controller similar to (2.10) but for lateral velocity. In this case, the heading of the vehicle can be directly controlled without visual input (based on the known absolute yaw angle) and convergence to the path is directly achieved by the lateral velocity controller:

$$\begin{array}{rcl} w_t & \leftarrow & -k_1\omega_y \\ x_t & \leftarrow & -k_1\dot{\Phi} \\ y_t & \leftarrow & -k_2\dot{\Theta} \\ z_t & \leftarrow & c \end{array} \tag{2.11}$$

.

Further details on the aforementioned controllers and their performance in practice are presented in later chapters of this thesis.

2.3 Navigation with Multiple Landmarks

While the proposed control laws are still theoretically applicable with a single landmark, in practice it would be not feasible to work with only one correspondence between views, as sensor noise and occlusions would render the approach unusable. On the other hand, since the translational flow component depends not only on the corresponding linear and angular velocities but also on the distance to the landmark Z (see (2.9)), the relative bearings of two landmarks at different distances would produce a different output for the controllers presented in 2.2. Thus, there is a question on how to extend the bearing-only strategy to multiple landmarks while at the same time dealing with bad correspondences.

One commonly used strategy is to simply average all bearing differences obtained from the set $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$ of landmark correspondences obtaining the mean values $\bar{\dot{\Theta}}, \bar{\dot{\Phi}}$ to be used as the input to the bearing-only controller, as a replacement for the individual $\dot{\Theta}_i, \dot{\Phi}_i$. As a consequence, the different responses that would be obtained from individual landmarks, each with its own Z_i, are also averaged. However, in terms of handling bad correspondences, the mean may not be the more robust choice, since it assigns equal weight to all samples being averaged. Thus, an alternative strategy is desired.

2.3.1 Mode-based Control

As an alternative to simply averaging all relative bearings, it is possible to employ the modes of the sets $\{\dot{\Theta}_i\}$ and $\{\dot{\Phi}_i\}$ instead of the averages $\bar{\dot{\Theta}}$ and $\bar{\dot{\Phi}}$. To obtain the modes, the two histograms for azimuth and elevation can be computed, where the mode will correspond to the value associated to the highest voted bin.

Using the mode as the input to the controllers equates to selecting the maximal set of landmarks that appear to be at a similar distance with respect to the camera, since for a fixed $\mathbf{v}$ and $\boldsymbol{\omega}$, it follows from (2.9) that the relative bearings for landmarks will differ only according to Z_i. Furthermore, in the presence of bad correspondences, it is highly unlikely that the selected bin corresponds to all badly matched points between views, unless the ratio of good to bad correspondences is really low. As a result, the mode can be quite robust to bad correspondences, which generally appear as low frequency values spread over the range of possible relative bearings.

This generalization of the bearing-only controller to multiple landmarks by means of the computation of the modes from the corresponding histograms can be related to the problem of finding a two-dimensional translation given by t_x, t_y in image-space and which relates the correspondences by

$$\mathbf{x_i} - \mathbf{x'_i} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

. A commonly used solution for this problem in the presence of outliers is known as 2-point RANSAC, which iteratively computes t_x, t_y by randomly selecting pairs of correspondences and then choosing the solution with higher consensus over all correspondences. In comparison to 2-point RANSAC, the histogram-voting approach only considers a fixed number of models (one for each bin) and directly chooses the one with the highest consensus, without iteration. Thus, histogram-voting presents itself as a simple and efficient alternative to the more expensive RANSAC approach, and which allows obtaining similar results[44].

2.3.2 Analysis of Mode-based Control

It could be argued that, if the input to the controller is given by the highest voted bin in the corresponding histogram, if the distribution of landmark depths Z_i is relatively homogeneous, this would produce a relatively flat histogram with no easily discernible mode. In this scenario, the value of the mode will be erratic, which would have an undesirable effect on the control outputs. However, there is an aspect to be considered that limits this problem in practice.

Since the camera has a fixed resolution with which it acquires the images, this means that image-flow quantities obtained by (2.6) will not allow perceiving the translational flow component produced by landmarks further away than some maximum depth $Z_{\max}$. As a consequence, for a given $\mathbf{v}$ and given image resolution, all landmarks with $Z > Z_{max}$ will correspond to the same bin of the histogram. In other words, many landmarks will appear to be at infinity and thus be independent to $\mathbf{v}$ (since the translational component in (2.9) vanishes). Therefore, many more landmarks will fall in the same bin corresponding to large Z and thus produce a distinguishable peak.

On the other hand, this same reason implies that it is likely that the mode of relative bearings will be obtained from very-distant landmarks, for which translational flow is close to zero, even in the presence of non-zero $\mathbf{v}$. Thus, in areas where these landmarks are relatively far, convergence towards the path (i.e. the reduction of v_x and v_y to zero) may be quite slow.

This can be considered an intrinsic limitation of the very simple formulation of the bearing-only controller, whose aim is to be able to function with mostly the information obtained from the camera and at the same be very simple but robust to outliers. It should be pointed, though, that lateral errors will in practice never be so high that this would represent a problem for the correct path-following behavior. In other words, the problem to be solved is mostly path-tracking and not so much on path re-entry under very large translational errors.

Capítulo 2

Principios de navegación basada en rumbos (resumen)

En este capítulo se detallan los principales aspectos a tener en cuenta al momento de considerar un método de navegación basado en los rumbos a referencias visuales del entorno.

El concepto de navegación basada en rumbos esta íntegramente relacionado con el concepto de flujo óptico, que se define como el movimiento relativo a una cámara de los puntos que ésta observa cuando la misma realiza un movimiento de traslación y rotación. Si se considera dicho movimiento relativo en coordenadas de la imagen de dicha cámara, se puede llegar a una fórmula relativamente simple, en la cual se evidencia que el movimiento de un punto en la imagen es producto de dos contribuciones separables: la asociada a la traslación y la asociada a la rotación. El primer movimiento, tiene un efecto que depende no solo de dicha traslación sino también de la distancia a la que se encuentren los puntos en la escena. Por otro lado, la contribución asociada a la rotación solo depende de la magnitud de dicho movimiento. Esta propiedad es la que permite considerar una navegación basada en medir dichos movimientos en la imagen, para relacionarlos con movimientos individuales de la cámara.

Más adelante, en este capítulo se proponen diversas leyes de control para un robot dotado con una cámara cuyo objetivo es repetir un camino previamente aprendido. Si se asume que en todo momento se tiene una referencia sobre el camino (a la cual solo existe una traslación lateral o vertical y una rotación arbitraria), la navegación basada en rumbos propone el control del ángulo de giro del robot en base a la diferencia del ángulo de acimut de pares de puntos correspondientes en ambas imágenes. En este caso, se asume siempre velocidad lineal constante hacia adelante. Se puede ver que con esta simple ley de control, se logra la convergencia de un robot a un camino aprendido. Para el caso de un robot aéreo, es necesario considerar tanto los posibles movimientos de rotación que este realiza como la capacidad de cambiar de altura y de moverse lateralmente, en contraste con un robot terrestre. En este sentido, se presentan también algunas variantes a la ley de control anterior.

Chapter 3

Visual Teach & Repeat Navigation Method

Based on the principles of bearing-only navigation outlined in the Chapter 2, an appearance-based approach to TnR navigation is presented. In this chapter, the proposed method is detailed. In particular, the issues arising from the use of aerial vehicles and the corresponding solutions are presented. In any case, since the method is applicable to both ground and aerial robots, hereafter no particular platform should be assumed unless stated otherwise.

3.1 Method Overview

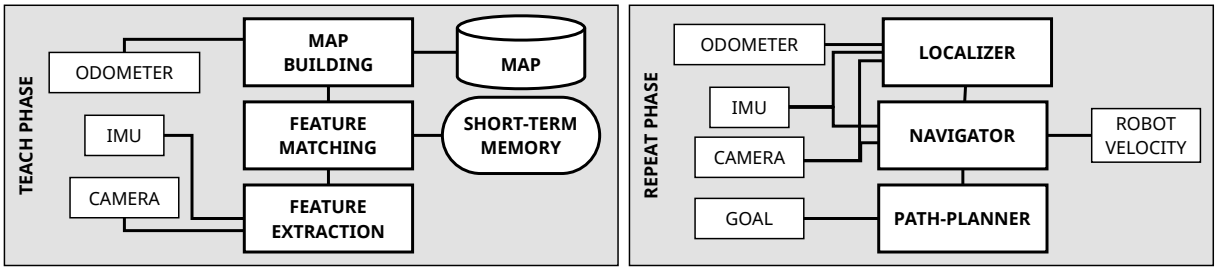


Figure 3.1: Conceptual diagram of the VTnR method, for the teach and repeat phases.

The proposed navigation method follows the teach & repeat scheme, where two distinct phases are distinguished (see Figure 3.1 on page 47).

The goal of *teach* phase is to build an appearance-based representation of the path traversed by the robot (i.e. a map). During this phase, the robot is moved semi-autonomously: the user only controls the robot's heading (and altitude, in the case of aerial robots) while moving at constant forward speed. While the robot moves, the map is built continuously. Since a globally-consistent representation of the environment is not required for navigation under the TnR scheme, the purpose of this map is to allow for efficient along-route localization and to obtain the required control inputs for the robot to later repeat the path. The proposed map representation thus combines topological information (in the form of a directed graph) and appearance cues, which describe the environment from the robot's point of view by means of salient visual features.

During the *repeat* phase, a localization module continuously estimates the robot's pose over the topological map. Once a goal location over the map is received from the user, a topological path is built from

the robot's current estimated pose to said goal (using Dijkstra's Shortest Path algorithm) and autonomous navigation can then start. By matching visual features currently observed to those expected to be seen at the estimated robot pose in the map (detected during the *teach* phase), inputs to a simple bearing-only controller can be computed, which allow the robot to autonomously reduce any lateral/vertical offset and heading error with respect to the desired path.

The main sensor employed by the approach is a single perspective camera which allows to capture images and, after applying image-processing algorithms, to obtain visual information of the environment in the form of point-based features. A second required source of information is a short-term estimation of the robot's motion. For ground-robots, encoders attached to the internal motors can be used for this purpose. For aerial robots, a downward looking optical-flow sensor can estimate both the horizontal and vertical motion. Finally, for the case of an aerial robot or a ground-robot moving over highly uneven terrain, information regarding the orientation of the camera (roll and pitch angles) is used to compensate the effect of rotational motion on the visual information. Finally, the navigation method controls the robot by issuing linear (horizontal and vertical) and angular (only for yaw) velocity commands.

3.2 Feature Processing

The proposed navigation method strongly depends on the detection of salient image features and the establishment of correspondences between those currently observed with those stored in the map. Thus, feature processing is a core component of the proposed solution and suitable algorithms for visual feature extraction, description and subsequent matching, i.e. to establish correspondences, are needed.

The first determining aspect in selecting the appropriate algorithms is the robustness of the extractor/descriptor algorithm pair, in terms of its ability to successfully detect and uniquely describe the same perceived elements of the environment from two distinct camera poses. Illumination changes, image noise and, in general, visual differences arising from changes in the environment such as new/missing items or the presence of moving obstacles (e.g. persons), all represent challenging situations which needs to be handled successfully by the feature processing algorithms. Secondly, efficiency becomes paramount when features need to be extracted, described and matched against the map for each camera image, in order to obtain the input data to the bearing-only controller. These operations need to satisfy real-time operation constraints of the complete navigation system, as imposed by the robot's dynamics.

3.2.1 Feature Extraction and Description Algorithm

To satisfy the aforementioned constraints, two extractor-descriptor pairs were chosen for the proposed VTnR method, based on a previous analysis[45, 46] where state-of-the-art algorithms were compared in terms of their robustness and efficiency. For feature description, in both cases the Binary Robust Independent Elementary Features (BRIEF)[7] algorithm was chosen. For feature extraction, the STAR algorithm, which is derived from the Center Surround Extremas (CenSurE)[47] was employed for early experiments with the proposed VTnR method. However, in later experiments, the chosen detector was the Good Features To Track (GFTT) algorithm, by Shi-Thomasi[20]. In contrast to STAR, GFTT is targeted towards real-time feature tracking in successive frames and is thus more efficient than STAR, which is aimed towards robustness. However, it practice GFTT was found to be comparable to STAR in terms of robustness. Also, GFTT can be more easily tuned than STAR, since the latter requires setting an absolute threshold which can greatly depend on the scene.

3.2.2 Rotational Invariance

One particular aspect to consider in the context of feature-matching of images captured by an aerial robot (or a ground robot moving over very uneven terrain) is the fact that the robot, and thus the camera, will be likely

to perform rotational motion around the optical axis (roll). While the robot will also perform rotations on the yaw and pitch axis, in the image this equates to a translational motion of pixels. On the other hand, rolling motion produces a very particular radial pattern in the image (recall Figure 2.5 on page 41). For this reason, while any feature-extraction/description algorithm is able to deal with camera pitch and yaw rotations, this is not generally the case for camera roll, a property which is defined as the "rotational invariance".

The BRIEF algorithm, in fact, does not ensure rotational invariance in the computed descriptors. While many other similar variants exist, ensuring not only rotational but also scaling invariance, it was found that BRIEF provided the best balance between efficiency and robustness. In order to take advantage of the BRIEF algorithm but to also be able to match features in the presence of camera roll, the proposed navigation method takes advantage of the knowledge of the robot's orientation (as obtained from inertial sensors present in the robot) to *derotate* the descriptor.

The process is based on the fact that BRIEF computes the descriptor by a performing many pairwise pixel differences inside the image-patch surrounding the detected feature. Thus, since under a certain roll the image-patch will rotate accordingly, if the pairs are rotated by the camera's roll angle prior to pairwise difference computation, the result is a descriptor which is able to recognize the same patch irregardless of this rotation. This method allows to overcome the performance penalty which is to be paid when expecting the description algorithm to recognize the image-patch orientation by exploiting prior knowledge obtained from others sensors and, at the same time, obtain the same performance of the original BRIEF approach. An experimental analysis of this strategy is presented in Section 5.4.

3.2.3 Feature Derotation

As it was mentioned in Chapter 2, when employing the bearing-only navigation controller in the presence of camera roll, the induced rotational flow needs to be first compensated by a process known as *flow derotation*. To do so, knowledge about the relative orientation between the two camera poses is required, such as with a robot for which absolute orientation is known. A simple approach is to directly derotate the projections of the corresponding world points in both views before estimating the optical-flow from the position difference of these projections. The projection $\mathbf{x} \in \mathbb{R}^2$ (in-homogeneous image coordinates) to a camera with orientation given by $R \in \mathbb{R}^{3\times3}$ is derotated by

$$\pi(R^{-1} \begin{bmatrix} \mathbf{x}^t \\ 1 \end{bmatrix})$$

where $\pi(\cdot)$ is the image-projection operator, defined as

$$\pi\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \frac{1}{z}\begin{bmatrix} x \\ y \end{bmatrix}$$

.

In the proposed VTnR approach, image-projections are derotated around both the roll and the pitch axis, when knowledge about the absolute camera orientation is available (aerial robot or ground robot with on-board IMU). In this case, the orientation matrix is built as:

$$R = R_z(\omega_z)R_y(0)R_x(\omega_x)$$

, where ω_x, ω_z in this case denotes the camera's absolute pitch and roll angles. In this way, "derotated" image-projections are obtained and, thus, when the optical-flow is computed the rotational component for these axis will be zero.

3.3 Map Building

As previously mentioned, the proposed map representation combines topological information, in the form of a directed graph, and appearance cues based on salient image-features. In the following sections, the map building process in terms of both of these aspects is described. A pseudo-code describing the complete algorithm is presented in listing 1.

3.3.1 Topological Information

During the teach phase, while the robot is guided to describe a path to be learned, topological information is embedded in the map by periodically adding *nodes* (after a certain distance has been traversed) which are connected to the previous ones by means of a *segment*s. While for a single path, the map reduces to a linear pose graph, this representation allows describing multiple interconnected paths (this generalization is considered in Section 3.3.3).

Segments represent the connectivity between two nodes, i.e. two positions in the world, and have a length and relative orientation w.r.t. their source node, computed by the cumulative motion of the robot between the pair of nodes. This motion is obtained by simple dead-reckoning, i.e. the integration of the robot's velocity estimation coming from internal sensors (such as motor encoders or optical-flow based estimation). No absolute pose information is stored in the map, since this is not required for any portion of the method.

Due to the relative nature of this representation, if the map is to be visualized, any node can be chosen as the reference coordinate frame from which the poses of the other nodes can be obtained by chaining the relative transform information present in each segment (length and orientation). Compared to the true path followed during the teach phase, the obtained map is: a) a simplified version of the robot's true path (since motion between nodes is assumed to be linear and turns are only recorded in segments), b) not globally consistent (the accumulation of relative motions is only consistent up to a certain distance). However, as the map's structure only serves to describe the topology of environment, these aspects have no impact on the navigation method itself.

3.3.2 Appearance Cues

Besides topological information, appearance cues useful for localization and navigation are also embedded in the map. During the teach phase, salient image-features are extracted from images and continuously tracked in subsequent frames while still visible. These features are associated to the segment currently being built, effectively describing the pose of the robot w.r.t. the path by visual information alone. In other words, no geometric structure is estimated in this process. This representation of visual information allows, during the repeat phase, to localize the robot along a given segment by means of feature-matching and, from these matches, the estimation of the optical-flow which is used as input to a bearing-only controller for the purpose of path-following (as described in Chapter 2).

The process of feature-tracking involves the use of a *short-term memory* (STM). Initially, the STM is filled with all features from the first image. Then, features detected in subsequent images are matched to those in the STM by means of nearest-neighbor search in descriptor-space. If a match for a feature in the STM is found, it is considered "tracked" and thus remains in the STM. Features in the STM that are not corresponded for some time $t_{\text{MAX\_NOT\_SEEN}}$ to others extracted in successive images are eventually removed, i.e. forgotten. New features detected in the current image which do not match with any in the STM are added (i.e. remembered) to this memory, to enable further tracking in following images.

However, for tracking over several frames to be successful, the descriptor of a feature in the STM needs to be kept up-to-date w.r.t. the current appearance of the matching feature in the latest image. Otherwise, as the appearance of the feature in following images changes, it would eventually not match anymore to

its corresponding feature in the STM. Thus, instead of individual features, the map holds *visual landmarks*, which consist of a list of *views* of a salient element in the environment. Each view then corresponds to a feature, represented mainly by its pixel-position and descriptor. As with nodes, new views are added after the robot has traveled a certain distance w.r.t. the last view. However, views are added more frequently than nodes.

The purpose of the STM is to overcome the problem of image noise and temporary occlusions, which forbid detection of a given feature but only for some time. Depending on the value of $t_{\text{MAX\_NOT\_SEEN}}$, the problem of temporary tracking loss can be handled robustly. However, for large values of this parameter, the number of features in the STM will rapidly grows, increasing the cost of nearest-neighbor search and possibly producing false matches.

To store the aforementioned visual information in the map, landmarks are associated to segments by means of landmark *references* (see Figure 3.2 on page 51). For each segment, a reference is added to a landmark if it was still present in the STM by when the segment is ended (i.e. seen at least once before it was forgotten). A reference not only points to a landmark but also contains information about the distance traveled by the robot, the *offset*, w.r.t. the beginning of the segment from which the landmark was first seen. This indexing by relative distance is also used for views when they are added to a given landmark.

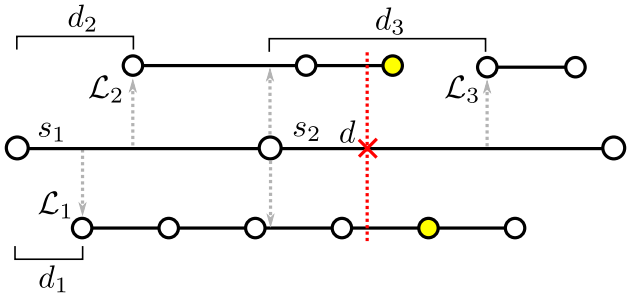


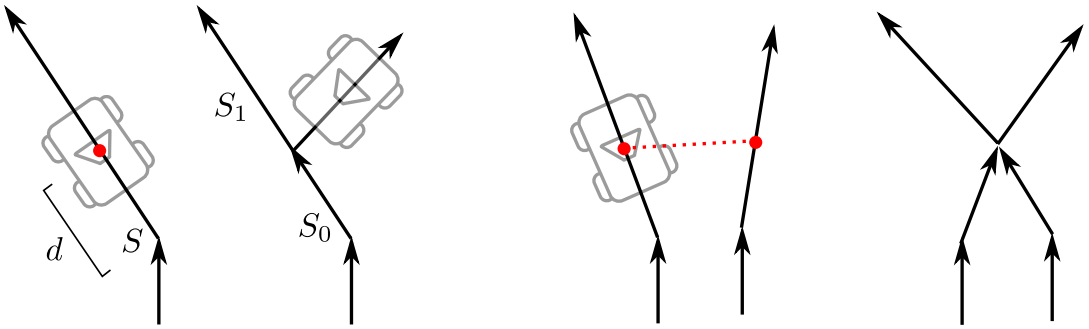
Figure 3.2: Relation between segments landmarks and views: an example is presented where three different landmarks $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ are visible from two contiguous segments s_1, s_2. Segment s_1 references landmarks $\mathcal{L}_1, \mathcal{L}_2$ and segment s_2 references all three landmarks. Landmark references are depicted as dashed arrows. The distances d_1, d_2, d_3 correspond to the landmark offsets of $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ respectively. The views expected to be seen at distance d of s_2 are marked as filled yellow circles.

This decoupling of views and segments allows to independently choose the thresholds used to decide when a new node or a new view should be added. While the granularity of nodes does not have an impact on localization or navigation (the map is only to describe the topology and for high-level path-planning), this is not the case for landmark views. Landmarks with many views would potentially allow to obtain a more precise localization and navigation, but at the expense of a larger map. Moreover, in contrast to a typical snapshot or key-frame based map, since landmarks are tracked independently of each other and with a user-defined granularity, the search-space for localization becomes effectively continuous in practice.

The organization of tracked features by means of views and landmark references, allows to retrieve during the repeat phase the set of feature that are expected to be seen at a given pose in the map, i.e. those seen during the teach phase at that location. Given a particular segment s_i and a distance d over s_i, the task is to obtain the set of landmark views for that location. The process starts by first obtaining all landmarks that were visible at that location. To do so, for each landmark reference associated to s_i, views with the smallest distance d_v such that

$$l_{\text{offset}} + d \geq d_v$$

are filtered. The landmark offset allows to compare d, which is measured w.r.t. the start of s_i to d_v, which is measured w.r.t. the beginning of the segment from which the landmark was initially tracked. The resulting views are thus the next-closest views at the given location over the segment. An example of view retrieval is presented in Figure 3.2 on page 51 (where the query pose is marked as a red line and the views that would be retrieved are represented in yellow).



(a) Addition of bifurcation **(b)** Addition of junction

Figure 3.3: Two scenarios of topological map augmentation. In the first case, the original path is extended by adding an alternate path, which leads to map a bifurcation point in graph. The second case corresponds to the situation where a previously mapped location is re-visited, leading to the addition of a junction point in the map.

3.3.3 Representing Multiple Connected Paths

While most TnR methods only allow teaching and repeating a single path from start to end, the purpose of a graph-based topological map is to allow describing a *network* of paths. When considering this generalization, two separate scenarios need to be distinguished, which differ according to whether an inbound or outbound edge is to be added to the graph.

The first scenario consists in the extension of the map by adding a bifurcation. In this case, an outbound connection from a node in the original map is added to represent an alternative path that can be followed. The second scenario corresponds to when the robot reaches a path already learned. To represent this situation, two inbound connections to a new node representing the junction are generated. In general, a map composed of bifurcations leads to a tree-shaped graph, while a map with junctions can lead to a graph with cycles.

To describe how the case of the addition of a bifurcation is handled in the proposed approach, let us assume that the robot is currently in the repeat phase, localized at a distance d along a certain segment S of the map (see Figure 3.3a on page 52). At this point, if the teach phase is started, an outbound path at this position is defined. As when entering the teach phase a new path is created by adding a new node representing the current robot's pose (line 1 of listing 1), in this scenario segment S needs to be first split at distance d into S_0 and S_1, of length d and $|S| - d$, respectively. Landmark references in S are distributed to S_0 and S_1 accordingly, while their offsets are also updated to account for this change.

For the second scenario of adding a junction (Figure 3.3b on page 52), the robot is again assumed to be in the repeat phase when it detects to be in already mapped location (the current view matches an already learned view). Then, a junction is added in the map, which describes the ability of reaching the old path from the current's robot pose . To do so, the two segments involved (that of the current and old path) are split at the point of the junction while sharing a common node. This new node will thus have two inbound and two outbound connections.

Since the second scenario opens up a new challenging problem, commonly known as *loop detection*, only the first case was experimentally evaluated in this thesis. However, the map representation and the chosen localization and navigation algorithms never assume a cycle-free map thus allowing for easily completing this generalization, if an appropriate loop detection mechanism is later introduced.

In effect, with both scenarios the classical TnR strategy is generalized by allowing to navigate the environment by only repeating the required portions of all learned paths in order to reach a given goal. Localization, path-planning and navigation over a graph-based map are described in the following sections.

3.3.4 Algorithm

When entering the teach phase, a new node is first added in order to describe the current robot pose (line 1). Otherwise, if the robot was previously in the repeat phase and localized over an existing map (corresponding to the scenario of the addition of a bifurcation), this new node will be the node at the splitting point of the currently occupied segment, as described in Section 3.3.3.

Algoritmo 1: Map Building during Teach Phase

Result: $M(N, S)$: map represented as a graph with set of vertices (nodes) N and edges (segments) S

1 $N \leftarrow N \cup \{n\}$ /* add new node to map */
2 $STM \leftarrow \emptyset$ /* short-term memory */
3 $T \leftarrow \emptyset$ /* landmarks tracked in current segment */
4 $d \leftarrow 0$ /* distance traveled from current segment start */
5 **Loop**
6 $F \leftarrow extract\_features()$ /* extract feature's from current image */
7 $d \leftarrow d + \Delta d$ /* add robot's linear displacement to distance traveled since last segment start */
8 **foreach** $l \in STM$ **do**
9 $f \leftarrow$ find_match(l,F)
10 **if** *matching landmark found* **then**
11 $l_{\text{last\_seen}} \leftarrow now$ /* reset landmark's last seen time */
12 $l_{\text{current\_view}} \leftarrow (f_{desc}, f_{pos}, L_{offset} + d)$ /* update the most recent view of l with current feature, at distance $L_{offset} + d$, relative to initial segment beggining */
/* save new view to landmark view list if traveled distance relative to last saved view exceeds threshold */
13 **if** $l_{views} = \emptyset$ *or dist*$(l_{current\_view}, l_{last\_saved\_view}) \geq \epsilon_v$ **then**
14 $l_{views} \leftarrow l_{views} + \{l_{current\_view}\}$
15 $F \leftarrow F - \{f\}$ /* remove the matched feature from the list of candidates */
16 **foreach** $f \in F$ **do**
17 $l_{\text{views}} \leftarrow \{(f_{desc}, f_{pos}, d)\}$ /* create a new landmark with a single view based on f, with offset d */
18 $STM \leftarrow STM \cup \{l\}$ /* start tracking new landmark */
/* maintain short-term memory */
19 **foreach** $l \in STM$ **do**
20 **if** $now - l_{last\_seen} > t_{MAX\_NOT\_SEEN}$ **then**
21 $T \leftarrow T \cup \{l\}$ /* store reference to landmark */
22 $STM \leftarrow STM - \{l\}$ /* stop tracking landmark */
/* check if new segment should be added */
23 **if** $d > \epsilon_S$ **then**
24 $N \leftarrow N \cup \{n'\}$ /* create target node for new segment */
25 $s \leftarrow \{n \mapsto n'\}$ /* create segment connecting nodes */
26 $s_{\text{angle,length,landmarks}} \leftarrow \text{angle}(n, n'), d, T$ /* set segment properties */
27 $S \leftarrow S \cup \{s\}$ /* add segment to map */
28 $d \leftarrow 0$ /* reset traveled distance along segment */
29 $n \leftarrow n'$ /* new node is now current node */
30 **foreach** $l \in STM$ **do**
31 $l_{offset} \leftarrow l_{offset} + d$
/* update landmark's offset */

Features are then continuously tracked while the robot moves (line 5) by means of the STM. To do so, for each image, features first extracted and then matched against the landmarks in the STM. If a match is found, the landmark's "last seen" stamp is updated. The last view of the landmark's view list is updated with the current feature image-position and descriptor, together with the distance used to index views. When the robot has traveled a certain distance w.r.t. the last view, a new slot is added to the list of views. All features in the image successfully matched to a landmark in the STM are removed from the list of extracted features and the remaining ones are added to the STM (line 16). Maintenance of the STM is then performed by removing landmarks "not seen" for some time. While doing so, a reference to the corresponding landmark is added to the set T of landmarks tracked for the current segment.

Finally, a check is performed to determine whether the current segment is to be finished depending on the distance traveled by the robot so far. If so, a new node representing the current robot's position and the corresponding segment are added to the map (connecting the previous current node to this new node). The length of the segment is determined by the integrated linear motion of the robot and its relative angle is obtained from the angle formed by the relative position of the pair of nodes. Also, all landmarks references stored in T for the current segment are added to the current segment. Before continuing with the following segment, all landmarks in the STM have their offsets updated to account for the addition of the new node.

3.4 Localization

The purpose of localization in the context of the path-following problem is to continuously find a pose in the map which can be used as a reference for navigation, as defined in Chapter 2. For the proposed VTnR method, a probabilistic approach based on the Monte Carlo Localization (MCL) method is used, which balances computational efficiency and localization robustness. Moreover, with MCL it is possible to globally (re-)localize which allows to initiate the repeat phase from arbitrary locations in the map and to recover from localization failures, a situation known as the *kidnapped robot problem*.

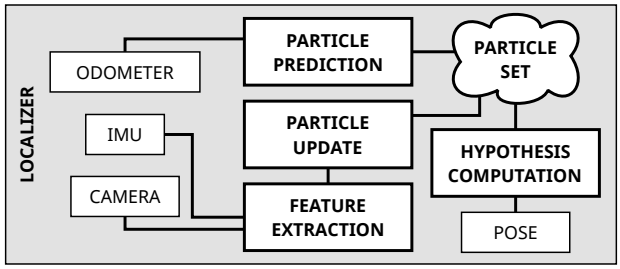


Figure 3.4: Conceptual diagram of the navigation module

In Figure 3.4 on page 54, a diagram of the modules composing the localization module are presented. The MCL-based localizer first extracts features in the current image (optionally compensating for pitch and roll, based on the output of the IMU), which are then used to test a set of localization hypothesis, called *particles*, by matching these against the features in the map. Then, dead-reckoning is used to predict the new set of localization hypothesis, to account for the robot motion. Finally, from the set of particles, a single-hypothesis is obtained, which is the output of the localization module.

3.4.1 Overview

Under a probabilistic formulation of the localization problem, the uncertainty in the robot's pose $\mathbf{x}_t$ at time t is given by the probability density function $p(\mathbf{x}_t)$, which is unknown. The Monte Carlo Localization approach estimates $p(\mathbf{x}_t)$ recursively using Bayesian filtering, under the Markov assumption (where the current state only depends on the previous one and not on the entire past history of states). A prediction

of $p(\mathbf{x}_t)$ is first obtained using a motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$, where $\mathbf{u}_t$ is the last motion command. Then, an *update step* is used to correct this prediction by means of a sensor model $p(\mathbf{z}_t|\mathbf{x}_t)$ which expresses the likelihood that the robot is at state $\mathbf{x}_t$ given the last sensor reading $\mathbf{z}_t$.

In contrast to other approaches such as Kalman Filtering, MCL is a non-parametric filter which does not make any assumption on the shape of $p(\mathbf{x}_t)$. With MCL, the probability density function is represented by means of a number of samples termed *particles*, which will be the localization hypothesis. Localization starts with an initial set of random particles $\{\mathbf{p}^i\}_{i=1\ldots M}$ uniformly distributed over the entire map. For each iteration of the method at time t, new particles $\mathbf{p}_t^i$ are first sampled from the motion model $p(\mathbf{x}_t|\mathbf{p}_{t-1}^i, \mathbf{u}_t)$ and then re-sampled by selecting each particle with a probability given by the likelihood $p(\mathbf{z}_t|\mathbf{p}_t^i)$.

3.4.2 State Definition

Given a topological map defined as in Section 3.3, the robot state $\mathbf{x}_t$ at time t will be defined as

$$\mathbf{x_t} = \begin{bmatrix} s_i & d \end{bmatrix} \tag{3.1}$$

, where s_i identifies a segment of the graph and d the distance overs_i. With a given pose $\mathbf{x}_t$, it is then possible to retrieve the set of expected landmark views at this location in the map, information which will be used to compute the sensor model for this pose, given the set of currently observed features. Moreover, the navigation module will also use these views to perform path-following, as will be explained later.

This simple state definition of (3.1), leads to a uni-dimensional search-space which can be sampled with a relatively small number of particles, in contrast to a full 6DoF state found in a typical metric approach.

