Freie Universität Berlin

Outdoor Visual Navigation of a Smart-Wheelchair with the Visual Compass Algorithm

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.) am
Fachbereich
Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

José Antonio Álvarez Ruiz

Berlin
2020

Betreuer:	Prof. Dr. Raúl Rojas González Dahlem Center for Machine Learning and Robotics Institut für Mathematik und Informatik Freie Universität Berlin
Erstgutachter:	Prof. Dr. Raúl Rojas González Dahlem Center for Machine Learning and Robotics Institut für Mathematik und Informatik Freie Universität Berlin
Zweitgutachter:	Prof. Dr. Marco Block-Berlitz Fakultät Informatik/Mathematik Hochschule für Technik und Wirtschaft Dresden
Tag der Disputation:	10. März 2021

Selbständigkeitserklärung

Hiermit versichere ich, dass ich alle Hilfsmittel und Hilfe angegeben habe und auf dieser Grundlage die Arbeit selbstständig verfasst habe.

Ich erkläre weiterhin, dass die Arbeit nicht schon einmal in einem früheren Promotionsverfahren eingereicht wurde.

Berlin, den 22. Juni 2020
José Antonio Álvarez Ruiz

Summary

Smart-wheelchair navigation is a daunting problem even when autonomous cars are slowly taking on the roads. Smart-wheelchairs ought to navigate in highly unstructured and diverse indoor and outdoor conditions that lack the conventions that simplify autonomous car driving. Furthermore, smart-wheelchairs have astringent limitations to the size, weight, energetic consumption, and cost of the components used for autonomous navigation. Despite the challenge, the development of smart assistive technology is necessary because the number of users is likely to rise due to demographic tendencies.

This dissertation explores the use of the visual compass algorithm VCA for outdoor route-following with a smart-wheelchair. The VCA infers the travel direction by aligning the current-view with snapshots stored in a visual memory. Image matching is performed using simple IDFs on low-resolution panoramic images, as inspired by insect navigation findings.

The VCA is an attractive alternative because it only requires an omnidirectional camera and modest computational resources. Nevertheless, the algorithm suffers from well-known limitations, and its performance for tasks like smart-wheelchair navigation is poorly understood because most studies evaluated have focused on highly controlled conditions.

This investigation presents three extensions to the VCA. The first one is the use of *texture-based* representations of the panoramic images to reduce aliasing. Then, the TVCC, which enables the use of multiple representations and IDFs. The third extension introduces KT, i.e., using visual memories to navigate with a different setup than the one used to acquire them, which is an essential aspect of large-scale applications. Furthermore, this dissertation presents an algorithm to correct off-route translational error without resorting to multiscale analysis or probatory movements.

Zusammenfassung

Autonomes Rollstuhlfahren ist ein extrem komplexes Problem. Besonders zu dem Zeitpunkt, an dem autonome Autos langsam die Straßen übernehmen. Autonome Rollstühle sollen in sehr unstrukturierten und diversen Umgebungen (innen und außen) fahren können. Solche Umgebungen bieten oftmals nicht, die expliziten Regeln und Infrastrukturen, die das Autofahren vereinfachen. Dazu sollten, autonome Rollstühle klein, leicht, energie-effizient und günstig bleiben, damit Menschen, die einen benötigen, sich einen leisten können. Deshalb, sind manche Komponenten, die zurzeit in autonomen Autos verwendet werden, nicht für autonome Rollstühle geeignet. Trotz dieser Herausforderung, ist es nötig, die Entwicklung intelligenter unterstützender Technologie anzutreiben, weil die Benutzeranzahl aufgrund demographischer Tendenzen steigen wird.

Diese Dissertation befasst sich mit der Anwendung des „Visual Compass Algorithm (VCA)“ für Außen Route-folgen. Der Algorithmus berechnet die Fahrtrichtung durch die Anpassung zweier panoramischer Bilder. Eins davon, wird am Zielort im Voraus aufgenommen, während das zweite eine aktuelle Beobachtung des Roboters darstellt. Die Bildanpassung erfolgt durch eine simple “Image Distance Function (IDF)”, in niedriger Auflösung panoramischer Bilder. Das Verfahren wurde von Forschungsergebnissen aus der Insektennavigation inspiriert.

Der VCA ist attraktiv für autonome Rollstühle, weil er nur zwei panoramische Bilder für die Berechnungen braucht, und weil er nur eine niedrige Rechenleistung benötigt, um die Echtzeitlaufzeit zu erreichen. Allerdings, hat das VCA mehrere Einschränkungen. Diese Dissertation stellt drei Erweiterungen für das VCA dar. In der ersten Erweiterung, werden panoramische Bilder anhand einer Textur repräsentiert, um den Aliasing-Effekt zu reduzieren. Die zweite Erweiterung ermöglicht multiple Repräsentationen der panoramischen Bilder und der IDF zu kombinieren. Die dritte Erweiterung “Knowledge Transfer” (KT), ermöglicht es zu navigieren, wenn die “Visual Memories” und “Current-Views” von verschiedenen Kamerakonfigurationen aufgenommen werden. KT wird als wichtige Voraussetzung für Großanwendungen dargestellt. Diese Dissertation präsentiert also einen Algorithmus, mit dem man Translationsfehler korrigieren kann.

To my mother and grandparents

Acknowledgements

I thank Prof. Dr. Raúl Rojas for accepting me into his group and mentoring me while also allowing me to explore and learn by myself. His insights, experience, dedication, and passion were a constant source of inspiration, motivation, and learning.

I also thank Mrs. Schöttker-Söhl for all of her support over the years. Prof. Dr. Marco Block-Berlitz and Dr. Bingyi Cao get my gratitude for being part of the academic commission in my thesis defense. Thanks, Prof. Dr. Tim Landgraf, for the many fruitful discussions on insect navigation and for being part of the academic commission. Thanks, Prof. Dr. Daniel Göhring, for always having an open ear for me.

This investigation had not been possible without the kind funding of the Deutscher Akademischer Austausch Dienst (DAAD) during the first years of my studies. The rest of the time, I covered my expenses by working part-time. I deeply thank my work colleagues for all their support and understanding.

Being a Ph.D. student was not always easy. Thank you very much, Catalina and Roswita, for helping me grow from the difficulties that emerged during the process. Also, a big thank you to Mr. Heß and Mrs. Pasaneck from the Graduate Studies Office for their kind support with the administrative aspects of my studies.

It was a great honor and a pleasure to share the joys and perils of life as a Ph.D. student with Margarita Diaz, Omar Mendoza, Fernando Wario, Salvador Sierra, Daniel Neumann, and Ricardo Carrillo. In a way, it was like having a small family away from home.

I am a very fortunate person for having many wonderful friends. Thank you very much, Alicia, Anastasija, Christina, Daria, Denitsa, Fernando, Giulia, Julia, Manuel, Nathalia, Lidia, Maria, Nicole, Patrick, Sarah, Stephi, Tetsushi, and Zhanna.

My girlfriend Caroline gets all my gratitude for the love, care, motivation, patience, and all the delicious cake. The last stages of my Ph.D. studies were quite challenging, but you always kept me going. Thank you very much for being such a beautiful person.

Last but not least, I thank my family for believing in me and encouraging me to follow my curious nature since I can remember.

Contents

Chapter 1

Introduction

The ability to travel autonomously is the defining characteristic of mobile robots. Progress in this area has been significant in the last decades, due in part to the growing interest in the development of autonomous cars. However, autonomous navigation is still a challenging problem, especially when the technology used for it is limited [1], and when highly unstructured and diverse environments are involved.

Vision-based navigation offers an attractive alternative because cameras provide dense measurements of the environment. Nevertheless, they are inexpensive, lightweight, compact, and require little power to operate compared to other types of sensors. Unfortunately, extracting navigation relevant information from raw images is challenging due to a myriad of nuances.

This investigation addresses the problem of visual route-following in an application, where highly unstructured and diverse environments are the norm, and in which the benefits of a vision-based navigation system are particularly attractive: smart-wheelchair outdoor navigation.

The results presented are based on the VCA, formulated initially to study visual navigation in insects (Zeil et al., 2003). In its pure form, the VCA estimates the relative orientation of two low-resolution panoramic images captured at nearby locations. It provides two outputs: a relative heading and a similarity measure that correlates locally with the spatial distance between the locations where the images were captured. A robot can use that information to home to a spatial location (Zeil et al., 2003) and follow a route employing consecutive homing operations to intermediate waypoints (Smith et al., 2008).

The VCA has the advantage of being relatively simple: It does not require object identification or other high-level cognitive capabilities. Additionally, it executes with low latency, even on modest hardware. The evaluation was performed on data acquired using the smart-wheelchair ROSStuhl, developed as part of the author's doctoral studies, and on simulated data where appropriate.

[1]For example, highly accurate GPS and range sensors greatly simplify the problem but are not adequate for all applications.

1.1 The Need for Smart-Wheelchairs

The exact reasons why a person requires a wheelchair are complex and diverse. However, they can be summarized into a single one: that person lacks mobility. That lack of mobility has profound consequences because it reduces a person's ability to engage in ADL, and makes them dependent on other's assistance. For example, let us consider financial implications as an example. People with mobility limitations often find themselves in a dire situation, because they are twice as likely to be unemployed, especially if their disability is chronic and severe. For example, male-households under disability suffer a 68 % income decrease within ten years of disability onset, and also reductions in after-tax income and food consumption. As of 2011, most persons with disabilities lived in rural areas in developing countries (Meyer and Mok, 2019; World Health Organization and others, 2011).

Smart-wheelchairs and other smart-assistance technologies will become increasingly important because the number of potential users is expected to increase. The main reasons behind are the aging of the world's population and the prevalence of mobility-limiting conditions[2] in the elderly. The current demographic trends indicate that their numbers, and the proportion of the population they represent, will double globally by 2050 (Pollack, 2005), see Table 1.1.

Country	percentage in 2000	percentage by 2050
Japan	23.3	42.4
India	7.5	20.1
Italy	24.1	40.6
Germany	23.2	34.5
USA	16.1	25.5
Mexico	6.9	26.2
Colombia	6.9	22.7
Botswana	4.2	6.0
Ethiopia	4.6	7.7
Brazil	7.8	25.9
Fiji	5.7	22.7
Jordan	4.6	19.0

Table 1.1: Percentage of population over 60 compared between 2000 and 2050 (Pollack, 2005).

Traditional-wheelchairs[3] offer only a partial solution because there are persons who cannot use them. Intuition may suggest that the inability to use a traditional wheelchair occurs only in the most severe cases, but moderate ones

[2]Like Alzheimer disease, Parkinson disease, amyotrophic lateral sclerosis, and multiple sclerosis

[3]*Traditional-wheelchairs* depend on a human operator to navigate. *Manual-wheelchairs* rely entirely on the user (using his arms) for steering and propulsion. In contrast, *power-wheelchairs* have motors that are controlled by the user through a control interface, like a joystick (Leaman and La, 2017).

can be sufficient. Those include motor disabilities (like fatigue and weakness), visual disabilities (like visual field neglect or loss), and cognitive ones (executive reasoning deficits or impaired attention). Smart-wheelchairs could improve the outlook of persons with disabilities who cannot operate a traditional-wheelchair because they navigate without the intervention of the user. In the US, the number of potential users for such systems due to a medical condition was estimated to be approximately two million in 2008 (Simpson et al., 2008). At the same time, it would reduce the demand for caretakers, which are insufficient, as 20 to 40 % of the persons with disabilities do not receive adequate assistance, even in wealthy countries (Meyer and Mok, 2019).

1.2 Smart-Wheelchair in the Age of Autonomous Cars

Smart-wheelchairs and autonomous cars are related problems because they share a common goal: to produce "intelligent" vehicles that navigate by themselves. However, both applications differ greatly in many ways, when seen in detail. Regarding the driving conditions, smart-wheelchairs must operate in less structured and more diverse environments than autonomous cars. A fully autonomous car would have to drive mostly in highways, cities, rural areas, and parking lots. It is still a daunting problem, but in general, they can be expected to operate under well-established traffic rules and with infrastructure especially crafted for that purpose. In the case of fully functional smart-wheelchairs, they would have to cover environments as diverse as sidewalks, restaurants, elevators, hospitals, metro stations, supermarkets, as well as boarding in and out of public transportation systems. The features in those environments vary greatly, and the conventions that simplify navigation of autonomous cars, e.g., lane markings (or lanes whatsoever), traffic signs, and rules, cannot be taken for granted. There are also important differences regarding the users. Perhaps the most remarkable is that the availability of a safety-driver, an assumption still present in autonomous cars, would not be suitable for smart-wheelchairs. As discussed earlier in this chapter, the main reason why people would require a smart-wheelchair is that they cannot operate a traditional-wheelchair themselves. Hence, some smart-wheelchair users would not be reliable safety drivers. Completely eliminating any human-provided safety is perhaps not foreseeable in the near future. However, it would have to occur at some point so that smart-wheelchairs fully serve the people who need them most.

Moreover, the equipment suitable for autonomous driving is more limited in smart-wheelchairs, because they have less space and power to host them, and because the footprint of the smart-wheelchair should remain small enough to navigate through narrow areas. Deployment is yet another major difference. While shared autonomous cars are seen as an attractive possibility to reduce the number of vehicles on the streets (Lang and Mohnen, 2019), smart-wheelchairs would be more useful if each user had a dedicated smart-wheelchair. Consequently, the overall system cost has to remain low, mainly because, as discussed earlier, people with disabilities often find themselves in a dire financial situation.

Given the above, smart-wheelchair navigation can be considered to be a more challenging problem than autonomous driving for cars. Despite recent testing of smart-wheelchairs, one in a hospital in Singapur, and one in an airport in Japan (Scudellari, 2017), smart-wheelchair technology has failed to reached widespread use (Leaman and La, 2017). According to Narin et al. (2018) this is not only due to the problem being more difficult than self-driving cars but also because of a mismatch between the research community agenda and the real needs and desires of smart-wheelchair users. The authors suggest lowering the short-term expectations of smart-wheelchairs in order to attain easier yet useful goals. Recent developments even advocate for a cooperative approach between the autonomous navigation system and the smart-wheelchair user (Fearn et al., 2019).

1.3 Problem Statement

Route-following using the VCA is often posed as successive homing operations to intermediate waypoints (Smith et al., 2007; Stone et al., 2018). The problem can be decomposed into two complementary subtasks: localization and homing. Localization consists of identifying the closest waypoint to the robot by finding the snapshot in a visual memory that better approximates the robot's location. Homing consists of moving towards that waypoint by visual servoing. Both processes rely on the alignment of the current-view and a snapshot (previously captured when the robot was at the waypoint). Image alignment is performed using an IDF that yields the relative orientation between both images and a measure of how different they are. Algorithms like the VCA are advantageous for applications that involve navigation in unstructured environments because it does not rely on the identification of landmarks. Previous results on the VCA provided encouraging results, but have also exposed some of the limitations of the algorithm (Zeil et al., 2003; Labrosse, 2006; Baddeley et al., 2011; Ardin et al., 2015). A common problem is that most evaluations were performed under highly controlled conditions. Thus, it is arguable that the performance of the VCA in challenging outdoor conditions is poorly understood. Moreover, several aspects that would make the VCA more usable for real-world applications are unresolved.

The first aspect is global localization, important because it would allow a smart-wheelchair to navigate without a localization prior, as long as the robot is nearby a waypoint. Additionally, global localization is necessary to correct off-route deviations (Smith et al., 2008), which would invariably occur due to motor noise and during the execution of obstacle avoidance maneuvers.

Localization failure detection is the second aspect. This aspect enables the robot to know when it is lost. Although a robot "getting lost" is undesirable, when that happens, the robot should know.

The third aspect is knowledge transfer, required when the setup used to acquire visual memories and to navigate, are different. The lack of knowledge transfer makes it necessary for each robot to have its own version of the visual memories.

The final aspect is that the VCA does not provide useful information to steer the robot to correct off-route translational error. There are two current solutions to this issue. The first one is to perform probatory translational movements to infer the correct travel direction (Wystrach et al., 2012; Möller and Vardy, 2006a). That solution emerged as an explanation for how insects navigate. However, performing random movements is difficult and undesirable in smart-wheelchairs because of their kinematic constraints and safety reasons. Another solution is to use a more sophisticated algorithm that takes accounts for scale changes under translation (Möller and Vardy, 2006b; Churchill and Vardy, 2008; Möller et al., 2014).

1.4 Aim and Scope

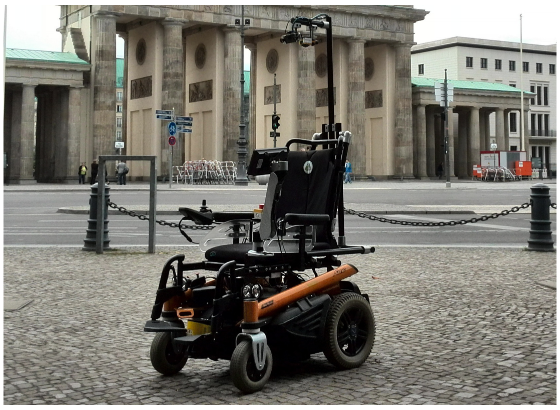


Figure 1.1: The smart-wheelchair ROSStuhl.

The aim of this investigation is to study the VCA for outdoor smart-wheelchair navigation, focusing on the aspects mentioned in the previous section. The evaluation is performed on data captured by a smart-wheelchair and on simulation data that resembles the conditions of a smart-wheelchair navigating outdoors.

Regarding KT, this dissertation limits to introducing it as an essential problem for real-world, large-scale applications and proposes a method to correct differences in intrinsic parameters. A complete KT, would likely also require to correct other aspects, like extrinsic parameters, and even the robot's footprint and kinematics. Some of these factors are briefly discussed later on, but this thesis does not address them.

This dissertation presents the design of the smart-wheelchair ROSStuhl and its route-following system. Unfortunately, the route-following system was not formally evaluated, and the aspects mentioned in the previous section are not covered by it.

The following general limitations in scope are found through this investigation. This investigation does not involve temporal integration of information. Global localization and failure detection are performed at each step, independently of other results. Moreover, omnidirectional vision is the only sensor modality considered. The route-following system of the ROSStuhl breaks this limitation in the sense that the update of the localization hypothesis occurs only after wheel-odometry estimates that the wheelchair has moved since the last update. Nevertheless, this limitation is not present in the experimental results presented.

1.5 Thesis Contributions

This dissertation's main contribution is the study of visual matching with the VCA and texture primitives based on the well-known LBP operator(Ojala et al., 1996, 2002). The second contribution is the TVCC algorithm, which allows merging the results of multiple VCA configurations. The fourth contribution is the introduction of KT as an essential aspect in the area of holistic visual navigation methods. The fourth contribution was the design and construction of the smart-wheelchair ROSStuhl. The fifth contribution is to study the effects of using a precise intrinsic model to unwrap panoramic images, instead of a naive model.

Chapter 2

Background and Related Work

2.1 Autonomous Navigation

Autonomous navigation is the ability by which a robot moves from its current location to a goal. The exact requirements of a navigation system may change between applications, but in general, a robot should navigate efficiently and safely. There are four skills necessary for navigation (Siegwart et al., 2004).

The first skill, *perception*, consists of inferring state information by interpreting sensory information. Perception is challenging for a variety of reasons, like ambiguities in sensor information (aliasing), and differences between what ideal and real sensors deliver. The second skill is *cognition* and consists of using a strategy to reach the goal, given state information and knowledge of the environment. Cognition includes problems like trajectory and route planning. Cognition is challenging because of imperfections in the state estimate of the robot and changes in the environment. In dynamic environments, robots generally plan strategies at different time horizons and readjust as necessary to cope with unexpected situations (Van Den Berg et al., 2006). The third skill, *motion control*, is used to move the robot by modulating the output of its actuators. The last skill is *localization* and allows the robot to know its location with respect to some reference. To that end, a localization system generates localization hypotheses by correlating exteroceptive sensory information with a representation of the environment: a map. Thrun et al. (2005) suggested the following four criteria to differentiate between localization problems:

Local vs. global This criterion depends on the certainty of the robot on its current pose.

- **Position tracking** The robot is very certain of its pose from the beginning and within a small margin of error. Therefore it only considers nearby locations when updating the localization estimate.
- **Global localization** The robot ignores its initial location, and it assumes to be lost. The robot may have to consider a large number of candidate locations to find its current one.

Kidnapped robot The kidnapped robot problem can be seen as a more challenging version of global localization because it requires the robot to recognize when it is lost. The recognition of localization failures is important, as it may help the robot to recover from them.

Static vs. dynamic In static environments, the only thing that changes in the state of the world is the position of the robot. All other objects do not move or change. In dynamic environments, objects other than the robot move and change their configuration.

Active vs. Passive In passive localization, the robot infers its location only by observing through its sensors. In active localization, the robot does not only move to reach the goal but also to gather information to improve its localization estimate.

Single vs. multiple-robot The first is the most common type and involves only one robot. In the second, multiple robots may share information to improve their localization estimates.

Another important aspect of localization is the type of representation used in the map. There are two main types:

Quantitative (metric) represent the world in terms of physical dimensions. The certainty grid is probably the best-known example of this category (Moravec and Elfes, 1985). The map consists of a rectangular grid whose cells represent a region of the environment. Each cell has a numeric value that indicates whether the region is free, occupied, or if such information is unknown.

Qualitative (topological) represent the world in a graph-like structure, where nodes represent places, and edges represented reachability between places Dudek et al. (1978); Mataric (1990); Liu et al. (2012).

Both types of representations have benefits and drawbacks, and there is no superior one. Quantitative representations are fine-grained but do not scale well to large environments. In contrast, qualitative representations are coarse but scale better. However, the drawbacks can be overcome by combining both representation types (Chatila and Laumond, 1985; Kuipers and Byun, 1991; Thrun et al., 1998).

Given the discussion above, it is clear that setting hard boundaries between the four skills is difficult because they may interoperate to different degrees. For example, closed-loop control requires a feedback signal (perception) so that the robot can evaluate the outcome of its actions(Kolter et al., 2010). Likewise, active-localization methods execute motor commands not only to move towards the goal but also with the intent of improving the localization hypothesis. The reader might find it conflicting that localization is posed as a separate skill because just as perception, its goal is to estimate state information. However, it is generally treated separately because it is such a fundamental issue in navigation. It is perception limited to one type of state information: the location of the robot. In contrast, perception covers a wide range of state information types, like the ego-motion of the robot, and the detection and tracking of various types of objects.

2.2 Visual Route-Following

A *spatial memory* stores sensory information that is associated with real-world locations and can be used to revisit them. In the context of this dissertation, the spatial memories contain visual information, which is convenient because it can be perceived passively from vast distances. Locations may be called either home if treated in isolation, or waypoint if they are part of a route.

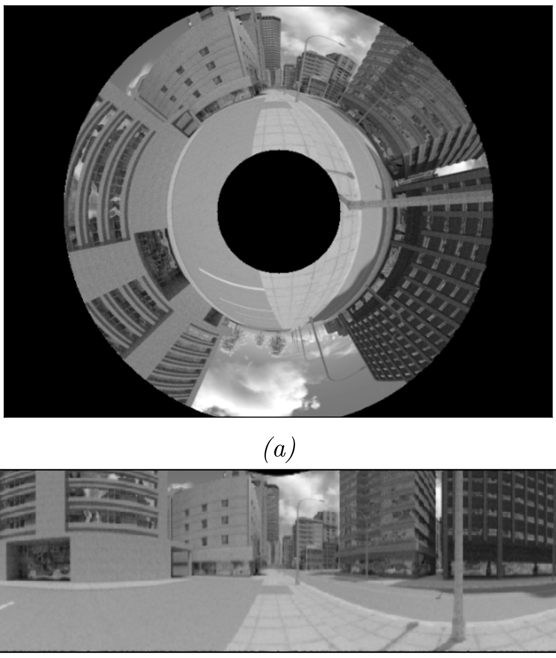
Visual spatial navigation involves several problems that can be enlisted as follows in order of increasing complexity (Stone et al., 2018). Recognizing whether a location is revisited comes first. Second, visual servoing using a beacon, i.e., a salient visual feature placed at home, as reference. Third, visual servoing using visual landmarks observable from home. Route-following comes in fourth place. In that case, the visual memory contains information about multiple interlinked waypoints that constitute a route. The robot travels along the route from beginning to end by sequentially homing through the waypoints. In this investigation, for example, there is one omnidirectional image (referred to as a *snapshot* (Cartwright and Collett, 1983, 1987)) associated with each waypoint. Although not mentioned by Stone et al. (2018), following a route made from segments of multiple visual memories could be placed fifth. According to the definitions provided in Section 2.1, route-following uses a qualitative representation because a route is a graph in which each node is a waypoint, and edges link pairs of consecutive waypoints. Since a reliable initial localization prior is often assumed, the problem is generally addressed as pose tracking. However, it can incorporate global localization and robot kidnapping if necessary, see Section 4.1.

2.3 Omnidirectional Cameras for Visual Navigation

Omnidirectional cameras have a FOV that covers at least one hemisphere [1]. To the best of the author's knowledge, they were introduced in robotics by Yagi and Kawato (1990) and have gained popularity ever since. Omnidirectional cameras are especially useful for navigation because thanks to their large FOV, visual landmarks are more likely to remain visible under varying viewpoint, and because they are arguably less likely to experience significant occlusion, see Payá et al. (2017) for a review. Figure 3.9 shows example images captured with such cameras. In this sense, it is important to differentiate between two representations. In the

[1]Cameras with a FOV narrower than one hemisphere are called directional throughout this document to avoid confusion.

first one, simply called *omnidirectional image*, elevation runs along radial scanlines w.r.t the center of the image, and azimuth along the circumference. In the second representation, called *panoramic image*, elevation runs along the vertical axis and azimuth along the horizontal axis. Omnidirectional images are transformed into panoramic images through a process called *unwrapping*, see Section 4.5.1.


(a)

(b)

Figure 2.1: 2.1a an omnidirectional image. 2.1b a panoramic image obtained by unwrapping the omnidirectional image.

Three types of omnidirectional cameras can be differentiated depending on the principle used to achieve a large FOV. *Dioptric* omnidirectional cameras, often called *fisheye*, rely on refraction in a similar way as directional cameras. However, they have a comparatively smaller focal length and more powerful lenses. See Figure 2.2b. *Catadioptric* omnidirectional cameras are based on the principle of reflection and are constructed by pointing a directional camera towards a curved mirror. There are different mirror shapes with different properties, e.g., conic, elliptic, hyperbolic, parabolic, spherical (Scaramuzza et al., 2006a), and also systems that have mirrors of composite shapes (Franz et al., 2008). See Figure 2.2a. *Polydioptric* omnidirectional cameras produce omnidirectional images by stitching multiple images acquired with directional cameras at different viewpoints. The advantage of this type of omnidirectional cameras is that they can acquire higher resolution images. However, they are considerably more expensive and bulky because they either need of mechanical elements to move a single camera, or a ruggedized construction to hold the directional cameras, triggering mechanisms, and computational resources to stitch the images. This investigation uses only catadioptric and dioptric omnidirectional cameras because they are better suited for smart-wheelchairs due to price and size constraints and because the VCA used in this dissertation operates on low-resolution images, see Section 2.4.

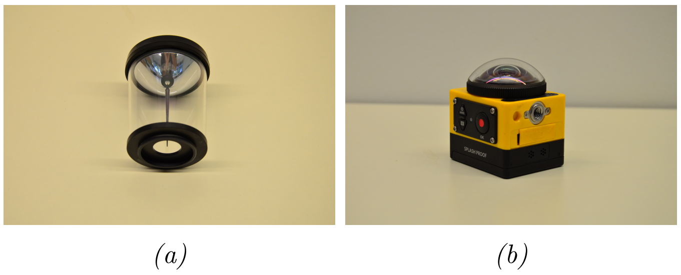


Figure 2.2: 2.2a A hyperbolic mirror used for catadioptric omnidirectional image acquisition. 2.2b A dioptric omnidirectional camera. See Section 3.1.3 for details.

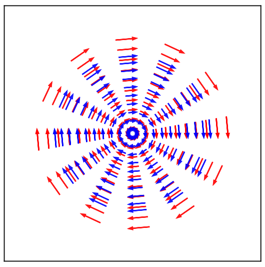
2.3.1 Optical Flow in Omnidirectional Images

Optical flow is the distribution of velocities of apparent displacement of visual stimuli that emerge from the motion of the observer and the objects on the scene (Gibson, 1950). The VCA presented in Section 2.4 relies on regularities of the optical flow in panoramic images under planar motion. Therefore, it is helpful to understand them before going into details of the algorithm. This discussion limits to the planar motion case in static environments. Section 4.5 contains a brief discussion on the effects of non-planar motion.

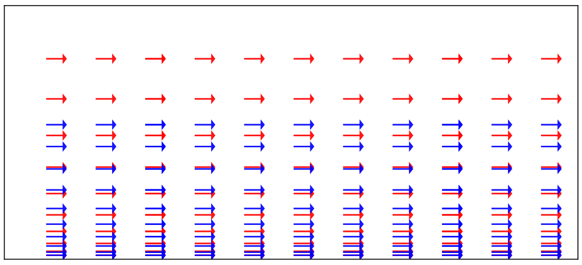
When an omnidirectional camera moves, the pixel coordinates of scene objects change. Under pure rotation, pixel coordinates displace horizontally with uniform magnitude and direction. The magnitude of that displacement correlates with the magnitude of rotation. Regarding its direction, if the omnidirectional camera rotates to the left, the visual information shifts horizontally to the right and conversely. No scale changes occur in this type of motion. See Figures 2.3a and 2.3b.

The case of pure translation is more complex because the magnitudes of displacement depend not only on the movement itself but also on the depth structure of the environment: distant objects displace less in the visual field than nearby ones. This effect is known as *parallax*. Moreover, translation induces changes in scale. Visual information expands from the center of expansion (in the direction of translation), and contracts towards the center of contraction (in the direction opposite to movement). Figures 2.3c and 2.3d show the optical flow resulting by moving the camera forward. Thus, visual information expands from the center of the panoramic image and contracts towards its borders. Notice how visual information moves in opposite directions on both sides, and the elevation and scale changes.

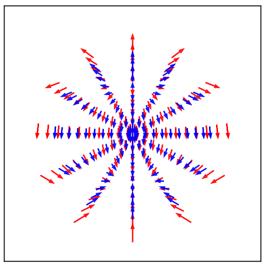
When rotation and translation occur together, the optical flow is a combination of the ones produced by the individual motions. For example, in Figures 2.3e and 2.3f, rotation is dominant in the sense that horizontal displacements occur in the same direction, but scale and elevation changes due to translation are noticeable. Because of that, if translation is of low magnitude, and especially if the scene contains mostly distant objects, the rotational optical flow would be dominant.



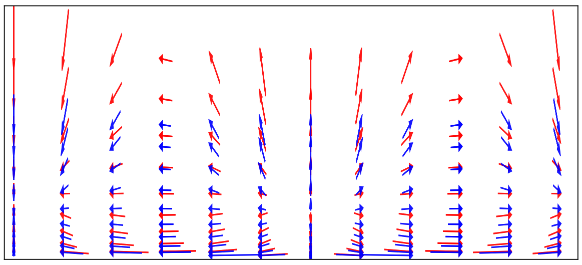
(a)



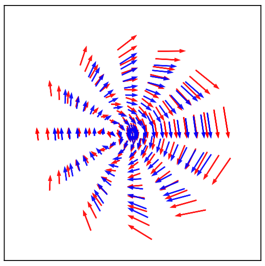
(b)



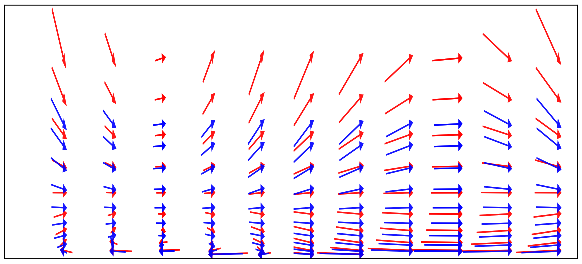
(c)



(d)



(e)



(f)

Figure 2.3: Optical flow under planar motion. Omnidirectional images on the left, and the corresponding panoramic images on the right. The arrows show the optical flow for visual landmarks at two different distances. Closer ones on red, farther ones on blue. 2.3a and 2.3b pure rotation. 2.3c and 2.3d pure translation (forward). 2.3d and 2.3e a combination of the two motions presented above.

2.4 The Visual Compass Algorithm (VCA)

The VCA was introduced by (Zeil et al., 2003) to investigate the image matching mechanisms used by insects, see Section2.5. In its pure form, the algorithm infers the relative orientation between two panoramic images captured at nearby locations and measures how different they are in terms of an IDF. That information can be used to perform homing, especially because IDF values obtained by aligning a current-view and a snapshot correlated with the spatial distance between the current location and home. This principle has been extended to perform visual route-following, and also to localize the against multiple candidate waypoints, and also to detect failures. Such extensions are presented in Section 4.1.

The VCA relies on the regularity of optical flow under planar motion and the dominance of rotation in the presence of translation discussed in Section 2.3.1. The algorithm is defined as a function v that estimates the relative heading between two panoramic images I and I' of size $W \times H$, by finding an azimuthal alignment that minimizes an IDF. The alignment process consists of an exhaustive iterative search over a space of azimuth rotations[2]. At each iteration, the algorithm calculates the distance between I and a version of I' that has been rotated θ degrees in azimuth using the function *rotate*. The best alignment occurs at the azimuth angle that produces the minimum IDF value:

$$v(I, I') = \arg\min_{\theta} idf(I, rotate(I', \theta)). \tag{2.1}$$

The VCA belongs to the *holistic methods* for visual navigation, characterized by using IDFs that operate at the pixel-level, instead of first identifying a sparse set of visual features to work with (See Section 2.6.1). Two common IDFs are the SSD (Zeil et al., 2003) and the SAD(Wystrach et al., 2016) defined in Equations 2.2 and 2.3, respectively.

$$SSD(I, I') = \frac{1}{W \times H} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} (I_{i,j} - I'_{i,j})^2 \tag{2.2}$$

$$SAD(I, I') = \frac{1}{W \times H} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left| I_{i,j} - I'_{i,j} \right| \tag{2.3}$$

The VCA performs efficiently even in modest hardware because it is used on low-resolution panoramic images and because rotations in azimuth appear as shifts in the columns of panoramic images. Moreover, the search space is relatively small because the rotations are performed in terms of full pixel shifts so that interpolation is not necessary. Let s be a function that maps the j-th column of a panoramic image to the corresponding column after a shift of δ pixels wrapping around at the image borders.

$$s(j, \delta) = (j + \delta) \ mod \ W. \tag{2.4}$$

[2]The IDF is sometimes called rIDF, because of this (Wystrach et al., 2016).

Then, let *shift* be a function that shifts all the columns of an image by δ pixels. The i-th row and j-th column are denoted as $I'(i,j)$ for clarity.

$$shift(I',\delta) = \begin{bmatrix} I'(0,s(0,\delta)) & I'(0,s(1,\delta)) & \dots & I'(0,s(W-1,\delta)) \\ I'(1,s(0,\delta)) & I'(1,s(1,\delta)) & \dots & I'(1,s(W-1,\delta)) \\ \vdots & \vdots & \ddots & \\ I'(H-1,s(0,\delta)) & I'(H-1,s(1,\delta)) & \dots & I'(H-1,s(W-1,\delta)) \end{bmatrix}. \tag{2.5}$$

Putting the concepts above together, the VCA can be defined as follows:

$$v(I,I') = \underset{\delta=0\dots W-1}{\arg\min}\; idf(I, shift(I',\delta)). \tag{2.6}$$

Through the rest of the document, δ will be used to denote the shift that produces the best alignment between the two images:

$$\delta = v(I,I'), \tag{2.7}$$

and d will be used to denote the IDF value at which the best alignment occurs:

$$d = idf(I, shift(I',\delta)). \tag{2.8}$$

In the VCA implemented in this investigation, column shifts are pre-computed and stored in a look-up table, and the rotation of images is done implicitly by shifting the coordinates inside of the IDF instead of producing new images. Such optimizations and simplifications are common practice (Stürzl and Mallot, 2002; Möller, 2009). However, the VCA is treated conceptually as above to avoid over-complicating the notation with implementation details.

