Mobile Robot Navigation Based on Difference Visual Streams

September 2016

HELIO PERRONI FILHO

Mobile Robot Navigation Based on Difference Visual Streams

Graduate School of Systems and Information Engineering

University of Tsukuba

September 2016

HELIO PERRONI FILHO

Abstract

Self-location is defined as the ability to recognize one's surroundings and reliably keep track of current position relative to a known environment. It is a fundamental cognitive skill for entities biological and artificial alike: living beings rely on it for performing key behaviors such as nest homing and predator evasion, whereas mobile robots require it to realize autonomous navigation.

At a minimum, self-location requires the ability to match current sensory input to memories of previously visited places, and to correlate perceptual changes to physical movement. Humans and other mammals rely mainly on visual cues for place identification, hinting at the possibility of robot systems using cameras and computer vision algorithms for the same task. However, visual input, while rich, is notoriously hard to interpret algorithmically, and is subject to large variations from factors such as lighting conditions, changes to environment composition, and the presence of moving obstacles.

This thesis proposes a new architecture for vision-based robot self-location, dubbed the Difference Image Correspondence Hierarchy, or DICH for short. DICH is inspired by research in biological cognition, constituting a self-location model that is perception-based instead of metric. Places, and robot location among them, are defined not in terms of geometric coordinates, but as memory and sensory patterns consistently correlated over time.

At the same time, DICH has from the beginning been conceived as a pragmatic approach to autonomous navigation, intended to work under real-world conditions. Accordingly, a DICH implementation based on well-known software libraries and off-the-shelf hardware is also presented. A variety of experiments demonstrate the architecture's features, its strengths and weaknesses. The thesis concludes with a discussion on results achieved and directions for further research.

Acknowledgments

> When one has finished building one's house, one suddenly realizes that in the process one has learned something that one really needed to know in the worst way – before one began.
>
> Friedrich Nietzsche

Thanks to my advisor Akihisa Ohya for guiding and supporting my research, to Claus Aranha for introducing me to him, and to my M.Sc. advisor Alberto Ferreira de Souza, who encouraged and helped me in my efforts to pursue a Ph.D. degree. Thanks also to Michael Cesare Gianturco for sharing several intriguing ideas, the practical implementation of which has been the drive to much of my work; Olivier Georgeon, for his insightful course on Developmental AI that directed me to the larger field of cognitive architectures.

Thanks to the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for financing my Ph.D. studies. Thanks to the University of Tsukuba for providing my wife and I with our living quarters, and for being a nice place to work and live. Also thanks to my colleagues in the Intelligent Robot Laboratory, in particular Ryan Pratama and Matsuzaki Sango.

These years in Japan would have been a lot tougher to get by, if not for some wonderful people. Kohei Hattori helped me a great deal as I was settling down to life in Japan, and I am forever indebted to Naoko Ohya's generosity. Alvaro Kanasiro and Kelly Tsutiya too supported me through many a bureaucratic conundrum, besides being good friends. And I will always fondly remember my conversations with Wagner Schmidt and Rafael Munia – though the subjects we debated were often grim, those were some of the best times of my life.

Thanks to my parents Helio and Diana for always supporting me through life, providing me a good education, and generally putting up with my difficult self.

Last but not least, I want to thank my wife Juliana. It is sometimes said that "behind every great man lies a great woman"; I'd hardly count myself as "great" and would rather have women stand besides men, but it's true that her support has been indispensable for the success of this little enterprise. This is in every sense as much your work as mine, my love.

Contents

List of Figures

List of Tables

Chapter 1

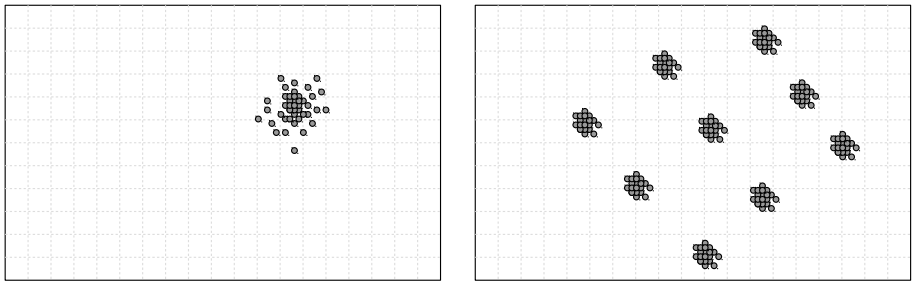
Introduction

This chapter discusses the subjects this thesis is concerned with, as well as its objectives. First it defines the problem of autonomous navigation, discussing its attendant parts and possible approaches. It is noted that self-location is a central requirement for effective navigation. Biologic self-location strategies are then described, and proposed as a template for the development of robotics solutions, helping select methods among the excess of available options. The Difference Image Correspondence Hierarchy (DICH) is presented as an effort in that direction, and its main features are briefly summarized. DICH is contrasted to similar projects in the field of Biologically Inspired Cognitive Architectures (BICA's), and shown to lie towards a more pragmatic philosophy. The chapter concludes with a statement of the objectives intended to be achieved with the architecture.

1.1 Autonomous navigation

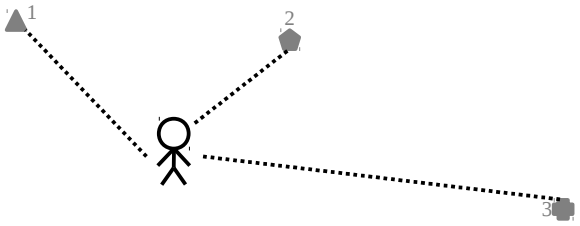
Autonomous navigation refers to the process by which a robot determines and then traverses a path between its current position and a specified destination. It is easily the most challenging topic in mobile robotics, as it demands domain of five distinct competencies: *perception* of the surrounding environment and its *representation* in a map of some sort, *localization* of the robot relative to it, *path planning* from current location to a specified destination, and finally *motion control* to execute the plan [1].

Generally speaking, path planning and motion control are relatively easy to perform, so long as it's assumed environment structure and robot position are known with certainty. Therefore, successful navigation is highly dependent on reliable perception, representation and localization, which are hard to achieve consistently. The fundamental problem is that those tasks depend on sensor readings, which by definition are only proxies to the actually relevant environment states, and imperfect ones at that. So it is that, for example, a laser scanner takes the time between a pulse is emitted and a reflection detected as a

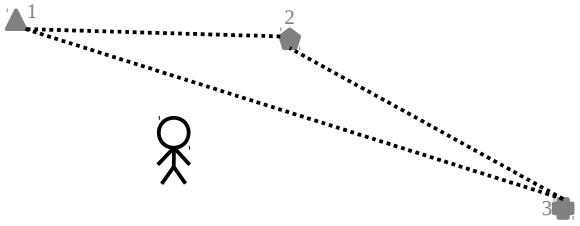


(a) Place cell (b) Grid cell

Figure 1.1: Activations (grey circles) of a single place cell (a) and a single grid cell (b) in a test subject at different locations within an enclosure. Whereas the place cell's activation is closely correlated to the animal's presence at a specific environment location, the grid cell is active at multiple, regularly-spaced locations.



(a) Egocentric frame



(b) Allocentric frame

Figure 1.2: Difference between egocentric and allocentric reference frames. In an egocentric frame, distances between landmarks are evaluated relative to the individual's own location; in an allocentric frame, landmark distances are evaluated relative to each other's location.

proxy for distances from obstacles – but not only are such measures subject to a degree of uncertainty, they may not always correlate at all with object distance, e.g., if a reflective object deflects laser pulses away from the sensor.

Responses to this fundamental sensor unreliability issue can be broadly divided between attempts at extracting more reliable *invariant representations* from raw input, and devising robust *inference methods* to derive environment state information from knowingly noisy readings. Most autonomous navigation methods will do some degree of both, while leaning more heavily on one or the other. Beyond this basic characterization, however, a wide variety of approaches exist, many based on completely different paradigms. For example, Simultaneous Localization and Mapping (SLAM) methods typically employ statistical inference methods to produce a probability distribution of current location over a Cartesian plane or 3D space [2], while appearance-based methods use invariant representations and similarity measures to represent current location as the similarity between present sensor inputs and collected records of previously visited places [3].

The large methodological differences among autonomous navigation methods may seem all but irreconcilable. This is unfortunate, since an overarching conceptual framework or design philosophy can provide important guidance to research and development programs, helping define problems and suggesting ways to find solutions. In such cases it can be useful to look into nature: living beings often face challenges not dissimilar to those addressed by engineering disciplines, and can therefore provide inspiration and insights into how they can be approached. In this case, studying how humans and other animals locate ourselves in space can help uncover general principles for describing and improving robot navigation.

1.2 Self-location

Humans constantly maintain a sense of our own location in space, updating it continuously as we move. This is also true for most mammals and many other animals. *Self-location* is the skill of relating sensory input to current position [4]. Psycho-physiologically, animal self-location is computed in two distinct brain regions. In the hippocampus, *place cells* fire in response to specific locations within an environment, encoding a seemingly egocentric (centered on the individual itself) representation of spatial location. Meanwhile, *grid cells* in the entorhinal cortex combine visual and self-moving cues to produce overall repeating activation patterns thought to encode allocentric (based on landmark locations relative to each other) spatial location [5]. Figure 1.1 shows example activation patterns for place and grid cells, and Figure 1.2 illustrates egocentric and allocentric reference frames.

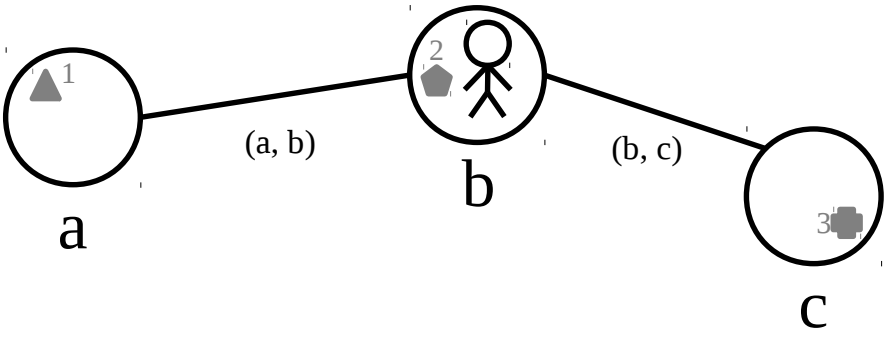


Figure 1.3: Common elements of a cognitive map. Space is divided in places (a, b, c) identified by landmarks (1, 2, 3) and connected through paths.

Spatial cognition is the field of study concerned with our ability to acquire, organize, and use spatial knowledge about environments. Current spatial cognition research asserts that biologic self-location is mainly supported by *cognitive maps*, mental representations of our physical environments, adapted for tasks such as place recognition and route planning [6]. Several cognitive map models exist [7], however most of them, in one way or another, incorporate *landmarks* as perceptual elements, *places* as physical locations characterized by particular landmarks, and *paths* as associations between places. Figure 1.3 illustrates the concepts.

Self-location seems to mostly rely on *visual cues* for sensory input, though it was also found to adapt well to changes in environment perception and even composition [8, 9]. Self-motion cues have also been implicated in the process [5]. These stimuli are employed to derive three kinds of knowledge useful for cognitive map construction and use:

- *Landmark knowledge* of the presence and configuration of particular features at each place [10];
- *Survey knowledge* of estimated shortest-distances between landmarks [11];
- *Route-road knowledge* of pathways between places [11].

Moreover, cognitive map construction is influenced by several heuristics that generally contribute to make representations more "regular" than reality – e.g., intersections such as street crossings are recorded as forming a 90^o angle more often than they really do [12, 13], and the relative positions of landmarks and place boundaries are distorted so as to better fit overall conceptual knowledge of the contexts in which they are located [14].

1.3 Difference Image Correspondence Hierarchy (DICH)

The Difference Image Correspondence Hierarchy (DICH) is an effort to solve the robot localization problem using biologic self-location as a template. It represents environments as collections of places arranged across paths and characterized by landmarks detected from visual input, while current location is inferred both from visual input cues (by searching for signals representative of landmarks) and a history of previous location inferences.

DICH is designed to enable localization in the context of Visual Teach and Repeat (VT&R) navigation. It assumes a differential drive mobile robot equipped with a single monocular camera. DICH builds upon previous work on template matching [15] and experiments in mobile robotics [16, 17], assimilating additional ideas from spatial cognition, embodied cognition [18] and computational neuroscience [19].

DICH takes *difference images* as its basic percept: these encode changes to visual input over short time intervals, an approach inspired in how our senses are dependent on change to work properly – for example, if our eyes are kept fixated onto a static image, vision starts to fade out [20]. Difference images are further processed by searching for fixation points, and then extracting Regions-of-Interest (ROI's) from them. When images have to be compared, these ROI's are searched over each other using cross-correlation, a biologically-plausible operation. Finally, models of working memory are used for keeping track of and detecting trends over match results, which enables the system to reliably compare stored and real-time inputs to estimate current location. These operations are detailed in Chapter 3.

DICH has a close relationship to the field of Biologically Inspired Cognitive Architecture (BICA's). BICA's combine results from cognitive science, computer science and neuroscience to both advance the understanding of human and animal cognition, and provide novel solutions to complex computational problems [21]. They exploit the observation that living systems can be used as a template to help define vague problems (e.g., "intelligence") as well as suggesting possible solutions and validation metrics. They can also provide inspiration for architectures and algorithms not necessarily targeted at "biological" problems. The opposite is also possible – results from computer science and robotics being used to assess theories on biologic cognition, suggesting new models or avenues of investigation.

Human vision and spatial cognition are both widely studied fields with a wealth of knowledge to offer, therefore a case can be made for the development of a biologically inspired vision-based autonomous navigation system. In fact, BICA research has repeatedly approached the problem of self-location; however, it tends to be taken as a test case to validate proposed cognitive models, rather than as an objective to be achieved in its own right. Methods are not developed to match task or environment requirements, but the opposite: given a BICA implementation, a task and (usually simulated) environment are selected that

demonstrate its capacities while abstracting away its limitations. Consequently, there is no convergence towards a common approach dictated by what works best under recurring real-world conditions, as is the case in mobile robotics. Instead, competing interpretations of psychophysiological data lead to diverging solutions. Such efforts include:

- The Enactive Cognitive Architecture (ECA) [22] can learn complex behaviors by chaining together (action, result) pairs. This is demonstrated by implementing an agent that learns its way across a very simplified 2D world where movement is atomic (the agent advances in "cell" units and turns in increments of 90^o) and perception is binary (the agent can either see an obstacle ahead or not). The agent moves of its own volition, without an externally specified destination;

- The Learning Intelligent Distribution Agent (LIDA) [23] is based on the Global Workspace Theory (GWT) [24] of functional consciousness in brains. Experiments demonstrating its ability for self-location were performed in an elaborate 3D environment, however the architecture was allowed to acquire data on landmarks (e.g., their distances from the agent) directly, rather than having to estimate it from visual observations;

- The Soar/SVS architecture [25] combines symbolic reasoning and prototype-based visual filters to infer environment state from computer-generated images depicting a simulated environment. This at least enforces a proper separation between environment and agent, as the later must parse its perceptions in order to infer the former's state, but the idealized environments are still far removed from the complexity of the real world.

In contrast, mobile robotics regards self-location as a requirement for implementing target applications such as autonomous navigation. Therefore, research focus on how the problem can be best solved in real-world conditions. These approaches are not irreconcilable, though – in fact, BICA research could gain novel insights from approaching the practical challenges inherent to the real world, while a more architectural and model-driven approach could open the way for novel solutions to autonomous navigation problems.

DICH, therefore, also constitutes an effort to apply the principles of BICA development to real-world requirements. While it draws extensively from results in neuroscience and cognition to build a biologically plausible self-location model, it has been developed from the beginning to operate in physical robots and environments, relies exclusively on visual data, and implements a learning model accommodating of goal-directed training and operation. It is expected that requiring BICA's to cope with real-world constraints will promote a convergence towards a set of best practices and methods, as seen in mobile robotics.

1.4 Research objectives

Research objectives can be roughly divided across three axes: technical, theoretical and social.

From a technical standpoint, my objective was to develop a visual navigation system that was robust while comparatively low-cost in terms of hardware and complexity. For that reason I decided to focus on monocular vision and similarity-based input analysis over time, which can be achieved with minimal resources and setup. I also wanted an architecture that could be easily adapted to run on multiprocessing platforms, since parallel programming is an ongoing challenge in computer science, and algorithms that can work efficiently in highly-parallel hardware are widely sought after.

Theoretically, I sought to apply concepts from neuroscience and cognition to create an autonomous navigation architecture that is perceptual rather than metric. This is commonplace among appearance-based navigation methods; yet, as will be discussed in Chapter 2, DICH application of time-consistent estimates to monocular visual input occupies a very restricted field in that field.

Finally, the availability of an intuitive, low-cost autonomous navigation system could have important social ramifications, as non-specialist users could find new and interesting applications for it. Autonomous cars are currently on the cusp of entering the consumer market, and are expected to have a tremendous influence in the transportation industry, enabling a variety of new businesses and upsetting the old. However these will in all likelihood be closed systems, whose owners will be discouraged (if not outright forbidden) to tinker with. DICH opens the possibility of offering consumers an open, low-cost autonomous navigation solution that would allow free experimentation.

Chapter 2

Related Work

This chapter discusses other works in mobile robotics with similar objectives and/or approaches. It starts with an overview of autonomous navigation approaches, in particular the SLAM paradigm. The following section makes the case for appearance-based navigation as a complement to SLAM, as well as an alternative in its own right. Visual Teach and Repeat (VT&R) is then proposed as the ideal scenario for appearance-based navigation, and past approaches to appearance-based VT&R navigation are discussed. Finally, the contributions of DICH relative to other navigation methods are discussed.

2.1 Autonomous navigation

Arguably the central topic in mobile robotics [1], *autonomous navigation* is, in its simplest formulation, the problem of getting a robot to drive from an origin point to a destination without human intervention. This obviously incurs a series of attendant problems, from purely physical constraints of autonomy or terrain traversability, to higher-level cognitive issues such as path planning, environment representation and sensor data interpretation.

The broad category of *navigation methods* refers to these later concerns: given a robot equipped with appropriate sensors and powertrain, such that the ability to physically perceive and move across the environment is assured, how can it be driven between specified and destination points without further human intervention. Navigation methods can be roughly divided into *map-based* and *mapless*, according to whether they depend on globally consistent environment representations. Map-based methods can be further divided into *metric*, which represent the environment geometrically relative to a predefined coordinate system, and *topological*, which break down the environment as a set of discrete nodes connected by relations of reachability [26].

Most navigation methods rely at some level on the existence of *landmarks*, enduring environment features that produce consistent, identifiable patterns in the sensor stream. To estimate robot location, methods assert the presence

and / or location of particular landmarks by cataloging and searching such patterns in sensor readings; this usually involves several preprocessing steps to attenuate the effects of noise or otherwise remove unimportant data. Different sensors and navigation methods will demand diverse landscape selection and recognition algorithms.

Simultaneous Localization and Mapping (SLAM) is a popular autonomous navigation paradigm. In the typical SLAM scenario, a robot, initially without any knowledge of its surroundings, must explore and construct a metric map-based representation of the environment, while at the same time keeping track of its location within it. SLAM methods commonly rely on laser or other range-finder sensors for input, and employ probabilistic approaches such as particle filters or the Extended Kalman Filter (EKF) to relate egocentric landmark location to robot movement across time. However, the operation range of filter-based SLAM systems is limited by the need to continuously keep track of all landmarks to ensure estimate convergence, causing computation requirements to grow steadily as the traversed area increased, and eventually overwhelming the system [27]. Moreover, as environment scale grows, range-finder sensor readings become increasingly sparse and unreliable.

2.2 Appearance-based navigation

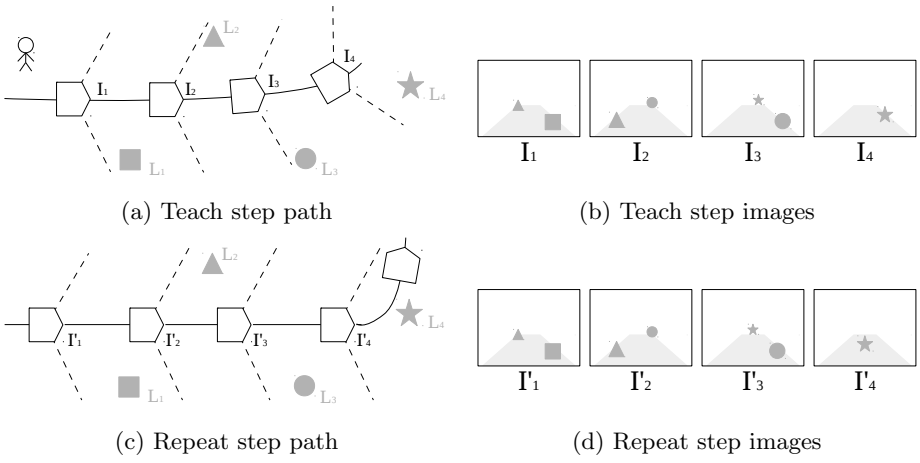
The limitations of metric map-based navigation systems have motivated work in topological methods. In particular, *appearance-based* navigation methods avoid the limitations of range-finder sensors by employing cameras, and use image data to construct place representations according to an *appearance model*. Environments are represented as collections of places, and current location is determined in terms of appearance similarity between known places and current input, which incidentally is closer to how we humans navigate our surroundings.

Several appearance-based methods have been advanced as localization modules to probabilistic SLAM backends. FAB-MAP and its variants [28] effectively lifted the operation range restriction of filter-based approaches, but have been found vulnerable to large appearance variations, as undergone by outside environments across seasons or weather conditions. Further research has addressed this limitation, either by employing image-based appearance models that do away with features, as in SeqSLAM [29], or by compensating for their weaknesses in some way – e.g., the plastic map model [30] repeatedly records features for the same environments over repeated trips, thus keeping track of appearance changes over time.

While effective under their prescribed scenarios, these hybrid appearance / SLAM methods are computationally expensive, and suffer from accuracy shortcomings that make them inadequate for some applications, e.g., route-following [31]. Moreover, they often demand external modules such as visual odometry [30] (or alternatively, assume the robot was moving at constant speed for the duration of data records [29]) and expensive omnidirectional [28] or stereo [30] cameras.

Table 2.1: Summary of recent appearance-based Visual Teach & Repeat navigation methods, characterized by sensors, test environments, and environment variations allowed between teach and repeat steps.

Method	**Sensors**	**Environments**		
		Types	**Variations**	
			Lights	**Moving elements**
Monocular depth estimation [31]	Monocular camera	Indoors, Outdoors	No	No
Homography estimation [32]	Monocular camera	Indoors, Outdoors	Yes	No
Monte Carlo localization [33]	Monocular camera	Indoors	No	No
Hybrid feature / segmentation path finding [34]	Monocular camera	Indoors, Outdoors	No	Yes
SURFnav [35]	360^o camera	Indoors	No	No
Scale-based visual servoing [36]	360^o camera	Indoors	No	Yes
Min-warping [37]	360^o camera	Indoors	Yes	No
Depth-feature learning [38]	Stereo camera	Indoors	No	No
Multi-stereo [39]	Pair of stereo cameras	Outdoors	Yes	No
Color-constant images [40]	Stereo camera	Outdoors	Yes	No



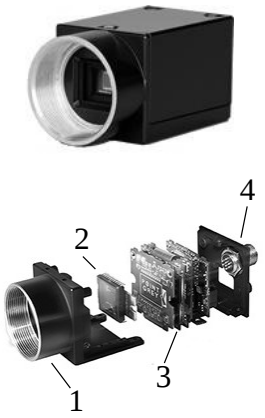
(a) Teach step path

(b) Teach step images

(c) Repeat step path

(d) Repeat step images

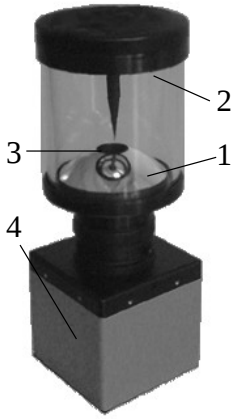
Figure 2.1: Visual Teach & Repeat (VT&R) navigation. In the teach step (a), a robot is driven over a route, collecting visual records (b) of landmarks $[\mathrm{L}_1, \ldots, \mathrm{L}_4]$ (field of view indicated by dashed lines). In the repeat step (c), the robot retraces the route autonomously, orienting itself by the differences between teach (b) and repeat (d) image records.



(a) Monocular camera

(b) Monocular image

Figure 2.2: Monocular camera (a) and example image (b). A digital monocular camera is composed of optic parts (1) for focusing light on an imaging sensor (2) which an electronic system (3) uses to collect images and send them across an external connection (4) to a processing system.

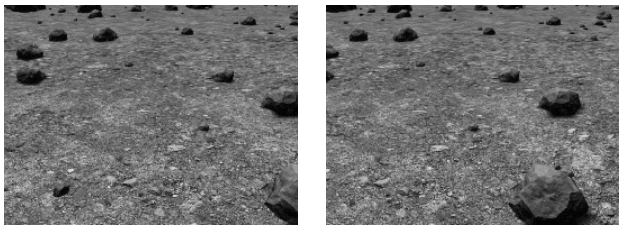


(a) 360^o camera

(b) 360^o image

Figure 2.3: Omnidirectional camera (a) and example image (b). A catadioptric omnidirectional camera is composed of a base convex mirror (1) that projects light from all around the system into a top concave mirror (2), sensing it across an aperture (3) and into a base-mounted monocular camera (4).

(a) Stereo camera



(b) Stereo image

Figure 2.4: Stereo camera (a) and example image (b). Stereo cameras are composed of two monocular cameras (1, 2) sharing a common field of vision and relaying images to the same control unit (3). They can be implemented as an integrated setup (4) or by mounting a pair of standalone cameras on a rig (5).

2.3 Visual Teach & Repeat

Appearance-based methods can also be valuable on their own. They are perfectly suitable for *Visual Teach and Repeat* (VT&R) navigation, where a robot is first led through a route by a guide (the teach step), and must later autonomously retrace the original path (the repeat step), orienting itself by sensor readings gathered during the guided stage [41]. See Figure 2.1 for a graphical example.

A variety of appearance-based methods have been proposed over the years. Table 2.1 provides a summary of recent contributions. They can be roughly classified in terms of employed sensors, the perception model used to process and reason about inputs, and environment conditions evaluated in experiments.

2.3.1 Sensors

Cameras are the main sensor solution in any VT&R system. According to their optics and output, cameras can be categorized as monocular, omnidirectional and stereo.

Monocular cameras are composed of optics elements focusing light on an imaging sensor, controlled by electronics that collect and relay images across a connection for processing (Fig. 2.2). They remain a popular choice [31, 32, 33, 34] due to their size, cost and simplicity. Moreover, monocular vision ap-

proaches can be promptly applied to the many already existing robots equipped with monocular cameras (typically for teleoperation purposes), making them appealing from a practical standpoint [31].

Omnidirectional cameras are characterized by a 360^o field of view in the horizontal direction. For this reason they are also occasionally called "360^o cameras". They are typically based on a *catadioptric* setup [42] that uses mirrors and lenses to direct light from all around the system into an upward-looking monocular camera (Fig. 2.3). Because views from all directions are recorded at once, omnidirectional VT&R methods [35, 36, 37] can potentially navigate between any two arbitrary points within a known environment, not just along a recorded route. In practice this is appealing in indoors environments, but not so much outdoors, where traffic rules and conventions usually restrict movement to the length of streets and sidewalks.

Finally, stereo cameras are composed of two monocular units sharing a common field of view, relaying images to a single controller (Fig. 2.4). Image disparities between cameras enable scene depth inference. Accordingly, VT&R methods based in stereo input [38, 39, 40] can make use of depth values together with image and feature data for recognizing places.

2.3.2 Perception models

Different sensor solutions display particular strengths and weaknesses, motivating distinct input processing models and methods to correlate readings to robot location.

Monocular VT&R methods have to address the fact that, on themselves, monocular images contain much data but precious little spatial information. One way around this limitation is to infer spatial information from image data; this may involve making assumptions about the spatial configuration of the system and/or the environment. For example, monocular depth estimation [31] can be performed to compute 3D coordinates for ground features recorded by a calibrated monocular camera, provided all features are assumed to lie on a ground plane local to the robot itself. The combination of 3D coordinates and features can then be used to construct a locally-consistent map to support navigation. An alternative is to do away with spatial estimates entirely – for the purposes of route-following, simple feature matching may be sufficient. This is the approach taken by the homography estimation method [32], which uses robust feature matching to relate visual inputs over time and construct a topological map of the environment. Monte Carlo VT&R [33] takes a similar approach, employing a particle filter to compensate for imprecisions of feature matching. Pure-appearance methods may also benefit from environment assumptions, however: hybrid feature / segmentation path finding [34] uses image segmentation to detect traversable paths where such structure can be found, complementing feature-matching procedures and increasing performance in outdoor environments.

Omnidirectional VT&R methods are often based on the concept of a *homing vector* indicating the direction, and possibly the distance, between current location and destination. Navigation is then reduced to the problem of computing the homing vector from current input and records previously taken at the destination. This can be done in a number of ways. SURFnav [35] collects ans tracks SURF features over time to infer distance and direction of recorded landmarks. This can be used to produce a series of path-correcting commands to make the robot converge towards a destination. A similar approach is used in scale-based visual servoing [36], where the scale component of SIFT descriptors is used to guide a control module. In contrast, min-warping [37] takes a more holistic approach to inputs, matching whole images in search of a discrete set of "warping" operations (translations and scalings) that will minimize mutual difference, and can be related to robot movements.

Stereo VT&R methods usually combine depth estimates and feature matching looking to compensate for each other's weaknesses. Depth-feature learning [38] uses stereo matching to produce feature points combining 3D coordinates computed from visual odometry and a 64-dimensional SURF-based feature vector. Features are collected in a topological map that is only locally metric-consistent, but nevertheless enables sufficiently correct localization for navigation tasks. Outdoor performance is limited, though, since features are weak against variations in appearance, especially general variations such as those caused by seasonal changes. Multi-stereo [39] attempts to mitigate this issue by adding a second stereo camera to the system: with a larger field of view and increased number of detected features, the probability of successful matching is increased. Color-constant images can be used to solve this problem the other way around, ensuring stable inputs to the stereo vision pipeline and thus consistent features [40]. However the algorithm that produces color-constant images needs to be trained, which may require several trips over target environments.

2.3.3 Environments

The environments robots travel along can be categorized as "indoors" and "outdoors". Indoors environments are human-made and include halls, offices, rooms, corridors and warehouses. They are generally characterized by a regular visual structure with repetitive elements, an abundance of straight edges meeting at right angles, and illumination that is either constant or varies among a couple of highly recurrent states – e.g., office lights might sometimes be off, but otherwise their positions and intensities remain the same.

In contrast, outdoors environments are dominated by changing illumination patterns, due to seasonal changes, weather, and the daily cycle of light and dark. Scenes may still contain the kind of structural regularities found indoors, especially in urban areas, but often these will at least share space with the organic shapes of plants and geographical formations. Outdoors environments include streets, sidewalks, parks, trails and wilderness regions.

Both indoors and outdoors environments are also subject to change. This can be gradual, as when ambient brightness slowly decreases in response to sunny weather turning cloudy; sudden but unwitnessed, as when a piece of furniture is added to a room between visits; or dynamic, caused by the presence of moving entities such as vehicles or people. Environment changes have important consequences for VT&R methods: landmarks may be removed between trips, become unrecognizable due to light changes, or be occluded by passing pedestrians. Likewise, transient environment features may be unwittingly selected as landmarks.