3.4.3 Motion Model

The motion model allows to predict the new position of each particle, given last motion information $\mathbf{u}_t$, which is defined as the linear distance traveled Δd_t with respect to time $t-1$ (obtained by simple dead-reckoning). The proposed motion model, is recursively defined as

$$f\left(\begin{bmatrix} s_i & d \end{bmatrix}, \Delta d_t\right) = \begin{cases} f\left(\begin{bmatrix} s_j^* & 0 \end{bmatrix}, \Delta d_t - |s_i|\right) & d + \Delta d_t \geq |s_i| \\ \begin{bmatrix} s_i & d + \Delta d_t \end{bmatrix} & \text{otherwise} \end{cases} \tag{3.2}$$

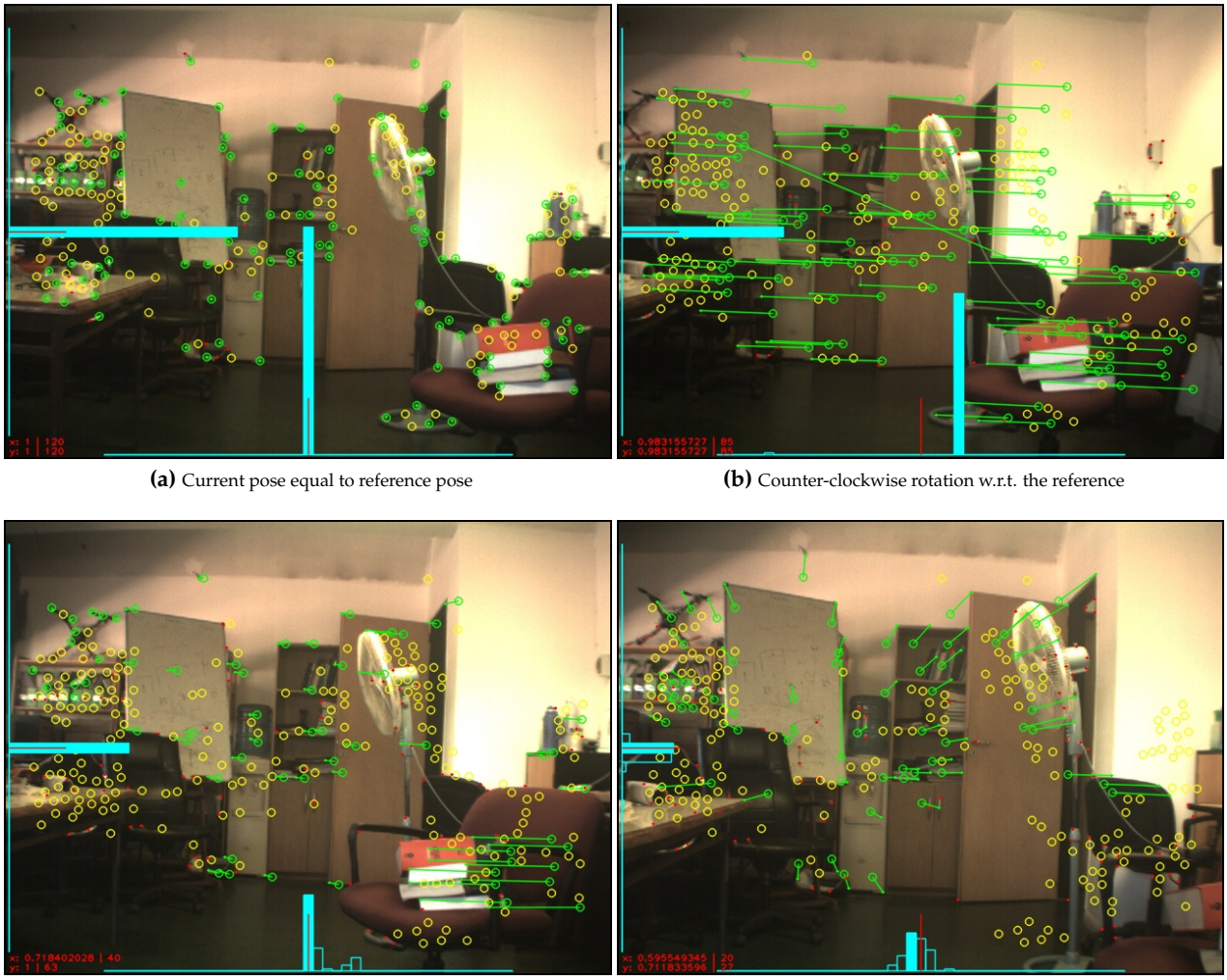
, where $|s_i|$ is the length of segment s_i and s_j^* is a random sample from the set of outbound edges stemming from the target node of s_i. In other words, in principle bifurcations are assumed to be equally likely to be followed during prediction, and thus the new position of a particle in the map is obtained by a random walk over the graph, up to a distance Δd_t.

In practice, the prediction step is applied using $\Delta d_t + \mu$ as the input to the motion model, where $\mu \sim \mathcal{N}(0, \sigma)$. This noise term is necessary to avoid premature convergence (i.e. a single high-probability particle chosen by MCL after various re-sampling steps) and to maintain the diversity of localization hypotheses, since Δd_t is not known precisely.

This noise term also contemplates the imprecision of this simple motion model, which does not explicitly account for the robot heading, which assumes that the robot's last linear displacement is equal to the displacement over the map. While this assumption would fail whenever the robot is not perfectly over the expected path, it was found to work well as a short-term prediction. The premise is that the sensor model together with the particle re-sampling quickly corrects this discrepancy.

3.4.4 Sensor Model

The sensor model has the purpose of estimating how likely is that the robot is at a given pose $\mathbf{x}_t$, given that it is currently observing a certain set of features in the image and given the features that would be visible at



(a) Current pose equal to reference pose

(b) Counter-clockwise rotation w.r.t. the reference

(c) Left-translation w.r.t. the reference

(d) Forward translation w.r.t. the reference

Figure 3.5: Distributions of azimuth and elevation differences obtained for different relative poses w.r.t. a reference. Azimuth histogram is on the lower edge while the elevation histogram is on the left-side of the images. Dots correspond to currently extracted features and circles correspond to landmarks. A green circle signifies a non-mached landmark, a red dot signifies a non-matched feature and a pair of green dot and circle signify a correspondence.

$\mathbf{x}_t$, i.e. the landmark views reference from segment s_i at distance d. To compute this likelihood, both sets of features are matched, from which information regarding the relative pose can be retrieved, by estimating the optical-flow.

Based on the principles outlined in Chapter 2, appearance cues that allow to distinguish between pairs of poses having only a rotational error from those having translational error as well can be identified. These cues are the basis of the proposed sensor model.

By revisiting the image-flow equations (2.3) and (2.5), if the distribution of bearings differences between two poses is analyzed (see Figure 3.5 on page 56), it becomes evident that in the absence of a translation error $\mathbf{v}$, only the rotational flow contribution will be present. Furthermore, since this contribution does not depend on the distance to the landmarks, the image-flow obtained for every pair of matching features will be the same. In other words, in the histograms for azimuth and elevation differences, a single peak will exist around the values ω_y and ω_x respectively (assuming $\omega_z = 0$, guaranteed by flow derotation). On the other hand, in the presence of translation, bearing differences in each histogram will be grouped according

to the relation between relative distances (see Section 2.3) and translation in each axis. Thus, in this case the histograms will appear more dispersed. The general case of combined rotation and translation differs from the latter only in that the histogram appears to be shifted by an amount proportional ω_x and ω_y.

At this point, two ambiguities should be pointed out. First, if all landmarks are at a very similar distance, again a single peak will be present in the presence of only a translation. This means that in the general case, a translation cannot be distinguished from a purely rotational error. This problem, though, is inherent to monocular vision and is not expected to be solved without further sensory information. Second, longitudinal translation induces an image-flow field that affects both azimuth and elevation axis (see Figure 3.5d on page 56). Thus, it is difficult to distinguish between a longitudinal error and a combination of lateral and vertical translation. However, since the feature-descriptor used to obtain correspondences is not scale-invariant, the number of matches rapidly decays in the presence of longitudinal translation.

In consequence, in order to recognize translational errors, a measure of histogram dispersion can be used as part of the sensor model , such as the the entropy H:

$$H(X) = -\sum_i p(x_i)\, log\, p(x_i) \tag{3.3}$$

where X is a discrete random variable, and x_i are the center-values of each bin in the histogram of X.

Thus, the likelihood that the robot is at a pose $\mathbf{x}_t$ given the azimuth and elevation differences $\dot{\Phi}$ and $\dot{\Theta}$ obtained by matching currently seen features against a pose in the map, disregarding any rotational errors, will be proportional to

$$H'(\dot{\Phi}) \cdot H'(\dot{\Theta})$$

, where $H'(X)$ is the inverse and normalized entropy

$$H'(X) = 1 - \frac{H(X)}{log(B)} \tag{3.4}$$

and B the number of bins in the corresponding histogram. The value of $H'(X)$ will thus be inside the $[0, 1]$ interval, where $H'(X) = 1$ represents the case of a histogram with a single peak.

In [48] the variance of bearing differences was used in a similar way. However, variance is sensitive to bad correspondences (as it happens to the mean, a problem described in Section 2.3), while $H(X)$ does not since it depends only on the shape of the distribution and not the values themselves. On the other hand, $H(X)$ does not depend on the number of samples which produced the corresponding histogram. For example, a histogram with N equal samples results in the same value for $H'(X)$, independently of N. This also implies that for very low N, the entropy cannot be trusted. In the localization problem, if two unrelated poses are compared, the entropy will be obtained from noise arising from purely bad correspondences. To avoid this problem, the number of samples k used to compute the entropy can be used to weight this measurement.

In conclusion, the likelihood given by the sensor model for a pair of poses $\mathbf{x}_t, \mathbf{p}_i$, is proportional to weight w_i:

$$w_i \leftarrow k \cdot H'(\dot{\Phi}) \cdot H'(\dot{\Theta}) \tag{3.5}$$

. Prior to re-sampling, these weights are first normalized by

$$w_i' = \frac{w_i}{\sum w_i}$$

.

As previously mentioned, since the entropy measurements will be noisy for low values of k, in these cases it was chosen to simply set the weight to zero to discard this particle during re-sampling. In fact, if both histograms appear to be mostly flat for a given particle, it will not useful for the mode-based path-following controller, since the mode will also be erratic. Thus, convergence towards these particles is to be

avoided, which could happen if all particles are bad hypotheses. In other words, the weight will be simply set to zero when

$$w_i < k_{\text{min}} \cdot (H'_{\text{min}})^2$$

, where k_{min} is the required minimum number of matches and H'_{min} the required minimum value for $H'(\cdot)$. To deal with these zero-weight particles, during re-sampling, if a particle has a weight of zero, it is replaced by a new particle obtained by uniformly sampling the state space of the particle (i.e. it is randomly placed in the map).

3.4.5 Single-hypothesis computation

While MCL continuously estimates a set of localization hypotheses, for path-following a single reference needs to be obtained. The idea is to compute a single-hypothesis as an estimation of the mode of the distribution of particles, i.e. the pose in the map for which the probability density is highest.

A naïve approach would be to take the center of mass of the positions of all particles as an estimate of the mode. However, since the map uses a manifold based representation it is undesirable to use an approach which relies on the consistency of absolute particle positions. Since some particles may be very far from each other, this procedure would be highly inaccurate over large graphs and with spread out particles. A better strategy in this case consists in first estimating the local density of weights nearby each particle and then computing the center of mass only of the particle positions relative to the highest-density particle. Since the map has the property of being locally euclidean, this computation is more precise than when very distant particles are compared.

Specifically, the single-hypothesis $\hat{\mathbf{p}}_i$ is obtained as follows. First, the local density of each particle $\mathbf{p}_i$ is computed by summing the weights of the particles up to a certain distance of $\mathbf{p}_i$. The particle $\mathbf{p}_i^*$ with the highest local density is deemed the winner of this process, which is defined as:

$$\mathbf{p}_i^* = \arg\max_i \delta(\mathbf{p}_i)$$

where $\delta(\mathbf{p}_i)$ is the local density given by

$$\delta(\mathbf{p}_i) = \sum_{\forall p_j \in \text{local}(\mathbf{p}_i)} w_j$$

and local$(\mathbf{p}_i)$ is the *local graph*[49] of particle $\mathbf{p}_i$. The local graph is a sub-graph of the complete graph (the map) containing only particles reachable from a given particle $\mathbf{p}_i$ (i.e. a topological path exists) and which are at most at distance τ:

$$\text{local}(\mathbf{p}_i) = \{\mathbf{p}_j / \text{ reachable}(\mathbf{p}_i, \mathbf{p}_j) \wedge |\text{lp}(\mathbf{p}_j, \mathbf{p}_i)| \leq \tau\}$$

where $\text{lp}(\mathbf{p}_j, \mathbf{p}_i)$ is the *local position* of particle $\mathbf{p}_j$ with respect to particle $\mathbf{p}_i$. This local position is obtained by chaining the relative transformations stored in each segment of the topological path connecting $\mathbf{p}_i$ to $\mathbf{p}_j$. In this way, $\text{lp}(\mathbf{p}_j, \mathbf{p}_i)$ is a relative representation of the position of $\mathbf{p}_i$ w.r.t. $\mathbf{p}_j$. Thus, the norm $|\text{lp}(\mathbf{p}_j, \mathbf{p}_i)|$ corresponds to the relative distance of $\mathbf{p}_j$ to $\mathbf{p}_i$, within the locally euclidean space defined by the local graph of $\mathbf{p}_i$.

Then, the weighted euclidean centroid $\mathbf{c}^*$ (using the weight of each particle) of the positions of the particles contained in the local graph of $\mathbf{p}_i^*$ is obtained as:

$$\mathbf{c}^* = \sum_{\forall p_j \in \text{local}(\mathbf{p}_i^*)} w_j \cdot \text{local}(\mathbf{p}_j)$$

. Finally, the single-hypothesis $\hat{\mathbf{p}}_i$ will be a pose in the sub-graph defined by local$(\mathbf{p}_i^*)$, found by the orthogonal projection of $\mathbf{c}^*$ to the segment $\hat{s}_i$ in the sub-graph for which the orthogonal distance is the smallest (i.e. the pose in the local graph closest to $\mathbf{c}^*$):

.

$$\begin{aligned} \hat{s}_i &= \underset{s_i \in \text{local}(\mathbf{p}_i^*)}{\arg\min}\ \text{dist}(c^*, \text{source}(s_i), \text{target}(s_i)) \\ \hat{d} &= \text{proj}(c^*, \text{source}(s_i), \text{target}(s_i)) \end{aligned} \tag{3.6}$$

where source(s_i) and target(s_i) are the local positions of the source and target nodes of segment s_i, dist(x, a, b) is the orthogonal distance of point x to segment $\overrightarrow{ab}$ and proj(x, a, b) is the orthogonal projection of point x to segment $\overrightarrow{ab}$.

As a result of this process, a pose $\hat{\mathbf{p}}_i$ is computed from the complete set of particles. However, this single-hypothesis is a valid representation of the particle set only when a distinguishable mode is present and if the current hypothesis have a high likelihood of being close to the true robot location in the map. Thus, a *quality* measure of the single-hypothesis is introduced, simply given by $\delta(\hat{p}_i)$. The single-hypothesis is deemed *valid* only when

$$\delta(\hat{\mathbf{p}}_i) > \delta_{\text{min}}$$

, where $\delta_{min} \in [0, 1]$ is a user-defined constant.

3.4.6 Maintaining Particle Diversity

While it is expected that particles represent the general shape of the underlying probability distribution, if this distribution is multi-modal (different candidate locations appear to be possible) in practice with MCL all particles actually converge to one of the modes. This behavior can become problematic for localization if not handled accordingly.

Since during re-sampling the new particle set is obtained by choosing existing particles from the previous set, it could occur that eventually all particles are equal, if one particle has a really high weight. This problem is known as "particle deprivation" since the sampling-based representation becomes degenerate, with a peak probability of one around a single value. Thus, the goal to avoid this situation is to maintain particle diversity.

One strategy is to make the motion model non-deterministic which has the consequence of, after the prediction step, introducing diversity among the various instances of a particle which was chosen multiple times during re-sampling. This non-determinism can be achieved by simply adding noise to the motion model (as explained in 3.4.3).

Another strategy is to allow for particles with zero weights, instead of forcing every particle to have a non-zero probability of being selected, which could ultimately result in the particle deprivation problem. During re-sampling, a particle with a weight of zero is handled by simply choosing a new random one instead. This has the effect of "re-initializing" a particle if it is not a plausible hypothesis and also allows for handling global initialization when most likely no particle represents a valid hypothesis.

3.4.7 Global Initialization

Global initialization means to solve the localization problem in the event of previous localization failure or when localization is first initialized without *a priori* information.

With the proposed approach, when localization is enabled during the repeat phase, particles are sampled randomly over the entire map. It is then expected that at least one particle will be closer to the true location of the robot and thus MCL will eventually converge. However, if no particle is valid, MCL may converge to an invalid set of particles. With the strategy of allowing zero-weight particles (see 3.4.6), this scenario is not as common. However, MCL may still take long time to place particles near the correct location.

Thus, a variant of the original MCL algorithm can be used which directly addresses this problem and is known as Augmented Monte Carlo[50]. With this variation, a number of the lowest-weighted particles

are randomly re-sampled, based on the observation of the average of the weights over time. When weights appear to be decreasing with respect to the overall trend, it is assumed that particles do not represent the true robot location very well, and thus different hypothesis should be tested.

Specifically, a number r of particles are randomly re-sampled according to

$$r = M \cdot \max(r_{min}, \max(0, 1 - \frac{w_{\text{fast}}}{w_{\text{slow}}}))$$

, r_{min} is the minimum desired proportion of particles that should be randomly sampled (irregardless of the observed weights), M is the total number of particles, $\bar{w}_t$ is the average of the weights at current time and w_{slow} and w_{fast} are two low-pass filtered versions of $\bar{w}$, according to

$$\begin{aligned} w_{\text{slow}} = &\quad w_{\text{slow}} + \alpha_{\text{slow}}(\bar{w} - w_{\text{slow}}) \\ w_{\text{fast}} = &\quad w_{\text{fast}} + \alpha_{\text{fast}}(\bar{w} - w_{\text{fast}}) \end{aligned}$$

, where typical values are $\alpha_{\text{slow}} = 0.015$ and $\alpha_{\text{fast}} = 0.1$. In other words, more random particles will be sampled when a sudden drop in $\bar{w}$ (measured as the ratio w_{fast} to w_{slow}) is detected.

3.5 Autonomous Navigation

During the repeat phase, a specific path over the complete map is followed autonomously. Once the robot is localized as outlined in Section 3.4 and a valid single-hypothesis is computed, a path from the current robot location leading to a given goal is first obtained. Then, path-following can start to repeat the portions of the complete map using the bearing-only controllers presented in Chapter 2.

For a diagram of the navigation module see Figure 3.6 on page 60.

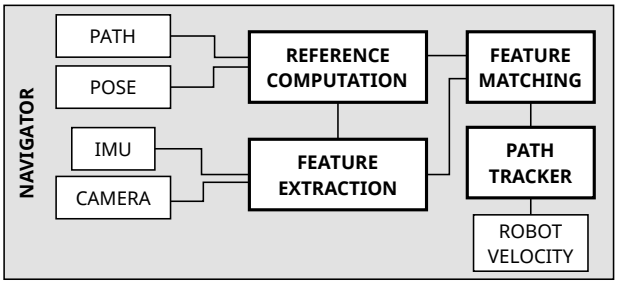


Figure 3.6: Conceptual diagram of the navigation module

3.5.1 Path-Planning

The process of path-planning in the proposed VTnR approach involves finding the shortest path $P = s_i, \ldots, s_j$ (using Dijkstra's algorithm) over the topological graph, defined as an ordered list of interconnected segments s_i to s_j, starting at the current robot pose $\mathbf{x}_t$ and ending at a given goal pose $\mathbf{g}$, where

$$\begin{aligned} \mathbf{x}_t = &\quad \begin{bmatrix} s_i & d \end{bmatrix} \\ \mathbf{g} = &\quad \begin{bmatrix} s_j & d' \end{bmatrix} \end{aligned}$$

and the length of the path $|P|$ is given by

$$|P| = \sum_{s_i \in P} |s_i|$$

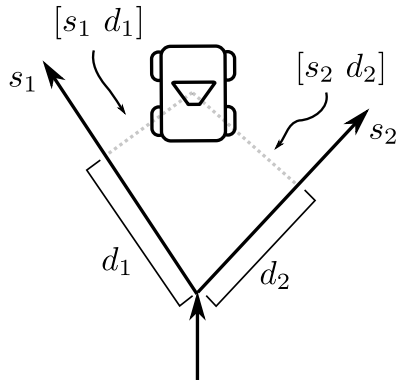


Figure 3.7: Two possible representations of the robot's pose in the map, in the presence of a bifurcation

. While a path with the lowest lenght $|P|$ is obtained in this process, since the map is not globally consistent, in reality it may not actually be the shortest one, but it will still be an acceptable solution.

During navigation, the pose of the robot is periodically checked to test whether it is still contained in the computed path. If not, a new path is searched. Finally, since $\mathbf{x}_t$ is the result of the probabilistic localization and will be probably noisy, the goal is considered to be reached if the goal is close enough to the robot, according to

$$|\mathrm{lp}(\mathbf{g}, \mathbf{x}_t)| < \epsilon_{\mathbf{g}}$$

, where $\epsilon_{\mathbf{g}}$ is user-defined constant.

3.5.2 Navigation Reference

For path-following under the bearing-only approach, a reference pose needs to be determined to guide the navigation. In principle, this reference would be the single-hypothesis obtained from navigation, i.e. the estimated location of the robot in the map. However, due to the use of a manifold representation, the pose of the robot is not uniquely defined as it is expressed in relative terms. Thus, from the actual single-hypothesis, an equivalent but alternative representation may be needed.

For example, consider the robot located in the middle of a bifurcation (see Figure 3.7 on page 61). The robot could then be localized with respect to either segment and still be correct. In fact, the proposed localization method will only find one possible representation (the single-hypothesis). Thus, in order follow one specific path, the navigation reference should be actually placed over the desired segment to be approached.

To compute the navigation reference $\mathbf{x}_t^*$, given the estimated robot pose $\mathbf{x}_t = \begin{bmatrix} s_i & d \end{bmatrix}$ and a path $P = s_a, \ldots, s_b$, three cases can be distinguished. If $s_i \in P$, then $\mathbf{x}_t^* = \mathbf{x}_t$. Otherwise, the projection of $\mathbf{x}_t$ to the path is obtained in the same fashion as (3.6), where in this case the local graph to consider is local$(\mathbf{x}_t)$. However, this is only possible if $\mathbf{x}_t$ is relatively close to the computed path. If not, no valid reference can be obtained and navigation will not be possible.

Another measure to handle the case of ambiguous pose representation, is to purposely bias localization towards poses over the path. To do so, during the prediction step of MCL, in the presence of a bifurcation, instead of choosing a random segment as in (3.2) if a given outbound segment is contained in P it can be chosen instead. This has the effect of *suggesting* a desired representation of the robot's pose in the presence of a bifurcation and a computed path. It should be stressed that this does not prevent MCL to eventually place particles on any other branch during re-sampling if it results in a better match.

3.5.3 Path-Following Controller

While a large number of approaches to the problem of visual path-following exist, since the focus of this thesis is not to propose a novel controller, a simple strategy based on the principle of bearing-only navigation was used.

Once a valid pose $\mathbf{x}_t$ and a navigation reference $\mathbf{x}_t^*$ are defined, a bearing-only path-following controller can be executed to drive the robot over the desired path P. Given the set of feature correspondences obtained by matching features seen in $\mathbf{x}_t$ and $\mathbf{x}_t^*$, bearing differences in the form of azimuth and elevation histograms can be computed. Then, any of the bearing-only controllers proposed in Chapter 2 can be applied to generate appropriate motion commands, in the form of linear and angular velocities.

However, so far path-following was simplified to the case of a controller attempting to reach a given setpoint, i.e. it is assumed to not change considerably during convergence as when following a straight path. In general, this type of control is usually termed *carrot follower*, since path-following is achieved by means of continuously trying to reach a reference pose, which moves as the robot moves. One problem of this approach is that when the path presents closed curves, the reference pose (and thus the relative error seen by the controller) will move during convergence.

Furthermore, carrot following requires for an error to already exist for a control response to be generated. In other words, the controller acts late, even while the path is known beforehand. Thus, in order to closely follow a non-trivial path in this way, the proposed bearing-only controllers would have to be tuned aggressively affecting the stability of the system (and which is generally not desired for the safe operation, specially on the case of aerial robots).

A similar problem (a delay in response) occurs due to the use of nested position/velocity/acceleration PID controllers, usually found in the embedded control hardware (the *autopilot*) of some aerial robots (as it is the case for the aerial platform used in this thesis). These lower-level controllers are also generally tuned for low speeds, which results in a slower control response when setpoints are rapidly changed (e.g. suddenly requesting zero velocity while the platform is moving) than if they were tuned tuned more aggressively.

Lookahead

For the aforementioned reasons, a carrot follower approach such as the one proposed so far will not perform as well as other more sophisticated controllers. However, in order to avoid introducing complexity using other forms of more advanced control (such as Motion Prediction Control or MPC), a simple modification to the proposed approach is introduced. Specifically, instead of placing the navigation reference at the closest location in the map, it is placed some *lookahead* distance ahead*[51]*. This strategy allows to anticipate the changes in the reference pose (due to a curved trajectory) and thus overcoming the inherent response delay, for a non-agressive control tuning.

In other words, instead of using $\mathbf{x}_t^*$, a reference $\mathbf{x}_{t+l}^*$ is used, representing a pose in the map ahead of $\mathbf{x}_t^*$ by some distance $L = |z_t| \cdot l$ ahead of $\mathbf{x}_t$ over the computed path, where z_t is the linear velocity of the robot and l the lookahead, measured in seconds. This formulation allows to set the lookahead independently on the robot's linear speed.

An experimental analysis of the effect of the introduction of the lookahead is presented in 5.6.3.

Bearing-only Controller

In this thesis, a bearing-only controller for the case of ground and aerial robots is considered. In particular, from the controllers outlined in Chapter 2, (2.10) was used for the majority of the experiments and is here

repeated:

$$\begin{array}{rcl} w_t & \leftarrow & -k_1\dot{\Phi} \\ y_t & \leftarrow & -k_2\dot{\Theta} \\ z_t & \leftarrow & c \end{array}$$

where w_t, y_t, z_t are the yaw, vertical and forward velocities, respectively. For the case of a ground-robot, vertical velocity has no meaning and only w_t and z_t are used. Alternatively, for vertical altitude, the altitude difference w.r.t. the reference, as estimated through the downward looking sonar range-finder of the aerial robot, was used in some experiments instead of $\dot{\Theta}$, for the reasons mentioned in Chapter 2. Finally, (2.11), which includes lateral-velocity control, was only experimentally evaluated in simulation as the required absolute yaw angle estimation was not precise enough in the real robot platform.

3.5.4 Algorithm

In listing 2, a pseudo-code description of the navigation algorithm is presented. Here, the robot is assumed to be localized (valid single-hypothesis) and a topological path should have been already computed. The navigation algorithm then proceeds as follows.

Algoritmo 2: Repeat Phase

Data: M: map of segments $\{S_1, \cdots, S_N\}$, P : a path over M , $\mathbf{x}_t$: the current pose of the robot

1 **Loop**

2 $\left[\begin{array}{cc} \mathbf{s}_i^* & d^* \end{array}\right] \leftarrow get\_reference(\mathbf{x}_t, P)$ `/* get current reference position (segment and distance along segment) */`

3 $V \leftarrow get\_landmark\_views(\mathbf{s}_i^*, d^* + L)$ `/* retrieve landmark views expected at a distance` L `(lookahead) ahead of the reference, w.r.t. the start of` $\mathbf{s}_i^*$ `*/`

4 $F \leftarrow extract\_features()$ `/* extract image features */`

`/* obtain correspondences between current features and expected landmark views */`

5 $M \leftarrow match(V, F)$

`/* compute corresponding azimuth and elevation difference histograms */`

6 $\dot{\Theta}, \dot{\Phi} \leftarrow compute\_histogram(M)$

`/* compute control values */`

7 $\begin{array}{rcl} w_t & \leftarrow & -k_1\dot{\Phi} \\ y_t & \leftarrow & -k_2\dot{\Theta} \\ z_t & \leftarrow & v_{\text{repeat}} \end{array}$

First, a reference pose is computed as described in 3.5.2, based on the current robot pose $\mathbf{x}_t$ and the computed path P. Then, the set of landmark views expected to be seen at a distance L ahead of the reference is retrieved as presented in 3.3.2. The features contained in these views are then matched to those extracted in the current image. From the set of correspondences, azimuth and bearing differences are obtained and the corresponding histogram is computed. Finally, the modes of these histograms are used as input to the bearing-only controller (2.10).

Capítulo 3

Método de navegación visual de aprendizaje y repetición (resumen)

En este capítulo se describe a grandes rasgos los componentes que conforman la solución propuesta al problema de la navegación visual siguiendo la técnica de aprendizaje y repetición. El método propuesto consiste principalmente de: un sistema de construcción de mapas representando caminos a seguir, un sistema de localización robusto y un método de seguimiento de caminos (basado en las leyes de control propuestas en el capítulo anterior).

El sistema de construcción del mapa tiene como objetivo representar un camino a ser repetido, mediante una descripción topológica que a su vez incluya la información visual necesaria para poder resolver los problemas de la localización y la navegación. Para la representación topológica se propone el uso de un grafo, en el cual un nodo dado corresponde a una ubicación determinada mientras que las aristas describen la conectividad de dichas ubicaciones. Para representar la información visual, en cada arista se almacena información respecto de la apariencia de características salientes observadas durante el recorrido. De esta forma, es posible luego obtener las características visuales que se esperan ver en una posición específica sobre el mapa.

Por otro lado, el método de localización consiste en determinar la posición del robot sobre el camino aprendido. Para ello se propone una formulación probabilística que sigue la técnica de Localización de Monte Carlo, donde se representan múltiples hipótesis de localización mediantes *partículas*. El objetivo, entonces, es determinar qué partículas representan con mayor probabilidad la pose real del robot. Para ello, se comparan las características visuales observadas actualmente por el robot con las que se esperaría ver en cada posición del mapa correspondiente. De este conjunto de partículas, entonces, se obtiene una única hipótesis que es la que se utilizará luego para guiar la navegación.

Por último, para resolver la navegación bajo el enfoque propuesto, se asume que la localización del robot está resuelta y que un camino sobre el grafo topológico desde dicha posición a una meta dada ya fue calculado. De esta forma, la navegación se reduce a aplicar en forma continua alguna de les leyes de control propuestas en el capítulo anterior, para guiar al robot sobre el camino deseado.

Chapter 4

Aerial Experimental Platform

In this chapter, the design and build process of an experimental aerial platform, the LRSE quadcopter (Figure 4.1 on page 67), which was used for experimental evaluation of the VTnR method proposed in this thesis, is presented.

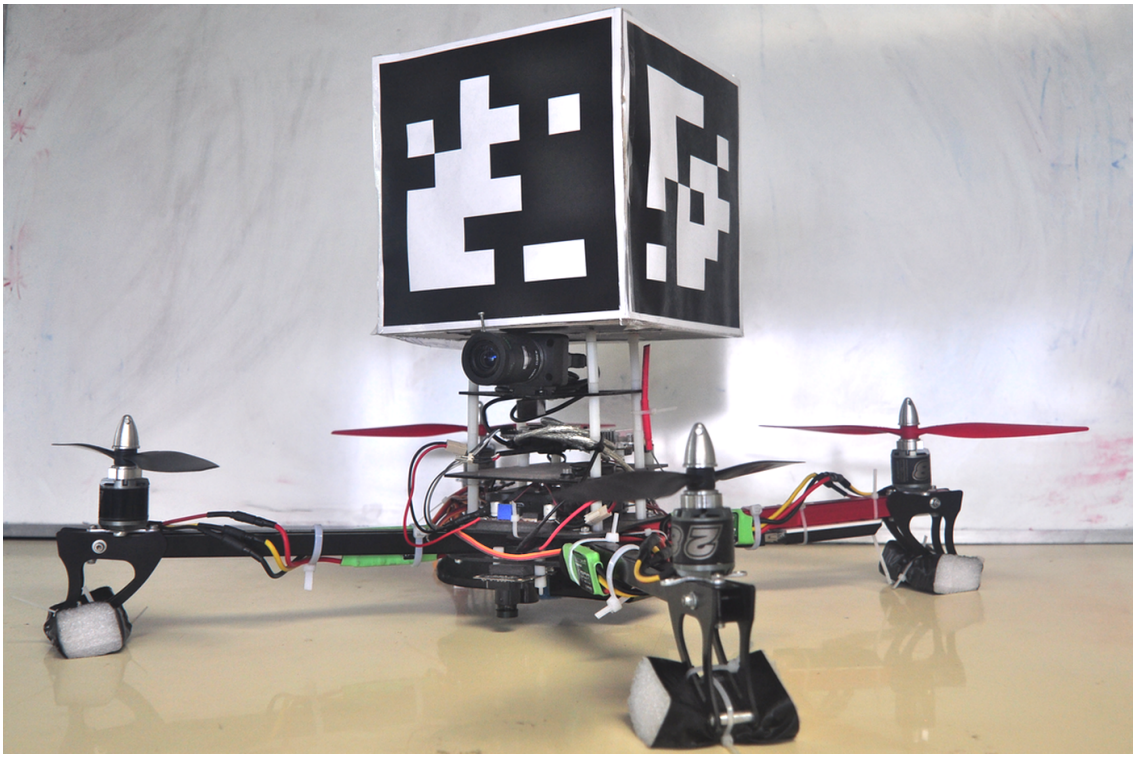


Figure 4.1: LRSE quadcopter (the cube is used by an external localization system, described in Chapter 5).