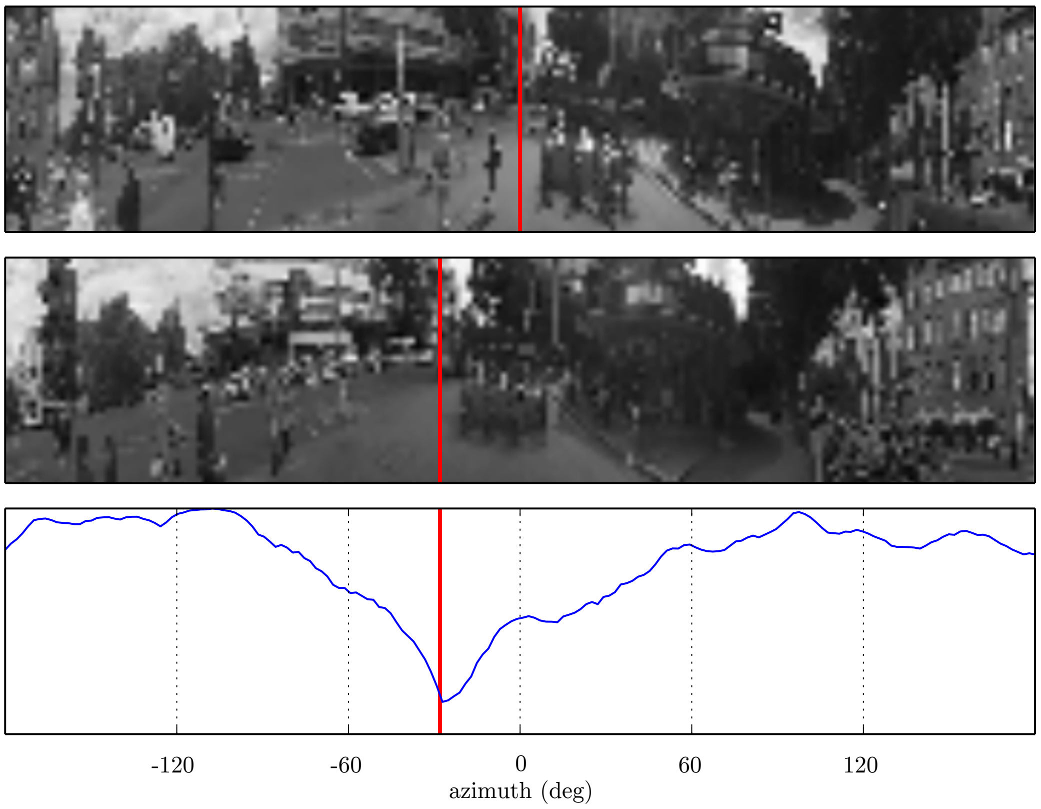


Figure 2.4: Example of alignment using the VCA. From top to bottom: the snapshot, the current-view, and the IDF values at each possible rotation (shift). The vertical red line indicates orientation 0 in the snapshot and the corresponding orientation in the current-view according to the minimum IDF value.

2.5 Insect Navigation

The VCA was inspired by research on insect navigation. This section provides a brief overview of findings on how insects use visual information to navigate. The overall narrative revolves around the idea that insects solve navigation using simple strategies. The content is aimed primarily at roboticists or computer scientists, who may not be acquainted with a broader context. Section 2.5.3 discusses some controversies in insect navigation to avoid giving the misleading impression that insects are simple and well understood by science. The interested reader is referred to Wehner (2003); Freas et al. (2018); Graham and Philippides (2017); Freas et al. (2018), for a comprehensive yet succinct treatment on the topic.

Despite the progress of the last decades, autonomous navigation in unstructured environments is still considered very challenging. Navigation is by no means a recent problem though, and evolution found solutions for it long before the first vertebrates appeared. Unsurprisingly, some roboticists are interested in replicating the astonishing navigation capabilities of animals in robots. As pointed out by Gaussier et al. (2000), this lead to a fruitful cooperation in which hypothesis on animal navigation are tested and refined through robotics or simulation experiments, and in which biomimetic robots and algorithms are consequently developed. For example Graham and Philippides (2014); Song et al. (2013); Jeong et al. (2006); Srinivasan et al. (1999); Weber et al. (1999); Stürzl and Möller (2007); Labrosse (2007); Möller et al. (2014), to name a few.

The most relevant findings in the context of this dissertation come from the study of how insects use vision to navigate. At first glance, insects may seem like humble creatures when compared with humans or other large animals, but they are a notorious example of evolutionary success and excellent navigators. They are also particularly convenient to study because some species (bees and ants especially) have specialized foragers whose main activity is to navigate between the nest and food sources along somewhat stable routes.

According to Graham and Philippides (2017), the study of insects may also result important to understand navigation in larger animals, because navigation strategies are likely phylogenetically spread, and because the study of insects is not plagued with unnecessary assumptions on the contribution that high-level processes, like object identification, have to navigation. Thereby, insects are more likely to be studied using a bottom-up approach, which favors simple explanations that rest on minimal assumptions.

2.5.1 Properties of Insect Eyes

Insects have compound eyes formed of multiple “mini-eyes” called ommatidia. In overly simplified terms, each ommatidium has a lens that projects incoming light against a sensing organ called *rhabdom*, which sends information to the brain via an optical nerve. Compound eyes provide insects with a panoramic FOV of low resolution. The resolution of compound eyes depends on a variety of structural and environmental characteristics. Structural characteristics include the interommatidial angle, optical quality, and rhabdom dimensions, whereas environmental ones include the illumination level and movement (Land, 1997b).

Despite their low resolution, compound eyes have remarkable properties like high acuity to motion, infinite depth of field, and low optical aberration (Warrant and Nilsson, 2006, Chapter 5). Compound eyes are also an example of evolutionary preservation because their design has been present for at least 500 million years (Buschbeck and Friedrich, 2008). Nevertheless, the basic design has gone through a variety of specializations at different levels. For example, in the case of interommatidial angles, some known variabilities are as follows. Variability over a single eye can be found in the *Cataglyphis Fortis* ants, in which the interommatidial angles are nearly constant if measured horizontally, but range from ≈ 3 degrees in a foveal band at the center of the eye, to ≈ 7 degrees at the borders when measured vertically (Zollikofer et al., 1995). Regarding differences at the species level, the males, but not the females of the *Syritta Pipiens* fly have a region of higher resolution at the center of the eyes useful for tracking females (Collett and Land, 1975a). Differences across species are even more dramatic, ranging from approximately 0.24 degrees for dragonflies to the tenths of a degree in the *Apterygota* (Land, 1997a).

After reviewing some peculiarities of insect eyes, it is natural to wonder what insects perceive. The results of Horridge (2009) suggest that some of the basic elements of visual perception, like color, are available to insects (Cheng et al., 1986; Chittka, 2004). It has also been suggested that they derive higher-level information from raw visual information. For example, insects that see in the UV range could easily segment the skyline and use it as a cue for navigation (Von Frisch, 1967; Basten and Mallot, 2010; Wystrach et al., 2012). Although depth perception is arguably advantageous for navigation, insects do not seem to possess the necessary mechanisms to estimate it (Papi, 2012, pp. 45). Instead, evidence suggests that they judge distances in terms of optical flow (Cheng et al., 1987; Vardy and Moller, 2005; Dittmar et al., 2010). Locust, for example, could be tricked to misestimate distances by manipulating the optical flow they perceive (Sobel, 1990). Likewise, bees misestimate distance if the scale of visual content used for optical flow estimation is altered (Esch and Burns, 1995). In a more dramatic example, they have been observed to drawn while flying over a calm lake that does not provide the necessary structure for optical flow estimation (Von Frisch, 1967).

2.5.2 Environmental Representation and Image Matching

Central foraging insects navigate along idiosyncratic routes between the nest and the food source. Decades of research indicate that insects recognize and approach places using visual information, among others. To that end, they store information perceived by their eyes in some internal representation while being at a place and later correlate it with current information. Early work by Wehner (1972) suggested that the internal representation is the raw retinal image and that the correlation consists of simple element-wise comparisons.

Regarding the flexibility of the matching process, evidence suggests that insects do not have intra- and inter-eye transfer, i.e., they can not match visual information when projected at different retinal positions. Therefore, insects move to align the snapshot and the current-view instead of simulating view-point changes as the VCA (Wehner and Flatt, 1977; Collett and Land, 1975b; Wehner et al., 1996; Narendra and Ramirez-Esquivel, 2017; Freas et al., 2018). Thus some authors have suggested that insects may represent a place by storing multiple snapshots captured at different view-points to compensate (Judd and Collett, 1998).

Insects also seem to use different parts of the visual field for image matching, depending on the environmental conditions. For example, the ant *Melophorus Bagoti* and some bees (Wehner, 1972) use the panorama contour (Graham and Cheng, 2009), whereas the ant *Paltothyreus Tarsatus*, who lives in environments covered by large trees, matches information from the canopy patterns above (Hölldobler, 1980). The experiments of Wystrach et al. (2016) on simulation also support that different parts of the visual field are more useful than others. Furthermore, the authors described that increasing resolution does not necessarily improve image matching performance as intuition may dictate.

As mentioned in Section 2.5.1, there is a rich variability in the properties of compound eyes. Behavioral studies also suggest that there are differences in the navigation capabilities of different insects. For example, the navigation performance of the bull ant (*Myrmecia pyriformis*) is easily disruptable by slight modifications of the configuration of visual landmarks observable from the nest (Narendra and Ramirez-Esquivel, 2017). In contrast, the red honey ants (*Melophorus bagoti*) habituate quickly to such changes (Freas et al., 2017; A. and Ken, 2017).

2.5.3 The Ongoing Debate on Insect Navigation

Despite the progress in understanding insects and their navigation capabilities, conclusive findings have been elusive. One particular question that raises significant controversy is whether insects use cognitive maps (Bennett, 1996; Cheung et al., 2014; Collett and Collett, 2006). The reason is in part because there are no agreements on how those maps are supposed to look like, but also because of inherent difficulties in testing related hypotheses (Collett and Collett, 2006). Moreover, some criticisms against the cognitive map hypothesis stem from the bottom-up approach because behaviors attributable to cognitive maps, like merging multiple routes or taking shortcuts (Müller and Wehner, 1988), may also be explained by simpler mechanisms like path integration (Menzel et al., 1998).

Another problem arises from the use of controlled environments for experimentation, because they present simple visual stimuli to the insects. Such settings are convenient to eliminate sources of uncontrolled variance. However, it has been noted that simple changes, like the use of landmarks with more complex textures, may lead to behaviors that would not be observed otherwise (Dittmar et al., 2011).

The final example relates the use of low resolution, an idea traceable to Mallock (1894), who studied the theoretical boundaries for compound eye resolution due to diffraction. Recently, Juusola et al. (2017) challenged that widely accepted idea by proposing that flies have hyperacute vision thanks to microsaccadic sampling. A claiming of that magnitude would probably require further validation, and if validated, it would have a profound impact on how insects are understood. For example, even a more conservative acuity estimate would be sufficient for the fruit flies to identify each other visually (Schneider et al., 2018).

2.6 Visual Navigation and Holistic Methods

Visual navigation is a rich field of study. Thus, there is a large variety of approaches that are differentiated mainly on how they process images to extract navigation relevant information, the precise nature of that information, and how they use it to navigate. This section provides a brief overview, with emphasis on holistic approaches, to which the VCA (Section 2.4) belongs. Figure 2.5 shows a modified version of the classification presented in Möller et al. (2010), with the difference that it includes the 3D visual compass (Differt, 2017). The classification has a tree structure that goes from general at the root to the specific at the leaves.

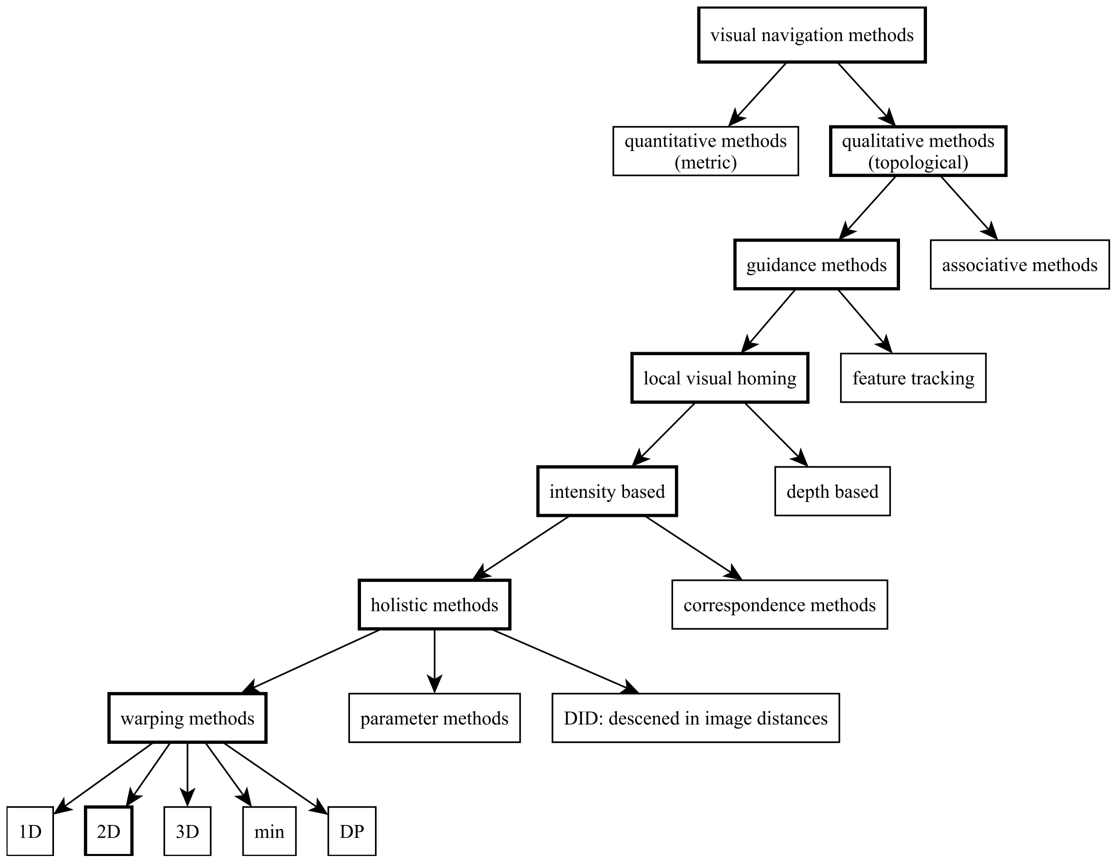


Figure 2.5: Classification of visual navigation approaches proposed by Möller et al. (2014). This version augments the original classification to include the 3D compass.

At each level, two types of approaches are discussed, and the next level expands only on the type that finally leads to the VCA.

The first level of the classification differentiates between qualitative and quantitative, discussed in Section 2.1. Methods at the second level are differentiated by the mechanism used to estimate the motor command for homing. In associative methods, the snapshots readily contain the motor commands to execute when the robot is at the respective location (Nelson, 1991; Gaussier et al., 1997). The main disadvantage of associative methods is that they require a dense sampling of the environment. In contrast, guidance methods offer a more flexible solution because they infer the motor commands from current-view and snapshots by estimating the view-point difference. At the third level, the classification takes into consideration how methods reuse information. Feature tracking methods take advantage of either temporal or spatial information that improves matching efficiency (Argyros et al., 2005; Remazeilles et al., 2006). Local homing methods, in contrast, do not exploit such information. Each alignment between current-view and snapshot is performed without prior information on the placement of visual landmarks in the image.

At the fourth level, local homing methods are split into two categories, depending on whether they use intensity or depth information to operate. Depth methods rely on sensors that can measure depth by some principle. Having depth placed at this level may appear conflicting with level one, which contains qualitative (metric) methods. The reason is that depth at the fourth level is used as a visual feature and is not integrated into a 3D reconstruction of the environment as it would occur at the first level (Stürzl and Mallot, 2002; Franz et al., 2007). Intensity methods are further divided into holistic and correspondence methods. Since holistic methods are the focus of this dissertation, there is a section devoted to them and their alternative. Then, different types of holistic methods are discussed.

2.6.1 Holistic and Correspondence Methods

The general approach in correspondence methods is to detect visual landmarks of interest first and then create a descriptor for each of them. Image matching occurs by matching those descriptors. The most common examples of this category are human-crafted descriptors used to describe points, like SIFT (Lowe, 2004; Churchill and Vardy, 2008; Liu et al., 2012), and to a lesser extent, other types of visual features like lines (Scaramuzza et al., 2009). However, there have also been attempts at learning the descriptors (Carlevaris-Bianco and Eustice, 2014; Lourenco et al., 2012; Masci et al., 2014). Correspondence methods have historically received most of the attention. However, they still present some limitations. Feature matching using descriptors like SIFT is less reliable with omnidirectional and panoramic images because of nonlinearities induced by optical distortion. There have been attempts to overcome that problem (Lourenco et al., 2012; Masci et al., 2014), but a generic solution is still missing. The third problem is that visual features may be sparse, and thus a large proportion of the visual content is not considered when comparing images.

In contrast, holistic methods operate on images as a whole. There are different variations to the concept, but in general, they do not resort to the identification and description of individual landmarks as in correspondence methods. They have been greatly influenced by findings in insect navigation, especially regarding the use of low-resolution retinal images to represent places and image matching based on element-wise comparisons. They have received less interest historically than correspondence methods and are thus arguably less mature. However, they are gaining momentum. Holistic methods have been reported to compare and even outperform the more mature correspondence methods in some situations, including unstructured environments (Differt, 2017), and being more robust to inaccuracies in the camera models (Scaramuzza and Siegwart, 2008). Moreover, holistic methods are faster (Fleer and Möller, 2017; Differt, 2017) and better suited for applications with scarce computational resources. Holistic methods can be further categorized into *parameter methods*, *Descend in Image Distances (DID)* methods, and *warping*.

Parameter methods represent and match images using a parameter vector. In contrast to correspondence methods, in parameter methods, there is only one parameter vector that represents the whole image instead of multiple ones to represent individual visual features. The *Average Landmark Vector (ALV)* is a classic example of this category (Lambrinos et al., 2000). In the ALV, the landmark vectors of individual visual landmarks in the image, i.e., a unit vector pointing in the direction of the landmark, are averaged to create the parameter vector. During homing, the homing vector is found by subtracting the AVL of the current-view and the ALV previously obtained at the reference. In its original formulation, the ALV requires compass-aligned images and reliable segmentation. Later, Hafner and Möller (2001) used Hebbian learning to learn a navigation strategy that mimics the ALV but does not depend on segmentation. Over the years, some variants of ALV have been created, like the Average Correctional Vector (ACV) (Weber et al., 1999; Smith et al., 2008) and the Distance Estimated Landmark Vector (DELV) (Yu and Kim, 2011).

Descent in Image Distance (DID) methods match images by comparing their pixel values using an IDFs that is not viewpoint invariant. DID methods are active (Section 2.1) because the robot has to move to evaluate the IDFs from different viewpoints and resemble thus the lack of intra- and inter-eye transfer of information in insects eyes, see Section 2.5.2. Two recent examples of this category encode routes as a whole without explicit representation of individual waypoints. Baddeley et al. (2011) proposed to use an Adaboost classifier and Haar-like features. The positive class represents that the current direction would make the robot advance along the route, and conversely for the negative class. Philippides et al. (2015) developed a similar approach using a neural network that learns the relevant features instead of choosing from a predefined set.

Multiple snapshot methods use multiple images to represent each location. Möller et al. (2014) considered these methods as the most biologically plausible of the given categorization. Gaussier et al. (2000) presents an interesting example of this category, based on a neural network to encode multiple snapshots.

Warping methods, similarly to DID methods, infer the homing vector using pixel-wise IDFs. However, they do not require probatory movements because they evaluate the IDFs on images that simulate the viewpoint changes. Röfer and Ofer (1997) presented an early image-based homing algorithm that operated on omnidirectional rings of pixels captured at the horizon. Franz et al. (1998a,b) used a similar method, with the difference that the ring contained the column averages of intensity values.

Later developments involved the use of panoramic images instead of rings of pixels. There are two particular publications that the author considers highly influential in this context. Zeil et al. (2003) used the VCA on natural images and performed an extensive analysis of the IDFs. The authors concluded that visual homing is feasible in outdoor environments using simple pixel-wise metrics, i.e., without extracting high-level features. They also provided a detailed experimental analysis on how the frequency of the visual content and the spatial arrangement of the objects in the scene affect the IDFs. A few years later, Stürzl and Zeil (2007) extended that study focusing on the effects that contrast and the 3D structure of the scene have on IDFs. The authors demonstrated how the catchment area of a location grows as the depth of the objects in the scene increases, and how simple operations result effective to provide illumination invariance to IDFs. Scaramuzza and Siegwart (2008) used the VCA on RGB images. The authors constrained the computation to the front and back sectors of the omnidirectional images because they had the most stable scale in their experimental conditions.

The authors justified their use of a holistic method, instead of a correspondence method, because of their robustness to errors in the camera calibration model. The estimation of a compass alignment in frequency domain also constitutes an important advance because it reduces the computational resources needed (Morbidi and Caron, 2017).

Other important investigations that provided useful insights on a variety of aspects of the VCA include the following. Wystrach et al. (2016) studied the effects of varying image resolution on the visual compass algorithm. Remarkably, the results suggest that the intuitive trade-off between resolution and accuracy did not occur in their data. Instead, lower resolutions increased performance. Other interesting results came from Ardin et al. (2015) and Raderschall et al. (2016), who evaluated the effects of off-plane rotation.

The algorithms discussed so far are not invariant to the 3D structure of the environment and have the so-called equidistant assumption described by Franz et al. (1998b). This assumption is present because the algorithms compare pixel values directly. Even in static environments, perfect pixel correspondence could only be achieved under pure rotation. However, when translation is present, perfect pixel correspondences are not possible because of the changes in scale and because of the effects of parallax.

Later efforts addressed the invariance to the 3D structure of the scene. Important developments include the simulation of translational viewpoint changes on the perceived scale of the visual content (Labrosse, 2007; Bellotto et al., 2008; Möller, 2009; Möller et al., 2014).

All the methods described so far in this section depend on the planar-motion assumption. There algorithms that can recover 3-axis rotations from omnidirectional images and are referred to as 3D compass. Because of the computational requirements, these type of methods are implemented in frequency domain using spherical harmonics (Makadia and Daniilidis, 2003, 2006; Friedrich et al., 2008; Differt, 2017). However, the research in this area is more limited than for robots under planar motion assumption. Other types of methods have targeted the use of skyline, for example using Zernike moments (Stone et al., 2018).

Chapter 3

The Smart-Wheelchair ROSStuhl

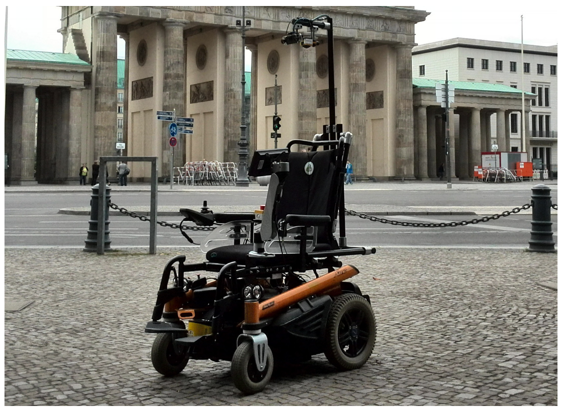


Figure 3.1: The smart-wheelchair ROSStuhl developed during this investigation.

The smart-wheelchair ROSStuhl was created as part of this investigation to acquire experimental data and to implement a route-following system on a real robot. Initially, this investigation intended to be a continuation of the “Alleine” smart-wheelchair (Nauth, 2014; Llarena and Rojas, 2016), which focused on indoor navigation using metrics maps and a lidar. Instead, the ROSStuhl is presented as a different system because the transition to outdoors required an almost complete hardware redesign. Likewise, the software was implemented from scratch to enable the transition to Linux and ROS (see below) because the software of Alleine was monolithic, and parts of it executed only on Windows. Parts of the experimental evaluation presented in Chapter 5 uses data acquired with the ROSStuhl. Unfortunately, that analysis was performed offline. The route-following system described throughout this chapter was only used for short demos and was not formally evaluated.

The route-following system of the ROSStuhl uses the VCA for pose-tracking. Thus, some of the novel concepts presented in Chapter 4 are not part of the implementation. Nonetheless, the system improves over the most puristic use of the VCA in three aspects. The first one is that it localizes the ROSStuhl in the neighborhood of a localization prior, which is arguably more robust than localizing against a single snapshot (Zhang and Kleeman, 2009). The second improvement is that the panorama of textons representation is supported (see Section 4.3.1). Finally, the system is more flexible in the sense that it can travel along routes that were not explicitly demonstrated by merging segments of multiple demonstrations, see Figure 3.2. Figure 3.3 on page 27 contains a block diagram of the complete system. The elements thereby depicted are explained through the remaining of the chapter. The reader is advised to revisit the diagram as necessary.

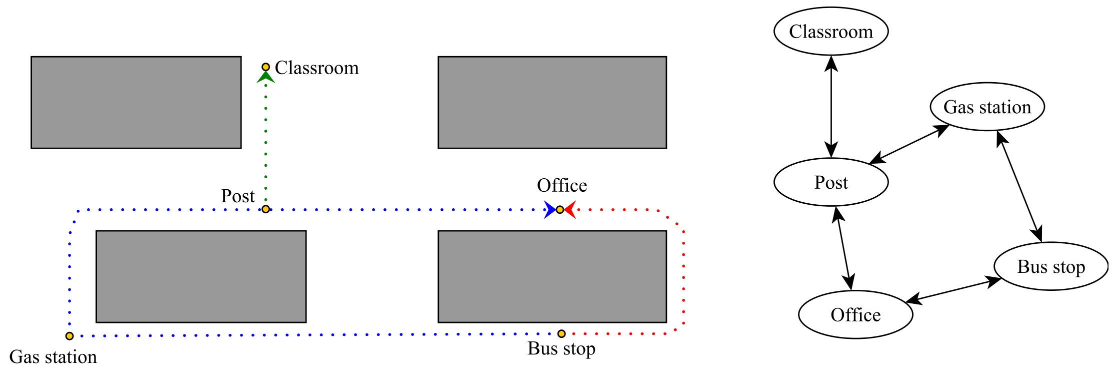


Figure 3.2: The route-following system of the ROSStuhl allows traveling along routes that pass along a set of places of interest by combining segments of visual memories that have place annotations. The colors of the dotted lines represent different visual memories, and the dots represent their snapshots. On the right, a graph that represents the connectivity between places used by the planner described in Section 3.2.2. Other examples of topological maps with omnidirectional images can be found, for example, in (Franz et al., 1998a; Goedemé et al., 2005).

There were two main criteria involved in the design of the ROSStuhl. The first criterion was to favor the use of low-cost hardware because of the price constraints inherent to the application, see Section 1.2. The second criterion was to create a robot using modern practices suitable for rapid prototyping. To that end, the author decided to implement the software of the ROSStuhl using ROS[1], a popular software platform for the development of modular and portable robotics applications. The author also considers that this decision follows a growing interest in developing assistance technology with ROS (Li et al., 2013; Nasri et al., 2016; Grewal et al., 2017). The integration with ROS even served as inspiration for naming the smart-wheelchair. The name ROSStuhl resembles the German word for wheelchair (Rollstuhl) and translates to English as “ROS chair”.

The ROSStuhl is described through the remainder of this chapter in terms of hardware, the software used to interact with the hardware (base system), and the route-following system. Although the reader is not assumed to be a ROS user, basic knowledge of the ROS-specific terminology would greatly simplify the discussion. Thus, the relevant concepts are summarized as follows.

[1]ros.org

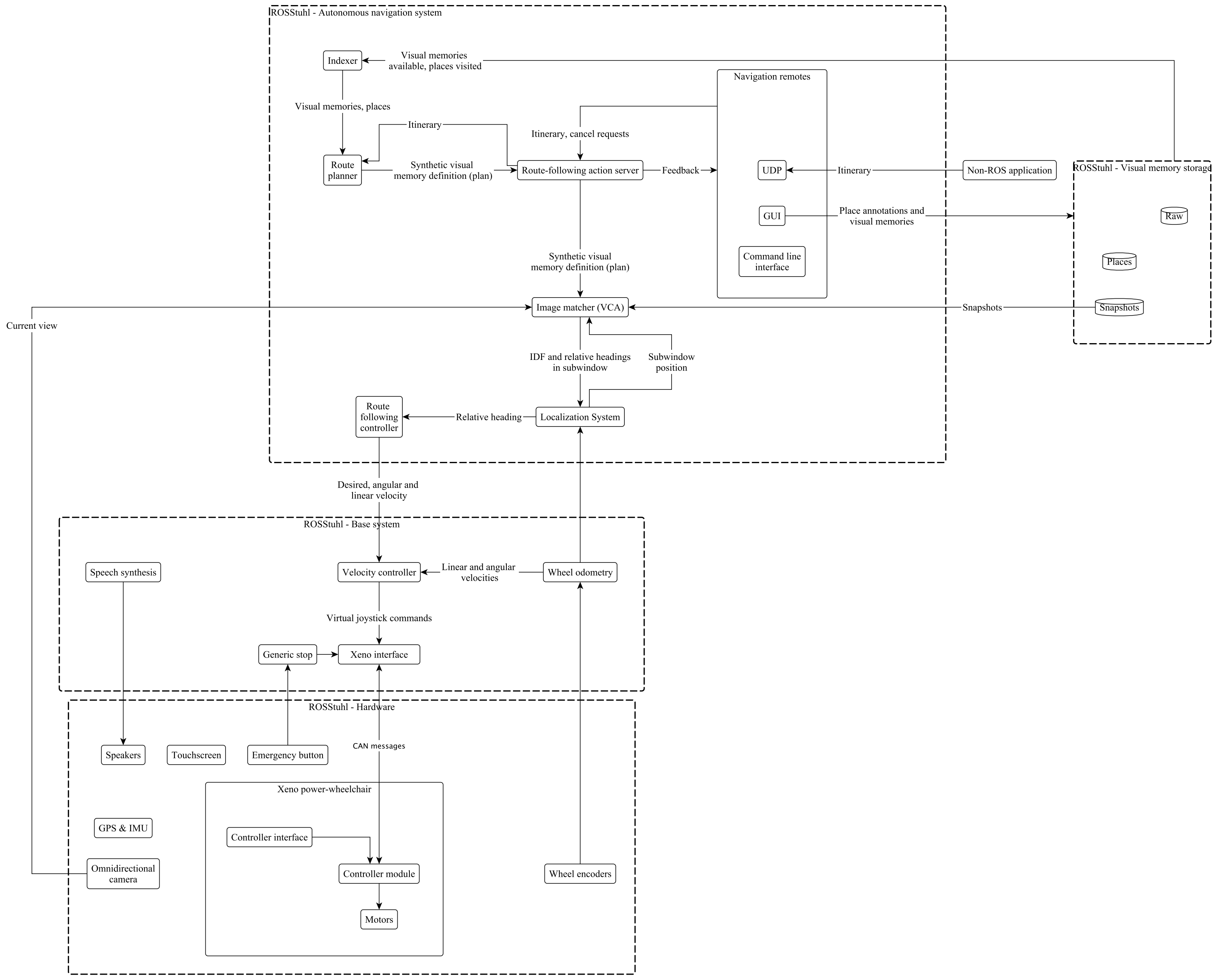


Figure 3.3: The smart-wheelchair ROSStuhl and its route-following system. The base system is in charge of the low-level interaction with hardware. The route-following system can follow routes using synthetic visual memories created from segments of visual memories. This diagram shows the flow information with an arrow.

A process that operates under ROS is called a *node*. Data structures called *messages* are passed asynchronously between nodes through buses called *topics*. A node that sends messages to a topic is called a *publisher*, whereas a node that receives them is called a *subscriber*. Synchronous and asynchronous RMI are called *services* and *actions*, respectively. Nodes that provide services or actions are called *servers*, and the nodes that execute them are called *clients*. In both cases, the client can send data structures as input arguments and retrieve others as return values. Beyond synchronicity, services and actions differ in the sense that the latter provide feedback messages during execution, and can be requested to abort execution, whereas the former do not provide such functionalities. Topics, services, and actions are identified and accessed by unique *names* arranged in a *namespace* hierarchy. For example, in topic /sensors/omnicamera/image_raw, /sensors and /sensors/omnicamera are namespaces, being the former the parent of the latter. Messages, the input and output values of actions and services, and the feedback messages of actions have static data types. For example, the topic /sensors/omnicamera/image_raw may be of type sensor_msgs/Image, a standard message type used to transfer images. Besides the payload, the message types used in this dissertation have a header that contains a timestamp to reference the data temporarily and the name of the coordinate frame used to reference the data spatially.

3.1 Hardware and Base System

3.1.1 The Xeno Power-Wheelchair by Otto Bock ®

Critical obstacle height	5 cm
Maximum passenger weight	136 kg
Maximum slope	17 % ($\approx$ 10 °)
Maximum speed	10 $\mathrm{km\,h^{-1}}$
Energetic travel autonomy	35 km
Batteries	24 volts (2 $\times$ 12 volts in series)

Table 3.1: Power-wheelchair specifications (Bock). The specifications of the ROSStuhl smart-wheelchair deviate because of the weight and power consumption of the components used for autonomous navigation.

The ROSStuhl was built by adapting a Xeno power-wheelchair fabricated by Otto Bock ®, with the specifications listed in Table 3.1(Bock). The power-wheelchair is controlled manually through a control interface that includes a joystick and several buttons, see Figure 3.4. The power-wheelchair also has an embedded computer called controller module that translates input events into motor commands depending on configurable speed level and monitors the state of the system. In terms of kinematics, the power-wheelchair consists of a differential drive with four wheels: the two rear wheels can rotate at different velocities and directions. The two front wheels rotate to comply with the movement elicited by the rear wheels.

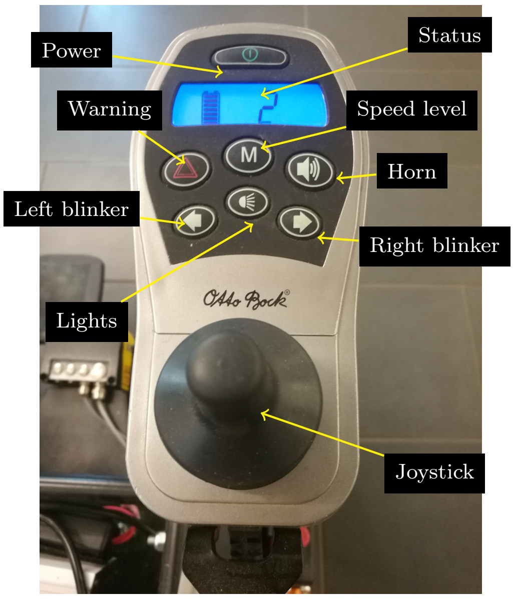


Figure 3.4: The control interface included with the Xeno power-wheelchair. The status display shows the battery level and the current speed level (2).

The hardware components used for route-following can be grouped into four categories. First, the power infrastructure which consists of a DC/AC sinusoidal inverter with a maximum power of 300 watts, several voltage AC/DC converters, and a powered USB hub. The second group consists of connectivity infrastructure, including a five port Gigabit Ethernet switch (Netgear), 2 WiFi adapters (ASUS, and Alfa Network), and the USB hubs mentioned above, which has six powered USB 3 ports for data transfer. The final component is a USB to CAN adapter fabricated by PEAK system. The third group consists of sensors, including one magnetic encoder (0.7 degrees of angular resolution) attached to each rear wheel, a microcontroller that processes the encoder pulses and sends that information to a serial port. These components were inherited from the Alleine smart-wheelchair. Alleine's lidar is also attached to the chassis but is not used in the ROSStuhl. Additionally, the ROSStuhl has a low-cost GPS (Breakout v3 by Adafruit) with a position accuracy of 1.8 meters and an IMU MPU 6050. The data of both sensors is processed by an Arduino Nano microcontroller that sends it to the ROSStuhl through a serial port connection. The data received from the serial port connections mentioned above is processed by ROS nodes that convert it and publishes it as standard ROS messages. The most important sensors, i.e., the omnidirectional cameras, are described in Section 3.1.3. The fourth group has two devices used to interact with the route-following system: a 12.1 inch Faytech T12 SW resistive touchscreen used to interact with the navigation GUI, and the emergency button. Finally, the fifth group has a single member: the navigation computer. The navigation computer is a Gigabit GB-BXi5G3-760 with an Intel Core i5-4200H 2.8GHz/3.40 GHz processor, 16 GB of primary storage, 512 TB SSD of secondary storage, and an NVIDIA GeForce GTX 760 GPU. This computer is more powerful than strictly necessary for navigation. However, the additional processing power resulted useful for outdoors development, see Figure 3.5 and Figure 3.6.