VT&R methods will sometimes explicitly take illumination variations into account [32, 37, 40, 39], especially when intended to work outdoors, but few studies give close consideration to the effects of moving elements [34, 36]. As a result, their suitability for use along people and other vehicles remains largely unverified.

2.4 DICH and VT&R

The Difference Image Correspondence Hierarchy (DICH) is a monocular appearance-based VT&R method for indoors and outdoors environments, resilient to illumination changes and the presence of moving elements. It can perform localization and route-following, and can deal with the kidnapped robot problem so long as the robot is left somewhere along a known path. As Table 2.1 and the previous sections show, it occupies a rare niche in the VT&R literature, where all these features are seldom seen together in a single system.

DICH is image-based, avoiding weaknesses related to visual features. It is also virtually unique among VT&R systems in its explicit use of record history to achieve estimation consistency – ambiguous and incorrect estimates are weeded out by selecting sequences of estimates consistent over time. In most systems this is a consequence from other constraints (such as the probabilistic prediction-updating of the Monte Carlo approach [33]), but in DICH it is stated explicitly from the beginning, and is at the heart of most method parts.

Therefore, DICH constitutes a comprehensive VT&R navigation solution for robots complying to the most general specifications. The next chapters will expand on the DICH method and its implementation, followed by experiments demonstrating its features.

Chapter 3

Difference Image Correspondence Hierarchy (DICH)

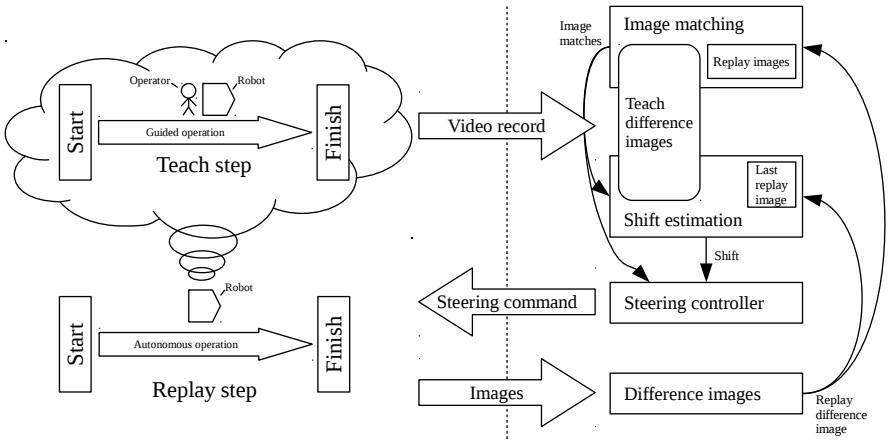
The DICH architecture was designed to work with a differential drive robot [1] equipped with a single front-mounted camera, under the VT&R navigation scenario. As summarized in Figure 3.1, operation is divided in a manually driven teach step, and an autonomous repeat step.

During the teach step the robot is driven over a route, collecting a video record as it goes. This record is used to build a visual account of the route, to be stored in memory as a list of *teach difference images*. During the repeat step, camera inputs are converted to *repeat difference images* on the fly, and paired to teach images according to similarity. In this way each repeat difference image is assigned a place along the route, in terms of positions in the teach list. Teach / repeat difference image pairs are also compared for *shift*, the apparent sliding of visual features on one image relative to the other. Shift is used to infer whether the robot is drifting away from the route. Both image pairing and shift information are then forwarded to a steering module, which drives the robot according to perceived drift and position along the teach route.

This chapter describes each of the above steps in detail. It opens with a description of difference images, the architecture's main percept. Next difference image pairing is explained, including a solution to the kidnapped robot problem. Shift estimation follows, and the chapter closes with a description of the steering model. Table 3.1 below explains the notation used in these sections; see the appendices for more details on the underlying theory.

Table 3.1: List of notational conventions used in DICH. Refer to the appendices for more details.

Notation	**Description**
$\mathbf{a}^n$	List or vector $\mathbf{a}$ of n elements.
$\lvert\mathbf{a}\rvert$	Number of elements in a list or vector $\mathbf{a}$. For $\mathbf{a}^n$, $\lvert\mathbf{a}\rvert = n$.
$\mathbf{0}^n$	A vector of dimension n and all cells equal to zero.
$\mathbf{a} +\!\!+ \mathbf{b}$	List or vector concatenation. For $\mathbf{a}^m$ and $\mathbf{b}^n$, $\mathbf{a} +\!\!+ \mathbf{b} = \mathbf{c}^{m+n}$ such that $\mathbf{c} = [a_0, \ldots, a_{m-1}, b_0, \ldots, b_{n-1}]$.
$\mathbf{A}^{m\times n}$	Matrix $\mathbf{A}$ of m rows and n columns. Usually dimensions are shown only the first time a matrix is mentioned.
$\mathbf{A}[i,j]$ a_{ij} $a_{i,j}$	Element at the i^{th} row and j^{th} column of matrix $\mathbf{A}$. Indexes start at 0 and count top-to-bottom, left-to-right.
$\mathbf{A}[i,:]$	i^{th} row of matrix $\mathbf{A}$.
$\mathbf{A}[:,j]$	j^{th} column of matrix $\mathbf{A}$.
$\mathbf{A}[i_0{:}i_n, j_0{:}j_n]$	Rectangular section of matrix $\mathbf{A}$, from top-left element $\mathbf{A}[i_0, j_0]$ to bottom-right element $\mathbf{A}[i_n - 1, j_n - 1]$.
$\sum \mathbf{A}$	Sum of all elements in matrix $\mathbf{A}$.
$\mathbf{A} \circ \mathbf{B}$	Hadamard (element-wise) product of matrices $\mathbf{A}$ and $\mathbf{B}$. For $\mathbf{A}^{m\times n}$ and $\mathbf{B}^{m\times n}$, $\mathbf{A} \circ \mathbf{B} = \mathbf{C}^{m\times n}$ such that $c_{ij} = a_{ij}b_{ij}$.
$\mathbf{A}^{\circ n}$	Hadamard (element-wise) power of matrix $\mathbf{A}$. For $\mathbf{A}^{m\times n}$, $\mathbf{A}^{\circ n} = \mathbf{C}^{m\times n}$ such that $c_{ij} = a_{ij}^n$.
$f^\circ(\mathbf{A})$	Element-wise application of function f to matrix $\mathbf{A}$. For $\mathbf{A}^{m\times n}$, $f^\circ(\mathbf{A}) = \mathbf{C}^{m\times n}$ such that $c_{ij} = f(a_{ij})$.
$\mathcal{N}_{cc}(\mathbf{A}, \mathbf{B})$	Normalized cross-correlation between $\mathbf{A}$ and $\mathbf{B}$. For $\mathbf{A}^{m_\mathbf{A}\times n_\mathbf{A}}$ and $\mathbf{B}^{m_\mathbf{B}\times n_\mathbf{B}}$ such that $m_\mathbf{A} \leq m_\mathbf{B}$ and $n_\mathbf{A} \leq n_\mathbf{B}$, $\mathcal{N}_{cc}(\mathbf{A}, \mathbf{B}) = \mathbf{C}^{(m_\mathbf{B}-m_\mathbf{A}+1)\times(n_\mathbf{B}-n_\mathbf{A}+1)}$ such that c_{ij} is the similarity between $\mathbf{A}$ and $\mathbf{B}[i{:}i+m_A, j{:}j+n_A]$.



(a) VT&R navigation (b) DICH architecture

Figure 3.1: Difference Image Correspondence Hierarchy (DICH) working environment and abstract diagram. (a) Visual Teach and Repeat scenario. A robot is first guided through a route (the teach step); later, after being brought back to the starting point, it must retrace the original path autonomously (the repeat step), using data collected during the teach step as a guide. (b) The DICH architecture. During the teach step, the robot collects a video record of the route, which is used to generate a sequence of images stored in long-term memory. During the repeat step these are compared to live visual input, to estimate the robot's position along the route and possible drift from it. This information is used to steer the robot as necessary.

3.1 Difference images

A recurring problem in computer vision is the construction of *invariant representations* – input transformations that consistently correlate to parameters relevant for a certain application, while being unaffected by other factors. For example, a visual navigation system could employ a representation that correlated well to the presence of landmarks, while being invariant to other objects or changes in brightness and viewpoint. A system's choice of invariant representation reflects its a-priori assumptions on environment nature, correlations between stimuli and world states, and acceptable trade-offs on percept invariance, distinctiveness and computation complexity [43].

In a dynamic context such as autonomous navigation, invariant representations can be designed by first looking not into immediate inputs themselves, but rather into how they change over time. This approach is supported by studies on biologic sensory organs, which have been repeatedly shown to require an ele-

ment of active change to work properly – e.g., visual perception fades over if eyes are fixated on a static scene and saccadic movements are suppressed [20]. Biologic senses are under strong pressure to deliver the most reliable information in the shortest time possible, having evolved to support critical animal behaviors such as predator evasion. Therefore, the reasoning goes, effective digital sensing could be achieved by borrowing biologic sensory strategies.

Dynamic Vision Sensors (DVS) apply this idea by measuring luminance differences across the visual field, and reporting the coordinates of changes that crossed a threshold. The resulting bursts of Address-Event Representations (AER) can be pooled to generate scene shape representations invariant to color, brightness and surface gradients [44]. DVS are often implemented in hardware, as banks of digital circuits on a CMOS chip [45]. It is however perfectly possible to reproduce its dynamics in software, using camera frames as input.

DICH employs *difference images* as its basic percept, invariant representations computed in a manner similar to the AER's produced by DVS devices. First, immediate differences between luminance images $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$ can be detected by computing relative difference between individual pixels, and then thresholding results:

$$\mathbf{K}_t = \mathrm{H}_\delta\left(\log^\circ\left(\mathbf{I}_t \circ \mathbf{I}_{t-1}^{\circ -1}\right)\right) \tag{3.1}$$

Where $\mathrm{H}_\delta()$ is defined as:

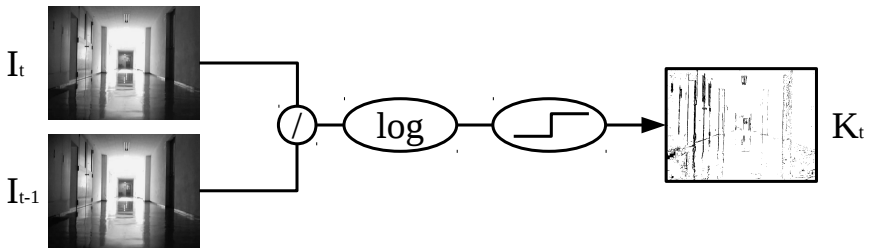
$$\mathrm{H}_\delta(\mathbf{A}) = \left[h_{ij} = \begin{cases} 1 \text{ if } a_{ij} > \delta \\ 0 \text{ otherwise} \end{cases}\right] \tag{3.2}$$

The use of logarithm ratio in formula 3.2 turns threshold δ into a *percentage*, making values easier to interpret – reasoning in terms of "δ% difference" is more intuitive than working with absolute luminance values. The thresholding operation itself produces a "sparse" representation with a higher signal-to-noise ratio. Still, experiments show that individual thresholded maps $\mathbf{K}_t$ lack consistence – large variations occur between successive maps at the local level, even if the general pattern remains similar. This can be mitigated by summing together several maps over a given range:

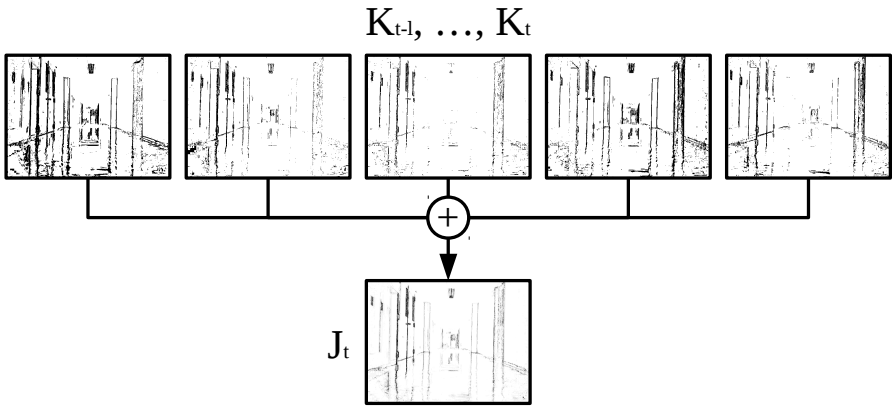
$$\mathbf{J}_t = \sum_{l=t-\tau}^{t} \mathbf{K}_l \tag{3.3}$$

Where $\mathbf{J}_t$ is a difference image computed over τ maps of threshold δ. Figure 3.2 illustrates the process.

Given appropriate values for parameters δ and τ, each difference image will contain regions of high signal concentration (caused by discontinuities such as edges) surrounded by empty areas (due to smooth object surfaces). Absent of saturation artifacts, difference images are largely invariant to ambient brightness, providing a degree of normalization across illumination conditions. Closeby objects will generate larger difference counts than those farther away, allowing a glimpse into the scene's structure. The amount of change will vary



(a) Thresholded map



(b) Difference image

Figure 3.2: Difference image computation. (a) Luminance differences between images are detected by computing relative differences, then thresholding the result. (b) Thresholded maps are summed to attenuate noise and improve consistency, producing a single difference image. Darker shades of gray indicate larger change counts.

depending on whether the robot is moving, which can be used as a self-motion cue. Finally, each difference image implicitly denotes a spatial range, delimited by the viewpoints from which the original images were captured.

3.2 Difference image pairing

As mentioned previously, the video record collected during the teach step is transformed into a sequence of difference images $\mathbf{J} = [\mathbf{J}_0, \ldots, \mathbf{J}_m]$, stored in memory. Each difference image implicitly denotes a place along the route; therefore, during the repeat step, it's possible to infer the robot's position by computing a difference image $\mathbf{J}'_j$ from current visual input and searching for its most similar "pair" $\mathbf{J}_i$ among teach difference images. Assuming the pairing is successful, the robot is expected to be in the vicinity of the viewpoints from which the images used to compute $\mathbf{J}_i$ were captured.

Difference image pairing in DICH works by extracting *Regions-of-Interest* from repeat difference images, computing *similarity* to teach images in terms of such regions, and finally identifying *similarity trends* among teach / repeat image pairs. These steps are detailed below.

3.2.1 Regions-of-Interest

When animals look at their surroundings, their eyes don't take everything at once; rather, they dart among *interest points* (e.g., corners or edges) to apprehend a set of visual *Regions-Of-Interest* (ROI's), which seem to provide enough information for effective recognition. This behavior is modeled in computer vision applications by Image Processing Algorithms (IPA's), which are used to select small patches called algorithmic Regions-Of-Interest (aROI's) from input images [46]. Image matching can be performed for aROI's in place of whole images, reducing resource requirements and increasing generality.

DICH extracts aROI's from difference images in the following manner. Let $\mathbf{J}'_j$ be the j^{th} difference image computed over a currently ongoing repeat step trip. For a *padding* α and $l = 2\alpha + 1$, the *salience map* $\mathbf{\Sigma}'_j$ is defined as:

$$\mathbf{\Sigma}'_j = \left[\sigma_{uv} = \sum \mathbf{J}'_j[u : u + l, v : v + l]\right] \tag{3.4}$$

Where $\mathbf{\Sigma}'_j[u, v]$ is the salience of an *interest point* of coordinates $(u+\alpha, v+\alpha)$, computed as the sum of $\mathbf{J}'_j$ values lying inside an aROI of top-left coordinates (u, v) and side l. Figure 3.3 illustrates the concept. It should be clear that $\mathbf{\Sigma}'_j$ can be quickly computed from the integral image of $\mathbf{J}'_j$ (see Appendix B for details).

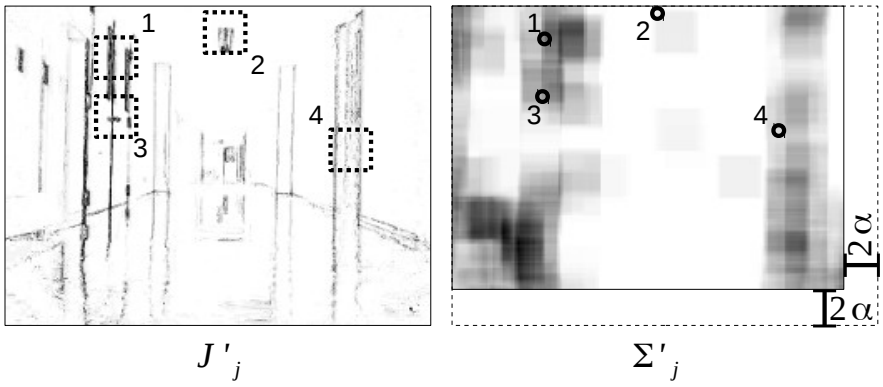


Figure 3.3: Difference image (left) and corresponding salience map (right). The dashed box around Σ'_j indicates the gap (of length 2α) between the boundaries of J'_j and its own. Dashed squares on J'_j delimit Regions-of-Interest (ROI's); corresponding salience values are indicated on Σ'_j by small circles. Darker shades of gray indicate higher values.

A set of disjoint aROI's can be selected from $\mathbf{\Sigma}'_j$ by iterating over the following steps:

1. Make $\mathbf{p}$ a list of all (u, v) for which $\mathbf{\Sigma}'_j[u, v] > 0$, sorted in decreasing salience order;
2. If $|\mathbf{p}| = 0$ then terminate, otherwise make $(u, v) = \mathbf{p}[0]$ and $\mathbf{p} = \mathbf{p}[1{:}]$ (i.e., remove the first item from $\mathbf{p}$);
3. If $\mathbf{\Sigma}'_j[u, v] > 0$ then record aROI (u, v, l);
4. Make $\mathbf{\Sigma}'_j[u - l : u + l, v - l : v + l] = 0$, then return to the second step.

In practice it may be preferable to use a separate matrix to keep track of added / discarded points instead of changing $\mathbf{\Sigma}'_j$ itself. Moreover, since difference image saliences are integers with a known upper bound of τl^2, image coordinates can be sorted in linear time using a distribution algorithm such as bucket sort [47], which may require $\mathbf{p}$ to be a data structure other than a flat list. The overall idea, however, remains the same. Algorithm 1 works out the details of the procedure.

Algorithm 1 DICH's aROI selection algorithm. Given a difference image and a padding value, a list of disjoint aROI's is returned. The algorithm runs in linear time as a function of the number of non-zero difference image pixels.

function AROIs($\mathbf{J}'^{m_J \times n_J}_j, \alpha, \tau$)
  $l \leftarrow 2\alpha + 1$
  $m \leftarrow m_J - l + 1$
  $n \leftarrow n_J - l + 1$
  $\mathbf{\Sigma}'^{m \times n}_j \leftarrow \left[\sigma_{uv} = \sum \mathbf{J}'_j[u : u + l, v : v + l]\right]$
  $\mathbf{p} \leftarrow [p_k = [] \mid 0 \leq k < \tau l^2]$ ▷ List of buckets indexed by salience
  for all $(u, v) \mid 0 \leq u < m\ ,\ 0 \leq v < n$ **do**
    $\sigma \leftarrow \mathbf{\Sigma}'_j[u, v]$
    if $\sigma > 0$ **then**
      $k \leftarrow \tau l^2 - \sigma$ ▷ Place more salient points at earlier buckets
      $\mathbf{p}[k] \leftarrow \mathbf{p}[k]\ +\!\!+\ [(u, v)]$ ▷ Append coordinates (u, v) to list $\mathbf{p}[k]$
    end if
  end for
  $\mathbf{r} \leftarrow []$ ▷ List of returned aROI's
  $\mathbf{E}^{m \times n} \leftarrow \mathbf{0}$ ▷ Matrix of selected points
  for all $p_k \in \mathbf{p}$ **do**
    for all $(u, v) \in p_k$ **do**
      if $\mathbf{E}[u, v] = 0$ **then**
        $\mathbf{r} \leftarrow \mathbf{r}\ +\!\!+\ [(u, v, l)]$
        $\mathbf{E}[u - l : u + l, v - l : v + l] \leftarrow 1$ ▷ Select (u, v) and all neighbors
      end if
    end for
  end for
  return r
end function

3.2.2 Difference image similarity

If $\mathbf{J}_i$ and $\mathbf{J}'_j$ are spatially related, then both images should contain a number of similar visual elements produced by common landmarks. This can be checked by comparing aROI's extracted from $\mathbf{J}'_j$ to regions of $\mathbf{J}_i$: a high proportion of visual matches would suggest both images come from the same place, or at least somewhere close by. A successful recognition would remain possible even if some aROI's from $\mathbf{J}'_j$ or their corresponding regions in $\mathbf{J}_i$ were changed, as aROI's capturing common visual elements would compensate for the failures of the others.

A similarity measure based on aROI's can be defined as follows. Let $\mathbf{r}_j$ be a list of aROI's extracted from $\mathbf{J}'_j$ according to Algorithm 1. For each $r_{j,k} = (u, v, l)$, the contents of $\mathbf{J}'_j$ within $r_{j,k}$ are compared to those of $\mathbf{J}_i$ within a *neighborhood* $(u - \beta, v - \beta, l + 2\beta)$. This is done by defining the following ranges:

$$\rho_{j,k} = (u \ : \ u + l \ , \ v \ : \ v + l) \tag{3.5}$$

$$\phi_{j,k} = (u - \beta \ : \ u + l + 2\beta \ , \ v - \beta \ : \ v + l + 2\beta) \tag{3.6}$$

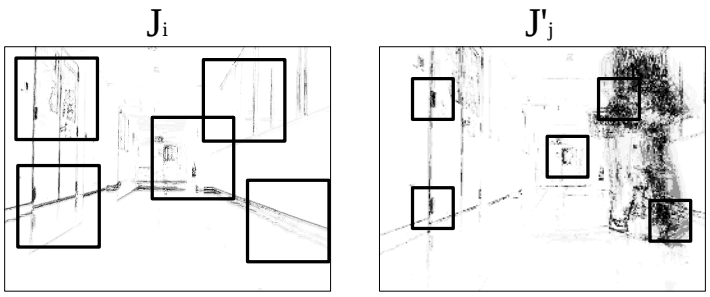
And then finding the maximum cross-correlation response between the contents of $\rho_{j,k}$ in $\mathbf{J}'_j$ and those of $\phi_{j,k}$ in $\mathbf{J}_i$:

$$s(i, j, k) = \max \mathcal{N}_{cc}(\mathbf{J}'_j[\rho_{j,k}], \mathbf{J}_i[\phi_{j,k}]) \tag{3.7}$$

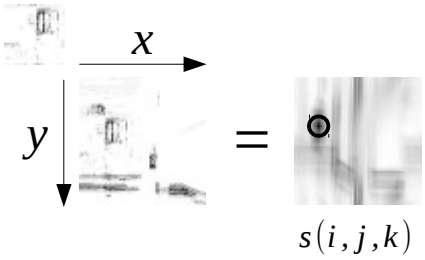
The similarity $s(i, j, k)$ between individual aROI's and neighborhoods will be influenced by the visual elements contained in both. If $\mathbf{J}_i$ and $\mathbf{J}'_j$ represent the same or close places, similar contents will be likely, leading to higher similarity values. However even then it may not be the case: if the place was changed between trips (e.g., furniture or other objects were added or removed), pedestrians and other moving entities were present in either step, or viewpoint direction diverged to a visible degree, aROI's and neighborhoods may differ to the point of irreconcilability. Therefore it's important that image similarity be defined as the sum of aROI's similarities:

$$\mathrm{S}(i, j) = \sum_k s(i, j, k) \tag{3.8}$$

Figure 3.4 illustrates the process.



(a) aROI's and neighborhoods



(b) Normalized cross-correlation

Figure 3.4: Computing similarity between an aROI and a teach difference image. (a) A number of aROI's are selected from a repeat difference image J'_j; for each patch, a larger neighborhood is extracted from teach repeat image J_i (only a few aROI's and corresponding neighborhoods are drawn for clarity). (b) For each (patch, neighborhood) pair, their normalized cross-correlation is computed, and the maximum response (indicated here by the black circle) is taken as the similarity score for that pair. Darker shades of gray represent higher response.

3.2.3 Similarity trends

The robustness of normalized cross-correlation and the invariance properties of difference images make aROI-based similarity fairly reliable on average, but not perfect. The following factors can lead to correlation errors:

1. Repetitive features (e.g. corridors with similar doors or other architectural elements);
2. Occlusion of neighborhoods, with next best matches being far away from the correct pair;
3. Very noisy neighborhoods that correlate well to mostly any patch.

Therefore, instead of relying on individual results, consistent similarity *trends* over whole image sequences must be identified. This is done by selecting a range of teach images, comparing each to all repeat images collected in recent history, and then using linear regression to identify a pairing trend $l = (m, b)$ in the similarity values.

First, let the history of latest w_r repeat images up to current repeat image $\mathbf{J}'_j$ be indexed by:

$$\mathbf{j}_j = \Big[j - w_r, \ldots, j \Big] \tag{3.9}$$

The range of teach difference images is selected in terms of previous pairing trend $l_{j-1} = (m_{j-1}, b_{j-1})$:

$$\mathbf{i}_j = \Big[jm_{j-1} + b_{j-1} - \frac{w_t}{2}, \ldots, jm_{j-1} + b_{j-1} + w_t \Big] \tag{3.10}$$

The *similarity map* $\mathbf{S}_j$ is then defined as:

$$\mathbf{S}_j^{w_t \times w_r} = \left[s_{uv} = \mathrm{S}\Big(\mathbf{i}_j[u] \ , \ \mathbf{j}_j[v] \Big) \right] \tag{3.11}$$

That is, $\mathbf{S}_j$ contains the similarities between each teach image in the range indexed by $\mathbf{i}_j$ and all recent repeat difference images indexed by $\mathbf{j}_j$. In a similarity map, high similarity values tend to gather in roughly diagonal clusters – the direction along which both teach and repeat map indexes increase. A diagonal line can be fit over such clusters by sing RANSAC [48] interpolation, thus finding the immediate trend $l_h = (m_h, b_h)$ which is used to update the previous estimate:

$$m_j = \lambda m_{j-1} + (1 - \lambda) m_h \tag{3.12}$$

$$b_j = \lambda b_{j-1} + (1 - \lambda) b_h \tag{3.13}$$

The use of the above formulas to update estimates, instead of taking l_h directly as the current estimate, is to avoid sudden trend changes caused by momentarily poor similarity measures; instead, similarity changes have to be

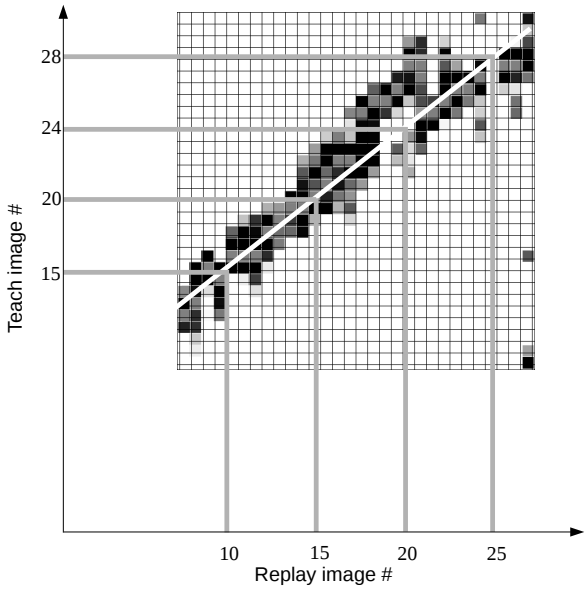


Figure 3.5: A similarity map represents at each cell the similarity between repeat (horizontal axis) and teach (vertical axis) difference images. Darker shades of gray represent higher similarity. The white line indicates the identified matching trend across the map. Gray lines indicate the estimated pairings between repeat and teach images.

sustained before they register as trend changes. The image pair $(\mathbf{J}_{p(j)}, \mathbf{J}'_j)$ can thus be selected such that:

$$p(j) = m_j j + b_j \tag{3.14}$$

Where teach difference image $\mathbf{J}_{p(j)}$ is the pair of repeat image $\mathbf{J}'_j$. Figure 3.5 illustrates pairing function estimation from a similarity map.

The discussion above assumed that for every repeat image $\mathbf{J}'_j$, a previous trend estimate l_{j-1} is available, thus reducing the problem of trend estimation to updating the previous estimate with new information. Obviously, this leaves out the question of how an initial trend can be identified. However this can be done simply by performing the above procedure for teach range $\mathbf{i}_0 = [0, \ldots, n_{teach}-1]$, where n_{teach} is the number of difference images in the teach record; repeat range $\mathbf{j}_0 = [0, \ldots, w_r - 1]$, that is, the first w_r replay images collected; and bypassing the update steps defined in Eq. 3.12 and Eq. 3.13. In this way initial position along the teach route can be determined, solving the kidnapped robot problem.

3.3 Shift estimation

Difference image pairing estimates show far the robot advanced along the teach route. The deviation from the original route can be inferred by computing the *shift* between teach and repeat images – a length of horizontal sliding of one image over the other, such that features of both are "matched" as well as possible. Figure 3.6 illustrates the concept.

Because scenes may change between teach and repeat trips (due e.g. to the presence of moving elements), it's not effective to compare images wholesale. Instead, given a matched pair $(\mathbf{J}_{p(j)} = g'(\mathbf{J}'_j), \mathbf{J}'_j)$, *columns* of width $w_\mathbf{C}$ are selected from teach image $\mathbf{J}_{p(j)}$ one at a time and cross-correlated to $\mathbf{J}'_j$. The resulting vectors are zero-padded and shifted to account for the different initial positions of each column, then summed to produce a *shift likelihood vector* $\mathbf{s}_j$ for the pair $(\mathbf{J}_{p(j)}, J_j)$:

$$\mathbf{s}_j = \sum_{k=0}^{n_J/w_\mathbf{C}} (\mathbf{0}^n +\!\!+ \mathcal{N}_{cc}(\mathbf{J}_{p(i)}[:, w_\mathbf{C}k : w_\mathbf{C}(k+1)], \mathbf{J}'_j)) \ll w_\mathbf{C}k \tag{3.15}$$

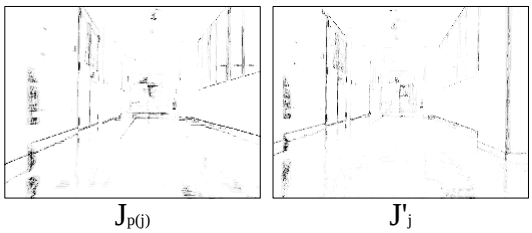
A shift vector describes shift likelihoods: the central value indicates the likelihood that no shift has taken place, while values prior to it represent the likelihood of a shift to the right, and values following, of a shift to the left. Shift vectors are generally consistent over extended ranges, but abrupt changes are sometimes observed, especially if there are differences in landscape composition (e.g. presence / absence of moving elements) between test and repeat streams. Such occasional mismatches can be averaged out by computing *summed shift vectors* such that:

$$\overline{\mathbf{s}}_j = \sum_{k=0}^{w_r} \mathbf{s}_{j-k} \tag{3.16}$$

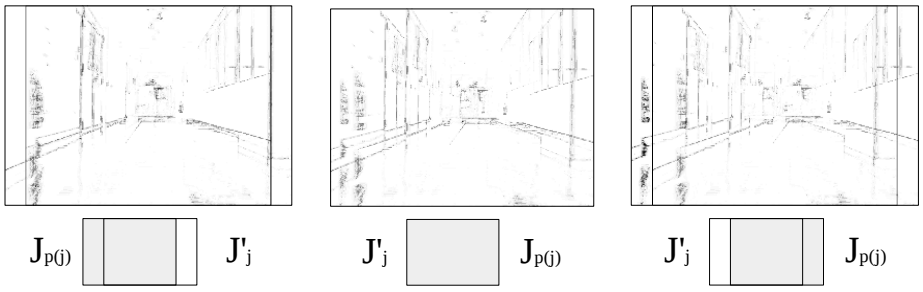
Finally, computing the weighted average of the summed shift vector will produce a definite shift estimate:

$$h_j = \frac{\sum_{k=k_{j,0}}^{k_{j,n}} (k - \frac{n}{2})\overline{\mathbf{s}}_j[k]}{\sum_{k=k_{j,0}}^{k_{j,n}} \overline{\mathbf{s}}_j[k]} \tag{3.17}$$

Where $k_{j,0} = h_{j-1} - \epsilon$ and $k_{j,n} = h_{j-1} + \epsilon$ demarcate a window around the position of the previous shift (so successive shift estimates will vary more smoothly over time), with $k_{0,0} = 0$ and $k_{0,n} = n$.