4.1 Motivation

The motivation to design a custom aerial robot platform is to overcome the difficulties inherent to using commercially available solutions. In this sense, the LRSE quadcopter is designed to be used as a research platform, where different sensors can be mounted depending on the task at hand. Moreover, it features

on-board open-source control hardware and software, which allowed for much easier interfacing of the proposed VTnR navigation method. At the same time, the build and experimental process presents many challenges that are sometimes hidden in a commercial platform and which are of interest to analyze in the context of the development of an autonomous navigation method.

For the proposed VTnR method, early experiments were performed using the Parrot AR.Drone V1 quadcopter, which features remote WiFi control and the sending of on-board sensors, composed mostly of a forward-looking camera and a downward-looking optical flow sensor. This platform is very lightweight ($\sim 420\,g$) and quite stable, allowing for high-level control by means of velocity commands. On the other hand, the main limitation of this platform is that it is not able to carry any other sensors or hardware, such as an on-board computer. Thus, computation was performed off-board when testing the proposed method with the Parrot AR.Drone. The second limitation, is the inability to access the state estimation and control hardware and software, which considerably limits its usefulness as a research platform.

With this experience in hand, the aim during the design process of the LRSE quadcopter was to obtain the same capability of self-stabilization and high-level control while adding the possibility of carrying extra sensors and on-board execution hardware.

4.2 Platform Description

The LRSE quadcopter is composed mainly of a set of high-level sensors, such as an on-board forward-looking camera and a down-looking optical-flow sensor, an on-board control hardware (the *autopilot*) which takes care of low-level control and pose estimation and a high-level embedded computer, where the proposed VTnR is executed (see Figure 4.2 on page 69). In the following sections, all components are described.

4.2.1 Base Components

The base components of the aerial platform include: the main structure or *frame*, electric motors, propellers, speed controllers and battery. These, together with the main flight computer (the *autopilot*) comprise the basic elements required for achieving a minimal aerial vehicle capable of flight. The remaining components such as the companion computer, sensors, etc. can be considered an optional *payload* which the vehicle will carry. This consideration is important when choosing these base components. The decision on which particular base-components to employ as part of the final design is non-trivial. A number of aspects come in to play which have to be considered and determined beforehand in order to drive the design process, such as the desired/acceptable platform size and weight, payload capability, flight autonomy and flight characteristics (smooth and stable or fast and aggressive).

Due to the increasing popularity of the multicopter-type aerial platforms, several tools have appeared which ease this design process, by non-expert users without engineering or aeronautical knowledge. These take the form of *calculators,* where the user can simply input a number of parameters and the tool can automatically estimates the efficiency of the component combination, which grossly determines the flight autonomy. This type of tools can offer great help to asses a particular combination of components and experiment with slight variants in order to consider alternatives of individual choices. However, some basic knowledge is required in order to find an *initial guess* from which an optimized design can be obtained. Thus, several factors can be considered beforehand which, in the scope of this thesis, can be used to find this starting point.

Main Structure or *Frame*

In terms of the main structure, the choice comes mostly down to three factors: weight, robustness and shape, which will determine the layout of the rest of components. Most frames are typically made of a combination of aluminum, fiber glass and/or carbon-fiber. In principle, a carbon-fiber frame can be the ideal option since

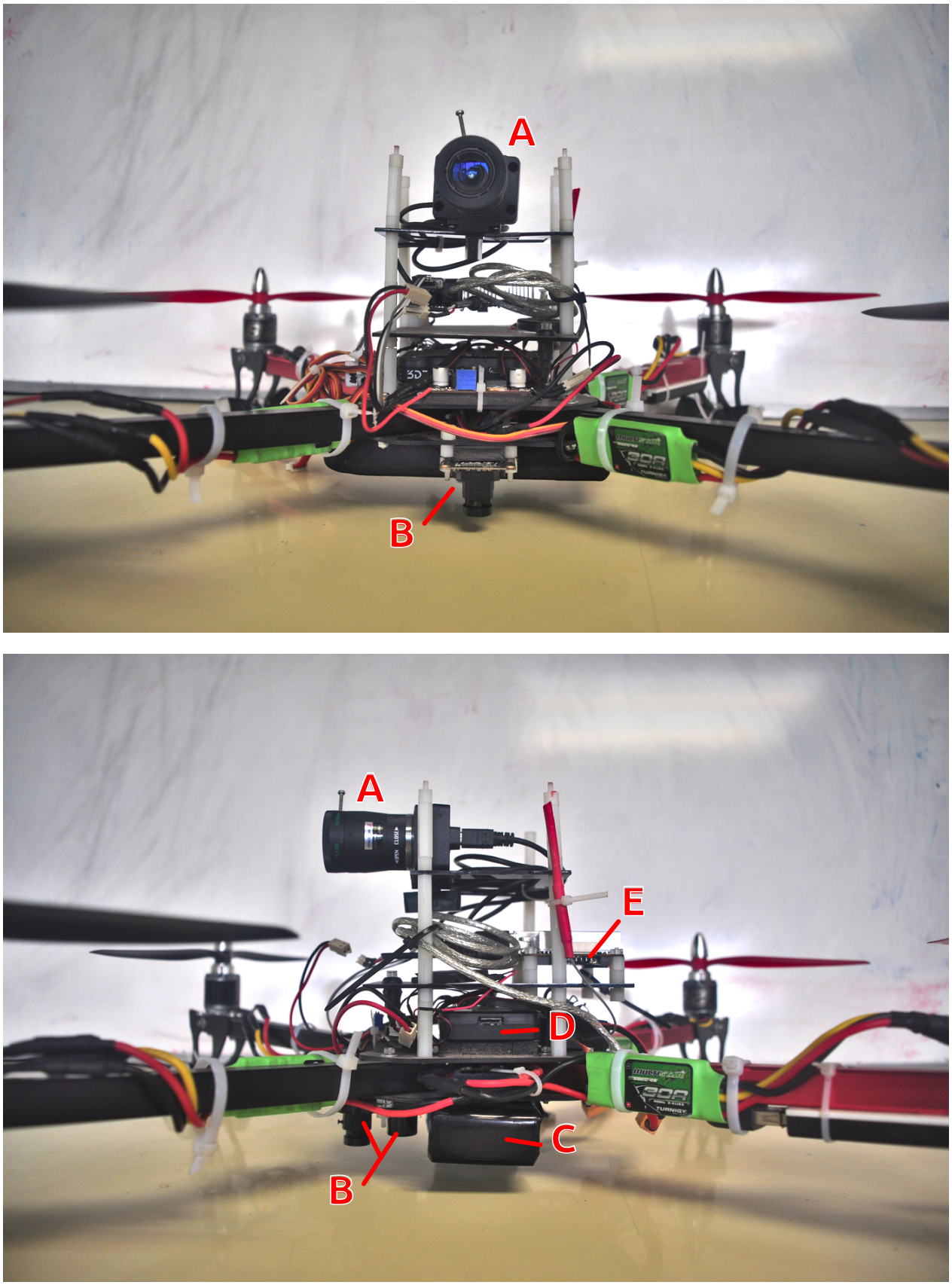


Figure 4.2: LRSE quad-copter side and front views. A: the Firefly MV camera, B: Optical-flow sensor (camera + sonar range-finder), C: battery, D: Pixhawk autopilot, E: Odroid-U3 embedded computer

this material is quite durable and of low weight. In this case, the negative aspect is its relatively high price. On the other hand, aluminum can be less expensive and also a stiff material, at the expense of some added weight. These two materials are typically used for the motor booms, while it is common to find fiber glass connecting-components such as the center-plate. This material is also light but not really durable so it can be weak point in the structure.

Figure 4.3: The x550 quad-copter frame

For the particular case of an aerial experimental platform, durability and weight are important factors, however in order to keep the price down an aluminum-fiber glass structure was chosen. In order to be able to accommodate the full set of components which the experimental platform will carry, the X550 frame was chosen, a medium-size structure of $55\,\mathrm{cm}$ in diameter and $312\,\mathrm{g}$ of weight (Figure 4.3 on page 70).

Motors and Propellers

For multi-copters, particular propellers are required in contrast to those used for standard fixed-wing vehicles given the high rotation speeds (in the order of 10 to $20,000$ rpm) and the thrust expected to produce by each motor. Since multi-copters generate lift solely using on their motors (and not using their structure, such as fixed-wing vehicles), the propeller pitch, i.e. the torsion angle of the propeller blades is also larger in this case, giving more thrust per revolution. Propeller sizes can vary greatly and it is difficult to determine in advance which size should be used, as this is a value that is related to other parameters such as the force of the motor.

The electrical motors employed with multi-copters vary with size and weight, but a main parameter is usually employed to describe a motor's thrust capability which is the KV or *RPM constant* parameter. This value describes the number of revolutions-per-minute that the motor will turn when one Volt is applied with no load. Generally, a low KV number indicates a slow-turning but high-torque motor, which would be coupled to a high-thrust propeller of large-size and/or large-pitch. On the other hand, a high KV motor will spin faster and will require a higher RPM to achieve the same amount of thrust than a lower KV motor. While slower turning motors with large propellers would allow to generate larger lift and thus fly with higher payload, the consumed current will also be higher since the propeller will have a higher load. Finally, while a relatively heavier platform with low KV values might lead to a more stable flight, the thrust response of the motors will also be slower, which may be problematic for some applications.

After considering different KV values and propeller sizes and by using on-line multicopter efficiency calculators, a Turnigy 2826 800kv motor was chosen ($54\,\mathrm{g}$ of weight) with a set of 10×4.5 propellers (10 in in diameter with $4.5\,\mathrm{in}$ of pitch). A slightly larger set of propellers of 11×4.5 were also considered to determine

(a) The Turnigy NTM 2826 800kV motor

(b) Standard CW and CCW 10x4.5 propellers

Figure 4.4: Chosen motor and propellers

if longer flight time was possible with a heavy payload. The maximum current draw of the motors is $20\,\mathrm{A}$ at full load.

Electronic Speed Controllers (ESC)

In order to control the rotation speed of the motors, which required high-currents to operate, by means of digital signals coming from the autopilot, a component known as Electronic Speed Controller (ESC) is required.

The ESC consists mostly of a high-current motor driver which receives power directly from the battery and generates the motor control signals, based on Pulse Width Modulation (PWM) digital signals, which encode the desired speed of the motor. The main parameters to consider for an ESC are its voltage rating and its current rating. The former corresponds to which type of battery can be used to power the ESC, while the latter determines how much current can it carry from the battery to the motor. It is generally desired for safety reasons that the ESC handles about 1.5 times the maximum current that the motor is able draw.

Battery

While different battery types exist, differing in their chemistry composition, the most widely used technology is by far Lithium-Polymer or LiPo. These type of batteries have generally high capacity and allow for high discharge-rates, such as those required for a multi-copter. At the same time, the weight is relatively low when compared to other types. However, it is still one of the heaviest components in multicopters.

The battery discharge rate is an important factor which dictates how much current can be continuously drawn without negatively affecting the battery life. For a multi-copter, for hover flight a relatively large current is constantly drawn and thus a battery with a high-discharge rate (leaving a fair margin from the expected current draw) should be used. Discharge rate is usually described with a "C" number, which indicates a safe current draw proportional to the capacity of the battery. For example, for a battery of 4200mAh capacity with 35C discharge-rate, a current of $35 \times 4.2A = 147A$ should be possible to be drawn continuously.

Figure 4.5: Turnigy Multistar 30A ESC

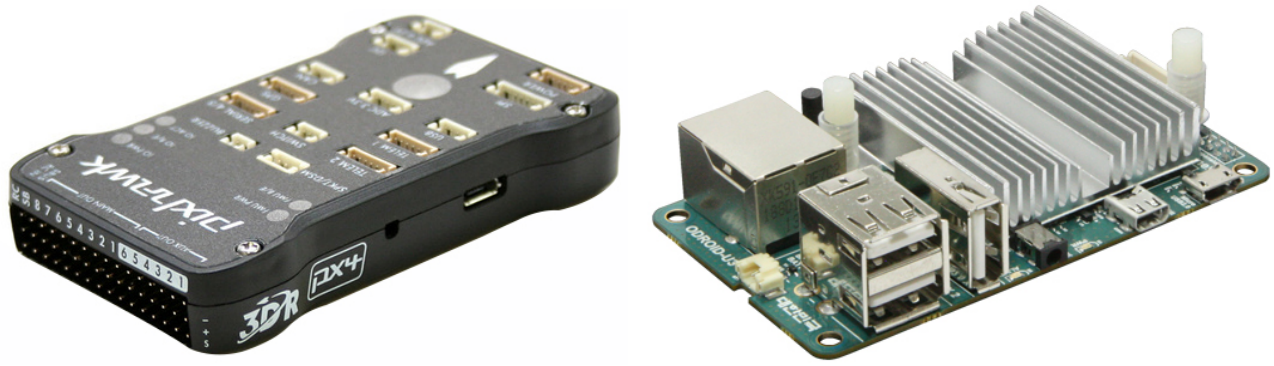
Figure 4.6: Lumenier 3s 5200mAh LiPo battery

For medium-sized multicopters, a rule of thumb is to have a battery capacity of about $4\,\mathrm{Ah}$, which equates to being able to provide a constant current of $4\,\mathrm{A}$ for one hour. Besides capacity, the voltage of the battery is to be considered. As batteries are in reality composed of many "cells" and each cell having a voltage range given by the battery chemistry, the choice comes down to selecting a number of cells, identified by the number "S". Again, typically 3S batteries are used for medium-sized quad-copters. While a thorough analysis should be done to determine the best battery choice, in terms of balancing weight (which is proportional to battery capacity) and flight duration, it is practically unfeasible due to the cost of acquiring multiple batteries. Thus, for the LRSE quadcopter, from the available options in the market, the most lightweight 3S battery at around 4 to $5\,\mathrm{Ah}$ was chosen. The selected option was then the Lumenier 3S 5200mAh, battery, which has a constant discharge rate of 35C, with bursts up to 70C.

4.2.2 Fight Computer (*autopilot*)

The autopilot is an essential component of any autonomous aerial robot. At the very least, it is required to achieve the low-level stabilization of the vehicle, since the fast dynamics and inherent instability of multicopters require fast closed-loop control. To this end, the autopilot is mainly composed of inertial sensors (a gyroscope, providing angular velocity readings, and an accelerometer, providing acceleration readings) and a micro-controller, which runs the software in charge of state estimation and vehicle control.

Numerous options are available nowadays, mostly differing in the level of complexity and available computational power. While hobby-oriented autopilot hardware is sufficient to achieve stabilization and



(a) The 3DR Pixhawk autopilot

(b) HardKernel's ODroid-U3 embedded computer

Figure 4.7: On-board control hardware

allow for manual operation of the vehicle, for the case of a research-oriented aerial platform it is necessary to be able to autonomously control the vehicle from embedded hardware, without human assistance. At the same time, for high-level navigation algorithms (such as the one proposed in this thesis), the main flight computer is not a suitable execution platform since it lacks the required processing power. Moreover, low-level control of the vehicle is a critical task and thus isolation from untested research-level software is desirable. High-level computation is generally handled by a second processing unit, usually referred to as the *companion computer* (see 4.2.3). Thus, a suitable communication interface between the companion computer and the autopilot needs to exist.

For the aforementioned reasons, for the LRSE quadcopter platform, the Pixhawk autopilot was chosen (see Figure 4.7a on page 73). This autopilot is the result of an independent, open-hardware project aiming at providing a high-end autopilot to the academic, hobby and industrial communities at low costs and high availability. The project is supported by the Computer Vision and Geometry Lab of ETH Zurich and by the Autonomous Systems Lab and the Automatic Control Laboratory. While originating in academia, it is nowadays commercially available and being manufactured by 3D Robotics.

The Pixhawk autopilot features two embedded ARM micro-controllers, the main running at 168 MHz and supports many connectivity options (such as UART, I2C, SPI). The on-board flight software is nowadays well maintained (although still not completely mature). This software is composed of a flight software-stack embedded in POSIX-like Real-Time Operating System (RTOS) called NuttX. One important aspect of the Pixhawk project is the tight integration with the ROS (Robot Operating System) middleware, a widely adopted system of robotics packages which was also used for the implementation of the VTnR method proposed in this thesis. The Pixhawk interfaces with ROS using a package named "MAVROS", which runs on the companion computer and allows to control the vehicle and obtain sensor readings using ROS standard communication interfaces.

4.2.3 Companion Computer

The companion computer is the hardware unit which is used to execute any high-level task which cannot run on the flight computer due to resource constraints (processing time, mainly) or for which execution on separate hardware is actually desired in order to not compromise the safety of low-level control. Examples of such tasks are the processing of complex sensors, such as a vision camera, for the purpose of high-level control, such as map-based localization and navigation. It is expected that these type of tasks will depend on pose estimates produced by the flight computer, however failure of high-level tasks should not imply the loss of control of the platform itself. Finally, the companion computer should be efficient in terms of its

weight- and energy-to-performance ratio. While the main energy drain of the aerial platform will come from the motion of the motors, the companion computer's size and weight should be adequate to the payload capabilities of the platform.

As a result of the development of highly integrated and efficient portable devices, general purpose computers which satisfy the aforementioned requirements are readily available as off-the-shelf. The great majority of these are based on type A ARM devices, which are 32 bit microprocessors capable of executing nowadays operating systems. On the other hand, while a varied offer of embedded computers exists, not many fulfill all requirements equally. ARM-based devices, do not generally offer the same processing power as traditional 32 bit processors such as those from major manufacturers as Intel or AMD. On the other hand, ARM devices are several times more power-efficient than the latter. Only recently, multi-core ARM processors in the range of GHz speeds have appeared as general purpose embedded computers, in small size and in affordable prices. While other medium-size or medium-priced options exist such as Intel Core M platforms, their size and power efficiency have not been up to par to other ARM-based devices. Thus, an ARM-based companion computer is, to this date, a very attractive platform.

One aspect of dealing with ARM devices is that not many software packages are yet readily optimized for such processors. While compilers can automatically introduce and employ SIMD instructions, it is still necessary to manually optimize code with the use of assembler code or compiler intrinsics in order to obtain sensible performance gains. While this can actually be true for any other processor, for embedded devices software execution is generally on the edge of real-time performance since the usual practice is to first develop and employ existing libraries for traditional CISC processors and only then port the software to the ARM device. This requires employing multi-platform optimizations or directly targeting the specific final platform to be used. This is probably the result of the relatively young age of these low-price and high-performance ARM devices, for which commonly used software libraries have just not caught up yet.

Arguably, one of the most known and successful embedded computers is the Raspberry Pi, developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. After selling about five to six million Raspberry Pi boards, in middle 2015, a second more powerful quad-core version, the Raspberry Pi 2, was released while still maintaining its 35 u$s price. Many software packages, including libraries and Linux distributions targeted this platform, considerably simplifying its use and adoption as a multi-purpose embedded hardware computer. However, while highly popular, other similar and more powerful computers have already been available from other manufacturers. Today, the Odroid line of computers from the HardKernel manufacturers offers several options, from quad-core to octa-core computers. These boards have been popular as companion computers in several mobile, mostly aerial, robot platforms. The performance of their quad-core computer is about N times higher than the Raspberry Pi 2, due to not only a faster clock speed but also a newer architecture version.

For the proposed aerial platform, the Odroid U3 was used as the companion computer (see Figure 4.7b on page 73). It features a quad-core 1.7 GHz Exynos 4412 Prime Cortex A9 processor, 2GB of RAM, and standard Ethernet and USB 2.0 connectors. A small I/O port is included, opening the possibility of accessing low-level protocols such as RS232, I2C and SPI. With a weight of only 48g and a size of 83x48mm, it is highly appropriate for small to medium mobile robot platforms.

4.2.4 Communication Links

While the purpose of the mobile platform is to execute all required tasks on-board the flight and companion computers, different communication links between a ground station and the aerial platform are available.

Radio-Control

First, a standard 2.4GHz radio link between the autopilot, by means of an eight-channel receiver, and a radio-control unit is available for low-level operation of the platform during various manual, semi-autonomous and full-autonomous control modes. This link exists for the main reason that not all flight phases are cur-

rently automated (such as the landing and take-off) since they are not the focus of the research presented in this thesis and manual control during these phases is considerably easier (by an experienced pilot) than robust automation. Moreover, for safety reasons the radio-control is always used as an override where the pilot can take full control at the flick of a switch, if needed. This radio link bypasses any high-level control and is thus a safer alternative than other forms of communication.

(a) Turnigy 9x Radio-Control and Receiver

(b) Turnigy 9x Radio-Control and Receiver

Figure 4.8: Communication Links

WiFi

A second communication link is present over standard WiFi, which allows remote access to the companion computer from a Laptop. The main purpose of this link is to launch the software to be executed on the companion computer and to initiate the logging of the data to be acquired during experiments. This data can be saved locally on the flight-computer and also be remotely viewed (particularly, the camera images) from a Laptop in order to obtain a first-person-view and to asses the information computed during flight.

A particular difficulty with this communication link is that standard WiFi works on the same 2.4 GHz band as the radio control. While this is not generally a problem (most wireless devices communicate on this band), the radio-control uses a specific anti-jamming technique known as Frequency Hoping Spread Spectrum or FHSS, which involves the transmission of the signals over a large number of rapidly changing frequencies covering the whole 2.4 GHz band. This has the effect that any communication taking place on a range of frequencies have little effect on the radio-control link since there is a high redundancy of data, even during a short period of time. On the other hand, while radio links such as WiFi use a similar strategy named Direct Sequence Spread Spectrum (DSSS), in this case the spread of the frequency is much smaller. Furthermore, in WiFi a fixed channel is used for a communication, centered around a given frequency, occupying only a 22 MHz band. As a result, the coexistence of 2.4 GHz WiFi and RC links becomes problematic, since FHSS ends up saturating the whole 2.4 GHz band, which WiFi is not able to cope with. When turning the radio-control transmitter on, the WiFi link becomes quickly unstable and becomes impossible to receive and transmit data.

Various solutions to this problem exist but several difficulties also appear. Radio-control transmitters using the 5GHz band are available but object penetration of the signal is considerably diminished increasing the chance of communication loss. WiFi on the 5 GHz band is also possible, but small WiFi dongles are

either not easily available or considerably more expensive than 2.4 GHz adapters. Finally, other communication links over lower frequencies such as 900 MHz or 433 MHz are available but they are generally of low throughput useful only for small volume of data such as for the transmission of telemetry or of high-level commands. Given that it is expected that the aerial platform should be capable of transmitting heavy data streams (such as live camera images), standard WiFi is required.

The chosen solution for this problem was to utilize a particular series of WiFi dongles, which are based on the Atheros AR9271 chipset, where some of the communication layers that are usually implemented in the hardware adapter are in this case expected to be solved by the operating system at the driver level. In particular, this feature opens up the possibility of extending the range of radio channels used for communication, since these chipsets actually supports a wide range of channels in the 2.3 to 2.7 GHz, 4.9 and 6.1 GHz ranges, but are simply not used by the Operating System. For this reason, a series of software modifications were introduced to the Linux kernel and the regulatory domain handling tools to expose these extra channels, particularly those in the 2.3-2.4GHz band. The regulatory domain handling tools, which set the allowed frequencies by country (based on their particular regulations) were also modified to include these extra channels, as allowed for the "CN" (China) domain set by default by the WiFi dongle. With these modifications it was the possible to establish WiFi communication using a pair of these USB dongles in the 2.3 GHz range, which falls outside of the 2.4GHz band where the radio-control performs frequency-hoping. Radio-control and WiFi links can then co-exist and and even WiFi interference from other devices is virtually eliminated (since the 2.3 GHz band is mostly unused).

On-board Computers Interface

The final communication link used in the LRSE quadcopter corresponds to the one interfacing flight and companion computers. This allows to control of the vehicle from high-level code based on high-level sensors (connected to the companion computer) and low-level pose estimates (obtained from the flight computer). This link is wired and based on the serial RS232 protocol, running at a high rate of 921600 bauds. For this communication the MAVLINK protocol is used. However, this low-level protocol is actually handled internally by the MAVROS package, with which the high-level computer actually communicates.

4.2.5 Main Sensors

While the aerial platform described in this chapter is designed to be a general purpose research platform, where different sensors can be mounted depending on the task at hand and the method to be evaluated through experiments, for this thesis a particular sensor configuration was employed and is here described.

Given the navigation method proposed in this thesis largely depends on vision, a machine-vision Firefly MV camera was mounted on the platform (see Figure 4.9b on page 77). This camera supplies images to the companion computer, which are processed in several stages. The standard resolution of the camera is 640x480 pixels at 30 frames per second, but it supports different configurations. The camera's lens is interchangeable, and supports both the standard CCTV lenses (CS-mount) and micro-lenses (M12-mount). More details about this sensor are presented in 5.1.4.

The second main sensor of the platform is the PX4FLOW optical-flow sensor module (see Figure 4.9a on page 77), which estimates ground relative-velocity and altitude. This sensor consists of a downward looking camera (imaging sensor and micro-lens), a three-axis gyroscope, a sonar rangefinder, and an embedded ARM micro-controller. The hardware module is programmed to process images from the camera in order to extract optical-flow. This information, is fused with the range measured by the sonar (to estimate ground-distance to the observed texture) and compensated from angular motions by measuring angular rates using the gyroscope. The flight computer employs the velocity and altitude information from the feed the pose estimation and allow for velocity-based control of the platform (sent by the companion computer or by the radio control, during manual override).

(a) The 3DR Pixhawk autopilot

(b) PointGrey's Firefly MV camera (no lens attached)

Figure 4.9: Main Sensors

4.3 Software Architecture

The software architecture of the LRSE quadcopter is composed of two main blocks (see Figure 4.10 on page 77): one running on-board the companion computer and one running in the autopilot.

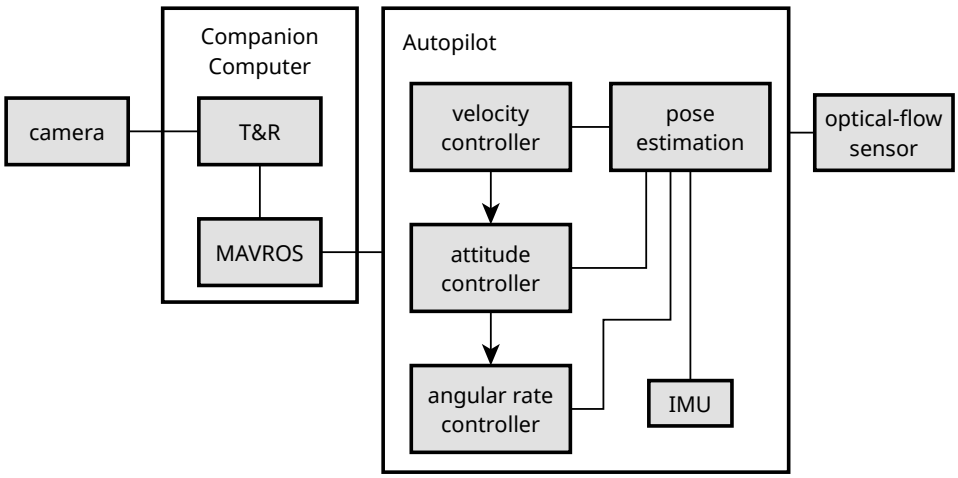


Figure 4.10: Nested control-loop architecture of the autopilot

The first software block depends on the task to be solved at hand, and in the context of this thesis it corresponds to the proposed VTnR method. The second block corresponds to the Pixhawk autopilot software stack, which consists mainly on a pose estimator and a series of nested controllers, which control the motors based on the pose estimates. As this chapter is focused on the platform itself, only the second block is described in the following sections.

4.3.1 Pose Estimation

The pose of the vehicle can be defined, in broad terms, as the joint estimate of its attitude and position. This estimate is used to feed the corresponding control loops in order to reach a desired setpoint. In reality, angular rates and velocities are also estimated due to the nesting of different controllers controlling also these variables (see section Section 4.3.2). Furthermore, while attitude and angular rates are normally estimated w.r.t. to the vehicle's local frame, the position and velocity can be estimated in both a local and global coordinate frames. The the local coordinate frame is defined with its origin placed in the vehicles initial pose and the global frame is defined in Earth's coordinates (latitude and longitude). In both cases, the NED convention is used, where X points North, Y points East and Z points down. This distinction is required since a vehicle may not necessarily have a sensor measuring the vehicle's pose in the global frame (e.g. a GPS) but only sensors measuring relative poses (when navigating indoors).

The currently employed strategy for pose estimation on the Pixhawk consists on the separate estimation of the attitude and of the position/velocity/acceleration of the vehicle, using an Inertial Filter. This approach presents various limitations (some of which were addressed in this thesis) inherent to the use of this filter. For this reason, the Pixhawk software community are currently migrating to a more robust Extended Kalman Filter estimation framework which would solve most of the issues encountered in this thesis. However, this migration is not yet complete at the time of writing, and further experiments using the new flight-stack are left as future work.

In the following section, the Inertial Filter is described for the case of position/velocity/acceleration, as it is of particular relevance for this thesis, since some improvements were here introduced.

Inertial Filter Estimation

The main idea behind an Inertial Filter estimator is to employ inertial sensors for a short-term prediction of position and velocity and then correct this prediction using the measured error with respect to sensors, which directly measure position and/or velocity.

In general, the prediction step of the filter updates a position x and velocity $\dot{x}$ estimates using the current acceleration $\ddot{x}$ as follows:

$$\begin{aligned} x &\leftarrow x + \dot{x}\Delta t + \ddot{x}\frac{1}{2}(\Delta t)^2 \\ \dot{x} &\leftarrow \qquad\qquad \ddot{x}\Delta t \end{aligned}$$

where $\ddot{x}$ can be obtained from an accelerometer sensor reading a as

$$\ddot{x} = a - b$$

with b being the accelerometer's bias estimate. While this value can normally be found off-line, it is usually estimated on-line since it is dependent on temperature and other factors, which can quickly throw off this type of estimation, where any error passes a double integration.

On the other hand, the correction step is defined in terms of the particular data obtained from an on-board sensor, where an error e is first computed between the sensed variable and its corresponding estimate (either position or velocity). Then, the estimate is linearly updated by this error, using a low-pass filter implemented an exponentially-weighted moving average,

$$x \leftarrow x + e\,\alpha\Delta t$$

where $0 < \alpha < 1$ is a smoothing factor, related to the frequency cutoff of the low-pass filter, for the case of a correction obtained by a position-based sensor. The correction for $\dot{x}$ is defined in the same way for a correction obtained by a velocity sensor. Note that, since position sensors are generally obtained with respect to global frames and are thus do not carry considerable accumulated error, for a position correction e, the velocity can also be corrected as:

$$\dot{x} \leftarrow \dot{x} + e\,\alpha^2\Delta t$$

.

For the case of the chosen FCU, this filter is used to estimate position, velocity and acceleration of the vehicle. Moreover, it is possible to integrate many different sensors which can be used to correct different variables and thus achieve different levels of precision. Finally, both local and global sensors can be used.

Sensor Fusion

The base sensors which are always present are the inertial measurement unit (IMU), composed of three-axis accelerometer and gyroscopes, a barometer and a magnetometer. In particular, since this thesis deals with GPS-denied navigation, no external positioning sensor (such as GPS or a motion-capture system) is included. However, an optical-flow sensor is connected to the system, providing horizontal velocity readings and also ground-altitude (measured from a sonar range-finder). This allows not only velocity-based control of the vehicle but also fairly precise position estimation and control (in a local world-frame).

For the position/velocity estimator, gyroscope readings are not employed. On the other hand, the accelerometer is used to estimate the vehicle's acceleration. Due to the principles behind the accelerometer's construction, this sensor measures not only accelerations imposed on the vehicle as a product of its actuators but also any other sensed external forces, most notably the Earth's gravity. Thus, the estimation of the accelerometer's bias values is not only required as a calibration procedure but also to compensate for acceleration readings obtained as a product of the gravity force. In order to estimate these biases, orientation w.r.t to the Earth's frame is assumed to be already estimated by the attitude estimator. This allows to separate the expected acceleration value of the Earth's gravity from the total forces sensed.

The on-board barometer is used by the estimation module to directly correct for the vehicle's altitude. However, this sensor can be problematic since any disturbance in the atmospheric pressure in the vehicle's surroundings causes unexpected drifts in its readings. Thus, while offering virtually unlimited range, for precise altitude estimation and, moreover, altitude control, this sensor is not ideal. Finally, for the case of position estimation in a local world-frame, the initial reading of the barometer (while the vehicle is sitting on the ground) is subtracted to obtain zero-altitude. However, any drifts during this reading can introduce an unwanted bias which is difficult to correct without any other sensor.

The values obtained by the optical-flow sensor are used to correct for horizontal velocity values. This implies that local position is only predicted (integrated from velocity) and never corrected. However, due to the precision of the readings of this sensor, the position can be quite accurate. While this sensor employs a sonar range-finder (required to properly scale visual motion into vehicle's velocity), the sonar can be used for altitude estimation and control (with respect to ground). However, the current implementation of the inertial filter does not using this sensor for this purpose. Furthermore, any discrepancy between the altitude estimate (obtained mainly from the barometer) and the sonar range finder is deemed to be caused by changing ground altitude (i.e. in the presence of ground obstacles such as stair steps). This implementation was found to be problematic and it was thus improved, as detailed in the following section.