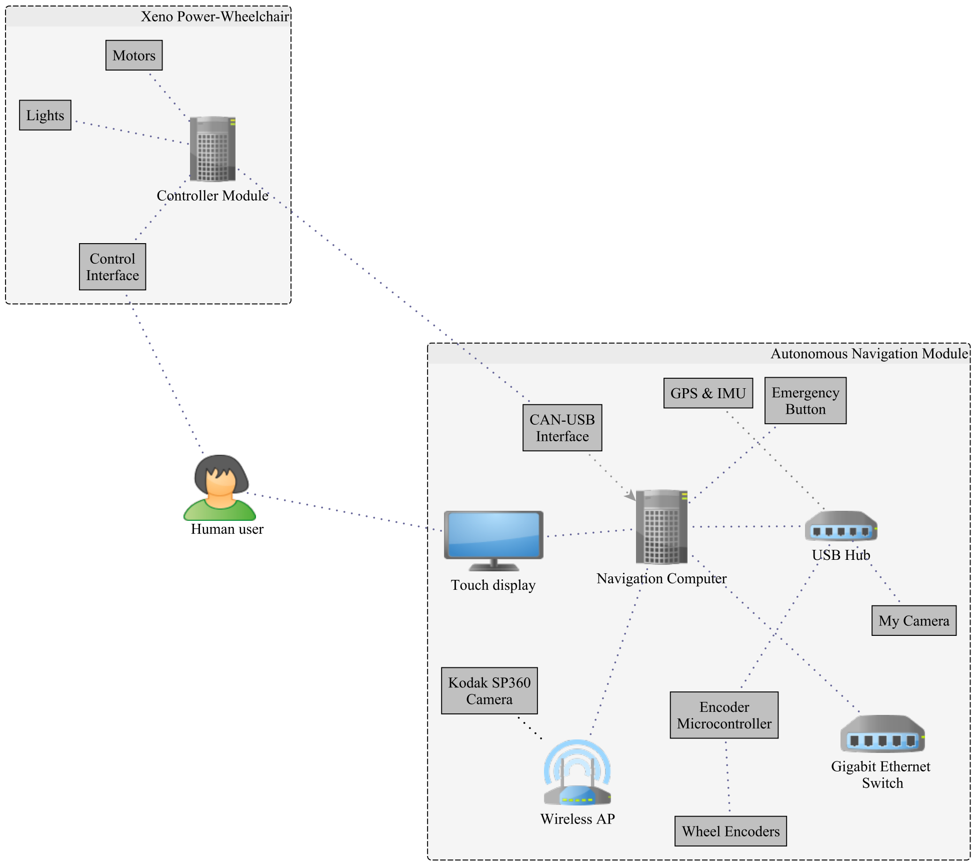


Figure 3.5: The hardware of the autonomous navigation module, and its interactions with the power-wheelchair.

3.1.2 Interaction with the Power-Wheelchair

The navigation computer interacts with the controller module via CAN bus. To that end, the CAN bus protocol was abstracted in a C++ library called xenocan [2] developed as part of this investigation. The xenocan library provides the following functionalities.

First, it polls status information periodically, e.g., voltage of the batteries, input events of the control interface, and the state of the lights, and sends it

[2]Implemented using sockeCAN https://www.kernel.org/doc/Documentation/networking/can.txt.

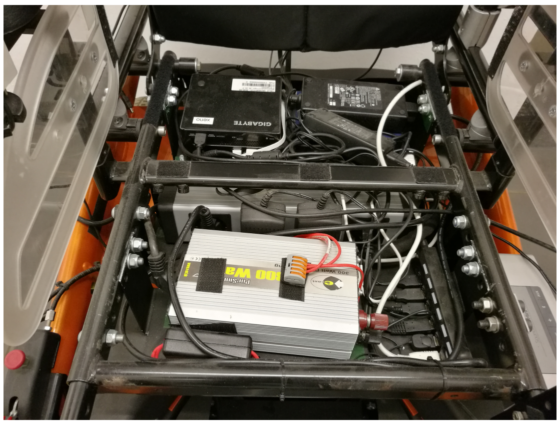


Figure 3.6: Most components used for autonomous navigation are stored underneath the seat.

to a user-defined callback for further processing. The second functionality allows controlling the power-wheelchair by injecting virtual joystick messages in the CAN-bus. The third functionality consists of switching between autonomous driving and manual mode when the horn button of the controller interface is pressed. Finally, the fourth functionality ensures that the speed level remains constant at a predefined value during autonomous driving. Maintaining the speed level constant is crucial for safety because the velocity controller (see below) is parameterized to operate at a particular speed level.

The xenocan library is wrapped by a ROS node, referred to as xeno interface, that exposes its functionality to ROS. Status messages, including whether the system is in autonomous or manual mode, are published periodically to a topic using a custom-defined message type. The same node reads joystick messages from a dedicated topic of type sensor_msgs/Joy, converts them to CAN messages, and injects them in the CAN bus to move the ROSStuhl when it is in autonomous mode.

Directly commanding the ROSStuhl using virtual joystick commands is inconvenient because the same virtual joystick command may result in very different accelerations depending on the properties of the ground, payload of the smart-wheelchair, etc. Therefore, the ROSStuhl implements a velocity controller (PID, using diff_drive_controller package [3]). The velocity controller subscribes to a topic of type geometry_msgs/Twist, containing the desired linear and angular velocities for the vehicle. Then, it estimates the error between desired and actual velocities (using wheel-odometry, which arrives through a topic of type nav_msgs/Odometry.) and produces a virtual joystick message that gradually reduces the difference.

The ROSStuhl also provides a generic stop interface through which any node can stop the system by publishing to a topic. The availability of such an interface simplifies the implementation of some use cases for a route-following robot by decoupling the logic used to stop. For example, a tour guide can be implemented

[3]http://wiki.ros.org/diff_drive_controller

by visiting places of interest in order, stopping at each of them, reading something about the place aloud, and then continuing to the next one. In the case of a delivery robot, the robot would travel to the collection location, stop and wait until the payload is retrieved, and then drive back to the base.

3.1.3 Omnidirectional Cameras and Calibration Message

The ROSStuhl has two omnidirectional cameras, see Figure 3.7 and Figure 3.9. The first one is a Kodak SP360, see Figure 3.10. This camera has a FOV of 214 degrees and fisheye optics. It produces 1024 × 1024 px images at 8 fps via WiFi. Because of the latency, the Kodak SP360 is of limited value for autonomous navigation. Nevertheless, it resulted adequate for experimental data acquisition due to its low cost and the quality of the images it produces. It was also used for route-following at low speeds before an omnidirectional camera with a higher framerate (see below) was available.

The second omnidirectional camera, called LoReNav, was constructed as part of this investigation, see Figure 3.8. LoReNav is a catadioptric omnidirectional camera with a hyperbolic mirror (VS-C450MR-TK by Vstone). The mirror has an approximate FOV of 210 degrees, configured to produce images of 640 × 480 pixels at 30 fps via USB. LoReNav has a housing and a sun cover. The housing holds the directional camera, and hyperbolic mirror in place, and provides a micro-USB port for connectivity. The bottom of the housing has a 1/4 inch UNC screw thread and a pivot pin. Thereby, LoReNav can be attached using standard photographic equipment. The sun cover protects the omnidirectional camera against direct sunlight. The directional camera sees both the hat and the mirror. The omnidirectional sees both the mirror and part of the sun cover. Because of that, the bottom side of the sun cover is covered with mirrors to avoid biasing the auto-exposure control.

The omnidirectional cameras are calibrated using a naive model used to unwrap, scale and, crop the VFOV of the panoramic images, see Section 4.5.1. The calibration parameters are passed to the relevant components using a custom message type called OmniCameraInfo, whose fields are provided in Table 3.2. A dedicated node publishes the parameters for each omnidirectional camera, at the same frequency and with identical timestamps as the images they model. By default, the topics of OmniCameraInfo messages are located in the same namespace as the image topics.

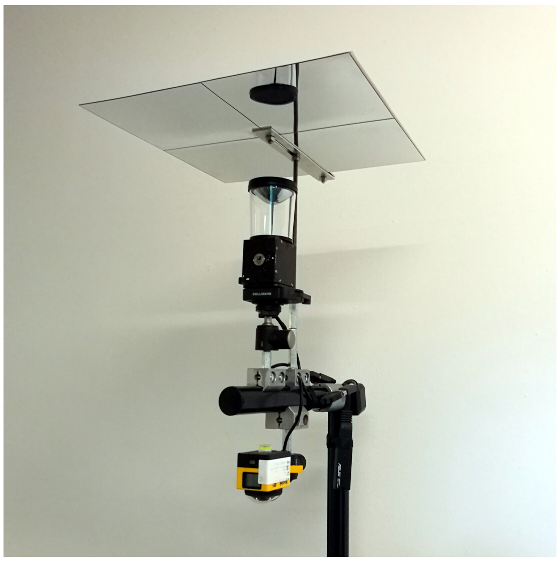
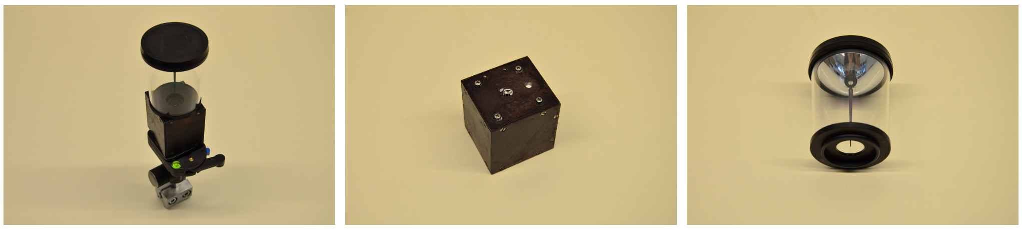


Figure 3.7: Omnidirectional cameras mounted on the ROSStuhl. The one above is the LoReNav camera, a catadioptric camera with a hyperbolic mirror. The sun cover reduces direct exposure to the sunlight. Its lower side is covered with mirrors to prevent biasing the exposure controller. The omnidirectional camera below is a Kodak SP360.



(a) (b) (c)

Figure 3.8: Building elements of the LoReNav camera. 3.8a closeup of the camera without the hat. 3.8b bottom of the camera. This side has a standard 1/4 UNC thread and a pivot pin used to fixate it to a standard camera plate. 3.8c VS-C450MR hyperbolic mirror.

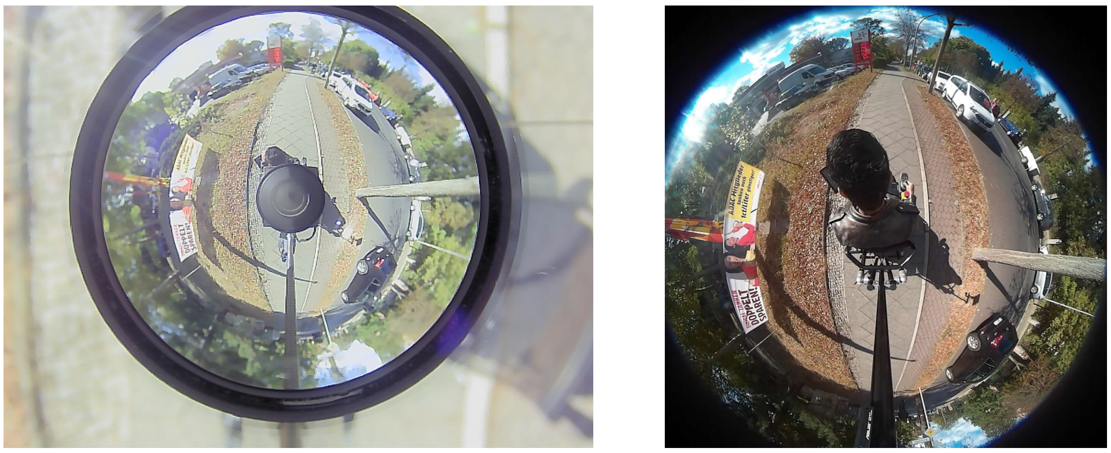


Figure 3.9: Omnidirectional images acquired at nearly the same time. On the left, by the LoReNav, and on the right by the Kodak SP360.

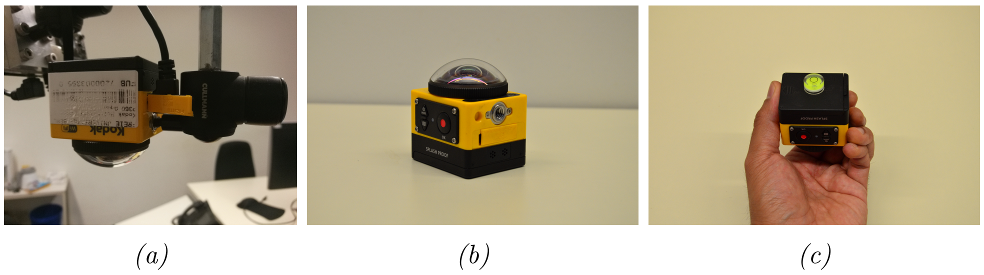


Figure 3.10: Kodak SP360 fish eye camera. 3.10a the camera mounted on the ROSStuhl. 3.10b close up of the camera. 3.10c bottom of the camera. A bull's eye level was glued to the camera to facilitate its alignment.

Names	Type	Description
header	Header	Header containing timestamp and name of the coordinate frame
model	string	Name of the camera, e.g., "Kodak SP360"
version	string	Unique identifier of the camera setup (based on date)
optics	string	Optics. "fisheye", "hyperbolic", etc.
fov	float32	FOV in radians
pole	float32[2]	Pixel coordinates of the image center
radius	float32	Radius in pixels respect the pole, at which there is only valid image data
border_radius	float32	Radius in pixels respect the pole, to the border of the omnidirectional image
orientation	string	"upward" or "downward"

Table 3.2: Fields of the OmniCamInfo message used to transmit calibration parameters of the naive model, see Section 4.5.1.

3.2 Route-Following System

Chapter 2 covered the theoretical foundations of visual route-following using the VCA. This chapter describes how those ideas were implemented into a modular route-following system. The route-following system relies on visual memories annotated with places of interest that were visited in each of them. Visual memories are acquired with the help of a human demonstrator that drives the smart-wheelchair and tags place of interest using a dedicated GUI (see Section 3.2.2) whilst the smart-wheelchair records the data received by its sensors (see Section 3.2.1).

For route-following, the user defines a travel itinerary that contains the names of the places the route has to pass through and the order in which they have to be visited. Then, the itinerary is sent to the route-following planner that outputs a synthetic visual memory definition. Route-following consists of traveling through the waypoints defined by the snapshots in the synthetic visual memory using the feedback of the VCA to control the smart-wheelchair, see Section 3.2.2 and Section 3.2.3.

3.2.1 Visual Memory Storage and Indexer

The route-following system stores sensor information and place names using ROSbags, a standard format used to store serialized ROS messages organized in topics. The storage is arranged in three directories called: raw, snapshots and, places. The data of each demonstration session is stored in a separate bag with corresponding names in each of the storages.

The rosbags of the raw directory contain sensor information at full sampling rate. The essential topics in them are the omnidirectional images, the corresponding OmniCamInfo messages, and the wheel odometry. Additionally, they also contain GPS and IMU data, albeit it is not used for navigation. The rosbags in this storage are not used for route-following because operating on them takes prohibitively long. In the experience of the author, merely opening a full rate bag may take minutes. Moreover, they are regularly moved to a different storage medium to free space in the hard drive of the navigation computer.

The rosbags in the snapshot directory serve as visual memories. They contain the same information as the rosbags in the raw storage, downsampled so that there is only one message in every topic for every 0.5 meters of travel distance. The bags in the places directory contain a single topic of timestamped strings. Each of those messages denotes the name of the place of interest the smart-wheelchair was at, at a certain time.

The route-following system also has a component called indexer that regularly scans the snapshots and places storage. During that process, the indexer collects information on what visual memories are present, what places where visited, and the snapshots and places correspondences. The indexer makes that information available to other components via a ROS service.

3.2.2 Route-Following Server, Planner, and Remotes

The functionalities necessary for route-following are implemented in different ROS nodes. The route-following action server is the most important of them, in the sense that it orchestrates the functioning of the others. The route-following action server receives the itinerary from the user, queries the planner, configures the remaining components, and allows to abort route-following operations.

The route planner is implemented as a service. It receives an itinerary as input and outputs a plan in the form of a synthetic visual memory definition. The itinerary consists of a list of places to visit, e.g., (A, B, C), and the plan consists of visual route segments that travel from A to B and from B to C. The place at the beginning of each segment is considered as an intermediate origin and the place at the end as an intermediate goal. For example, B is in the intermediate goal when traveling from A to B, and an intermediate origin, when traveling from B to C.

The planner uses the information provided by the indexer to create an undirected graph in which nodes are places, and edges are visual memory segments that connect them, see Figure 3.2. Each edge is assigned a weight that corresponds to the number of snapshots in the segment. Since the distance between snapshots is fixed, the weights of the graph correlate with travel distance. Finally, the planner uses Dijkstra's algorithm to search for the path of minimum cost in the graph that connects all the places in the itinerary. Thus, plans are generated to minimize travel distance because the snapshot bags are spatially filtered by it, see Section 3.2.1. A synthetic visual memory definition is a list of visual memory segment, each containing the following fields:

Visual memory name The name of the visual memory in the snapshots storage that contains the segment.

Start time Timestamp of the start of the segment, i.e., when the ROSStuhl was at the intermediate origin place, according to the rosbag in the places storage.

End time Timestamp of the end of the segment, i.e., when the ROSStuhl was at the intermediate place, according to the rosbag in the places storage.

Reverse A binary field indicating whether the desired travel direction is reversed, respect the direction traveled during demonstration. This field would be true if the visual memory was captured while traveling from place A to place B, whereas the itinerary request to travel from B to A.

The interaction with the route-following action server is simplified for the programmer thanks to an abstract Python class called Remote. There are currently three specializations of the Remote. The first in the form of an interactive command-line program. The second acts as a proxy between the ROS ecosystem and non-ROS clients using UDP.

The third specialization is found in the navigation GUI developed using Kivy [4], following a minimalistic design with high-contrast colors and large visual elements to ease outdoor operation. The GUI has three sections and displays only one at any given time, depending on the situation. The first section is used to request route-following operations and provides a menu to define itineraries and a button to initiate navigation. The second section is automatically activated during route-following, and displays feedback of the operation, and allows canceling it by pressing a button. The third section is used for demonstration. To that end, the demonstrator selects the names of the places of interest along the route from a menu, presses the start button, and starts driving. In the background, the GUI starts recording sensor data in a new bag in the raw storage. While driving, the demonstrator is presented with a list of places that have to be visited. Every time a place of interest is reached, the demonstrator informs the GUI by pressing a button. Arrival to places is recorded in a rosbag in the places storage. Once the last place is reached, the GUI software stops recording and filters the raw bag, creating thus a new bag in the snapshots storage.

3.2.3 Image Aligner, Localization System, and Route-Following Controller

The image aligner loads a synthetic visual memory definition on request of the route-following action server. Hereby, the image aligner concatenates the individual segments into a single visual memory. If the travel direction of a segment is reversed, the planner concatenates the snapshots of that segment in inverse order. After loading, the image aligner executes the VCA on the current-view delivered by an omnidirectional camera and each of the snapshots in a subwindow of the synthetic visual memory. The resulting relative headings and IDF values, along with the timestamps at which the snapshots were captured and their indexes in the synthetic visual memory, are published to a topic.

The localization system subscribes to the topic just mentioned and finds the snapshot that has the smallest IDF value. The index associated with that snapshot serves as localization hypothesis. Then, the localization system sets the start and end of the subwindow so that they are centered around the localization hypothesis using a ROS service of the image aligner. The localization system only allows changes in the localization hypothesis if the smart-wheelchair has traveled at least 0.2 m (according to odometry) since the last update.

The localization system also publishes a list of the places that have been reached so far to a topic. It assumes that a place has been reached if the index of the localization hypothesis is larger or equal than the corresponding index in the synthetic visual memory. The route-following action server listens to that topic, and whenever a new place is added to the list, assumes that the robot is currently located at that place and adds that information to its feedback messages.

Guidance is performed by the route-following controller. This component receives the relative heading of the localization hypothesis and calculates linear

[4]A toolbox for rapid development of touch-based applications using Python (kivy.org).

and angular velocities used to command the smart-wheelchair. If the localization hypothesis corresponds to a segment with inverted travel direction, the relative heading is inverted accordingly by adding π radians to it. The estimation of velocities is currently implemented using a simple heuristic. If the magnitude of the relative heading is more than 30 degrees, the smart-wheelchair rotates on its axis. If not, the linear velocity is inversely proportional to the magnitude of the relative heading, and the angular velocity is proportional to it. The resulting desired velocity values are passed to the velocity controller by publishing to a topic of type geometry_msgs/Twist. The boundaries for velocity and acceleration are enforced already by the velocity controller. Thus, they do not have to be taken into consideration by the route-following controller.

3.2.4 Image Representation Pipeline

The omnidirectional images go through multiple processing steps before being passed to the VCA. Those steps are arranged in a pipeline that is defined using the parameter server of ROS. A pipeline consists of a series of processes that take a model and an image as input and output an updated version of them. The first element receives the image and OmniCamInfo from the omnidirectional camera, transforms the image into another image (may also modify the model), and pass its output to the next element. This process continues until the last element is reached. The resulting images are then ready for the VCA. Figure 3.11 has an example pipeline with three elements. The first element sets the minimum and maximum VFOV of the image. The second element unwraps the omnidirectional image into a panoramic image. The third downscales the panoramic image to a lower resolution, and the last one transforms the panoramic image into a panorama of textons. A similar concept is implemented in the evaluation toolbox presented in Section 5.4, which contains the authoritative implementation. The image preprocessing pipeline is implemented as a C++ library that is linked with the image aligner for efficiency reasons.

```
$ rosparam get /visual_navigation/pipeline
- ReducedVerticalFOV: {max_fov: 50.0, min_fov: -40.0}
- Unwrap: None
- LowResolution: {azimuth_resolution: 2.5, elevation_resolution: 2.5}
- LBP: {points: 4, radius: 20.0, type: DEFAULT}
```

Figure 3.11: Preprocessing pipeline defined using ROS's parameter server. The definition is interpreted during the instantiation of the image processing pipeline.

Chapter 4

Aspects of the VCA for Real-World Applications

Research on the VCA has provided many fruitful insights on the virtues and limitations of the algorithm, especially for route-following as pose-tracking. However, the study of the VCA has been mostly limited to conditions that do not resemble those experienced by a smart-wheelchair navigating outdoors. Thus, the performance of the VCA under such conditions is arguably poorly understood.

Moreover, the author proposes that the capabilities of the VCA require of some extensions to make it more promising for real-world applications. This dissertation considers the following extensions: Global localization, failure detection, and knowledge transfer.

Global localization consists of finding the corresponding snapshot in a visual memory given a current-view. Consequently, failure detection determines whether that correspondence is correct. Knowledge transfer would allow visual memories to be shared by multiple robots. This chapter also presents an algorithm to cope with a well-known limitation of the VCA: it does not provide useful relative heading estimation when the robot is displaced off-route unless a more complex warping function is used (Möller and Vardy, 2006b; Churchill and Vardy, 2008; Möller et al., 2014), or probatory movements are performed (Wystrach et al., 2012; Möller and Vardy, 2006a).

4.1 Global Localization and Failure Detection

Given that the VCA is inspired by insect navigation, it results natural to wonder whether insects can perform global localization, and detect failures. Evidence for global localization insects can be found in Wehner et al. (1996). In their study, the authors observed that ants deprived of path-integration information could navigate along a route after being displaced. The authors suggested that that was possible because insects memorized visual information along the entire route and used it to navigate. Similar findings were reported by Kohler and Wehner (2005); Narendra (2007a,b); Mangan and Webb (2012).

Regarding global localization with the VCA, Smith et al. (2008) measured its performance in the presence of off-route displacements due to motor noise. The authors reported poor performance because the VCA only found the exact best match $\approx 50\%$ of the time. Unfortunately, it is difficult to generalize their results to outdoor navigation for two reasons. The first one is that they performed the experiments in an indoor visually-dull environment that does not reflect the rich texture present in outdoor scenes. The second reason is that the environment only had nearby objects, which is particularly problematic for the VCA in its pure formulation due to the effects of parallax. Moreover, it is questionable that the perfect match is necessary for route following. In contrast, the results of Baddeley et al. (2011) suggest that reliable global localization is possible with the VCA. Ardin et al. (2015) later showed that global localization can be performed with some robustness to non-planar viewpoint changes (pitch). Unfortunately, both authors also used visually dull scenarios, but with a depth structure that better resembles outdoor environments. The use of visually dull scenes is rooted in the sensitivity of insect eyes to different wavelengths, which would allow them to easily segment the sky and the foreground (Freas et al., 2017; Wystrach et al., 2012; Stone et al., 2018; Basten and Mallot, 2010). However, performing such a segmentation without specialized hardware would be difficult in a robot.

Regarding the detection of localization failures, two reported behaviors, i.e., search (Wehner et al., 1996) and backtracking (Wystrach et al., 2013), indicate that it may be performed by insects. The existence of those behaviors makes it reasonable to think of a mechanism that triggers their onset and termination: when insects believe to be lost, they initiate a contingency behavior, perform it until they believe to be localized again, and then continue navigating as usual.

Another related idea comes from Wystrach et al. (2012). The authors proposed the use of different localization strategies that are used depending on the situation: Raw visual information when traveling along the route and the skyline contours when the robot is away from the route. In a similar fashion to the triggering of contingency behaviors, it is reasonable to think that switching strategies requires that insects either realize when to use each strategy or a way of merging multiple ones into a single estimation.

4.2 Global Localization and Failure Detection Using a Tree of VCA Configurations

Global localization and failure detection are desirable skills for a robot, and more so, in applications that rely on route-based representations of the environment. The reason is that the reduced spatial coverage of the route-based representation allows only for a relatively small margin for error. If the robot travels in the wrong direction, it might easily stray away from the safety of the route, and if far away enough, it may not be able to localize again. The implementation of contingency behaviors like search or backtracking observed in insects carries many complications in a robot. Thus, it is desirable that the robot remains on the route and detects failures as soon as possible to avoid dangerous situations and the execution of contingency behaviors.

Given a current view c, global localization using a visual memory $M = \{m_i\}_{i=1}^{n}$ with n snapshots, consists of finding the snapshot that corresponds to c (Smith et al., 2008; Baddeley et al., 2011; Ardin et al., 2015). Through the rest of the document, d_i denotes the IDF value of the best alignment between current-view and the i-th snapshot in a visual memory. For example, the index of the snapshot corresponding to c is obtained by

$$l = \underset{i=1..n}{\arg\min}\, d_i, \tag{4.1}$$

and d_l is the corresponding IDF value [1]. Conveniently, the same concept can be used to localize the robot in the vicinity of a localization prior by limiting the search range to the snapshots around it. Localization failure detection can be performed by classifying the localization hypothesis as either a success (positive class) or a failure (negative class). For example, by applying a threshold τ on the IDF value

$$P = \begin{cases} 1 & \text{if } d_l \leq \tau \\ 0 & \text{otherwise} \end{cases}. \tag{4.2}$$

This simple approach is attractive for several reasons. The first is that it reuses the IDF value used to localize the robot, and thus the computational cost of classification is negligible. The second is that it is flexible because it can be used if the robot is localizing against a single snapshot, around the vicinity of a localization prior, or against an entire visual memory.

The performance of Equations 4.1 and 4.2 is likely affected by factors including look-alike places and appearance changes that occur between the acquisition of a visual memory and route-following. Such factors are not under the control of the robot. However, a robot using the VCA has control over two basic aspects that affect how similar it considers two places to be: the IDF, and the representation of the panoramic images. Through the rest of this document, a combination of IDF and two parameterized representation functions (one for the snapshot and one for the current-view) is called a *configuration*.

Research on the VCA so far considers representation as mutually exclusive. Either one or the other is used, and if at all, they are compared against each other. This view was challenged by Wystrach et al. (2012), who suggested that ants may actually use different representations depending on the situation. The authors also proposed a switching mechanism depending on a threshold on the IDF value as in Equation 4.2. A similar use of a threshold was proposed by (Nelson, 1991) as part of an associative method. This investigation develops on those results and introduces the TVCC, which allows switching between multiple VCA configurations.

[1]Biologists favor route representations that do not depend on explicit storage of waypoints (Baddeley et al., 2011, 2012). However, the contents of this section are developed using an explicit encoding because it is more convenient in robotics, as it facilitates operations like route planning, see Section 3.2.2.

The formalization begins by defining the VCA so that all elements involved are given explicitly. Besides the IDF, the VCA also depends on two representation functions r and r', and their parameters ξ and ξ', that transform the current-view and snapshot into the images that are passed to the IDF. The distinction becomes essential to the contents of Section 4.5. For now, assume that $r^{(i)} = r'^{(i)}$ and $\xi^{(i)} = \xi^{(i)}$.

The notation uses a superscript in parenthesis, not to be confused with a power, to refer to the j-th among n VCA configurations. As before, a subscript denotes the i-th snaphsot of a visual memory. Thus the VCA can be defined as:

$$v^{(j)}(c, m_i) = \arg\min_{\delta} idf^{(j)}(r^{(j)}(c, \xi^{(j)}), shift(r'^{(j)}(m_i, \xi'^{(j)}), \delta)), \tag{4.3}$$

$$\delta_i^{(j)} = v^{(j)}(c, m_i), \tag{4.4}$$

and

$$d_i^{(j)} = idf^{(j)}(r^{(j)}(c, \xi^{(j)}), shift(r'^{(j)}(m_i, \xi'^{(j)}), \delta^{(j)})), \tag{4.5}$$

The TVCC consists of n VCA configurations, each with a corresponding threshold

$$T = \left\{(v^{(j)}, \tau^{(j)})\right\}_{j=1}^{n}. \tag{4.6}$$

Classification consists of evaluating the threshold classifiers at each stage until one passes the test. In that case, the localization is considered successful. If none passes the test, the localization is considered a failure.

$$P_T = \bigvee_{j=1}^{n} d_l^{(j)} \leq \tau^{(j)}. \tag{4.7}$$

$$\begin{tikzcd}[column sep=small]
 & d_l^{(1)} \leq \tau^{(1)} \arrow[dl, dash] \arrow[dr, dash] & & & \\
1 & & d_l^{(2)} \leq \tau^{(2)} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & 1 & & \ldots \arrow[d, dash] & \\
 & & & d_l^{(n)} \leq \tau^{(n)} \arrow[dl, dash] \arrow[dr, dash] & \\
 & & 1 & & 0
\end{tikzcd}$$

Figure 4.1: TVCC. The classification is performed in stages (from top to bottom), each consisting of a threshold classifier. The evaluation of threshold test branches to the left if true, and to the right if false in the figure. A localization hypothesis is considered as successful if any stage of the TVCC passes the threshold test, and as failure otherwise.

In the TVCC, each configuration leads to a different localization hypothesis l. Hence, the computation can easily become prohibitively expensive if operating on a large visual memory and if the deeper stages have to be evaluated regularly. This problem can be prevented if two assumptions are met. The first assumption is that the robot usually has a localization prior. Therefore, local localization is generally sufficient. In the cases when local localization fails, then the search range can be extended globally to find a new localization hypothesis that serves as prior for the next steps. The second assumption is that the first stages correctly classify successful localizations as such most of the time, inhibiting thereby the execution of deeper stages [2].

This investigation does not provide a method for learning the TVCC. The feasibility of the approach is evaluated on a model learned by brute force search, see Section5.3. Future directions are discussed in Section 6.5.

4.3 Reducing Aliasing using Panoramic Images of Labels

A good performing VCA configuration should allow reliable global localization and failure detection. Intuitively, the chances of performing well at these tasks can be increased by using configurations that are discriminative: corresponding locations result in small IDF values and conversely. This dissertation proposes the use of panoramic images of labels to that end. Thereby, pixel values are labels that categorize the visual information of their neighborhood. There is an important consideration in this respect. Recalling the contents of Section 2.3.1, translation leads to changes in scale and pixel positions at which scene points are projected. Thus, panoramic images of labels must provide some degree of scale and position invariance so that images can be matched under translation. More precisely, scale invariance means that corresponding scene points are assigned the same labels, even if their apparent size on the image changes. Position invariance means that corresponding pixels (after alignment) have the same label, even if they only approximately image corresponding scene points. Both properties are naturally present when working on gray level images because values of neighboring pixels are often similar.

The proposed approach integrates seamlessly with the warping performed by the VCA, up to the use of a new IDF. This modification is necessary because panoramic images of labels contain categorical values. Thus, common IDFs like SSD and SAD are not well suited to align them. This investigation proposes an IDF called PLD, that takes values from 0 to 100

$$\mathrm{PLD}(I, I') = \frac{100}{W \times H} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \begin{cases} 1 & \text{if } I_{i,j} \neq I'_{i,j} \\ 0 & \text{otherwise} \end{cases} . \tag{4.8}$$

[2]This assumption is analog yet opposite to that of the cascade classifier: early stages correctly classify most false examples correctly.

4.3.1 Panoramas of Textons using the LBP operator

The labeling of pixels is done in terms of texture, an idea partially inspired by the results of Dittmar et al. (2010, 2011). In their publications, the authors studied how texture affects the flying behavior of bees and concluded that image matching in terms of texture explains flying behavior only if the texture is salient.

Nevertheless, the authors did not provide a clear definition of what they meant by *texture* and used gray level images to explain flying behavior in terms of texture. This is perhaps not surprising because the word texture is often used colloquially and has been elusive of a unique formal definition (Coggins and Jain, 1985). This investigation treats texture in a more formal way that emerged in the field of texture analysis. Texture is considered as a property perceived in terms of basic elements called *textons* (Julesz, 1981). This implies that, although texture can be calculated from gray level images, it is not explicitly contained in them.

Thus, the textons define categories used for pixel labeling. Textons are extracted from gray level images using an established technique in texture analysis called LBP (Ojala et al., 1996, 2002). Figure 4.5 shows the difference between using PLD and panoramas of textons and SAD on raw images. The PLD leads to a more salient IDF value at the best alignment and exhibits thus less aliasing. The LBP operator takes a gray level image as input and outputs an LBP image[3]. The pixel values of LBPs image, called LBP codes, can be seen as labels: each value corresponds to the texture primitive that better describes the neighborhood of that pixel. The derivation of four variants of the operator is presented to the bare minimum. Please refer to the original sources for details.

In LBP, texture T, defined as a local property of a gray level image, is the joint distribution of gray levels of a reference pixel g_c, and P ($P > 1$) surrounding pixels sampled along a circle of radius R centered at g_c (see Figure 4.2)

$$T = t(g_c, g_0, g_1, ..., g_{P-1}). \tag{4.9}$$

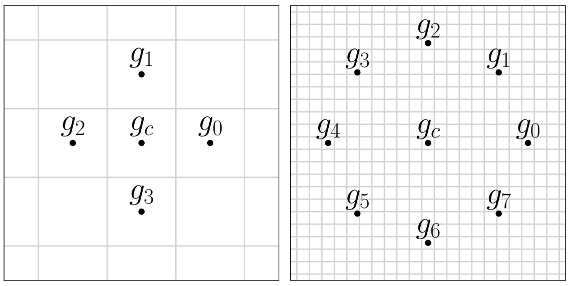


Figure 4.2: LBP operator parameterization examples. R defines the spatial extent around the reference position g_c, and P defines the number of points sampled. On the left, for $P = 4$ and $R = 1$. On the right for $P = 8$ and $R = 8$.

[3]From now on, the terms panorama of textons, panorama of labels, and LBP image are used indistinctively.

After some manipulations of Equation 4.9, texture is approximated by a binary pattern that is invariant to any monotonic gray level variation

$$T \approx t(s(g_0 - g_c), s(g_1 - g_c), ..., s(g_{P-1} - g_c)), \tag{4.10}$$

where $s(x)$ is a step function, i.e.,

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4.11}$$

The variants of the LBP operator presented differ in how the binary pattern is translated into a label. In the first variant, denoted as $\text{LBP}_{P,R}$, the label is obtained by simply converting the binary pattern to base 10, i.e.,

$$\text{LBP}_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p. \tag{4.12}$$

The second LBP variant is a rotation-invariant version of $\text{LBP}_{P,R}$' and is denoted by ${\text{LBP}_{P,R}}^{\text{ri}}$, where ri stands for rotation invariant [4]. In this variant, the binary pattern is rearranged before the conversion to base 10 as

$${\text{LBP}_{P,R}}^{\text{ri}} = \min \left\{ ROR(\text{LBP}_{P,R}, i) | i = 0, 1, ..., P-1 \right\}, \tag{4.13}$$

where $ROR(x, i)$ performs i circular bit-wise right shifts of the binary pattern x. For example, the binary patterns 00100_2, 10000_2, and 00001_2 would all be arranged as 00001_2 and have the label 1.