(a) Difference image pair



(b) Left offset (c) No offset (d) Right offset

Figure 3.6: Simplified example of image shift detection. A pair of difference images (a) are overlaid at different offsets; the offset that produces the smallest amount of feature mismatch between images is taken as their shift. Example overlays at a "left" offset (b), no offset (c) and "right" offset (d) are shown. In this case "left" and "right" are relative to the repeat image $\mathbf{J}'_j$ – the idea is that $\mathbf{J}'_j$ is kept fixed and $\mathbf{J}_{p(j)}$ slides over it.

3.4 Steering

DICH's steering model is simple. It assumes a differential mobile platform, that is, one where motion is provided by a pair of left and right wheels with separate velocities $(v_{L,t}, v_{R,t})$ such that:

$$v_t = \frac{v_{R,t} + v_{L,t}}{2} \tag{3.18}$$

$$\omega_t = \frac{v_{R,t} - v_{L,t}}{l} \tag{3.19}$$

Where v_t and ω_t are respectively the robot's linear and angular velocities, and l is the distance between the wheels. Given desired values for v_t and ω_t, the equations above enable the calculation of suitable values for $v_{L,t}$ and $v_{R,t}$.

Steering assumes a constant linear velocity v. Angular velocity is varied over time according to the following rule:

$$\omega_t = d\omega \tag{3.20}$$

Where ω is a system parameter, and d is computed as:

$$d = \begin{cases} -1 \text{ if } h_j > 0 \\ 1 \text{ if } h_j < 0 \\ 0 \text{ otherwise} \end{cases} \tag{3.21}$$

So that, for example, if the robot starts to drift left, a negative angular velocity is effected so that the robot starts to steer rightwards; the opposite would be true if the robot drifted right.

Chapter 4

Implementation

DICH is implemented as a collection of software modules running inside process nodes, instantiated on a computer connected to a differential drive mobile robot. The next sections describe each of these elements in detail.

4.1 Hardware Components

The hardware platform employed in the development of DICH includes a computer platform for running the software on, and a robotics platforms for real-world experiments. For the former, a fairly standard Lenovo X201 notebook boarding a dual-core Intel Core i5 clocked at 2.4GHz was used [49]. Each core can run two hardware threads simultaneously, bringing the number of possible active execution contexts up to four.

The robotics platform employed for the whole of research was the Yamabico M1, an original design developed by the Intelligent Robot Lab [50]. It's a relatively large differential drive robot, at $40cm$ height, $30cm$ width and $35cm$ length. Yamabico robots are mostly built on milled aluminum parts and commodity components. The M1 model in particular boards a power electronics module feeding two brushless motors and a T-Frog control board [51] from a pair of 12V batteries housed at the frame's bottom.

The T-Frog is designed to receive commands in real-time from a PC connected to it through a USB interface. Accordingly, an acrylic board is mounted at the top of the robot, so that a controller notebook or other portable device can be placed on it. A Hokuyo URG-04LX laser scanner [52] is also mounted to the robot's front, but it was left unused. By default the M1 is not equipped with a camera; this was solved by the fixation of a Logitech C920 to its top. The C920 is a common web camera with a 52 degrees diagonal FOV, 4:3 aspect ratio, and 30 Frames-Per-Second (FPS) frame rate [53]. Figure 4.1 shows the complete experimental setup of robot, controller notebook and camera.

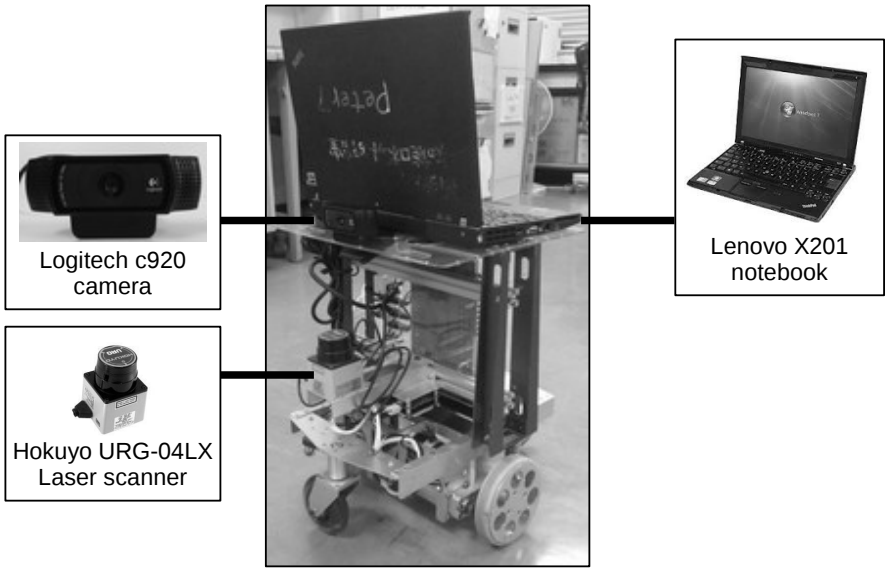


Figure 4.1: Yamabico M1 differential drive robot, mounted with control computer and front-facing camera. The pre-boarded laser scanner was left unused.

4.2 Software Components

DICH's software modules were written in C++. Modules communicate using the Robot Operating System (ROS) message passing framework [54]. Kubuntu, a KDE-based variation of Ubuntu, was used as the main development and execution environment [55]. OpenCV [56] was used for most of the basic image manipulation tasks, while Intel's Thread Building Blocks (TBB) [57] provided function-level parallelization.

The implementation of the DICH method itself is divided across three modules. The **Difference Image** module computes difference images from visual input coming directly from a camera or replayed form a recoding. The **Image Pairing** module receives repeat difference images and pairs them teach difference images from an in-memory base. Finally, the **Shift Estimation** module receives repeat difference images along with pairing information, returning shift estimates between each teach / repeat image pair. The later modules can be made to share a common in-memory teach record base, thus reducing memory requirements. A number of minor modules was also implemented, to deal with assistant tasks such as user interaction.

Different module configurations can be instantiated as node *networks* to realize different use cases. These include driving the robot and collecting data during the teach step, performing localization (and optionally steering) during the repeat step, and computing ground truth data for performance evaluations.

4.2.1 Teach network

The teach network is responsible for driving the robot in response to operator commands, and recording route data as a teach record. Figure 4.2 provides a graphical representation of the network. Two user interface modules are responsible for collecting user commands from a terminal prompt and displaying current sensor input in a GUI window. User commands are sent to a command translator module that communicates with the steering module, which controls the robot's motors. Commands are also sent to a data recorder module, to be saved to file as teach records along with visual input. In order to avoid recording useless data, images are saved to disk only when the robot is in movement.

4.2.2 Repeat network

The repeat network is responsible for executing the DICH method on visual records, collect live visual input during the repeat step, and/or steer the robot. When started, the network first instantiates a single ROS node running the main DICH modules, responsible for difference image computation, image pairing and shift estimation, and loads a teach record to memory (Fig. 4.3). Then, if running in offline mode, it loads a repeat record for synchronous processing, writing image pairing and shift estimates to a log file (Fig. 4.4). If however the network is started in online mode, each DICH module is assigned to its own thread within the ROS node; this is done so that real-time performance requirements can be met. Additionally, steering and camera control nodes are started to establish communication with the robot's hardware (Fig. 4.5).

4.2.3 Ground truth network

Ground truth data for test sessions (Chapter 5) was computed by manually comparing the frames of teach and repeat step video recordings. A simple GUI application was developed for this purpose, shown in Figure 4.6. Given a pair of teach records representing a teach and a repeat step, the application displays every τ^{th} frame of the teach record; the user then chooses (using the UP and DOWN keys) which repeat record frame best matches it, and (using the LEFT and RIGHT keys) an optimal sliding that best matches the features of both images. To make feature matching easier for the user, the selected repeat record frame is superimposed to the teach frame at 50% transparency. As matches are selected (using the ENTER key), records containing the matched images' indexes and sliding are simultaneously printed to console and saved to a text file.

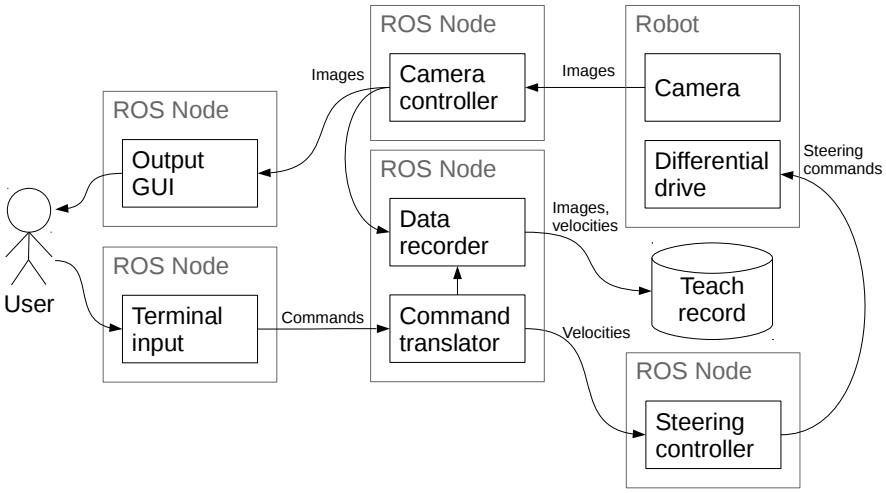


Figure 4.2: DICH teach network.

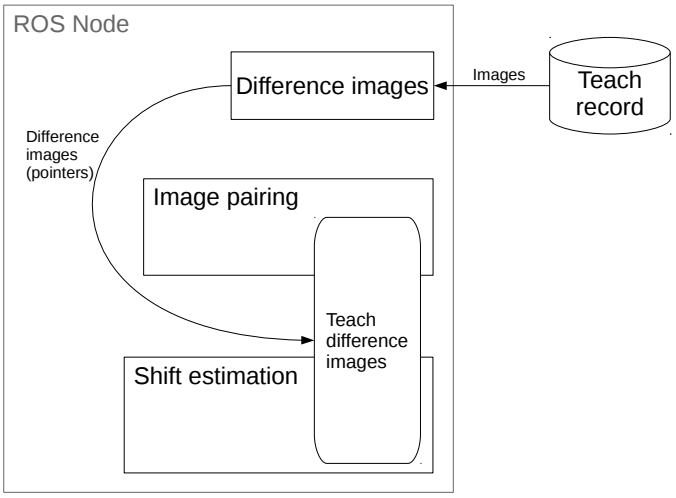


Figure 4.3: DICH repeat network initial setup. A single ROS node is instantiated to run the main DICH modules, and teach step records are loaded to a memory area shared by the image pairing and shift estimation modules.

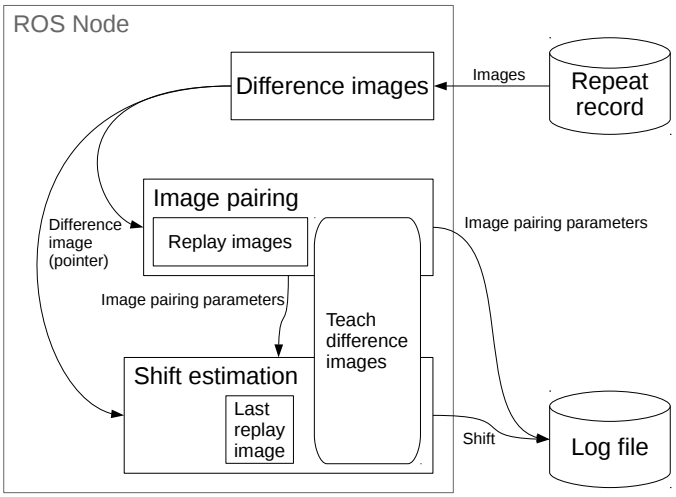


Figure 4.4: DICH repeat network in offline mode. After the initial setup is complete, images are read from a repeat record, localization estimates computed and written to disk.

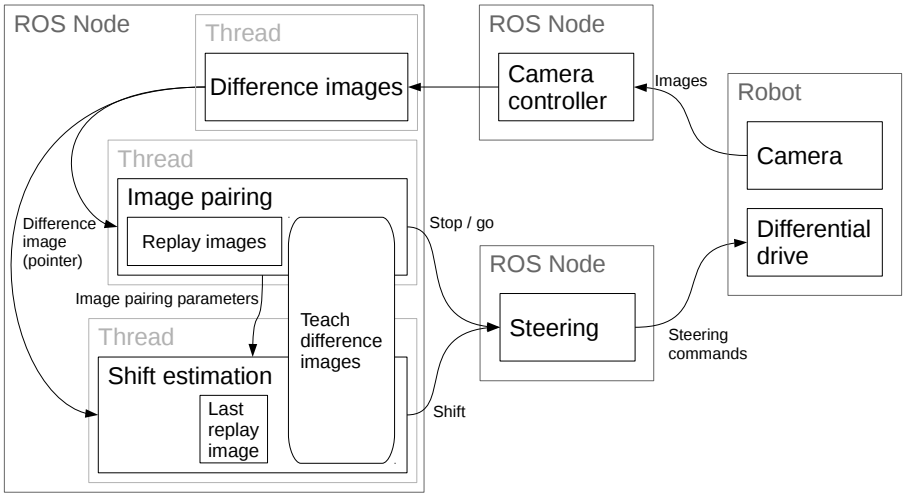
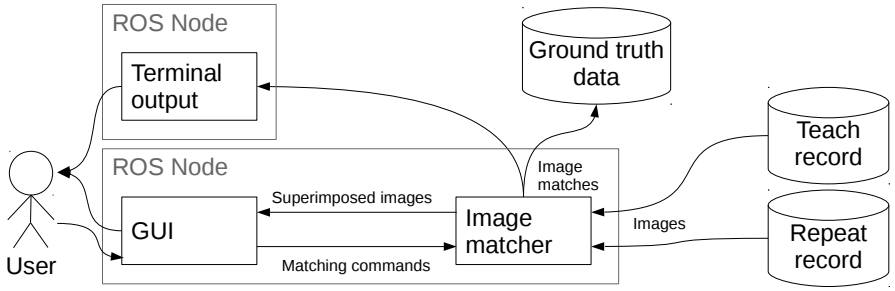


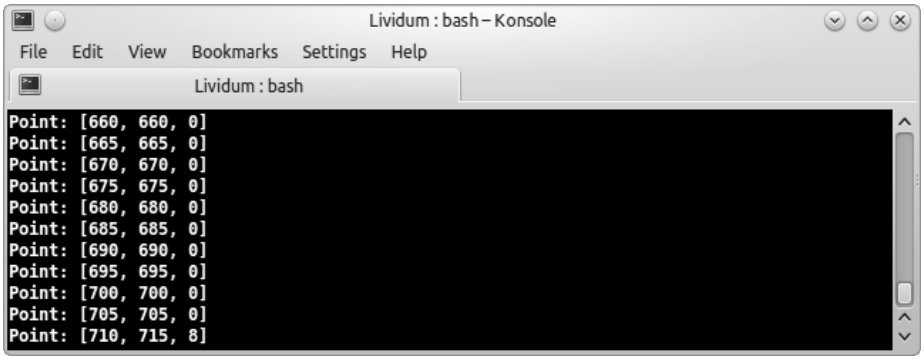
Figure 4.5: DICH repeat network in online mode. After the initial setup is complete, each DICH module is assigned to its own thread, and additional modules are started to enable communication with the robot's hardware.



(a) Ground truth network

(b) Misaligned images

(c) Aligned images



(d) Console output

Figure 4.6: Ground truth computation. (a) Network diagram. (b) For every τ^{th} teach record image, the user selects a repeat frame that best matches it. (c) Then the user aligns the features of the superimposed image pair. (d) Selected matches are printed to console and also saved to disk.

Chapter 5

Experiments

In order to evaluate DICH for both localization and navigation tasks, three sets of experiments were performed. Localization experiments used DICH in offline mode on videos collected in real-world environments, evaluating its ability to recognize places and detect route deviation under differing environment and viewpoint configurations. Navigation experiments used DICH in online mode to evaluate its ability to correctly steer the robot towards a recorded route. Finally, *extremis* offline experiments tested the limits of the method, matching it against video records especially prepared to emphasize specific forms of input variation. Unless otherwise stated, system parameters were set according to Table 5.1 below.

A single experiment – also called *test session*, or simply *test* – is composed of two trips over a selected route: a reference or *teach step* trip, and a comparison or *repeat step* trip. During teach step trips the robot was always controlled by a human operator. Repeat trips for localization and extremis experiments were also manually driven; only in navigation experiments would the robot drive itself. For each experiment, test results are reported as similarity maps and shift maps, along with final estimates and manually computed ground truth data.

Similarity maps are composed by plotting all similarity values computed over a repeat step on a global coordinate frame, where the horizontal axis represents repeat image index, and the vertical axis, teach image index. Higher similarity values are represented as darker shades of gray. A full line over the map indicates image pairings, and a dashed line, ground truth values. Below each similarity map an error graph shows pairing estimation error for each repeat image: positive values indicate the estimated pairing was further along the teach record than the ground truth, and negative values, estimates that were further behind.

Table 5.1: List of DICH parameters and values used in experiments. Each parameter is accompanied by a short description and a reference to the Chapter 3 equation where it was first introduced.

Parameter	**Value**	**Reference**	**Description**
δ	0.1	Eq. 3.2	Difference image threshold ratio
τ	5	Eq. 3.3	Difference image accumulation count
$m_{\mathbf{J}}$	192	-	Scaled difference image height
$n_{\mathbf{J}}$	256	-	Scaled difference image width
α	12	Eq. 3.4	Interest point padding
β	12	Eq. 3.7	Region-Of-Interest padding
w_r	20	Eq. 3.9	Repeat image range
w_t	50	Eq. 3.10	Teach image range
λ	0.98	Eq. 3.12	Image pairing trend update rate
$w_{\mathbf{C}}$	16	Eq. 3.15	Shift estimation column width
v	0.3	Eq. 3.18	Linear velocity
ω	0.1	Eq. 3.19	Angular velocity

Shift maps are constructed simply by plotting averaged shift vectors column-wise. Higher shift likelihood values are represented as darker shades of gray. The horizontal axis represents repeat image index, and the vertical axis, image shift. A full line over the map indicates shift estimates, and a dashed line, ground truth values. Positive shift values indicate left shift, and negative values, right shift. Below each shift map an error graph shows shift estimation error for each repeat image: positive values indicate the estimated shift was further left than the ground truth, and negative values, estimates that were further right.

5.1 Localization Experiments

Localization experiments evaluated DICH's performance in detecting and quantifying teach route deviations. Accordingly, the robot was driven manually in the repeat as well as teach steps, and video records were processed offline afterwards. Two separate environments were used for tests, one indoors, another outdoors, described in Fig. 5.1 and Fig. 5.2 respectively.

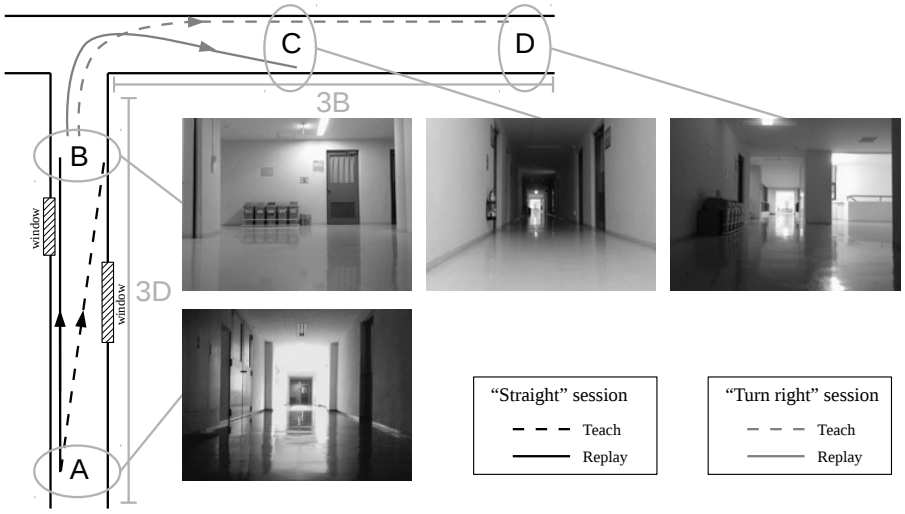


Figure 5.1: Indoors experiment environment and localization test sessions. Corridor "3D" is illuminated from the outside through two windows, while "3B" is artificially lighted. Points A, B, C and D are roughly $20m$ apart. Localization sessions "straight" and "turn right" are performed in this environment.

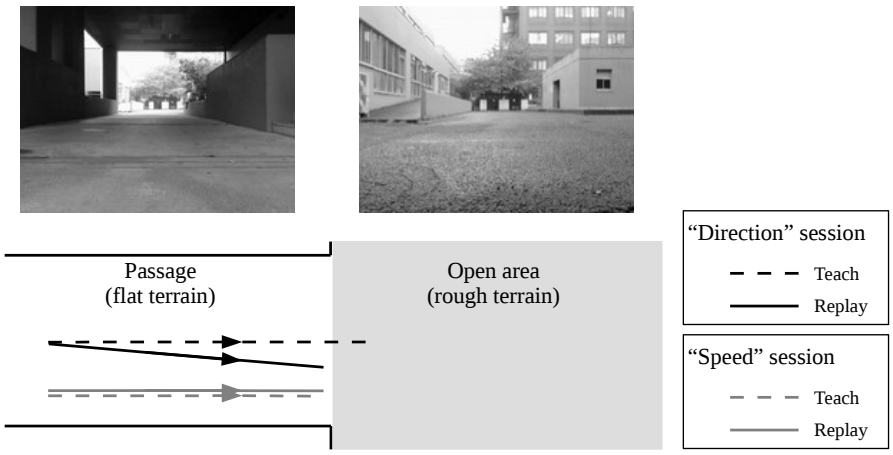


Figure 5.2: Outdoors experiment environment and localization test sessions. A relatively narrow passage between two buildings lead into a wider open area. The passage's floor was flat and smoother than open area's, which was too rough and would not allow the robot to reach very far. Localization sessions "direction" and "speed" are performed in this environment.

5.1.1 Indoors Experiments

In the indoors environment, two corridors (labeled "3D" and "3B" after the buildings hosting them) join perpendicularly at one end. Except for two windows at corridor 3D, very little natural light enters the space. Movement sensors control lighting on corridor 3B, but not on 3D. Test sessions "straight" and "turn right" were recorded in this environment; see Figure 5.1 for an illustration.

Session "straight" started and ended in corridor 3D; it evaluates the method's resilience against moving elements and lighting variations within records. Trip length was set to $20m$, but in the teach step the robot started close to the left wall and slowly drifted rightwards until stopping close to the right wall, while in the repeat step the robot remained close to the left wall for the duration of the route. The corridor was deserted in the teach step, but from frame 144 into the repeat step three people come from behind, staying on the right side of the field of view until about frame 280. The corridor was initially dark, then became brighter as the robot moved into windowed and artificially lighted regions. Image pairing and shift estimation test results are shown in Figure 5.3 and Figure 5.4 respectively. Image pairing estimates agreed well with ground truth data for the first two thirds of the route, then started to diverge; shift estimates on the other hand remained close to ground truth values until the end, occasionally diverging but coming back later on.

Session "turn right" emphasizes lighting changes between records and more dynamic viewpoint changes in a restricted environment, as well as the ability to match a shorter repeat step against a longer teach step, in a regular environment with a higher chance of image mismatch. Both teach and repeat steps start at the end of corridor 3D and continue through corridor 3B after a sharp turn right, but in the teach step the robot goes much further; it also enters a initially dark corridor 3B, with lights turning on automatically as it advances. This resulted in sharp illumination differences between parts of the teach and repeat steps. No moving elements were present in either step. Image pairing and shift estimation test results are shown in Figure 5.5 and Figure 5.6 respectively. Image pairing estimates never quite agreed with ground truth data, but never diverged either; shift estimates deviated starkly from ground truth while the robot was turning around the corner between corridors, but otherwise shift errors were small.

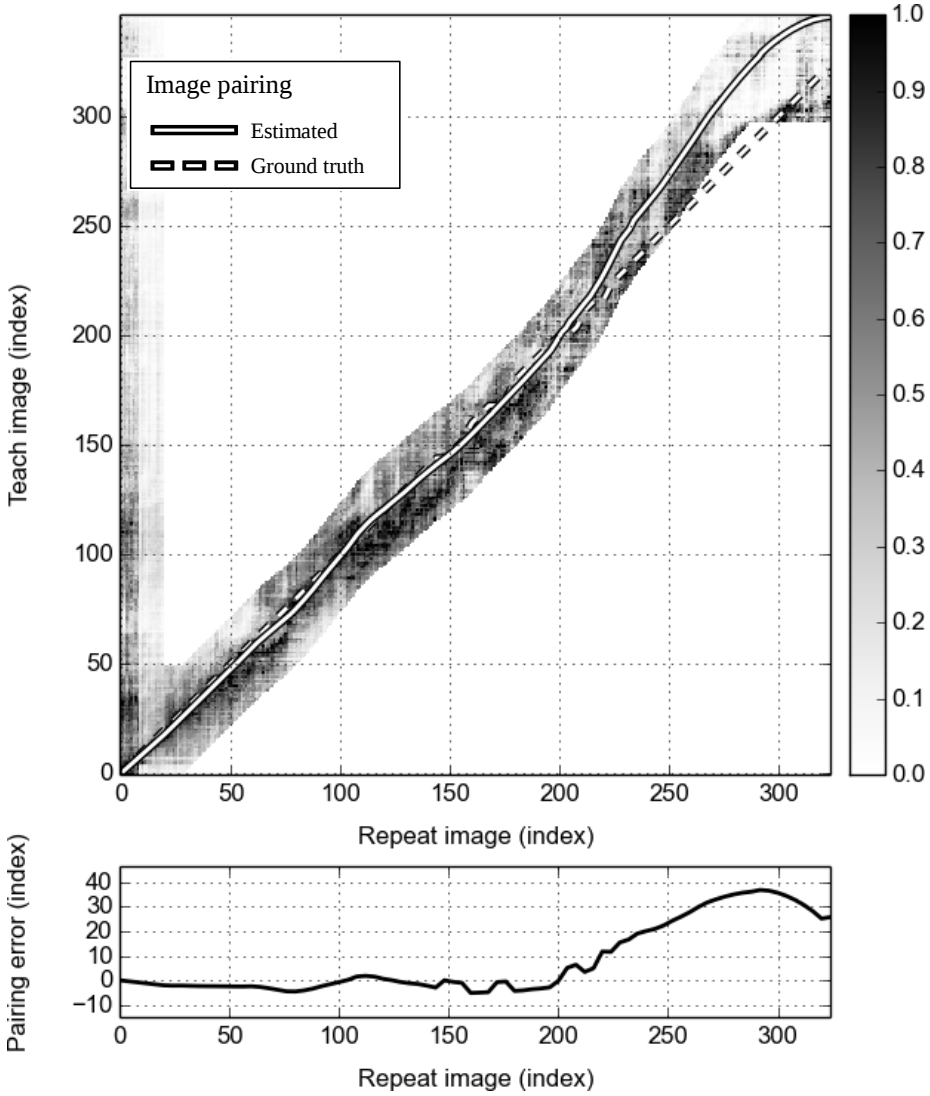


Figure 5.3: Similarity map and image pairing estimate results for the indoors "straight" localization test session. See description at beginning of the chapter for interpretation details.

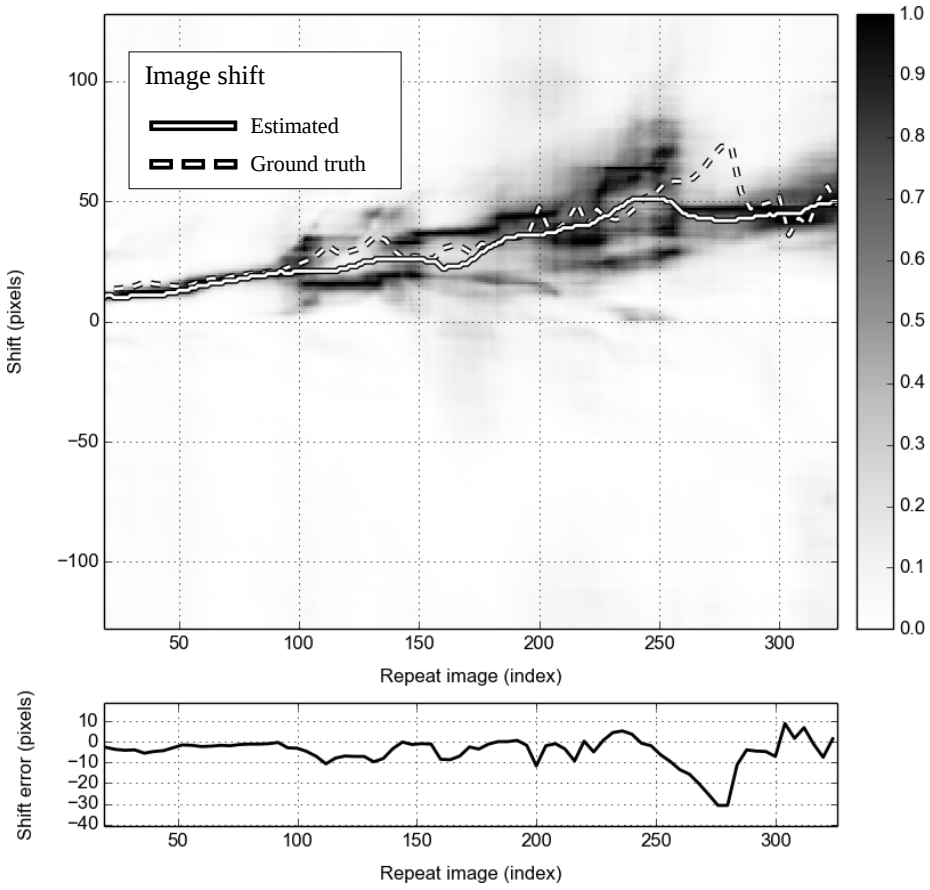


Figure 5.4: Shift map and image shift estimate results for the indoors "straight" localization test session. See description at beginning of the chapter for interpretation details.