Estimator Improvements

For altitude estimation, the current implementation of the Inertial Filter has a strong reliance on the barometer readings. Under GPS-based localization, this choice is reasonable since the inaccuracy of the barometer is still well below the high-variance and altitude readings obtained from standard GPS receivers. Thus, in this case, the barometer actually helps to correct altitude estimation. On the other hand, in the absence of GPS readings, altitude estimation relying purely on the barometer can be quite inaccurate. In particular, in GPS-denied navigation the vehicle is assumed to be close to ground and to obstacles, which even increases the requirement on precision altitude (and position) estimation and control.

To this end, in the scope of this thesis, the estimator was modified in order to rely mainly on sonar for altitude estimation and only use barometer when either the vehicle is outside of sonar range or sonar readings are missed for a considerable time. The sonar sensor present in the optical-flow board is the HRLV-

EZ4, which is capable of millimeter resolution in the $0.3m$ to $5.0m$ range (although usable maximum range is limited to $4m$ in practice). Given its great precision, this sensor is vastly superior for close-range altitude readings. In contrast, this sensor measures range to ground, which may not always be equivalent to altitude w.r.t. initial takeoff position, since the ground may have changing altitudes itself and even obstacles. However, for close-to-ground navigation, assuming ground-distance to be equivalent to ground-altitude is generally more robust than attempting to detect changing ground height using an Inertial Filter, since it is difficult to distinguish abrupt (or even gradual) ground height changes from momentary invalid readings. Other more advanced filters, such as the Kalman Filter, can handle this particularly well.

In broad terms, the main difference under this new altitude estimation scheme consists in taking the ground distance measured by the sonar and using the difference to the current altitude estimate to obtain an error value. To replace barometer-based estimation, both the accelerometer bias values and the altitude estimate itself are corrected based on this error. However, many difficulties arise when attempting a robust implementation. First, the vehicle may be outside of valid sonar range and thus no correction can be performed. While this may mean a problem only when going over maximum altitude, before takeoff the vehicle will also be outside of sonar range (unless the sensor mount-position and the platform height allow for a reading above the minimum of $0.3m$ even on ground, which is not the case for the LRSE quadcopter). Second, while the sonar is particularly precise, many sonar readings can be missed when flying over terrains such as grass, which disperse sound waves. Third, short invalid readings, which can be seen as spikes in a plot of the sensor readings over time, can also appear due to echoes (i.e. sound waves bouncing off of nearby surfaces). Finally, while a simple solution could be to simply switch to barometer altitude estimation for this usually short periods of time, the barometer drift w.r.t to current ground height greatly difficult this approach.

To overcome all these issues, low-pass filtered readings are maintained for both the sonar and barometer readings. Sonar-based altitude correction is only performed when the low-pass filtered sonar values are within an acceptable range of $0.35m$ and $3.5m$, where readings are deemed to be accurate enough. Using the actual sonar readings to decide whether the vehicle is in the valid range is not possible due to the aforementioned problems. When in range, not only altitude correction is performed, but the barometer offset is constantly updated as the difference between the low-pass filtered barometer reading (so that the high-frequency noise of the sensor does not throw off the compensation) and the current altitude estimate. In this way, when switching from sonar to barometer, at least for a small period of time, the barometer drift should not affect considerably the altitude estimation. The only problem in this case is that, while the vehicle is on the ground and has not yet taken off, or even after takeoff but before reaching the minimum altitude for sonar to be usable, no barometer correction is possible which guarantees both estimates would match. In any case, this only limits the possibility of autonomous takeoff, which is not a particularly hindering limitation for the purposes of this thesis.

Finally, while the main goal of these improvements is to obtain a more accurate altitude estimation, the velocity estimate obtained from the optical-flow sensor readings is expected to be also improved. This is due to the fact that the original implementation directly used an estimate of ground distance obtained from the difference between the altitude estimate (based only on barometer) and a ground height estimate (obtained from the difference between sonar and barometer altitude estimates). Under the present approach, since the altitude estimate is already guaranteed to be w.r.t to ground, optical-flow values can be directly scaled by this estimate and thus any drift, due to barometer errors, does not affect velocity corrections.

4.3.2 Low-level Controllers

As for the pose estimation problem, the strategy employed by the FCU is to separate attitude control (orientation and rotational velocity) from position/velocity control. This allows for different controllers to be implemented and chosen at run-time. The current type of controller used in both cases is the classical P.I.D. (Proportional, Integrative, Derivative) which is well known and widely applied for the case of aerial autonomous vehicles. While this controller is not provably stable in general (and, in practice, due to the inherent instability of a multicopter vehicle it is easy to fall within unstable control regions) it is easy to

understand and fairly easy to tune by inexperienced users. Given this is a type of linear controller and that the vehicle dynamics are highly non-linear, a nested control-loop approach is chosen where one controller "sits" on top of a lower-level controller, on multiple levels.

Velocity and Position Controllers

For velocity and position control, the position controller (P-term only) operates on top of the velocity controller (full PID), which in turn operates on vehicle acceleration by modifying vehicle's thrust vector direction and intensity (by means of the attitude controller). The vehicle's position, velocity and acceleration estimates are used as inputs for the control loops.

For position, a simple P-controller is used since it is assumed that a well-tuned full PID velocity control can already deal with difficulties such as wind-compensation and other external disturbances. While other more sophisticated non-linear controllers are more suitable for position/velocity control on aerial vehicles, for the purposes of this thesis, this kind of PID control suffices (considering the limitations described in 3.5.3). The task of exploring more advanced control strategies such as Motion Prediction Control (MPC) is left as future work.

Controller Improvements

Similarly to the improvements introduced in the position/velocity estimator, the position/velocity controller was also modified. In the original flight code, under velocity-control mode, velocity was not actually directly controlled but by means of a position-controller using a carrot-follower strategy. In this way, a velocity setpoint is interpreted not as the target velocity of the platform but as the rate of motion of a position setpoint (i.e. the "carrot") which is followed by internally controlling velocity. This scheme has the benefit that, under zero-velocity set-points, the vehicle will actually attempt to hold its position, which would not necessarily occur when trying to remain at zero-velocity, due to drift, for example due to wind or any other disturbance. The downside, is that a velocity-setpoint will move the position setpoint to a virtually invisible location which will introduce considerable response delays when requesting the vehicle to stop (since the position setpoint could still be far ahead from the vehicle). To achieve actual velocity control, the internal position/velocity control was modified by cleanly decoupling position from velocity control. This allowed to much greater control of the vehicle, which was required for controlled autonomous flight when validating the proposed VTnR approach.

Capítulo 4

Plataforma aérea experimental (resumen)

En este capítulo se describen el diseño y construcción de una plataforma aérea experimental, el cuadracóptero LRSE, que fue utilizado para la realización de diversos experimentos bajo el método de navegación propuesto.

La construcción del cuadracóptero LRSE fue motivada por la inexistencia de una plataforma comercial de bajo costo que estuviera orientada a fines de investigación. Esto implica la capacidad de llevar a bordo tanto diversos sensores (dependiendo de la tarea a resolver) como de unidades de procesamiento. Debido a que uno de los objetivos de esta tesis es el de desarrollar un método de navegación que pueda ser ejecutado a bordo de un vehículo aéreo, estas capacidades resultan necesarias.

El cuadracóptero esta compuesto por diversos componentes, de los cuales se pueden distinguir los componentes básicos tales como la estructura, motores, hélices, batería y controladores de velocidad, de otros más complejos como la electrónica a bordo y los sensores de alto nivel. Con respecto a la electrónica, se distinguen dos unidades de hardware: el autopiloto y la computadora de procesamiento. El autopiloto resuelve todas las tareas relacionadas con la estabilización, control y estimación del estado de la plataforma, mientras que la computadora es la que es utilizada para ejecutar algoritmos tales como el propuesto en esta tesis. Entre los sensores utilizados en la plataforma se encuentran principalmente un sensor de flujo óptico (que permite el control de la plataforma mediante comandos de velocidad) y una cámara convencional que es la que se utiliza como entrada principal al método de navegación.

Además del desarrollo físico de esta plataforma, el trabajo incluyó la introducción de mejoras considerables respecto de la estimación y control tanto de la velocidad de la plataforma como de su altitud. Esto fue necesario debido a problemas presentes en el código correspondiente.

Chapter 5

Results

This chapter presents the results obtained from experimental analysis of the proposed VTnR method. Various aspects are considered such as execution times on different hardware platforms, localization precision, navigation robustness, among others. It should be noted that experiments with early versions of the approach presented in this thesis can be found in [41, 42] and are not completely included here.

While the main goal is to perform on-line experiments under realistic conditions to asses the expected performance in the real-world, in order to obtain detailed results and repeatable experiments, in some cases execution of the method over pre-recorded datasets or using simulated platforms was performed. Moreover, due to the unavailability of suitable aerial-robot datasets, third-party sensor data obtained with ground-robots was used.

All relevant parameters of the proposed method are described in Figure A.1 on page 138.

5.1 Hardware and Software

In this section, the robotic platforms, their sensors and processing platforms employed for on-line and off-line experiments are described. Also, implementation details of the proposed VTnR are presented.

5.1.1 LRSE quadcopter

The experimental aerial robot platform described in Chapter 4, the LRSE quadcopter, was used for performing experiments with the VTnR approach, executing in real-time and on-board the robot. The platform was manually controlled during takeoff and landing, while the teach and repeat phases were performed with the VTnR controlling the robot (in the teach phase, only heading and altitude are controlled by the user).

For the purpose of establishing ground-truth pose information, in some experiments a cube with special visual patterns printed on its sides was attached to the robot (see Figure 4.1 on page 67). By filming the robot with an external camera, using the AprilTag[52] augmented-reality tracking system, the position of the robot can be recovered. A GoPro Hero4 camera was used for this purpose, attached to a tripod and filming in 4K resolution (3840×2160 px) at the maximum frame-rate of 15 fps. The GoPro camera has a very wide lens which allowed to film an entire $10 \times 15\,\mathrm{m}$ area where some of the experiments were performed.

5.1.2 Pioneer P3-AT

The Pioneer P3-AT is a commercial ground-robot manufactured by AdeptRobotics, which consists in a four-wheel differential-drive base and features an internal controller which computes wheel odometry (from

wheel encoders) and allows control by an external computer by means of velocity commands. The robot can be equipped with one to three lead-acid batteries, giving a total of about two hours of autonomy.

The mobile base was extended with an aluminum frame on which different sensors or computers can be mounted for performing experiments (see Figure 5.2a on page 90). A standard Laptop was used as processing platform (described in the section 5.1.3) and a PointGrey Firefly MV camera was used for acquiring images (described in 5.1.4), running at a resolution of 640×480 at 30 fps. The Laptop was connected to the embedded robot controller (via a USB to serial cable converter) for the purpose of sending velocity commands to the robot while receiving odometric information.

In this thesis, this platform was used to acquire a dataset in a long indoor corridor in the second floor of the FCEyN-UBA Pabellon I building. In this case, for the purpose of estimate localization precision of the proposed method, this robot was also equipped with Sick Tim310 2D laser range-finder. This dataset is described in 5.3.1.

5.1.3 Processing Units

For experimental analysis of the VTnR method using pre-recorded datasets or live-data from a robotics simulator, a standard Laptop was used. This computer has an Intel Corei5 processor with 4GB of RAM. For execution on-board the LRSE quadcopter, the Odroid-U3 computer described in Chapter 4 was used.

5.1.4 Firefly MV Camera

For experiments performed using the Pioneer P3-AT robot and the LRSE quadcopter platform, the Firefly MV camera was used. This camera is very small ($37\,\mathrm{g}$ of weight and $44 \times 34 \times 24.4\,\mathrm{mm}$ in size) allowing to be used even on payload-limited platforms (such as the LRSE quadcopter). It has a maximum resolution of 640×480 px, with a possible frame between 30 fps and 60 fps. It has a standard USB connection and uses the `IEEE1394` communication protocol (originally used over FireWire connections) allowing greater control over internal settings in contrast to standard USB cameras. In all cases, auto-exposure was left enabled (letting the camera control settings to obtain a constant level of light) but shutter-speed was fixed for the purpose of avoiding image-blur.

The camera also features a standard CS lens-mount (found in typical CCTV lenses) allowing to choose the appropriate lens for the task. The lens for experiments was a $3.3 - 8\,\mathrm{mm}/f1.4$ zoom lens, set at around $5\,\mathrm{mm}$ and near maximum diaphragm aperture (allowing the greatest input of light). For the quadcopter platform the possibility of a smaller lens with fixed focal-length was considered (such as the M12-type lenses), since the zoom lens is relatively large and fixed focal-length lenses are usually sharper (particularly important when using lower resolutions). However, a lens with the appropriate focal-length and also including the required IR-cut filter was not available.

5.1.5 Method Implementation Details

One of the main goals of this thesis is not only to propose a novel VTnR navigation method for aerial robots, but also to obtain an implementation which is able to run efficiently on an embedded platform (such as those found on-board aerial robots). Moreover, it should allow to be easily tested by other researchers and thus be flexible enough to be used on different robot platforms and with different sensor configurations. To satisfy both of these goals, an open-source implementation of the proposed method was developed using established software-libraries and frameworks.

The implementation was programmed using the C/C++ programming language. For all visual-processing tasks, the OpenCV Computer Vision library was used. This library is well optimized for a number of platforms, including the case ARM-based embedded computers. In particular, feature extraction and matching are broadly parallelized when a multi-core processor is used and more finely using SIMD (Single Instruction

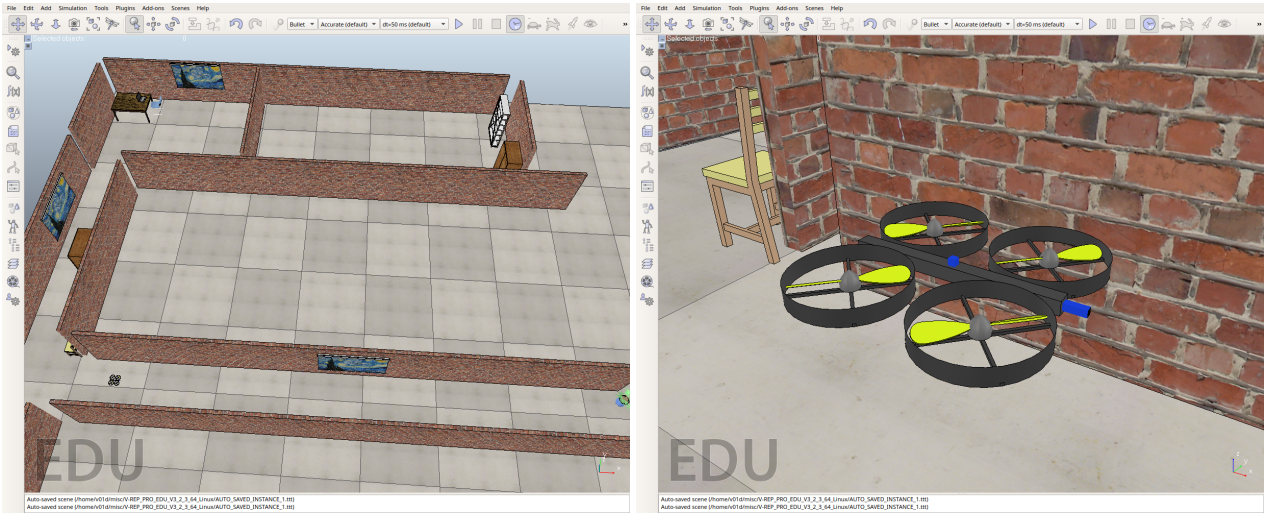
Multiple Data) instructions.

The Robot Operating System (ROS) framework[53] was used as the basis of the complete implementation. ROS is both a set of software packages and a distributed computation and communication framework, which targets re-usability and standardized interfaces to individual tasks. ROS follows the *message passing* design, where different units of computations (*nodes*) send messages to each other (e.g. images or other sensor-readings). This allows, for example, to implement a ROS node which processes images, independently of where this images come from, be it directly from the camera driver node or from a pre-recorded dataset.

In terms of performance, since ROS nodes actually run as independent processes, for the VTnR method implementation *nodelets* were used instead, which correspond to independent units of code that are loaded and run in the same executable. As a consequence, communication overhead is considerably diminished (no serialization or message copying is required). Furthermore, nodelets can process messages in parallel using *threads*. Each of the modules presented in Chapter 3 (localization, map-building, navigation and path-finding) were isolated in an individual nodelet. This allowed to parallelize the computation of different independent tasks. In particular, the update-step of the MCL-based localization module runs independently of the navigation module. While the latter module requires a recent pose estimate to function, the dead-reckoning based prediction-step of the localizer (which runs independently of the update-step, since no image-processing is involved) allows to successfully perform path-following even if the update-step of the localizer is considerably slower.

5.2 V-Rep Simulator

For the purpose of analyzing the closed-loop performance of the VTnR method and for obtaining greater control over experimental conditions, the mobile robotics simulator V-Rep, by Coppelia Robotics, was used. V-Rep provides a full-physical simulation, a series of simulated sensors which can be used to acquire data and several types of mobile robots. For experiments, an indoor environment composed of hallways with various objects (see Figure 5.1a on page 87) with a simulated quadcopter-style robot were simulated (see Figure 5.1b on page 87).



(a) Simulation environment **(b)** Simulated quadcopter robot

Figure 5.1: V-Rep Simulator

The aerial robot responds to velocity commands, while internally performing lower level control in the

same manner as the real quadcopter used for other experiments: to follow a velocity setpoint, a first PID loop controls the orientation of the vehicle, while a second PID loop controls the angular rates of the platform by operating on simulated rotors. As full aerodynamics are not simulated, each motor is assumed to simply generate a force proportional to the thrust setpoint.

A camera with a field-of-view of $60°$ and 320×240 px resolution was simulated. Odometric motion estimation (as would be obtained by a downward looking optical-flow sensor) and orientation (as would be estimated by the autopilot based on inertial sensors) are also provided by the simulator. Ground-truth information of the actual robot pose is also available for posterior experimental analysis. It should be noted that V-Rep does not add any noise to odometric and orientation sensor information, however a simulated camera with low-resolution represents a useful testing case.

Finally, the simulator also provides a ROS interface which allows to transparently execute the VTnR method without any change to the implementation.

5.3 Datasets

A series of datasets were used for the purpose of analyzing localization precision of the VTnR method. To do so, datasets where the same path was traversed multiple times were used, where one pass was used for building a map and the remaining to test localization precision, based on ground-truth pose information. Given the unavailability of datasets of this kind for the case of aerial-robots, the chosen datasets were recorded using ground-robots. However, the three employed datasets, vary considerably in terms of challenging illumination conditions, uneven terrain and structured and unstructured environments.

Since the proposed method is not metric, the estimated pose cannot be directly compared to ground-truth pose data. However, what is important in terms of localization precision is to obtain the closest pose over the learned map, i.e. that which minimizes longitudinal error. Thus, to measure localization precision the relative transform between the current robot pose and the one corresponding to the estimated location over the map is computed. In particular, the longitudinal translation component of this transformation is extracted, which corresponds to the localization error of the method. The lateral component in this case is not significant since it is not expected that the robot will traverse the exact same path in all passes.

Since in the proposed method localization simply produces a pose described with a segment number and distance, the process of obtaining this relative transform requires that, during the teach phase, a correspondence is established between a pose in the appearance-based map and a pose in real-world units. To do so, for each node in the map, the current time-stamp is recorded. This allows to, given a segment number and along-segment distance, to obtain an interpolated time-stamp, with which the absolute-pose can be retrieved from the corresponding recording of the teach pass recording. However, not all datasets contain absolute pose information, such as that typically obtained from an external localization system. The process used to obtain relative pose error in these cases is described in the following sections for each corresponding dataset.

5.3.1 Indoor FCEyN Level 2 Dataset

This dataset was recorded using a the Pioneer P3-AT mobile robot described in 5.1.2, over a corridor of the FCEyN-UBA Pabellón I building (Figure 5.2a on page 90), of approximately $100\,\mathrm{m}$ long. The robot was manually driven along the corridor performing two passes over the hallway, while sensor data was acquired. During the recordings, faculty members occasionally passed by the robot, resulting in occlusions and appearance-differences between the teach and repeat phases. A floor-plane with the path traversed by the robot is presented in Figure 5.2b on page 90.

Due to the impossibility in an indoor setting of using an external localization system such as GPS, for the purpose of establishing relative pose information among the two passes, besides camera images and odometric information, the readings of a laser range-finder were also recorded. The relative transform

between a pair of readings corresponding to the teach and repeat runs was obtained by *scan-matching*. This process consists of comparing the two point-clouds obtained by the range-finder mounted on the robot, from two different robot poses. Using the Iterative Closest Point (ICP) algorithm (the libpointmatcher[54] implementation was used), a rigid transformation is computed, which is the inverse of the relative transform between the robot poses. Thus, instead of comparing absolute poses as described in the previous section, in this case laser readings were used to obtain the same relative pose error information.

5.3.2 Indoor QUT Level 7 S-Block Dataset

This dataset[55] was acquired at the seventh-floor of the S-Block Queensland University of Technology (QUT) Gardens Point Campus in Australia using the Guiabot platform (see Figure 5.3a on page 91), which features various sensors such as a stereo-vision pair of cameras, a Microsoft Kinect RGB-D sensor, a laser range-finder and odometric information of the differential-drive robot base.

The datasets consists of a long traverse of the office halls (see floor-plan Figure 5.3b on page 91 and examples images in Figure 5.3c on page 91). While the dataset was recorded in one pass, three distinguishable closed-loop tours made by the robot were used to run three independent teach phases. The remaining passes over each loop were then used for the repeat phase, where localization precision was measured.

The on-board laser range-finder was used to to build a metric map of the environment using the GMapping SLAM solution[56]. With the built map, the laser was then used to localize the robot for the entire dataset using a laser-based Adaptive Monte Carlo Localization method[57]. In this way, ground-truth information is available for this dataset in the form of absolute poses with respect to the built map.

In contrast to the FCEyN Level 7 dataset, this dataset is much longer. On the other hand, it features sharp images captured by the on-board cameras and the motion of the platform is very smooth. This dataset thus allows to establish a baseline for the localization precision, under mostly benign conditions.

5.3.3 Outdoor SFU Mountain Dataset

The third dataset[58] used for experiments corresponds to outdoor recordings of two mountainous trails (part A and part B, see see Figure 5.4b on page 92), in four different conditions: dry and sunny, wet (after rain), at dusk and at night (see Figure 5.4c on page 92, Figure 5.4d on page 92 and Figure 5.4e on page 92). This dataset is considerably challenging due to extreme changes across recordings and the continuously changing illumination. Furthermore, the forward-looking cameras, are only sharp near the center of the image, where mostly distant objects are seen. Finally, while traversing the mountainous terrain the robot platform exhibits considerable roll and pitch motions. This is another important difference with respect to the two previously described datasets recorded in structured indoor scenarios.

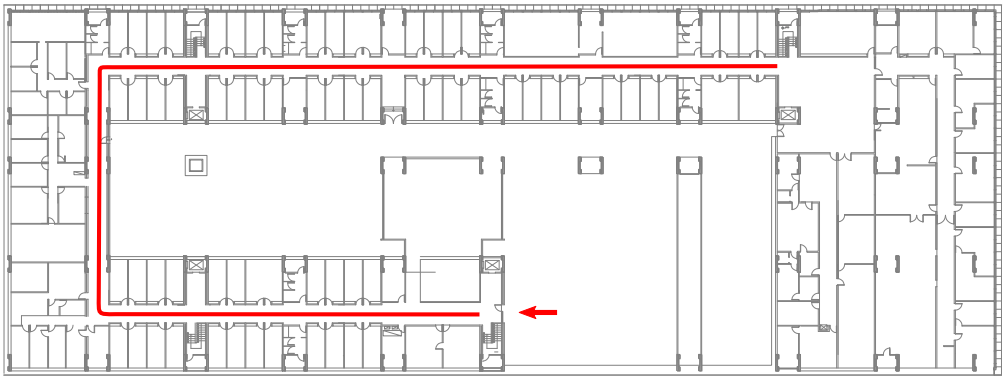
From the complete dataset, two teach phases were run for the "dusk" pass of part A and B, respectively. For part A, two repeat phases were performed with "dry" and "wet" passes, while for part B, only one repeat phase was run with the "wet" pass, since the "dry" recording was considerably different from the other two passes in terms of illumination and overall appearance.

The robot platform (see Figure 5.4a on page 92) is a ClearPath Husky four-wheel drive robot. The sensor payload includes two color foward-looking cameras (in stereo configuration), and four monochrome cameras pointing to the sides, to the back and up. The robot also includes a consumer-grade Garmin GPS sensor and an UM6 inertial measurement unit (IMU) with absolute yaw measurement from the included magnetometer.

Due to the inaccuracy of the available GPS (in the order of several meters), relative pose information was obtained based on the available "place matches" distributed with the dataset. This data consists of manually built time-stamp correspondences between different recordings, for poses approximately every $10\,\mathrm{m}$ of the trajectories. To obtain relative transformations between recordings, for every pair of successive entries for a given pass in the "place matches" database, the current robot location is obtained by interpolating the

corresponding time-stamps. Then, an interpolated time-stamp is built in the same way for the pose obtained as a result of localization, for the entries corresponding to the teach pass recording. Then, the interpolated time-stamps are used to find interpolated distances between these successive entries. Finally, the difference between these distances is used as an estimate of the longitudinal error.

(a) Pioneer P3-AT during dataset recording

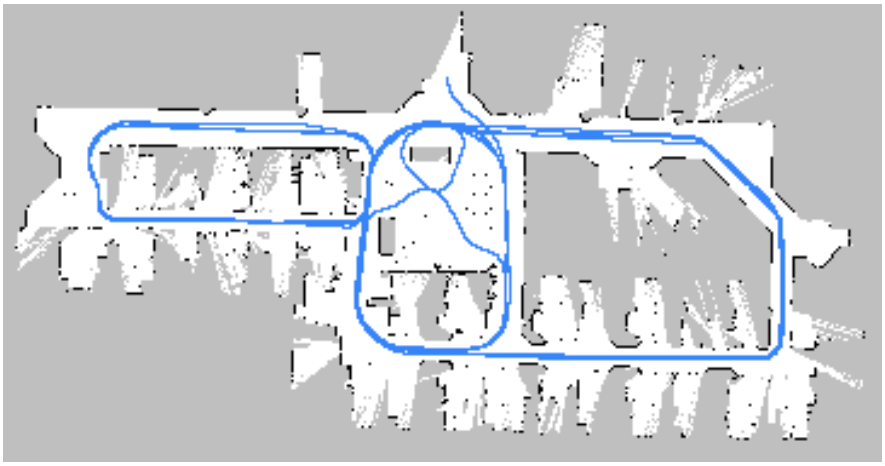


(b) Floor-plan and traversed path

(c) Example images from dataset

Figure 5.2: FCEyN Level 2 Dataset

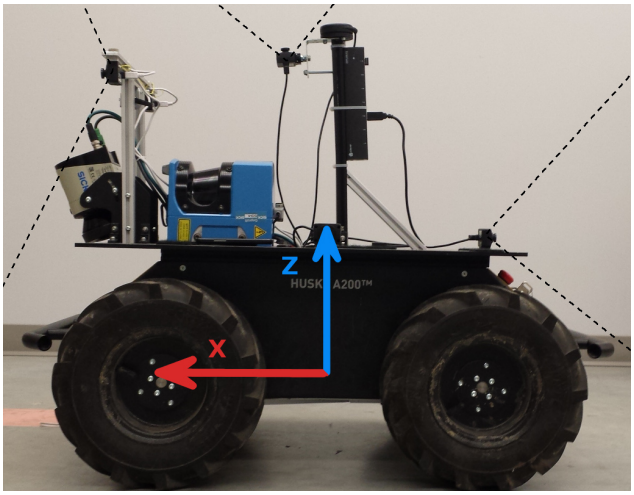
(a) Guiabot platform used for dataset recording



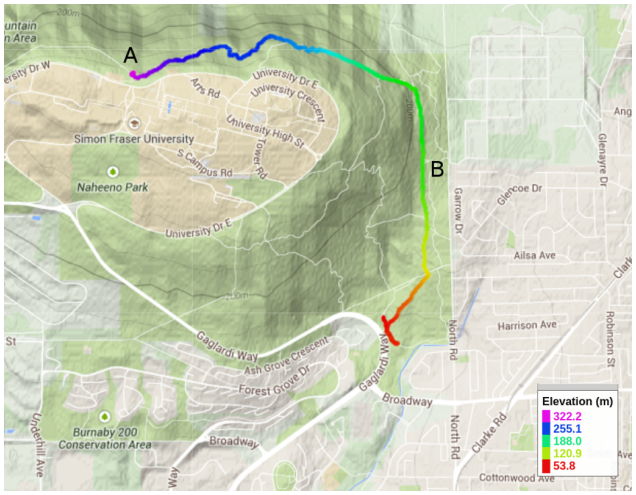
(b) Map built from laser range-finder and traversed path

(c) Example images from dataset

Figure 5.3: QUT Level 7 Dataset



(a) Husky platform used for dataset recording



(b) Path traversed by the robot for part A and B

(c) Images from part A (dry, wet, dusk and night)

(d) Images from part B (dry, wet, dusk and night)

(e) Images from part B (dry, wet, dusk and night), covered forest section

Figure 5.4: SFU Mountain Dataset

5.4 Descriptor's Rotational Invariance

As previously mentioned in 3.2.2, the feature detection and description algorithms used to obtain visual information in the VTnR approach correspond to the GFTT and BRIEF algorithms, respectively. Moreover, since rotational invariance is important for the case of aerial robot navigation (where camera roll cannot be disregarded), a strategy for adding rotational invariance to the BRIEF descriptor was proposed, based on the knowledge of the absolute roll angle of the camera (based on the roll angle estimate of the vehicle itself).

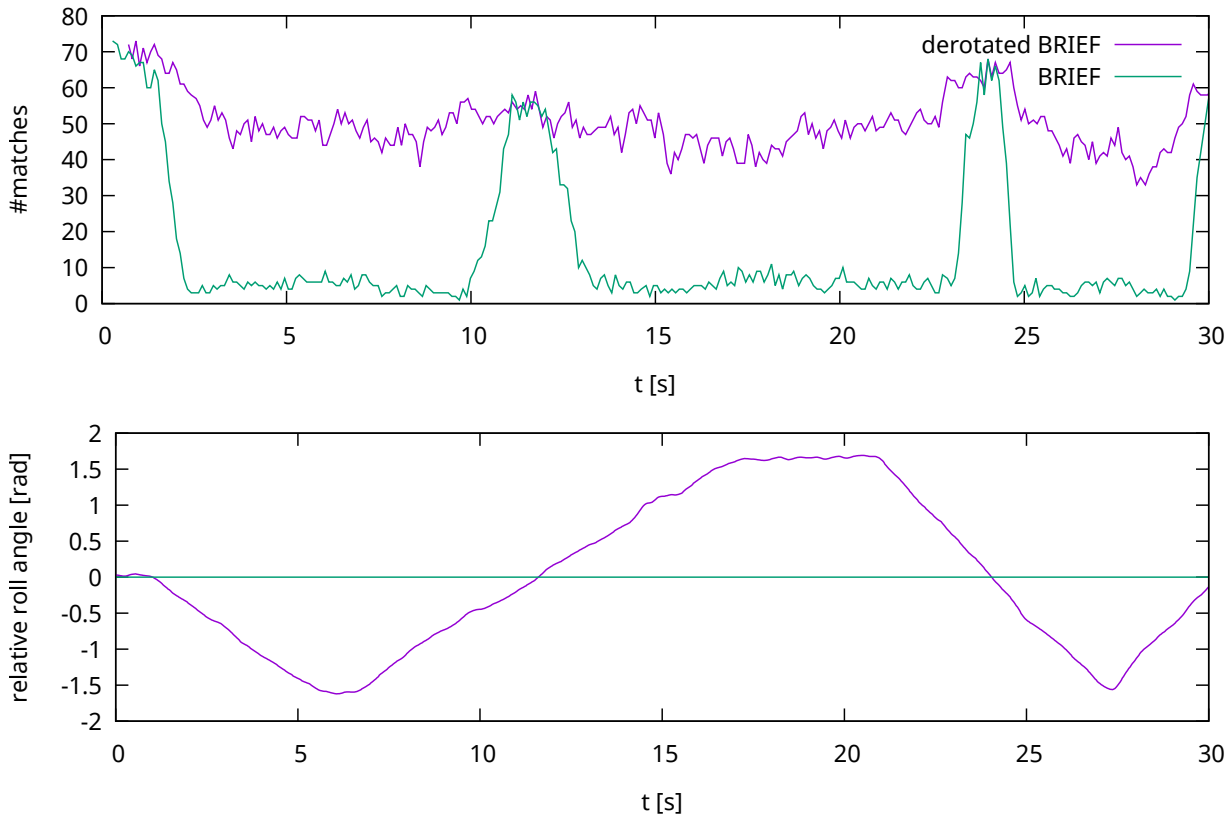


Figure 5.5: Feature rotational invariance experiment: number of feature matches and roll angle over time, comparing standard and derotated BRIEF descriptor

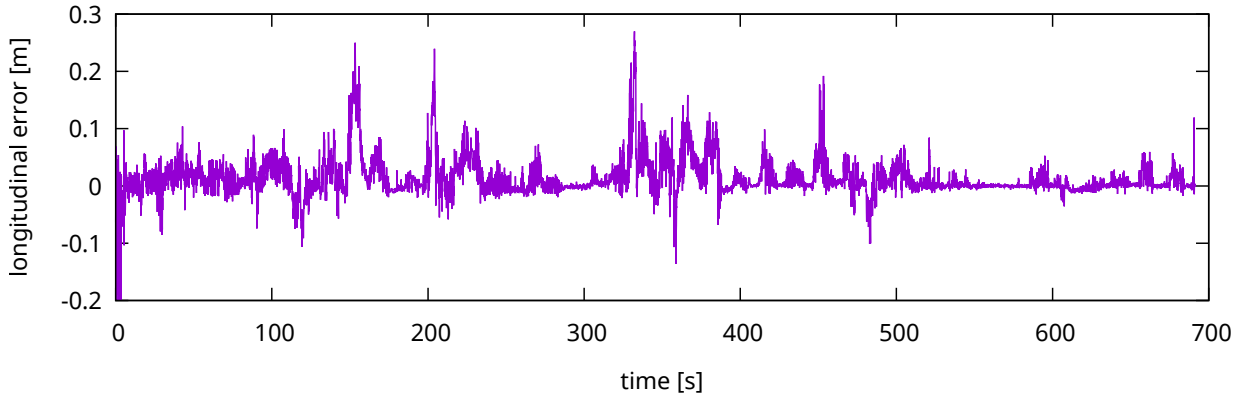
To analyze the effectiveness of this strategy, a small dataset was acquired with the LRSE quadcopter in hand, performing rolling motions in both directions covering the full $[-180°, 180°]$ range (with only slight unintended yaw and pitch motions). In the observed scene, all objects were mostly at the same distance. The initial image of this dataset is set as the reference and the following images are used to obtain feature-matches and the corresponding azimuth and elevation difference histograms, w.r.t. the reference. Features where detected using the GFTT algorithm, and described using the standard BRIEF algorithm, with and without derotation of the based on the camera's roll angle (as obtained from the autopilot).