The two remaining LBP variants are called *uniform*. They differentiate between two types of bit patterns when converting the bit code into a code: uniform and non-uniform. Uniform patterns are assigned a unique label, whereas non-uniform patterns are all assigned a common code, different from those assigned to uniform patterns. A binary pattern is considered uniform if it has at most two spatial transitions from "on" to "off" or vice-versa with wrap-around (Ojala et al., 2002), see Figure 4.3. For example, binary patterns 11111_2 and 00000_2 have zero spatial transitions, whereas patterns 000011_2 and 001100_2 have two. Uniform LBP variants are denoted with superscript u2 to differentiate them from non-uniform ones.

Uniform LBP variants are convenient because they produce a smaller number of labels, and yet they assign a unique label to most texture primitives found in natural scenes. Just like before, there is one variant of uniform LBP that is rotation invariant and one that is not. In the former, denoted as ${\text{LBP}_{P,R}}^{\text{riu2}}$, the label assigned to a uniform binary pattern consists of the number of "on" bits in the pattern. The labels of the latter, denoted as ${\text{LBP}_{P,R}}^{\text{u2}}$, are calculated using Equation (4.12). For example, the binary pattern 000111_2 would have code 8 if using ${\text{LBP}_{P,R}}^{\text{u2}}$, and code 3 if using ${\text{LBP}_{P,R}}^{\text{riu2}}$.

[4]The notation used in this dissertation is slightly different to what is used in the original

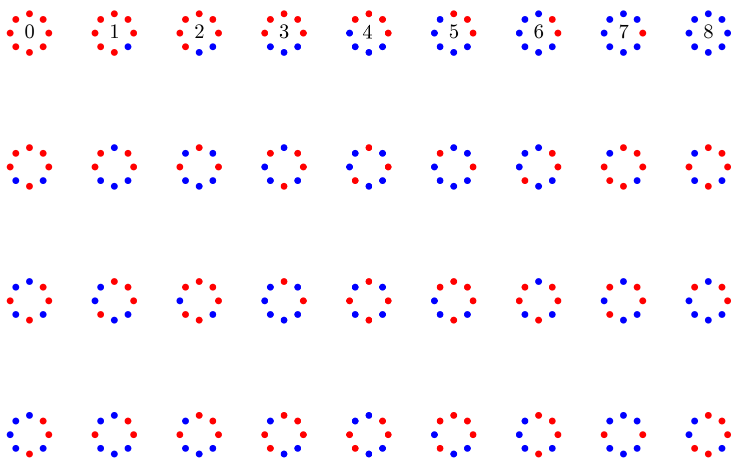


Figure 4.3: Uniform LBP codes $\text{LBP}_{8,R}{}^{riu2}$. Uniform LBP codes are those that have at most two value transitions in the bit pattern. For example, there are only 9 uniform LBP patterns (top row) among all $2^P = 256$ LBP patterns. From the second row to bottom, some examples of non-uniform LBP codes. All of those patterns are assigned LBP code 9.

4.3.2 Panorama of Column Histograms of Labels

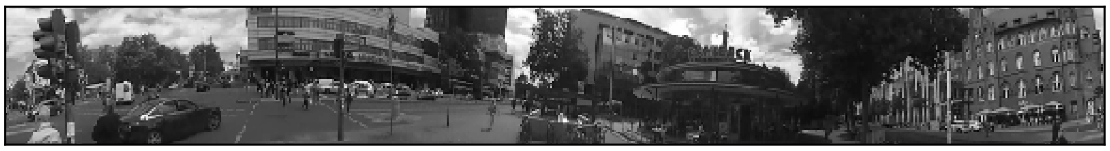
The previous section proposed the use of the LBP operator to create panorama of textons, which are then aligned using an ad-hoc IDF called PLD. That approach is closely related to the so-called hLBPI (Yang and Chen, 2013), which is extremely rare in practice.

Instead, the ubiquitous practice is to use LBP images to compute histogram-based descriptors by concatenating multiple local histograms of LBP codes (Ahonen et al., 2004; Ojala et al., 1996; Yang and Chen, 2013). Interestingly, the use of histogram-based descriptors stems from the necessity of position invariances briefly mentioned in the previous section. In view of this, using histogram-based descriptors can be seen as a corollary to the creating panorama of textons with LBP.

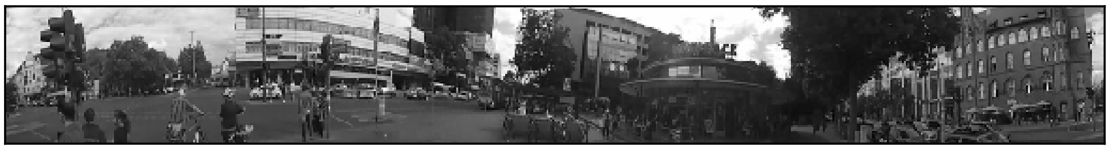
This section introduces a representation called PCHL that is analog to LBP descriptors but takes the horizontal wrap-around in panoramic images into consideration, and integrates seamlessly with the VCA warping.

The process to create a PCHL image takes a panorama of textons of size $W \times H$ and outputs a PCHL of size $W \times (cn)$, where c is the maximum number of possible labels granted by the parameterization of the LBP operator, and n is the number of vertical regions. Each column j of the PCHL is constructed by concatenating n local histograms, each obtained from one of n vertical regions, each of size $s \times (H/n)$ centered horizontally at j, see Figure 4.6.

sources, because simplifies identifying if an LBP variant is either rotation invariant or uniform (explained later).



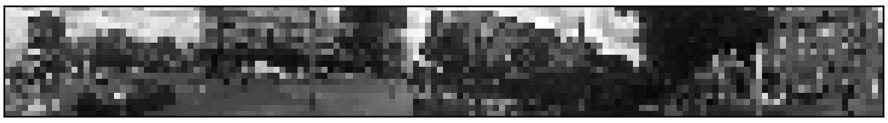
(a)



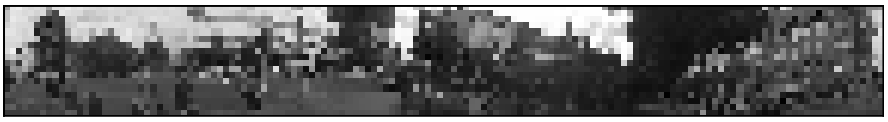
(b)

Figure 4.4: Example of 4.4a snapshot and 4.4b current-view at high resolution before further preprocessing, see Figure 4.5.

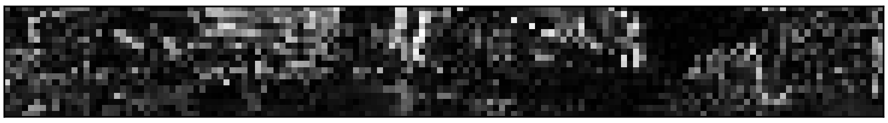
Figure 4.7 compares the panorama of textons with PLD against different parameterizations of glsPCHL. As seen, increasing n, causes the PCHL to approximate the values of the PLD (spatial information is better preserved). In contrast, varying w leads to almost identical IDF values. In this case, no obvious advantage of using this representation is noticeable. On the contrary, using PCHL increases aliasing, in the form of multiple global minima. Figure 4.7 illustrates the same comparison for global localization. Note that the global minima of the panorama of textons is highly localized, whereas the global minima of PCHL is found in a broad peak, besides of its IDFs showing multiple local minima.



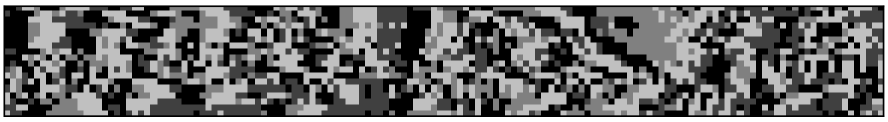
(a)



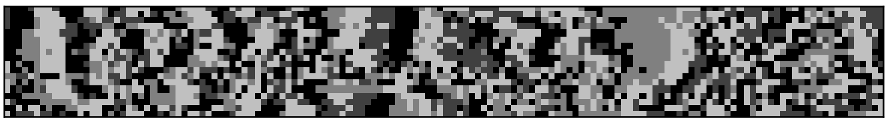
(b)



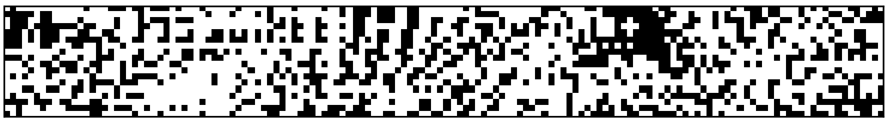
(c)



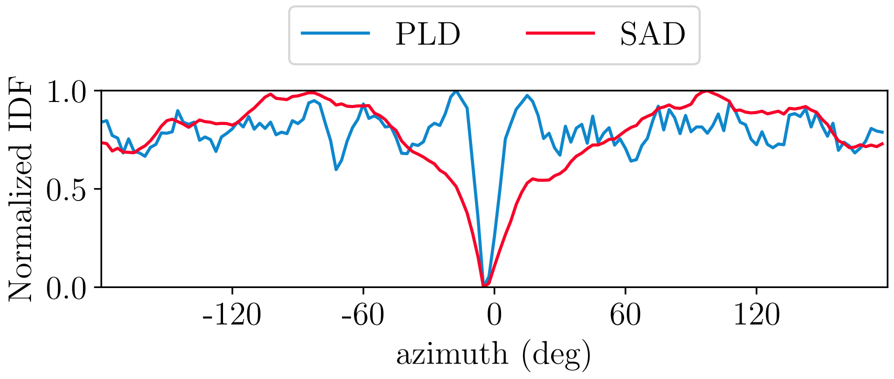
(d)



(e)

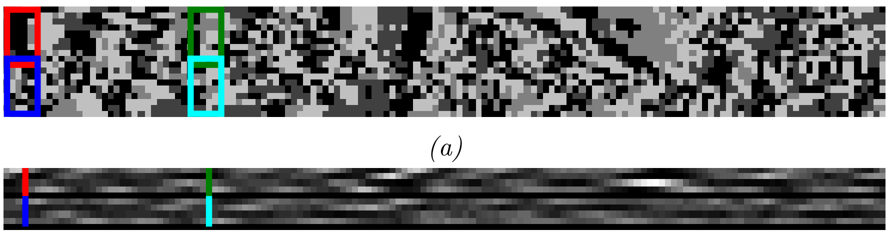


(f)



(g)

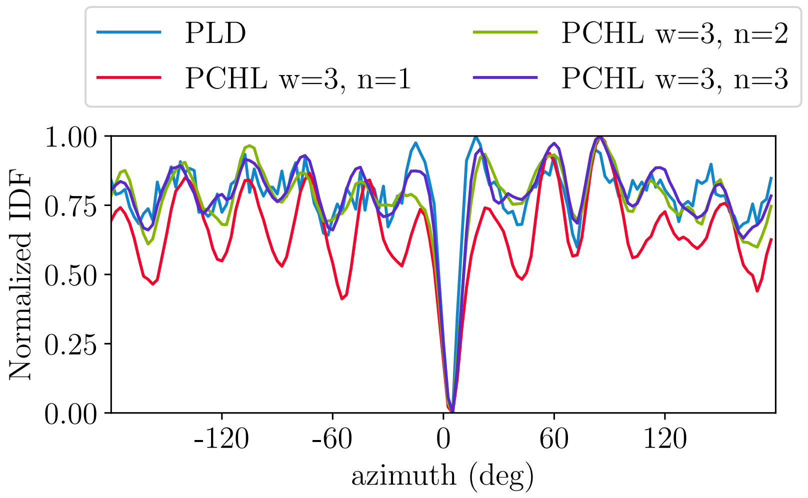
Figure 4.5: PLD on panorama of textons compared to SAD on raw images. 4.5a snapshot and 4.5b current-view at low-resolution raw images (see Figure 4.4). 4.5c pixel-wise differences at the best alignment in the SAD sense. 4.5d snapshot and 4.5e current-view as panorama of textons, extracted from 4.5a and 4.5b, respectively. Both images were artificially colored so that every label (LBP code) has a distinctive color. 4.5f pixel-wise differences at the best alignment in the PLD sense. White pixels stand for corresponding pixels with the same label, and conversely for black. Matching labels tend to group together, though most of them do not match. 4.5g Normalized PLD and SAD obtained during alignment. PLD with panorama of textons exhibits less aliasing in the global minima.



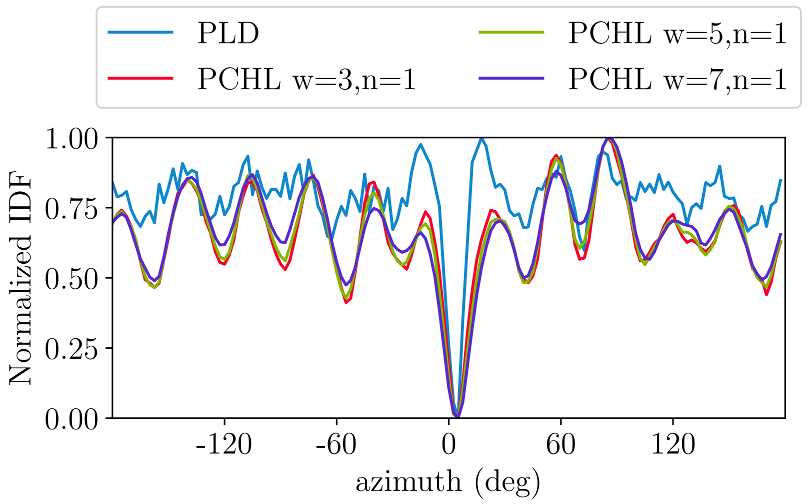
(a)

(b)

Figure 4.6: Histograms of textons computation with $n = 2$ and $s = 5$. 4.6a LBP image used to compute a PCHL. 4.6b the resulting PCHL. The regions in red and blue correspond to the same column, and likewise for those in green and cyan. The histograms obtained by each region in the LBP image are concatenated to form the corresponding column of the PCHL.

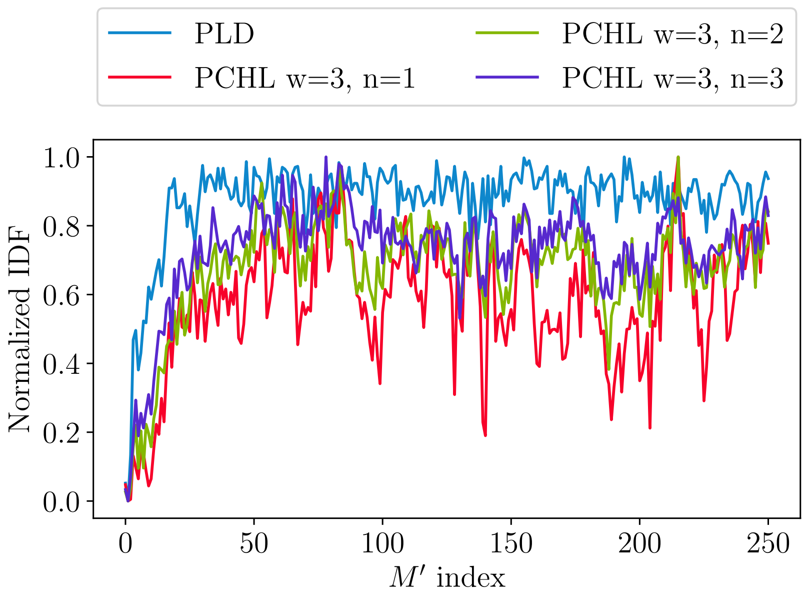


(a)

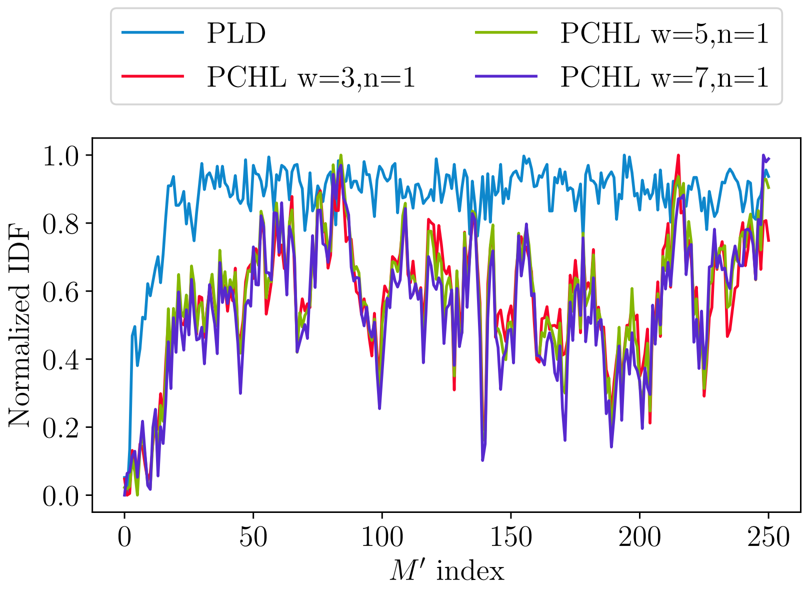


(b)

Figure 4.7: Comparison of panorama of textons with normalized PLD and different PCHL parameterizations with normalized SAD obtained from aligning two images. 4.7a for fixed w and varying n. 4.7b for fixed n and varying w.



(a)



(b)

Figure 4.8: Comparison of panorama of textons with normalized PLD and different PCHL parameterizations with normalized SAD obtained during global localization, only the IDF of best match is displayed. The true best match is the first snapshot (index 0) 4.7a for fixed w and varying n. 4.7b for fixed n and varying w.

4.4 Estimation of Off-Route Translation Direction

The VCA results useful to calculate a corrective steering command to travel along a route if the robot is on the route. Unfortunately, it is of limited usability to bring back the robot to the route, i.e., to navigate towards the closest waypoint after straying away. The reason for this is that the relative orientation estimated by the VCA is relative to viewing direction while on-route. If the robot has strayed away, the VCA would yield a relative orientation that causes the robot to travel parallel to the route, instead of going back to it (Zeil et al., 2003), see Figure. Wystrach et al. (2012); Möller and Vardy (2006a) proposed to perform probatory translatory movements and use the IDF values obtained after them to estimate the direction of travel based on the direction IDF change. If translating in a direction decreases the IDF value, that direction leads to the goal, and vice-versa, see Figure 4.9. Unfortunately, such movements may have to be performed at random. Unfortunately, the performance of random movements is undesirable in applications like smart-wheelchair navigation because they would render the driving behavior less predictable, thus less safe (Surden and Williams, 2016). Moreover, performing the necessary movements carries complications, especially for non-holonomic robots like the ROSStuhl. Another complication of the random movement approach is that it requires the acquisition conditions to be stable so that the gradient of IDF reflects the direction of travel. However, this is hardly expectable in real-world applications. Other authors have proposed the use of more warping functions that operate on multiple scales (Möller and Vardy, 2006b; Churchill and Vardy, 2008; Möller et al., 2014).

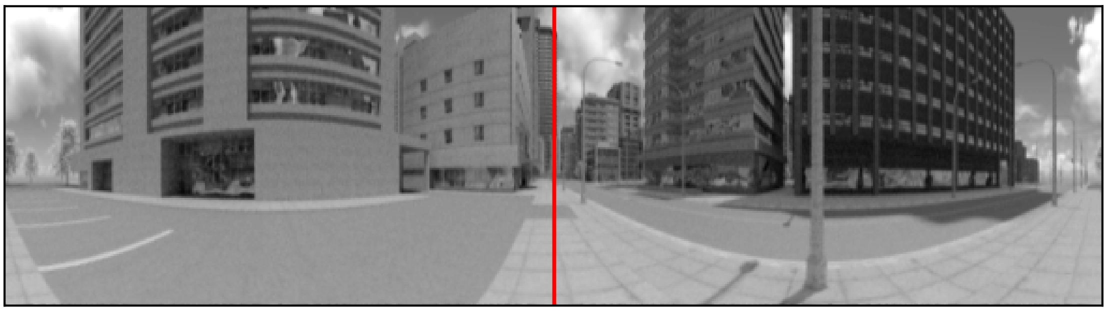
This section introduces an algorithm that estimates the direction of off-route displacement given only a current-view c, and snapshot that represents a nearby waypoint along the route m_l. The benefit is that it does not rely on probatory movements nor multi-scale analysis. The computation consists of the following steps. First, c and m_l are aligned using the relative rotation estimated by the VCA. This step is analog to the “rotating on the spot” observed in ants by Wystrach et al. (2012). The second step consists of computing the horizontal displacement of corresponding columns, i.e., those for which IDF is minimal. Spurious matches are filtered out using the ratio of IDF values of the best and second-best match, similarly as when matching SIFT features (Lowe, 2004), and by the magnitude of the displacements. Since this procedure is meant to be used to correct small off-route deviations, the displacement can be expected to be constrained to relatively low magnitudes for distant objects, i.e., those that are most likely to reoccur in both images.

This process is performed independently over two subimages that correspond to the front and back of the panoramic image because optical flow would have opposite directions in them. Finally, the estimates of both regions are merged as

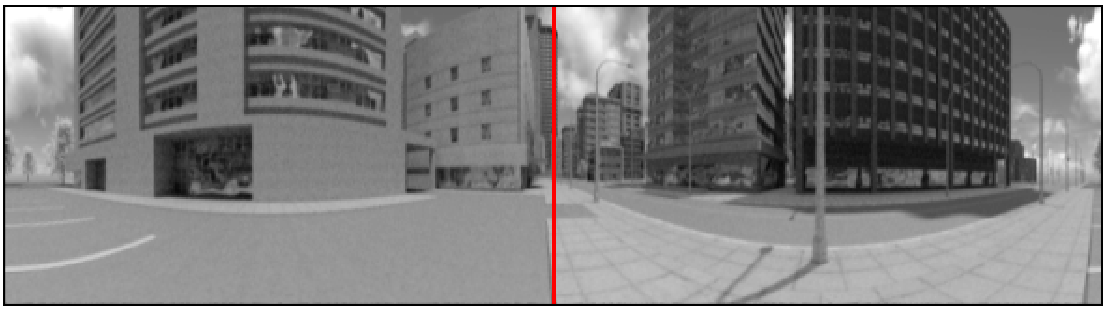
$$T_x = t_x(front(c), front(m_l), 1) + t_x(back(c), back(m_l), -1), \tag{4.14}$$

where $front$ and $back$ extract the corresponding subimages from the snapshot m_l and the current-view c. $front$ simply crops the image to preserve the area in the center. $back$ first rotates the image by 180 degrees in azimuth before cropping in the same way. k provides invariance to the direction of perceived column displacement, see Algorithm 1. The sign of T_x is positive if the robot is placed to the right of the route, and negative if it placed to the left.

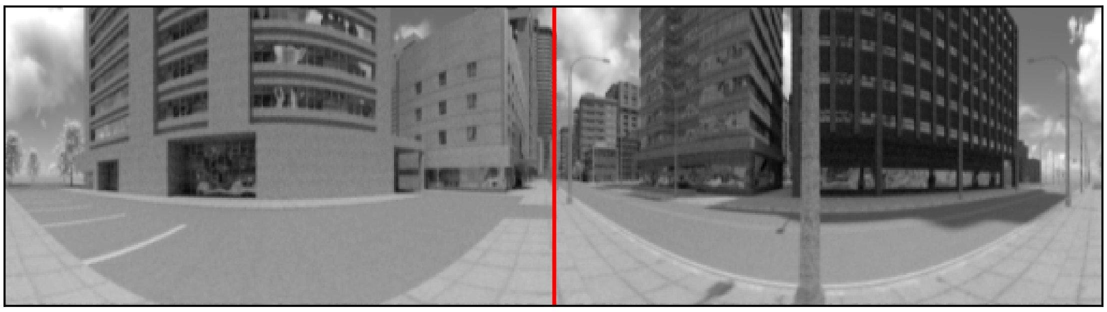
```
Function $t_x(m_l, c, k)$:
    input : A snapshot $m_l$ and a current-view $c$ of size $W \times H$ px. $k$
            should be positive for images around the focus of expansion,
            and negative for images around the focus of contraction.
    output: A real number whose sign indicates the direction of
            horizontal optical flow, whose magnitude indicates
            confidence.
    $P \leftarrow 0$
    $N \leftarrow 0$
    for $i \leftarrow$ to $W - 1$ do
        $D \leftarrow \{\}$
        $\Delta \leftarrow \{\}$
        for $j \leftarrow 0$ to $W - 1$ do
            $d \leftarrow idf(m_{*,i}, c_{*,j})$
            $\delta \leftarrow k(i - j)$
            Append $d$ to $D$
            Append $\delta$ to $\Delta$
        end
        $r \leftarrow argsort(D)$
        $sbr \leftarrow log(D_{r_0}/D_{r_1}$
        if $\Delta_{r_0} \leq 5 \ \wedge sbr \leq rthres \wedge \Delta_{r_0} \neq 0$ then
            if $\delta < 0$ then
                $P \leftarrow P + 1/d$
            end
            else
                $N \leftarrow N + 1/d$
            end
        end
    end
    return $log(P, N)$
```
Algorithm 1: Algorithm used to estimate the direction of off-route translation.



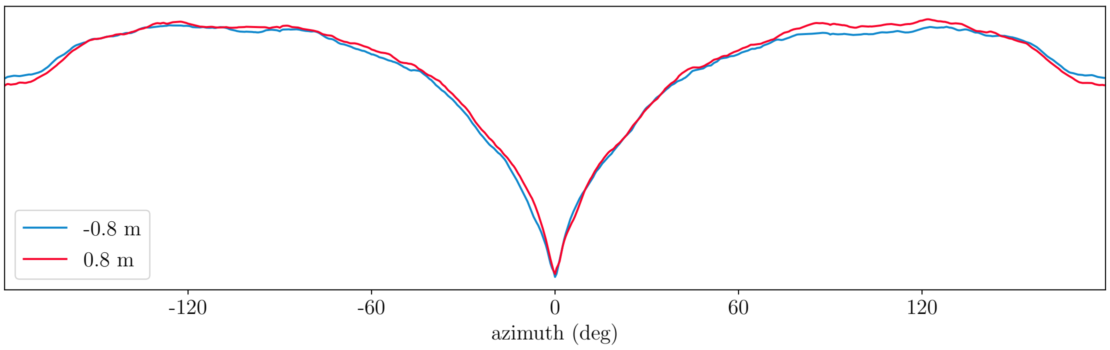
(a)



(b)

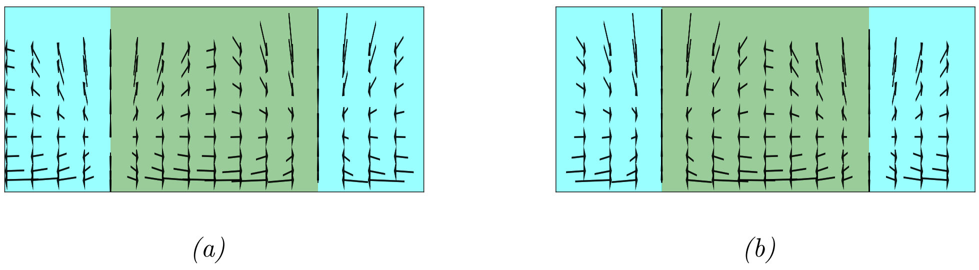


(c)



(d)

Figure 4.9: SAD IDF values under pure off-route translation. The VCA only accounts for orientation. Thus, under pure off-route translation, a robot navigating with the VCA would move forward instead of going back to the route by minimizing the off-route translation. 4.9a snapshot. 4.9b and 4.9c current-views translated 0.8 m and -0.8 m w.r.t. the snapshot, respectively. 4.9d SAD values obtained by aligning 4.9a with 4.9b and 4.9c.



(a) *(b)*

Figure 4.10: Optical flow under off-route translation. The optical flow at the front (green) has the opposite sign as the spatial displacement, and optical flow at the back (cyan) has the same sign. 4.10a optical flow resulting when the robot translates to the left. 4.10b optical flow resulting when the robot translates to the right.

4.5 Knowledge Transfer

Research on the VCA assumes that the same setup is used to capture the visual memories and to navigate with them. That assumption is reasonable in biology, but it is not adequate in robotics, and holding to it has negative implications. In single-robot applications, the constant setup assumption may break, for example, due to the replacement of a defective camera. In that case, it may be necessary to capture all the visual memories again. In multiple robot applications, matters are even worse because every robot comprises a unique setup, and therefore requires a unique version of the visual memories. Thus, the constant setup assumption inhibits the use of the VCA in real-world, large-scale applications because they invariably involve multiple setups in one form or the other.

This investigation introduces KT as a problem of study in the context of the VCA. KT consists of navigating when the setup used for visual memory acquisition (called reference) and autonomous navigation (called alternative) are different and of any steps that facilitate that process.

Before going into details, it is necessary to place KT for the VCA in the context of related problems. First of all, KT is not a new problem in more general terms because its importance is readily acknowledged in other types of navigation. For example, to use 3D maps generated using expensive lidar sensors for localization using low-cost monocular directional cameras (Wolcott and Eustice, 2014; Caselitz et al., 2016), or directional stereo cameras (Xu et al., 2017). The problem also relates to the omnidirectional stereo problem, which has been studied at least since the late ninety nineties (Benosman et al., 1996; Gluckman and Nayar, 1998), and continues to be an active research topic (Shih-Schon Lin and Bajcsy, 2003; Wang et al., 2012). However, KT for the VCA is arguably a more difficult problem than omnidirectional stereo because, in KT, it is not possible to use a structured pattern for calibration and because of the complete lack of time synchronization and stable arrangement of the omnidirectional cameras over time. Moreover, stereo systems are generally designed using cameras that have nearly identical intrinsic parameters, whereas this dissertation later shows, this needs not to be the case for the VCA. It is also worth noting that there are previous investigations on the VCA that, although they did not formulate KT as an explicit problem, set a precedent to it. They are discussed where appropriate later in this section.

Navigating with different setups is a more challenging problem because it involves all complications found when using the same setup, with the addition of error sources due to setup differences. Fortunately, they are primarily systematic, and thus correctable. A complete KT requires correcting four types of error sources, each related to different types of parameters. The types of parameters are enlisted in order of increasing difficulty to solve. Each of them is described next, assuming that they are the only difference between reference and alternative setup.

The first type is the photometric characteristics of the image sensors. They cause that corresponding scene points are assigned different intensity values. Pa-

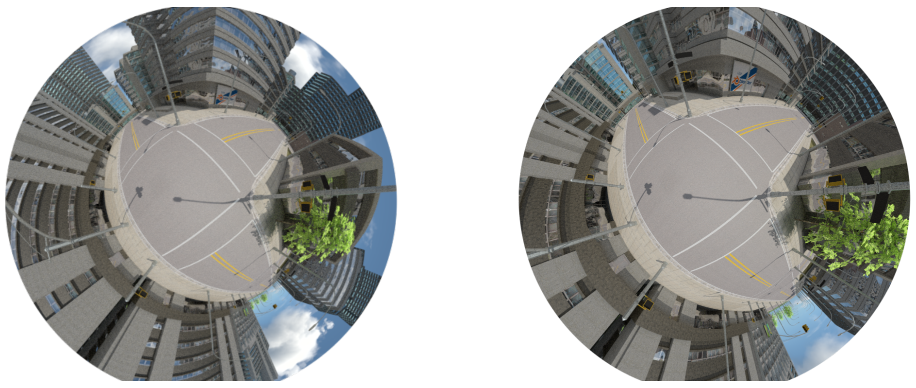


Figure 4.11: Two synthetic omnidirectional images captured at the exact same pose with two different omnidirectional cameras. Alignment of images produced by different setups has not yet be attempted with the visual compass algorithm, however it would greatly simplify large-scale, real-world applications, like smart-wheelchair navigation.

rameters of this type include temporal dark noise, quantum efficiency, dynamic range (Jähne, 2010). They are placed first because they are, to some extent, analog to navigating under different illumination conditions. Research in the VCA shows that the algorithm performs well under those conditions using simple illumination invariant representations (Zeil et al., 2003). Additionally, some of them, like vignetting, can be corrected through a photometric calibration.

The second type relates to intrinsic parameters of the omnidirectional cameras, like the distortion parameters and the field of view. Wystrach et al. (2016) analyzed the effects of FOV and resolution, setting thus a precedent. They are placed second, because similarly to the photometric parameters, they can be estimated for each camera independently and because there are established methods designed to estimate them. However, as discussed later on, research in the VCA has neglected this aspect.

In the third place, there are the extrinsic parameters of the omnidirectional camera, w.r.t. the coordinate system of the robot. An omnidirectional camera used with the VCA has to be installed in a particular way: optical axis normal to the ground plane and aligned with the axis of rotation of the robot. The yaw of the sensor does not require a particular aligment, but it is generally aligned with the xy plane in robot coordinates, and so that the forward direction of the robot corresponds with the top of the omnidirectional image.

Although a perfect sensor alignment is hardly attainable in practice, it can be expected to be precise, and thus discrepancies in pitch, roll, and yaw, and x and y translation can be expected to be minimal. Translation along z is perhaps the extrinsic parameters that may have a larger margin of variation. Previous studies in the VCA can be considered as a precedent in this area. They can be separated into two groups. The first group studied the effects of planar view-point changes. That group constitutes the majority by a large margin because the VCA is intended to operate under planar motion. The second group studied the effects of non-planar view-point changes, i.e., along the z-axis (Freas et al., 2018), pitch(Ardin et al., 2015), and roll (Raderschall et al., 2016). The results offer some insights that may be useful for KT: All types of viewpoint changes, except for yaw, are detrimental to image matching. They cause the IDF values of the best match to becomes less salient (increasing aliasing). Moreover, the size of that effect consistently increases with the magnitude of the viewpoint change. This type of parameters is considered more difficult because estimating the viewpoint differences requires images acquired by both setups. The correlation between the magnitude of viewpoint change and its impact on IDF makes it reasonable to think of solving this problem using a gradient-based method. However, this was not pursued covered in this investigation.

Finally, the fourth type is differences due to the embodiment of the robot, including kinematics and the shape and size of the robot. For example, if the reference setup was a small differential robot, and the alternative setup was a large Ackerman robot, they may not be able to follow the same routes. In the opinion of the author, differences of this type are probably the hardest to solve, in part because the VCA does not use any kind of metric information for its operation.

4.5.1 Intrinsic Rectification for the VCA

This investigation considers only intrinsic parameter differences. The author considers it to be a sensible first step because it is arguably a simpler problem. Moreover, this problem needs to be solved first, if at all, to produce panoramic images of matching sizes to pass to the VCA and to simplify matters by removing nonlinearities. Thus, correcting intrinsics is likely to simplify the correction of extrinsic and can be seen as a preliminary step.

Furthermore, the role of intrinsic parameters has been neglected so far in research of the VCA. The author is only aware of few publications that provide details on the intrinsic model and the unwrapping procedure. During the latest stages of writing this document, the author became aware of the work by Fleer (2017), in which the authors use the Scaramuzza model to unwrap panoramic images and perform intrinsic calibration of the camera to correct differences due to intrinsic parameters. However, they did not use it with knowledge transfer in consideration.

This seems justifiable because the VCA and similar have been found to perform well without relying on a precise intrinsic model. In some cases, that robustness to imprecisions in the model is a rationale behind the selection of those algorithms (Scaramuzza and Siegwart, 2008).

However, this can hardly be expected if the reference and alternative setups have very different intrinsic parameters. Moreover, even if a constant setup is used, it is valid to question whether using a precise intrinsic model provides any benefit.

Unwrapping using a Naive Model

The intrinsic model is used in the VCA to unwrap the omnidirectional images, i.e., mapping the visual information of the omnidirectional image, into a panoramic image. In the following, subscript p is used to distinguish symbols specific to the panoramic image. Unwrapping can be understood as a simple polar to cylindrical coordinate conversion. However, it is described with a seemingly overcomplicated notation and unnecessary steps for reasons that will become apparent in the following two sections. Moreover, the model expects that the top and right of the image correspond with the front and right of the robot, respectively. In cases where that is not possible, the images have to be flipped first.

The main unwrapping process used in this dissertation relies on a *naive* model with the following basic parameters. The pixel coordinates of the center of the image $t = (x_c, y_c)^T$, the AOV α, the radius of the border ρ_{bor}, i.e., up to the AOV, and the radius at which the image contains only valid pixels ρ_{max}. All parameters are given in pixels, except for α, which is given in radians. The parameters of the naive model are obtained by human intervention. α is taken directly from the datasheet of the omnidirectional camera, whereas the other parameters are labeled by hand. Besides, the model has two parameters that are derived from the basic ones: the maximum resolutions in azimuth ψ^*, and elevation θ^*. The panoramic image is unwrapped so that it contains only valid pixels (including possible occlusions at the center). Therefore, its width and height are $W_p = 2\pi/\psi^*$ and $H_p = \alpha/2/\theta^*$, respectively.