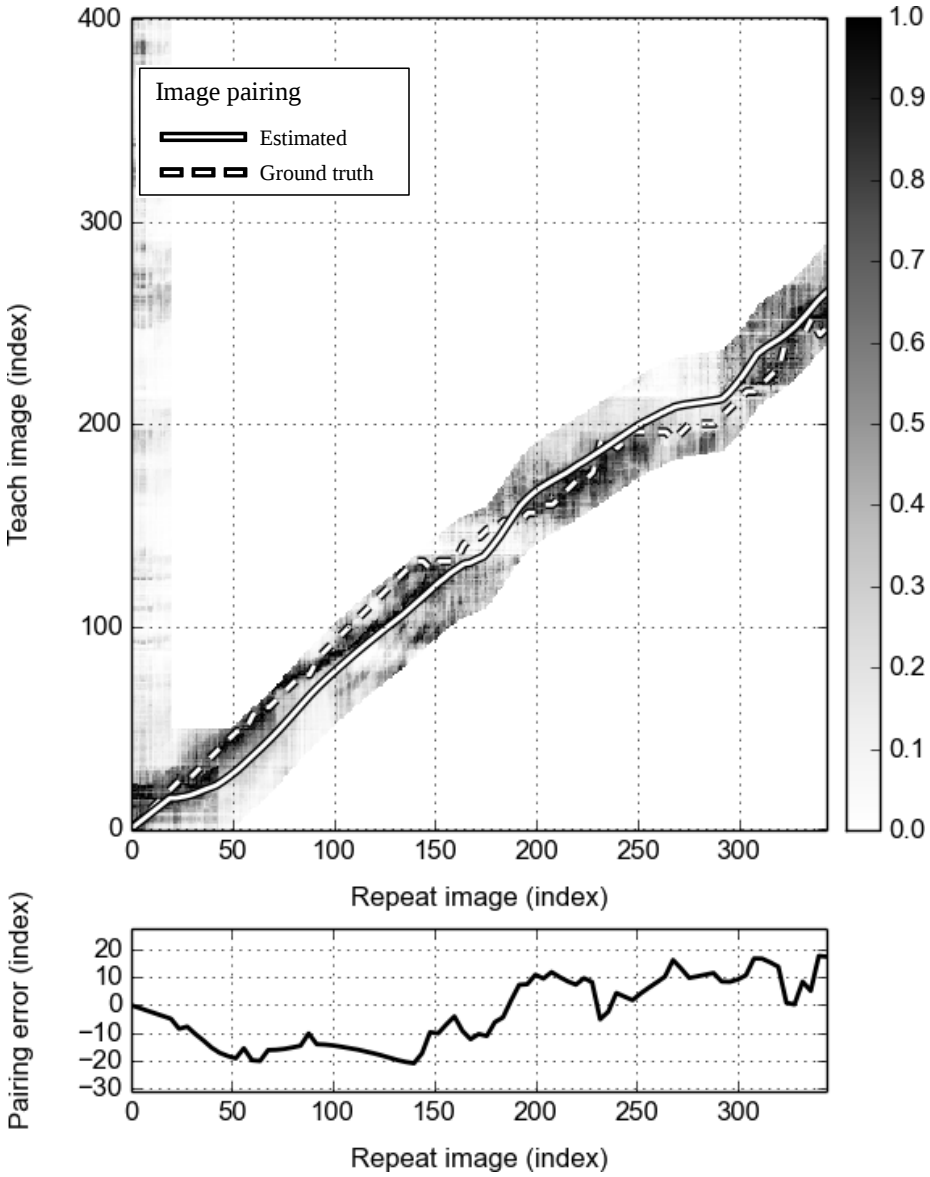


Figure 5.5: Similarity map and image pairing estimate results for the indoors "turn right" localization test session.

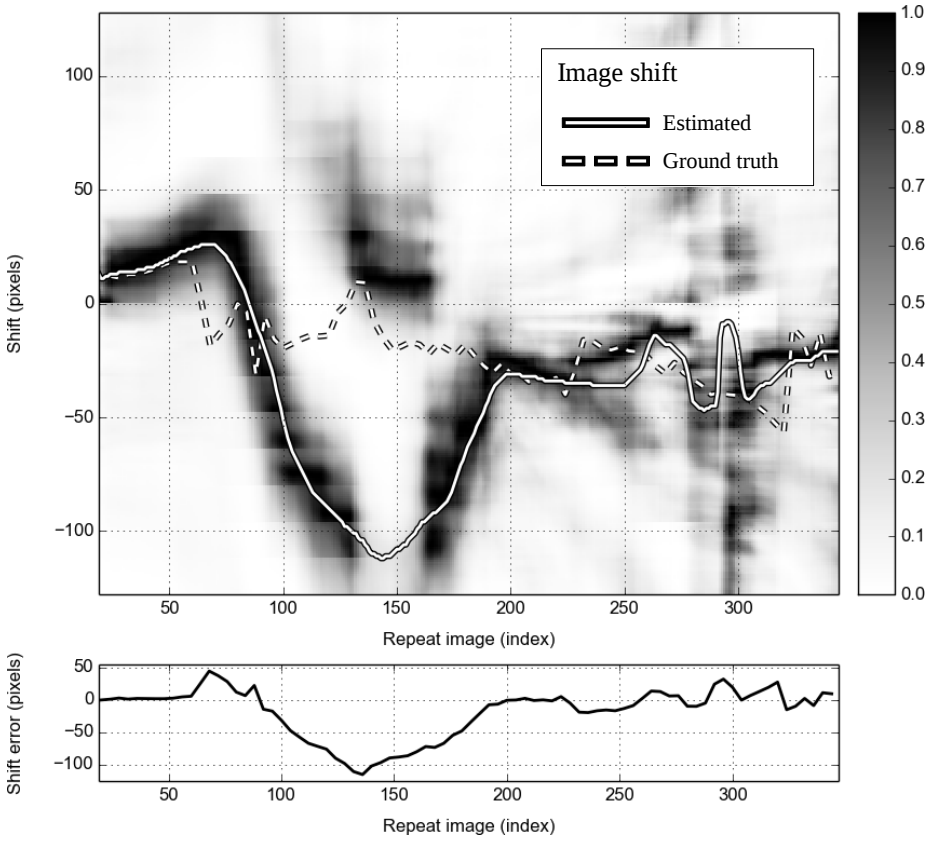


Figure 5.6: Shift map and image shift estimate results for the indoors "turn right" localization test session. See description at beginning of the chapter for interpretation details.

5.1.2 Outdoors Experiments

In the outdoors environment, a relatively narrow passage between two buildings led into a wider open area. The passage's floor was rough enough to add a noticeable amount of vibration to robot movement; the open area's floor was too rough for the robot to reach very far. Sessions "direction" and "speed" were recorded in this environment; see Figure 5.2 for an illustration.

Session "direction" demonstrates the method's ability to work outdoors under increasing route divergence, presence of moving elements and travel length differences. In the teach step the robot moved straight ahead for $20m$, eventually reaching the patch of rough terrain in the open area. A pedestrian came from behind the robot about from frame 49 into the step, staying on the left side until about frame 220. In the repeat step the environment was deserted; the robot advanced for just $15m$ to avoid the rough surface ahead, all the while drifting to the right of the teach step route. Image pairing and shift estimation test results are shown in Figure 5.7 and Figure 5.8 respectively. While neither estimate ever quite agreed with ground truth values, errors remained low and constant for the duration of the experiment.

Session "speed" shows how the method deals with speed differences between steps. In the teach step the robot moved straight ahead for $15m$ at a speed of $0.3m/s$, whereas in the repeat step it moved at a speed $0.6m/s$, while following essentially the same path. The environment was deserted through both steps. Image pairing and shift estimation test results are shown in Figure 5.9 and Figure 5.10 respectively. As in the previous experiment, while a degree of divergence between estimates and ground truth was always present, estimates generally followed the trends indicated by ground truth data.

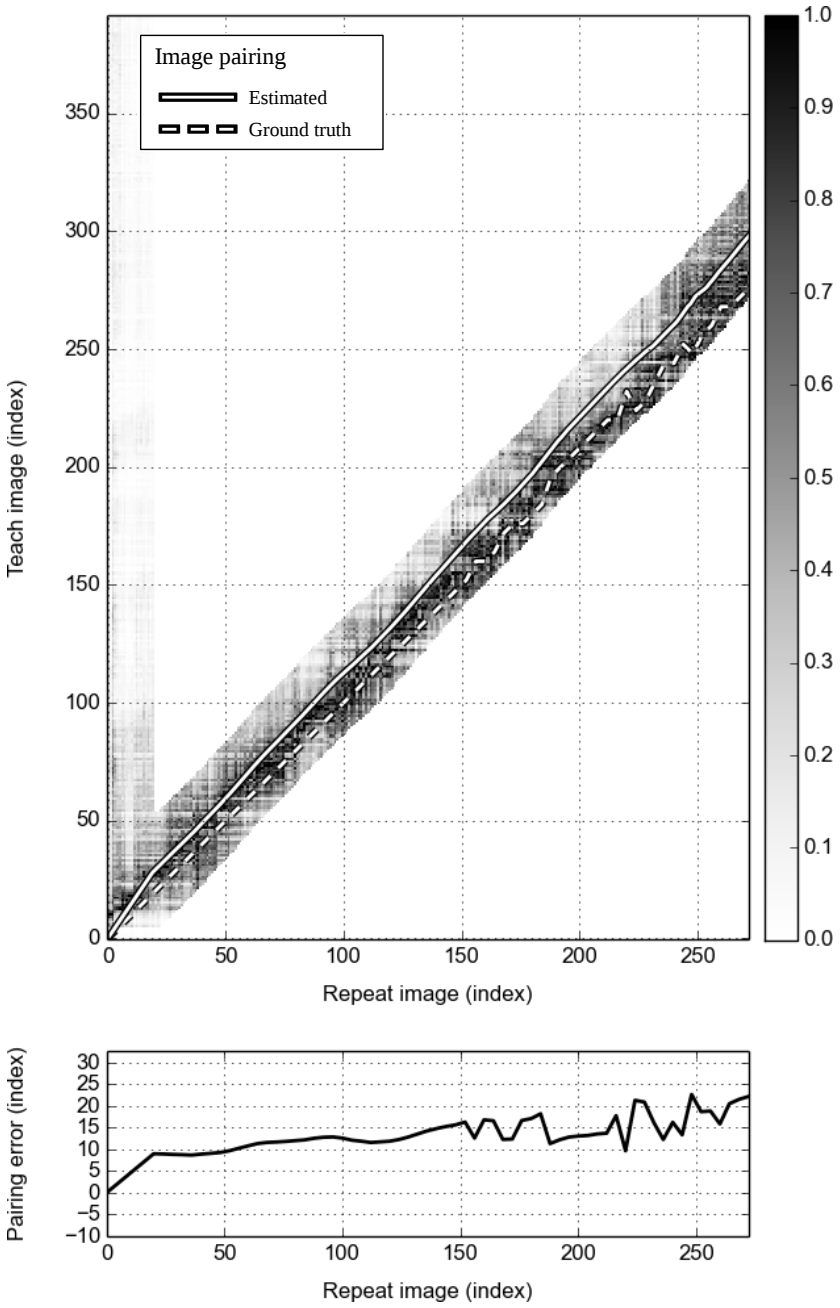


Figure 5.7: Similarity map and image pairing estimate results for the outdoors "direction" localization test session. See description at beginning of the chapter for interpretation details.

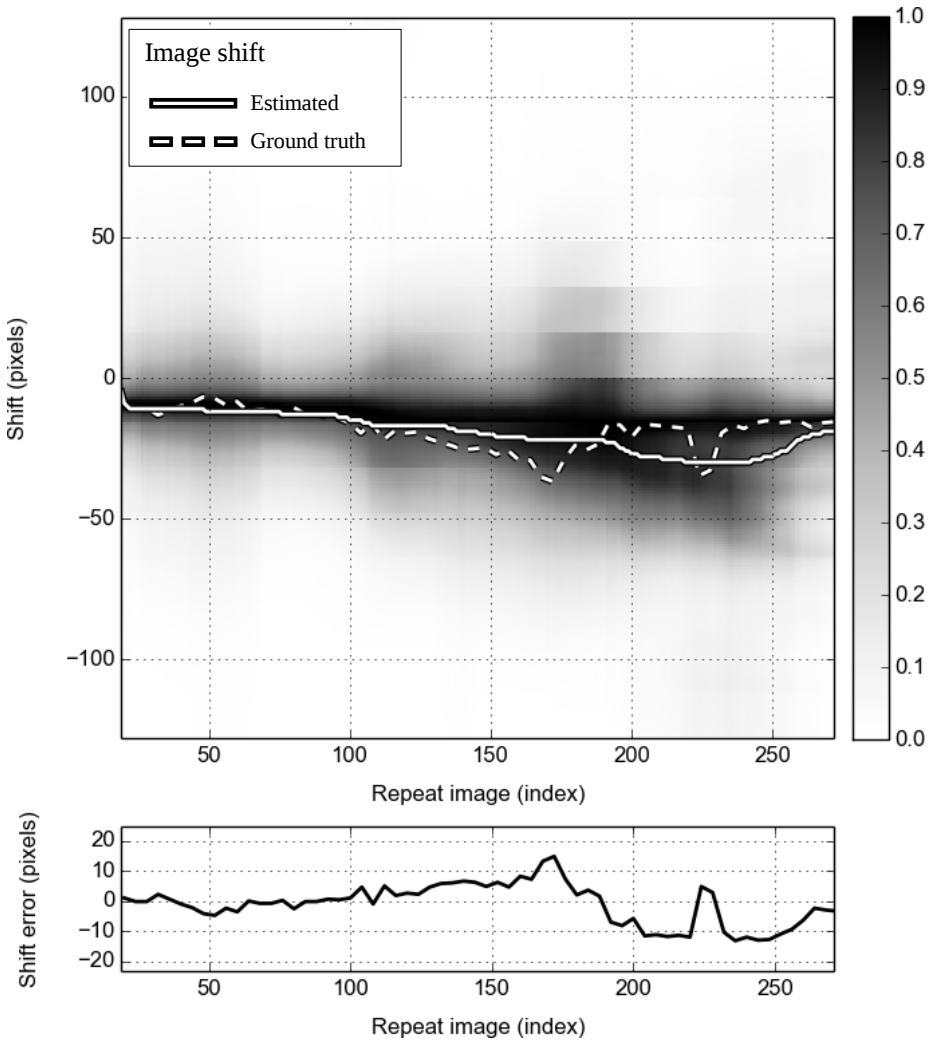


Figure 5.8: Shift map and image shift estimate results for the outdoors "direction" localization test session. See description at beginning of the chapter for interpretation details.

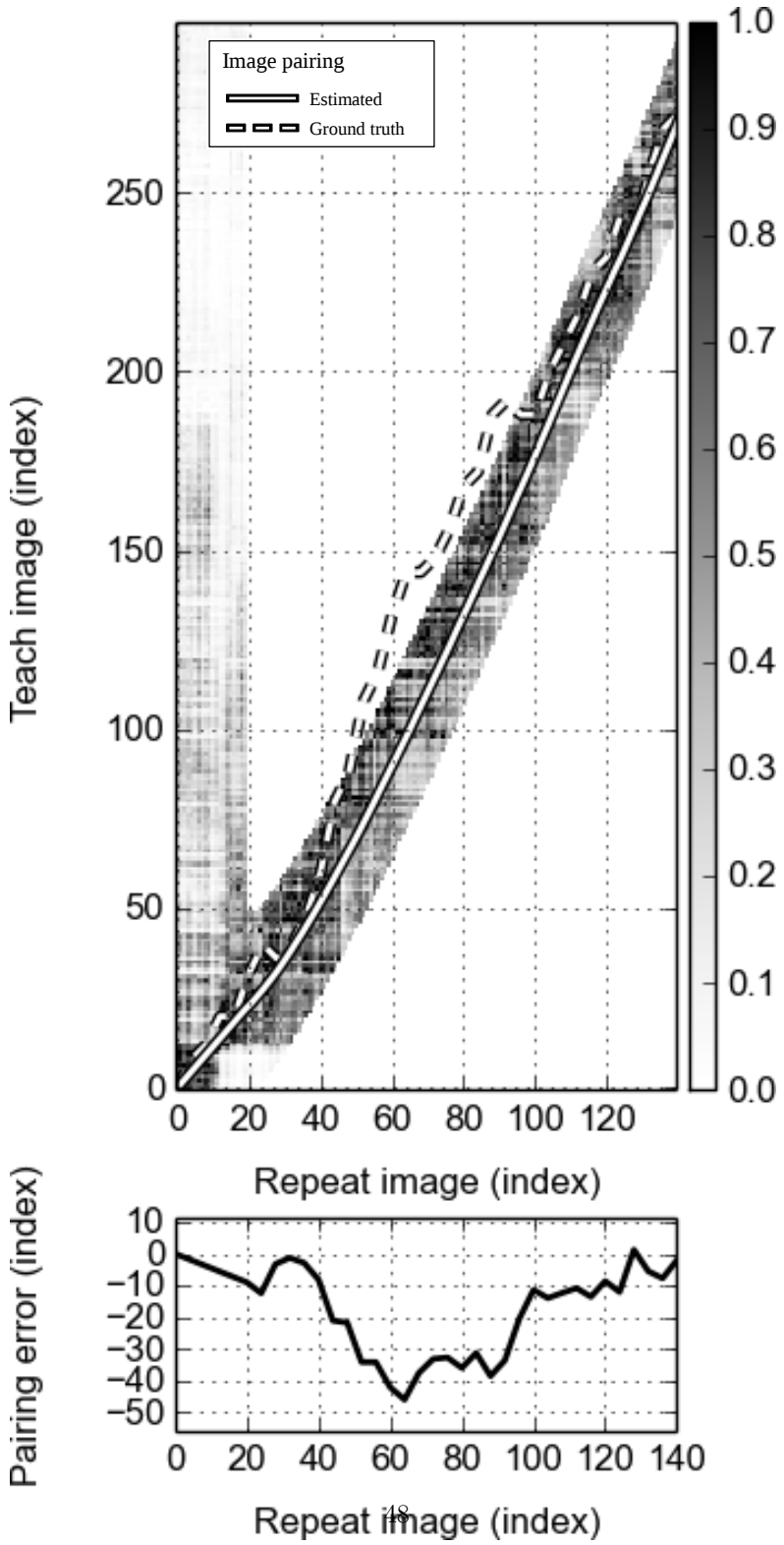


Figure 5.9: Similarity map and image pairing estimate results for the outdoors "speed" localization test session. See description at beginning of the chapter for interpretation details.

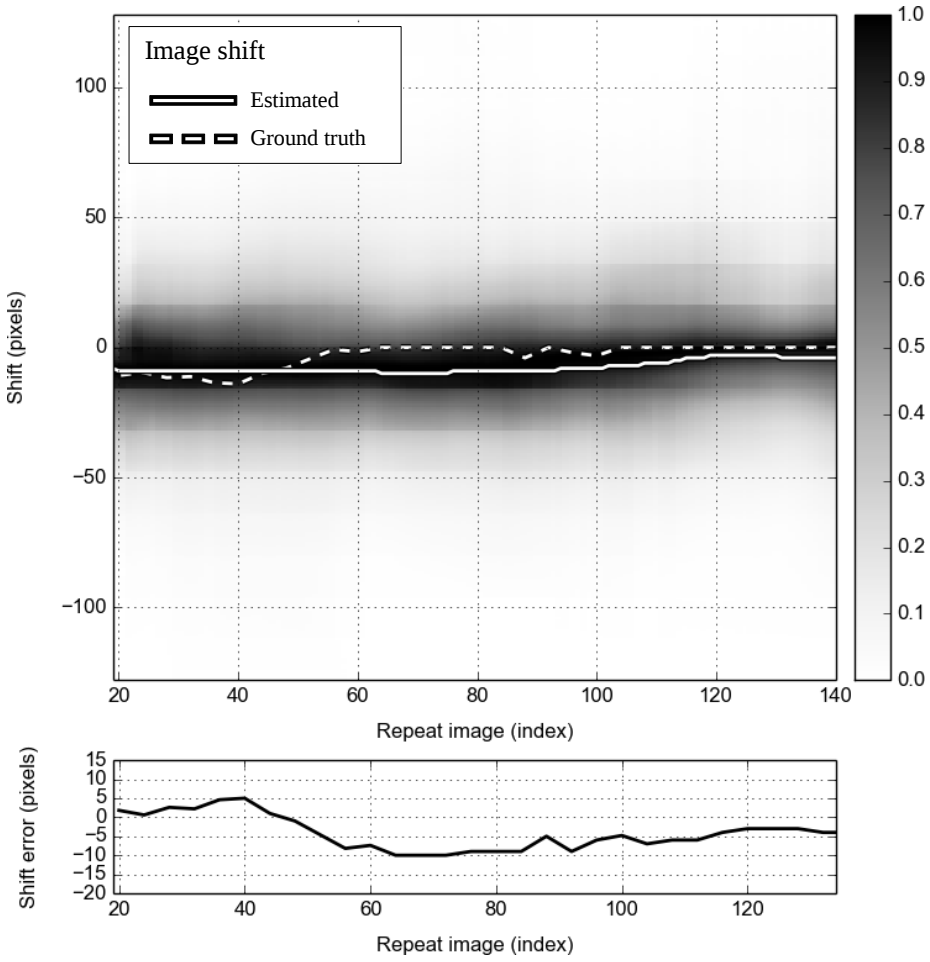


Figure 5.10: Shift map and image shift estimate results for the outdoors "speed" localization test session. See description at beginning of the chapter for interpretation details.

5.2 Navigation Experiments

Localization experiments demonstrate DICH can correctly estimate location relative to teach step routes despite input variations. However, steering adds a feedback element that cannot be assessed from offline tests. Therefore indoors and outdoors navigation experiments were also performed. An initial image pairing trend $l_0 = (1, 0)$ was assumed in order to save time, but parameters were otherwise the same as in localization experiments.

Indoors navigation experiments were performed in the same corridor 3D and initial position as the "straight" localization test. Two tests were performed, dubbed "maintain" and "converge". Figure 5.11a illustrates the environment and respective teach routes.

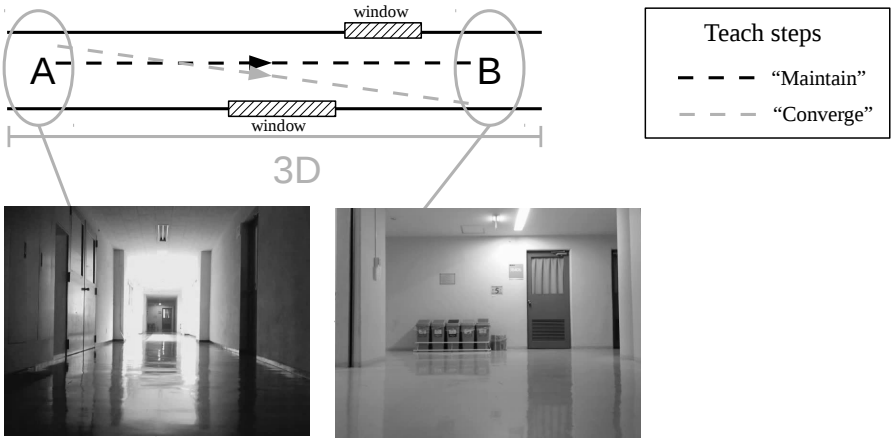
In the "maintain" experiment, for the teach step the robot was driven over the environment keeping parallel to corridor walls until reaching a destination point approximately $20m$ away from the starting position. In the repeat step the robot set off already in the correct route, and was tasked with remaining on it. Image pairing and shift estimation test results are shown in Figure 5.12 and Figure 5.13 respectively. Moreover Figure 5.14 shows odometry records for teach and repeat steps. As can be seen the robot diverged a little from the teach route towards the end, yet errors remained low for the extent of the repeat trip.

In the "converge" experiment, for the teach step the robot was driven over the environment drawing a diagonal from a start position closer the left wall to a destination closer to the right wall approximately $20m$ away. During the repeat step the robot, initially advancing parallel to corridor walls, had to steer itself towards the teach route, converging towards it. Image pairing and shift estimation test results are shown in Figure 5.15 and Figure 5.16 respectively. Moreover Figure 5.17 shows odometry records for teach and repeat steps. As can be seen DICH guided the robot to quickly converge towards the correct route, with errors kept to a minimum.

Outdoors experiments were performed on a bicycle parking lot behind building 3D. A single teach trip was recorded, late on the afternoon of a clear weather day (2017-07-28 18:00 for precise date and time). Two replay trips then used this record as reference: "evening", early evening the next day (2016-07-29 19:00), and "afternoon", two days later at noon (2016-07-30 12:00). Figure 5.11b shows a diagram of the environment and example images for each trip.

Image pairing, shift estimation and odometry results for session "evening" are shown in Figure 5.18, Figure 5.19 and Figure 5.20 respectively. Results were generally positive: DICH was able to correctly pair images even though the camera automatically halved its framerate to account for the reduced illumination (this is why the repeat axis of the similarity map is so shorter than the teach axis). Shift estimates were more timid and varied more slowly than what was found manually, but mostly it correctly followed the trend of the changes.

Image pairing and shift estimation and odometry results for session "afternoon" are shown in Figure 5.21, Figure 5.22 and Figure 5.23 respectively. Again image pairing proved very reliable, while shift estimates were comparatively timid but on the right track.



(a) Indoors navigation environment

(b) Outdoors navigation environment

Figure 5.11: Navigation experiments environments and teach routes. Sample images illustrate environment appearance at various circled regions. (a) Corridor 3D from indoors localization experiments was reused for indoors navigation experiments. (b) A parking lot behind building 3D was used as outdoors navigation environment. Images taken at different times of the day show how the environment's appearance varies in response to changes in illumination.

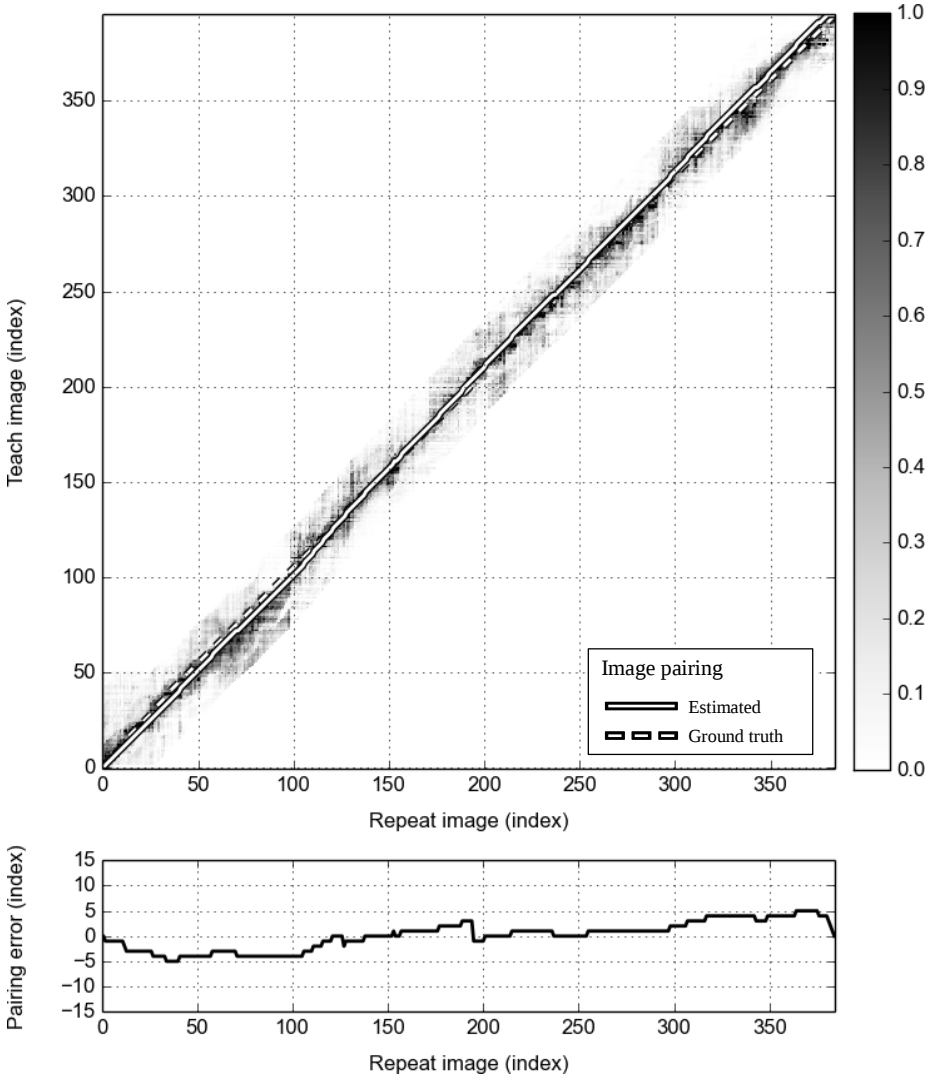


Figure 5.12: Similarity map and image pairing estimate results for navigation experiment "maintain". See description at beginning of the chapter for interpretation details.

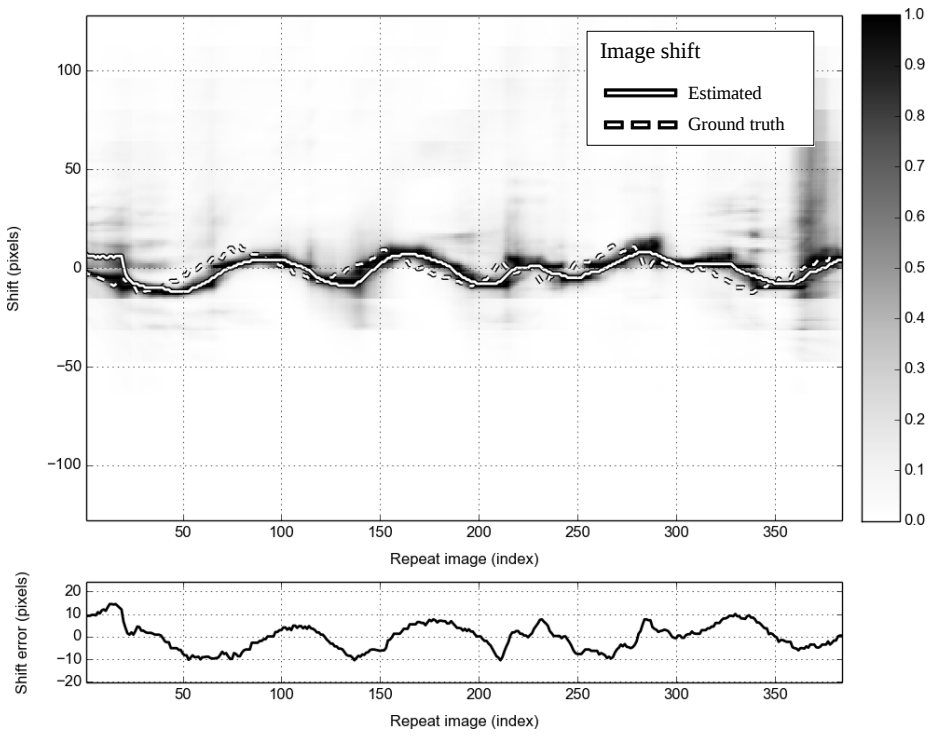


Figure 5.13: Shift map and image shift estimate results for navigation experiment "maintain". See description at beginning of the chapter for interpretation details.

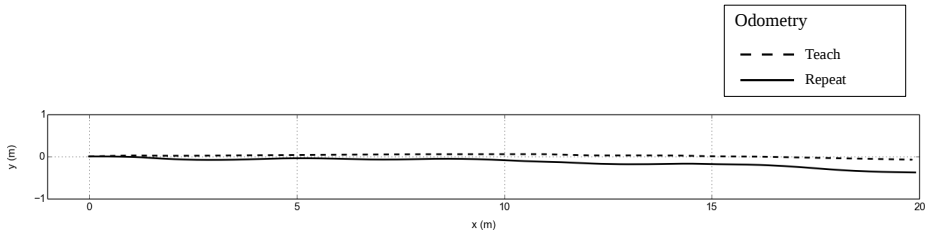


Figure 5.14: Odometry results for navigation experiment "maintain". Horizontal axis is position along the length of the corridor, and vertical axis, along its width. Dashed line indicates teach step route, and full line, repeat step driving.