In Figure 5.5 on page 93 the number of matched features over time is presented along the estimated camera roll angle. With the standard descriptor, the number of matches quickly drops after about $\pm 10°$ of roll rotation, eventually reaching minimum values for roll angles of $\pm 30°$. On the other hand, the derotated BRIEF descriptor is able to match features between 40 and 70 features, independently of the roll angle, which is the expected result of the approach.

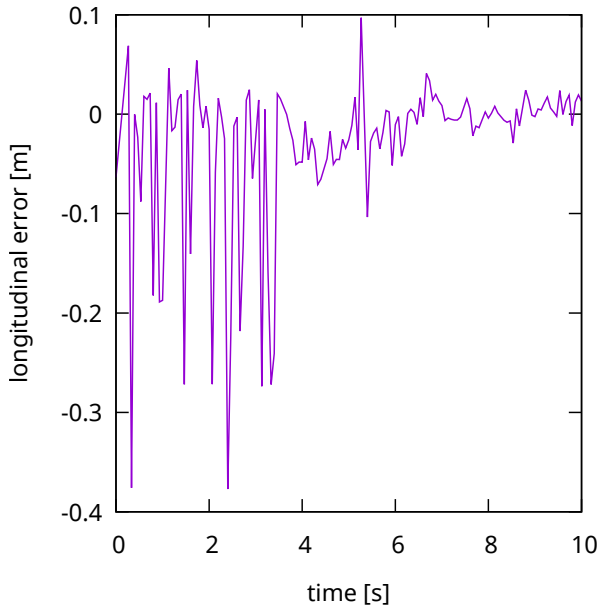
5.5 Robust Localization Experiments

In order to asses the precision of the appearance-based localization component of the proposed VTnR approach, the longitudinal error with respect to the estimated robot location over the map was measured by executing the method over the aforementioned datasets, as described in Section 5.3. This error is presented as a function of time, for each execution of the localization method. Since this error fluctuates along each pass, to provide a more concise analysis, the distribution of the longitudinal error as a relative frequency histogram is also presented.

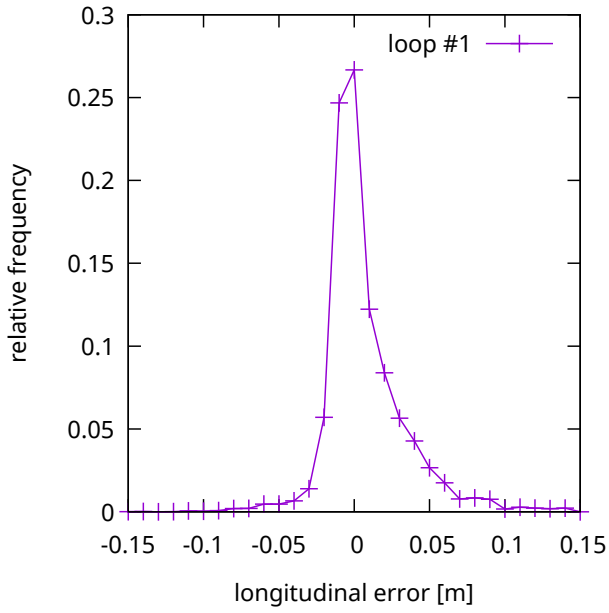
Furthermore, when analyzing the time-based error plots, it should be considered that MCL generally requires a certain number of iterations before converging to a good estimate, since initially the particles are randomly dispersed over the entire map. Thus, it is interesting to also note how long it takes for a good initial estimate to be obtained, i.e. one with low-enough longitudinal error which would allow for safe navigation.



(a) Longitudinal error as a function of time



(b) Initial convergence of the localization method



(c) Histogram of longitudinal errors

Figure 5.6: FCEyN Level 2 localization experiment

5.5.1 FCEyN Level 2 Dataset

For the FCEyN Level 2 dataset, a teach pass was performed with the Pioneer P3-AT robot, following the trajectory presented in Figure 5.2b on page 90. For the second run, the localization module was enabled and the estimated robot pose was used to determine a laser reading to which compare the current reading, for the purpose of obtaining relative pose information w.r.t. to the reference, as outlined in 5.3.1.

The results of this experiment are presented in Figure 5.6a on page 94 as longitudinal error values as a

function of time and in Figure 5.6c on page 94 in the form of a relative-frequency histogram. In Figure 5.6b on page 94, the initial seconds of the localization error is shown, to visualize the speed of convergence.

Analyzing the obtained results, it can be seen that the localization error is generally quite small, in the order of $0.1\,\mathrm{m}$ in most cases and reaching a maximum of $\sim 0.25\,\mathrm{m}$ in some situations. It can be also seen that even though localization already started with a low error, it quickly converges to a better estimate after around $4\,\mathrm{s}$.

5.5.2 QUT Level 7 Dataset

For this experiment, three separate closed-loop sections of the complete trajectory comprising the Level 7 dataset were used to learn three different paths. Since the dataset repeats each of these loops a total of three times, localization was performed over the paths three times, including the initial pass used to build the map. The purpose of also including this pass is to establish a baseline where any encountered localization errors will consist of a lower bound for errors expected in the remaining passes.

Together with the error plots, the measured quality (local density) of the estimated pose is presented along the threshold used to determine if the estimate is valid or not for navigation. Finally, in order to allow for correct interpretation of the localization errors, the segment identifier corresponding to the estimated robot pose is also presented. Since segments are sequentially number, this number is expected to grow linearly (robot speed is mostly constant) and only to reset when the robot starts the loop again.

Observing the obtained results (figures 5.7 to 5.10), it can be seen that localization errors are generally within $0.3\,\mathrm{m}$ for all cases. Furthermore, the behavior across the repetitions of each path is consistent. When analyzing the plots, peaks of about $1\,\mathrm{m}$ maximum errors can be seen at very distinct points in time. However, when observing the segment identifier in each case, it can be seen that these errors correspond to the case of the robot finishing the path and then starting a new pass. Since the start and ending of the teach pass were manually signaled, the map does not close perfectly and thus there exists an unmapped region which produces a localization failure and later triggers a global re-localization (requiring about $1\,\mathrm{s}$ to converge back). The speed of convergence at the beginning of the first pass is around $30\,\mathrm{s}$ in two of the paths, and virtually zero for the other one. In any case, for these slower initializations, the initial error is only about $1.5\,\mathrm{m}$, which is then reduced to a value below $0.2\,\mathrm{m}$. Finally, when the single-hypothesis quality is observed in all cases, it can be seen that it is effectively below the acceptance threshold in these situations (and above everywhere else), signaling that the localization system correctly signals the estimated as being invalid (which would avoid navigation failure).

5.5.3 SFU Mountain Dataset

The third localization experiment was performed using the SFU Mountain Dataset, a highly challenging scenario for appearance-based localization. Both part A and B were used, using the "dusk" variant for the teach phase in both parts. Then, the "dry" and "wet" variants of part were used to analyze localization quality against the corresponding learned paths. The night variant of each trail was simply dismissed since it was considerably different from the other cases. Moreover, for part A, the "dry" version was not suitable for localization due to the considerable changes in illumination and is was also disregarded. This was not a problem for part A since this parts features mostly tall trees and lighting is more consistent (although still challenging in itself).

Localization errors as a function of time are presented in Figure 5.11 on page 100 ("dry" variant of part B), in Figure 5.12 on page 100 ("wet" variant of part B) and in Figure 5.13 on page 101 ("wet" variant of part A). The histogram of localization errors for all trails is presented in Figure 5.14 on page 101. It should be noted that the process used to obtain longitudinal error values (described in 5.3.3) is prone to errors and thus values precise values cannot be expected.

In this case, the overall obtained results are considerably worse than with the previous datasets. However, this is expected given the challenging conditions previously mentioned. For the "wet" pass of trail A,

error values below $2\,\mathrm{m}$ were obtained in general, reaching about $5\,\mathrm{m}$ in certain cases. For the case of part B, errors are mostly around $5\,\mathrm{m}$ with peaks of $10\,\mathrm{m}$ in several cases. While the majority of the trail followed in this part corresponds to open canopy, at around the $1200\,\mathrm{s}$ mark a highly-vegetated was traversed, which was very dark in the "dusk" variant (Figure 5.4e on page 92). For this reason, localization completely fails for both variants in this section.

While the results in this case are not as promising, this analysis serves to consider the limitations of the method.

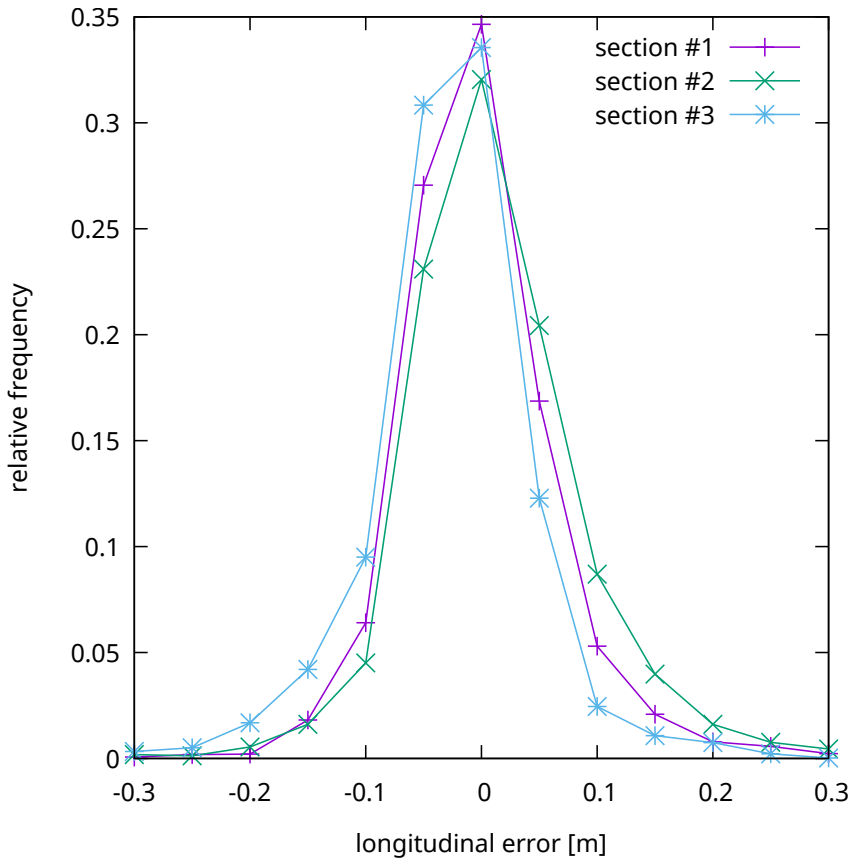


Figure 5.7: Level 7 Dataset Histogram of longitudinal errors, for the three sections used for experiments

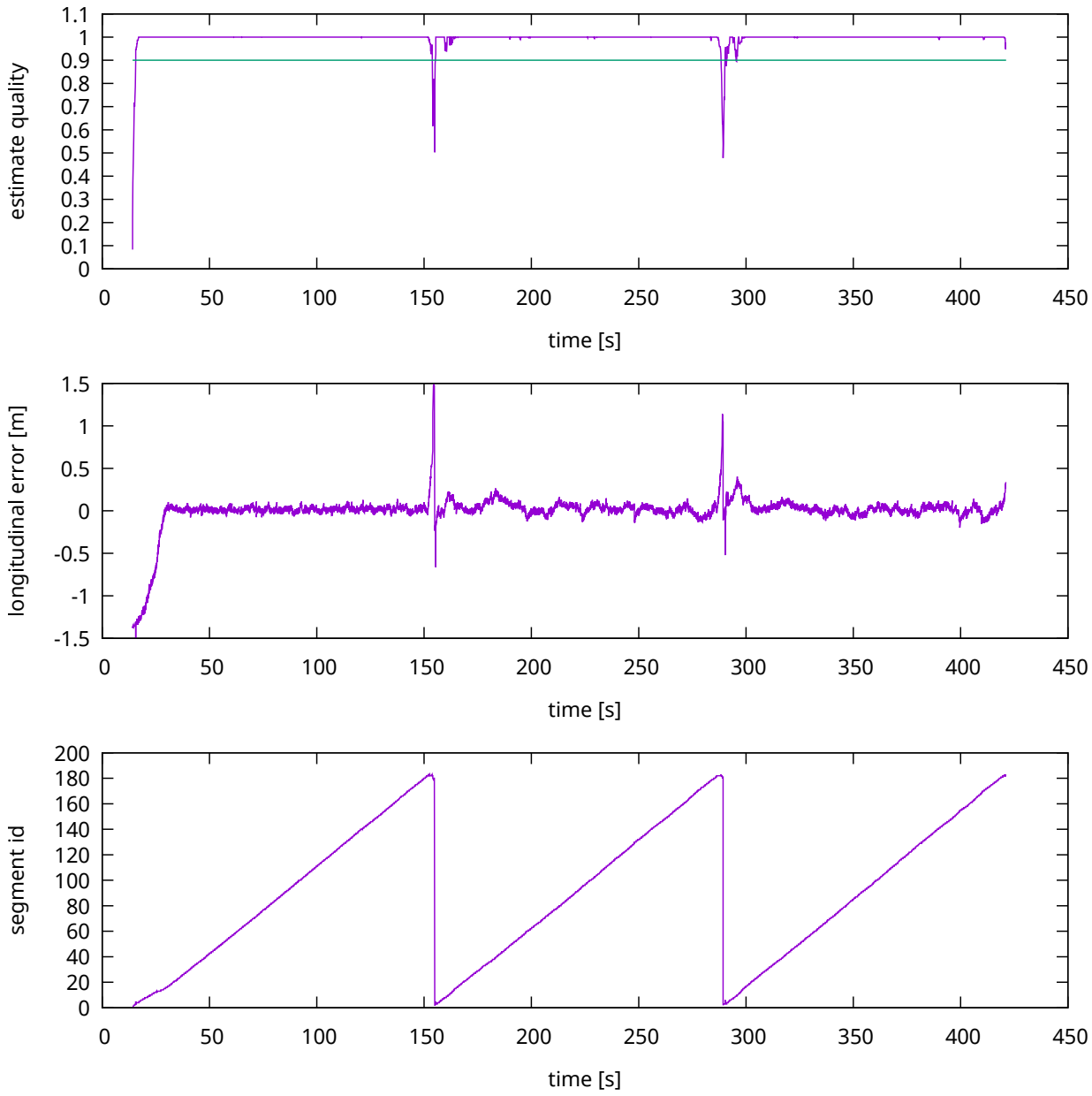


Figure 5.8: Longitudinal error as a function of time, for Section #1 of Level 7 Dataset

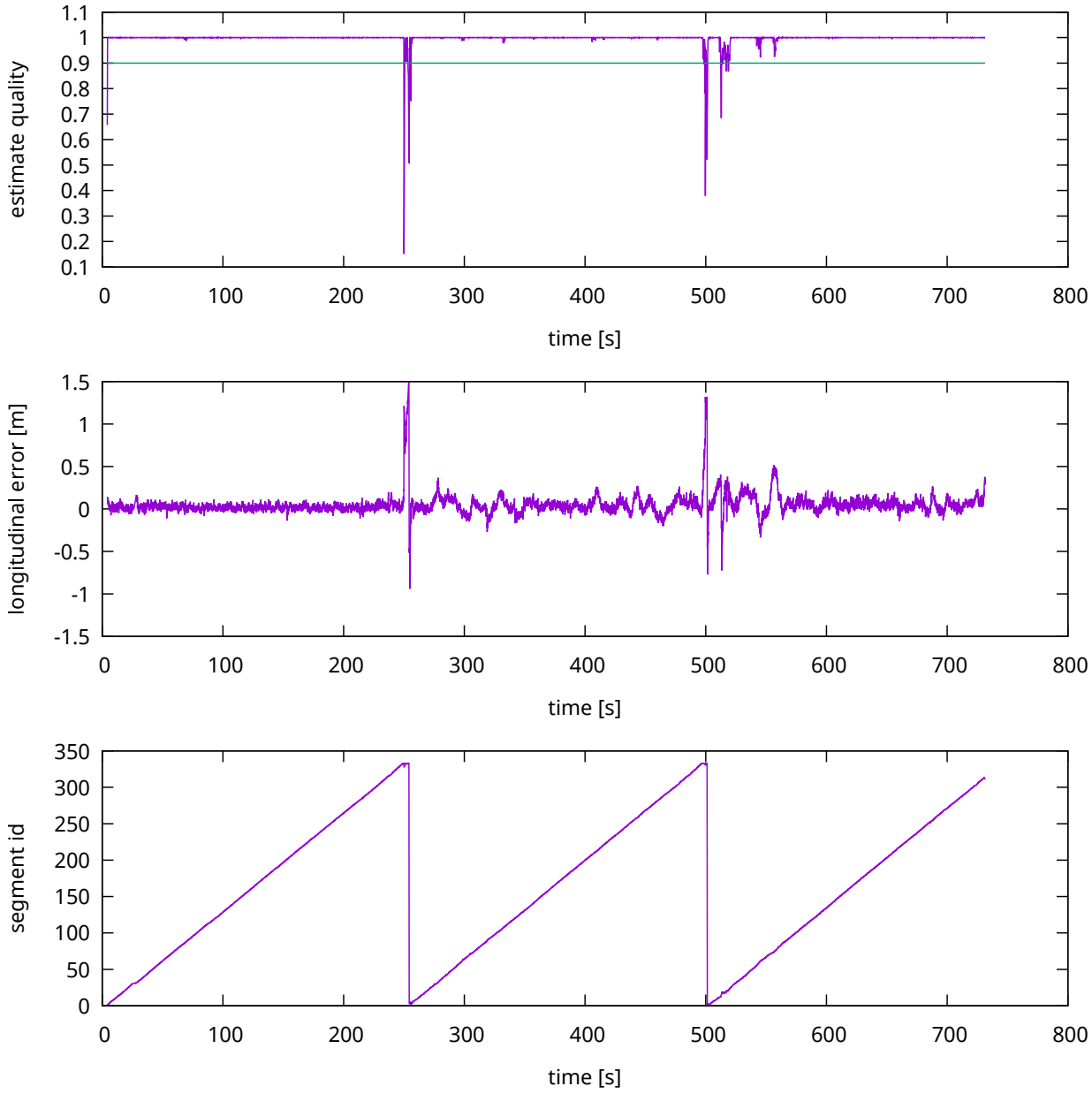


Figure 5.9: Longitudinal error as a function of time, for Section #2 of Level 7 Dataset

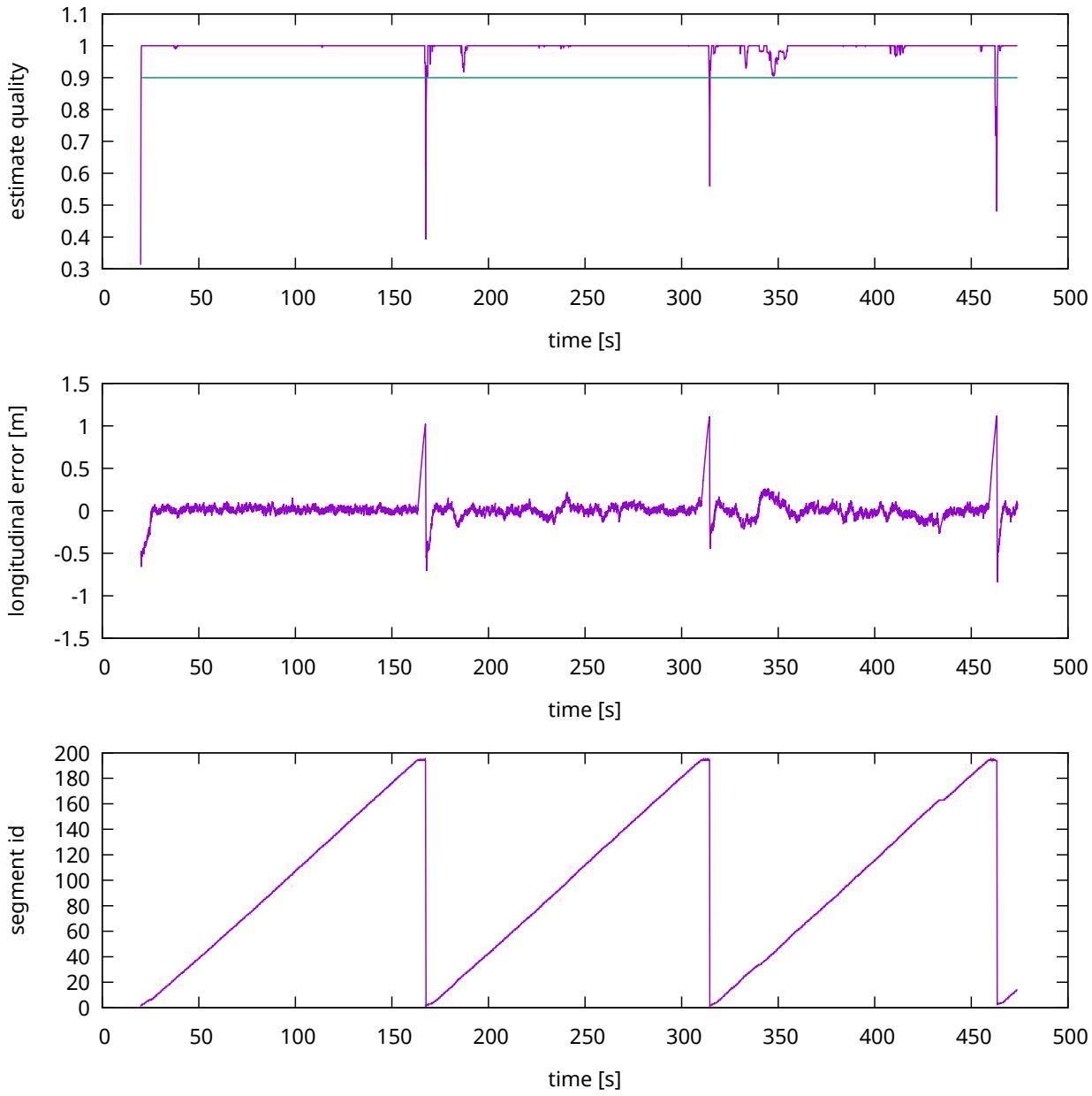


Figure 5.10: Level 7 Dataset Section #3 localization experiment

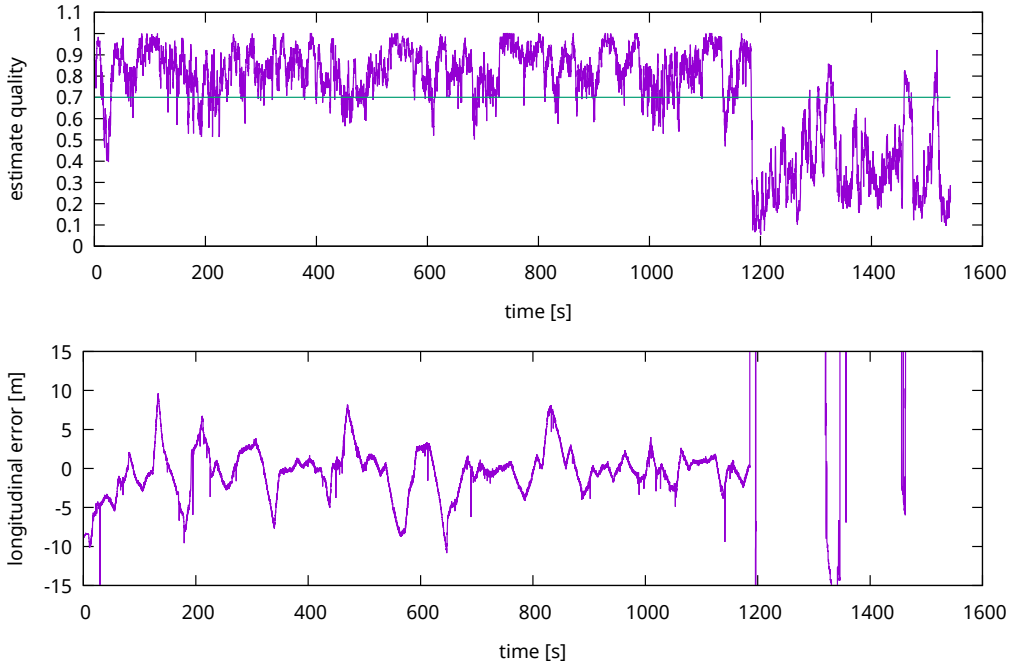


Figure 5.11: Longitudinal error as a function of time, for the "dry" variant of part B of SFU Mountain Dataset

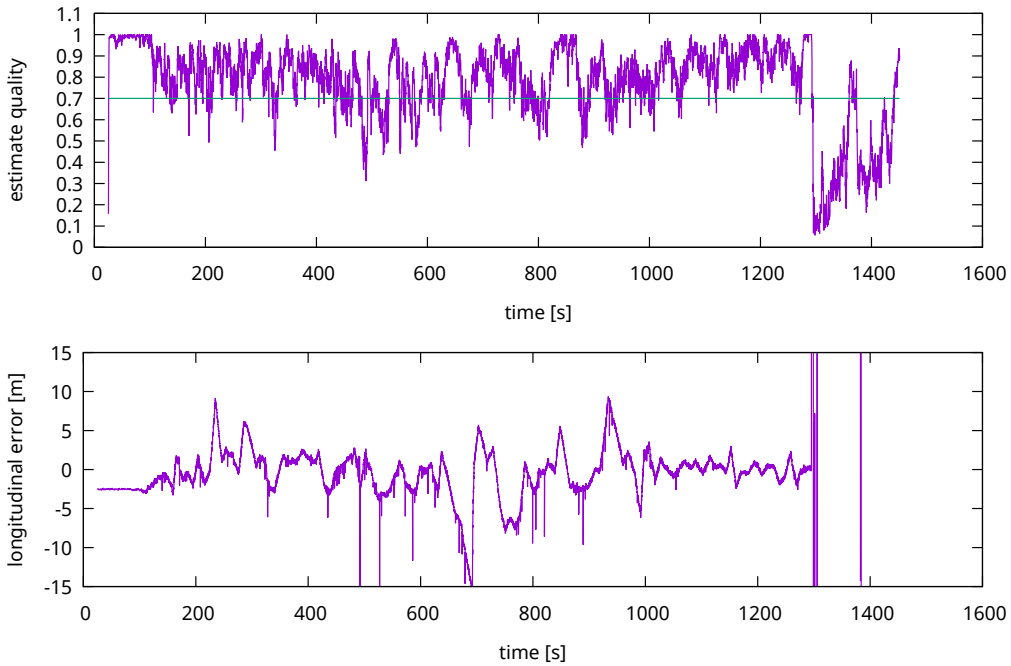


Figure 5.12: Longitudinal error as a function of time, for the "wet" variant of part B of SFU Mountain Dataset

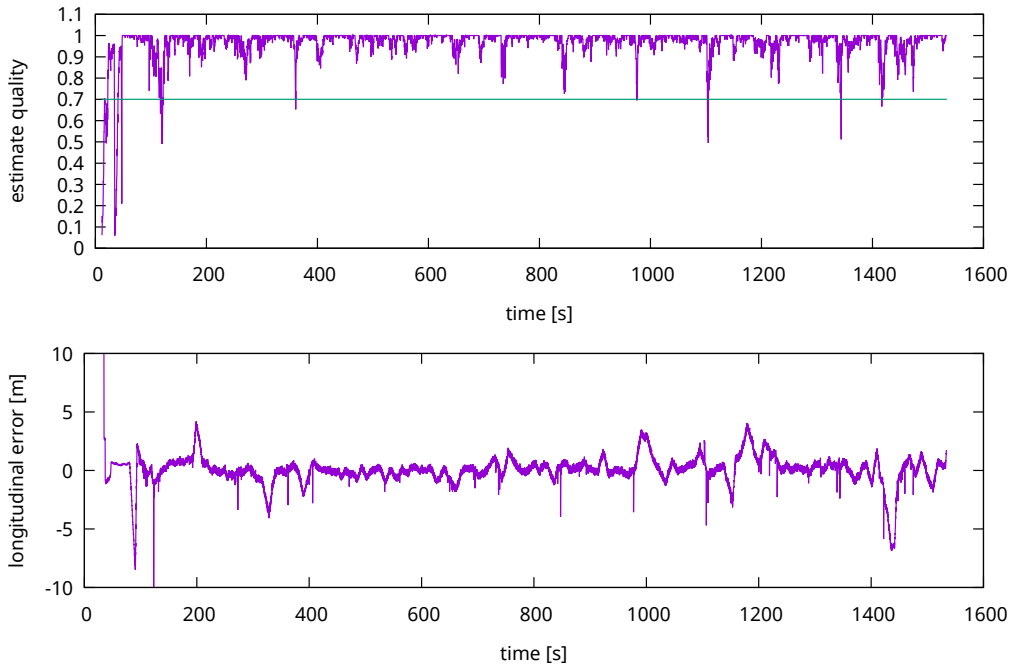


Figure 5.13: Longitudinal error as a function of time, for the "wet" variant of part A of SFU Mountain Dataset

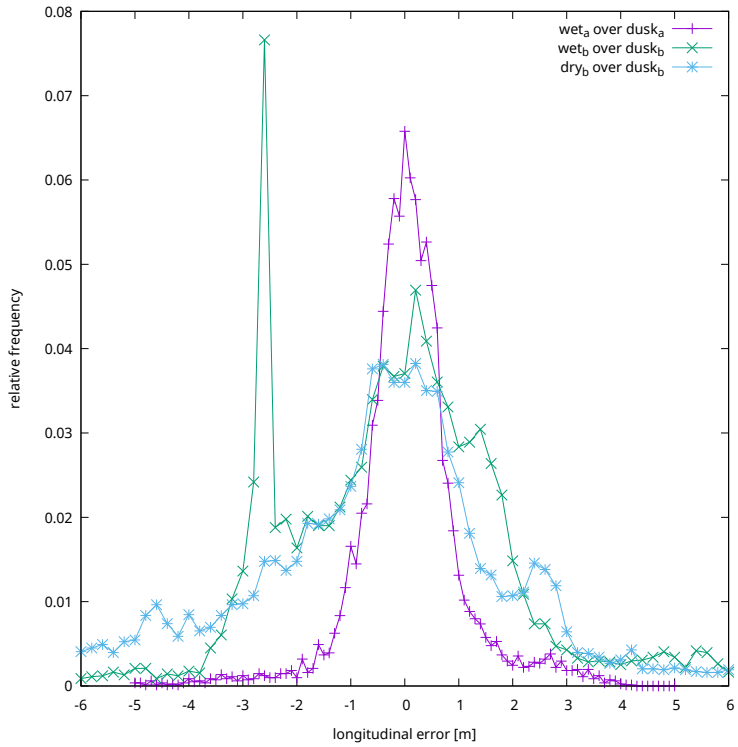


Figure 5.14: Histogram of longitudinal errors, for parts A and B of the SFU Mountain Dataset

5.6 Navigation Experiments

In this section, a series of experiments with the aim of exhibiting the closed-loop performance of the method are presented. In other words, the focus is put on the evaluation of the navigation task of the repeat phase.

A first round of experiments are focused on analyzing the response of the bearing-only control strategy (considering the different controllers presented in Chapter 2) for the case of a single fixed reference, instead of having a complete map to be followed. A second set of experiments aim to demonstrate in practice the convergence towards a simple straight trajectory, in the presence of lateral translation. Another set of experiments were then performed to show the usefulness of lookahead strategy for the bearing-only controller in the context of a curved path. Then, the ability to follow a map containing bifurcations is also analyzed. Finally, the case of an outdoor navigation with an aerial robot is included to demonstrate the validity of the approach in uncontrolled conditions.

The aforementioned experiments were performed both with the V-Rep simulator as well as with the LRSE quadcopter platform (executing on-board the embedded Odroid-U3 computer).