Coordinates of the omnidirectional image are denoted by $\mathbf{u} = (u, v)^T$, and the coordinates of the panoramic image by $\mathbf{u}_p = (u_p, v_p)^T$. Panoramic image columns relate to azimuth ψ by

$$\psi = -\psi^* \left(u_p - \frac{W_p}{2} \right), \tag{4.15}$$

and to elevation θ by

$$\theta = \theta^* \left(W_p - v_p - 1 \right). \tag{4.16}$$

In that way, azimuth zero is located at the center of the panoramic image, with positive values to the left and negative values to the right in steps of ψ^*. Moreover elevation, starts at zero at the bottom and increases in steps of size θ^* towards the top.

Elevation value is converted to the corresponding radius ρ in the omnidirectional image using the following equation

$$\rho = \frac{\theta}{\theta^*}. \tag{4.17}$$

Finally, the polar coordinates (ψ, ρ) are converted to omnidirectional image coordinates

$$u = \rho \cos\left(\psi + \frac{\pi}{2} \right) + x_c, \tag{4.18}$$

$$v = \rho \sin\left(\psi + \frac{\pi}{2}\right) + y_c. \tag{4.19}$$

The naive model is only an approximation, as it disregards nuances present in real omnidirectional cameras, like optical distortion and misalignment of the image acquisition components. The clearest indication of that is the assumption of a linear relation between radius and elevation in Equation 4.17. Nevertheless, unwrapping omnidirectional images in this way a common practice. In the contexts of the VCA, this is the case because the algorithm is known to perform well in the absence of a precise unwrapping. However, this practice may also be common in a broader context. For example, the omnidirectional camera calibration toolbox (Scaramuzza et al., 2006b) provides code to unwrap panoramic images using a simple polar to cylindrical coordinate conversion, despite the availability of a precise model of the camera. However, using a naive model to unwrap images is not appropriate in the context of KT for several reasons. The first reason is the AOV of calibrated sensors may deviate from vendor specifications. The nuances of real-world cameras may vary so that pixel correspondences cannot be maintained, even if extrinsic parameters are kept equal. The following two sections describe the model implemented in Scaramuzza et al. (2006b) and how it can be used to unwrap panoramic images that preserve pixel correspondences.

Scaramuzza Omnidirectional Camera Model

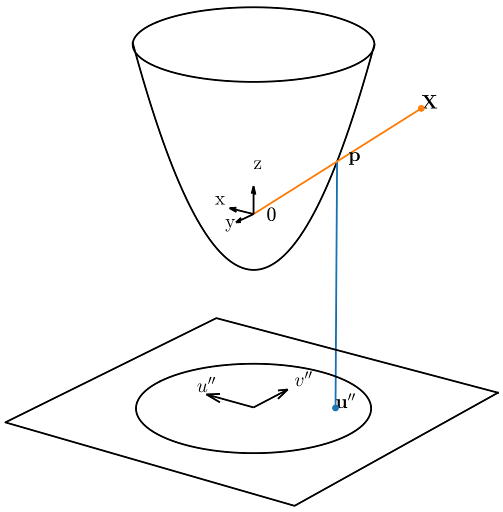
Scaramuzza et al. (2006a) introduced an omnidirectional calibration model, referred to as the *Scaramuzza model*, and later released as a publicly available toolbox that estimates its parameters (Scaramuzza et al., 2006b) [5]. One important benefit of the Scaramuzza model is that it can represent different types of central omnidirectional cameras using a common abstraction. Central cameras are those that fulfill the single center of projection property, i.e., all rays pass through a common point (the center of projection) before reaching the sensor. Examples of such systems include the combination of a hyperbolic mirror and a perspective camera, a parabolic mirror and a telecentric camera, and some types of fisheye cameras[6].

The Scaramuzza model allows calculating the direction of a scene point $\mathbf{X}$, given the sensor coordinates of its image $\mathbf{u}' = (u', v')$. The first part of the model corrects misalignments of the camera components by transforming the sensor coordinates into ideal omnidirectional image coordinates $\mathbf{u}'' = (u'', v'')$., i.e., perfectly centered and with a circular border (see Figure 4.12)

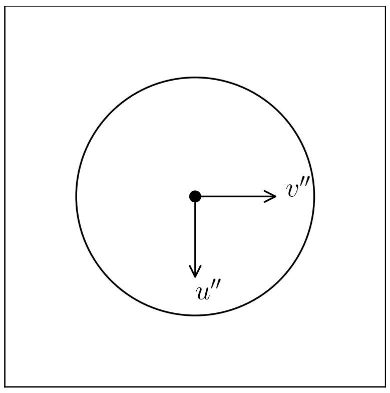
$$\begin{pmatrix} u'' \\ v'' \end{pmatrix} = \begin{pmatrix} c & d \\ e & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} u' \\ v' \end{pmatrix} - \begin{pmatrix} x_c \\ y_c \end{pmatrix} \right] = \mathbf{A}^{-1}(\mathbf{t} - \mathbf{u}'), \tag{4.20}$$

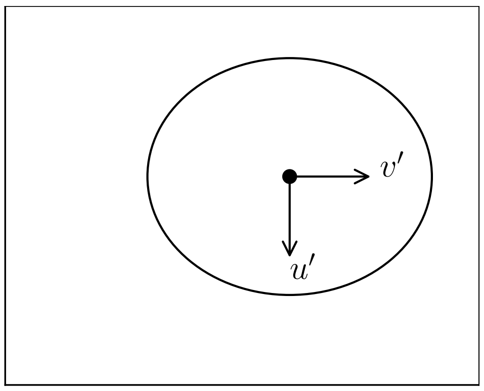
where $\mathbf{t} = (x_c, y_c)$ is the center of the image, and c, d, and e are the parameters of linear map $\mathbf{A}$. The relation between sensor coordinates $\mathbf{u}'$, and a vector $\mathbf{p}$ that

[5]The definitions provided in this section are derived from the implemention in the toolbox.
[6]The model cannot calibrate fisheye cameras if their AOV exceeds 195 degrees.



(a)





(b) (c)

Figure 4.12: Coordinate systems in the Scaramuzza model. 4.12a Ideal image coordinates and viewpoint. 4.12c Ideal image coordinates, i.e. the omnidirectional is perfectly centered and has a circular border. 4.12b Sensor coordinates. Real world cameras may no produce ideal images due to misalignment of its component. The effect has been exaggerated for illustrative purposes.

emanates from the viewpoint **O** in the direction of a scene point **X** is defined, up to a scaling factor λ, $\lambda > 0$, by

$$\lambda \mathbf{p} = \lambda \begin{pmatrix} u'' \\ v'' \\ w'' \end{pmatrix} = \begin{pmatrix} u'' \\ v'' \\ f(u'', v'') \end{pmatrix} \tag{4.21}$$

where f is a function that maps ideal coordinates to a distance along z, depending on their distance w.r.t. the center. The authors define f as

$$f(u'', v'') = a_0 + a_1 \rho'' + a_2 \rho''^2 + ... + a_N \rho''^N, \tag{4.22}$$

where $\rho'' = \sqrt{u''^2 + v''^2}$, and $\{a_i\}_{i=0}^N$ are the coefficients of an N-th degree polynomial.

Unwrapping Panoramic Images using the Scaramuzza Model

This section proposes a procedure to unwrap panoramic images using the Scaramuzza model. This has two main advantages over the approach based on the naive method. The first one is that the Scamuzza method better resembles the image acquisition of a real-world camera. The second is that the parameters of the Scaramuzza model depend on human intervention, only to acquire the data used for optimization. In contrast, the parameters of the naive method rely entirely on a human.

Unwrapping with the Scaramuzza model also consists of transforming coordinates in the panoramic image $\mathbf{u_p} = (u_p, v_p)$, into the corresponding sensor coordinates of the omnidirectional image $\mathbf{u}'$. The generation of panoramic image coordinates and their conversion to azimuth and elevation are identical to the naive model (see Equation 4.15 and Equation 4.16).

However, there are two key differences. The first one is that the Scaramuzza model differentiates between sensor and ideal coordinates, whereas the naive approach makes no such distinction. The second is that radius and elevation are not assumed to be linearly related. Instead, the radius corresponding to an elevation is calculated as

$$\rho'' = h(\theta). \tag{4.23}$$

h is estimated by piecewise linear interpolation between all elevations present in the panoramic image, and the corresponding radii

$$\theta = \tan^{-1}\left(\rho'', -f(u'', v'')\right), \tag{4.24}$$

where $\tan^{-1}$ is the two-argument inverse tangent function, called `atan2` in some programming languages. In the next step, the polar coordinates (ψ, ρ'') are converted to ideal omnidirectional image coordinates by

$$u'' = \rho'' \cos\left(\psi + \frac{\pi}{2}\right) \tag{4.25}$$

$$v'' = \rho'' \sin\left(\psi + \frac{\pi}{2}\right) \tag{4.26}$$

Finally, the ideal omnidirectional coordinates are converted to sensor coordinates using $\mathbf{A}$, and then referenced relative to the top-left corner by adding $\mathbf{t}$.

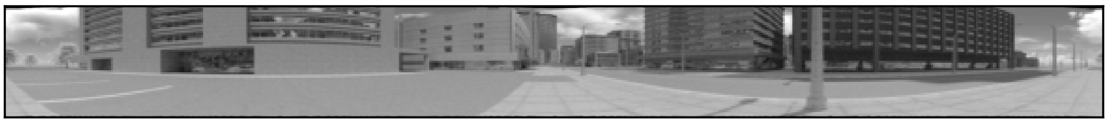
$$\mathbf{u}' = \mathbf{A}\mathbf{u}' + \mathbf{t}. \tag{4.27}$$

The mapping performed by Equation 4.23 could have also been performed using an inverse function $f^{-1} : \rho'' \to z$ that is part of the Scaramuzza model, and then Equation 4.24. However, not all the calibration models used in this dissertation were provided with the parameters for $f^{-}1$. The Scaramuzza toolbox provides a method for approximating f^{-1}, but the author of this dissertation found the piecewise linear interpolation explained above easier to implement and adequate for panoramic image unwrapping.

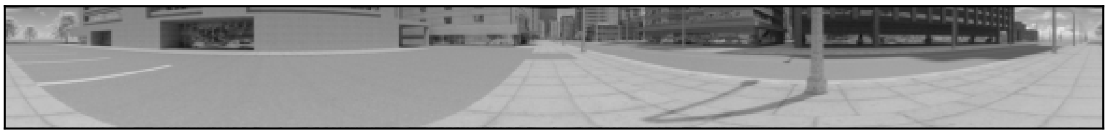
4.5.2 Complete Pipeline

This section provides the complete procedure used to rectify panoramic images acquired by different setups so that differences in intrinsic parameters have a minimal impact on the IDF values. For simplicity, the procedure is explained as if every step produced an intermediate result image. In practice, it is possible to avoid the generation of intermediate images.

The first step consists of unwrapping each of the omnidirectional images as described in the previous section.



(a)

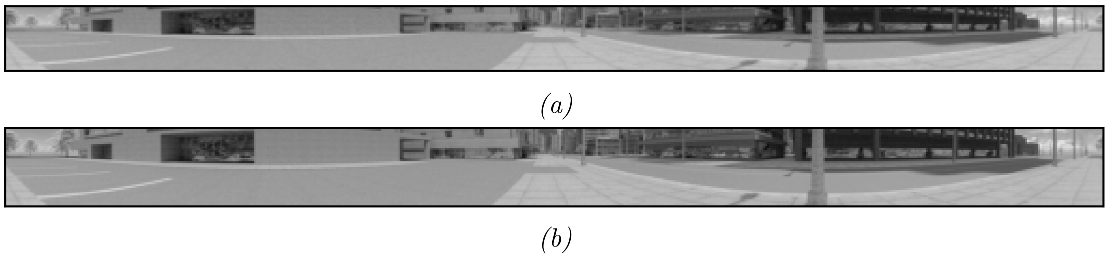


(b)

Figure 4.13: Panoramic images unwrapped using the Scaramuzza method. The Zhang2006 camera is displayed in 4.13a and the Zivkovic2005 camera in 4.13b.

The second step consists of equalizing the FOV of the images. For this step, let α_{min} and α_{max} be the AOV that contains only valid pixels in the reference setup. The corresponding symbols of the alternative setup are differentiated by a tick in the following.

Both panoramic images have to be cropped so that they contain elevations in the range $[\max(\alpha_{min}, \alpha'_{min}), \min(\alpha_{max}, \alpha'_{max})]$. The resulting elevation values are converted to a radius for each of the panoramic images using Equation 4.23. After this step, both images show the same part of the scene. However, pixel correspondences may not be accurate due to differences in resolution.



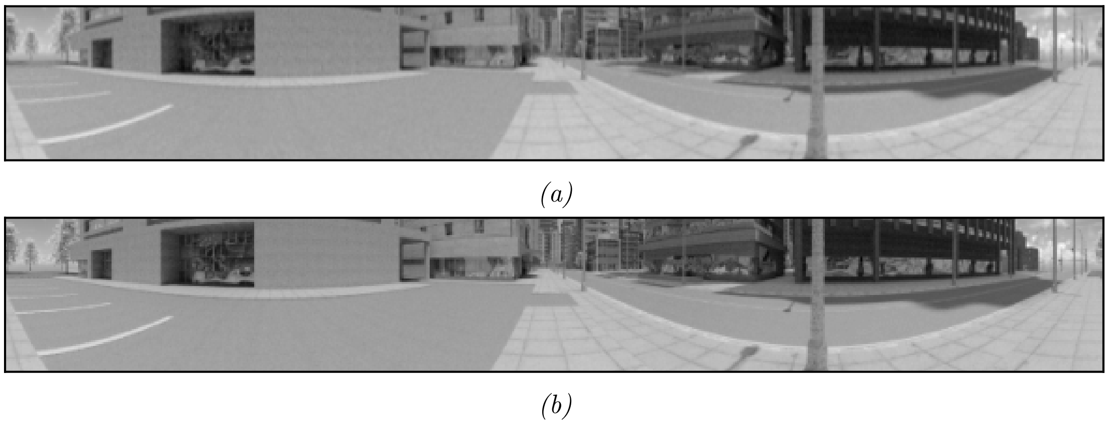
(a)

(b)

Figure 4.14: Panoramic image after vertical FOV equalization. Both images show the same part of the scene. Note that the images were generated at full resolution in azimuth and elevation. Because of that, the resolution in elevation and azimuth is different. The Zhang2006 camera is displayed in 4.14a and the Zivkovic2005 camera in 4.14b.

The third step consists of equalizing the resolution by scaling the panoramic images to a common resolution κ^*.

The scaling factors in azimuth and elevation of the reference image are calculated as $\widehat{\psi} = \kappa^*/\psi^*$, and $\widehat{\theta} = \kappa^*/\theta^*$, respectively. The scaling factors of the alternative image are likewise calculated using its respective resolutions ψ'^*, and θ'^*. After that, both images have the same size, and corresponding pixels in the two images depict the same part of the scene. See Figure 4.15.



(a)

(b)

Figure 4.15: Panoramic image after resolution equalization. After this stage the images can be processed with the visual compass algorithm because they have the same width and height. Both images show the same part of the scene at the same resolution. Resolution in azimuth and elevation was also equalized, thus the different aspect ratio compared to the images in 4.14. The Zhang2006 camera is displayed in 4.15a and the Zivkovic2005 camera in 4.15b.

Figure 4.16 shows the IDF values obtained by aligning the images in Figure 4.15 using the SAD metric. Note that although the images appear identical, their IDF value at the best alignment is not zero. The reason for that is that intensity values of the pixel are no identical.

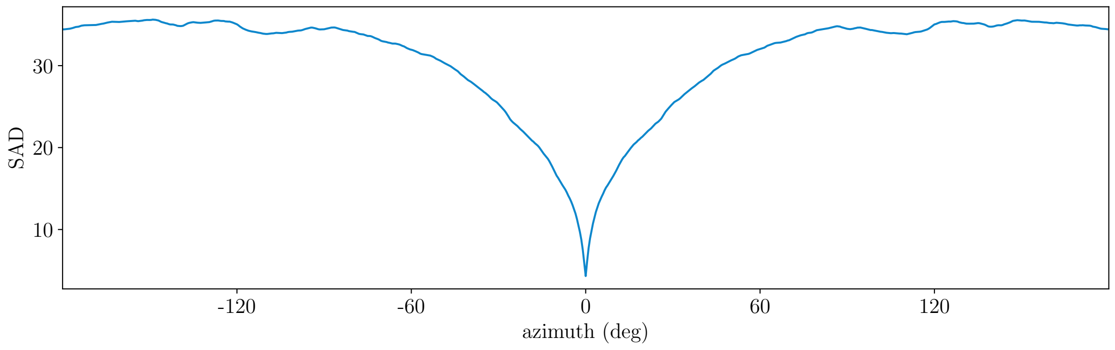
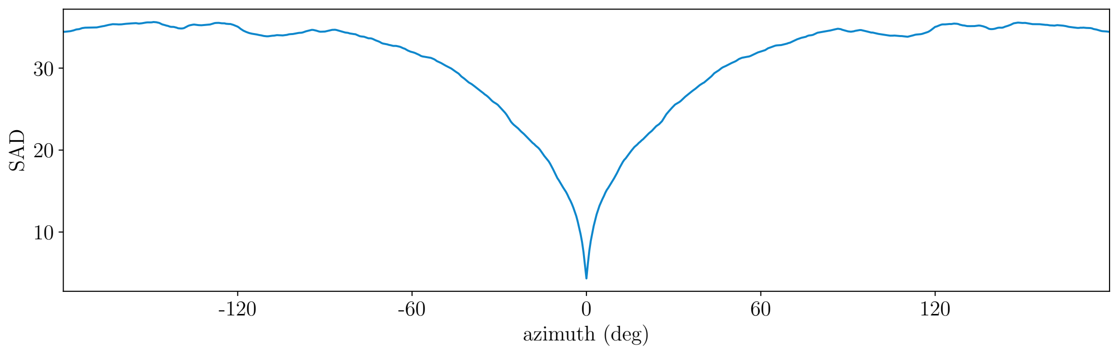
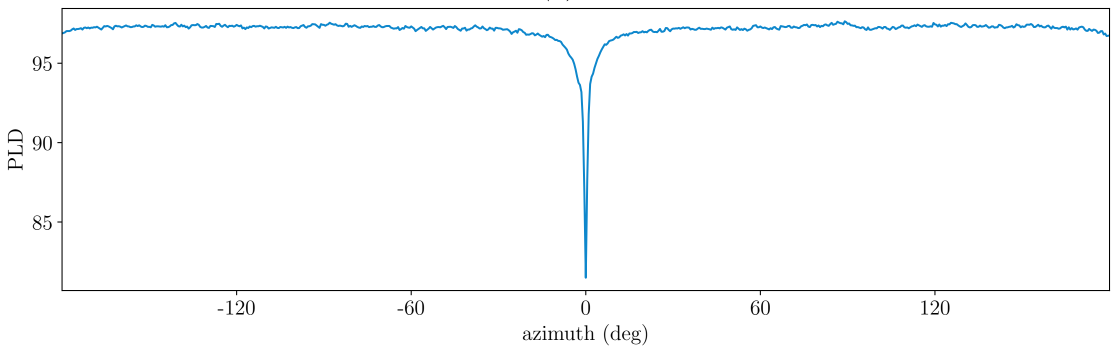


Figure 4.16: IDF values resulting from using the SAD IDF on two panoramic images.

The last step of the pipeline consists of providing illumination invariance by using an adequate representation. Figure 4.17 shows the IDF values obtained on two different illumination invariant representations.



(a)



(b)

Figure 4.17: IDF values of two intrinsically rectified images using illumination-invariant representations. 4.17a using the SAD on images with zero-mean intensity. 4.17b using the PLD on a panorama of textons extracted using $LBP_{8,3}$.

Chapter 5

Experimental Evaluation

5.1 Global Localization Evaluation

The analysis is performed on data obtained by aligning all panoramic images of two visual memories against each other. The images of one of the visual memories, denoted by M, serve as snapshots, and the images of the other, denoted by C, are used as current-views.

The results of alignment are summarized in two matrices, IDF matrix $D \in \mathbb{R}^{|M| \times |C|}$ and shift matrix $S \in \mathbb{N}^{|M| \times |C|}$. The evaluation focuses only on the correspondence of snapshots and current-views, and not on the relative orientation estimation. Thus the S matrix is not used. Element $D_{i,j}$ contains the IDF value at the best alignment between snapshot M_i and current-view c_j. The corresponding element in S, contains the shift of the best alignment. The localization hypothesis for the j-th current-view is obtained as $l_j = \arg\min_i D_{i,j}$, see Section 4.1.

Moreover, the analysis requires of ground truth correspondences for the elements of M and C. Groundtruth is a vector $g \in \mathbb{N}^{|C|}$, where each element g_j contains the true best match, i.e., m_{g_j} is the best match of c_j. For real-world data, ground truth was defined manually on key current-views, and the remaining correspondences were obtained by interpolating the gaps using travel distance. First, the correspondences at the start and end of M and C are set. Then, the remaining correspondences were interpolated, and the results inspected visually. Further key correspondences were added in sections where the correspondences are not correct, until all correspondences are deemed correct. In the case of synthetic data, ground truth is known after generation because the world coordinates of waypoints are precisely known. The localization error is calculated as

$$e_j = l_j - g_j. \tag{5.1}$$

5.2 Global Localization and Failure Detection Evaluation

The performance of localization error detection is measured using recall at precision of 1, denoted by $r@p1$ Milford et al. (2014). $r@p1$ takes values in the range $[0..1]$ and represents the proportion of correct localizations without allowing for false positives. The calculation of $r@p1$ depends on two thresholds. The first one, ϵ, represents the maximum allowed localization error magnitude, at which localization is considered successful. The evaluation considers ϵ values in the range $[0, 5]$, to assess the magnitude of global localization error necessary for the system to detect a localization failure. A second threshold τ, represents the maximum IDF value at which the robot considers to be localized successfully, see Section 4.1. The value of τ is calculated by brute force search, so that recall is maximized and $p = 1$ is achieved, if possible, at each values of ϵ. Evaluating both thresholds together leads to four *localization cases* (see Table 5.1):

True positive (tp) The robot believes to be correctly localized when that is indeed the case.

False positive (fp) The robot believes to be correctly localized when it is actually lost.

True Negative (tn) The robot believes to be lost when that is indeed the case.

False Negative (fn) The robot believes to be lost when it is actually correctly localized.

	$D_{l_j,j} \leq \tau$	$D_{l_j,j} > \tau$
$\lvert e_j \rvert \leq \epsilon$	tp	fn
$\lvert e_j \rvert > \epsilon$	fp	tn

Table 5.1: Localization cases in terms of the localization hypothesis error and thresholds ϵ and τ. Both conditions must evaluate to true for a case to occur.

Precision p and recall r are computed from the frequencies of localization cases (denoted in capitals)

$$p = \frac{TP}{TP + FP}, \tag{5.2}$$

and

$$R = \frac{TP}{TP + FN}. \tag{5.3}$$

5.3 Assessment of the Feasibility of the TVCC

This section proposes a method to evaluate the feasibility of the TVCC, see Equation 4.6 and Equation 4.7. The TVCC would be deemed as feasible if using it produces a considerable increase in failure detection performance, compared to the best performing VCA configuration alone. The evaluation procedure simulates the operation of the TVCC as using the following steps (see Table 5.2 for an example):

1. Select the N best performing VCA configurations in terms of $r@p1$, and sort them in order of descending performance. The selected VCA configurations, their ordering, and the threshold ϵ used to achieve $r@p1$ define a TVCC altogether.
2. The localization cases of a stage are replaced by those of a later stage in cases where the condition $d \leq \epsilon$ is false. Thus fn and tn are replaced tp. fp are not present in the data because the stage thresholds do not allow so. The replacement is performed in a greedy manner, i.e., only until a stage provides a replacement.
3. $r@p1$ is calculated from the resulting localization cases.

j	0	1	2	3
Stage 0	**tp**	fn	tn	**tn**
Stage 1	tn	**tp**	fn	fn
Stage 2	fn	fn	tn	tn
Stage 3	tp	tp	**tp**	tn
Stage 4	tn	tp	tp	fn
Result	tp	tp	tp	tn

Table 5.2: Simulation of a TVCC with five stages. Localization cases are obtained from the best five VCA configurations and are merged, simulating the operation of a TVCC. In each case, the entry that remains in the result (replacement stops) is highlighted using bold font. This example is only for illustration purposes. In the first and fourth cases (j=0, and j=3, respectively) no replacement was performed. In the second case (j=1) the fn of the first stage was replaced by a tp found by stage 1. In the third case (j=2), a tn of the first stage was replaced by a tp of stage 3.

Note that the procedure just described is bound to improve recall, and does not affect precision, since fp cases are not allowed due to the restriction of achieving precision of 1. Thus, all replacements would increase the number of tp and reduce the number of fn. The localization hypothesis used to evaluate the TVCC is found in a similar way. It corresponds to the localization hypothesis of the stage used for replacement. For example, in Table 5.2 l_0 is found by stage 0, l_1 by stage 1, l_2 by stage 3, and l_3 by 0.

5.3.1 Whisker Plots

Some of the results are presented using whisker plots, a common representation for unimodal distributions, see Figure 5.1. There are many whisker plot variants, and to avoid confusion, the variant used in this dissertation is explained in the following. See Frigge et al. (1989) for details.

Before explaining the plot, it is important to recall some basic concepts. Let $X = \{x_i\}_{i=0}^{N-1}$ be a set of N observations. A percentile is a value in the range of observations, below which a determined percentage of the observations fall. For example, the 10th percentile is a value for which 10 % of the observations have a smaller value. A quartile is a related term. The first (Q_1), second (Q_2), and third (Q_3) quartiles correspond to the 25th, 50th, and 75th percentiles, respectively. Conveniently, Q_2 is the same as the median value and is used to represent the mode. Spread is represented using the IQR, defined as $IQR = Q_3 - Q_1$. Therefore, 50% of the observations around the median are contained in the range defined by the IQR, 25% between the median and Q_1, and 25 % above the median and Q_3.

In a whisker plot, Q_1 and Q_3 are represented by a box, whose horizontal edges are located at them, and the median is represented by a horizontal bar inside the box. Besides representing the mode and spread of the distribution, whisker plots also represent the range of observations. The lower limit corresponds to the smallest observation x_i that is greater than $Q_1 - 1.5 \times IQR$. Likewise, the upper limit corresponds to the largest observation that has a value less than $Q_3 + 1.5 \times IQR$. The lower- and upper-limit are connected to the box by vertical bars called whiskers, from which the plots get their name. At the end of the whiskers, there is another set of horizontal bars to denote the range of observations.

Observations beyond those limits are considered outliers and are represented with markers placed at the observation values.

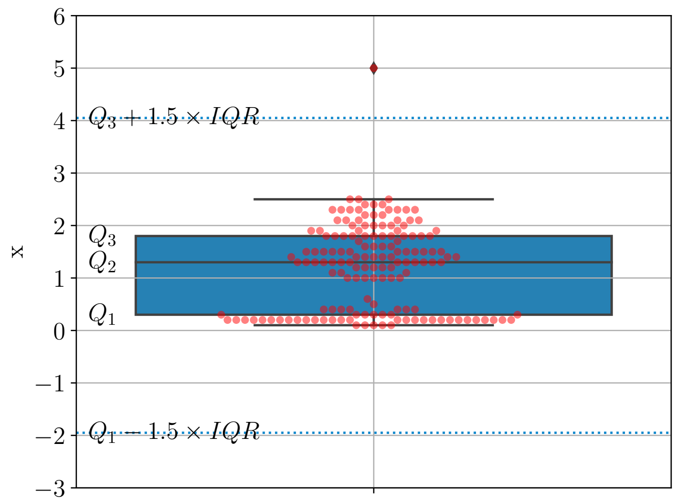


Figure 5.1: Example of a whisker plot. The red dots represent the observations. They are not strictly part of the whisker plot but were overlaid for illustrative purposes. Observations with equal values are grouped horizontally. In this plot, $Q_1 = 0.3$ $Q_2 = 1.34$ $Q_3 = 1.8$, and $IQR = 1.5$. The upper and lower limits at which observations are considered as inliers are displayed using dotted lines. There was a single outlier at 5.

5.4 VCA Offline Analysis Toolbox

The evaluation described in this section was performed using a toolbox developed as part of this investigation. Most of it is implemented using Python 3. For efficiency reasons, the VCA was implemented using a superset of Cython that simplifies defining IDFs and generates the necessary code. OpenMP and Python's multiprocessing module were used as appropriated to provide multithreading capabilities.

The toolbox provides several functionalities. First, it provides abstractions to access visual memories and their snapshots. Visual memories are stored in an ad-hoc filesystem hierarchy that uses PNG format for omnidirectional images and CSV files for all other types of data, e.g., GPS and odometry.

The second functionality allows defining representation pipelines in terms of simple building elements in a similar way as described in Section 3.2.4. Each building block takes a snapshot as input and outputs a transformed version of it, i.e., one with modified, added, or removed information.

The third functionality consists of aligning visual memories (using one as reference and another one as current-views as described at the beginning of the chapter). To that end, the toolbox allows defining experiments in the form of combinations of VCA configurations and visual memory pairs. The results of those experiments (matrices D and S) are stored in HDF format and can be accessed using custom abstractions specially crafted for that purpose. Besides providing access to the evaluation results, the abstractions can recreate the representation pipelines used, regenerate the panoramic images used during the experiment, and produce intermediate representations of the results that are useful for their evaluation.

5.4.1 Image Preprocessing Pipeline and Representation Elements

The toolbox enables defining representation pipelines by chaining simple elements together. Each of them receives as input a snapshot, which contains an image, a calibration model, and other types of data useful for analysis. Produces a modified version of the snapshot and passes it as input to the next element until the last one is reached. The basic representation used in this dissertation is simply called raw and consists of the steps listed below.

RGB image Returns an RGB omnidirectional image by loading it from secondary storage.

Grayscale converter Takes an RGB image and converts it to grayscale.

VFOV selector (optional) Selects a section of the VFOV based on two parameters that determine the lower and upper limits. This change is only reflected in the calibration model, which would later affect unwrapping.

Scaramuzza model (optional) Changes to calibration model to the Scaramuzza model described in Section 4.5.1. This element has to be called before the unwrapper so that panoramic images are generated with accurate elevations and azimuths.

Unwrapper Unwraps an omnidirectional image into a panoramic image using the calibration model. All snapshots use the naive model by default.

Angular resolution downsampler Downsamples the panoramic image so that it complies with desired azimuth and elevation resolutions.

More elaborated representations are created based on the raw one. The following additional representation elements are currently implemented.

Sobel ($\nabla_{dx,dy,k}$) Transforms a grayscale image into an image of gradients using the Sobel operator. dx and dy defined the order of the partial derivatives, and k the size if the kernel used for convolution.

Zero mean ($\mathcal{N}(0, \sigma)$) Returns an image with grayscale values centered at zero. That image is obtained by subtracting the mean grayscale value. Provides global illumination invariance.

Local zero mean ($\mathcal{N}_k(0, \sigma)$) Returns an image with grayscale values locally centered at zero. That image is obtained by subtracting the mean grayscale value calculated around each pixel. Provides local illumination invariance.

LBP (See Section 4.3.1 for notation) Transforms a gray level image into a panorama of textons (LBP image). It takes three parameters. The number of points to use P, the radius R, and the variant of LBP to use.

PCHL ($H(n, s)$) Returns a PCHL, see Section 4.3.2. The LBP element has to be calculated before this one. n defines the number of vertical regions, and s their width. The details of the underlying LBP representation are provided separately.

5.5 Visual Memories used for Evaluation

5.5.1 Synthetic Visual Memory Generation

Some of the experiments presented in this dissertation were performed on synthetic data because it allows control over different aspects known to affect the performance of the visual compass algorithm. These include: the relative placement of the camera, the illumination conditions, the presence of dynamic objects, and the 3D structure of the environment (Zeil et al., 2003; Stürzl and Zeil, 2007; Möller et al., 2014; Ardin et al., 2015; Raderschall et al., 2016; Freas et al., 2018).

Synthetic data is created using patched version of Blender[1] (refered from now on simply as blender), that generates images that mimic those coming from real omnidirectional cameras (Zichao Zhang et al., 2016). Blender was built from source and patched inside a Docker container [2] that mimics the execution environment used when the patch was released. Omnidirectional images are rendered using the Scaramuzza model described in Section 4.5.1.

The rendering of images was performed using a rendering toolbox developed based on the patched blender version, using blender's Python 3 scripting capabilities. The rendering toolbox creates visual memories using the same format understood by the VCA analysis toolbox presented in Section 5.4.

The rendering of omnidirectional is based on the Scaramuzza model. The parameters for the two catadioptric cameras with hyperbolic mirrors used are provided in Table 5.3. Examples of the synthetic images are provided in Section 5.5.2.

The depth images contain points at infinity at the leaves of the trees, which are filtered as follows. First, a binary image called "mask" is generated by thresholding the depth image. "On" pixels correspond to infinity, and "off" pixels are closer than infinity. Then, the opening morphological operator with a structuring element of size 3×3 is applied to the mask image, and the result is stored in another image called "filtered mask". The filtering turns off the pixels of small "on" blobs. Then another binary image called "mask difference" is generated. The "on" pixels in that image are those that are different between the mask image and the filtered mask. Finally, a new corrected depth image is generated. It is identical to the original depth image with one difference. The depth values of the small blobs at infinity were replaced by the median of valid depths (not at infinity) in their surroundings, see Figure 5.2.

[1]blender.org
[2]docker.com

	Zhang	Zivkovic
a_0	-70.25584688010954	-1.314797×10^2
a_1	0	0
a_2	0.0005433216164761568	1.848991×10^3
a_3	$2.102936744599082 \times 10^{-5}$	$2.251206 \times 10{-}7$
a_4	$-8.54310806364036 \times 10^{-9}$	-7.343233×10^{-1}
c	1	0.999845
d	0	-1.314797×10^2
e	0	0.000008
x_c	0	0
y_c	0	0
W	640	1024
H	480	768

Table 5.3: Scaramuzza model parameters for two catadioptric cameras with hyperbolic mirror. The parameters of the Zhang camera were obtained directly from the source code of the blender patch (Zichao Zhang et al., 2016). The parameters for the Zivkovic camera (Zivkovic and Booij, 2005) were obtained from one of the examples provided in the omnidirectional calibration toolbox (Scaramuzza et al., 2006b).

5.5.2 The Urban Canyon Visual Memories

The Urban Cayon visual memories were created using a 3D model provided by Zichao Zhang et al. (2016), see Figure 5.3, and the procedure explained in the previous section. All visual memories had exactly 657 snapshots. One route was used as reference, and other routes parallel to the reference were used as current-views. The current-view routes were rendered at ±0.2, ±0.4, and ±0.8 meters w.r.t. the reference. See Figure 5.4.

5.5.3 Real-Wold Visual Memories Captured with the ROSStuhl

These visual memories were captured using the Kodak SP360 omnidirectional camera using the GUI described in Section 3.2.2 and stored as described in Section 3.2.1. Then, the rosbags of the snapshots storage were exported to the format understood by the VCA evaluation toolbox before analysis. In all cases, the VFOV was cropped to cover 45 degrees of elevation. The omnidirectional camera was pointed upwards to make the illumination conditions more variable, especially important because, in some cases, the timespan between image acquisition was limited.

Each route was demonstrated twice to have a reference and current-view. Due to limitations in the precision at which the human driver can retrace routes, off-route translational error is likely present. The author considers it safe to assume that off-route error remains well under 20 to 30 cm at all times due to the size of the vehicle, the navigable areas, and because the driver used environmental features (like distance to the edge of sidewalks) as visual references while driving. The number of snapshots has slight discrepancies between reference and current-views because the wheel-odometry, used to select snapshots, provides only 2D motion data and is error-prone on long distances.

5.5.4 The Steglitz1 and Steglitz2 Visual Memories

These visual memories were captured in a busy urban area in the Steglitz district of Berlin. The timespan between acquisition was of two hours, which allowed for noticeable differences in the configuration of the sun, shadows, and clouds. The scenery consists mostly of tall buildings and sporadic vegetation, pedestrians, and cars, which provide short-term appearance changes. See Figure 5.5 for examples. The route started at coordinates 52°27′24.68″N 13°19′13.16E and covered an approximate travel distance of $\approx$ 730 meters. Steglitz1 and Steglitz2 visual memories contain 1444 and 1437 snapshots, respectively. See Figure 5.5.