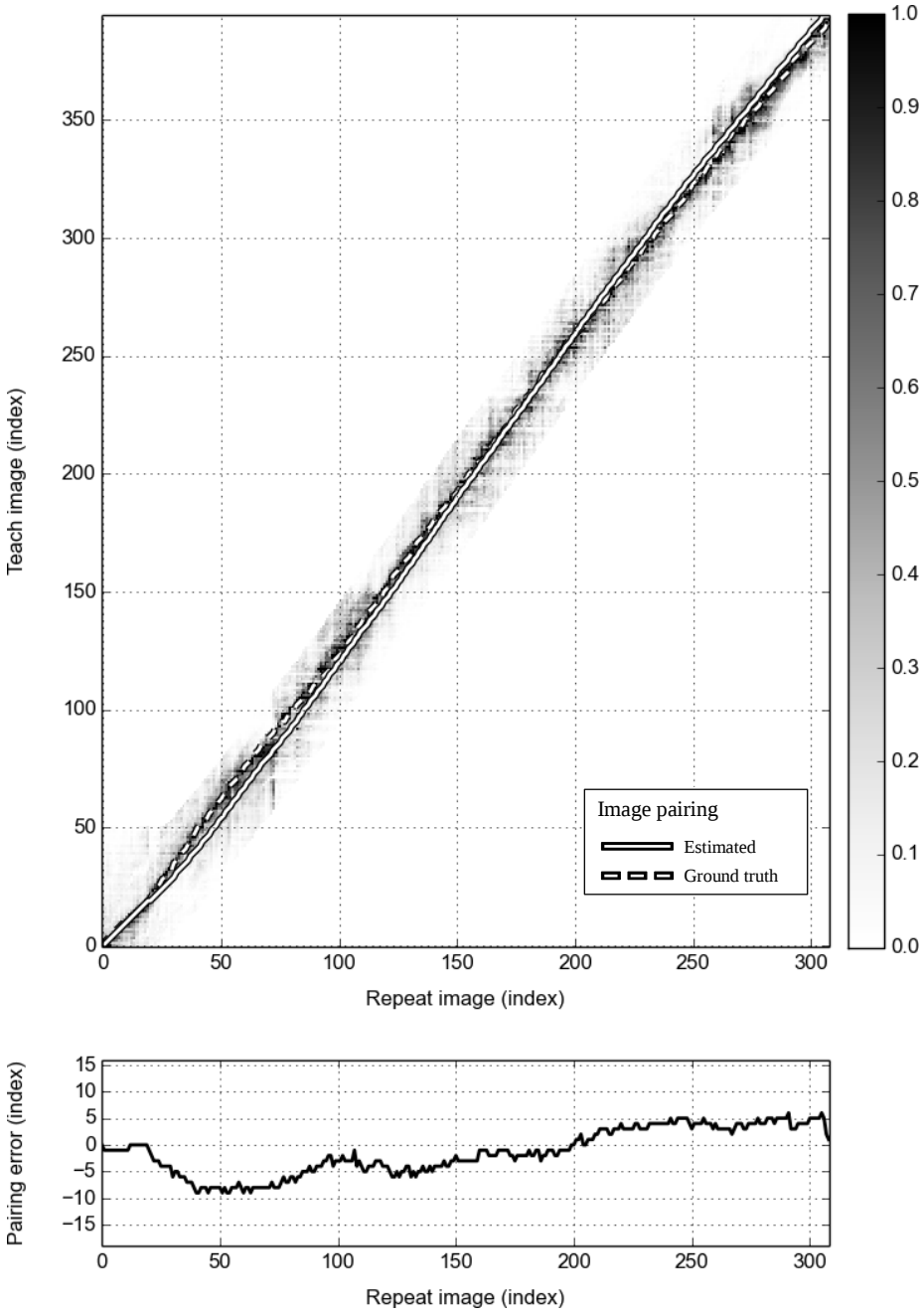


Figure 5.15: Similarity map and image pairing estimate results for navigation experiment "converge". See description at beginning of the chapter for interpretation details.

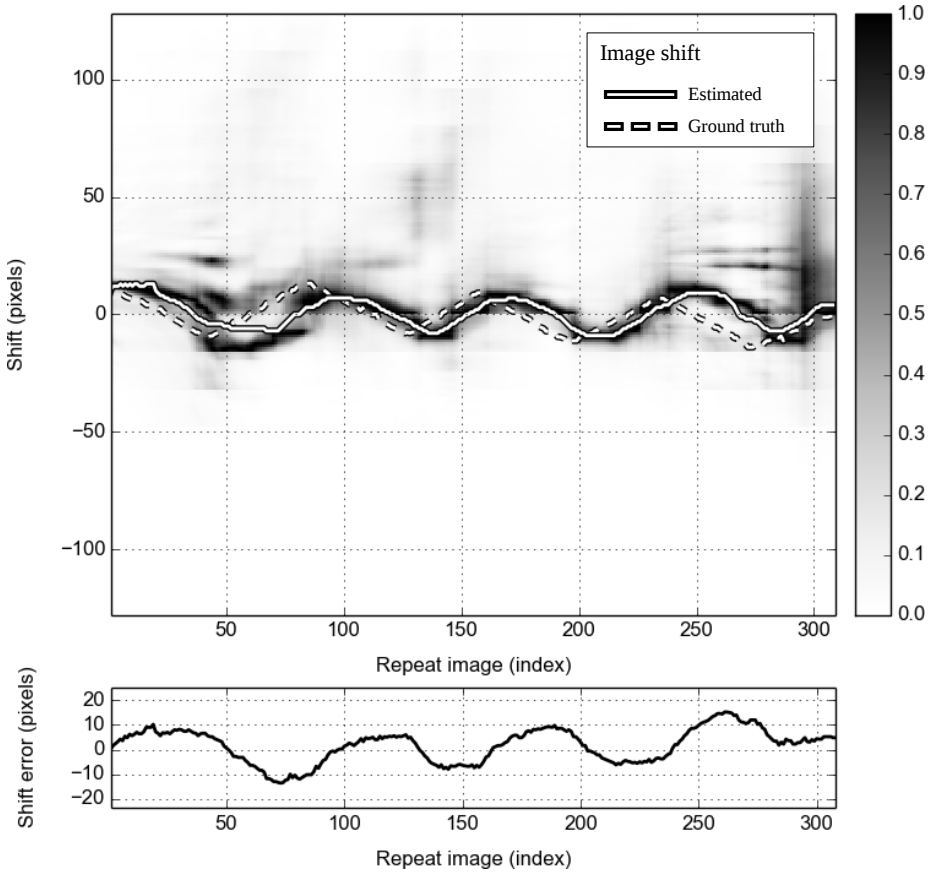


Figure 5.16: Shift map and image shift estimate results for navigation experiment "converge". See description at beginning of the chapter for interpretation details.

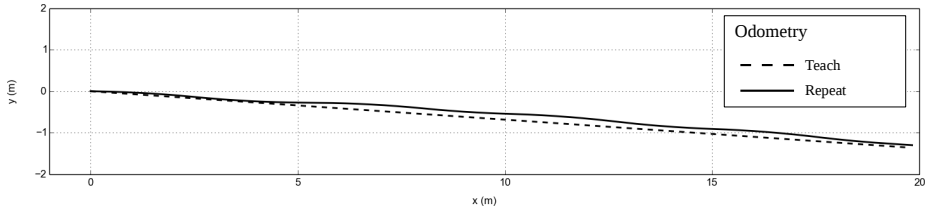


Figure 5.17: Odometry results for navigation experiment "converge". Horizontal axis is position along the length of the corridor, and vertical axis, along its width. Dashed line indicates teach step route, and full line, repeat step driving.

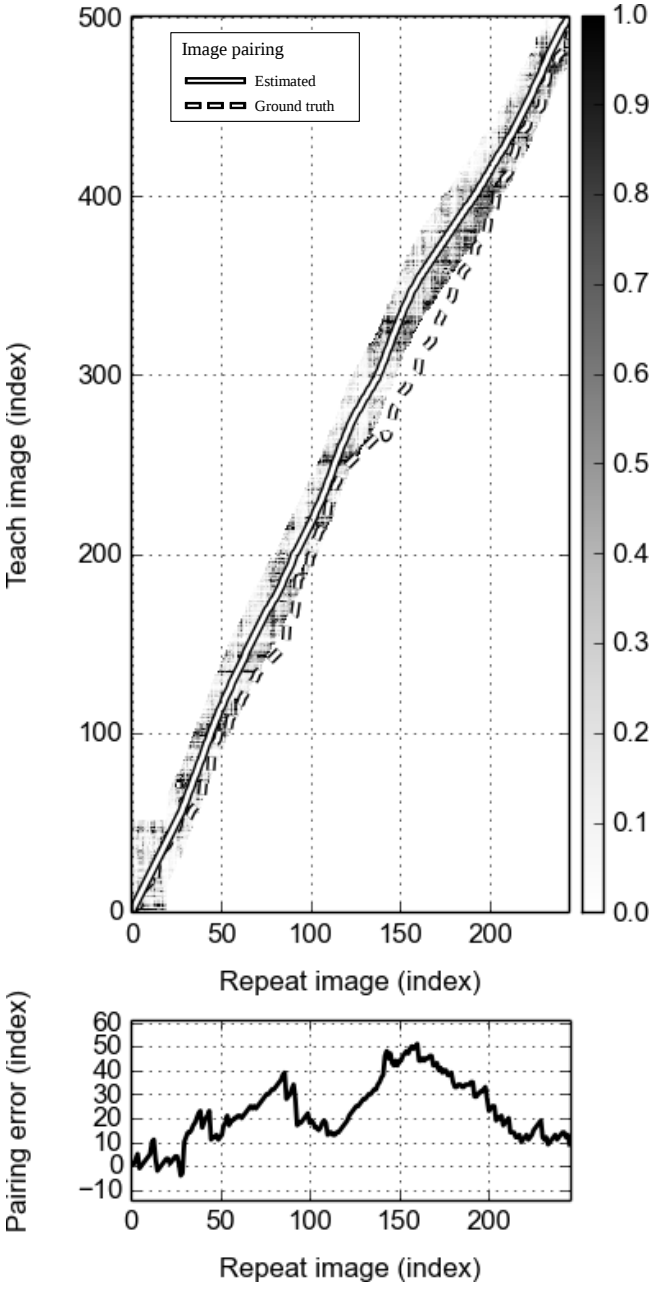


Figure 5.18: Similarity map and image pairing estimate results for navigation experiment "evening". See description at beginning of the chapter for interpretation details.

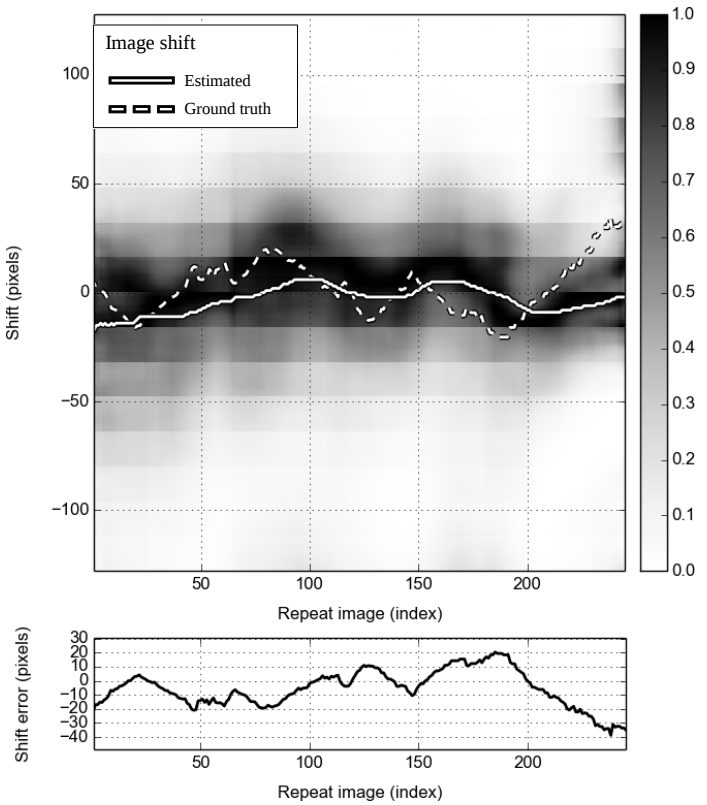


Figure 5.19: Shift map and image shift estimate results for navigation experiment "evening". See description at beginning of the chapter for interpretation details.

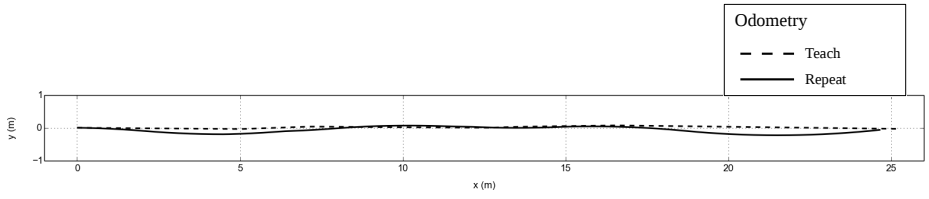


Figure 5.20: Odometry results for navigation experiment "evening". Horizontal axis is position along the length of the corridor, and vertical axis, along its width. Dashed line indicates teach step route, and full line, repeat step driving.

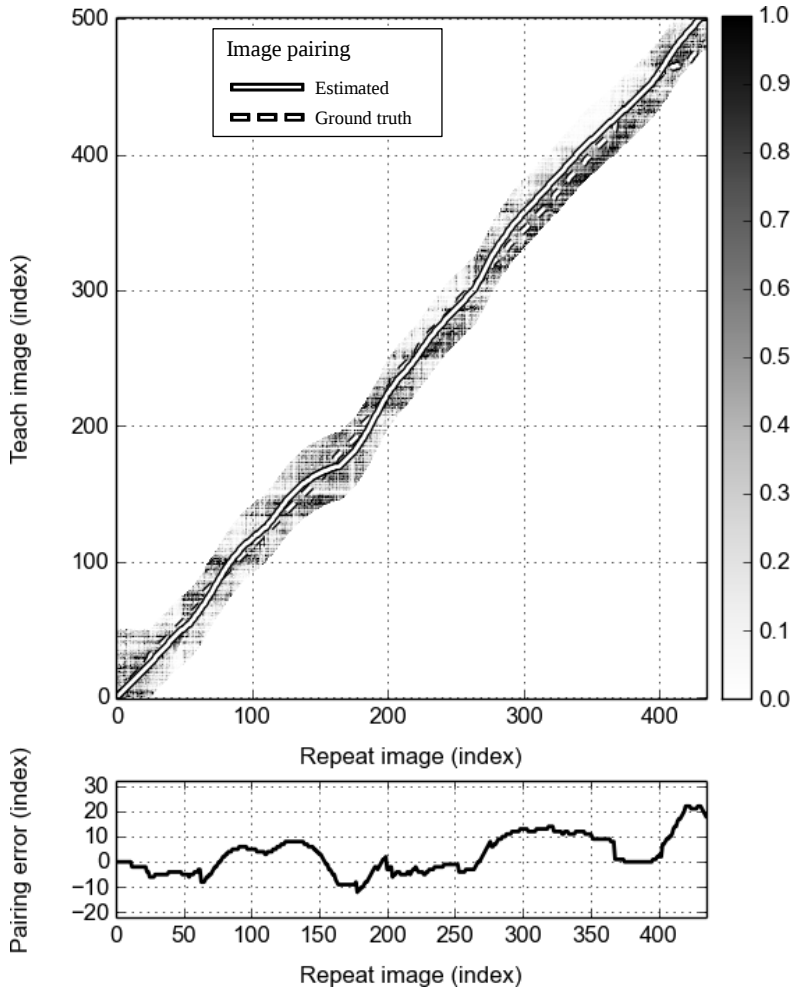


Figure 5.21: Similarity map and image pairing estimate results for navigation experiment "afternoon". See description at beginning of the chapter for interpretation details.

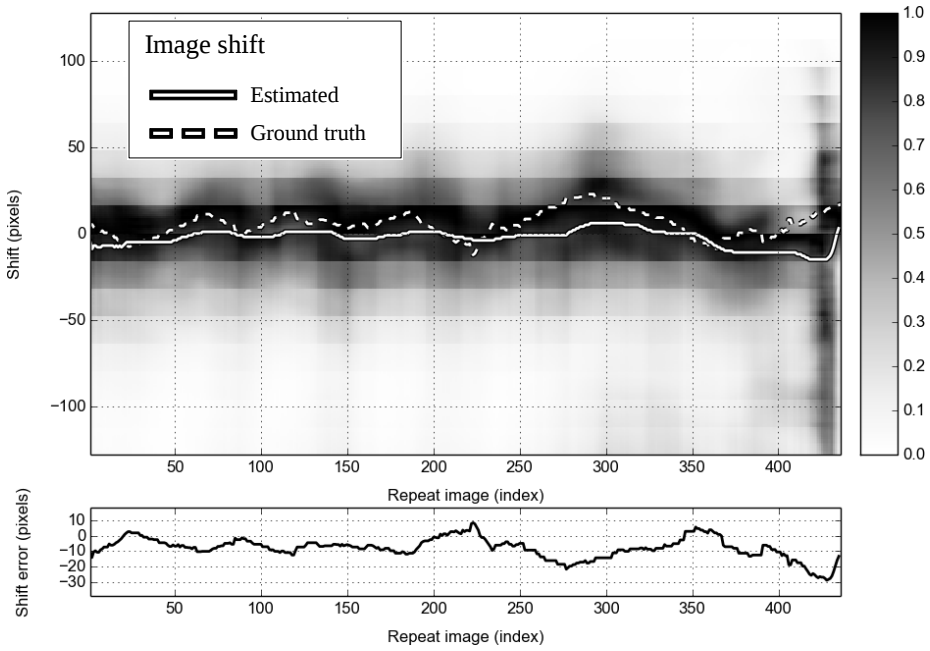


Figure 5.22: Shift map and image shift estimate results for navigation experiment "afternoon". See description at beginning of the chapter for interpretation details.

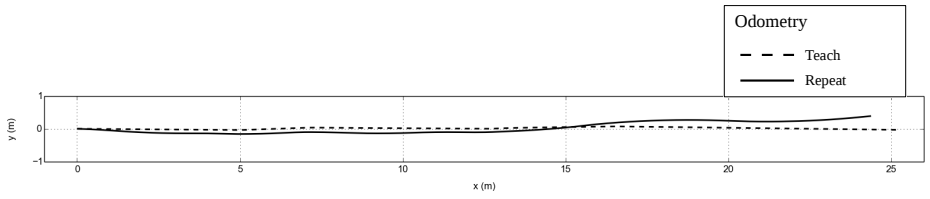


Figure 5.23: Odometry results for navigation experiment "afternoon". Horizontal axis is position along the length of the corridor, and vertical axis, along its width. Dashed line indicates teach step route, and full line, repeat step driving.

5.3 Extremis Experiments

Experiments performed on real-world scenarios demonstrate how DICH performs in practice. However, such cases tend to be simultaneously affected by several different sources of input variation, making an analysis of their relative effects difficult. For that reason an additional set of offline experiments was performed, using video records prepared to only (or at least, mainly) contain specific classes of differences. These make possible to separately evaluate the performance impact of different variations, and determine at which point they may cause a method *breakdown* – a runway divergence between estimates and ground truth.

For each of the input variation classes identified below, several test sessions were prepared. Each session is affected by the given variation to a specific degree. Ground truth data was computed for each session, and the Mean Absolute Error (MAE) between ground truth and estimates for both image pairing and shift was computed. Similarity and shift maps for the smallest and most extreme variations are also shown for comparison. In order to give a fuller account of variation effect, similarity values were computed for the whole range of teach and repeat difference images. Unless otherwise noted, all trips were recorded along the "straight" route in corridor 3D of the indoors environment.

5.3.1 Contrast

In contrast experiments, a reference trip record would be edited by reducing the contrast in record images by a given amount. Then a localization test would be performed between original and edited records. Contrast reduction was performed by the following formula:

$$\mathbf{I}' = \mathbf{I} - c(\mathbf{I} - \bar{\mathbf{I}}) \tag{5.1}$$

Where $\mathbf{I}$ is a record image and $0 < c < 1$ is a contrast reduction constant. This formula has the effect of compressing pixel intensity variations around the mean $\bar{\mathbf{I}}$ by a ratio c; Figure 5.24 shows some examples. Experiments demonstrate it's possible to effect a breakdown of DICH estimates by feeding it images of sufficiently dimmed contrast, however images have to be dimmed to the point that they become almost unrecognizable; contrast differences as high as $c = 0.6$ have little to no effect on the method. Experimental results are shown in Figure 5.25, and shift and similarity maps for $c = 0.6$ and $c = 0.8$ are show in Figures 5.26, 5.27, 5.28 and 5.29 respectively.

(a) Original image (b) $c = 0.6$ (c) $c = 0.8$

Figure 5.24: Contrast deviation test example inputs. (a) Original image. (b) Reduced contrast by ratio $c = 0.6$. (c) Reduced contrast by ratio $c = 0.8$.

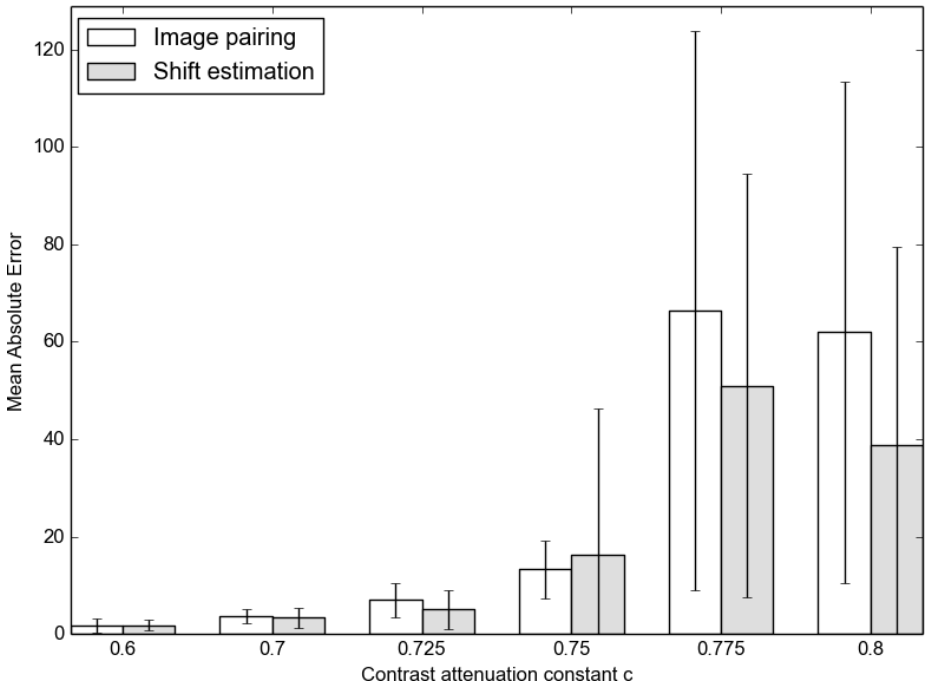


Figure 5.25: Contrast deviation test results.

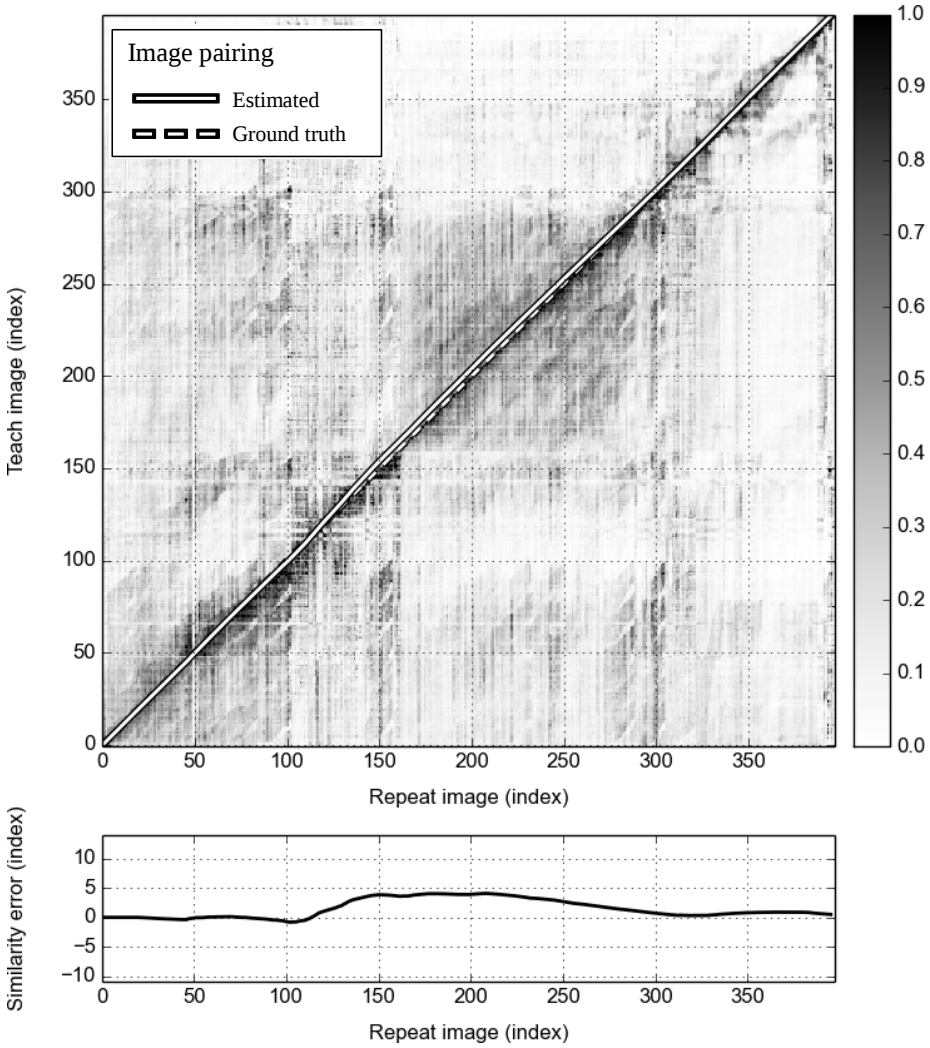


Figure 5.26: Contrast difference $c = 0.6$ similarity map.

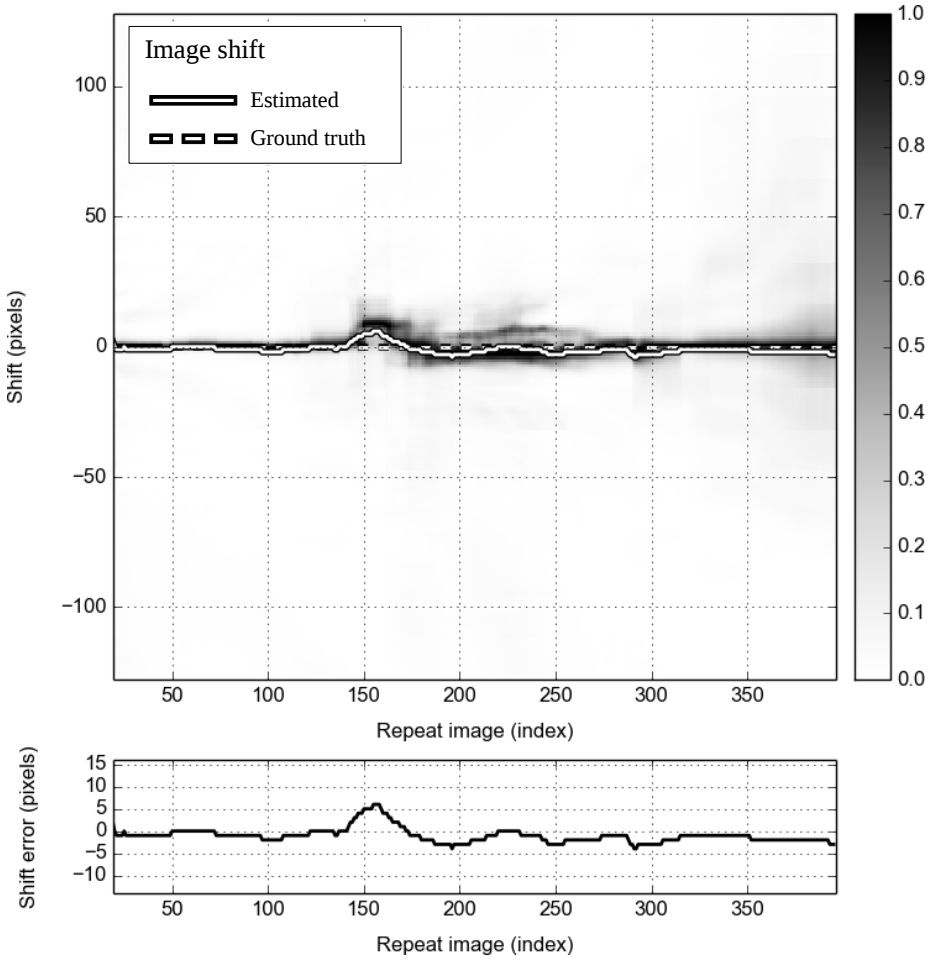


Figure 5.27: Contrast difference $c = 0.6$ shift map.

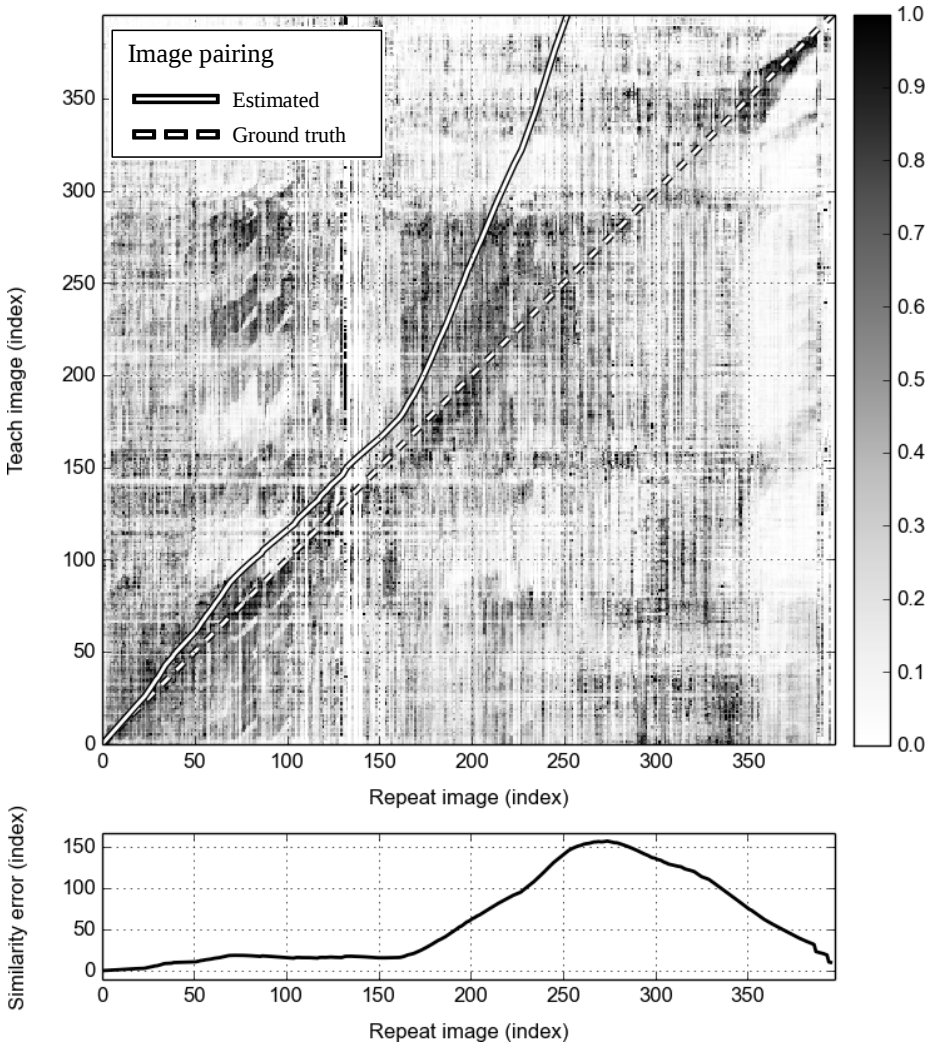


Figure 5.28: Contrast difference $c = 0.8$ similarity map.