5.6.1 Bearing-only Control with Fixed Reference Pose

To perform this round of experiments, a subset of the proposed VTnR method was used where a reference pose can be set at any given time (during manual control) and later used as the target of the bearing-only controller. The aerial robot was manually controlled up to a given altitude of $\sim 1\,\mathrm{m}$, when the reference view was acquired. Then, the platform was manually commanded to introduce a certain offset w.r.t. the reference, distinguishing between the case of an orientation and an altitude offset. Finally, the robot was switched to autonomous control, using using the bearing-only controller (2.10), and convergence towards the reference pos was observed. Another experiment including a lateral offset was performed, to analyze the case of the alternative controller (2.11), which affects lateral velocity directly based on knowledge of the absolute yaw angle. An initial set of test runs where performed where the constants of the aforementioned controllers were tuned, prior to the experiments presented in this section. The goal was to avoid any oscillations while reaching for the fastest response.

For the experiments performed with the simulator, ground-truth altitude and heading information were readily available and were thus used to analyze the response of the controller. The visual information used as input to the bearing-only controllers (mode of azimuth and elevation differences) were also analyzed. On the other hand, due to difficulties in establishing ground-truth information for the experiments using the LRSE quadcopter, the estimated altitude and heading angle obtained by the autopilot were used as a reference instead.

It should be pointed that the lateral controller was only tested in the simulated environment as the LRSE quadcopter does not have a precise-enough yaw estimation. This problem is most likely due to the very basic estimation filters described in Section 4.3.1. It is expected that the use of more robust EKF-based state estimator will allow for this controller to be tested with the LRSE quadcopter.

Simulation

In these experiments, the standard bearing-only controller was tested as previously described. The lateral controller was tested by introducing all three orientation, lateral and vertical displacements w.r.t. the reference, in two different occasions and observing the convergence as with the previous cases.

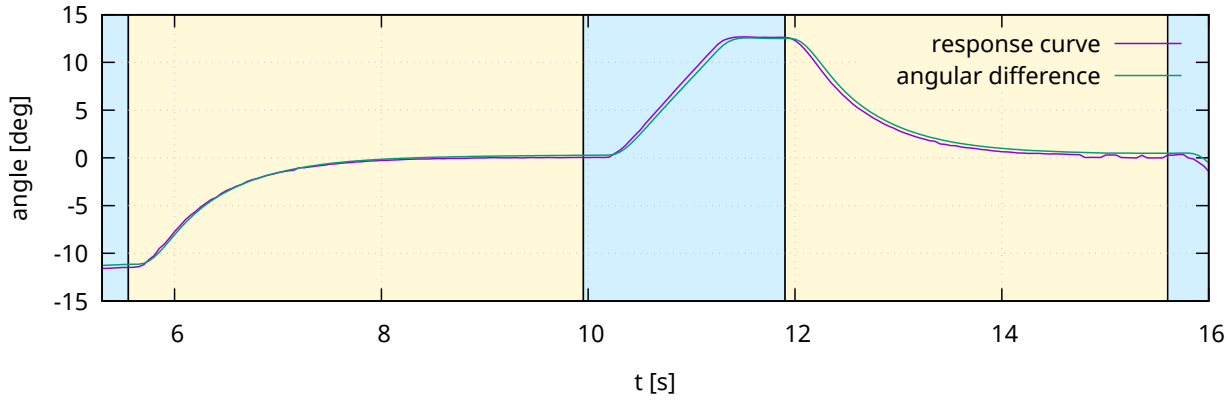
In figures 5.15b and 5.15a the case of vertical and angular displacement are presented, where the manual and autonomous phases are distinguished with a colored background. For the lateral controller case, the evolution of the individual relative errors over time along a spatial overhead view (including the orientation of the platform) are presented in Figure 5.17 on page 105.

The results demonstrate the correct response of the two previously proposed vision-based controllers

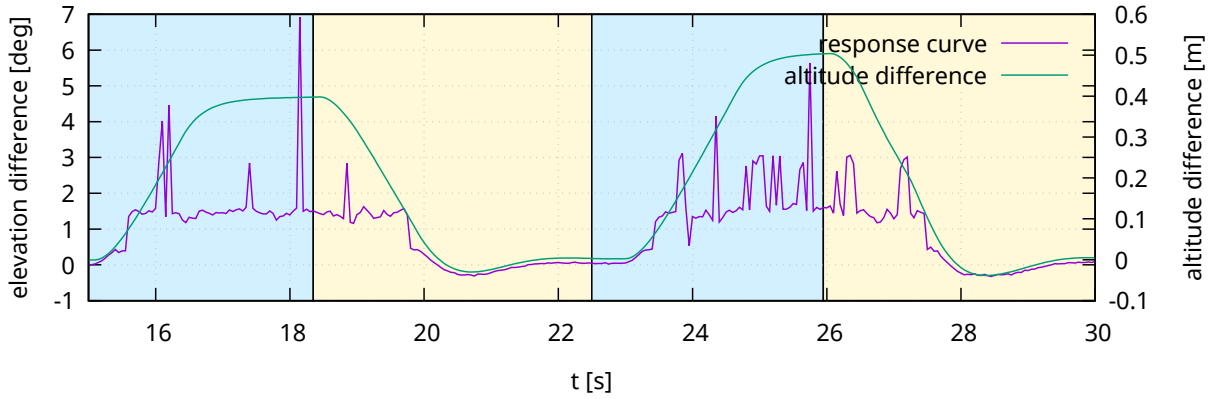
in a simulated environment. When the quadrotor was manually displaced and then left to operate autonomously, it returned back to the reference with negligible errors. While response appears to be quite slow, this behavior was desired to avoid any oscillations with more aggressive constants. It should also be noted that the introduced errors are higher than what would be found during path-following.

When comparing the relative pose errors w.r.t. the reference, to the corresponding visual information, a direct correlation can be seen as expected. For the case of heading control, the mode of the azimuth differences follows the relative yaw angle quite closely. On the other hand, the mode of the elevation differences appears to "flatten" and become noisy for a relative altitude greater than $0.2\,\mathrm{m}$. One likely reason, is the fact that the reference view corresponds to a view of a hallway, where features at difference distance are seen. Thus, the histogram becomes more dispersed and the mode estimated from the histogram becomes erratic. In consequence, while the response of the controller is correct, this can be seen as a shortcoming of the vision-based altitude control.

As for the lateral controller, it can be seen that all individual relative pose errors are quickly and independently minimized. However, as previously mentioned, this was only possible due to the high accuracy of the yaw estimation, which was not yet achieved with the real platform.



(a) Vision-based heading controller response

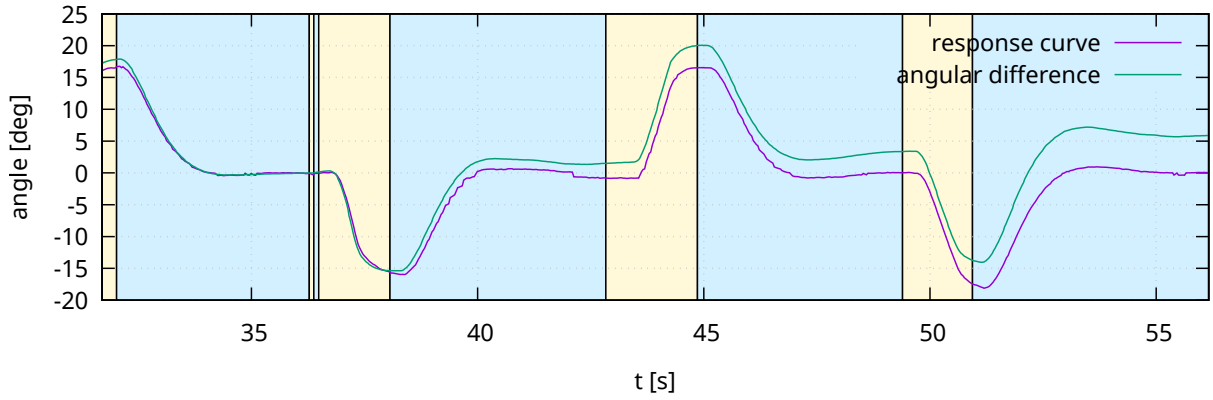


(b) Vision-based altitude controller response

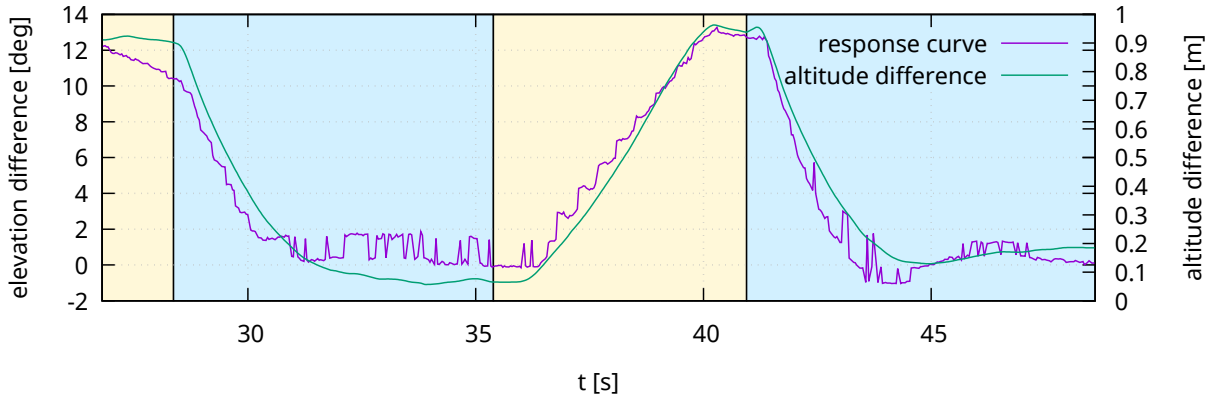
Figure 5.15: Simulated quadcopter altitude controller response curve: blue regions indicate platform under manual control and yellow regions indicate autonomous control

Real-Robot The experiments performed with the LRSE quadcopter are similar to those described in the previous section. In this case, the robot was flown in an indoor environment, facing a wall at around $5\,\mathrm{m}$. In this case, in addition to the vision-based altitude controller, one directly based on the altitude estimate

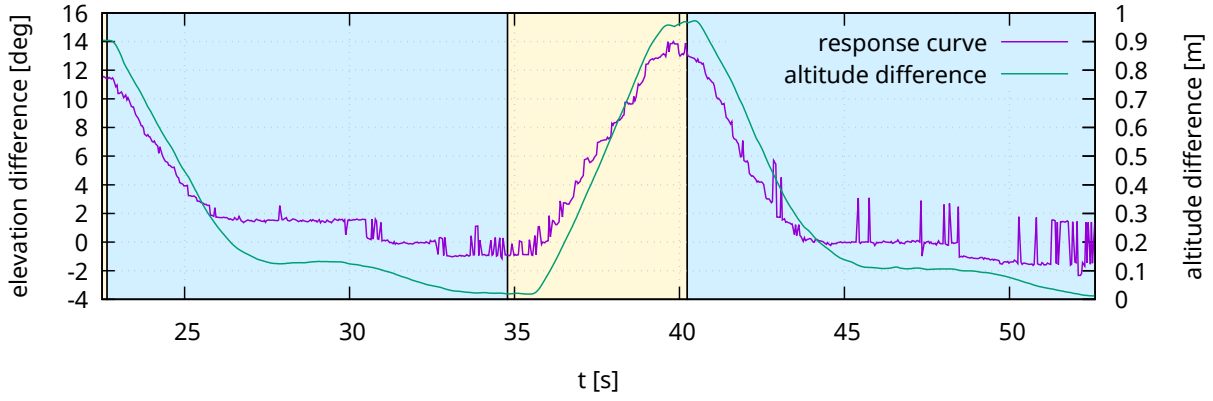
obtained from the down-looking sonar sensor of the robot was tested. The purpose of this experiment is to asses whether vision-based altitude control presents any advantage over to the standard approach based on a sonar sensor.



(a) Vision-based angular controller response curve



(b) Vision-based altitude controller response curve



(c) Sonar-based altitude controller response curve

Figure 5.16: LRSE quadcopter, response of the bearing-only controller

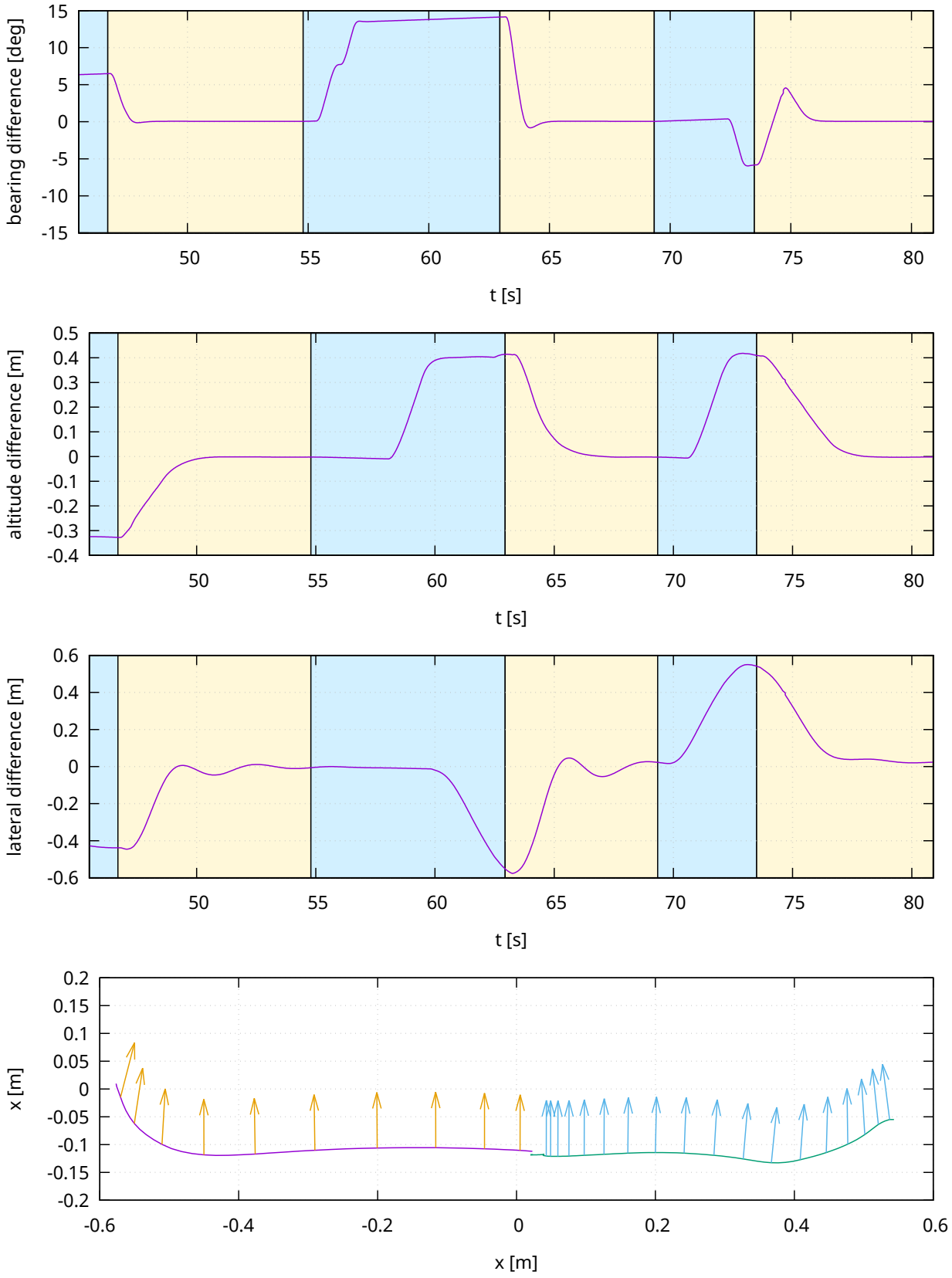


Figure 5.17: Simulated quadcopter, response of controller with yaw-derotation and lateral velocity

When analyzing the overall results (Figure 5.16 on page 104), again it can be seen that the response is as expected. For the particular case of the heading control, when comparing the angular difference w.r.t. to the mode obtained from visual information and from the autopilot estimate, a slow drift in the latter is observed. By observing the camera images during autonomous operation, it can be seen that with the bearing-only controller convergence is effectively achieved. Thus, it can be stated that it is the yaw angle estimate of the autopilot which drifts and not the vision-based estimate. This results speaks about the main advantage of vision for the purpose of bearing-only navigation.

Regarding the altitude control, the response both for the sonar-based and vision-based approaches is quite similar, achieving convergence with similar final errors to the reference. While it would appear that both approaches are equally valid, due to the fact that a wall was observed by the robot during the experiments the behavior previously obtained in simulation was not exhibited. In this case, the sonar-based approaches has the advantage. On the other hand, in case objects or even the steps of a stair are overflown, the sonar-based approach would have the effect of changing altitude to keep the distance to the ground constant. On the contrary, since by using vision the whole scene is used as a reference, this problem would not occur. In the end, a better strategy would be to fuse both measurements to allow for a more robust approach.

5.6.2 Straight Path

After having examined the in-place response of the proposed bearing-only controllers in the presence of various offsets w.r.t. a fixed reference, in this case the path-following response of the controllers were analyzed by teaching then replaying a straight path, with initial lateral and vertical offsets.

This experiment was performed both in simulation and using the LRSE quadcopter. In the former case, lateral, longitudinal and vertical errors over time, w.r.t. the path, were obtained. Moreover, an overhead and lateral view of the trajectories followed in each pass was also reconstructed. For the LRSE quadcopter experiments, ground-truth pose information was obtained using the external localization system described in 5.1.1. However, due to the difficulty in establishing a relation between the VTnR method's pose estimate and the true robot location for each time, only the trajectory (and not the errors w.r.t. the path) was possible to obtain. Moreover, for safety reasons, in these experiments the quadcopter was not vertically displaced. The main difficulty when flying indoors with this platform is the fact that when performing pitch an roll motions at higher altitudes, the ground-area sensed by the sonar range-finder becomes larger and the readings are noisier. Even under vision-based altitude control, the autopilot corrects vertical velocity based on sonar-readings and thus can easily become unstable in this situation. Thus, path-following experiments performed with changing altitudes where only performed in simulation.

Simulation

A straight path of approximately $9\,\mathrm{m}$ was first taught. Then, the repeat phase was initiated both with and without lateral and vertical displacements. Lateral offsets of approximately ± 1.5, $\pm 0.7\,\mathrm{m}$ and vertical offsets of approximately $1\,\mathrm{m}$ were introduced. A goal was set at the end of the path and the robot was left to operate autonomously until it detected reaching the goal.

In figure Figure 5.19 on page 108, a lateral and overhead view of the trajectories followed by the robot in this experiment is presented. In Figure 5.18 on page 107, lateral, longitudinal and vertical offsets w.r.t. the path are also presented. It should be reminded that a longitudinal offsets corresponds to a localization error, while lateral and vertical correspond to navigation errors.

Analyzing the results, it can be seen that the behavior of convergence to the path is effectively achieved. Regarding lateral versus vertical convergence, it can be noticed that a lateral error takes much longer to be minimized, which is expected due to the bearing-only controller used. However, after replaying the complete path, lateral errors were mostly around $\pm 0.3\,\mathrm{m}$. On the other hand, vertical offsets quickly disappeared after only $5\,\mathrm{s}$. Finally, localization errors where generally within $0.2\,\mathrm{m}$, after a few initial seconds required

to converge.

Real-robot

The experiment in this case was quite similar to that in simulation. The taught path length was around $7\,\mathrm{m}$ and lateral offsets where within $0.4\,\mathrm{m}$, since the area used for experiments was not large enough to allow for larger offsets, which would have required a longer path to achieve path-convergence. Overall, convergence was also observed obtained in this case, although not as noticeable as with the experiments performed in simulation.

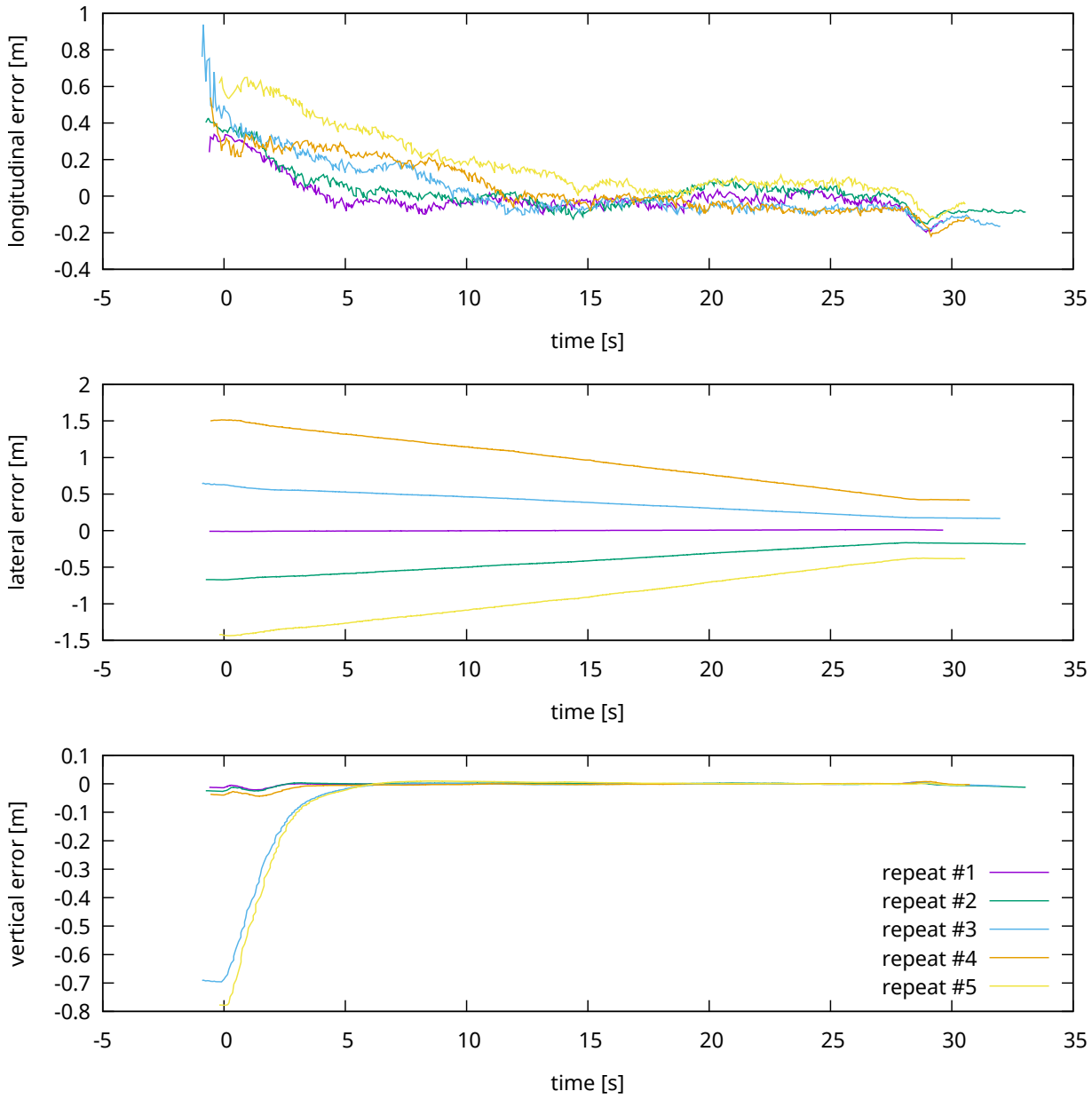


Figure 5.18: Simulated quadcopter, straight segment following: longitudinal, lateral and vertical errors w.r.t. the path

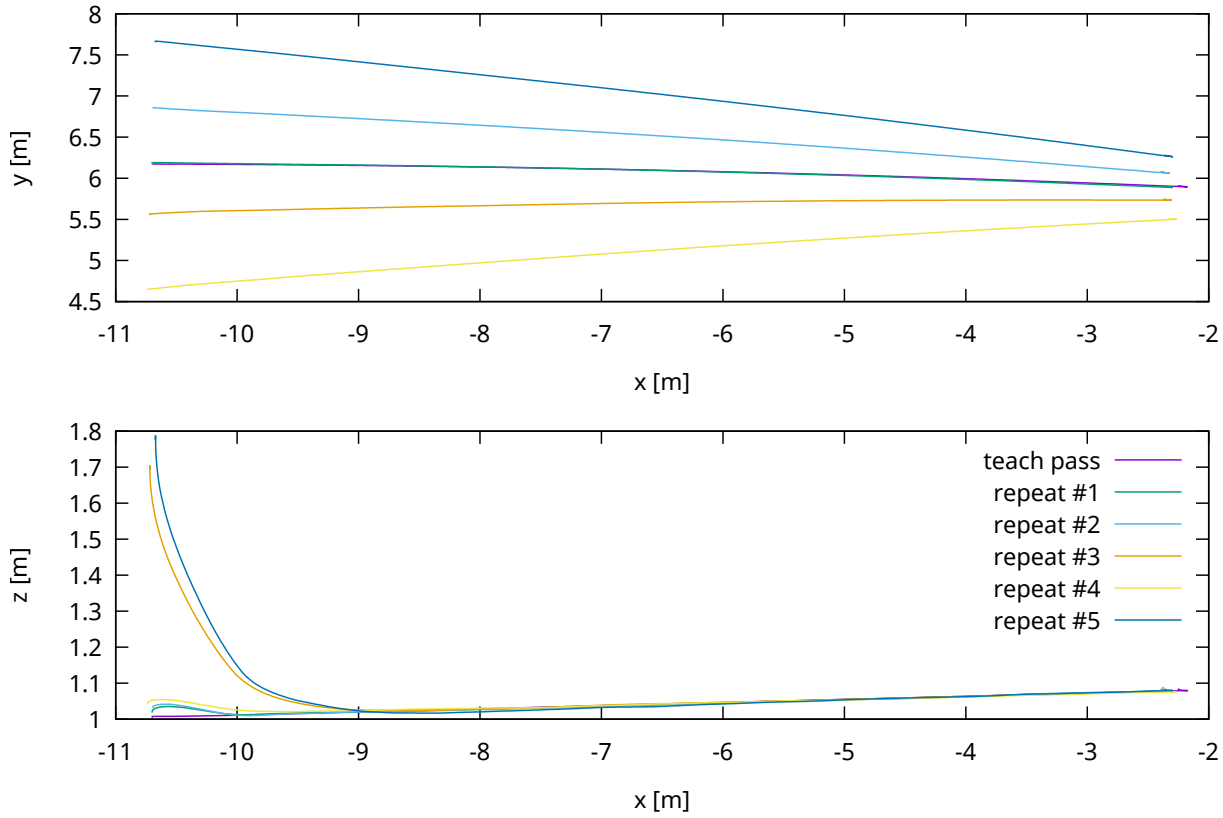


Figure 5.19: Simulated quadcopter, straight segment following: overhead and lateral trajectory views

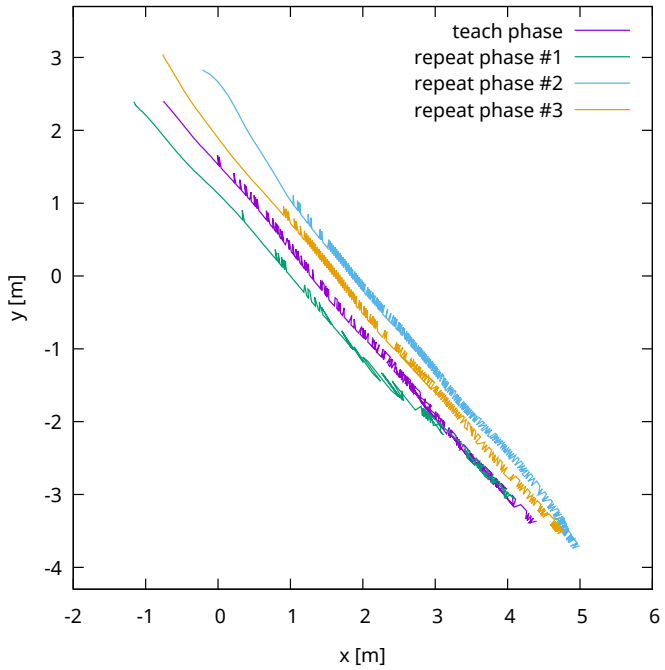


Figure 5.20: LRSE quadcopter, straight segment following: overhead view

5.6.3 Lookahead

In 3.5.3, a variation of the bearing-only controller originally proposed in Chapter 2 was presented, which introduced the notion of a *lookahead*, for the purpose of obtaining an improved trajectory-following response with curved paths. For this reason, a series of experiments were performed to demonstrate the positive effect of the lookahead. To do so, the original controller without lookahead (lookahead value of zero) was compared to the controller with non-zero lookahead, by repeating a "C"-shaped path. Again, as mentioned in the previous section, both absolute and relative pose information was obtained for the experiments in simulation, while only the overhead trajectory was obtained with the LRSE quadcopter. Moreover, altitude changes were only performed in simulation.

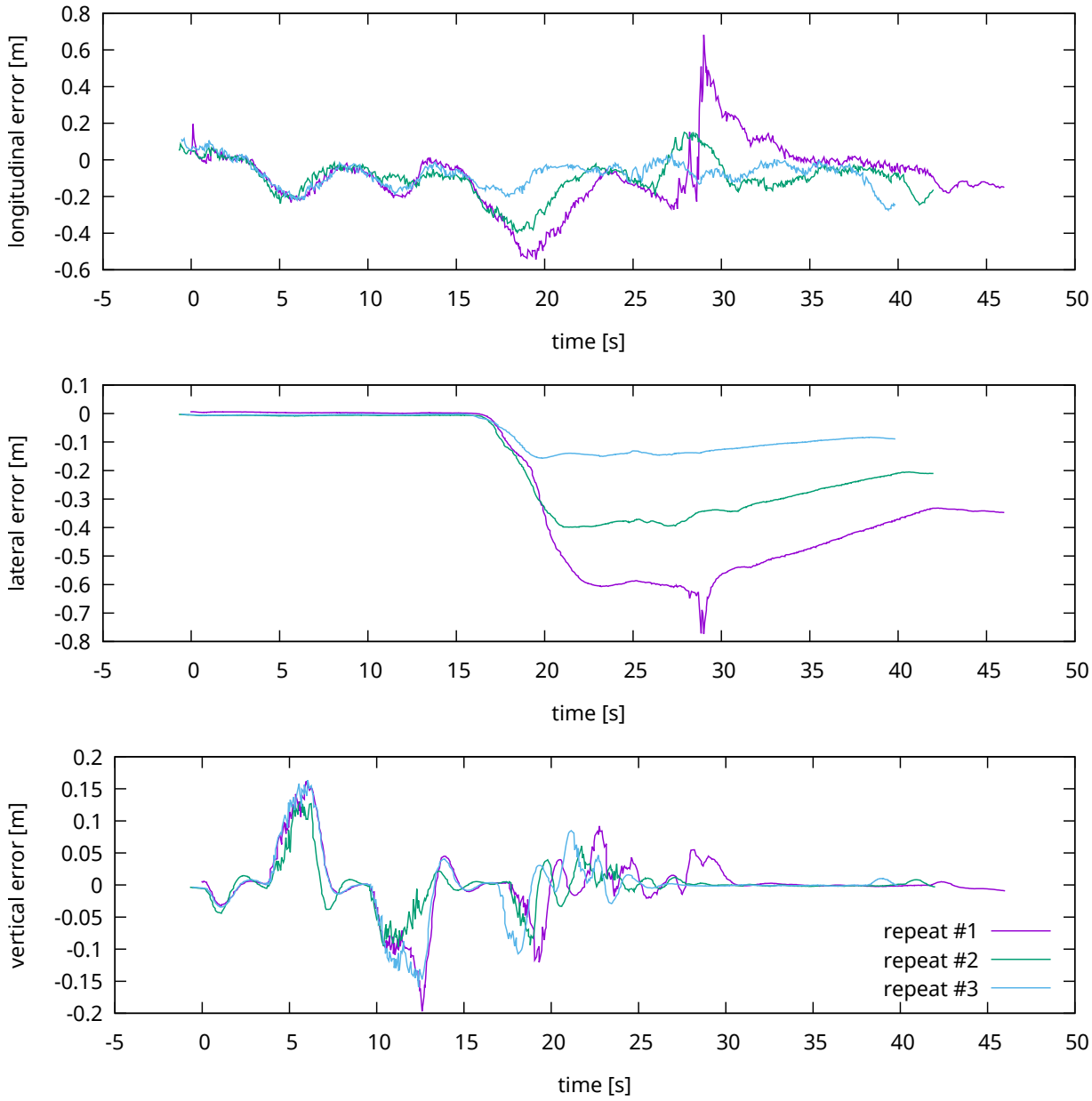


Figure 5.21: Simulated quadcopter, curved-path following with altitude changes: longitudinal, lateral and vertical errors

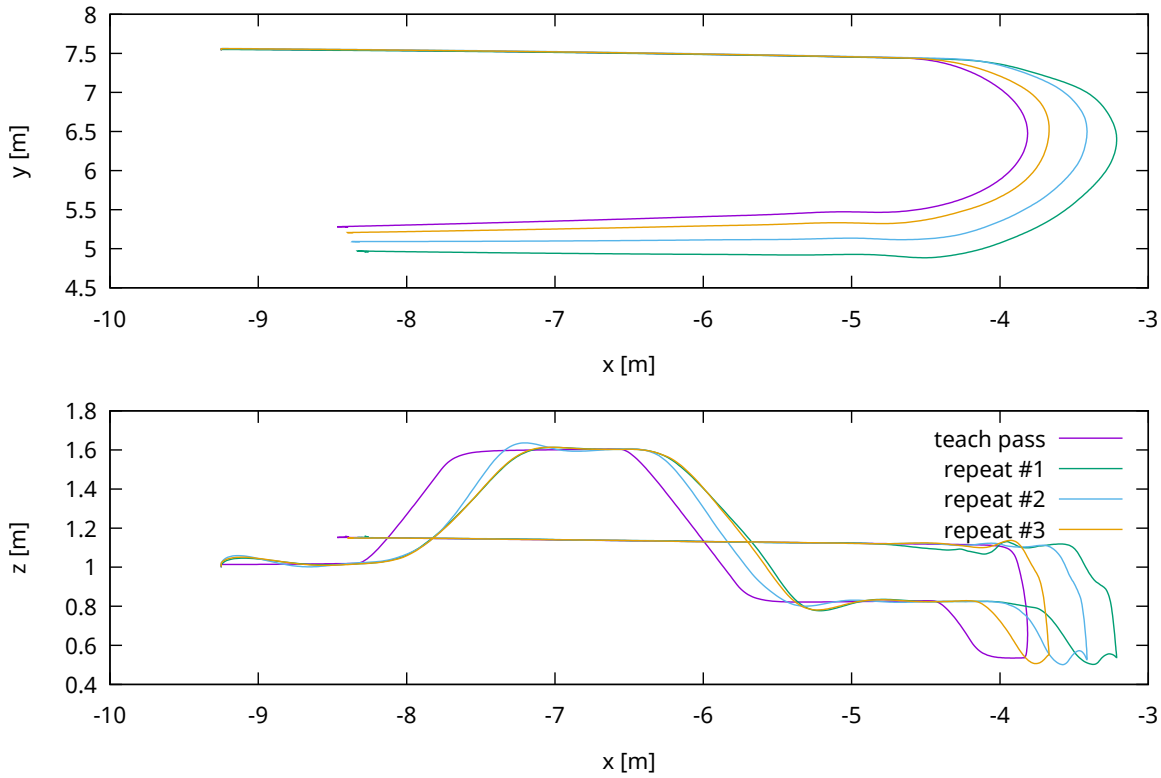


Figure 5.22: Simulated quadcopter, curved-path following with altitude changes: overhead and lateral trajectory views

Simulation

In this experiment, three repeat phase passes were performed. In the first, the lookahead term was set to zero and conservative value for response constant of the angular and altitude controllers was used. When observing the trajectories (Figure 5.22 on page 110) and the relative errors (Figure 5.21 on page 109), it can be seen that after encountering the curve, in this case a considerable lateral error appears and is then never fully diminished. Moreover, localization was close to failure, as it can be seen from the longitudinal error plot. Then, higher values of the controller response constants where used, still without the use of the lookahead. In this case, while a smaller lateral error is obtained after the curve, angular oscillations started to appear and localization was still unstable. Finally, when using a lookahead of $0.5\,\mathrm{s}$ and the original angular and altitude response constants, tracking was much improved and localization errors were always within $0.2\,\mathrm{m}$.