5.5.5 GardenOvercast and GardenSunny Visual Memories

These visual memories were acquired at the Botanical Garden of the Freie Universität Berlin. The scenery consists mostly of vegetation and rustic roads. Dynamic objects, like cars and pedestrians, were practically not present. However, these visual memories are challenging for different reasons than the Steglitz ones. The first reason is that the timespan between captures is two months, and the weather was very different. Thus, drastic appearance changes are noticeable, e.g., the shape of trees and the color and texture of the sky. Moreover, the route passes through 3 planes that have places that look practically identical (disregarding orientation). The route has a U-shape, and places located at one half of the U look very similar to places at the corresponding position of the other half. Finally, parts of the road were covered on cobblestone, and the wheelchair had to cross two shallow irrigation canals, which induced different degrees of non-planar motion. GardenOvercast has 298 snapshots, and GardenSunny has 304 snapshots. The start of the route is at coordinates 52°27′20.60″N 18°19′21.52E. See Figure 5.6.

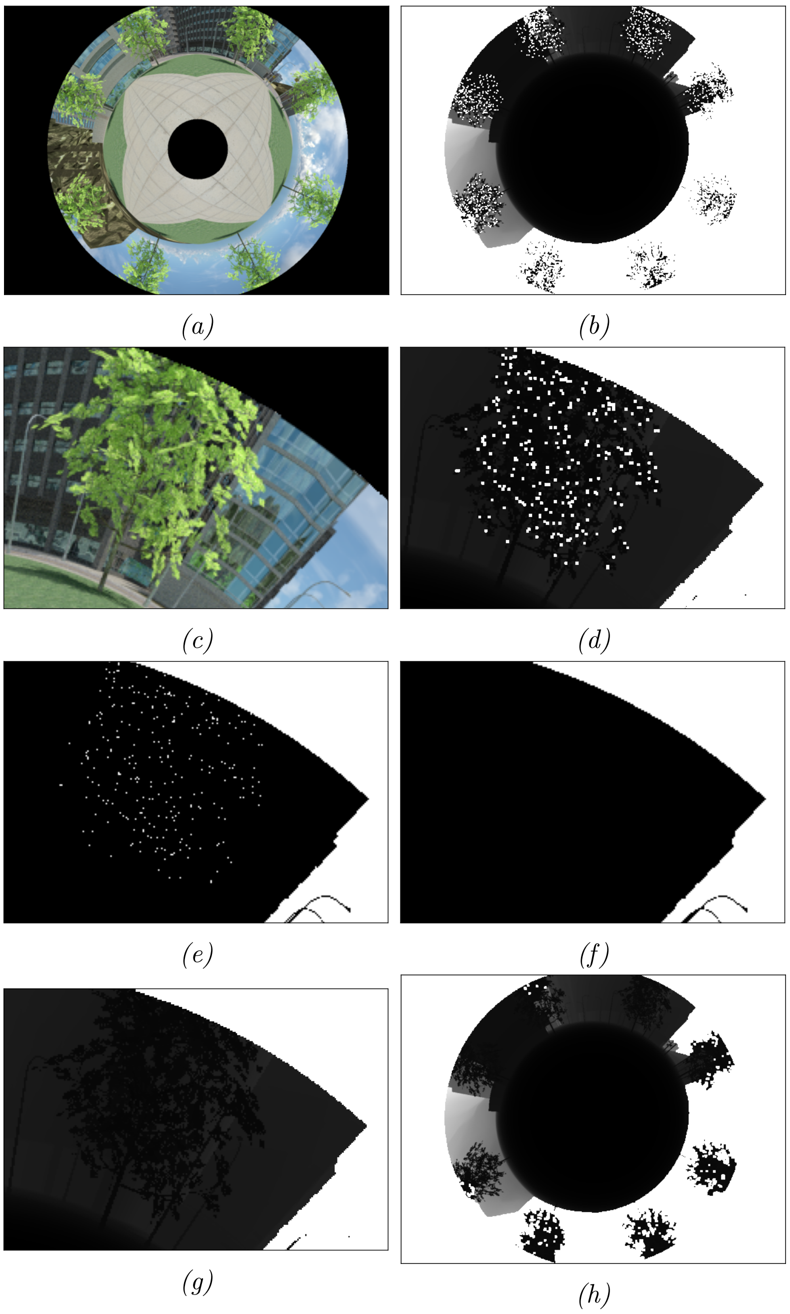


Figure 5.2: Depth image preprocessing. RGB images in this figure are only presented for the reader and are not involved in processing. In the case of depth images, darker pixels are closer than lighter ones. 5.2a Omnidirectional RGB image. 5.2b Omnidirectional depth image. 5.2c and 5.2d depict a zoomed part of the RGB and depth image that serves as an example, respectively. Note the presence of small blobs of depths at infinity. 5.2e Binary mask, "on" pixels indicate values at infinity, and "off" pixels indicate values closer than infinity. 5.2f Binary mask filtered using a morphological operator to remove the small blobs at infinity. 5.2g depth image with the values at infinity replaced by the median depth (not considering infinity values) of the surrounding pixels. 5.2h resulting omnidirectional depth image.

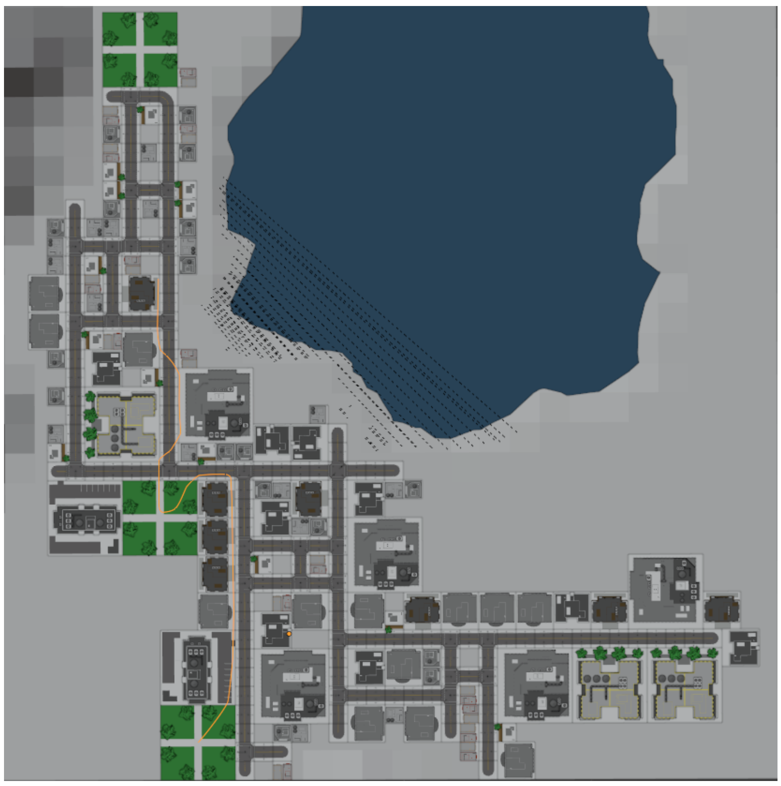


Figure 5.3: The Urban Canyon 3D model used to generate synthetic data. The scenery is that of a small city surrounded by mountains. The sky is rendered using a texture of a partially cloudy sky. The model was provided by Zichao Zhang et al. (2016).

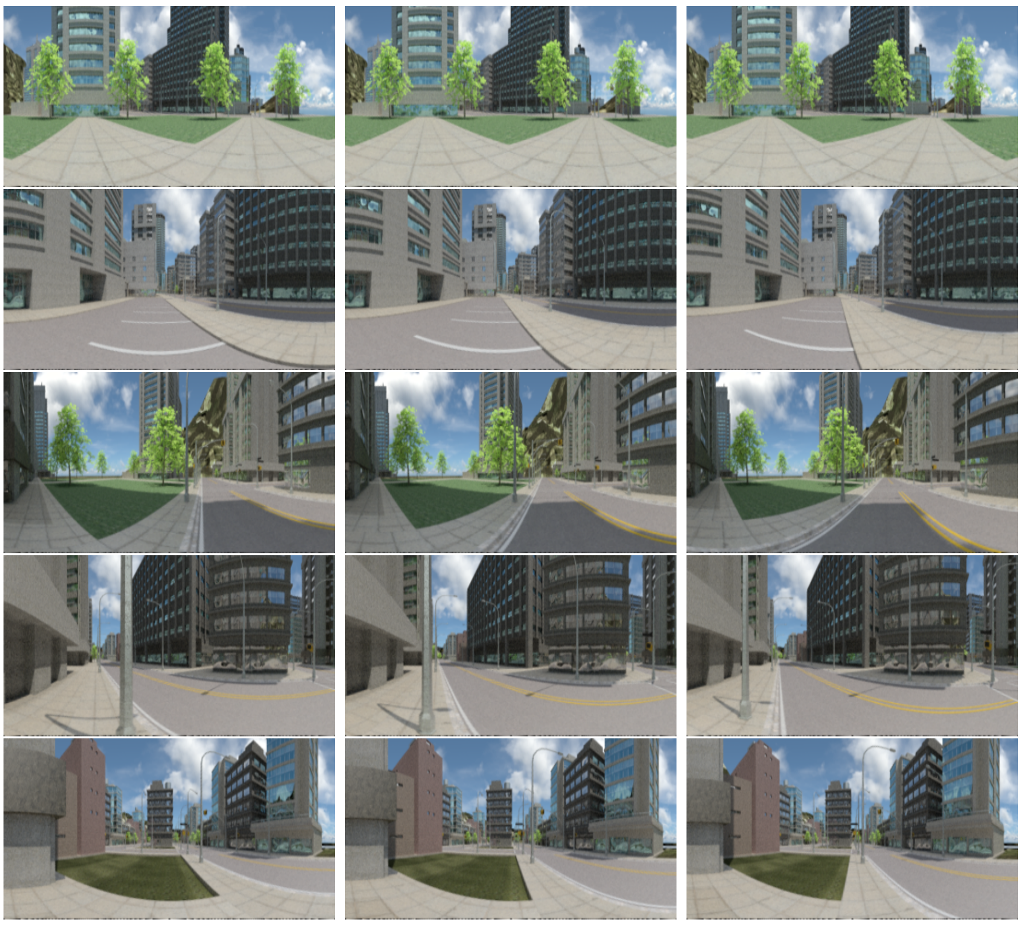
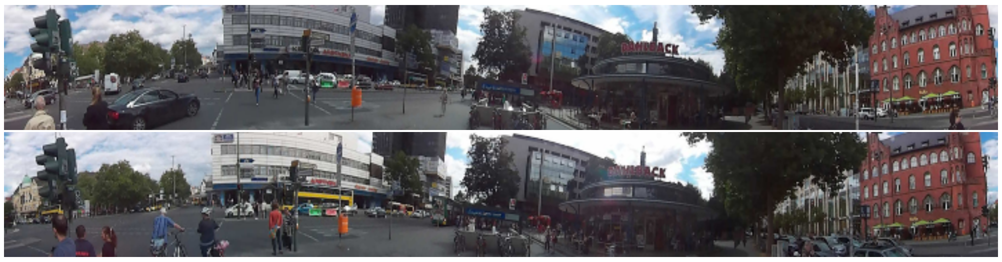
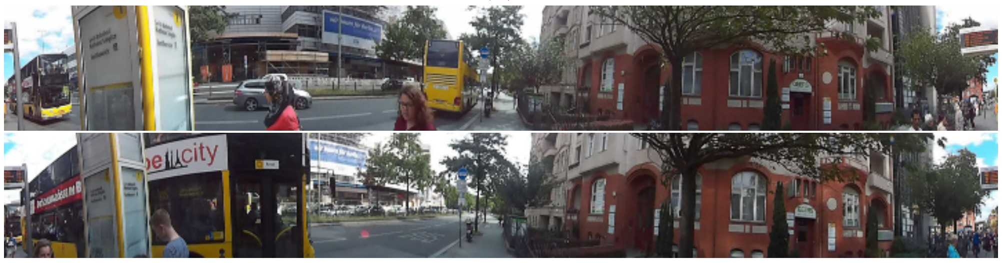


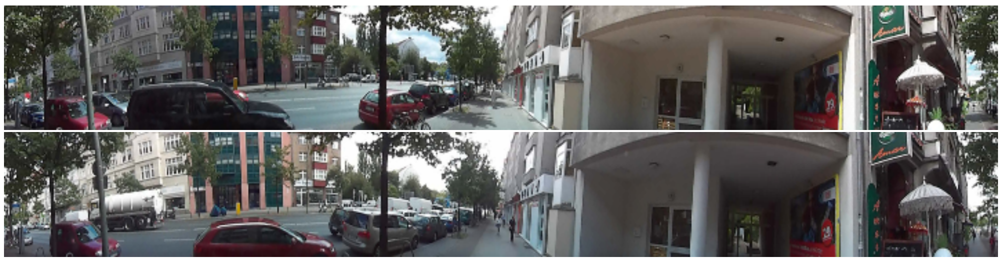
Figure 5.4: Examples snapshots at corresponding locations for the UrbanCanyon visual memories.The center colum shows the reference visual memory. The left and right columns show snapshots acquired at -0.8 and 0.8 meters w.r.t. the reference, respectively. The first and last rows show the snapshots at the beginning and end of the route, respectively. The images were cropped horizontally.



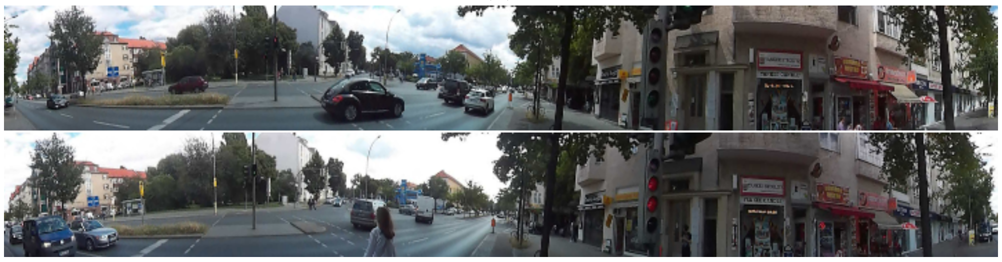
(a)



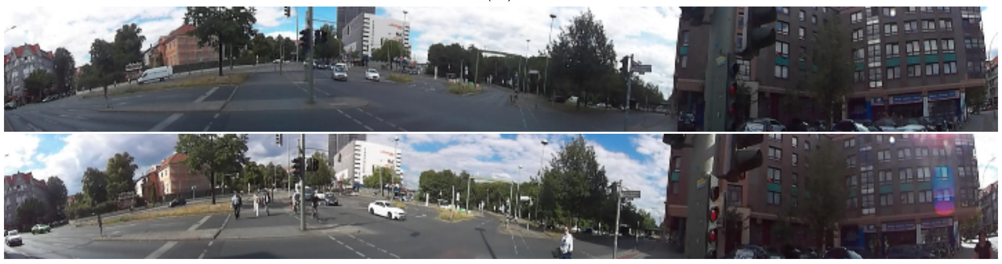
(b)



(c)

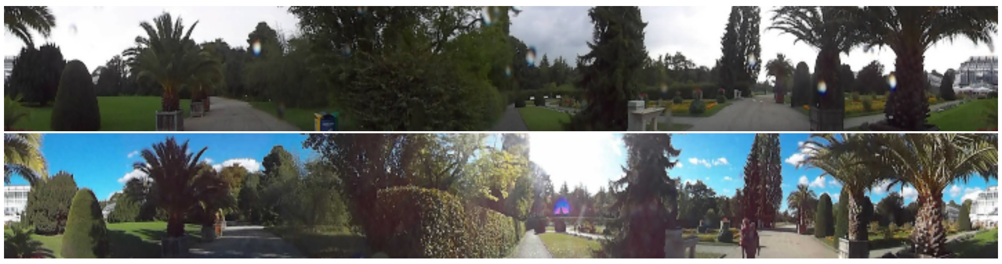


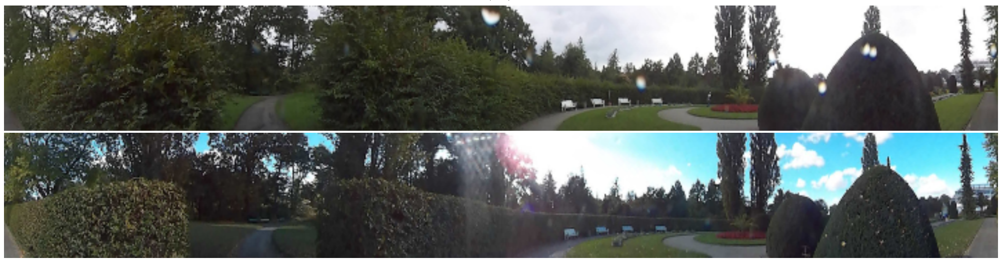
(d)

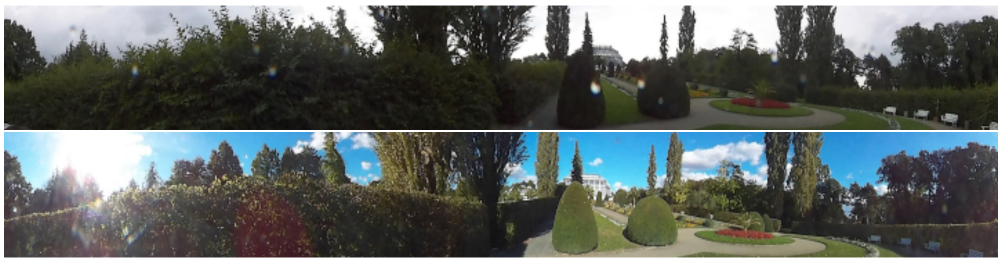


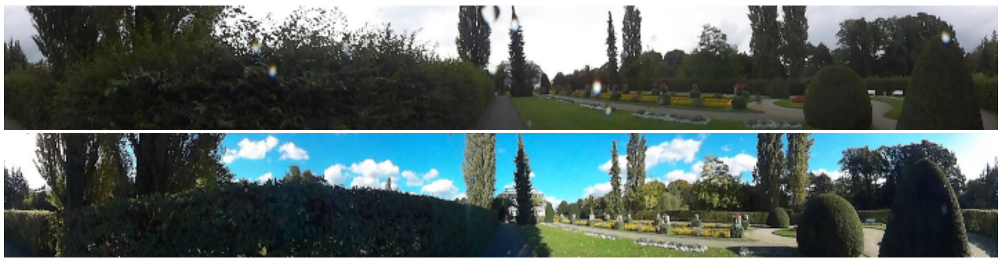
(e)

Figure 5.5: Example snapshots at corresponding locations. Steglitz1 on the left and Steglitz2 on the right. The first and last rows show the snapshot at the beginning and end of the route, respectively.


(a)


(b)


(c)


(d)

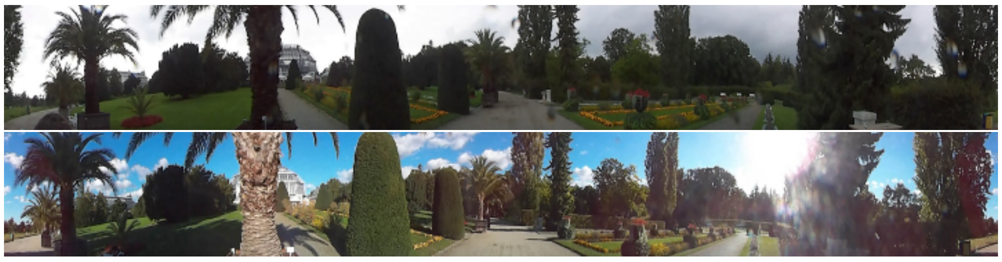

(e)

Figure 5.6: Examples snapshots at corresponding locations. OvercastGarden on the left, and SunnyGarden on the right. The first and last rows show the snapshot at the beginning and end of the route, respectively.

5.6 VCA Configurations Evaluated

The experimental results evaluated the performance of different VCA configurations. Configurations based on the panorama of texton use the PLD IDF, and the remaining ones use SAD. Results are presented for two angular resolutions: 5.0 deg/px and 2.5 deg/px.

The $\mathcal{N}_k(0,\sigma)$ representations are generated for $k =\in \{3,4,..,9\}$. The panorama of textons are parameterized using all combinations of a set of P values, and a set of R values. P take on values from 2 to 9 in increments of 1, and R takes on values from 1 to 3 in increments of 0.5. Note that the same set R is used irregardless of the angular resolution. Therefore, the FOV at which textons are sampled is doubled in the 5.0 deg/px w.r.t the 2.5 deg/px configurations. The Sobel representations are evaluated only using vertical edges ($dx = 1$) and for varying kernel sizes between 3 and 9. Regarding the PCHL representations, they are evaluated for $n = 1$ and s values $1, 3, 5$. The P and R and LBP variant used to extract the underlying panorama of textons are selected depending on the situation.

5.7 Experimental Results

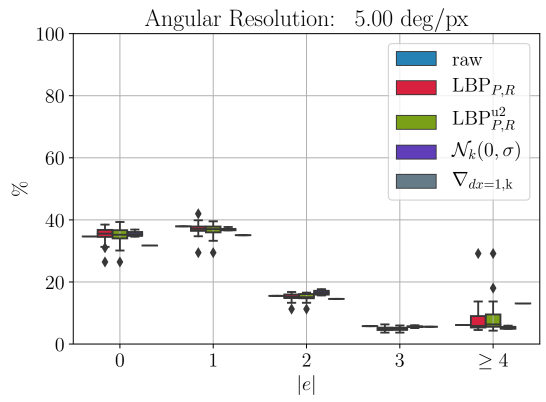
5.7.1 Panorama of Textons compared to Intensity Representations

Global Localization and Localization Classification Performance using Single Configurations

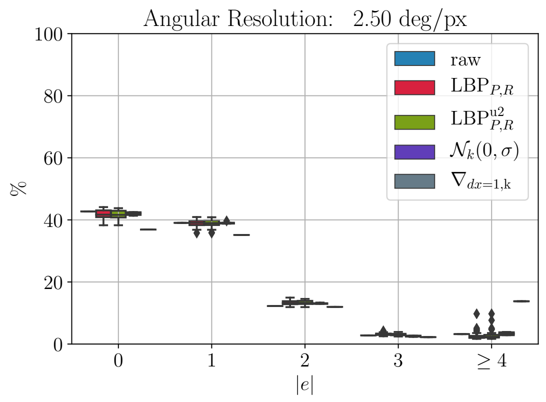
Steglitz Visual Memories

The top row of Figure 5.7 shows the global localization performance in terms of the percentage of snapshots that reached a certain localization error magnitude $|e|$. Differences for the different values of $|e|$ are mostly negligible regardless of the representation. $|e| = 0 \approx 40$ % of the times, and $|e| \leq 1$, $\approx 80\%$ of the times. Differences due to resolution were of approximately 5 to 10% being the representations based on 2.5 deg/px angular resolution the top performers.

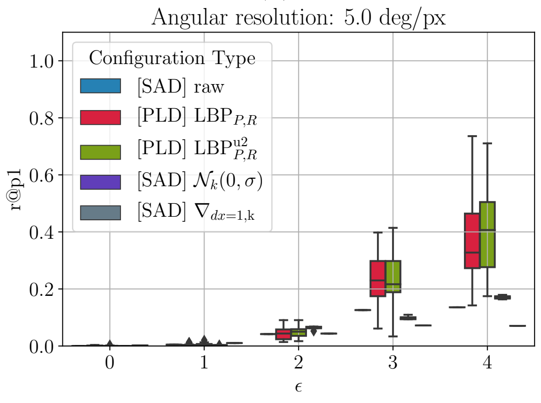
Regarding the intensity-based representations, there are some interesting findings. The IQR of $\nabla_{dx=1,k}$configurations is so small that their whisker plots appear as a line, which indicates different kernel size k did not have an effect on performance. Notably, the raw configuration performed similar to the others, despite its lack of illumination invariance. For 5.0 deg/px and $|e| = 0$, it even performs better than the panorama of textons at the 75th percentile, and better than $\mathcal{N}_k(0,\sigma)$ and $\nabla_{dx=1,k}$. This indicates that illumination differences in the visual memories were not responsible for performance differences.. The panorama of textons representations were the top performers. However, the difference respect to other representations is also between 5 to 10 %. Figure 5.8 shows the performance of the LBP parameterizations used to create panoramas of textons.



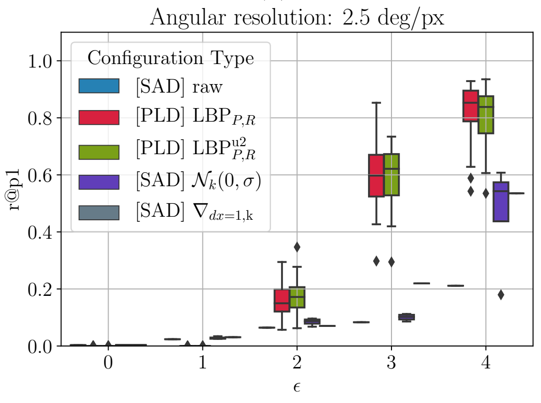
(a)



(b)



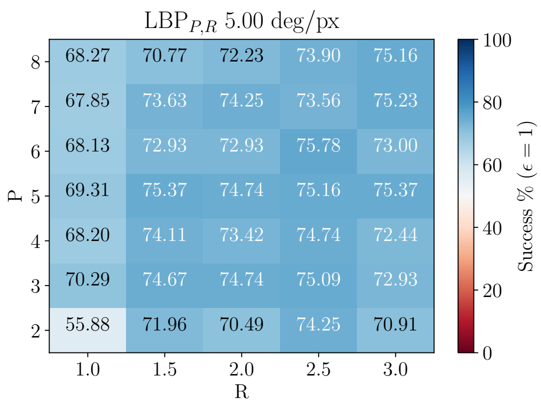
(c)



(d)

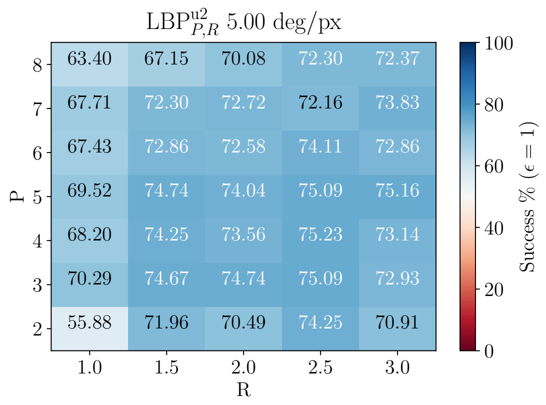
Figure 5.7: Performance summary on the Stegtliz visual memories. Top row, global localization: 5.7a and 5.7b configurations with 5.0 *deg/px and* 2.5 *deg/px angular resolution, respectively. Bottom row, localization classification performance: 5.7c and 5.7d, configurations with* 5.0 *deg/px and* 2.5 *deg/px angular resolution, respectively.*

In contrast, localization classification performance in terms of r@p1 varied greatly depending on the configuration used, see bottom row of Figure 5.7. In this case, the intensity-based representations performed poorly in general for all ϵ values. The best cases are found for $\epsilon = 4$ using 2.5 deg/px angular resolution: $\mathcal{N}_k(0,\sigma)$ reached ≈ 0.57 at its 75th percentile, whereas $\nabla_{dx=1,k}$ reached ≈ 0.55 approximately. In comparison, the panorama of textons performed better, especially at 2.5 deg/px of angular resolution and for $\epsilon \geq 3$. However, the distributions have long tails and outliers towards the lower range, indicating that several of the configurations performed poorly. See Figures 5.9 and 5.10 for details on the performance of different parameterizations.



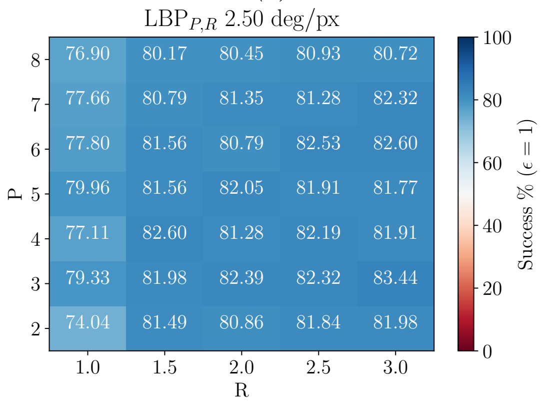
P \ R	1.0	1.5	2.0	2.5	3.0
8	68.27	70.77	72.23	73.90	75.16
7	67.85	73.63	74.25	73.56	75.23
6	68.13	72.93	72.93	75.78	73.00
5	69.31	75.37	74.74	75.16	75.37
4	68.20	74.11	73.42	74.74	72.44
3	70.29	74.67	74.74	75.09	72.93
2	55.88	71.96	70.49	74.25	70.91

(a)



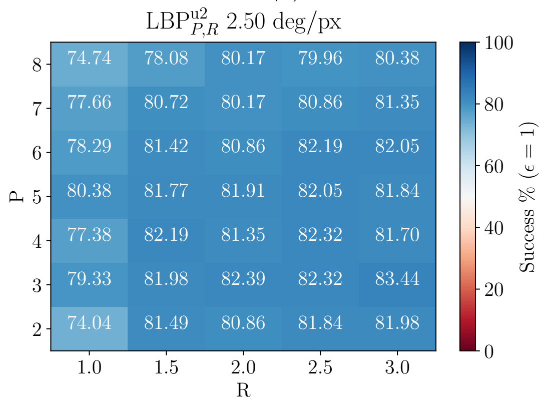
P \ R	1.0	1.5	2.0	2.5	3.0
8	63.40	67.15	70.08	72.30	72.37
7	67.71	72.30	72.72	72.16	73.83
6	67.43	72.86	72.58	74.11	72.86
5	69.52	74.74	74.04	75.09	75.16
4	68.20	74.25	73.56	75.23	73.14
3	70.29	74.67	74.74	75.09	72.93
2	55.88	71.96	70.49	74.25	70.91

(b)



P \ R	1.0	1.5	2.0	2.5	3.0
8	76.90	80.17	80.45	80.93	80.72
7	77.66	80.79	81.35	81.28	82.32
6	77.80	81.56	80.79	82.53	82.60
5	79.96	81.56	82.05	81.91	81.77
4	77.11	82.60	81.28	82.19	81.91
3	79.33	81.98	82.39	82.32	83.44
2	74.04	81.49	80.86	81.84	81.98

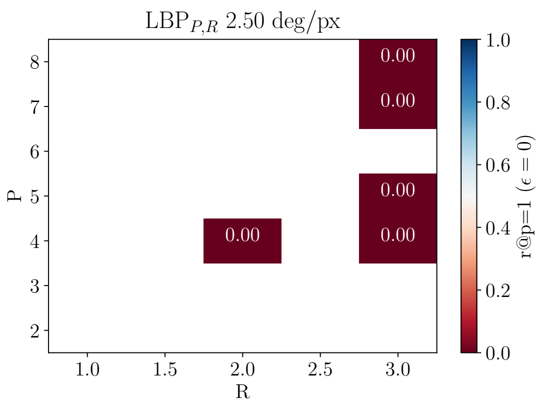
(c)



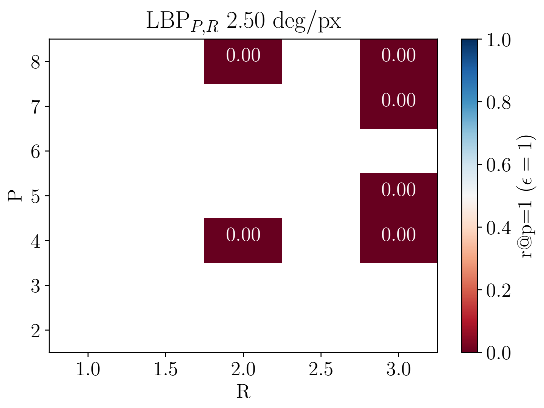
P \ R	1.0	1.5	2.0	2.5	3.0
8	74.74	78.08	80.17	79.96	80.38
7	77.66	80.72	80.17	80.86	81.35
6	78.29	81.42	80.86	82.19	82.05
5	80.38	81.77	81.91	82.05	81.84
4	77.38	82.19	81.35	82.32	81.70
3	79.33	81.98	82.39	82.32	83.44
2	74.04	81.49	80.86	81.84	81.98

(d)

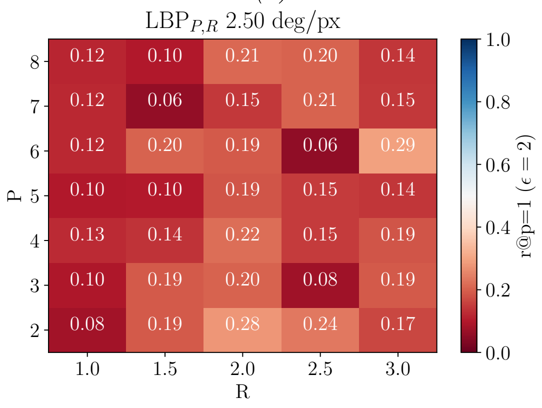
Figure 5.8: Localization error for different parameterizations of the LBP operator on the Steglitz visual memories. Using a resolution of 2.5 *deg/px increased performance by* ≈ 10 % *over configurations with resolution of* 5.0 *deg/px. This shows the LBP operator is robust to parameterization for global localization purposes.*



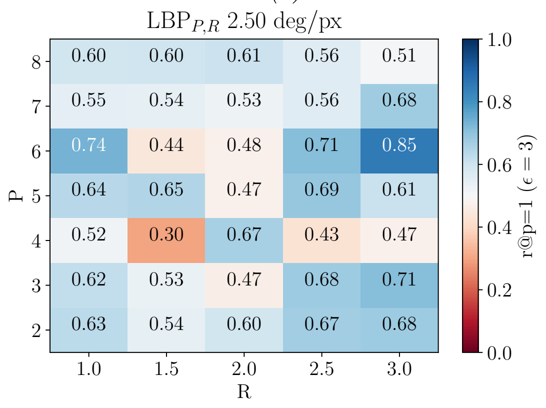
(a)



(b)

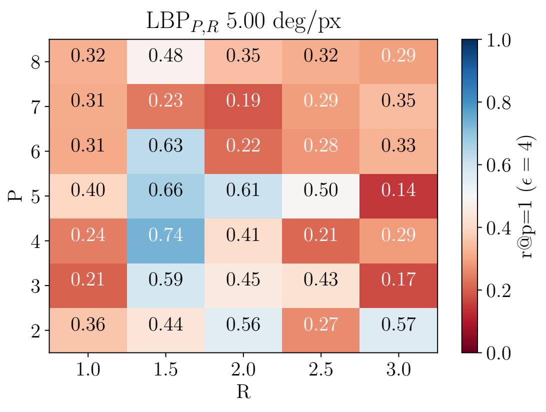


(c)

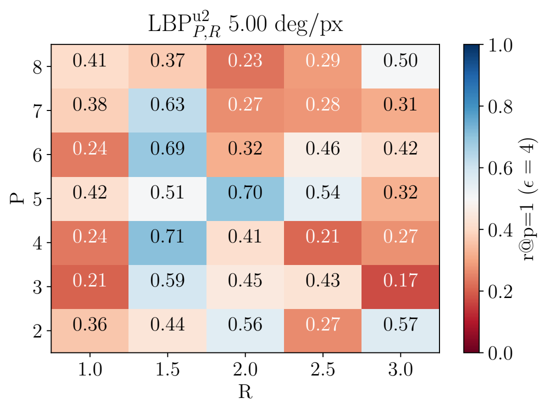


(d)

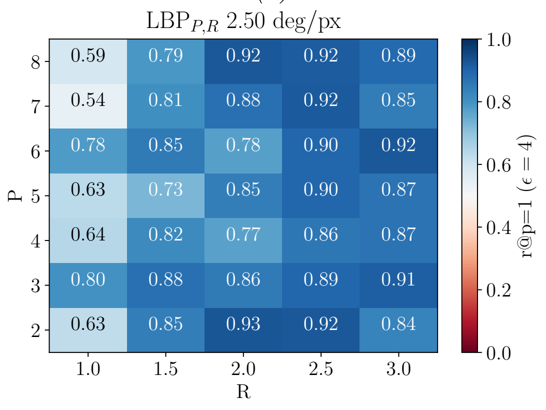
Figure 5.9: Localization classification performance in terms of r@p1. Poor performance cases, found for $\epsilon \leq 3$. For $\epsilon < 2$ only 5 configurations found a suitable threshold ϵ to achieve precision of one. However, the recall level is so low that it rounds to zero with two decimals of precision. For $\epsilon = 2$ all configurations reached precision 1 but the recall is at most 0.15. For $\epsilon = 3$ most of the configurations had r@p1 above 0.5, and one of them reached up to 0.85.