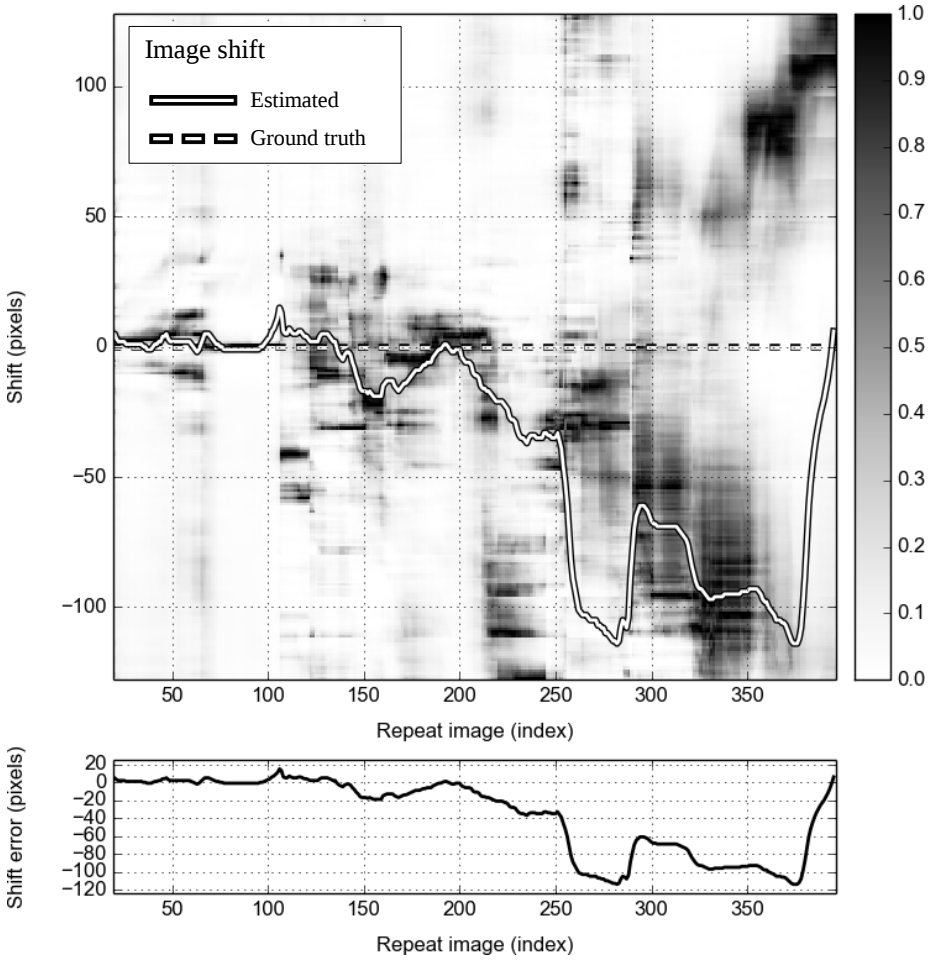


Figure 5.29: Contrast difference $c = 0.8$ shift map.

5.3.2 Occlusion

In occlusion experiments, a reference trip record was edited by adding a random mask of a certain width w (given as a fraction of the visual field's width) at the center of record images, then a localization test was performed between original and edited records. Example images can be seen in Figure 5.30. As in the previous case, experiments show it is possible to effect a breakdown of DICH estimates through occlusion alone, but it requires most of the visual field to be covered. Experimental results are shown in Figure 5.31, and shift and similarity maps for $w = 0.1$ and $w = 0.6$ are show in Figures 5.32, 5.33, 5.34 and 5.35 respectively.

(a) Original image (b) Occlusion $w = 0.1$ (c) Occlusion $w = 0.6$

Figure 5.30: Occlusion test example inputs. (a) Original image. (b) Occlusion mask of width $w = 0.1$ pixels. (c) Occlusion mask of width $w = 0.6$ pixels.

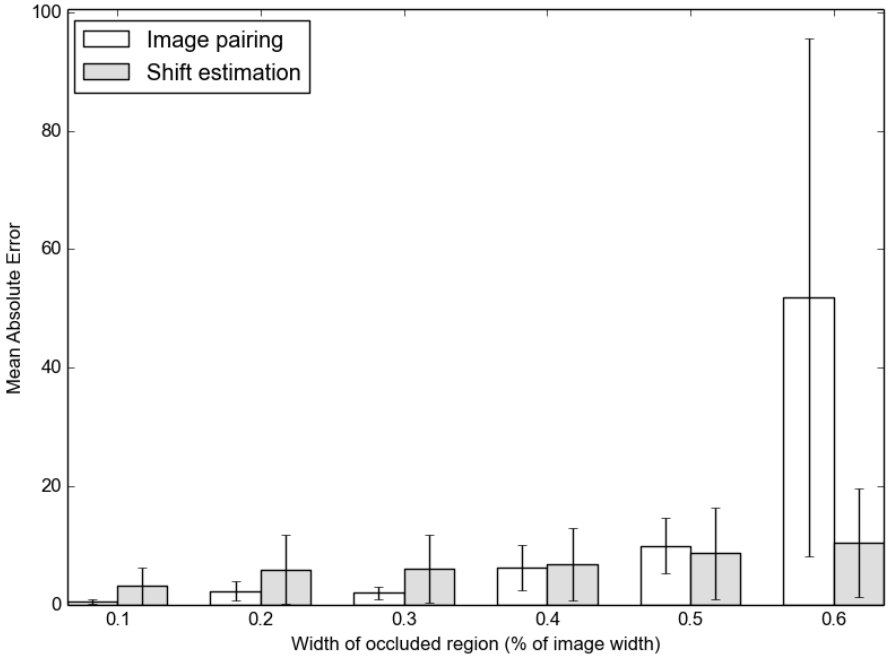


Figure 5.31: Occlusion test results.

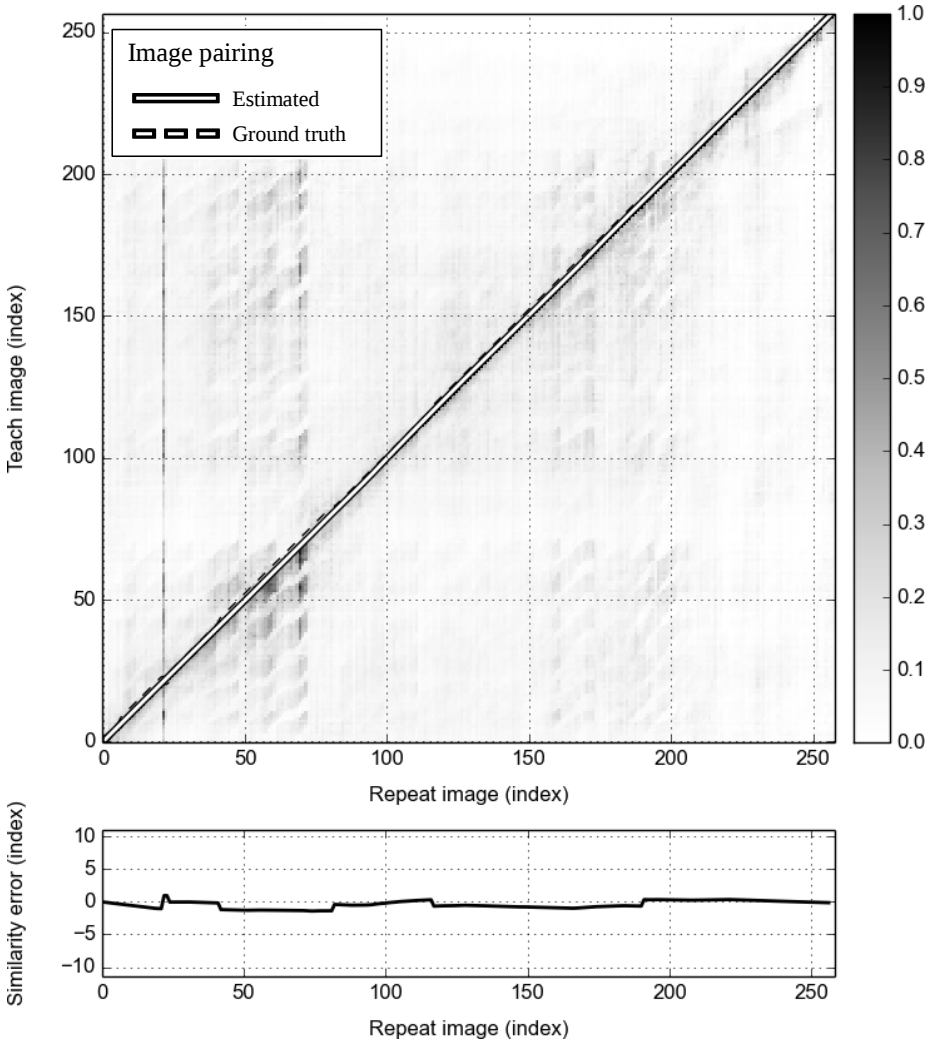


Figure 5.32: Occlusion $w = 0.1$ similarity map.

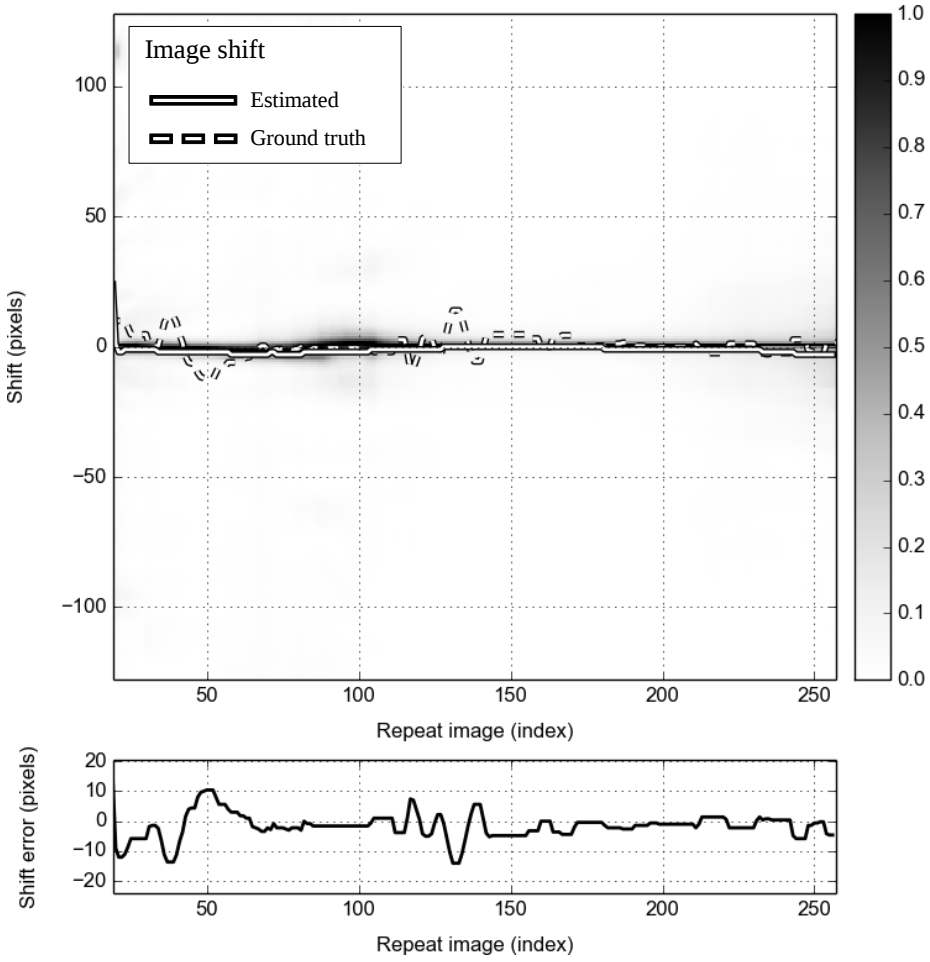


Figure 5.33: Occlusion $w = 0.1$ shift map.

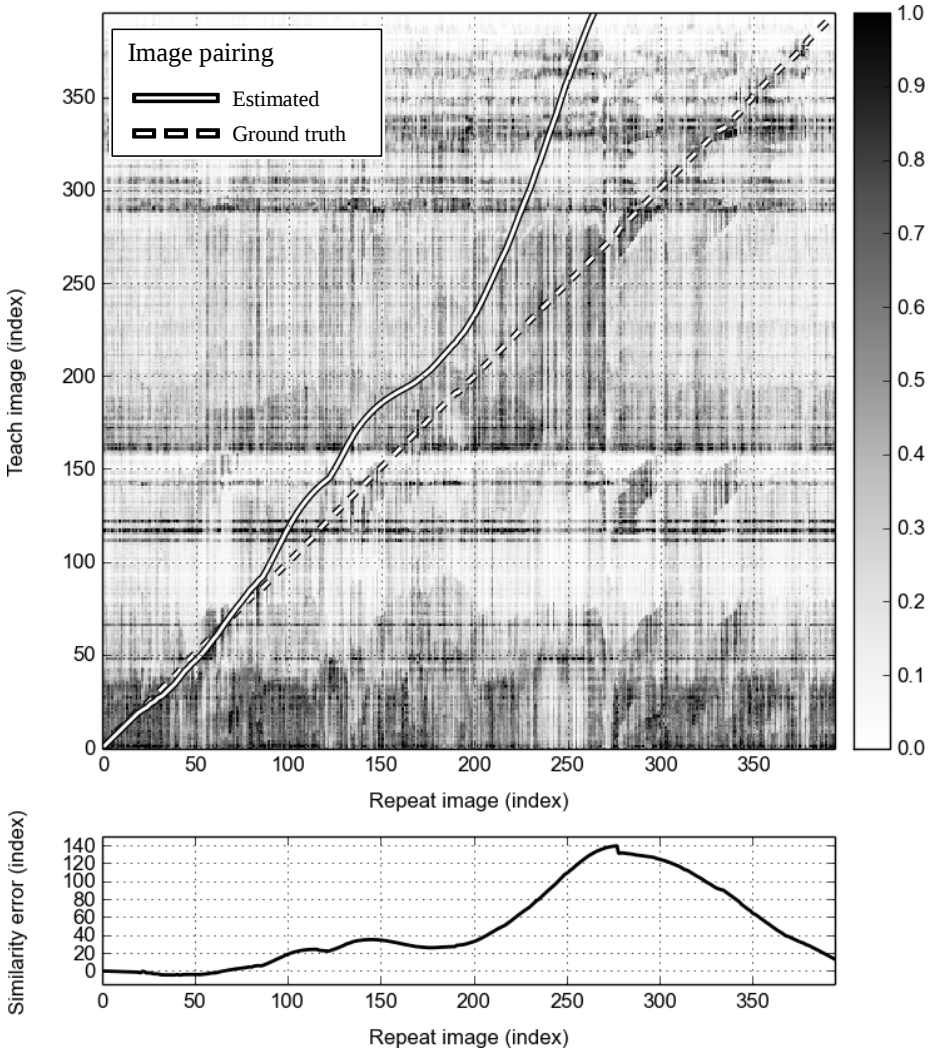


Figure 5.34: Occlusion $w = 0.6$ similarity map.

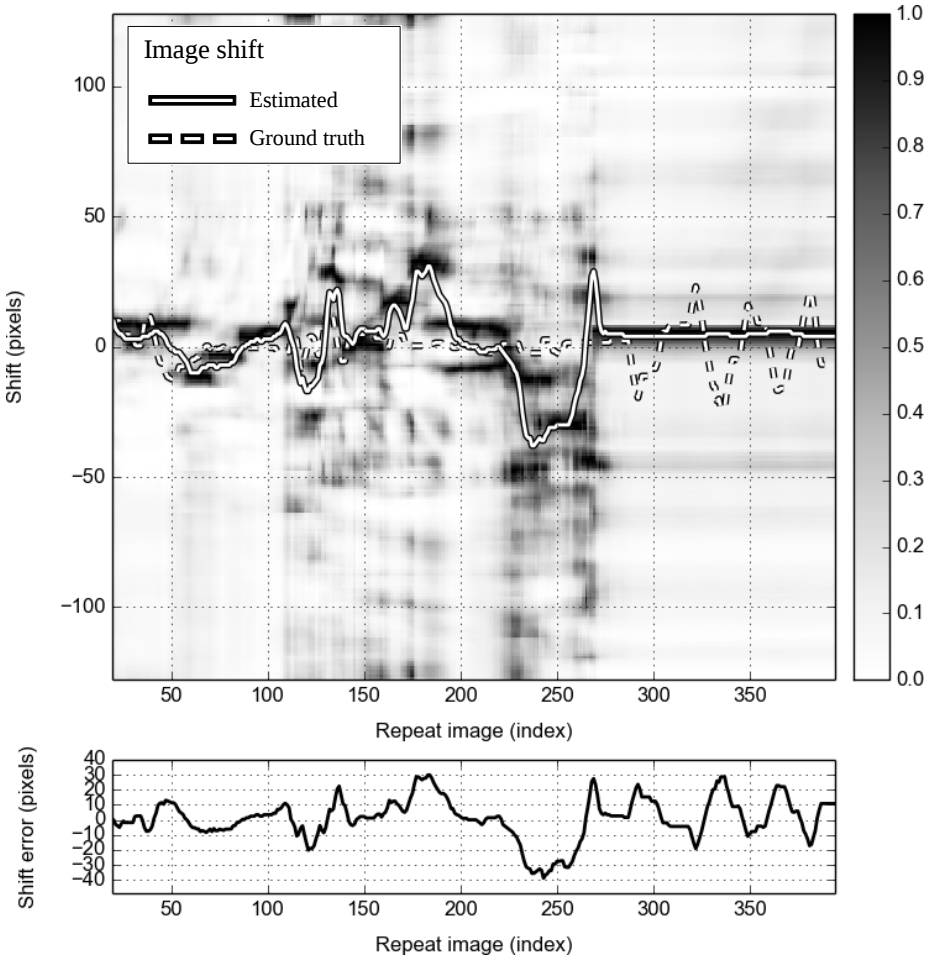
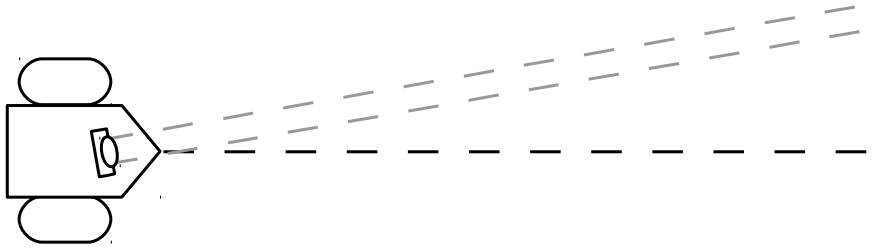


Figure 5.35: Occlusion $w = 0.6$ shift map.

5.3.3 Angle

In angle difference experiments, the robot was driven over a straight route several times. Before each trip the robot's camera would be rotated leftward, deviating from the robot's advancing direction by a set angle, as in Figure 5.36a. Example images collected at different deviation angles are shown in Figure 5.36b-d. DICH proved surprisingly resilient to angle differences, with average error increasing steadily but no breakdown until view angle was all but completely different. Experimental results are shown in Figure 5.37, and shift and similarity maps for 1^o and 30^o difference angles are show in Figures 5.38, 5.39, 5.40 and 5.41 respectively.



(a) Angle deviation setup

(b) 0^o deviation

(c) 5^o deviation

(d) 10^o deviation

Figure 5.36: Angle deviation test setup. (a) Before each test the camera on top of the robot is turned a certain amount, so its gaze (gray dashed lines) will deviate from the robots movement direction (black dashed line) by an specific angle. (b-d) Example images collected at different deviation angles.

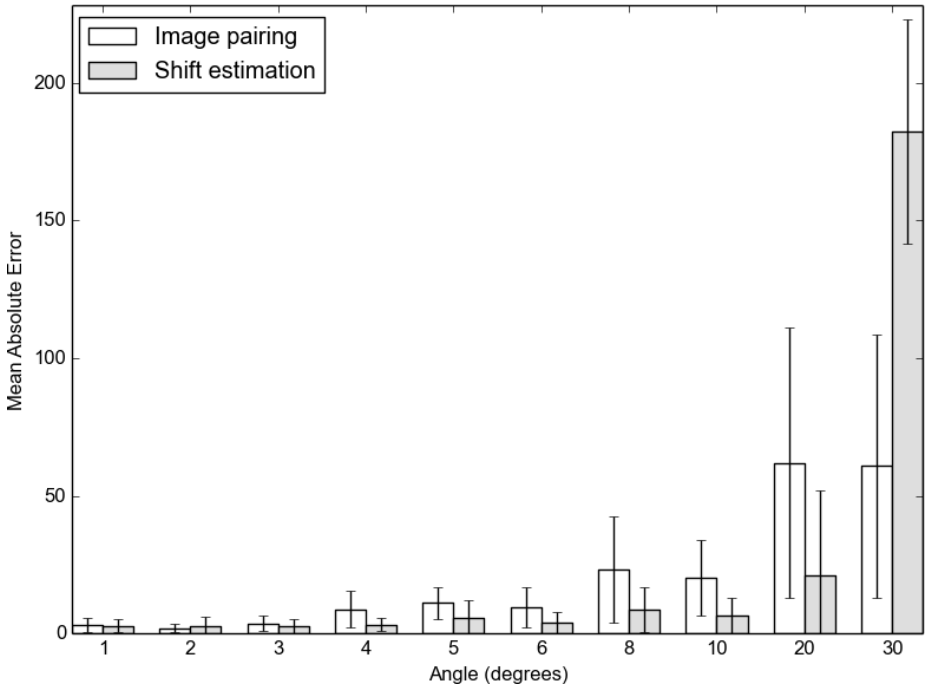


Figure 5.37: Angle deviation test results.

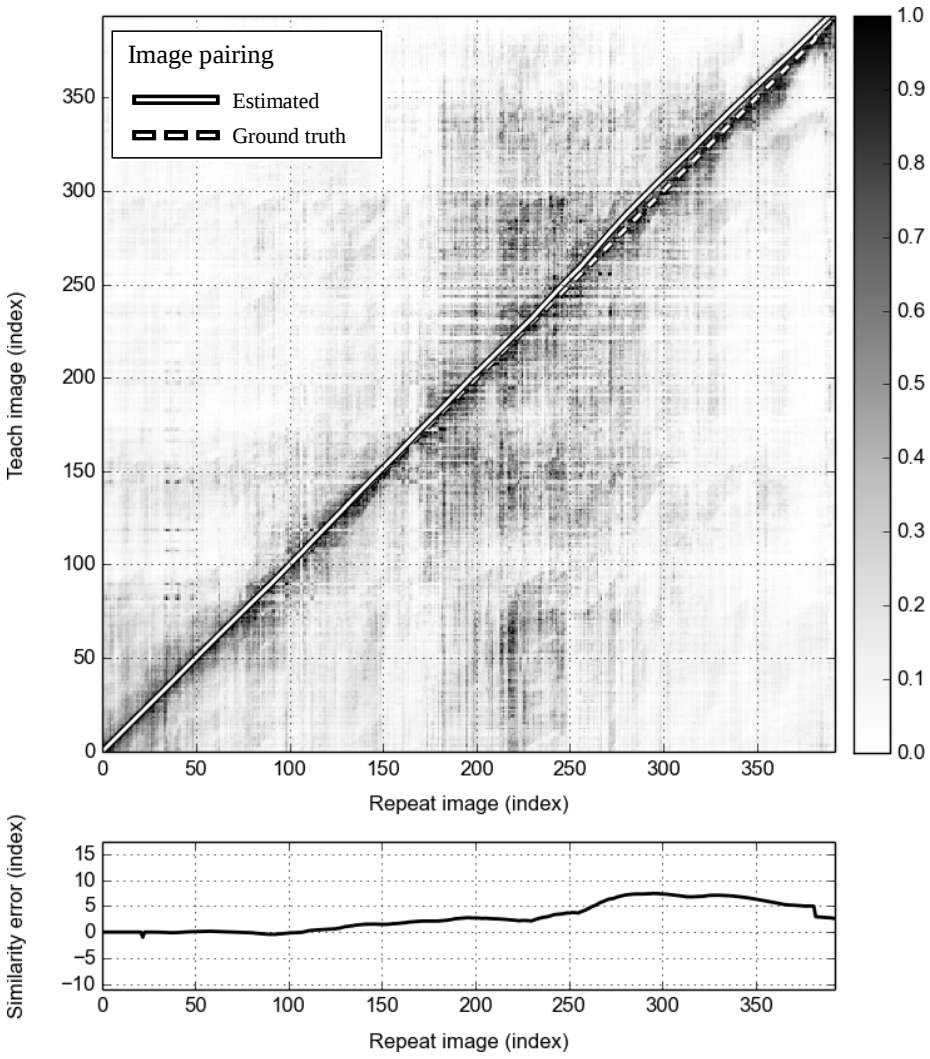


Figure 5.38: Angle difference 1^o similarity map.

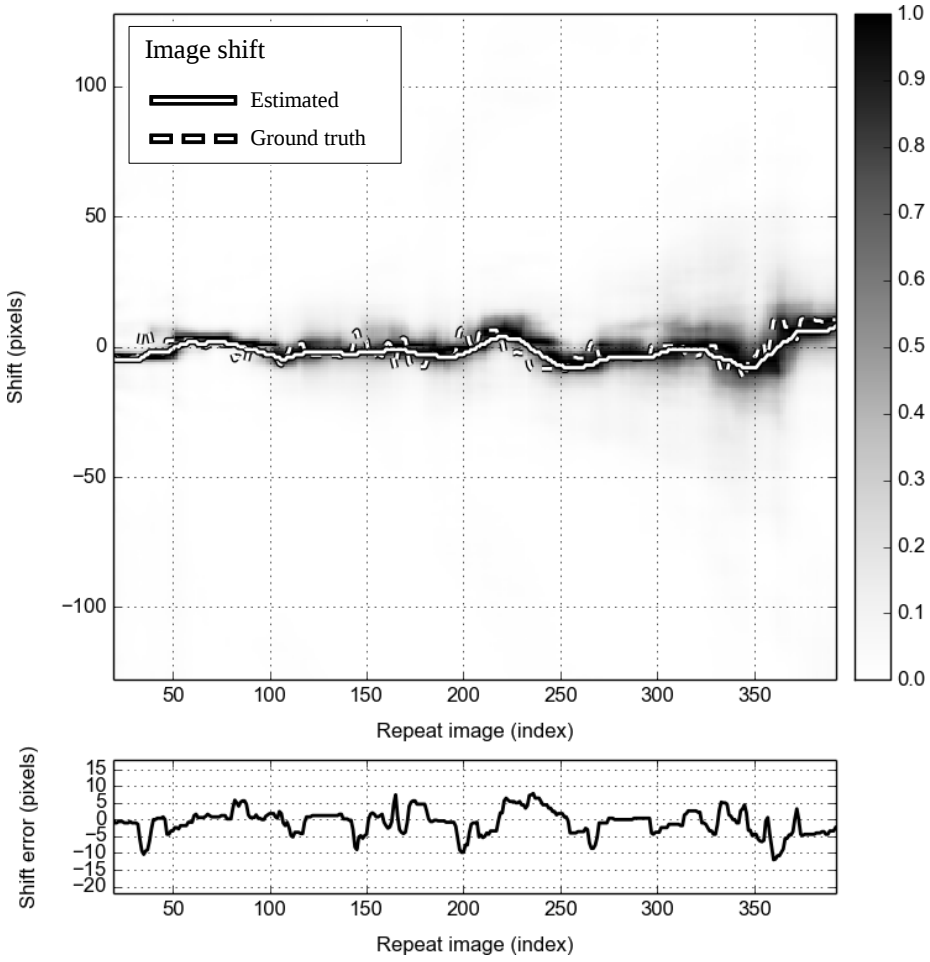


Figure 5.39: Angle difference 1^o shift map.

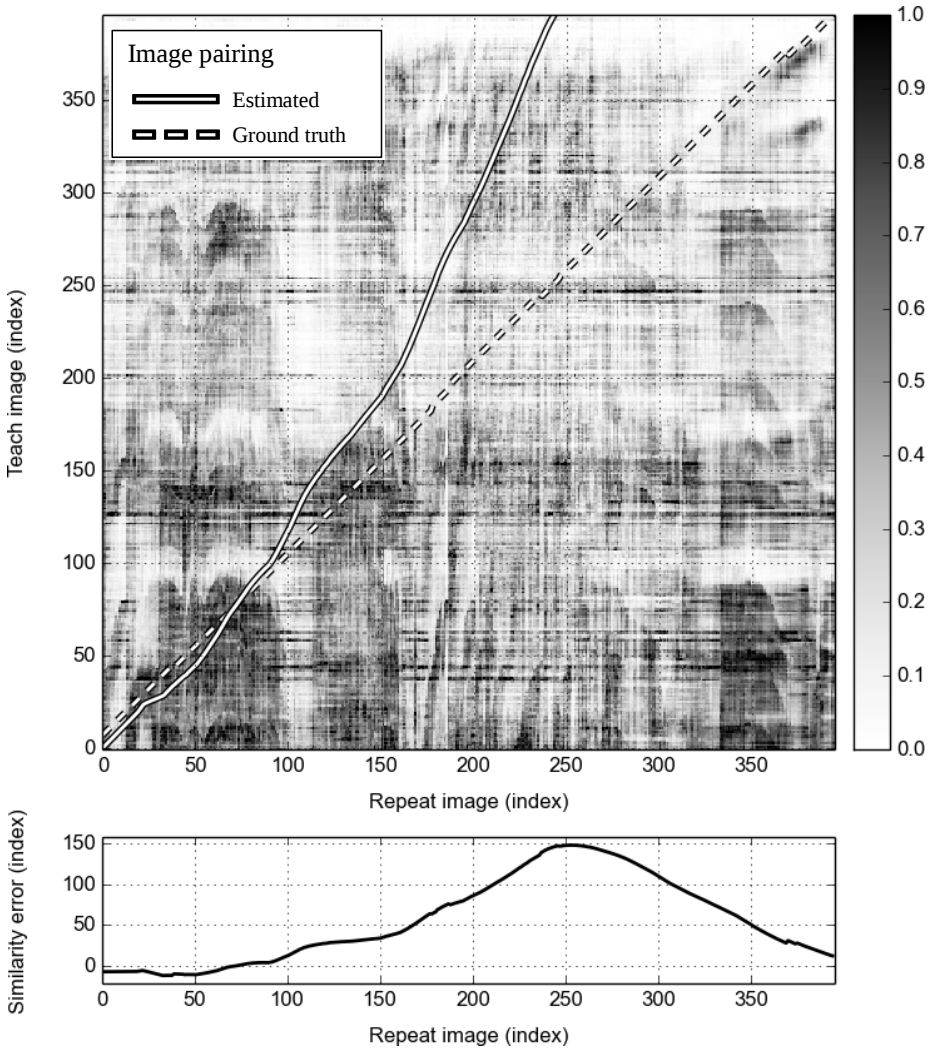


Figure 5.40: Angle difference 30^o similarity map.