On the other hand, when observing altitude response in the presence of vertical offsets, it can be seen that the lookahead does not seem to have such a noticeable effect. However, this is not surprising since altitude is here directly controlled by affecting vertical-velocity, in contrast to lateral errors which are indirectly controlled by a combination of affecting the heading and moving forward. Thus, it is expected that lateral response will be slower and thus the lookahead strategy will be more effective than for altitude response.

Real-robot

Using the LRSE quadcopter, a similar trajectory was taught and the case of conservative and more aggressive angular response constants with zero lookahead was compared to the use of non-zero lookahead. Observing the overhead trajectories followed by the robot (Figure 5.23 on page 111), a similar effect to that obtained in the previous experiments can be seen. Without the use of lookahead, when following the first turn, a

noticeable lateral error occurs when no lookahead is used. Using a more aggressive constant allows to partially reduce this error, but at the expense of motion stability. On the contrary, using a lookahead of $0.5\,\mathrm{s}$, a less aggressive angular response constant can be used, achieving more stability and at the same time obtaining a closer tracking of the trajectory.

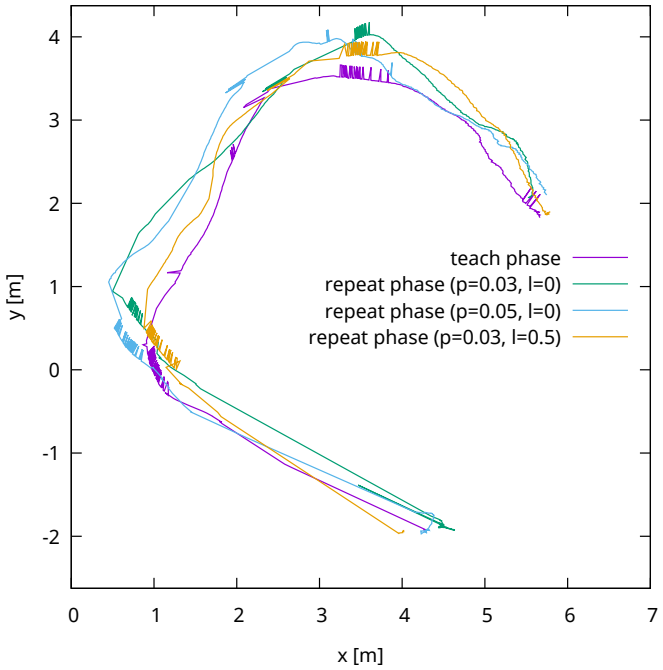


Figure 5.23: LRSE quadcopter, curved-path following: overhead trajectory view

5.6.4 Bifurcation Following

To validate the proposed approach for following more complex paths, a simple experiment was performed with the simulated platform, where one path was first taught and later extended.

To add the alternative path, after the initial teach phase, the simulation was reset and the robot was first told to repeat the previous path up to a desired bifurcation point. Then, teach mode was engaged and a straight trajectory from this point was taught. Then, starting from the beginning of the original path, two runs of the repeat phase were performed where the goal was set to reach the ending of each path, respectively. In Figure 5.25 on page 112 the overhead trajectory is presented and in Figure 5.24 on page 112 the lateral and longitudinal errors w.r.t. to the estimated pose in the map, along with the pose's segment identifier are presented.

Observing the obtained results, the expected behavior of the proposed method for repeating a path with bifurcations can be seen. In terms of navigation precision, from the overhead trajectory it can be concluded that the robot is able to precise follow each path smoothly, even while at the bifurcation point the scene looked quite similar from the viewpoint of the robot. Regarding localization, a brief error when the single-hypothesis is near the bifurcation point can be seen in the longitudinal error w.r.t. to the estimated pose and in the pose segment identifier as well. On the other hand, when observing the lateral error w.r.t. to the estimated pose, since the pose is not correct, the an error in the lateral axis is also generated. However, this is not actually a path-tracking error, which can be confirmed by observing the overhead trajectory, which does not exhibit this jump of about $1.5\,\mathrm{m}$.

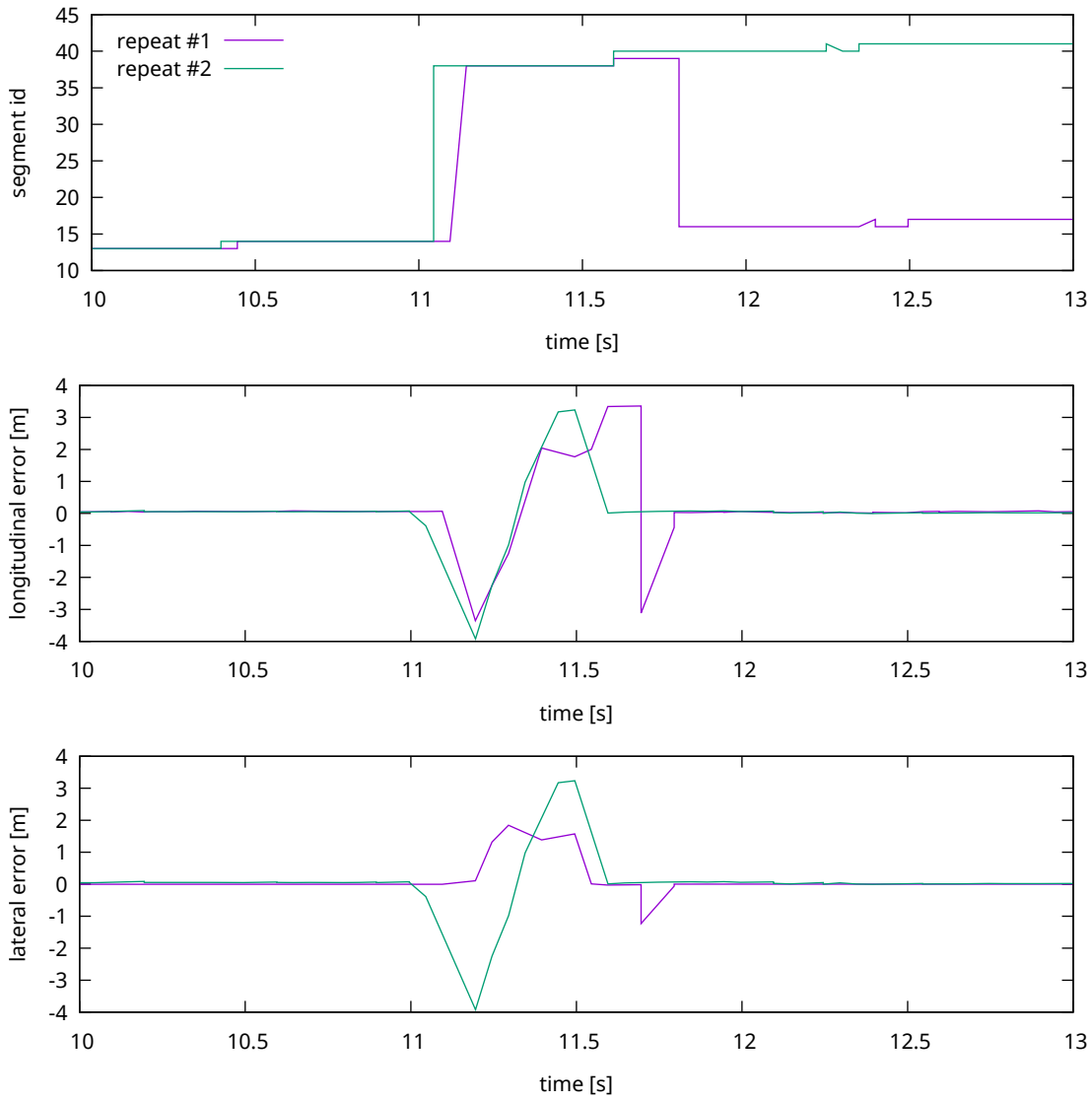


Figure 5.24: Pose errors with respect to estimated robot location in the map, and segment identifier of the robot pose, when following each bifurcation of the map

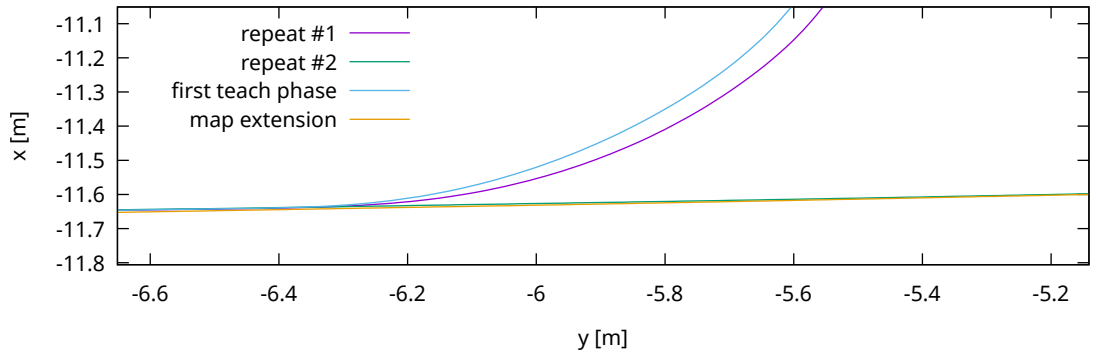


Figure 5.25: Trajectories followed by the robot when teaching and repeating a bifurcated path

5.6.5 Outdoor Navigation

While previous experiments focused on particular aspects of the proposed VTnR method, these were mostly performed under controlled (whether real or simulated) indoor environments. Thus, it is interesting to evaluate the performance of the method in an uncontrolled outdoor environment. However, due to the inability to obtain ground-truth pose information this experiment mostly serves as a proof of concept and rough validation of the method. Moreover, while internal variables such as localization quality and number of feature matches would provide an interesting analysis, in this case it was not possible since after the experiment it was found that this data was lost due to failure of the internal storage memory. Due to time constrains, this experiment was not possible to repeat and it is left as future work. For these reasons, only video of the platform being commanded to learn a path and then observed while it repeated the path is available. Selected frames from these videos are presented in this section.

In the experiment two paths were learned and repeated: a small square-shaped path (see 5.26) and a longer "L"-shaped path, the latter repeated a total of three times (see 5.27). In general, from the external point of view, the performance during the repeat phase was satisfactory in both scenarios. As in the second experiment the robot was taught a path over a $\sim 1\,\mathrm{m}$ wide cement side-walk, and it was seen that it remained roughly over this area, it can concluded that lateral error was withing approximately $1\,\mathrm{m}$ as well. In the third repetition of the longer path, though, as the platform was flying over the edge of the sidewalk and a steel pole was there present, a manual override had to be issued, where the robot was laterally displaced towards the path again, and autonomous navigation was then resumed.

Figure 5.26: Photo of the LRSE quadcopter performing autonomous navigation in uncontrolled outdoor environment (closed circuit)

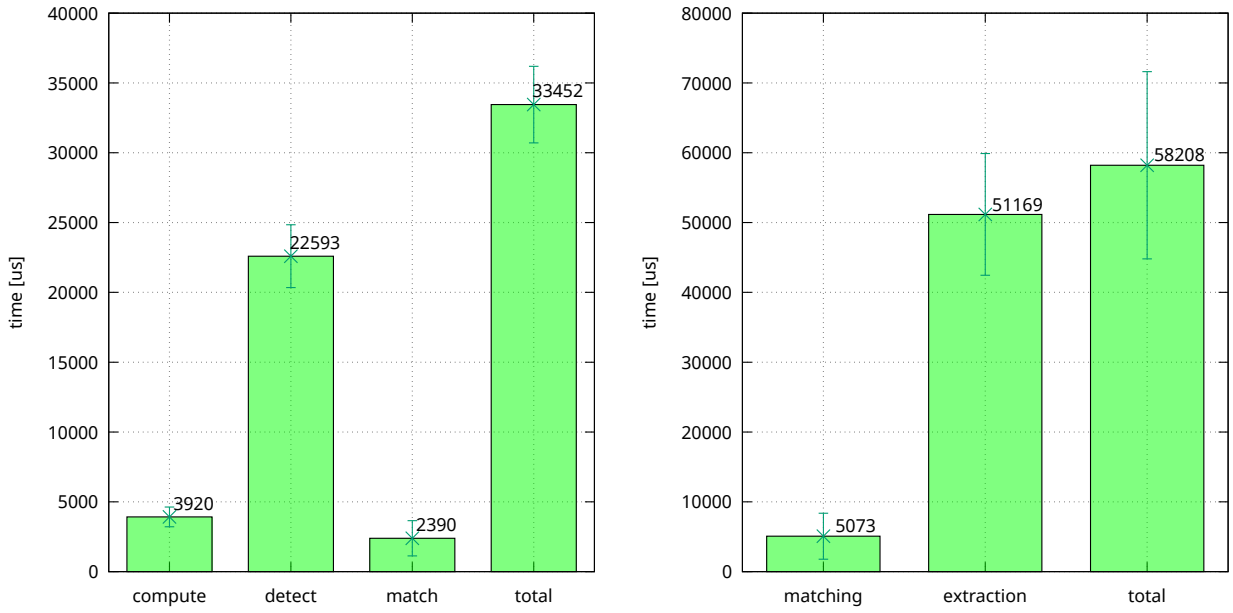
Figure 5.27: Photo of the LRSE quadcopter performing autonomous navigation in uncontrolled outdoor environment (trail following)

5.7 Execution Performance Analysis

In this section, the computational performance of the VTnR method when executing on-board the LRSE quadcopter is analyzed. To do so, two short teach and repeat phases were executed in an indoor setting. In each case, the times spent on the execution of the most-demanding portions associated with the map-building, localization and navigation tasks were measured (see Figure 5.28 on page 115). These portions correspond to the main visual-processing steps: feature-extraction, descriptor-computation and feature-matching.

For the case of the map-building task of the teach phase (see Figure 5.28a on page 115), the majority of the execution time required upon the arrival of each camera image is spent on feature detection, requiring about $22\,\mathrm{ms}$ in average, whereas the second most-demanding task is the descriptor computation, which takes about $4\,\mathrm{ms}$. Only a very brief period of about $2.4\,\mathrm{ms}$ is spent on matching currently-extracted features to those in the STM. In average, around 70 features were extracted from camera images, of which an average of 52 of 63 were matched to those of the STM.

For the repeat phase, the computational requirements of the navigation and localization tasks were also measured (Figure 5.28b on page 115 and Figure 5.29a on page 116, respectively). Again, the majority of the execution time is spent on the feature extraction (i.e. detection plus description) step, whereas most of the remaining time is spent on feature-matching, which includes the computation of the histograms and extraction of the modes. Finally, for the localization task, the most demanding step corresponds to the computation of the particle weights, which involves matching currently extracted features to those in the map, for each particle. A total of 50 particles were used in this case, resulting in about $2.5\,\mathrm{ms}$ for each particle, which is consistent with the matching time spent in the map-building task.

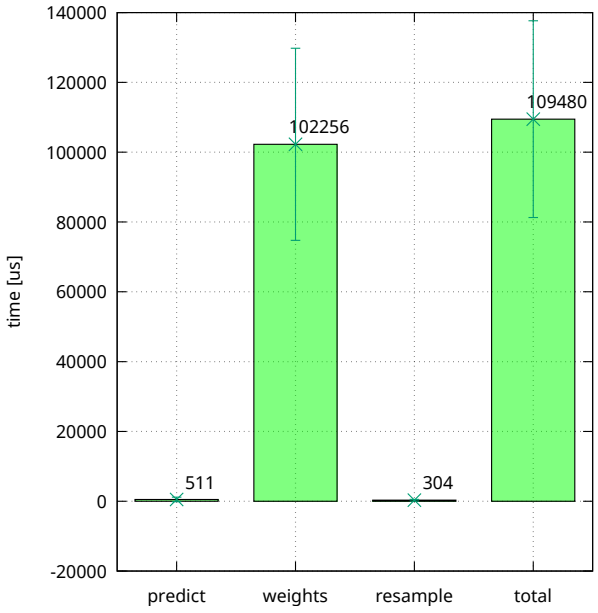


(a) Map-building task **(b)** Navigation task

Figure 5.28: Execution times of the main steps associated to each task of the method

Regarding the results obtained for the teach phase, the per-image processing time is sufficiently low to allow for around process images at $30\,\mathrm{Hz}$, which is the usual frame-rate of most cameras. Regarding the navigation and localization tasks, while the per-image processing time is considerable higher in the latter (allowing processing images at around $10\,\mathrm{Hz}$), since both of these tasks are handled in parallel, what is of

importance in terms of navigation is the processing rate of the former, which is around $15\,\mathrm{Hz}$. For a quadcopter moving at around $0.4\,\frac{\mathrm{m}}{\mathrm{s}}$, as it is the case of the performed experiments, this response times appear to be sufficient. For faster moving robots, considering that nowadays there are newer and more powerful embedded computers of similar size and power requirements, and the fact that the feature extraction and descriptor computation are internally parallelized and highly-optimized using SIMD instructions, a hardware upgrade may allow obtaining faster processing rates.



(a) Localization task

Figure 5.29: Execution times of the main steps associated to each task of the method

Capítulo 5

Resultados (resumen)

En este capítulo se presentan los resultados obtenidos a partir de la realización de diversos experimentos utilizando el método de navegación propuesto. Los experimentos se dividen principalmente en tres grupos: análisis de precisión y robustez de la localización, calidad de la navegación en diversos contextos y performance del método al ejecutar sobre una plataforma embebida.

El primer grupo de experimentos de localización se realizó fuertemente sobre conjuntos de datos pregrabados disponibles públicamente. La razón de esta elección es la necesidad de contar con información de verdad absoluta respecto de la pose real del robot, para poder establecer una métrica sobre la calidad de la localización bajo el enfoque propuesto. Los datos utilizados corresponden a recorridos realizados mediante robots terrestres, debido a la inexistencia de datos obtenidos con robots aéreos, que fueran adecuados para evaluar el método propuesto. A partir de las pruebas realizadas con diversos conjuntos de datos, se pudo observar que la calidad de la localización era muy buena (menor a $20\,cm$ en general) en casos relativamente favorables, mientras que se observaron errores de localización mayores en casos más desafiantes (alrededor de $1\,m$).

Para evaluar la calidad de la navegación se realizó primero un conjunto de experimentos que se enfocaron en evaluar la respuesta de los controladores basados en rumbos, al tratar de alcanzar una referencia fija y al seguir primero un camino recto y luego uno curvo. En el primer caso, se observó la correcta respuesta de los controladores que, si bien resultó ser lenta dicho resultado era esperable para las condiciones experimentales utilizadas. Por otro lado, al seguir una línea recta, se evaluó la capacidad del robot de converger nuevamente al camino luego de haber sido desplazado inicialmente. Finalmente, se evaluó la habilidad de seguir caminos curvos, que es particularmente relevante dado que se propuso una estrategia específica (la de colocar la referencia más adelante que de la posición actual) para lidiar con estos casos. Ambos experimentos resultaron satisfactorios y fueron realizados tanto en simulación como con el cuadracóptero LRSE. Un último experimento se realizó a modo de prueba de concepto al aprender y repetir un camino con el cuadracóptero LRSE en un entorno exterior y no controlado. Si bien no se pudo extraer datos numéricos sobre la localización y navegación en este caso debido a dificultades técnicas, se evidenció a grandes rasgos el correcto funcionamiento del sistema.

Por último, se realizaron mediciones de tiempos computacionales asociados a la ejecución en la plataforma embebida, de las diversas tareas de localización, navegación y aprendizaje de un camino. Como era esperable se observó que las tareas más demandantes corresponden a las de procesamiento de características visuales. Se pudo ver también que los tiempos de ejecución son suficientemente chicos como para permitir la navegación del robot aéreo desplazándose a velocidades relativamente bajas. Asimismo, los resultados muestran que es factible el incremento de los tiempos de respuesta a partir del uso de hardware mas actual, debido a que los tiempos se concentran sobre tareas que pueden ser resueltas de mejor forma al existir un mayor poder de cómputo disponible.

Chapter 6

Conclusions

In this thesis an appearance-based navigation method based on the Teach and Replay strategy for aerial and ground-robots is presented. This type of navigation is particularly useful for scenarios such as remote inspection, parcel delivery or search & rescue, where to solve the task, a previously learned path needs to be repeated, ideally without human intervention.

The appearance-based formulation is based on the use of a topological graph where visual information is stored, allowing to solve the localization and navigation problems without reconstructing the structure of the environment. Moreover, the TnR approach to navigation enables a relative representation of the robot pose, instead of using a single privileged coordinate system. The topological map thus results in manifold representation of the learned path. In consequence, the typical problem of pose drift in the localization task, which usually requires complex approaches and computationally expensive algorithms (such as Bundle Adjustment) to avoid this error to spoil the approach, is in this case evaded.

Compared to other existing approaches to VTnR navigation, a robust and efficient localization strategy based on Monte Carlo Localization (MCL) is used. The pose of the robot over the map is estimated by means of a series of location hypothesis (the *particles*) which are use to sample the underlying unknown probability distribution. One of the main benefits of this approach is that localization complexity is independent of the map size, and only on the number of particles. Thus, it is possible to easily balance computational efficiency over localization robustness and precision. Second, with MCL, solving localization failure and localizing without *a priori* information is possible, which in the context of the TnR becomes a necessary ability for handling real-world scenarios. Finally, thanks to the appearance-based formulation of the problem, localization only deals with a one-dimensional search-space, which further simplifies the problem and minimizes certain limitations of the MCL approach (such as the particle deprivation problem).

Navigation in the proposed approach is solved using a very simple bearing-only control strategy. While this scheme was already used extensively for the case of ground-robots, only few works applied this strategy to aerial-robots. Furthermore, either switched-control laws were used or the underlying principles were mostly overlooked. In this thesis, bearing-only navigation in 3D space is analyzed by relating this approach to the concept of optical-flow path-following. Also, a simple and robust strategy to deal with multiple landmarks and many bad correspondences is presented, without resorting to more complex random sample-and-consensus approaches commonly used. This strategy involves the computation of azimuth and elevation bearing-differences and the extraction of the corresponding modes to guide navigation.

Another considerable contribution compared to the classical TnR approach where a single-path is learned and then fully repeated, navigation in the proposed method is generalized by considering apropriate structures and algorithms which allow to represent a network of interconnected paths. As a result, to reach a given goal over the whole map, only the required set of paths can be repeated up to the desired location. Using the graph-based topological approach, bifurcations and junctions can be represented and localization and navigation can be handled without requiring a globally consistent map.

As the main goal with the proposed method is for it to be run on-board aerial robots with relatively limited computational power, the visual-processing tasks (which are the most computationally demanding) are based on the use of efficient feature extraction, description and matching algorithms. For this purpose the BRIEF descriptor is used, which balances efficiency and robustness, together with the GFTT feature extraction algorithm. While this combination does not offer rotational invariance, which is required for roll motions usually performed by aerial robots during flight, a simple descriptor derotation strategy is used based on the estimated roll angle of the robot. This same information is used to derotate the estimated optical-flow values, which make possible the use of the same bearing-only controllers used typically for ground-robots, but for the case of aerial-robots.

Together with the development of the VTnR navigation approach, this thesis also presents an experimental aerial platform, the LRSE quadcopter, which was designed and built for the purpose of allowing experimental evaluation of the proposed navigation method using a platform which is able to carry the required processing and sensing hardware on-board. Facing experimentation using this platform, although quite challenging, has allowed to obtain a VTnR navigation method which is able to handle real-world conditions. Also, both the build process and the experimentation with this platform resulted in an invaluable learning experience. Moreover, in order to obtain a stable platform which could be autonomously operated by means of high-level velocity commands in indoor environments (without the use of GPS), considerable changes where required to the Pixhawk autopilot source-code. Several of these changes were shared with the open-source Pixhawk community, which then resulted in contributions to the project itself.

6.1 Analysis of Experimental Results

In order to validate the proposed approach, and to establish both its scope and limitations, several experimental evaluations where performed.

The first group of experiments where strongly focused in analyzing the novel MCL-based localization strategy, using an appearance-based formulation based on the analysis of the distribution of the relative-bearings. This analysis was challenging due to the inexistence of publicly available implementations of similar VTnR approaches and limited set of suitable datasets, for which only the case of ground-robots repeating a path multiple times were found. Also, as the appearance-based approach using a manifold map representation does not produce a metric pose which can be directly compared to ground-truth pose information, for each acquired dataset a particular strategy was required to obtain measurements regarding localization quality.

From the complete set of localization experiments over pre-recorded datasets, it can be concluded that the appearance-based localization method proposed in this thesis effectively allows for robust localization when suitable visual information exists. Being strongly based on appearance and not on structure, this implies that significant environment changes or low-quality images greatly affect the quality of the localization. Still, under not so challenging conditions, localization errors where mostly around 0.1 and $0.3\,\mathrm{m}$, which is adequate bearing-only navigation and also relatively-low considering the very simple formulation of the approach and the sensors used.

Another set of experiments were focused on the evaluation of the bearing-only control strategy for path-following. The response of the proposed controllers where tested in three stages: with a fixed reference, with a straight path and a curved path.

The first scenario allowed to validate the response in practice of the proposed bearing-only controllers. While response was slow, bearing-only control is not expected to be applied in-place. Still, it allows to anticipate the type of response that would be obtained during path-following. This experiments also allowed to relate visual-variables (the inputs to the bearing-only controllers) to the corresponding individual relative errors w.r.t. the reference. In this sense, while the relative angle can be directly correlated to the value of the mode of azimuth differences, for the case of a vertical offset, this correlation is not so direct. As expected, a translation has the effect of dispersion of elevation difference histogram and thus the mode becomes erratic.

While the mode-based control is still applicable, its limitations become evident. At this point, it could be concluded that better approach would be to directly consider the complete distribution instead of just the mode, however the challenge is then how to deal with bad correspondences.

For the case of straight-path following, initial vertial and lateral offsets w.r.t. the learned path were observed to be diminished by the use of the bearing-only control strategy. In this case, as vertical velocity is directly affected by the proposed controllers, the vertical offset was diminished much more quickly than a lateral offset. As expected from the described principles govering bearing-only navigation, the speed of lateral error reduction is dependant on the distance to the features, when using the bearing-only control strategy.

The third group of path-following experiments where focused on analyzing the behavior in the context of curved paths. Since the proposed bearing-only controller is quite simple (and also based on a P-controller), in order to robustly follow curved paths in the proposed approach, a lookahead was introduced when setting the navigation reference. This lookahead allows to anticipate the path-changes, where the zero-lookahead bearing-only control is not sufficient to achieve correct path-tracking due to the slow response to lateral errors. With the performed experiments, the effectiveness of this strategy is demonstrated, allowing to reduce lateral path-tracking errors but without much effect on vertical errors (which is not really problematic due to the direct control of vertical velocity).

Since the proposed VTnR method includes the ability to follow a network of paths, featuring bifurcations and junctions, experiments were performed to analyze this property as well. However, the case of detecting and mapping junctions was not fully solved since the required task of loop-detection was considered to be out of the scope of this thesis. To analyze the case of mapping and following a bifurcation, a path was first learned and then extended. Localization was found to be able to deal with the bifurcation, only with a very brief pose error, positively demonstrating the validity of the strategies chosen to address this problem (navigation reference computation, particle prediction over the path, etc.). Following each of the bifurcations was shown to be smooth, even in the presence of the slight localization error.

A final set of navigation experiments where performed, in this case focusing on the evaluation of the VTnR approach in an outdoor and uncontrolled scenario. Given the technical challenges in performing this experiment, only proof-of-concept results where able to be obtained by visually inspecting the behavior of the robot during path-following. In general the experiments were performed successfully, only requiring a brief manual-override to avoid colliding with a metal pole, which was very close to the learned path. While the robot followed the path within the width of a narrow sidewalk, this experience speaks about the real challenges in dealing with the problem of autonomous navigation, where robustly and safely dealing with the surrounding objects is inevitably required.

Finally, the computational performance of the method was measured, when executing the implemented system on-board the embedded hardware of the LRSE quadcopter. While with previous experiments the ability to effectively localize and navigate running the proposed VTnR on-board the robot was demonstrated, an analysis of which portions of the complete method are the most computationally demanding was performed. As expected, these tasks correspond to most intensive image-processing steps, which consist in the feature extraction, description and matching, continously performed. It was observed that navigation control-loop (which depends on the image-processing speed) could run at about $15\,\mathrm{Hz}$ in average conditions. While localization is slower at $10\,\mathrm{Hz}$, this frequency corresponds to the update-step of MCL, which has the effect of correcting any accumulated errors during prediction. Since both the localization and navigation image-processing tasks run in parallel, and since the prediction step can produce localization estimates at the same frequency of dead-reckoning based motion estimates, this slower frequency does not limit the response time of the bearing-only controller. Thus, improvements only in the execution speed of the navigation task would have a direct impact in the ability to follow paths at higher speeds.

6.2 Future Work

Based on the obtained results in this thesis, a series of opportunities of improving the proposed method can be identified.

Regarding the proposed bearing-only control strategy, several improvements can be considered. First, while the use of histograms allow for simple and efficient computation of the mode and entropy values, due to this discretization, erratic values were observed in some experiments. A mode-seeking algorithm, such as Mean Shift[59] could be used instead of taking the peak of the histogram, thus avoiding discretization problems. For the entropy measure, a similar strategy could be used, possibly allowing better distinguishing different particles by obtaining a smoother value for their weights. Also, since the main limitation of the mode-based bearing-only control is that the response to lateral and vertical offsets depends on the distance of the features agreeing with the mode, a better approach would possibly involve the use of the entire distribution of bearing differences.

In the case of a vehicle with a precise knowledge of the absolute yaw angle, experiments performed in simulation have demonstrated the feasibility of a controller directly affecting lateral velocity based on visual information. The difficulty remains thus in obtaining this precise yaw angle. This issue is related to an important limitation of the very basic state estimation filters currently used in the autopilot's software-stack. Since at the writing of this thesis, a very much improved autopilot software-stack, based on EKF filters for state estimation, is soon to be released, a significant pending task is to re-visit some of the performed experiments using this new software. It is expected that not only the yaw angle will be much better estimated, but also overall robustness will be achieved when flying both indoors and outdoors by means of velocity-commands based on readings obtained from the PX4FLOW sensor, possibly extending the scenarios in which the LRSE quadcopter can be robustly flown and thus, where the VTnR approach can be evaluated. As a result, a more in-depth experimental analysis in real-world conditions would probably be possible, which is also a pending aspect that follows from the work performed in this thesis.