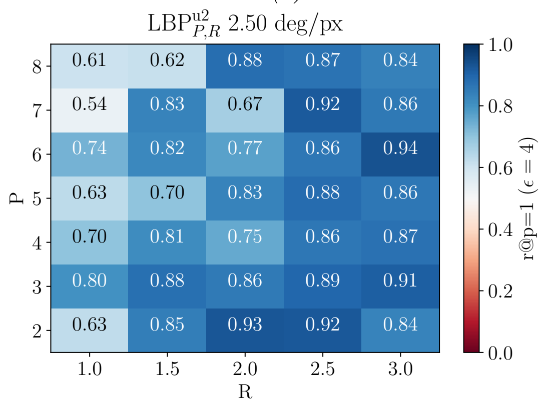
(a)



(b)



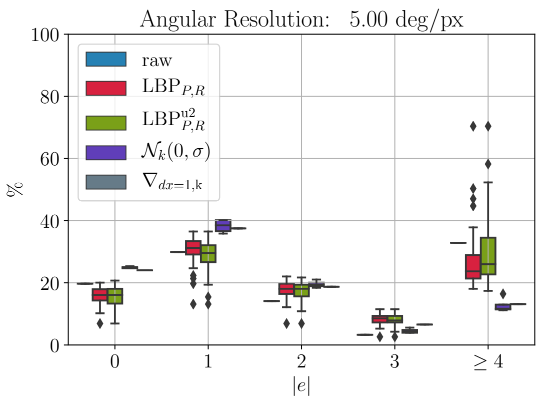
(c)



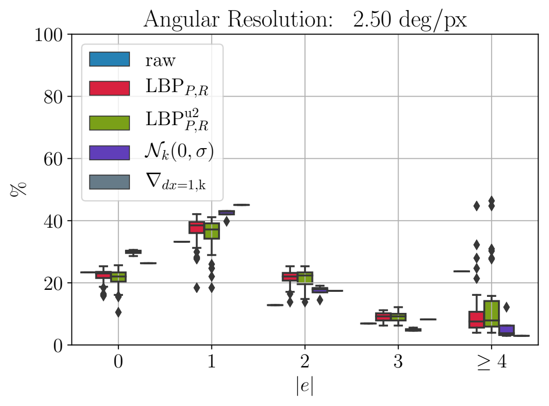
(d)

Figure 5.10: Localization classification performance in terms of r@p1 for $\epsilon = 4$. Higher resolution configurations perform notably better as they reach r@p1 above 0.9. Configurations of corresponding P and R have similar performances, suggesting that the selection of those parameters is more important than the selection of an LBP variant. Unfortunately, there is not a clear pattern on how P and R affect performance. For example LBP$_{6,2.0}$ has r@p1 $= 0.72$, the surrounding configurations are all above 0.8, but one of them LBP$_{7,2.5}$ has r@p1 $= 0.92$.

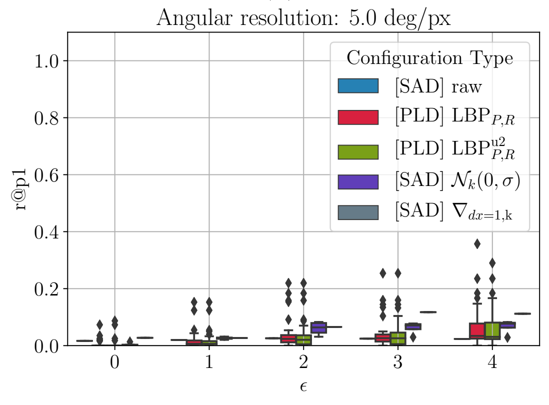
Garden Visual Memories



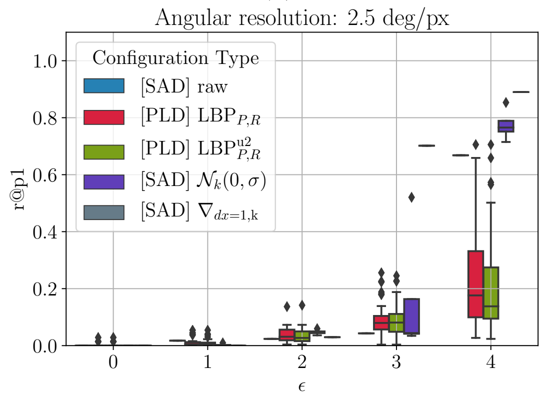
(a)



(b)



(c)



(d)

Figure 5.11: Performance summary on the Garden visual memories. Top row, global localization performance: 5.11a and 5.11b, configurations with 2.5 *deg/px and* 5.0 *deg/px angular resolution, respectively. Bottom row, localization classification performance: 5.11c and 5.11d configurations with* 2.5 *deg/px and* 5.0 *deg/px angular resolution, respectively.*

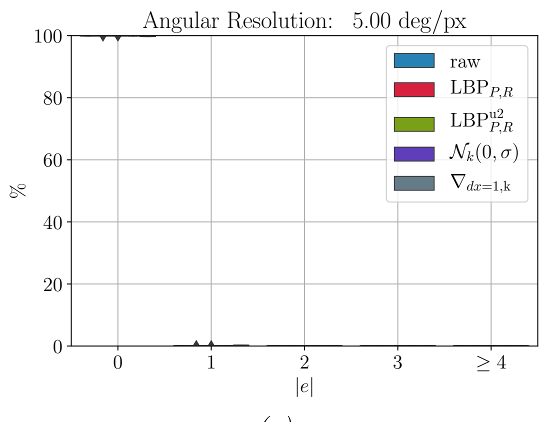
Figure 5.11 shows the whisker plots that summarize performance in these visual memories. The top row shows global localization performance, and the bottom row shows localization classification performance. Overall, performance was lower in these visual memories than in the Steglitz ones. The localization hypothesis had an error $|e| = 0$ only roughly 20% of the time, and $|e| \leq 1$ only $\approx 60\%$ of the time. The intensity-based configurations with illumination invariance, i.e., $\nabla_{dx=1,k}$ and $\mathcal{N}_k(0,\sigma)$ were the best overall performers. The raw representation performed comparably well for $|e| = 0$. However, it was the worst performer because a large percentage of cases reached $|e| \geq 4$. The panorama of textons performed just slightly worse than the $\nabla_{dx=1,k}$ and $\mathcal{N}_k(0,\sigma)$ for $|e| \leq 3$. However, they had a large percentage of cases with $|e| \geq 4$ when using 5.0 deg/px angular resolution. The percentage of those cases was lower with 2.5 deg/px angular resolution, but still, illumination-invariant intensity-based representations performed better.

Localization classification performed poorly (below 30 %) for all ϵ values and both angular resolutions evaluated. The only exceptions were found for $\epsilon = 4$ and 2.5 deg/px angular resolution, where $\nabla_{dx=1,k}$ reached as much as 0.9 of r@p1, followed by $\mathcal{N}_k(0, \sigma)$ that reaches 0.8 at the 75th percentile, and raw with ≈ 0.7. The panorama of textons were the worst performers and exhibited a large variability in the results, both in range and also in IQR. In this case, they performed worse than the raw representations, indicating that illumination differences were not responsible for the differences in localization classification.

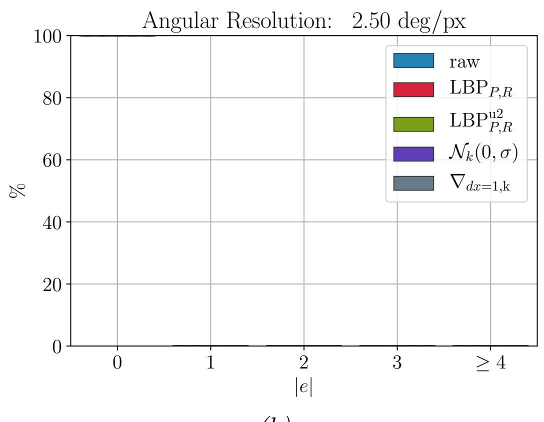
Robustness to off-route Translation using Synthetic Visual Memories

In this part of the study, the different configurations are compared in terms of their robustness to off-route translation. Figures 5.12 and 5.13 summarize performance off-route error of -0.2 and 0.2 m, respectively. At this magnitude of off-route translation, all the representations perform perfectly: global localization achieve $|e| = 0$ 100 % of the time, and localization classification always reached r@p1 = 1 for $\epsilon = 0$.

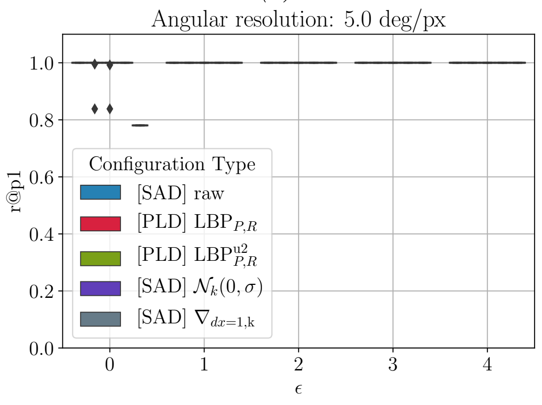
Figures 5.14 and 5.15 summarize performance under off-route translational error of -0.8 and 0.8 m, respectively. In contrast to the results at ± 0.2 m, with ± 0.8 off-route translation, the results degrade drastically. Regarding global localization, the localization error $|e| \leq 1$ almost 100 % of the time, and differences due to resolution are approximately 10 %. Regarding localization classification, the representations operating at 5.0 deg/px performed better. The raw representation and some parameterizations of $\mathcal{N}_k(0, \sigma)$ reached r@p1 at $\epsilon = 2$. The panorama of textons performed better than the other representations for $\epsilon = 2$. However, for $\epsilon \geq 2$, they performed comparably well to the other representations only beyond the 75th percentile. This indicates that the panorama of textons required a precise parameterization to perform well.



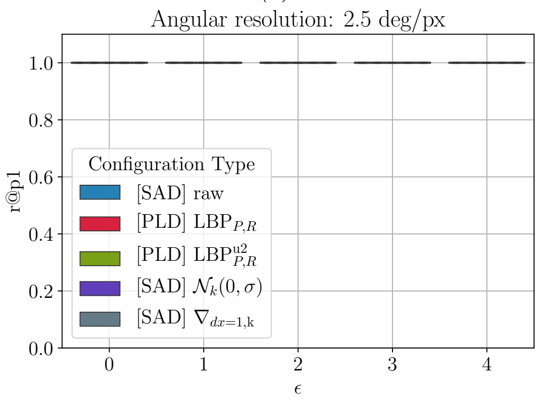
(a)



(b)

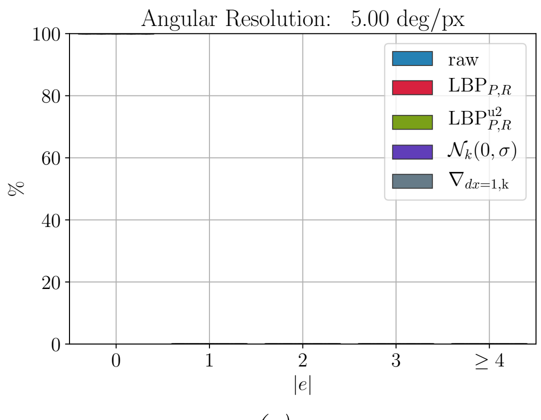


(c)

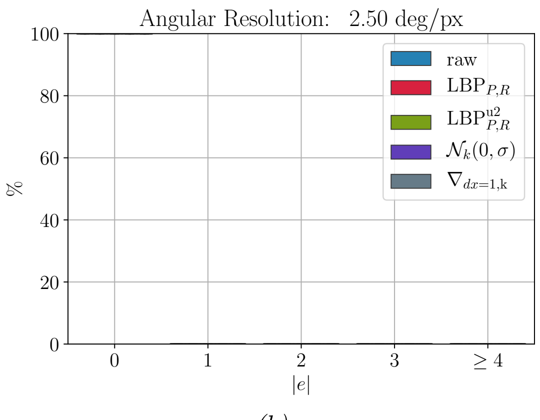


(d)

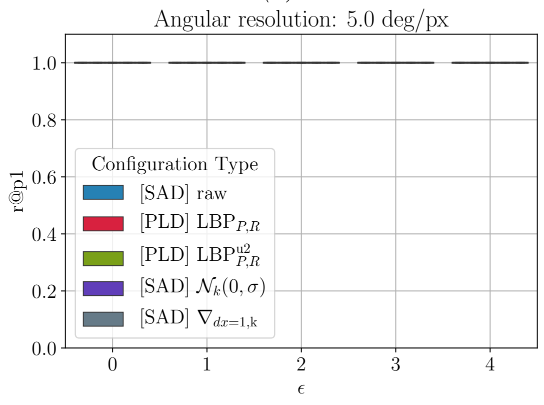
Figure 5.12: Performance summary on UrbanCanyon visual memories for -0.2 m off-route translation. Global localization performance: configurations 5.12a at 2.5 *deg/px angular resolution and 5.12b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.12c at* 2.5 *deg/px angular resolution and 5.12d* 5.0 *deg/px at angular resolution.*



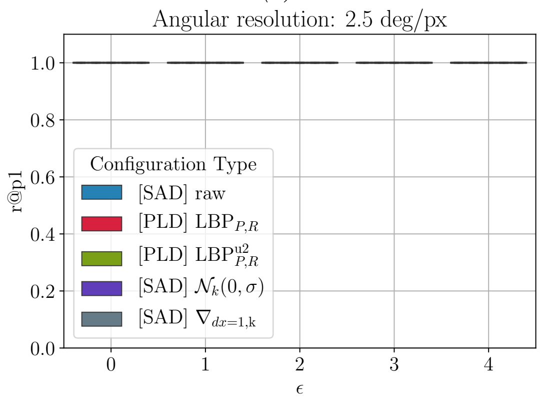
(a)



(b)

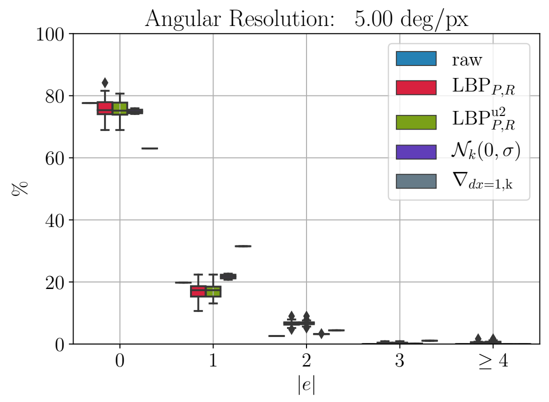


(c)

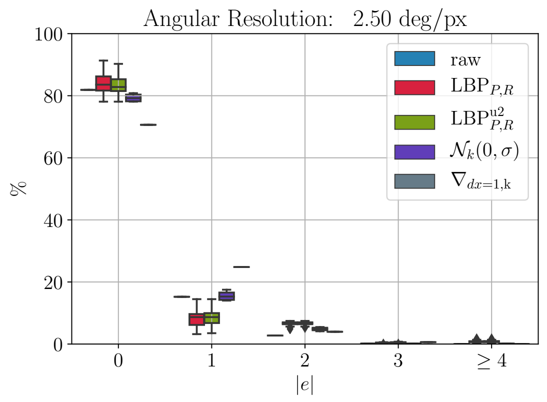


(d)

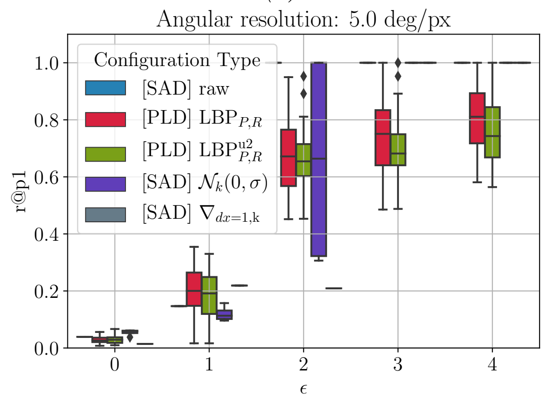
Figure 5.13: Performance summary on UrbanCanyon visual memories for 0.2 m off-route translation. Global localization performance: configurations 5.13a at 2.5 *deg/px angular resolution and 5.13b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.13c at* 2.5 *deg/px angular resolution and 5.13d* 5.0 *deg/px at angular resolution.*



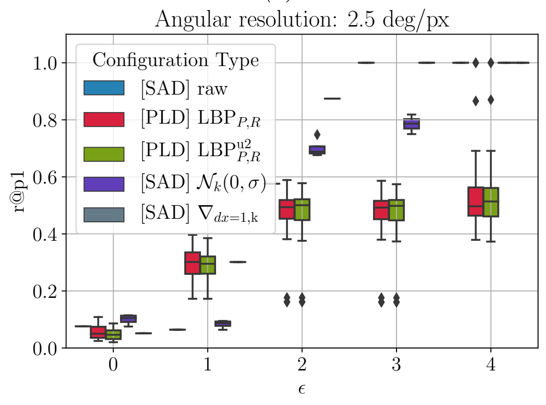
(a)



(b)

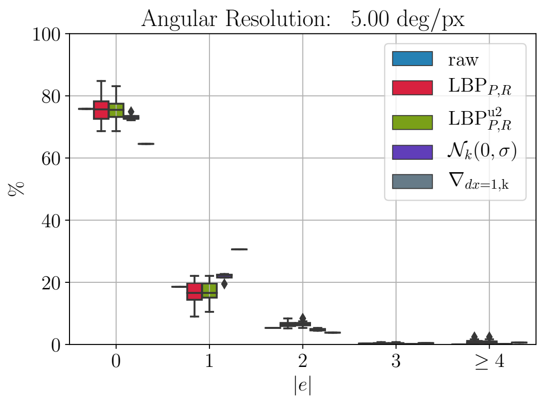


(c)

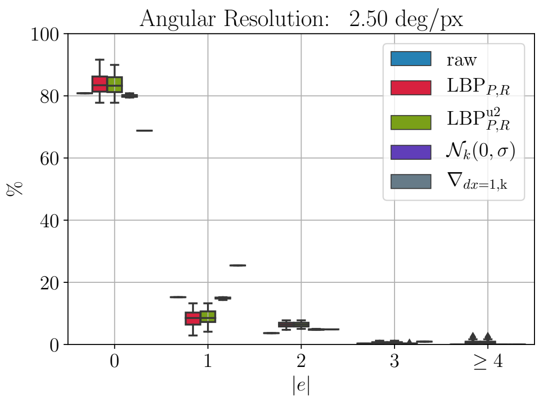


(d)

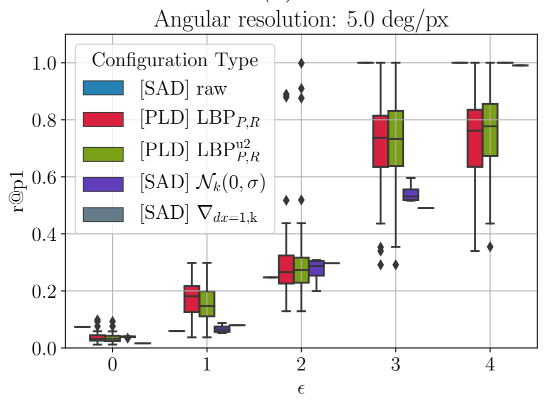
Figure 5.14: Performance summary on UrbanCanyon visual memories for -0.8 m off-route translation. Global localization performance: configurations 5.14a at 2.5 *deg/px angular resolution and 5.14b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.14c at* 2.5 *deg/px angular resolution and 5.14d* 5.0 *deg/px at angular resolution.*



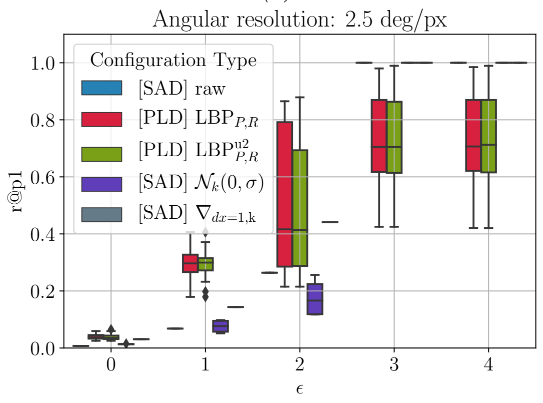
(a)



(b)



(c)



(d)

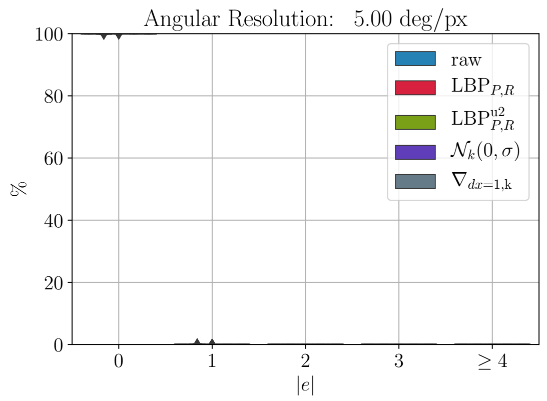
Figure 5.15: Performance summary on UrbanCanyon visual memories for 0.8 m off-route translation. Global localization performance: configurations 5.15a at 2.5 *deg/px angular resolution and 5.15b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.15c at* 2.5 *deg/px angular resolution and 5.15d* 5.0 *deg/px at angular resolution.*

5.7.2 Robustness to Off-route Translation using Synthetic Data and Reduced VFOV

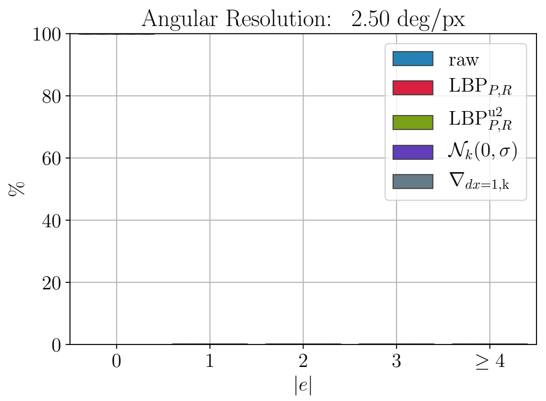
The previous results in synthetic visual memories had an major difference w.r.t. those in the real world. In synthetic visual memories, configurations based on 5.0 deg/px angular resolution, reached $r@p1 = 1$ for $\epsilon = 0$ and small off-route deviation (± 0.2 m). Performance degraded considerably for a off-route translation of ± 0.8. The off-route translation in real-world visual memories is not known, but due to road vehicle sizes, its likely to be closer to ± 0.2 than ± 0.8 in general. Specially in the Garden visual memories, as the road was barely wide enough for the vehicle, and obstacle avoidance maneuvers were never necessary during demonstration.

However, there were differences in the setup, especially regarding the VFOV. In real-world visual memories, the VFOV was only of 45 degrees, with ≈ 17 degrees below the horizontal plane. In contrast, the synthetic visual memories had ≈ 100 degrees of VFOV equally distributed above and below the horizontal plane. This raises the question whether that large difference in VFOV was responsible for differences in the results. To answer that question, the experiment presented in the previous section was repeated with a VFOV that approximates that of the real-world visual memories.

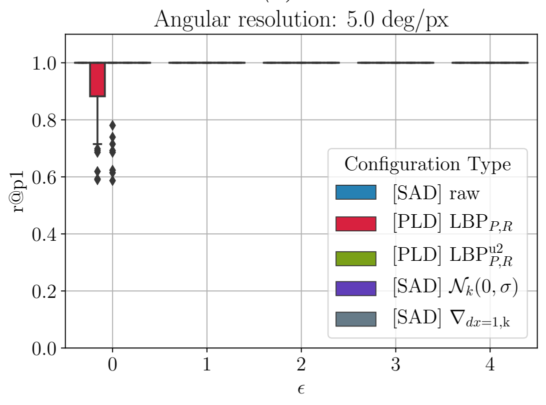
Figure 5.16 and Figure 5.17 show that under an off-route displacement of ± 0.2 m, global localization still finds the correct corresponding snapshot 100 % of the time. In contrast, some LBP parameterizations perform poorly, reaching $r@p1 \approx 0.5$.



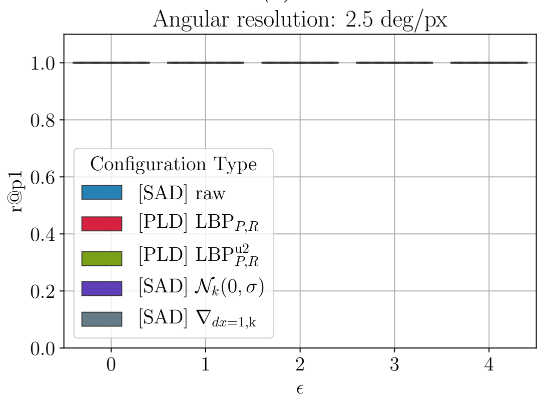
(a)



(b)

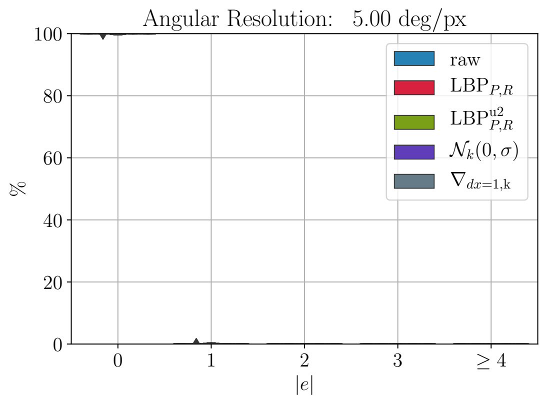


(c)

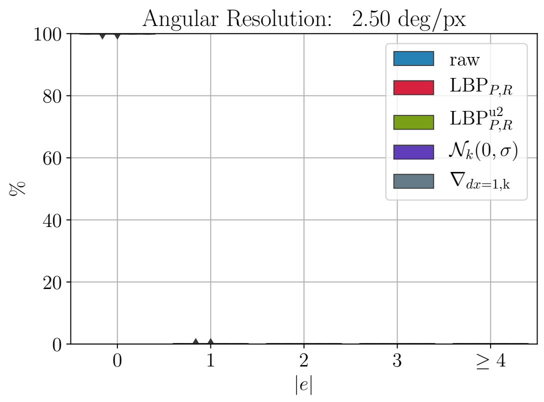


(d)

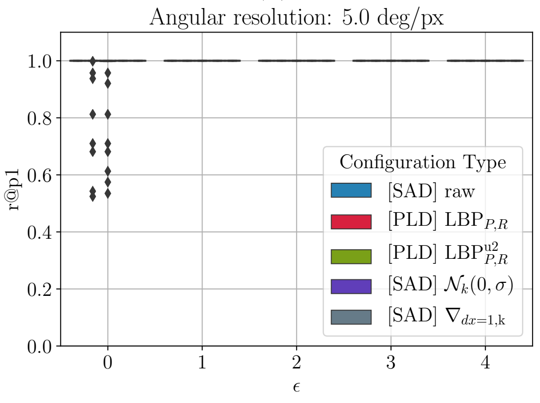
Figure 5.16: Performance summary on UrbanCanyon visual memories for -0.2 m off-route translation with reduced VFOV. Global localization performance: configurations 5.16a at 2.5 *deg/px angular resolution and 5.16b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.16c at* 2.5 *deg/px angular resolution and 5.16d at* 5.0 *deg/px angular resolution.*



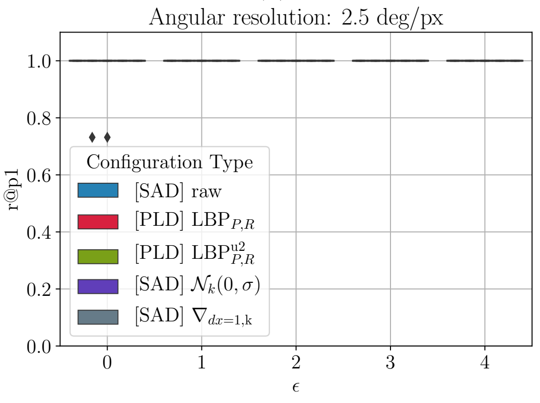
(a)



(b)

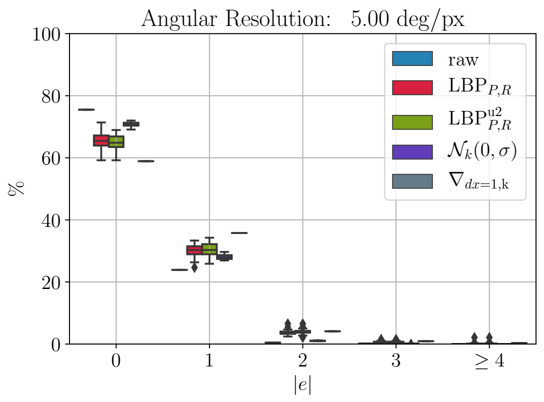


(c)

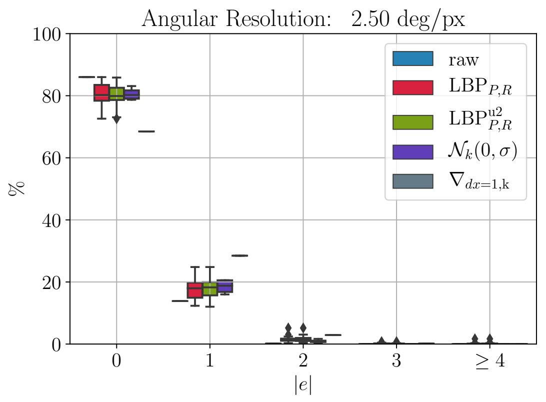


(d)

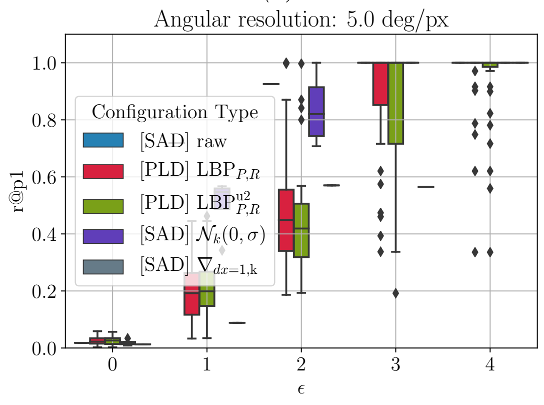
Figure 5.17: Performance summary on UrbanCanyon visual memories for 0.2 m off-route translation with reduced VFOV. Global localization performance: configurations 5.17a at 2.5 *deg/px angular resolution and 5.17b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.17c at* 2.5 *deg/px angular resolution and 5.17d at* 5.0 *deg/px angular resolution.*



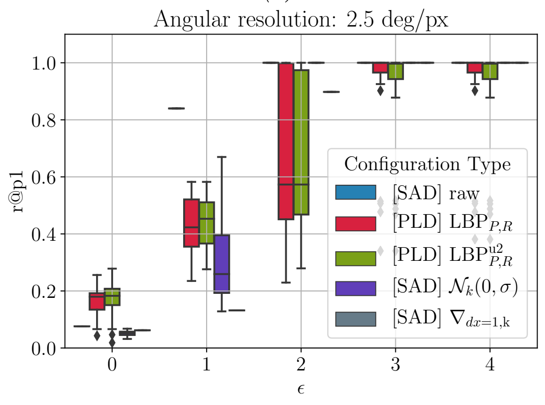
(a)



(b)

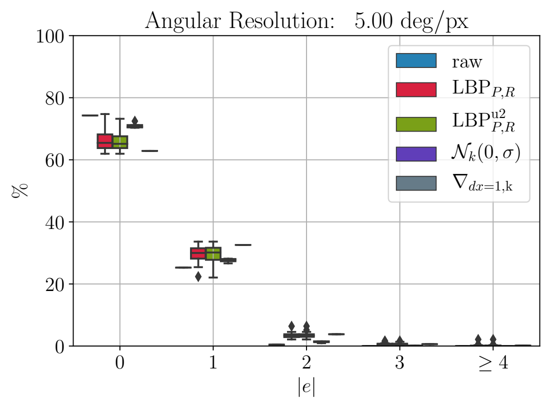


(c)

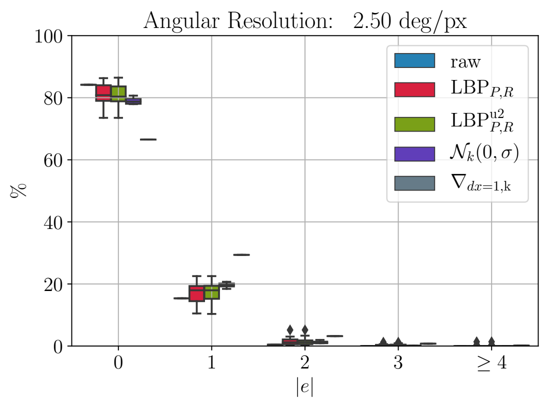


(d)

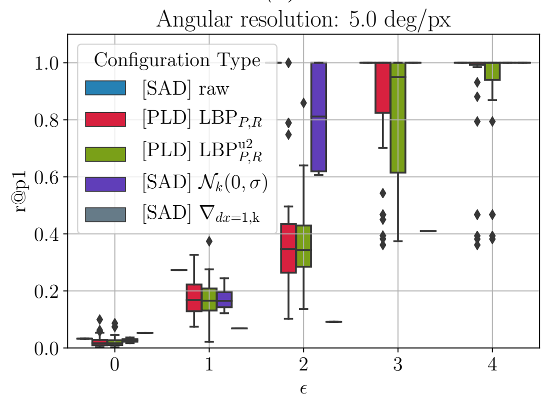
Figure 5.18: Performance summary on UrbanCanyon visual memories for -0.8 m off-route translation with reduced VFOV. Global localization performance: configurations 5.18a at 2.5 *deg/px angular resolution and 5.18b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.18c at* 2.5 *deg/px angular resolution and 5.18d at* 5.0 *deg/px angular resolution.*



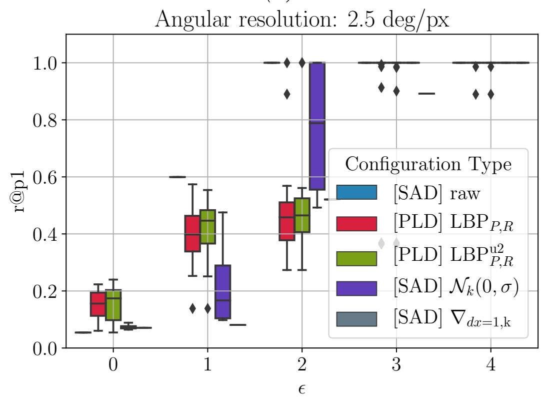
(a)



(b)



(c)



(d)

Figure 5.19: Performance summary on UrbanCanyon visual memories for 0.8 m off-route translation with reduced VFOV. Global localization performance: configurations 5.19a at 2.5 *deg/px angular resolution and 5.19b at* 5.0 *deg/px angular resolution. Localization classification performance: 5.19c at* 2.5 *deg/px angular resolution and 5.19d* 5.0 *deg/px at angular resolution.*

Interpretation of the Results

Two important insights come forefront. The first one is that global localization is barely affected by the choice of a configuration. In contrast, localization classification performance depends on the configuration used and the environment. One explanation for this is that global localization only necessitates the existence of a global minimum in the neighborhood of the snapshot corresponding to a current view. In contrast, localization classification based on thresholding of the IDF values has a more astringent requirement. It necessitates that correct classifications (tp and tn) and wrong classifications (fp and fn) fall within non-overlapping ranges of IDF values. That requirement is even more strict because of the use of r@p1 as metric because it willingly sacrifices recall for the sake of precision. Nevertheless, in a real navigation scenario, this is a sensible choice, given the dangers of navigating while lost and unaware of it.

Due to the nature of real-world data, it is not possible to completely isolate all possible sources for differences in the results. However, it is possible to approximate an explanation given the knowledge of the acquisition conditions. In real-world datasets, the human-demonstrator was driving at low speed and mainly in a straight line. This likely helped to eliminate non-planar motions. Non-planar motions due to ground properties were especially unlikely in the Steglitz visual memories because the ground was predominantly flat. In the Garden visual memories, the road was not entirely flat, which likely induced low-magnitude non-plannar motions due to vibrations. Moreover, the vehicle was moving over slopes, which likely induced roll in the smart-wheelchair due to the action of gravity. However, this does not seem to hinder matching in the sense that global localization with the raw representation often confuses current-views with snapshots in parts of the route with opposite slopes and aliased appearance. Thus, the author considers it unlikely that ground properties were to blame.