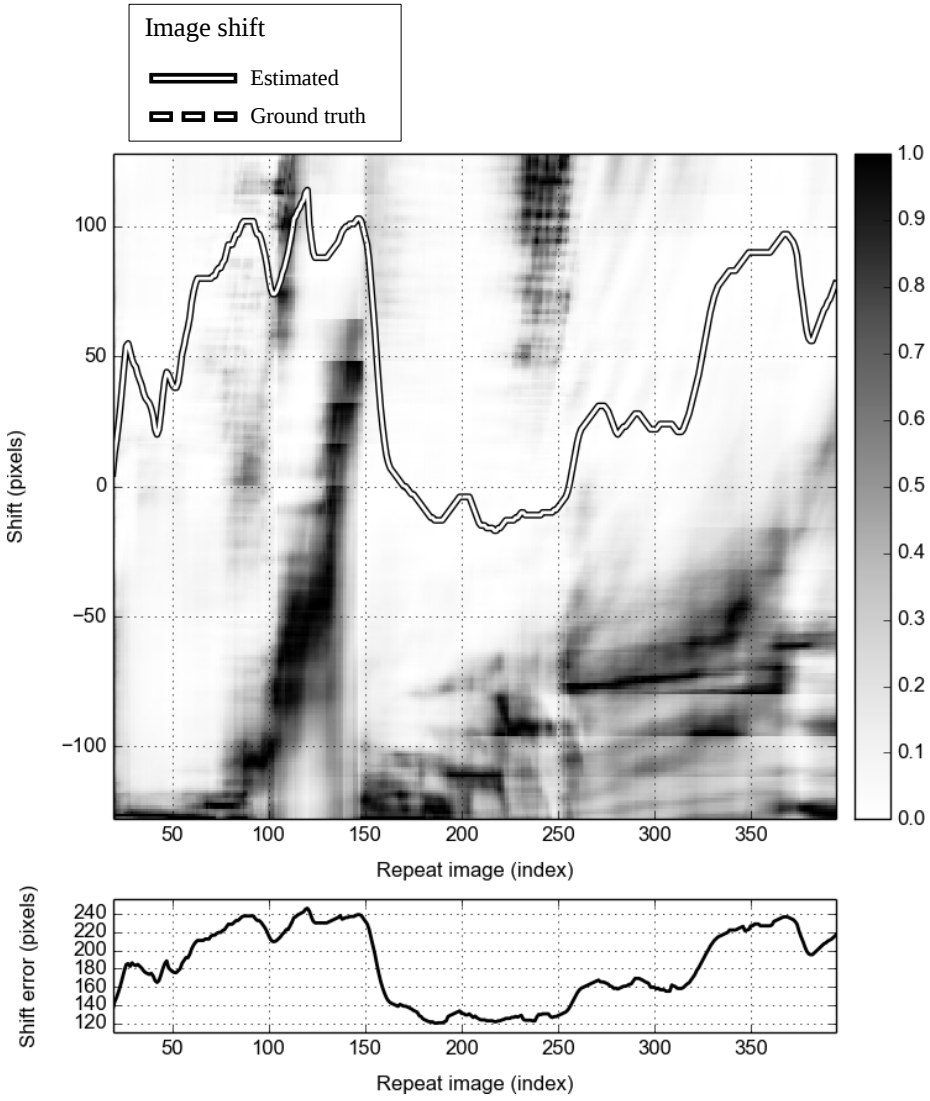
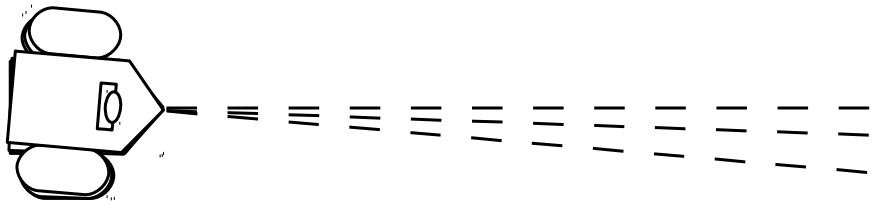


Figure 5.41: Angle difference 30^o shift map.

5.3.4 Direction

While in angle difference experiments the robot would always advance in the same direction, but "look" at different angles, in direction experiments the robot always "looked" forward, but advanced at different angles relative to the corridor's walls; see Figure 5.42a for an illustration. Because performing this experiment in a corridor would cause the robot to run into walls too quickly, a larger environment was selected for it: the entrance hall at the 2^{nd} floor of building 3B. Figure 5.42b-d shows some pictures taken from it. Again DICH proved surprisingly resilient against changes in direction, breaking down only at the largest direction differences. Experimental results are shown in Figure 5.43, and shift and similarity maps for 1^o and 30^o direction differences are show in Figures 5.44, 5.45, 5.46 and 5.47 respectively.



(a) Angle deviation setup

(b) 0^o deviation

(c) 10^o deviation

(d) 30^o deviation

Figure 5.42: Direction deviation test setup.

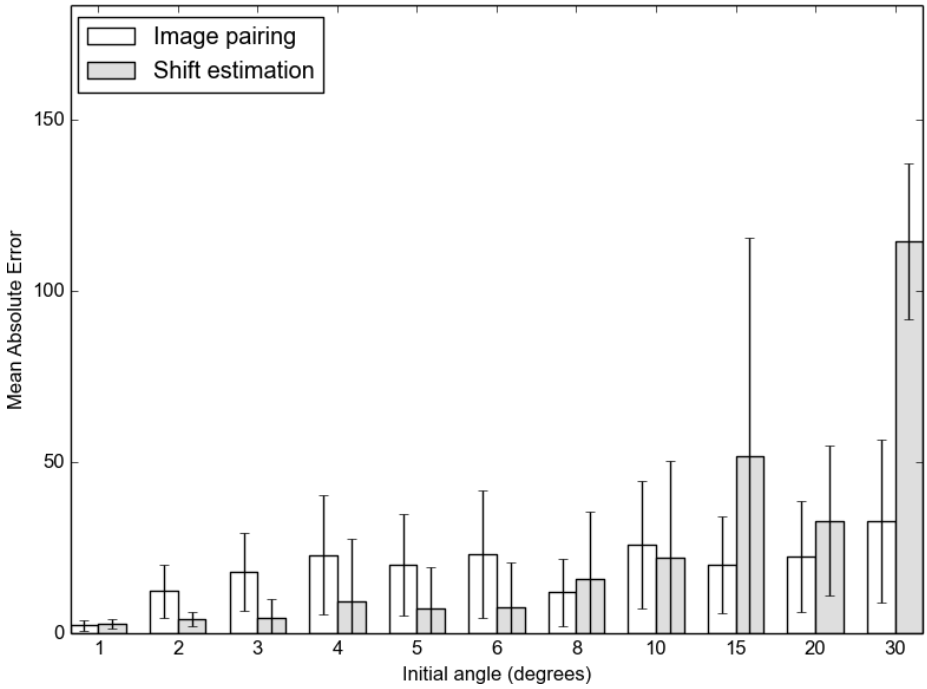


Figure 5.43: Direction deviation test results.

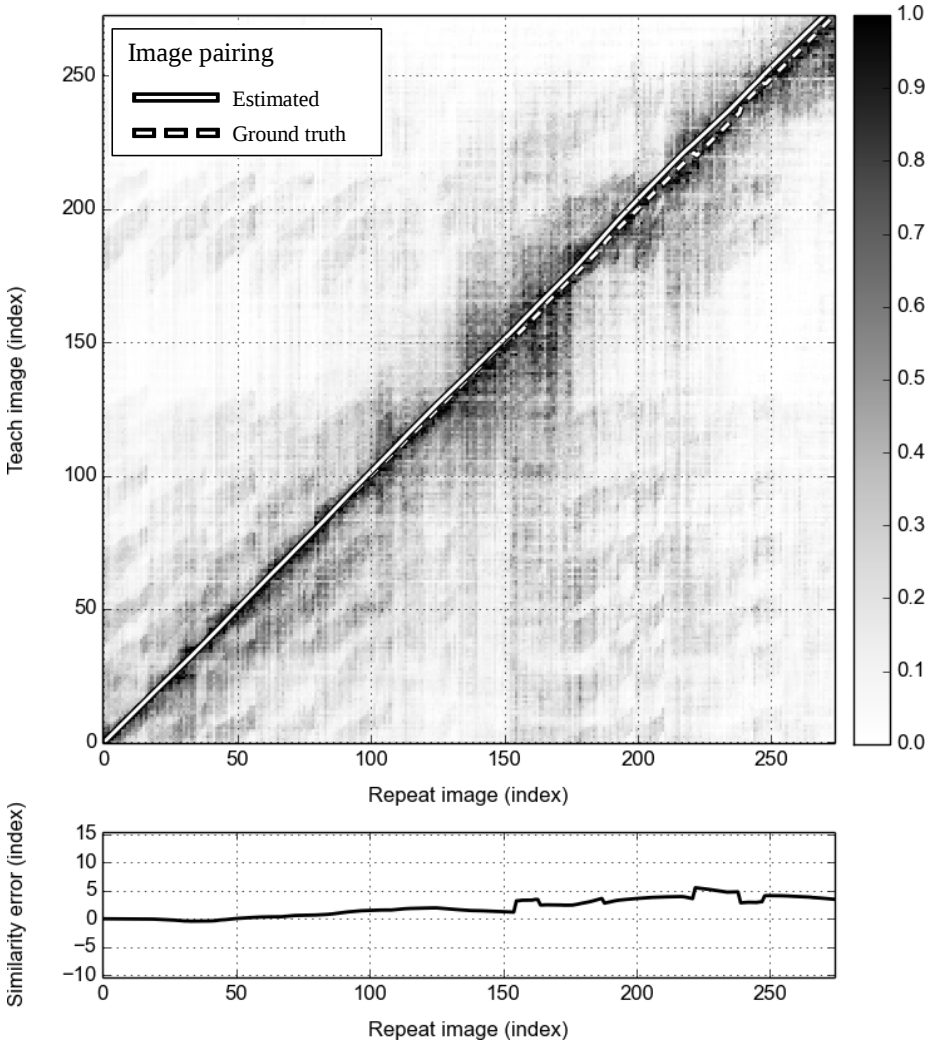


Figure 5.44: Direction deviation 1^o similarity map.

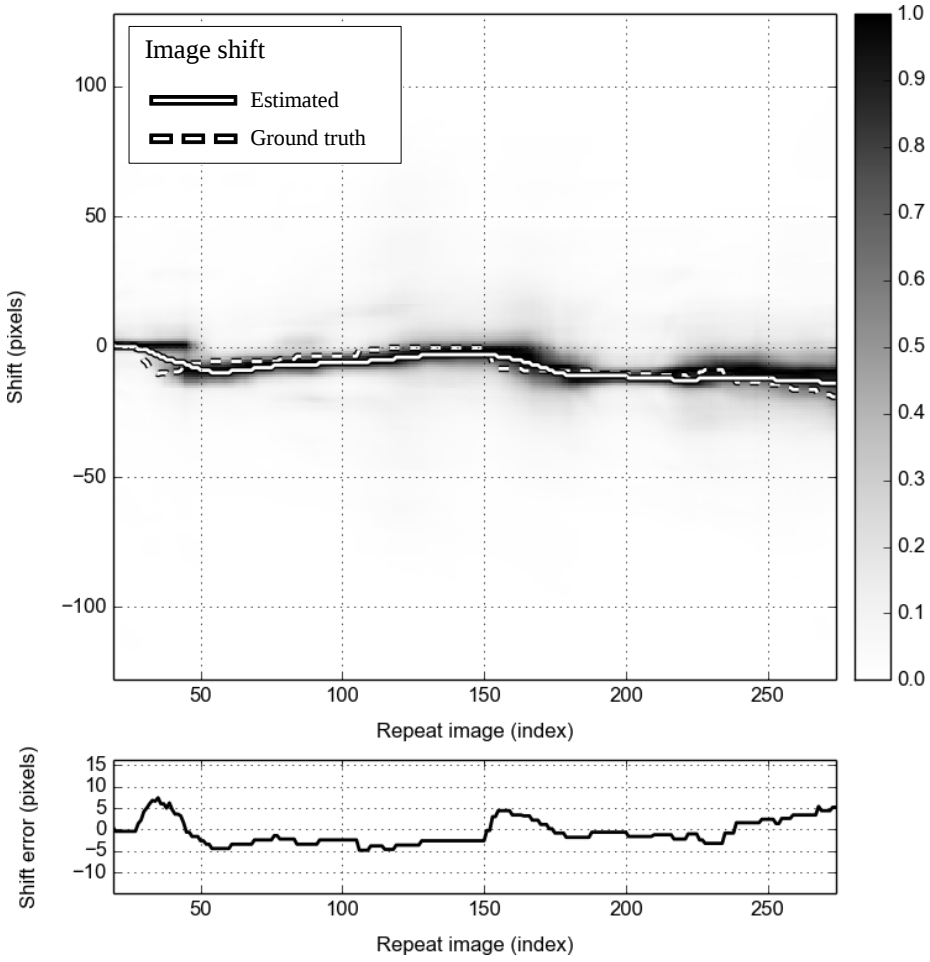


Figure 5.45: Direction deviation 1^o shift map.

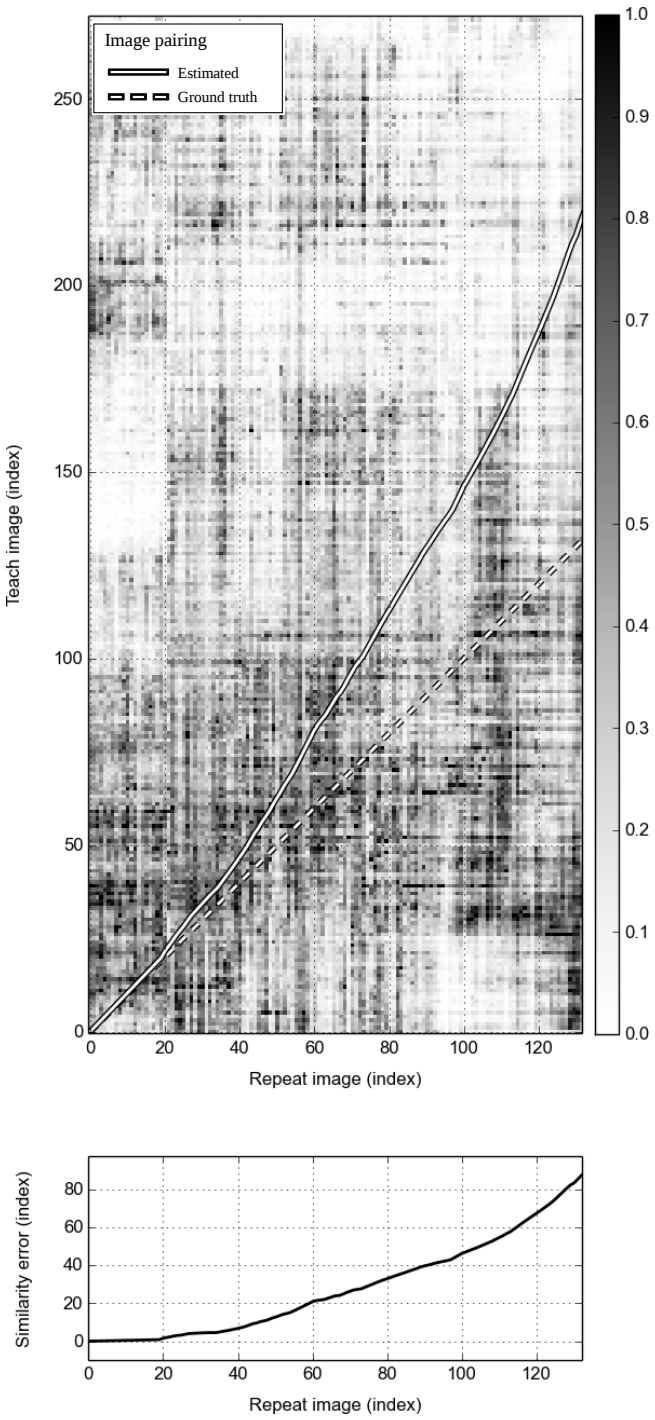


Figure 5.46: Direction deviation 30^o similarity map.

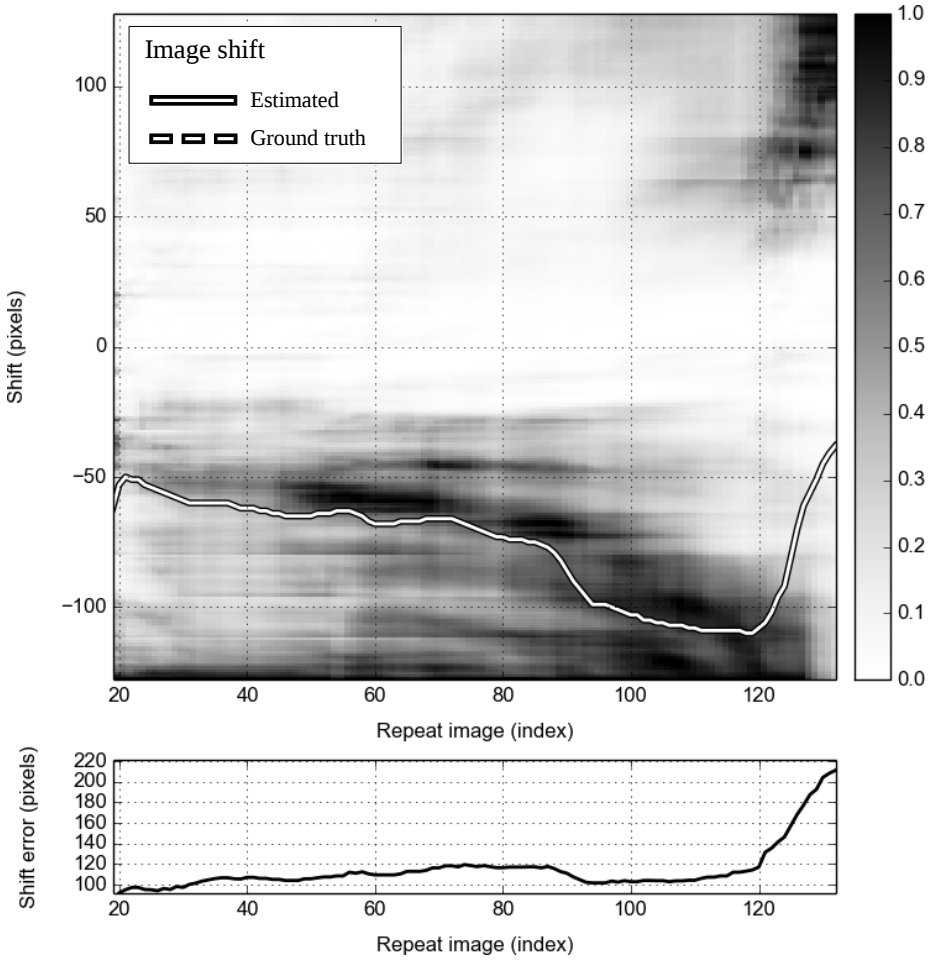


Figure 5.47: Direction deviation 30^o shift map.

Chapter 6

Discussion

In this chapter the experimental results previously reported are discussed, being interpreted in terms of DICH features. From this the architecture's strengths and weaknesses are derived. The chapter concludes with a discussion on possible extensions and further research.

6.1 Interpretation of Results

In judging the experimental results reported in Chapter 5, it may be more convenient to start from the end, looking first into the extremis experiments. These measure the effects of input variations across specific modes – namely, image contrast, visual field occlusion, observation angle and direction of advancement. Results show that as long as route differences are kept to a minimum, DICH can withstand large input variations over any one mode and still produce consistent localization estimates. Resilience against each variation mode can be linked to specific DICH features.

Difference images are largely invariant to contrast differences, thus accounting for much of the robustness observed in contrast experiments. What differences get through are further abstracted by the use of normalized cross-correlation, which matches image elements more by their "shape" (i.e., the phase component of the visual signal) than by their "values" (the magnitude component). Only when contrast is reduced by 60% do the loss of visual information starts to affect the method, and it's not until it's reduced by 80% that it actually breaks down completely.

The use of Regions-of-Interest enables DICH to gracefully work around occlusions: ROI's assigned to occluding objects will fail to produce good matches, and similarity estimates will be dominated by those ROI's that fell on visible landmarks. As long as the occluded area isn't large enough to fit most ROI's, this remains true even if the occluding objects are more salient than the rest of the image: what ROI's can't fit inside the occluded area will necessarily fall on valid landmarks.

Angle differences are trivially accounted for by the use of visual search on both the image pairing and shift estimation steps of DICH. While normalized cross-correlation does not account for viewpoint yaw (i.e., rotation around the vertical axis), on average its effects were insufficient to compromise the overall method. The bias towards consistent trends in the computation of final pairing and shift estimates also contributed for DICH's robustness at times when estimates became ambiguous.

Finally, direction differences are accounted for by the combination of Regions-Of-Difference, visual search and trend estimation. The use of ROI's reduce the impact of relevant visual elements "falling off" image borders due to large viewpoint differences, while visual search and trend estimation make possible to make sense of even poor matches from landmarks recorded from different angles.

All these features contribute to enable successful localization and navigation, as seen in the respective experiment sessions. The results for the "converge" navigation experiment were especially positive: even after initially moving towards a different direction, the robot was able to quickly converge towards the teach route and remain close to it. This indicates that the sort of large direction differences engineered in some of the localization experiments are unlikely to occur during navigation, since any deviation would be corrected well before it was allowed to accumulate.

6.2 Contributions

The Difference Image Correspondence Hierarchy (DICH) is a monocular appearance-based Visual Teach & Repeat (VT&R) method for indoors and outdoors environments, resilient to illumination changes and the presence of moving elements. It can perform localization and route-following, and can deal with the kidnapped robot problem so long as the robot is left somewhere along a known path. It occupies a rare niche in the VT&R literature, where all these features are seldom seen together in a single system.

DICH is image-based, avoiding weaknesses related to visual features. It is also virtually unique among VT&R systems in its explicit use of record history to achieve estimation consistency – ambiguous and incorrect estimates are weeded out by selecting sequences of estimates consistent over time. In most systems this is a consequence from other constraints (such as the probabilistic prediction-updating of the Monte Carlo approach [33]), but in DICH it is stated explicitly from the beginning, and is at the heart of most method parts. Therefore, DICH constitutes a comprehensive VT&R navigation solution for robots complying to the most general specifications.

6.3 Strengths and Weaknesses

DICH presents several attractive features for system integrators building mobile robotics solutions. The reliance on a single uncalibrated camera for visual input makes it suitable to a large number of already-deployed platforms, requiring at most a relatively inexpensive camera installation. Its ability to perform localization and route following along known paths covers many useful use cases, the teach-and-repeat approach makes it simple to configure. Its robustness makes it applicable in a wide variety of environments. In short, DICH constitutes a comprehensive Visual Teach & Repeat (VT&R) navigation solution for robots with minimal specification requirements.

Moreover, DICH's approach to the VT&R problem strikes a favorable balance between performance and theoretic principles. Starting from a few assumptions (consistent robot location over time, inputs correlating to movement) and applying a set of common concepts (ROI's, visual cross-correlation, trend estimation) it manages to construct and architecture that is both computationally light and robust. The use of concepts from human cognition and neuroscience opens the possibility that DICH may eventually be applied to support research in those fields.

Nevertheless DICH is not without its shortcomings. As currently defined, the method includes a large number of free parameters, most of which lack a clear estimation method. The current, empirically constructed value set proved sufficiently accurate for experimental purposes, however it would still be preferable if they could be optimized automatically from data sets, or at least estimated from definite procedures. DICH also suffers from a rather limited environment representation model. Its one-dimensional formulation may be sufficient for location across known routes, but is unable to perform loop-closing or otherwise locate the robot in bidimensional space.

6.4 Extensions and Further Research

DICH environment representation is purportedly modeled after human cognitive maps, which are composed of *places* characterized by *landmarks* and connected through *paths*. Difference images and their contents implicitly represent places and landmarks respectively, while image order over a record stands in for paths. What this model misses, however, is the possibility of branching paths: each place can at most have one place "before" and another "after" it. This could be addressed by stitching together trip records depending on departure and destination places, as illustrated in Figure 6.1.This would in turn require a mechanism for managing teach records, planning paths in terms of record sequences and combining selected records.

DICH can solve the kidnapped robot problem so long as the post-kidnapping position is close enough to a known path. However this is a time-consuming task; moreover, the question of path record selection hasn't been addressed, with records being selected manually in localization experiments. Both issues

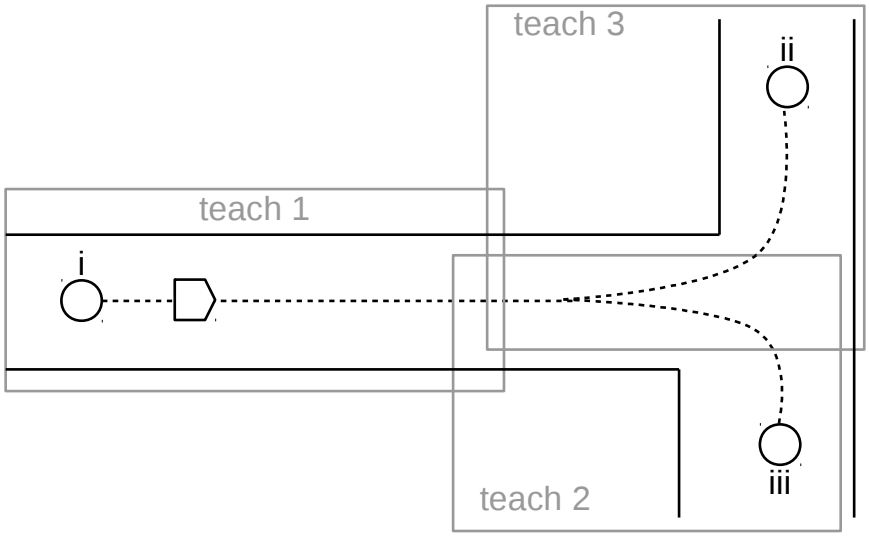


Figure 6.1: Implementing branching paths in DICH. Three teach records cover intersecting paths over an environment. A repeat trip going from place (i) to either places (ii) or (iii) could be performed by stitching either record 2 or 3 at the end of record 1.

should be addressed in future research.

Work on DICH parameters should be a priority for further research. This could be approached from three fronts: reducing the number of parameters, perhaps by having some parameters be derived from others; methods for parameter optimization; and a review of human vision research to select "best" values from psychophysiological experiments. This last option would have the added benefit of possible new insights for further method improvements. Another goal for future research should be expanding environment representation into a bidimensional model, and add the ability for performing loop-closing detection.

Chapter 7

Conclusion

The Difference Image Correspondence Hierarchy (DICH) is a biologically inspired autonomous navigation method for differential drive robots operating under the Visual Teach & Repeat paradigm. Its name comes from the use of difference images as basic percept, searching correspondences between image contents as main information processing operation, and a modular structure organized as a hierarchy.

Autonomous navigation is the process by which a robot moves between departure and destination points without further human control. It is subject to a number of constraints, such as safety and effectiveness. At the bare minimum, any robot navigation method should include some sort of environment representation, and a way to locate robots relative to it. A wide variety of proposed solutions to both problems exist, to the point that it may actually be difficult to relate them all in a general model. DICH addresses this point by taking a biologically inspired view of the localization problem. Self-location is a fundamental cognitive skill for living beings, therefore its study can provide relevant insights and suggest methodological approaches for robotic localization, and by extension autonomous navigation.

Visual processing in DICH is based on concepts from visual cognition: difference images mirror the movement bias of visual receptors, aROI's are inspired in fixation points and scanpath theory, and image correlation takes after convolution-based neural processing models. Spatial cognition meanwhile provides a basic framework for representing environments as collections of places, characterized by landmarks, and connected through paths. Crucially, in accordance to the premises of embodied cognition, none of these concepts are directly represented in the architecture: landmarks influence the contents of difference images, their differences ensure image pairing will correlate to robot position along places, and image ordering in memory follows the position of respective viewpoints along paths, but never does DICH equates these proxy inputs to the physical quantities from which they originate. This is an important philosophical aspect, which is not aways observed as it should.

Being completely reliant on visual information for localization, DICH can be

classified along appearance-based navigation methods, which are ideally suited for Visual Teach & Repeat (VT&R) navigation. VT&R navigation is performed in two steps, called "teach" and "repeat". During the teach step a robot is driven over a route by a human operator, collecting a video record of the route in the process. Then in the repeat step it must retrace the route, using the video record as reference. DICH is a monocular image-based method shown to be effective both indoors and outdoors. It can perform localization and route-following along, and can deal with the kidnapped robot problem so long as the robot is left somewhere along a known path. It occupies a rare niche in the VT&R literature, where all these features are seldom seen together in a single system.

DICH navigation can be divided in four steps. First, difference images are computed from both teach step video records and live camera images: these encode changes to visual input over time. Teach and repeat difference images are compared to establish an image pairing trend. Image pairs are then inspected for shift, the apparent sliding of one image's visual elements relative to the other. Finally pairing and shift information are used to steer the robot. These four steps are repeated in sequence until the robot reaches its destination.

DICH was implemented in C++, as a collection of Robot Operational System (ROS) nodes, running atop an Intel computer connected to a differential drive robot. It was subjected to three different kinds of experiment, dubbed localization, navigation and extremis. Localization extremis sought to demonstrate the method's capacity for localization under different conditions, both indoors and outdoors. Navigation experiments evaluated the effect of steering feedback into the method. Finally extremis experiments checked at which point specific forms of input variation would cause the method to break down.

Currently the method's main weakness is the need to set parameters manually for optimal performance. However, since it assumes a record of a previous trip is available before operation starts, it may be possible to implement analysis methods to estimate good initial values for image matching parameters. Extending the system environment representation is also a topic for future work.

Appendix A

Mathematical Constructs

A.1 Lists

A list is "a linearly ordered collection of values of the same general nature" [58]. A list is denoted by a bold-faced lowercase letter, e.g., $\mathbf{a}$. The contents of a list are denoted by a sequence of comma-separated elements surrounded in square brackets, e.g., $[1, 2, 3]$ or $[a, b, c]$. Lists can contain other lists, e.g., $[[1, 2], [3, 4, 5]]$ is a list containing two lists, one with two and the other with three elements. The empty list is written $[\,]$, and the singleton list (containing just one element a) is written $[a]$. Unlike a set, a list may contain the same element more than once: the list $[a, a]$ contains two elements of same value, and is different from the list $[a]$ which contains only one. Besides being denoted explicitly, lists can also be defined as *generative expressions*: the expression $[i \mid 0 \leq i < 10]$ is the list of all integer values from 0 to 9, inclusive.

The length of a list $\mathbf{a}$ is the number of elements it contains, and is denoted $|\mathbf{a}|$. Thus for a list $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}]$ containing n elements, $|\mathbf{a}| = n$. Individual elements are referred by following the list letter with an offset enclosed in square brackets, so for example $\mathbf{a}[k] = a_k$. Alternatively a list item may be referred by the list letter followed by an offset subscript, i.e., $\mathbf{a}[k] = a_k$. Sublists can be denoted using *ranges*: for a range $u{:}v$ such that $u < v$, $\mathbf{a}[u{:}v] = [a_u, \ldots, a_{v-1}]$. An open range covers all offsets up to either end left unspecified, i.e., $\mathbf{a}[{:}v] = [a_0, \ldots, a_{v-1}]$ and $\mathbf{a}[u{:}] = [a_u, \ldots, a_{n-1}]$.