Another desireable improvement would be to complete the generalization of learning a network of paths, for the case of mapping junctions. This task requires finding and appropriate loop-detection algorithm in order to generate "loop events" which can the be represented in the map as described in the corresponding section. Rought experiments where performed using the DLoopDetector[60] software-library, obtaining promising results. Still, correctly handling false-positives is important and could possibly handled using the same appearance-based principles used for localization, where the number of matches together with entropy of bearing differences allows to discriminate valid from invalid locations.

In terms of computational efficiency, some optimizations could be performed. For example, since navigation and localization tasks are solved in parallel, this currently implies that feature detection and description run once for each image processed in both cases. While localization does not run as frequently as navigation, the resulting redundant image processing is not as significant. Still, as these tasks become faster (for example, on newer and more powerfull versions of the employed embedded hardware), this performance penality becomes a limitation. A simple approach in this case is to distinguish a feature extraction and detection module, which can cache the results obtained for each incoming image, and even perform feature processing on-demand.

In more general terms, possible extensions to the approach can also be identified, in some cases allowing to overcome some of its inherent limitations. For example, instead of a perspective-type camera, a camera with a large field-of-view could be used to obtain much richer visual information in one image. This, of course, would require re-considering the principles of bearing-only navigation, which currently assume a perspective camera. Another possibility would be to directly add a second camera, for example looking sideways or even backwards. This would allow to distinguish lateral errors directly and also repeating a path backwards.

Finally, it would be interesting to consider the applicability of the proposed VTnR navigation method to other alternative platforms, such as fixed-wing aerial vehicles or even legged ground-robots, which would considerably open the spectrum of applications that are possible to face with the proposed approach. In

this sense, it is hoped that the particular strategies chosen in this work to deal with all aspects concerning TnR navigation, will provide a strong foundation to tackle further extensions to the approach, such as the previously described.

Capítulo 6

Conclusiones

En esta tesis se presenta un método de navegación basado en apariencias siguiendo la estrategia de aprendizaje y repetición (*visual teach and* repeat o VTnR) para robots aéreos y terrestres. Este enfoque al problema de la navegación es particularmente útil para escenarios tales como la inspección remota, entrega de paquetes o búsqueda y rescate, donde para resolver la tarea un camino previamente aprendida necesita ser repetido, idealmente sin la intervención humana.

La formulación en base a apariencias se basa en el uso de un grafo topológico en el cual se almacena la información visual, lo que permite resolver los problemas de localización y de navegación sin requerir de la reconstrucción de la estructura del entorno. Además, el enfoque VTnR permite una representación relativa de la pose del robot respecto de una referencia en el mapa, en lugar de utilizar un único sistema de coordenadas privilegiado. Así, el mapa topológico consiste en una representación tipo *manifold* (o variedad) de la ruta aprendida. En consecuencia, el típico problema de la deriva de la pose estimada en el contexto de la localización, que por lo general requiere de enfoques complejos y algoritmos computacionalmente costosos (como *Bundle Adjustment*) para evitar que este error eche a perder el enfoque, es en este caso eludido.

En comparación con otros enfoques existentes a la navegación VTnR, se utiliza una estrategia de localización robusta y eficiente, basada en *Monte Carlo Localization* (MCL). La pose del robot sobre el mapa se estima por medio de una serie de hipótesis de localización (las partículas), que permiten muestrear la distribución de probabilidad desconocida subyacente. Uno de los principales beneficios de este enfoque es que el costo computacional de la tarea de localización es independiente del tamaño del mapa, y sólo depende del número de partículas. Así, es posible equilibrar fácilmente la eficiencia computacional sobre la robustez de localización y precisión. En segundo lugar, con MCL, la resolución de fallas en la localización y de la localización global sin información *a priori* resulta posible, lo que en el contexto de la navegación VTnR se convierte en una habilidad necesaria para poder afrontar escenarios desafiantes del mundo real. Por último, gracias a la formulación del método en base a apariencias, el espacio de búsqueda asociado al problema de la localización es de una sola dimensión, lo que simplifica aún más el problema y minimiza ciertas limitaciones del enfoque MCL (tales como el problema de la privación de partículas o *particle deprivation problem*).

La navegación bajo el enfoque propuesto se resuelve utilizando un simple estrategia muy simple de control basada en rumbos (*bearing-only navigation*). Si bien este enfoque es utilizado ampliamente para el caso de la robots terrestres, pocos trabajos han aplicados esta estrategia a los robots aéreos. Por otra parte, en estos trabajos solo se han propuesto leyes de control conmutadas y los principios básicos subyacentes al enfoque fueron mayormente pasados por alto. En esta tesis, el control visual basado en rumbos se analiza para el caso de un espacio 3D, relacionando este problema con el concepto seguimiento de caminos basado en flujo óptico (*optical-flow*). Además, se presenta una estrategia simple y robusta para emplear simultáneamente múltiples puntos de referencia (*landmarks*) y el caso de obtener muchas correspondencias malas, sin tener que recurrir a enfoques más complejas y no determinísticos de muestreo y consenso (RANSAC), comúnmente utilizados. La estrategia propuesta se basa en el cálculo de las diferencias de los ángulos de acimut (*azimuth*) y elevación (*elevation*) a las referencias y la extracción de las modas estadísticas correspondientes

con el objetivo de guiar la navegación.

Otra contribución considerable en comparación con el enfoque TnR clásico, bajo el cual una única ruta es aprendida y luego repetida completamente, en el método propuesto dicho enfoque se generaliza al considerar las estructuras de datos y los algoritmos que permiten representar una red de caminos interconectados. Como resultado, para alcanzar un objetivo dado en todo el mapa, sólo el sub-conjunto de caminos necesarios pueden ser repetidos para alcanzar la ubicación deseada. Utilizando el enfoque topológico basado en grafos, las bifurcaciones y los cruces pueden ser representados y la localización y la navegación pueden ser resueltas sin necesidad de un mapa globalmente consistente.

Como el objetivo principal es que el método propuesto puede ser ejecutado a bordo de robots aéreos que cuenten con unidades de procesamiento de limitada capacidad computacional, las tareas de procesamiento visual (que son las más exigentes computacionalmente) se basan en el uso de la algoritmos eficientes de extracción de características visuales, descripción y de establecimiento de correspondencias. Por esta razón, para la descripción de una característica visual se utiliza el algoritmo BRIEF, el cual equilibra la eficiencia y la robustez, mientras que para la extracción de características se utiliza el algoritmo GFTT. Aunque esta combinación no ofrece invarianza a movimientos rotacionales, que se requiere para compensar los movimientos realizados por robots aéreos durante el vuelo, se emplea una simple estrategia de desrotación de los descriptores basada en el ángulo de balanceo (*roll*) estimado por el robot. Asimismo, se utiliza esta información para desrotar los valores estimados de flujo óptico, lo que hace posible el uso de los controladores basados en rumbos utilizados para robots terrestres, pero para el caso de robots aéreos.

Junto con el desarrollo del enfoque de navegación VTnR, en esta tesis también se presenta una plataforma robótica aérea experimental, el cuadracóptero LRSE, que fue diseñada y construida con el fin de permitir la evaluación experimental del método de navegación propuesto, dado que es capaz de llevar a bordo el *hardware* de procesamiento y sensado necesarios. La realización de experimentos utilizando esta plataforma, aunque bastante difícil, ha permitido obtener un método de navegación VTnR capaz de afrontar las condiciones del mundo real. Además, tanto el proceso de construcción y experimentación con esta plataforma dieron como resultado una experiencia de aprendizaje muy valiosa. Por otra parte, con el fin de obtener una plataforma estable que pudiera ser operado en forma autónoma mediante comandos de velocidad en ambientes interiores (sin el uso de GPS), se han tenido que realizar cambios considerables al código propio del autopiloto Pixhawk. Muchos de estos cambios fueron compartidos con la comunidad Pixhawk de código abierto, lo que a su vez dio lugar a contribuciones a dicho proyecto en sí mismo.

6.1 Análisis de los resultados experimentales

Con el fin de validar el método propuesto, y para establecer tanto sus alcances como sus limitaciones, se realizaron diversos experimentos.

El primer grupo de experimentos se centró fuertemente en el análisis de la estrategia de localización basada en MCL, que se basa en el análisis de la distribución de las diferencias de rumbos a las referencias. Este análisis fue desafiante debido a la inexistencia de implementaciones de acceso público de enfoques VTnR similares y la falta de conjuntos de datos (*datasets*) adecuados, por lo que se pudieron utilizar únicamente datos adquiridos mediante robots terrestres. Además, como el enfoque basado en apariencias mediante una representación tipo *manifold* no produce una pose métrica que puede ser comparada directamente con datos de verdad absoluta (*ground-truth*), para cada conjunto de datos adquiridos se requirió de una estrategia particular para obtener mediciones respecto de la calidad de la localización.

A partir del conjunto de experimentos realizados sobre datos pre-grabados, se puedo concluir que el método propuesto de localización basado en apariencias permite de manera efectiva y robusta localizar al robot, siempre y cuando exista información visual adecuada. Siendo fuertemente basado en apariencias y no en la reconstrucción de la estructura del entorno, esto implica que cambios significativos en la apariencia del entorno o al utilizar imágenes de baja calidad se afecta en gran medida la calidad de la localización. Aún así, en condiciones no tan desafiantes, los errores de localización en general han sido de entre $0{,}1$ y $0{,}3\,m$, lo

cual es adecuada para la navegación tipo VTnR y particularmente teniendo en cuenta la simple formulación del enfoque y de los sensores utilizados.

Otro conjunto de experimentos se centraron en la evaluación de la estrategia de control basada en rumbos para el seguimiento de caminos. La respuesta de los controladores propuestos se evaluaron en tres etapas: con una referencia fija, con un camino recto y una trayectoria curva.

El primer escenario permitió validar la respuesta en la práctica de los controladores basados en rumbos. Si bien la respuesta fue generalmente lenta, tampoco se buscaba que dicho control fuera suficiente para una referencia fija. Aún así, permitió anticipar el tipo de respuesta que se obtendría durante el seguimiento de caminos. Estos experimentos también permitieron relacionar variables visuales (las entradas a los controladores) a los correspondientes errores relativos individuales con respecto a la referencia. En este sentido, mientras que el ángulo de orientación relativo se puede correlacionar directamente con el valor de la moda de las diferencias de acimut, para el caso de una desviación vertical, esta correlación no es tan directa. Como era de esperar, una traslación tiene el efecto de dispersar el histograma de diferencias de elevación y por lo tanto la moda se vuelve errática. Si bien el control basado en la moda sigue siendo aplicable, sus limitaciones son evidentes. En este punto, se podría concluir que un mejor enfoque consistiría en examinar directamente la distribución completa en lugar de sólo la moda, sin embargo, el reto es entonces cómo hacer frente a las malas correspondencias.

Para el caso de seguimiento de un camino recto, desplazamientos verticales y lateral se introdujeron inicialmente. En este caso, como la velocidad vertical se ve directamente afectada por los controladores propuestos, el desplazamiento vertical disminuyó mucho más rápidamente que un desplazamiento lateral. Como era de esperar a partir de los principios descritos que regulan la navegación basada en rumbos, la velocidad de la reducción de error lateral depende de la distancia a las características.

El tercer grupo de experimentos se centró en el análisis del comportamiento de seguimiento de caminos en el contexto de una trayectoria curvas. Dado que el controlador basado en rumbos es bastante simple (y también basado únicamente en un controlador P), con el fin de seguir en forma robusta una trayectorias curva en el enfoque propuesto, se introdujo la noción de un *lookahead* a la hora de establecer la referencia de navegación. Esto permite anticiparse a los cambios del camino, donde el control original no es suficiente para lograr el correcto seguimiento de la ruta debido a la lenta respuesta a los errores laterales. Con los experimentos realizados, la eficacia de esta estrategia se demuestra, lo que permite reducir los errores laterales, sin mucho efecto sobre errores verticales (lo que no es realmente un problema debido al control directo de la velocidad vertical).

Dado que el método propuesto posee la capacidad de seguir una red de caminos, con bifurcaciones y cruces, se realizaron experimentos para analizar esta propiedad también. Sin embargo, el caso de la detección y representación de cruces no se resolvió completamente ya que la tarea requerida de detección de ciclos se consideró que quedaba fuera del alcance de esta tesis. Para analizar el caso de la representación y seguimiento de una bifurcación, se aprendió un camino que luego fue extendido. Se encontró que la localización era capaz de hacer frente a la bifurcación, sólo con un error de pose muy breve, lo que demuestra positivamente la validez de las estrategias elegidas para abordar este problema (cómputo de referencia de navegación, la predicción de las partículas sobre la trayectoria, etc.). El seguimiento de las bifurcaciones resultó ser suave, incluso en la presencia del error de localización ligera.

Un último conjunto de experimentos de navegación fue llevado a cabo, en este caso centrándose en la evaluación del enfoque VTnR en un escenario al aire libre y no estructurado. Teniendo en cuenta los desafíos técnicos en la realización de este experimento, los resultados consistieron mayormente en una prueba de concepto, dado que se observó el comportamiento del método principalmente mediante la inspección visual durante el seguimiento de camino. En general, los experimentos se llevaron a cabo con éxito, requiriendo únicamente la toma de control manual del vehículo durante un breve momento para evitar la colisión con un poste de metal, que estaba muy cerca de la ruta aprendida. Esta experiencia habla sobre los desafíos que surgen al tratar el problema de la navegación autónoma en ambientes reales, donde se requiere un método robusta y seguro frente a un entorno incierto.

Finalmente, se midió el rendimiento computacional del método, al ejecutar el sistema implementado

a bordo del *hardware* integrado del cuadracóptero LRSE. Con los experimentos anteriores, se demostró la capacidad de localizar y navegar utilizando el enfoque propuesto mientras ejecutaba a bordo del vehículo. En este caso, se realizó además un análisis de qué porciones del método completo son los más computacionalmente demandantes. Como era de esperar, dichas tareas corresponden a la mayoría de los pasos de procesamiento de la imagen, que consisten en la extracción de características, su descripción y su seguimiento en forma continua. Se observó que el bucle de control de la navegación (que depende de la velocidad de procesamiento de imágenes) pudo funcionar a aproximadamente $15\,Hz$ en promedio. Si bien la localización es más lenta (unos $10\,Hz$), esta frecuencia corresponde al paso de actualización de MCL, que tiene el efecto de corregir cualquier error acumulado durante la predicción. Dado que ambas las tareas de procesamiento de imágenes de localización y de navegación son resueltas en paralelo, y dado que la etapa de predicción puede producir estimaciones de localización a la misma frecuencia que las estimaciones de movimiento, esta frecuencia más lenta no limita el tiempo de respuesta del controlador. Por lo tanto, las mejoras sólo en la velocidad de ejecución de la tarea de navegación tendrían un impacto directo en la capacidad de seguir caminos a velocidades más altas.

6.2 Trabajo Futuro

Sobre la base de los resultados obtenidos en esta tesis, se puede identificar una serie de oportunidades de mejora del método propuesto.

Con respecto a la estrategia de control propuesta, varias mejoras se pueden considerar. En primer lugar, mientras que el uso de histogramas permite para el cálculo simple y eficiente de los valores de la moda y la entropía, debido a esta discretización, se observaron valores erráticos en algunos experimentos. Un algoritmo de búsqueda de modas, tal como Mean Shift[59] se podría utilizar en lugar de tomar el pico del histograma, evitando así dichos problemas de discretización. Para la medida de entropía, una estrategia similar podría utilizarse, posiblemente permitiendo distinguir mejor a las partículas mediante la obtención de un valor más suave para sus pesos. También, puesto que la principal limitación de control propuesto en base a las modas, es que la respuesta a los desplazamientos laterales y verticales depende de la distancia de las características que coinciden con la moda, un mejor enfoque posiblemente implique el uso de toda la distribución de diferencias.

En el caso de un vehículo con un conocimiento preciso del ángulo absoluto de guiñada (*yaw*), los experimentos realizados en simulación han demostrado la viabilidad de un controlador que afecte directamente a la velocidad lateral sobre la base de la información visual. La dificultad consiste entonces en la obtención de este ángulo de orientación precisa. Este problema está relacionado con una importante limitación de los filtros de estimación de estado muy básicos que se utilizan actualmente en el software del autopiloto. Dado que al momento de escritura de esta tesis, una versión considerablemente mejorada del software del autopiloto, basado en filtros EKF para la estimación del estado, se encuentra próxima a ser lanzada, una tarea pendiente significativa es volver sobre algunos de los experimentos llevados a cabo utilizando este nuevo software. Se espera que no sólo el ángulo de orientación será mucho mejor estimado, sino que también se logrará mayor robustez en general al volar tanto en ambientes interiores como al aire libre utilizando comandos de velocidad basándose en las lecturas obtenidas del sensor PX4FLOW, posiblemente extendiendo así los escenarios en los que la cuadracóptero LRSE puede ser volado y, por lo tanto, donde el enfoque VTnR puede ser evaluado. Como resultado, un análisis experimental en mayor profundidad y en condiciones reales se haría posible, lo cual es también un aspecto pendiente que se desprende de los trabajos realizados en esta tesis.

Otra mejora deseable sería completar la generalización del aprendizaje de una red de caminos, para el caso de los cruces en el camino. Esta tarea requiere un algoritmo de detección de bucles adecuado con el fin de generar “eventos bucle" que puedan ser representados en el mapa como se describe en el apartado correspondiente. En este sentido se realizaron experimentos iniciales utilizando el software DLoopDetector[60] , obteniendo resultados prometedores. Aún así, manejar correctamente los falsos positivos es importante lo que posiblemente podría ser resuelto utilizando los mismos principios basados en apariencias utiliza-

dos para la localización, donde el número de correspondencias junto con la entropía permiten discriminar ubicaciones válidas de las no válidas.

En términos de eficiencia computacional, se podrían realizar también algunas optimizaciones. Por ejemplo, ya que las tareas de navegación y de localización se resuelven en paralelo, en la actualidad esto implica que la detección de características visuales y su descripción son tareas que se ejecutan una vez para cada imagen procesada en ambos casos. Si bien la localización no se ejecuta con la misma frecuencia que la navegación, el procesamiento redundante de la imagen resultante no es tan significativo. Aún así, a medida que dichas tareas ejecuten a mayor velocidad (por ejemplo, utilizando las nuevas versiones del hardware que fue utilizado para el procesamiento a bordo), esta penalidad en el rendimiento se convierte en una limitación. Un enfoque simple en este caso es utilizar un módulo de extracción de características y de detección por separado, que puede almacenar en caché los resultados obtenidos para cada imagen de entrada, e incluso realizar el procesamiento de función bajo demanda.

En términos más generales, posibles extensiones del enfoque propuesto también se pueden identificar, permitiendo en algunos casos superar algunas de sus limitaciones inherentes. Por ejemplo, en lugar de utilizar una cámara de tipo perspectiva, una con un gran campo de visión (lente gran angular) podría ser utilizada para obtener información visual mucho más rica en una sola imagen. Para esto, por supuesto, sería necesario volver a considerar los principios en los cuales se basa la navegación, dado que en la actualidad se asume una cámara perspectiva. Otra posibilidad sería añadir directamente una segunda cámara, por ejemplo mirando hacia los lados o incluso hacia atrás. Esto permitiría distinguir errores laterales directamente y también la repetición de un camino hacia atrás.

Por último, sería interesante considerar la aplicabilidad del método de navegación propuesto a otras plataformas alternativas, tales como vehículos aéreos de ala fija o incluso robots artrópodos (de patas articuladas), lo que abriría considerablemente el espectro de aplicaciones que son posibles con el enfoque propuesto. En este sentido, se espera que las estrategias particulares elegidas en este trabajo para hacer frente a todos los aspectos relativos a la navegación VTnR, proporcionen una base sólida para hacer frente a nuevas extensiones para el enfoque, como los descritos anteriormente.

Bibliography

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM." *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 1052–1067, 2007. 1

[2] G. Klein and D. Murray, "Parallel Tracking and Mapping on a camera phone," *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009. 1

[3] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, no. April. IEEE, 1996, pp. 83–88. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=503577http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=503577 1.2, 1.4, 1.2

[4] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative Learning Control to improve mobile robot path tracking in challenging outdoor environments," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–181, nov 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6696350 1.3, 1.3

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1077314207001555 1.3, 1.4, 1.3

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, no. [8, 1999, pp. 1150–1157. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410 1.3, 1.3

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6314 LNCS, no. PART 4, 2010, pp. 778–792. 1.3, 1.3, 3.2.1

[8] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2548–2555. 1.3, 1.3

[9] R. Brooks, "Visual map making for a mobile robot," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. Institute of Electrical and Electronics Engineers, 1985, pp. 824–829. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1087348 1.4

[10] a. Howard, G. Sukhatme, and M. Mataric, "Multirobot Simultaneous Localization and Mapping Using Manifold Representations," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1360–1369, jul 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1677949http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1677949 1.4

[11] P. Newman, J. Leonard, J. Tardos, and J. Neira, "Explore and return: experimental validation of real-time concurrent mapping and localization," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, no. May. IEEE, 2002, pp. 1802–1809. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1014803 1.4

[12] J. Marshall, T. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 400–421, jun 2008. [Online]. Available: http://doi.wiley.com/10.1002/rob.20242 1.4

[13] P. Krüsi, B. Bücheler, F. Pomerleau, U. Schwesinger, R. Siegwart, and P. Furgale, "Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching," *Journal of Field Robotics*, vol. 32, no. 4, pp. 534–564, jun 2015. [Online]. Available: http://doi.wiley.com/10.1002/rob.21524 1.4

[14] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual Navigation for Mobile Robots: A Survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, nov 2008. [Online]. Available: http://link.springer.com/article/10.1007/s10846-008-9235-4$\delimiter"026E30F$nfiles/639/Bonin-Fontetal.-2008-VisualNavigationforMobileRobotsASurvey.pdf$\delimiter"026E30F$nfiles/640/s10846-008-9235-4.htmlhttp://link.springer.com/10.1007/s10846-008-9235-4 1.4

[15] E. Baumgartner and S. Skaar, "An autonomous vision-based mobile robot," *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 493–502, mar 1994. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=280748 1.4

[16] Y. Matsumoto and K. Ikeda, "Visual navigation using omnidirectional view sequence," *Intelligent Robots and ...*, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=813023 1.4

[17] T. Ohno, A. Ohya, and S. Yuta, "Autonomous navigation for mobile robots referring pre-recorded image sequence," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 2. IEEE, 1996, pp. 672–679. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=571034 1.4

[18] L. Tang and S. Yuta, "Vision based navigation for mobile robots in indoor environment by teaching and playing-back scheme," in *International Conference on Robotics and Automation*, 2001, pp. 3072–3077. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=933089 1.4

[19] K. Bekris, A. Argyros, and L. Kavraki, "Exploiting panoramic vision for bearing-only robot homing," *Imaging beyond the pinhole camera*, vol. 33, pp. 229–251, 2006. [Online]. Available: http://link.springer.com/content/pdf/10.1007/978-1-4020-4894-4{_}12.pdf 1.4

[20] Jianbo Shi and Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. IEEE Comput. Soc. Press, 1994, pp. 593–600. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=323794 1.4, 3.2.1

[21] Z. Chen and S. Birchfield, "Qualitative vision-based mobile robot navigation," *Robotics and Automation, 2006. ICRA ...*, pp. 2686–2692, 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1642107http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1642107 1.4

[22] ——, "Qualitative vision-based path following," *Robotics, IEEE Transactions on*, vol. X, no. X, pp. 1–6, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4814552 1.4

[23] Z. Chen and S. T. Birchfield, "Vision-Based Path Following Without Calibration," vol. 25, pp. 749–754, 2010. 1.4

[24] A. M. Zhang and L. Kleeman, *Robust Appearance Based Visual Route Following for Navigation in Large-scale Outdoor Environments*, 2009, vol. 28. 1.4

[25] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular Vision for Mobile Robot Localization and Autonomous Navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, jan 2007. [Online]. Available: http://link.springer.com/10.1007/s11263-006-0023-y 1.4

[26] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool, "Omnidirectional vision based topological navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 219–236, 2007. 1.4

[27] O. Booij, B. Terwijn, Z. Zivkovic, and B. Kröse, "Navigation using an appearance based topological map," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 3927–3932, 2007. 1.4

[28] S. Šegvić, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Computer Vision and ...*, vol. 2, no. July 2008, pp. 172–187, feb 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S107731420800129Xhttp://linkinghub.elsevier.com/retrieve/pii/S107731420800129X 1.4

[29] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, no. 2006, pp. 1–27, 2010. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/rob.20342/full 1.4

[30] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Preucil, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, pp. 511–533, jul 2010. [Online]. Available: http://doi.wiley.com/10.1002/rob.20354 1.4, 1.5

[31] L. E. Clement, J. Kelly, and T. D. Barfoot, "Monocular Visual Teach and Repeat Aided by Local Ground Planarity," in *Field and Service Robotics*, 2015, pp. 1–14. 1.4

[32] J. Courbon, Y. Mezouar, N. Guénard, and P. Martinet, "Vision-based navigation of unmanned aerial vehicles," *Control Engineering Practice*, vol. 18, no. 7, pp. 789–799, 2010. 1.4

[33] J. Courbon, Y. Mezouar, and P. Martinet, "Indoor navigation of a non-holonomic mobile robot using a visual memory," *Autonomous Robots*, vol. 25, no. 3, pp. 253–266, 2008. [Online]. Available: http://link.springer.com/article/10.1007/s10514-008-9093-8 1.4

[34] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Procedings of the Alvey Vision Conference 1988*, pp. 147–151, 1988. [Online]. Available: http://www.bmva.org/bmvc/1988/avc-88-023.html 1.4

[35] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera," in *Proceedings - Conference on Computer and Robot Vision, CRV 2014*. Ieee, may 2014, pp. 238–245. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6816849http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6816849 1.4

[36] T. Nguyen, G. Mann, and R. Gosine, "Vision-Based Qualitative Path-Following Control of Quadrotor Aerial Vehicle with Speeded-Up Robust Features," *Computer and Robot Vision ...*, pp. 321–327, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6816860 1.4

[37] J. Martínez-Carranza, R. Bostock, S. Willcox, I. Cowling, and W. Mayol-Cuevas, "Indoor MAV auto-retrieval using fast 6D relocalisation," *Advanced Robotics*, pp. 1–12, 2015. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/01691864.2015.1094409 1.4

[38] T. Do, L. C. Carrillo-arce, and S. I. Roumeliotis, "Autonomous Flights through Image-defined Paths," pp. 1–16, 2015. 1.4

[39] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 510–517, 2012. 1.4

[40] D. Nistér and H. Stewénius, "Scalable Recognition with a Vocabulary Tree." [Online]. Available: http://www.vis.uky.edu/ 1.4

[41] T. Krajnik, M. Nitsche, S. Pedre, and L. Preucil, "A simple visual navigation system for an UAV," in *Systems, Signals and Devices*, 2012, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6198031 1.5, 1.4, 5

[42] M. Nitsche, T. Pire, T. Krajník, M. Kulich, and M. Mejail, "Monte Carlo Localization for Teach-and-Repeat Feature-Based Navigation," *Advances in Autonomous Robotics Systems*, pp. 13–24, 2014. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10401-0{_}2 1.5, 1.4, 5

[43] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999. 1.5, 1.4

[44] D. Scaramuzza, "1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints," *International Journal of Computer Vision*, vol. 95, no. 1, pp. 74–85, apr 2011. [Online]. Available: http://link.springer.com/10.1007/s11263-011-0441-3 2.3.1

[45] T. Krajnik, P. De Cristóforis, J. Faigl, H. Szücsová, M. Nitsche, L. Preucil, and M. Mejail, "Image Features for Long-Term Mobile Robot Autonomy," *ICRA 2013 Workshop on Long-Term Autonomy*, 2013. [Online]. Available: http://robotica.dc.uba.ar/public/papers/icra2013.pdf 3.2.1

[46] T. Krajnik, P. De Cristóforis, M. Nitsche, K. Kusumam, and T. Duckett, "Image features and seasons revisited," in *Mobile Robots (ECMR), 2015 European Conference on*. IEEE, 2015, pp. 1–7. 3.2.1

[47] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5305 LNCS, no. PART 4, 2008, pp. 102–115. 3.2.1

[48] D. Sabatta, "Bearings-only Path Following with a Vision-based Potential Field," pp. 2755–2760, 2014. 3.4.4

[49] W. Maddern, M. Milford, and G. Wyeth, "Towards Persistent Localization and Mapping with a Continuous Appearance-Based Topology," *Robotics: Science and Systems VIII*, vol. 2012, pp. 4224–4230, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6577940 3.4.5

[50] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005. 3.4.7

[51] R. Hogg, a.L. Rankin, S. Roumeliotis, M. McHenry, D. Helmick, C. Bergh, and L. Matthies, "Algorithms and sensors for small robot path following," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, no. May, pp. 3850–3857, 2002. 3.5.3

[52] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407. 5.1.1

[53] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009. [Online]. Available: http://pub1.willowgarage.com/{~}konolige/cs225B/docs/quigley-icra2009-ros.pdf 5.1.5

[54] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets: Open-source library and experimental protocol," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013. 5.3.1

[55] L. Murphy, T. Morris, U. Fabrizi, M. Warren, M. Milford, B. Upcroft, M. Bosse, and P. Corke, "Experimental comparison of odometry approaches," in *International Symposium on Experimental Robotics*, 2012. 5.3.2

[56] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007. 5.3.2

[57] P. Pfaff, W. Burgard, and D. Fox, "Robust Monte-Carlo localization using adaptive likelihood models," *Springer Tracts in Advanced Robotics*, vol. 22, pp. 181–194, 2006. 5.3.2

[58] J. Bruce, J. Wawerla, and R. Vaughan, "The SFU Mountain Dataset: Semi-Structured Woodland Trails Under Changing Environmental Conditions," in *IEEE Int. Conf. on Robotics and Automation 2015, Workshop on Visual Place Recognition in Changing Environments*, 2015. 5.3.3

[59] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995. 6.2

[60] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012. 6.2

Appendix A

VTnR Method Parameters

Parameter	Default	Unit	Range	Significance
v_{teach}	0.3	$\frac{m}{s}$	$(0, \infty)$	Forward velocity used during teach phase
v_{repeat}	0.3	$\frac{m}{s}$	$(0, \infty)$	Forward velocity used during repeat phase
ϵ_s	0.2	m	$(0, \infty)$	Maximum segment length, used to decide when to finish the current segment
ϵ_v	0.05	m	$(0, \infty)$	Distance traveled since last view, to decide when a new one should be added to a landmark
$l_{\text{MAX\_NOT\_SEEN}}$	0.5	s	$[0, \infty)$	Maximum time a landmark is held in the STM
σ	0.1		$[0, \infty)$	Standard-deviation of the Gaussian additive noise term used in the MCL motion model
k_1	0.03	-	$(0, \infty)$	Proportional-term constant of heading controller
k_2	0.1	-	$(0, \infty)$	Proportional-term constant of altitude controller
$r_{\dot{\Theta}}$	60	$^\circ$	$(0, \infty)$	Extent of values covered by the complete elevation differences histogram
$r_{\dot{\Phi}}$	60	$^\circ$	$(0, \infty)$	Extent of values covered by the complete azimuth differences histogram
$\tau_{\dot{\Theta}}$	1.3	$^\circ$	$(0, \infty)$	Extent of values covered by a single bin in the elevation differences histogram
$\tau_{\dot{\Phi}}$	1.3	$^\circ$	$(0, \infty)$	Extent of values covered by a single bin in the azimuth differences histogram
k_{min}	5	-	$[0, \infty)$	Minimum number of feature matches to consider the weight of a particle as valid
H'_{min}	0.5	-	$[0, 1)$	Minimum ammount of inv. entropy to consider the weight of a particle as valid
α_{slow}	0.015	-	$(0, 1)$	Slower low-pass filter coefficient for mean of MCL particle weights
α_{fast}	0.1	-	$(0, 1)$	Faster low-pass filter coefficient for mean of MCL particle weights
δ_{min}	0.9	-	$[0, 1]$	Minimum required density to consider the MCL single-hypothesis as valid
$\tau_{\mathbf{x}^*}$	0.5	m	$(0, \infty)$	Local-graph size around highest-density particle to compute the single-hypothesis
$\tau_{\hat{\mathbf{x}}}$	1	m	$(0, \infty)$	Local-graph size around current pose, used to find navigation reference
r_{min}	0.01	-	$[0, 1)$	Minimum proportion of particles to be randomly re-sampled during MCL update step
M	50	-	$(0, \infty)$	Number of particles used for localization
$\epsilon_{\mathbf{g}}$	9	σ	$[0, \infty)$	Goal reached threshold (obtained as n-times the motion model noise term)
l	0.5	s	$[0, \infty)$	Lookahead time used to find the navigation reference
$F_{\text{max\_descr\_dist}}$	60		$[0, 255]$	Maximum distance in descriptor-space to allow matching a pair of features
$F_{\text{min\_feat\_dist}}$	10		$(0, \infty)$	Minimum distance in pixels allowed between any pair of detected features
$F_{\text{dist\_ratio}}$	1.3		$[1, \infty)$	Ratio of allowed distance in descriptor-space, between 1st and 2nd nearest-neighbor match
$F_{\text{max\_features}}$	70		$(0, \infty)$	Maximum number of features to be detected in an image
$F_{\text{min\_quality}}$	0.01		$(0, \infty)$	GFTT feature-detector minimum feature quality threshold

Table A.1: Parameters involved in the VTnR method, along their units, meaning and default values