Off-route translation is another factor that may have affected the results. The experiments on simulation demonstrate that even under an ideal static environment, large off-route translation (± 0.8 m) can severely hinder localization classification. However, the same experiments evidence that if the magnitude of such translations is low, e.g., ± 0.2 m, then localization classification performance is not affected. It is not possible to quantify the precise off-route translation present in real-world datasets because of the lack of precise position information. However, due to the size of the smart-wheelchair, and the size of the areas used for navigation, off-route translational error was unlikely larger than 20 to 30 cm in the worst case. Thus, the author considers it as an unlikely source. In view of the above, the author considers that the main source of differences was the appearance of objects present in the scene and the existence of moving objects. In the Garden visual memories, objects in the scene were static, and consisted pervasively of the sky, the road, and vegetation. The coverage of the sky was also considerably larger in these visual memories, given the lack of nearby tall structures. The scene was practically unaltered despite the 2 months time span between captures. However, the weather conditions were very different. $\nabla_{dx=1,k}$ and $\mathcal{N}_k(0,\sigma)$ performed better than raw, meaning that illumination invariance was necessary for good performance. However, the panorama of textons performed poorly.

In the Steglitz datasets, the objects were buildings, roads, pedestrians and cars, and the proportion of visible sky was smaller thanks to the presence of tall buildings in large segments of the route. The texture present in these objects is more structured than in the Garden visual memories, because it consists of man-made objects. Thus, it appears as if these types of objects lead to a better IDF values that comply with the requirements of localization classification. The main distinctive characteristic of the Steglitz datasets, which was not present in any other, was the presence of moving objects. Thus, the results suggest that the panorama of textons were able to filter them out of the scene. However, this was only possible when using the higher resolution (2.5 deg/px).

However, higher resolution is not necessarily helpful as exhibited by the results of Section 5.7.1. In a static scene and under large off-route translational errors (± 0.8 m), higher resolution resulted detrimental to the results. The author of this dissertation interprets these results based on Wystrach et al. (2016). The low resolution was able to filter out details that are misleading to the image matching process.

5.7.3 Performance of the PCHL

n good-performing panorama of textons represetations were selected based on the results of previous section. Those representations were used to construct PCHL with varying parameters. The results are summarized in the following table.

Global localization performance.

Localization classification performance.

5.7.4 Performance of the TVCC

The effects on localization classification performance is evaluated using the top n configurations found in Section 5.7.1.

5.7.5 Evaluating the Need for a Precise Intrinsic Model

This section explores if, and to what extent, the use of a precise intrinsic model for unwrapping improves image matching results, being all other things equal. The experiment consists of evaluating the configurations used before on the UrbanCanyon visual memories with 0.8m off-route translation, and compare them to the results presented in Section 5.7.1.

Chapter 6

Conclusions

6.1 Texture-based Representations for Holistic Methods

Research on the VCA has been historically conservative when it comes to the configurations used. Most research has limited to use low-resolution raw images, perhaps with simple preprocessing steps to achieve illumination invariance (Zeil et al., 2003; Binding and Labrosse, 2006; Möller et al., 2014), or to the use of skyline contours (Möller, 2002; Wystrach et al., 2012). This is probably not surprising given the biological inspiration of the VCA. However, ever since Wehner (1972) suggested that the visual memory of insects is formed by storing retinotopic images, there has been a debate on what information is extracted from them and used for navigation.

Dittmar et al. (2010) suggested the use of texture. However, the authors attempted to explain their observations in terms of raw panoramic image matching. This dissertation followed an approach that its author considers more grounded in terms of texture analysis: panoramic images of textons. To the best of this dissertation author's knowledge, that approach had not been used with the VCA. Moreover, using LBP images directly as representation, known as, Holistic LBP Image (hLBPI), is extremely rare even in the most common application of LBP: facial recognition (Yang and Chen, 2013). In most applications, LBP images are an intermediate step towards the computation of a density-based descriptor. The feasibility of using a similar approach with the VCA was also evaluated. However, using the panorama of textons performed better than the PCHL representation.

Panorama of textons were the best performers among the representations evaluated in this dissertation: They provided the best results both in global localization and in the detection of failure cases. All representations performed poorly on failure detection in a highly symmetric environment. However, even in that case, the panorama of textons produces a stronger signal.

The performance of the panorama of textons depends on the LBP variant and its parameters. However, in general, non-rotation invariant variants performed better than rotation invariant ones. This situation is analog to what has been already described using point-based feature descriptors (Valgren and Lilienthal, 2010).

Rotation invariance is detrimental to performance because that is not a viewpoint variation generally present when navigating with a wheeled robot. However, this may not be the case with other applications, like flying robots. Insects seem to handle this situation by aligning their heads to counteract the off-plane rotation (Ardin et al., 2015; Raderschall et al., 2016).

6.2 Complete Knowledge Transfer

For example, using gradient-descent-based optimization seems reasonable given that their effect size correlates with the magnitude of the viewpoint differences.

6.3 Global Localization and Failure Detection

The VCA performed better at global localization than it was originally expected given the results of Smith et al. (2008), especially on simulation, and despite off-route error of larger magnitude than what could be attributed to motor noise. Performance on real world data was also encouraging because, in most cases, the algorithm found either the correct best match or the second-best match (one snapshot apart). Thus, in most cases, the global localization error remains less than 0.5 m. Experiments on the SunnyGarden and OvercastGarden were the exception. Global localization performed poorly on them, but that was expectable because the environment is highly symmetric. However, even in that case, the panorama of textons yield a stronger signal, as seen in Figure. Therefore, for that type of environment, the robot would require a localization prior to navigate, i.e., know its initial position. In practical applications, this could be done with a low-cost GPS. However, the analysis presented in this dissertation is limited to global localization.

Unfortunately, Smith et al. (2008) did not provide important details, e.g., distance between snapshots, that allow a more detailed interpretation of their results. Nevertheless, the results of this dissertation suggest that the VCA can be used for global localization and failure detection as long as places have a distinctive appearance.

6.4 Knowledge Transfer is Possible

This dissertation provided a proof of concept of the feasibility of knowledge transfer. It used a simplified scenario in which source and target omnidirectional cameras had the exact same extrinsic parameters but different intrinsic parameters.

Knowledge transfer consisted simply of a rough equalization of the VFOV. The unwrapping procedure used does not take into account important aspects of omnidirectional sensors, like nonlinearities in image acquisition. Because of this, it was obvious from the beginning that accurate pixel correspondences would not be available even if the extrinsics of two images were exactly the same.

Nevertheless, the author of this dissertation decided to test the feasibility of knowledge transfer under these conditions for two reasons. The first one is that holistic methods are often considered and claimed to perform without the necessity of an accurate calibration model. Therefore, the evaluation could find evidence either in favor or against that claim. The second reason is that this is a quite difficult scenario, and moving towards easier scenarios by means of a model like that of Scaramuzza et al. (2006a) would result relatively easy.

This dissertation just scratched the surface of knowledge transfer. A complete transfer procedure should also consider differences in camera extrinsic parameters, nonlinearities of the image acquisition, differences induced by the sensors used. However, with the results provided, the author of this dissertation hopes to open the discussion for this neglected aspect of the VCA and related methods that is important for their applicability on large-scale real world navigation.

6.5 VCA Configurations Can Be Combined

Previous research on the VCA considered configuration in a mutually exclusive way. Either one or the other is used, and perhaps compared against each other. The experiments presented in this dissertation suggest that multiple representations can be used to improve performance, because one representation perform well in different ocassions. This dissertation proposed the TVCC to that end.

One option could be to use labeled datasets of localization failure and success cases under different conditions. However, this is a complex learning problem because of the asymmetry in the class distributions. The cascade classifier is particularly well-known known to require very large amounts of training data so that reliable false-positive rates can be estimated and each stage and because each stage requires a new set of negative examples. That situation would also arise in the TVCC. It would be more convenient if the model parameters could be inferred directly from the limited examples contained in the visual memory used to follow a route.

6.6 Implications for Smart-Wheelchairs and Robots

Several of the aspects addressed in this dissertation are very relevant for the real world applicability of the VCA, and related algorithms. I do not consider that the algorithms are yet mature enough for application on security critical applications, like wheelchair navigation. However, the results suggest that progress in those directions can be achieved by relatively simple means.

The author of this dissertation considers that further progress in holistic methods could result an attractive venue for applications like smart-wheelchair navigation for several reasons.

The first one is that having a camera as the main sensor is very convenient because they are relatively low-cost sensors. The second is that algorithms like the VCA can perform even in modest hardware. Furthermore, their algorithmic simplicity makes the VCA very well suited for implemetation on low power-consumptions hardware, like FPGAs.

6.7 Continued Necessity to Experiment on Complex Scenarios

There were some unexpected results in some of the experiments. For example, based on the results of Wystrach et al. (2016), it was not expected that lower resolution performs worse than higher resolution. That was true for synthetic data, but not for real world data. Initially, the author of this dissertation attributed the performance difference to two reasons. The first one was the difference in VFOV: real world data had a smaller VFOV, mostly limited at low elevations. Because of this, it was possible that nearby objects were present to a lesser extent. The second reason was the off-route error induced by the human driver. Further experimentation suggested that that was not the cause.

From that, the author of this dissertation suggests that the appearance changes due to the sky and moving objects were the main reason and that the higher resolution helped to reflect those changes in the IDF values. Having a larger VFOV coverage had probably mitigated the effects of the sky, but had increased the likelihood that other objects, like pedestrians, affected the IDF values. Another possible reason for the discrepancy with Wystrach et al. (2016), could be due to image configuration. The authors used binary images that differentiated between sky and foreground. Although this is straightforward to test in simulation, it is more complex when working in complex real world scenery. Although it is known that some insects can easily segment the sky and the foreground (Basten and Mallot, 2010; Möller, 2002; Stone et al., 2018; Freas et al., 2017), such segmentation is not straightforward for a robot. Especially, one that does not have specialized sensors for the purpose, like UV cameras.

A similar discrepancy between expectation and outcome occurred regarding global localization performance of the VCA, as explained before. From this, the author of this dissertation concludes that more research of the VCA and related methods under challenging outdoor conditions is required.

One of the main lessons obtained in this investigation is that the performance of the VCA can be very different in simulation, i.e., controlled conditions, compared to real world data. The author of this dissertation concludes that even more exhaustive evaluations on real world data are necessary. This necessity has been pointed out by (Zeil et al., 2003), and also by researchers on insect navigation. Some of the results presented in this dissertation can be seen as supporting that necessity.

6.8 Possible Implications in Biology

It is difficult to extrapolate the results of a computer vision experiment to biology. However, experiments like the ones presented in this dissertation are a common venue to, at least, offer possible explanations to the mechanisms behind insect navigation. The results exposed in this dissertation may provide some hints to visual processing in insects. The first one is that global localization can be performed reliably, and it is highly robust to parameterization. The second is that failure detection is a much more difficult problem, at least with the classifiers used in this dissertation. The results show that simple illumination invariance is astonishingly effective in some cases. Moreover, the results show that different resolutions may be necessary for different conditions. Or perhaps some other mechanism like spatiotemporal fusion may be at play, instead of using more discriminative representations as hereby proposed.

6.9 Future Work

6.9.1 Learning Trees of VCA Configurations

This dissertation provided evidence that supports the possibility of combining multiple VCA configurations. However, a method for learning a tree of VCA configurations was not proposed. The model used for experimentation was constructed using the same thresholds used to calculate $p@r = 1$. Therefore, the generalization power of this approach is debatable. Thus, it would be interesting to devise a learning algorithm and test its generalization capabilities.

6.9.2 Integrating the VCA with High-Resolution 3D Metric Maps

One of the benefits of using a route-following system, as described in this dissertation, is that it is easy to acquire a representation of the environment. A human only has to drive along a route while the robot captures snapshots. However, the advent of autonomous cars has increased the interest in high-definition (HD) 3D maps. Although HD maps may seem irreconcilable with route-based approaches as the one presented in this dissertation, it would be interesting to study how to combine them. For example, HD maps could be used to generate synthetic views that the route-based approach could use to navigate. The route-based approach could be used to provide coverage to areas without a metric map with the help of a demonstrator. Similar efforts have been performed, in related problems so that an expensive mapping setup can be coupled with an inexpensive autonomous navigation one (Benosman et al., 1996; Gluckman, 1998).

6.9.3 Other Types of Pixel Representations

One of the main findings of this dissertation is that the use of discriminant pixel values can improve the performance of the VCA in some situations. The results presented rely on LBP, a classic technique, in texture analysis. After the introduction of LBP, there have been many developments in the area, either in the form of new LBP variants or in the form of alternative texture primitives. Therefore, it would be very interesting to see how other types of pixel labels perform.

Bibliography

A., F. C. and Ken, C. (2017). Learning and time-dependent cue choice in the desert ant, melophorus bagoti. *Ethology*, 123(8):503–515.

Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). *Face Recognition with Local Binary Patterns*, pages 469–481. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ardin, P., Mangan, M., Wystrach, A., and Webb, B. (2015). How variation in head pitch could affect image matching algorithms for ant navigation. *Journal of Comparative Physiology A*, 201(6):585–597.

Argyros, A. A., Bekris, K. E., Orphanoudakis, S. C., and Kavraki, L. E. (2005). Robot homing by exploiting panoramic vision. *Autonomous Robots*, 19(1):7–25.

Baddeley, B., Graham, P., Husbands, P., and Philippides, A. (2012). A model of ant route navigation driven by scene familiarity. *PLoS computational biology*, 8(1):e1002336.

Baddeley, B., Graham, P., Philippides, A., and Husbands, P. (2011). Holistic visual encoding of ant-like routes: Navigation without waypoints. *Adaptive Behavior*, 19(1):3–15.

Basten, K. and Mallot, H. A. (2010). Simulated visual homing in desert ant natural environments: efficiency of skyline cues. *Biological Cybernetics*, 102(5):413–425.

Bellotto, N., Burn, K., Fletcher, E., and Wermter, S. (2008). Appearance-based localization for mobile robots using digital zoom and visual compass. *Robotics and Autonomous Systems*, 56(2):143 – 156.

Bennett, A. T. (1996). Do animals have cognitive maps? *Journal of Experimental Biology*, 199(1):219–224.

Benosman, R., Maniere, T., and Devars, J. (1996). Multidirectional stereovision sensor, calibration and scenes reconstruction. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 161–165 vol.1.

Binding, D. and Labrosse, F. (2006). Visual local navigation using warped panoramic images. In *Proceedings of Towards Autonomous Robotic Systems. University of Surrey, Guildford, UK*, pages 19–26.

Bock, O. *Xeno - Instructions for use.*

Buschbeck, E. K. and Friedrich, M. (2008). Evolution of insect eyes: Tales of ancient heritage, deconstruction, reconstruction, remodeling, and recycling. *Evolution: Education and Outreach*, 1(4):448–462.

Carlevaris-Bianco, N. and Eustice, R. M. (2014). Learning visual feature descriptors for dynamic lighting conditions. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2769–2776.

Cartwright, B. A. and Collett, T. S. (1983). Landmark learning in bees. *Journal of comparative physiology*, 151(4):521–543.

Cartwright, B. A. and Collett, T. S. (1987). Landmark maps for honeybees. *Biological Cybernetics*, 57(1):85–93.

Caselitz, T., Steder, B., Ruhnke, M., and Burgard, W. (2016). Monocular camera localization in 3d lidar maps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1926–1931.

Chatila, R. and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 138–145. IEEE.

Cheng, K., Collett, T. S., Pickhard, A., and Wehner, R. (1987). The use of visual landmarks by honeybees: Bees weight landmarks according to their distance from the goal. *Journal of Comparative Physiology A*, 161(3):469–475.

Cheng, K., Collett, T. S., and Wehner, R. (1986). Honeybees learn the colours of landmarks. *Journal of Comparative Physiology A*, 159(1):69–73.

Cheung, A., Collett, M., Collett, T. S., Dewar, A., Dyer, F., Graham, P., Mangan, M., Narendra, A., Philippides, A., Stürzl, W., Webb, B., Wystrach, A., and Zeil, J. (2014). Still no convincing evidence for cognitive map use by honeybees. *Proceedings of the National Academy of Sciences*, 111(42):E4396–E4397.

Chittka, L. (2004). Color vision in bees : mechanisms, ecology and evolution. *How simple nervous systems create complex perceptual worlds.*

Churchill, D. and Vardy, A. (2008). Homing in scale space. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1307–1312.

Coggins, J. M. and Jain, A. K. (1985). A spatial filtering approach to texture analysis. *Pattern recognition letters*, 3(3):195–203.

Collett, M. and Collett, T. S. (2006). Insect navigation: No map at the end of the trail? *Current Biology*, 16(2):R48 – R51.

Collett, T. S. and Land, M. F. (1975a). Visual control of flight behaviour in the hoverfly syritta pipiens l. *Journal of comparative physiology*, 99(1):1–66.

Collett, T. S. and Land, M. F. (1975b). Visual spatial memory in a hoverfly. *Journal of comparative physiology*, 100(1):59–84.

Differt, D. (2017). Real-time rotational image registration. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 1–6.

Dittmar, L., Egelhaaf, M., Stürzl, W., and Boeddeker, N. (2011). The behavioral relevance of landmark texture for honeybee homing. *Frontiers in Behavioral Neuroscience*, 5:20.

Dittmar, L., Stürzl, W., Baird, E., Boeddeker, N., and Egelhaaf, M. (2010). Goal seeking in honeybees: matching of optic flow snapshots? *Journal of Experimental Biology*, 213(17):2913–2923.

Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. (1978). Robotic exploration as graph construction. *J. Comput., vol*, 7(3).

Esch, H. E. and Burns, J. E. (1995). Honeybees use optic flow to measure the distance of a food source. *Naturwissenschaften*, 82(1):38–40.

Fearn, T., Labrosse, F., and Shaw, P. (2019). Wheelchair navigation: Automatically adapting to evolving environments. In Althoefer, K., Konstantinova, J., and Zhang, K., editors, *Towards Autonomous Robotic Systems*, pages 496–500, Cham. Springer International Publishing.

Fleer, D. (2017). Visual tilt estimation for planar-motion methods in indoor mobile robots. *Robotics*, 6(4):32.

Fleer, D. and Möller, R. (2017). Comparing holistic and feature-based visual methods for estimating the relative pose of mobile robots. *Robotics and Autonomous Systems*, 89:51 – 74.

Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998a). Learning view graphs for robot navigation. *Autonomous Robots*, 5(1):111–125.

Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998b). Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, 79(3):191–202.

Franz, M. O., Stürzl, W., Hübner, W., and Mallot, H. A. (2007). A robot system for biomimetic navigation–from snapshots to metric embeddings of view graphs. In *Robotics and cognitive approaches to spatial mapping*, pages 297–314. Springer.

Franz, M. O., Stürzl, W., Hübner, W., and Mallot, H. A. (2008). *A Robot System for Biomimetic Navigation – From Snapshots to Metric Embeddings of View Graphs*, pages 297–314. Springer Berlin Heidelberg, Berlin, Heidelberg.

Freas, C. A., Whyte, C., and Cheng, K. (2017). Skyline retention and retroactive interference in the navigating australian desert ant, melophorus bagoti. *Journal of Comparative Physiology A*, 203(5):353–367.

Freas, C. A., Wystrach, A., Narendra, A., and Cheng, K. (2018). The view from the trees: Nocturnal bull ants, myrmecia midas, use the surrounding panorama while descending from trees. *Frontiers in Psychology*, 9:16.

Friedrich, H., Dederscheck, D., Mutz, M., and Mester, R. (2008). View-based robot localization using illumination-invarant spherical harmonics descriptors. In *Proceedings of the International Joint Conference on Computer Vision and Computer Graphics Theory and Applications (VISAP). INSTICC*. Citeseer.

Frigge, M., Hoaglin, D. C., and Iglewicz, B. (1989). Some implementations of the boxplot. *The American Statistician*, 43(1):50–54.

Gaussier, P., Joulain, C., Banquet, J., Leprêtre, S., and Revel, A. (2000). The visual homing problem: An example of robotics/biology cross fertilization. *Robotics and Autonomous Systems*, 30(1):155 – 180.

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

Gibson, J. J. (1950). The perception of the visual world.

Gluckman, J. (1998). Real-time omnidirectional and panoramic stereo. *Proc. Image Understanding Workshop, 1998*.

Gluckman, J. and Nayar, S. K. (1998). Ego-motion and omnidirectional cameras. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 999–1005.

Goedemé, T., Tuytelaars, T., and Van Gool, L. (2005). Omnidirectional sparse visual path following with occlusion-robust feature tracking. dans in: 6th workshop on omnidirectional vision, camera networks and non-classical cameras, omnivis05. *Conjunction with ICCV*, 2005.

Graham, P. and Cheng, K. (2009). Which portion of the natural panorama is used for view-based navigation in the australian desert ant? *Journal of Comparative Physiology A*, 195(7):681.

Graham, P. and Philippides, A. (2014). *Insect-Inspired Visual Systems and Visually Guided Behavior*, pages 1–9. Springer Netherlands, Dordrecht.

Graham, P. and Philippides, A. (2017). Vision for navigation: What can we learn from ants? *Arthropod Structure & Development*, 46(5):718 – 722. From Insects to Robots.

Grewal, H., Matthews, A., Tea, R., and George, K. (2017). Lidar-based autonomous wheelchair. In *2017 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE.

Hafner, V. V. and Möller, R. (2001). Learning of visual navigation strategies. In *Proc. European Workshop of Learning Robots (EWLR-9), Prague.*

Hölldobler, B. (1980). Canopy orientation: A new kind of orientation in ants. *Science*, 210(4465):86–88.

Horridge, A. (2009). What does an insect see? *Journal of Experimental Biology*, 212(17):2721–2729.

Jeong, K.-H., Kim, J., and Lee, L. P. (2006). Biologically inspired artificial compound eyes. *Science*, 312(5773):557–561.

Judd, S. and Collett, T. (1998). Multiple stored views and landmark guidance in ants. *Nature*, 392(6677):710.

Julesz, B. (1981). Textons: The elements of texture perception and their interactions. *Nature*, 290(5802):91–97.

Juusola, M., Dau, A., Song, Z., Solanki, N., Rien, D., Jaciuch, D., Dongre, S. A., Blanchard, F., de Polavieja, G. G., Hardie, R. C., et al. (2017). Microsaccadic sampling of moving image information provides drosophila hyperacute vision. *Elife*, 6:e26117.

Jähne, B. (2010). Emva 1288 standard for machine vision. *Optik & Photonik*, 5(1):53–54.

Kohler, M. and Wehner, R. (2005). Idiosyncratic route-based memories in desert ants, melophorus bagoti: how do they interact with path-integration vectors? *Neurobiology of learning and memory*, 83(1):1–12.

Kolter, J. Z., Plagemann, C., Jackson, D. T., Ng, A. Y., and Thrun, S. (2010). A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. In *2010 IEEE International Conference on Robotics and Automation*, pages 839–845.

Kuipers, B. and Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1-2):47–63.

Labrosse, F. (2006). The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941.

Labrosse, F. (2007). Short and long-range visual navigation using warped panoramic images. *Robotics and Autonomous Systems*, 55(9):675 – 684. Towards Autonomous Robotic Systems 2007: Mobile Robotics in the UK.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30(1–2):39 – 64.

Land, M. F. (1997a). The resolution of insect compound eyes. *Israel Journal of Plant Sciences*, 45(2-3):79–91.

Land, M. F. (1997b). Visual acuity in insects. *Annual review of entomology*, 42(1):147–177.

Lang, L. and Mohnen, A. (2019). An organizational view on transport transitions involving new mobility concepts and changing customer behavior. *Environmental Innovation and Societal Transitions*, 31:54 – 63.

Leaman, J. and La, H. M. (2017). A comprehensive review of smart wheelchairs: Past, present, and future. *IEEE Transactions on Human-Machine Systems*, 47(4):486–499.

Li, R., Oskoei, M. A., and Hu, H. (2013). Towards ros based multi-robot architecture for ambient assisted living. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3458–3463.

Liu, M., Pradalier, C., Pomerleau, F., and Siegwart, R. (2012). The role of homing in visual topological navigation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 567–572.

Llarena, A. and Rojas, R. (2016). I am alleine, the autonomous wheelchair at your service. In Menegatti, E., Michael, N., Berns, K., and Yamaguchi, H., editors, *Intelligent Autonomous Systems 13*, pages 1613–1626, Cham. Springer International Publishing.

Lourenco, M., Barreto, J. P., and Vasconcelos, F. (2012). srd-sift: Keypoint detection and matching in images with radial distortion. *IEEE Transactions on Robotics*, 28(3):752–760.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Makadia, A. and Daniilidis, K. (2003). Direct 3d-rotation estimation from spherical images via a generalized shift theorem. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–217.

Makadia, A. and Daniilidis, K. (2006). Rotation recovery from spherical images without correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1170–1175.

Mallock, A. (1894). I. insect sight and the defining power of composite eyes. *Proceedings of the Royal Society of London*, 55(331-335):85–90.

Mangan, M. and Webb, B. (2012). Spontaneous formation of multiple routes in individual desert ants (cataglyphis velox). *Behavioral Ecology*, 23(5):944–954.

Masci, J., Migliore, D., Bronstein, M. M., and Schmidhuber, J. (2014). *Descriptor Learning for Omnidirectional Image Matching*, pages 49–62. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mataric, M. J. (1990). A distributed model for mobile robot environment-learning and navigation. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB.

Menzel, R., Geiger, K., Joerges, J., Müller, U., and Chittka, L. (1998). Bees travel novel homeward routes by integrating separately acquired vector memories. *Animal Behaviour*, 55(1):139 – 152.

Meyer, B. D. and Mok, W. K. (2019). Disability, earnings, income and consumption. *Journal of Public Economics*, 171:51 – 69. Trans-Atlantic Public Economics Seminar 2016.

Milford, M., Scheirer, W., Vig, E., Glover, A., Baumann, O., Mattingley, J., and Cox, D. (2014). Condition-invariant, top-down visual place recognition. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5571–5577.

Möller, R., Krzykawski, M., and Gerstmayr, L. (2010). Three 2d-warping schemes for visual robot navigation. *Autonomous Robots*, 29(3):253–291.

Möller, R. and Vardy, A. (2006a). Local visual homing by matched-filter descent in image distances. *Biological cybernetics*, 95(5):413–430.

Möller, R. and Vardy, A. (2006b). Local visual homing by matched-filter descent in image distances. *Biological Cybernetics*, 95(5):413–430.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121.

Morbidi, F. and Caron, G. (2017). Phase correlation for dense visual compass from omnidirectional camera-robot images. *IEEE Robotics and Automation Letters*, 2(2):688–695.

Müller, M. and Wehner, R. (1988). Path integration in desert ants, cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85(14):5287–5290.

Möller, R. (2002). Insects could exploit uv–green contrast for landmark navigation. *Journal of Theoretical Biology*, 214(4):619 – 631.

Möller, R. (2009). Local visual homing by warping of two-dimensional images. *Robotics and Autonomous Systems*, 57(1):87 – 101.

Möller, R., Horst, M., and Fleer, D. (2014). Illumination tolerance for visual navigation with the holistic min-warping method. *Robotics*, 3(1):22–67.

Narendra, A. (2007a). Homing strategies of the australian desert ant melophorus bagoti i. proportional path-integration takes the ant half-way home. *Journal of Experimental Biology*, 210(10):1798–1803.

Narendra, A. (2007b). Homing strategies of the australian desert ant melophorus bagoti ii. interaction of the path integrator with visual cue information. *Journal of Experimental Biology*, 210(10):1804–1812.

Narendra, A. and Ramirez-Esquivel, F. (2017). Subtle changes in the landmark panorama disrupt visual navigation in a nocturnal bull ant. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 372(1717).

Narin, B., Brian, M., and Smart, W. D. (2018). A critical look at smart wheelchairs. *arXiv preprint arXiv:1809.00291*.

Nasri, Y., Vauchey, V., Khemmar, R., Ragot, N., Sirlantzis, K., and Ertaud, J.-Y. (2016). Ros-based autonomous navigation wheelchair using omnidirectional sensor. *International Journal of Computer Applications*, 133(6):12–17.

Nauth, P. (2014). Akzeptanzverbesserung autonomer assistenzroboter durch verhaltensadaptivität. *Technische Unterstützungssysteme, die die Menschen wirklich wollen*, page 381.

Nelson, R. C. (1991). Visual homing using an associative memory. *Biological Cybernetics*, 65(4):281–291.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Papi, F. (2012). *Animal homing*. Springer Science & Business Media.

Payá, L., Gil, A., and Reinoso, O. (2017). A state-of-the-art review on mapping and localization of mobile robots using omnidirectional vision sensors. *Journal of Sensors*, 2017.

Philippides, A., Graham, P., Baddeley, B., and Husbands, P. (2015). *Using Neural Networks to Understand the Information That Guides Behavior: A Case Study in Visual Navigation*, pages 227–244. Springer New York, New York, NY.

Pollack, M. E. (2005). Intelligent technology for an aging population: The use of ai to assist elders with cognitive impairment. *AI magazine*, 26(2):9.

Raderschall, C. A., Narendra, A., and Zeil, J. (2016). Head roll stabilisation in the nocturnal bull ant myrmecia pyriformis: implications for visual navigation. *Journal of Experimental Biology*, 219(10):1449–1457.

Remazeilles, A., Chaumette, F., and Gros, P. (2006). 3d navigation based on a visual memory. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2719–2725.

Röfer, T. and Ofer, T. R. (1997). Controlling a wheelchair with image-based homing. In *Manchester University*, pages 66–75.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006a). A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006b). A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701.

Scaramuzza, D. and Siegwart, R. (2008). Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026.

Scaramuzza, D., Siegwart, R., and Martinelli, A. (2009). A robust descriptor for tracking vertical lines in omnidirectional images and its use in mobile robotics. *The International Journal of Robotics Research*, 28(2):149–171.

Schneider, J., Murali, N., Taylor, G. W., and Levine, J. D. (2018). Can drosophila melanogaster tell who's who? *PloS one*, 13(10).

Scudellari, M. (2017). Lidar-equipped autonomous wheelchairs roll out in singapore and japan. IEEE Spectrum, Oct. 2017.

Shih-Schon Lin and Bajcsy, R. (2003). High resolution catadioptric omnidirectional stereo sensor for robot vision. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 1694–1699 vol.2.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2004). Autonomous mobile robots. *Massachusetts Institute of Technology.*

Simpson, R. C., LoPresti, E. F., and Cooper, R. A. (2008). How many people would benefit from a smart wheelchair? *Journal of Rehabilitation Research & Development*, 45(1).

Smith, L., Philippides, A., Graham, P., Baddeley, B., and Husbands, P. (2007). Linked local navigation for visual route guidance. *Adaptive Behavior*, 15(3):257–271.

Smith, L., Philippides, A., Graham, P., and Husbands, P. (2008). *Linked Local Visual Navigation and Robustness to Motor Noise and Route Displacement*, pages 179–188. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sobel, E. C. (1990). The locust's use of motion parallax to measure distance. *Journal of Comparative Physiology A*, 167(5):579–588.

Song, Y. M., Xie, Y., Malyarchuk, V., Xiao, J., Jung, I., Choi, K.-J., Liu, Z., Park, H., Lu, C., Kim, R.-H., et al. (2013). Digital cameras with designs inspired by the arthropod eye. *Nature*, 497(7447):95.

Srinivasan, M., Chahl, J., Weber, K., Venkatesh, S., Nagle, M., and Zhang, S. (1999). Robot navigation inspired by principles of insect vision. *Robotics and Autonomous Systems*, 26(2):203 – 216. Field and Service Robotics.

Stone, T., Mangan, M., Wystrach, A., and Webb, B. (2018). Rotation invariant visual processing for spatial memory in insects. *Interface Focus*, 8(4):20180010.

Stürzl, W. and Mallot, H. A. (2002). Vision-based homing with a panoramic stereo sensor. In Bülthoff, H. H., Wallraven, C., Lee, S.-W., and Poggio, T. A., editors, *Biologically Motivated Computer Vision*, pages 620–628, Berlin, Heidelberg. Springer Berlin Heidelberg.

Stürzl, W. and Möller, R. (2007). An insect-inspired active vision approach for orientation estimation with panoramic images. In Mira, J. and Álvarez, J. R., editors, *Bio-inspired Modeling of Cognitive Tasks*, pages 61–70, Berlin, Heidelberg. Springer Berlin Heidelberg.

Stürzl, W. and Zeil, J. (2007). Depth, contrast and view-based homing in outdoor scenes. *Biological Cybernetics*, 96(5):519–531.

Surden, H. and Williams, M.-A. (2016). Technological opacity, predictability, and self-driving cars. *Cardozo L. Rev.*, 38:121.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Thrun, S., Gutmann, J.-S., Fox, D., Burgard, W., Kuipers, B., et al. (1998). Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *AAAI/IAAI*, pages 989–995.

Valgren, C. and Lilienthal, A. J. (2010). Sift, surf & seasons: Appearance-based long-term localization in outdoor environments. *Robot. Auton. Syst.*, 58(2):149–156.

Van Den Berg, J., Ferguson, D., and Kuffner, J. (2006). Anytime path planning and replanning in dynamic environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2366–2371. IEEE.

Vardy, A. and Moller, R. (2005). Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1-2):47–89.

Von Frisch, K. (1967). The dance language and orientation of bees.

Wang, Y., Gong, X., Lin, Y., and Liu, J. (2012). Stereo calibration and rectification for omnidirectional multi-camera systems. *International Journal of Advanced Robotic Systems*, 9(4):143.

Warrant, E. and Nilsson, D.-E. (2006). *Invertebrate vision*. Cambridge University Press.

Weber, K., Venkatesh, S., and Srinivasan, M. (1999). Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97.

Wehner, R. (1972). Dorsoventral asymmetry in the visual field of the bee,apis mellifica. *Journal of comparative physiology*, 77(3):256–277.

Wehner, R. (2003). Desert ant navigation: how miniature brains solve complex tasks. *Journal of Comparative Physiology A*, 189(8):579–588.

Wehner, R. and Flatt, I. (1977). Visual fixation in freely flying bees. *Zeitschrift für Naturforschung C*, 32(5-6):469–472.

Wehner, R., Michel, B., and Antonsen, P. (1996). Visual navigation in insects: coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199(1):129–140.

Wolcott, R. W. and Eustice, R. M. (2014). Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183.

World Health Organization and others (2011). *World report on disability 2011*. World Health Organization.

Wystrach, A., Beugnon, G., and Cheng, K. (2012). Ants might use different view-matching strategies on and off the route. *Journal of Experimental Biology*, 215(1):44–55.

Wystrach, A., Dewar, A., Philippides, A., and Graham, P. (2016). How do field of view and resolution affect the information content of panoramic scenes for visual navigation? a computational investigation. *Journal of Comparative Physiology A*, 202(2):87–95.

Wystrach, A., Schwarz, S., Baniel, A., and Cheng, K. (2013). Backtracking behaviour in lost ants: an additional strategy in their navigational toolkit. *Proceedings of the Royal Society B: Biological Sciences*, 280(1769):20131677.

Xu, Y., John, V., Mita, S., Tehrani, H., Ishimaru, K., and Nishino, S. (2017). 3d point cloud map based vehicle localization using stereo camera. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 487–492.

Yagi, Y. and Kawato, S. (1990). Panorama scene analysis with conic projection. In *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pages 181–187 vol.1.

Yang, B. and Chen, S. (2013). A comparative study on local binary pattern (lbp) based face recognition: Lbp histogram versus lbp image. *Neurocomputing*, 120:365 – 379. Image Feature Detection and Description.

Yu, S.-E. and Kim, D. (2011). Landmark vectors with quantized distance information for homing navigation. *Adaptive Behavior*, 19(2):121–141.

Zeil, J., Hofmann, M. I., and Chahl, J. S. (2003). Catchment areas of panoramic snapshots in outdoor scenes. *J. Opt. Soc. Am. A*, 20(3):450–469.

Zhang, A. M. and Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. *The International Journal of Robotics Research*, 28(3):331–356.

Zichao Zhang, Rebecq, H., Forster, C., and Scaramuzza, D. (2016). Benefit of large field-of-view cameras for visual odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808.

Zivkovic, Z. and Booij, O. (2005). How did we built our hyperbolic mirror omnidirectional camera-practical issues and basic geometry. *University of Amsterdam, Tech. Rep. IAS-UVA-05-04*.

Zollikofer, C., Wehner, R., and Fukushi, T. (1995). Optical scaling in conspecific cataglyphis ants. *Journal of Experimental Biology*, 198(8):1637–1646.