Two lists can be joined together using the *concatenation operator* $+\!\!+$. For two lists $\mathbf{a} = [a_0, \ldots, a_m]$ and $\mathbf{b} = [b_0, \ldots, b_n]$, $\mathbf{a} +\!\!+ \mathbf{b} = [a_0, \ldots, a_m, b_0, \ldots, b_n]$. Individual elements can be appended to either list end by combining concatenation with singleton lists, e.g., $\mathbf{a} +\!\!+ [c] = [a_0, \ldots, a_m, c]$. Element deletion is not defined explicitly, but list ranges can be used to remove elements from either end, e.g., $\mathbf{a}[1{:}] = [a_1, \ldots, a_m]$, and list concatenation can be combined with ranges to effect deletion of middle-list elements, e.g., $\mathbf{a}[{:}k] +\!\!+ A[k + 1{:}] = [a_0, \ldots, a_{k-1}, a_{k+1}, \ldots, a_m]$.

A.2 Vectors

For the purposes of this thesis a vector is defined as a list of numbers. Vectors inherit all list notations and operations, with the only remark that a vector's length is called its *dimension*. When a vector is specified for the first time its dimension may be denoted as a superscript, e.g., $\mathbf{a}^n$ denotes a vector of dimension n; care must be taken however so this isn't mistaken by an exponentiation. The term may also be used to refer to a vector's offset, e.g., $a[0]$ is the "first dimension" of a. A vector where all dimensions have the same value can be specified as a bold-faced number with the dimension as superscript, e.g., if $\mathbf{z} = \mathbf{0}^n$ then $|\mathbf{z}| = n$ and $\mathbf{z}[i] = 0$ for all $0 \leq i < n$.

Vector addition, subtraction, product and exponentiation are defined element-wise, i.e., for two vectors $\mathbf{a}^n$ and $\mathbf{b}^n$:

$$\mathbf{a} + \mathbf{b} = [a_i + b_i \mid 0 \leq i < n] \tag{A.1}$$

$$\mathbf{a} - \mathbf{b} = [a_i - b_i \mid 0 \leq i < n] \tag{A.2}$$

$$\mathbf{ab} \quad = [a_i b_i \quad \mid 0 \leq i < n] \tag{A.3}$$

$$\mathbf{a}^x \quad = [a_i^x \quad \mid 0 \leq i < n] \tag{A.4}$$

Vectors can also be *shifted*. Intuitively, shifting a vector moves its values "left" or "right" by a given number of positions; values moved "over the edge" are discarded, and positions left empty are filled with zeros. Formally, the *left shift* and *right shift* operators are defined as:

$$\mathbf{a} \ll k = \mathbf{a}[k{:}] +\!\!+ \mathbf{0}^k \tag{A.5}$$

$$\mathbf{a} \gg k = \mathbf{0}^k +\!\!+ \mathbf{a}[{:}n-k] \tag{A.6}$$

A.3 Matrices

A matrix is defined here as an extension of the vector concept into two dimensions. It is a natural way to represent bidimensional structure in data sets, such as in images. DICH makes extensive use of matrices to represent visual inputs, store and process information.

A matrix is denoted by a bold-faced capital letter, e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, etc. When a matrix is specified for the first time, a superscript may be used to indicate its row and column dimensions: $\mathbf{A}^{5\times6}$ indicates that matrix $\mathbf{A}$ has 5 rows and 6 columns. Individual matrix cells are denoted by following the matrix letter with row and column offsets enclosed in the indexing operator $[\cdot]$, so e.g., $\mathbf{A}[i, j]$ denotes the cell at the i^{th} row and j^{th} column of $\mathbf{A}$. Offsets start at zero and count top-to-bottom and left-to-right, so the top-left cell of $\mathbf{A}$ is $\mathbf{A}[0, 0]$ while e.g., $\mathbf{A}[2, 1]$ refers to the cell at the third row and second column of $\mathbf{A}$.

Individual rows are denoted by substituting ":" for the column offset at the index operator, so e.g. $\mathbf{A}[2, :]$ refers to the 3^{rd} row of $\mathbf{A}$. Individual columns are denoted similarly, e.g. $\mathbf{A}[:, 1]$ denotes the 2^{nd} column of $\mathbf{A}$. Subsections of a matrix can be denoted by ranges at the matrix operator, so e.g., $\mathbf{A}[0 : 2, 3 : 6]$

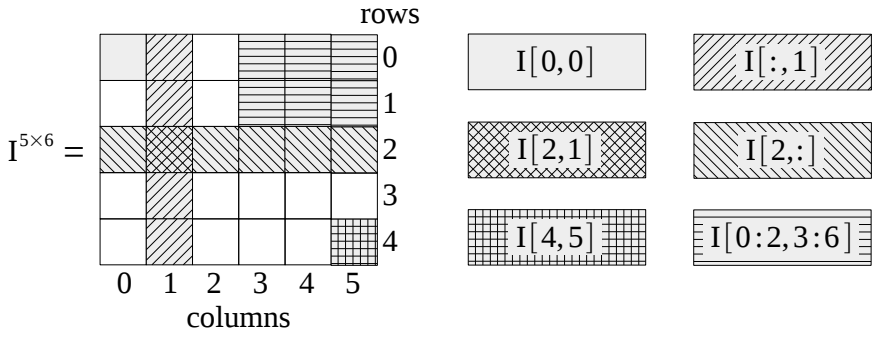


Figure A.1: Basic matrix notation. A matrix is denoted by a capital letter with row and column sizes indicated as a superscript. Cells can be referenced individually or grouped as rows, columns or bidimensional ranges.

denotes a range covering rows 0 and 1 and columns 3 to 5 of $\mathbf{A}$. It should be noticed that a range $a : b$ denotes offsets from a to $b-1$, so that any two ranges $a : b$ and $b : c$ are disjoint. Figure A.1 illustrates these notations with examples.

Occasionally it may be cumbersome referring to a matrix cell using the index notation $\mathbf{A}[i, j]$. In such cases a cell may be denoted by the matrix letter in lowercase, followed by row and column offsets in subscript. For example, $\mathbf{A}[i, j]$ may be denoted as a_{ij}, or $a_{i,j}$ if a clear separation between offsets is warranted.

Most matrices are numeric constructs, but some matrices are defined *analytically* in terms of a generative function. For a matrix $\mathbf{A}$ where every cell value is defined as $\mathbf{A}[i, j] = f(i, j)$ for $0 \leq i < m$ and $0 \leq j < n$, $\mathbf{A}$ may be defined in analytic notation as $\mathbf{A}^{m\times n} = [a_{ij} = f(i, j)]$.

Addition and subtraction are defined for matrices element-wise, i.e.:

$$\mathbf{A}^{m\times n} + \mathbf{B}^{m\times n} = \mathbf{C}^{m\times n} = [c_{ij} = a_{ij} + b_{ij}] \tag{A.7}$$

$$\mathbf{A}^{m\times n} - \mathbf{B}^{m\times n} = \mathbf{C}^{m\times n} = [c_{ij} = a_{ij} - b_{ij}] \tag{A.8}$$

Matrix multiplication is defined as an extension of the euclidean inner product for two dimensions:

$$\mathbf{A}^{m\times p}\,\mathbf{B}^{p\times n} = \mathbf{C}^{m\times n} = \left[c_{ij} = \sum_{k=0}^{p-1} a_{i,k} b_{k,j}\right] \tag{A.9}$$

Moreover, *element-wise* matrix multiplication, also known as the Hadamard product, is defined as:

$$\mathbf{A}^{m\times n} \circ \mathbf{B}^{m\times n} = \mathbf{C}^{m\times n} = [c_{ij} = a_{ij} b_{ij}] \tag{A.10}$$

Element-wise matrix exponentiation, also called the Hadamard power, is defined for any $\mathbf{A}^{m\times n}$ as:

$$\mathbf{A}^{\circ x} = \mathbf{C}^{m\times n} = \left[c_{ij} = a_{ij}^{x}\right] \tag{A.11}$$

The combination of Hadamard power and product can be used in lieu of element-wise division, i.e., for any $\mathbf{A}^{m\times n}$ and $\mathbf{B}^{m\times n}$:

$$\mathbf{A} \circ \mathbf{B}^{\circ -1} = \mathbf{C}^{m\times n} = \left[c_{ij} = \frac{a_{ij}}{b_{ij}} \right] \tag{A.12}$$

The concept of element-wise operations can be generalized to arbitrary functions. For a function f and matrix $\mathbf{A}^{m\times n}$, the element-wise application of f to $\mathbf{A}$ is denoted as:

$$f^{\circ}(\mathbf{A}) = \mathbf{C}^{m\times n} = \Big[c_{ij} = f(a_{ij}) \Big] \tag{A.13}$$

There are also operations involving a matrix and a scalar. For a matrix $\mathbf{A}^{m\times n}$ and scalar b, the following operations are defined:

$$\mathbf{A} + b = \mathbf{C}^{m\times n} = \Big[c_{ij} = a_{ij} + b \Big] \tag{A.14}$$

$$\mathbf{A} - b = \mathbf{C}^{m\times n} = \Big[c_{ij} = a_{ij} - b \Big] \tag{A.15}$$

$$\mathbf{A}b \ = \mathbf{C}^{m\times n} = \Big[\ c_{ij} = a_{ij} b \ \Big] \tag{A.16}$$

$$\frac{\mathbf{A}}{b} \ = \mathbf{C}^{m\times n} = \Big[\ c_{ij} = \frac{a_{ij}}{b} \ \Big] \tag{A.17}$$

Operations behave as expected in relation to commutativity and other arithmetic properties, i.e., $\mathbf{A} + b = b + \mathbf{A}$, $b - \mathbf{A} = -(\mathbf{A} - b)$, etc.

Matrix operations also include some summary operations. The sum of all elements of a matrix $\mathbf{A}^{m\times n}$ is defined as:

$$\sum \mathbf{A} = \sum_{i,j}^{m,n} \mathbf{A}[i,j] \tag{A.18}$$

The arithmetic mean (or simply the mean) is defined as:

$$\overline{\mathbf{A}} = \frac{\sum \mathbf{A}}{mn} \tag{A.19}$$

And the magnitude, as an extension of the euclidean norm:

$$\|\mathbf{A}^{m\times n}\| = \sqrt{\sum \mathbf{A}^{\circ 2}} \tag{A.20}$$

Appendix B

Images

B.1 Multi-channel images

While it was previously said that DICH uses matrices to represent images, some additional considerations are required in the case of *multi-channel* images, such as color images. The main issue is that while matrices are bidimensional constructs, multi-channel images span three dimensions: not only picture elements (or *pixels*) are disposed in a matrix-like grid, each is composed of multiple values, corresponding to image channels. For example, in an RGB image, each pixel contains three values, corresponding to the Red, Green and Blue channels.

Matrix operations can be extended to images if pixels are represented as vectors. For example, let $\mathbf{I}$ be an image such that:

$$\mathbf{I}^{m\times n} = \Big[\mathbf{i}_{ij} = [r_{ij}, g_{ij}, b_{ij}]\Big] \tag{B.1}$$

Where r_{ij}, g_{ij} and b_{ij} correspond to the Red, Green and Blue channel values of pixel $\mathbf{i}_{ij}$. Then by the definitions of vector arithmetic operations (Eq. A.1, Eq. A.2, Eq. A.3 and Eq. A.4) it's clear that matrix addition (Eq. A.7), subtraction (Eq. A.8), multiplication (Eq. A.9), element-wise product (Eq. A.10) and exponentiation (Eq. A.11) can all be seamlessly extended to images.

Matrix operations work on multi-channel images *channel-wise* – channels are processed independently, as if they were stored in separate matrices. This separation can be made explicit through the use of the channel function, defined as:

$$\mathrm{C}_k(\mathbf{I}^{m\times n}) = \mathbf{C}^{m\times n} = \Big[c_{ij} = i_k \mid \mathbf{i} = \mathbf{I}[i,j]\Big] \tag{B.2}$$

The channel function is useful for defining operations that work on an image's own channels. For example, the luminance $\mathbf{L}(\mathbf{I})$ of image $\mathbf{I}$ is defined (according to the CCIR 601 specification [59]) as:

$$\mathbf{L}(\mathbf{I}) = 0.299\ \mathrm{R}(\mathbf{I}) + 0.587\ \mathrm{G}(\mathbf{I}) + 0.114\ \mathrm{B}(\mathbf{I}) \tag{B.3}$$

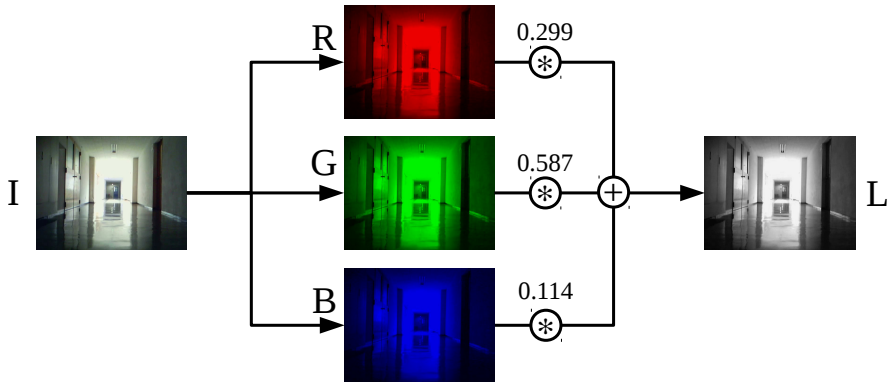


Figure B.1: Luminance image computation. The Red (R), Green (G) and Blue (B) channels of RGB image $\mathbf{I}$ are extracted and summed after being multiplied by appropriate weights. The result is luminance image $\mathbf{L}$.

For R($\mathbf{I}$), G($\mathbf{I}$) and B($\mathbf{I}$) respectively the Red, Green and Blue channels of $\mathbf{I}$ defined as:

$$\mathrm{R}(\mathbf{I}) = \mathrm{C}_0(\mathbf{I}) \tag{B.4}$$
$$\mathrm{G}(\mathbf{I}) = \mathrm{C}_1(\mathbf{I}) \tag{B.5}$$
$$\mathrm{B}(\mathbf{I}) = \mathrm{C}_2(\mathbf{I}) \tag{B.6}$$

Figure B.1 illustrates the operation.

B.2 Integral images

Also known as "summed area tables", *integral images* are data structures for quickly computing the sum of a rectangular range on a matrix [60]. Given a matrix $\mathbf{I}$, its integral image $\mathbf{I}_\Sigma$ is defined as:

$$\mathbf{I}_\Sigma = \left[\sigma_{ij} = \sum_{u=0}^{u \leq i} \sum_{v=0}^{v \leq j} \mathbf{I}[u, v] \right] \tag{B.7}$$

In practice, $\mathbf{I}_\Sigma$ can be quickly computed in a single pass over $\mathbf{I}$, exploring the fact that:

$$\mathbf{I}_\Sigma[i, j] = \mathbf{I}[i, j] + \mathbf{I}_\Sigma[i, j-1] + \mathbf{I}_\Sigma[i-1, j] - \mathbf{I}_\Sigma[i-1, j-1] \tag{B.8}$$

Once $\mathbf{I}_\Sigma$ has been computed, it can be used to evaluate several measurements over arbitrary rectangular matrix ranges in constant time. First, the sum $\sum \mathbf{I}[i_0 : i_n, j_0 : j_n]$ can be computed as:

$$\sum \mathbf{I}[i_0 : i_n, j_0 : j_n] = \mathbf{I}_\Sigma[i_0, j_0] - \mathbf{I}_\Sigma[i_0, j_n] - \mathbf{I}_\Sigma[i_n, j_0] + \mathbf{I}_\Sigma[i_n, j_n] \tag{B.9}$$

The magnitude $\|\mathbf{I}[i_0 : i_n, j_0 : j_n]\|$ can be computed over $\mathbf{I}^2_\Sigma$, the integral image of $\mathbf{I}^{\circ 2}$:

$$\|\mathbf{I}[i_0 : i_n, j_0 : j_n]\| = \sqrt{\mathbf{I}^2_\Sigma[i_0, j_0] - \mathbf{I}^2_\Sigma[i_0, j_n] - \mathbf{I}^2_\Sigma[i_n, j_0] + \mathbf{I}^2_\Sigma[i_n, j_n]} \tag{B.10}$$

The mean $\overline{\mathbf{I}[i_0 : i_n, j_0 : j_n]}$ derives naturally from the sum:

$$\overline{\mathbf{I}[i_0 : i_n, j_0 : j_n]} = \frac{\sum \mathbf{I}[i_0 : i_n, j_0 : j_n]}{(i_n - i_0)(j_n - j_0)} \tag{B.11}$$

The derivation of the standard deviation $std(\mathbf{I}[i_0 : i_n, j_0 : j_n])$ is a little more involved, however. First, for $\mathbf{R}^{m \times n} = \mathbf{I}[i_0 : i_n, j_0 : j_n]$, standard deviation is defined as:

$$std(\mathbf{R}) = \sqrt{\frac{\sum (\mathbf{R} - \overline{\mathbf{R}})^{\circ 2}}{mn}} \tag{B.12}$$

Applying the binomial theorem to $(\mathbf{R} - \overline{\mathbf{R}})^{\circ 2}$ gives:

$$std(\mathbf{R}) = \sqrt{\frac{\sum \left(\mathbf{R}^{\circ 2} + \overline{\mathbf{R}}^2 - 2\,\overline{\mathbf{R}}\,\mathbf{R} \right)}{mn}} \tag{B.13}$$

Given the identities of the sum operator and the definition of magnitude:

$$std(\mathbf{R}) = \sqrt{\frac{\|\mathbf{R}\|^2 + mn\overline{\mathbf{R}}^2 - 2\,\overline{\mathbf{R}} \sum \mathbf{R}}{mn}} \tag{B.14}$$

Separating the fraction terms we get:

$$std(\mathbf{R}) = \sqrt{\frac{\|\mathbf{R}\|^2}{mn} + \overline{\mathbf{R}}^2 - 2\overline{\mathbf{R}}\frac{\sum \mathbf{R}}{mn}} \tag{B.15}$$

But since $\overline{\mathbf{R}} = \frac{\sum \mathbf{R}}{mn}$ we have:

$$std(\mathbf{R}) = \sqrt{\frac{\|\mathbf{R}\|^2}{mn} + \overline{\mathbf{R}}^2 - 2\overline{\mathbf{R}}^2} = \sqrt{\frac{\|\mathbf{R}\|^2}{mn} - \overline{\mathbf{R}}^2} \tag{B.16}$$

Finally, returning $\mathbf{R} = \mathbf{I}[i_0 : i_n, j_0 : j_n]$, $m = (i_n - i_0)$ and $n = (j_n - j_0)$ gives:

$$std(\mathbf{I}[i_0 : i_n, j_0 : j_n]) = \sqrt{\frac{\|\mathbf{I}[i_0 : i_n, j_0 : j_n]\|^2}{(i_n - i_0)(j_n - j_0)} - \overline{\mathbf{I}[i_0 : i_n, j_0 : j_n]}^2} \tag{B.17}$$

Where $\|\mathbf{I}[i_0 : i_n, j_0 : j_n]\|$ and $\overline{\mathbf{I}[i_0 : i_n, j_0 : j_n]}$ can be computed from integral images as shown in Eq. B.10 and Eq. B.11.

B.3 Image correlation

Measuring the similarity between two images is a common task in computer vision applications. A typical use case is *template matching*, where a template $\mathbf{T}^{m_T \times n_T}$ is searched for a best match across an image $\mathbf{I}^{m_I \times n_I}$, such that $m_T < m_I$ and $n_T < n_I$. Template matching is usually performed by using *normalized cross-correlation* [61] to construct a map of similarities between $\mathbf{T}$ and $\mathbf{I}$ at every possible offset:

$$\mathcal{N}_{cc}(\mathbf{T}, \mathbf{I}) = \left[c_{ij} = \frac{\sum \left[(\mathbf{T} - \overline{\mathbf{T}}) \circ (\mathbf{I}_{ij} - \overline{\mathbf{I}_{ij}}) \right]}{std(\mathbf{T}) std(\mathbf{I}_{ij})} \right] \tag{B.18}$$

Where $\mathbf{I}_{ij} = \mathbf{I}[i : i + m_T, j : j + n_T]$. The coordinates of the best match between template and image are the coordinates of the highest value at the similarity map:

$$(i^*, j^*) = \arg\max \mathcal{N}_{cc}(\mathbf{T}, \mathbf{I}) \tag{B.19}$$

As it is defined, Eq. B.18 can be expensive to evaluate, even if integral images are used to speed up the computation of $\overline{\mathbf{I}_{ij}}$ and $std(\mathbf{I}_{ij})$. However, its expression can be simplified if $\mathbf{T}$ is replaced with a normalized template $\mathcal{N}(\mathbf{T})$ such that $\overline{\mathcal{N}(\mathbf{T})} = 0$ and $std(\mathcal{N}(\mathbf{T})) = 1$:

$$\mathcal{N}(\mathbf{T}) = \frac{\mathbf{T} - \overline{\mathbf{T}}}{\|\mathbf{T} - \overline{\mathbf{T}}\|} \tag{B.20}$$

Substituting $\mathcal{N}(\mathbf{T})$ for $\mathbf{T}$ in Eq. B.18 and applying appropriate summation identities gives:

$$\mathcal{N}_{cc}(\mathbf{T}, \mathbf{I}) = \left[c_{ij} = \frac{\sum [\mathcal{N}(\mathbf{T}) \circ \mathbf{I}_{ij}] - \overline{\mathbf{I}_{ij}} \sum \mathcal{N}(\mathbf{T})}{std(\mathbf{I}_{ij})} \right] \tag{B.21}$$

Which, given that $\sum \mathcal{N}(\mathbf{T}) = 0$, further reduces to:

$$\mathcal{N}_{cc}(\mathbf{T}, \mathbf{I}) = \left[c_{ij} = \frac{\sum [\mathcal{N}(\mathbf{T}) \circ \mathbf{I}_{ij}]}{std(\mathbf{I}_{ij})} \right] \tag{B.22}$$

Looking at the formula above, it's clear that the numerator terms of all c_{ij} could be computed as the cross-correlation $\mathcal{N}(\mathbf{T}) \star \mathbf{I}$, whereas the denominator terms could be given by:

$$\mathrm{D}(\mathbf{I}, m_T \times n_T) = \left[d_{ij} = std(\mathbf{I}[i : i + m_T, j : j + n_T]) \right] \tag{B.23}$$

Leading to the following definition:

$$\mathcal{N}_{cc}(\mathbf{T}, \mathbf{I}) = (\mathcal{N}(\mathbf{T}) \star \mathbf{I}) \circ \mathrm{D}(\mathbf{I}, m_T \times n_T)^{\circ -1} \tag{B.24}$$

This alternate definition enables some useful optimizations – in particular, the convolution theorem states that:

$$\mathcal{N}(\mathbf{T}) \star \mathbf{I} = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{N}(\mathbf{T}))^* \circ \mathcal{F}(\mathbf{I})) \tag{B.25}$$

Where $\mathcal{F}(\cdot)$ is the Fourier transform operator and * denotes the complex conjugate [62]. The use of fast Fourier transform algorithms and (where a single visual input is to be cross-correlated several times, e.g. in a visual search across a sequence of memory records) the caching of Fourier transforms can significantly speed up computations.

Bibliography

[1] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[3] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1023–1029. Ieee, 2000.

[4] Patrick Peruch, Jean Pailhous, and Christian Deutsch. How do we locate ourselves on a map: A method for analyzing self-location processes. *Acta Psychologica*, 61(1):71–88, 1986.

[5] Edvard I Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.*, 31:69–89, 2008.

[6] Robert J Sternberg and Karin Sternberg. *Cognitive psychology*. Nelson Education, 2015.

[7] Mark Wagner. *The geometries of visual space*. Psychology Press, 2006.

[8] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[9] Robert U Muller and John L Kubie. The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *J Neurosci*, 7(7):1951–1968, 1987.

[10] Perry W Thorndyke. Distance estimation from cognitive maps. *Cognitive psychology*, 13(4):526–550, 1981.

[11] Perry W Thorndyke and Barbara Hayes-Roth. Differences in spatial knowledge acquired from maps and navigation. *Cognitive psychology*, 14(4):560–589, 1982.

[12] Ian Moar and Gordon H Bower. Inconsistency in spatial knowledge. *Memory & Cognition*, 11(2):107–113, 1983.

[13] Alastair D Smith and Gillian Cohen. Memory for places: Routes, maps, and object locations. *Memory in the real world*, pages 173–206, 2008.

[14] Tatjana Seizova-Cajic. The role of perceived relative position in pointing to objects apparently shifted by depth-contrast. *Spatial vision*, 16(3):325–346, 2003.

[15] Helio Perroni Filho and Alberto Ferreira de Souza. On multichannel neurons, with an application to template search. *Journal of Network and Innovative Computing*, 2(1):10–21, 2014.

[16] H Perroni Filho and A Ohya. Mobile robot path drift estimation using visual streams. In *System Integration (SII), 2014 IEEE/SICE International Symposium on*, pages 192–197. IEEE, 2014.

[17] Helio Perroni Filho and Akihisa Ohya. Image correspondence based on interest point correlation in difference streams: Method and applications to mobile robot localization. *Journal of Robotics and Mechatronics*, 28(2):234–241, 2016.

[18] Margaret Wilson. Six views of embodied cognition. *Psychonomic bulletin & review*, 9(4):625–636, 2002.

[19] K. Pribram. Holonomic brain theory. *Scholarpedia*, 2(5):2735, 2007. revision #91358.

[20] K. Rayner and M. Castelhano. Eye movements. *Scholarpedia*, 2(10):3649, 2007. revision #126973.

[21] AV Samsonovich, ML Anderson, KV Anokhin, GA Ascoli, G Biswas, GA Carpenter, B Chandrasekaran, A Chella, KA De Jong, S Franklin, et al. Biologically inspired cognitive architectures. 2009.

[22] Olivier L Georgeon, James B Marshall, and Riccardo Manzotti. ECA: An enactivist cognitive architecture based on sensorimotor modeling. *Biologically Inspired Cognitive Architectures*, 6:46–57, 2013.

[23] Tamas Madl, Stan Franklin, Ke Chen, and Robert Trappl. Spatial working memory in the lida cognitive architecture. In *Proceedings of the 12th international conference on cognitive modelling*, pages 384–390, 2013.

[24] Bernard J Baars. The conscious access hypothesis: origins and recent evidence. *Trends in cognitive sciences*, 6(1):47–52, 2002.

[25] Samuel Wintermute. Imagery in cognitive architecture: Representation and control at multiple levels of abstraction. *Cognitive Systems Research*, 19:1–29, 2012.

[26] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, nov 2008.

[27] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: A survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, pages 363–371. IOS Press, 2008.

[28] Mark Cummins and Paul Newman. Highly scalable appearance-only slam-fab-map 2.0. In *Robotics: Science and Systems*, volume 1, pages 12–18. Seattle, USA, 2009.

[29] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649. IEEE, 2012.

[30] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

[31] Lee E Clement, Jonathan Kelly, and Timothy D Barfoot. Monocular visual teach and repeat aided by local ground planarity. *Proc. Field and Service Robotics (FSR)*, 2015.

[32] Matthew Gadd and Paul Newman. A framework for infrastructure-free warehouse navigation. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3271–3278. IEEE, 2015.

[33] Matías Nitsche, Taihú Pire, Tomáš Krajník, Miroslav Kulich, and Marta Mejail. Monte carlo localization for teach-and-repeat feature-based navigation. In *Advances in Autonomous Robotics Systems*, pages 13–24. Springer, 2014.

[34] Pablo De Cristóforis, Matias Nitsche, Tomáš Krajník, Taihú Pire, and Marta Mejail. Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments. *Pattern Recognition Letters*, 53:118–128, 2015.

[35] Martin Dörfler and Libor Přeučil. Employing observation angles in pose recognition; application for teach-and-repeat robot navigation. In *Modelling and Simulation for Autonomous Systems*, pages 165–172. Springer, 2015.

[36] Ming Liu, Cedric Pradalier, and Roland Siegwart. Visual homing from scale with an uncalibrated omnidirectional camera. *Robotics, IEEE Transactions on*, 29(6):1353–1365, 2013.

[37] Ralf Möller, Michael Horst, and David Fleer. Illumination tolerance for visual navigation with the holistic min-warping method. *Robotics*, 3(1):22–67, 2014.

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

[39] Michael Paton, François Pomerleau, and Timothy D Barfoot. Eyes in the back of your head: Robust visual teach & repeat using multiple stereo cameras. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 46–53. IEEE, 2015.

[40] Michael Paton, Kirk MacTavish, Chris J Ostafew, and Timothy D Barfoot. It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1519–1526. IEEE, 2015.

[41] D. Burschka and G. Hager. Vision-based control of mobile robots. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1707–1713, 2001.

[42] Shree K Nayar. Catadioptric omnidirectional camera. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 482–488. IEEE, 1997.

[43] Andrea Vedaldi. *Invariant representations and learning for computer vision*. ProQuest, 2008.

[44] Tobi Delbruck. Frame-free dynamic digital vision. In *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pages 21–26, 2008.

[45] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 db $15\mu s$ latency asynchronous temporal contrast vision sensor. *Solid-State Circuits, IEEE Journal of*, 43(2):566–576, 2008.

[46] Claudio M Privitera and Lawrence W Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(9):970–982, 2000.

[47] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[48] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981.

[49] Thinkpad x201, x201s. Accessed: 2016-06-20.

[50] Intelligent robot laboratory – introduction. Accessed: 2016-06-20.

[51] T-frog project (japanese). Accessed: 2016-06-20.

[52] Scanning range finder (sokuiki sensor) – urg-04lx. Accessed: 2016-06-20.

[53] Hd pro webcam c920. Accessed: 2016-06-20.

[54] roscpp Overview: Callbacks and Spinning. Accessed: 2015-11-13.

[55] Kubuntu. Accessed: 2016-06-20.

[56] Opencv. Accessed: 2016-06-20.

[57] Intel threading building blocks. Accessed: 2015-11-13.

[58] Richard S. Bird. An introduction to the theory of lists. In *Proceedings of the NATO Advanced Study Institute on Logic of Programming and Calculi of Discrete Design*, pages 5–42, 1987.

[59] Charles Poynton. *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[60] Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.

[61] John P Lewis. Fast template matching. In *Vision interface*, volume 95, pages 15–19, 1995.

[62] Bernd Girod, Marcus Andreas Magnor, and Hans-Peter Seidel. *Vision, Modeling, and Visualization 2004: Proceedings, November 16-18, 2004, Standford, USA*. IOS Press, 2004.

List of Published Papers

1. Helio Perroni Filho and Akihisa Ohya, "Mobile Robot Path Drift Estimation using Visual Streams", IEEE/SICE International Symposium on System Integration (SII), Tokyo, Japan, December 13-15, 2014, Proceedings, pp. 192-197. DOI: 10.1109/SII.2014.7028036

2. Helio Perroni Filho and Akihisa Ohya, "Image Correspondence Based on Interest Point Correlation in Difference Streams: Method and Applications to Mobile Robot Localization", Journal of Robotics & Mechatronics, Volume 28, No. 2, 2016, pp. 234-241. DOI: 10.20965/jrm.2016.p0234

3. Helio Perroni Filho and Akihisa Ohya, "A Biologically Inspired Architecture for Visual Self-location", Biologically Inspired Cognitive Architectures (BICA) for Young Scientists, 2016, pp. 297-303.
DOI: 10.1007/978-3-319-32554-5_38