Visual Homing in Dynamic Indoor Environments

Matthew D. Szenher

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2008

Abstract

Our dissertation concerns robotic navigation in dynamic indoor environments using image-based visual homing. Image-based visual homing infers the direction to a goal location S from the navigator's current location C using the similarity between panoramic images I_S and I_C captured at those locations. There are several ways to compute this similarity. One of the contributions of our dissertation is to identify a robust image similarity measure – mutual image information – to use in dynamic indoor environments. We crafted novel methods to speed the computation of mutual image information with both parallel and serial processors and demonstrated that these time-savers had little negative effect on homing success. Image-based visual homing requires a homing agent to move so as to optimise the mutual image information signal. As the mutual information signal is corrupted by sensor noise we turned to the stochastic optimisation literature for appropriate optimisation algorithms. We tested a number of these algorithms in both simulated and real dynamic laboratory environments and found that gradient descent (with gradients computed by one-sided differences) works best.

Acknowledgements

I would like to thank my wife Lucy Raimes for her much-needed love and support. Thanks also to my adviser Barbara Webb who offered her advice and wisdom during our innumerable Friday afternoon talks. Finally, I would like to acknowledge the colleagues who have offered encouragement, advice, and a few pints along the way: Tim Lukins, Richard Reeve, Hugo Rosano, Darren Smith, and Jan Wessnitzer.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Matthew D. Szenher*)

Table of Contents

Chapter 1

Introduction

1.1 Navigation by Robots and Insects

Robot navigation – in particular visual homing – is the primary concern of this dissertation. We define navigation as do Franz and Mallot [2000]: "Navigation is the process of determining and maintaining a course or trajectory to a goal location."

Many robotic navigation algorithms found in the literature explicitly answer the questions *Where is the robot in some suitable coordinate system?* and *Where is the goal in the same coordinate system?*. The localisation problem has been described by some researchers as "the most fundamental problem to providing a mobile robot with autonomous capabilities" (Thrun et al. [2001]). Explicit metric localisation requires a map of landmark locations and descriptions of those landmarks. While navigating, the robot has to solve the so-called data association problem: determine which landmarks it is currently sensing given its current sensor data and the landmark descriptions provided (see e.g. Leonard et al. [2001]). The robot then uses the egocentric bearing and/or range of the identified landmarks to determine its pose (position and orientation) in map coordinates. As we discuss in greater detail in Chapter 2, localisation is often probabalistic in nature. In early work, maps were provided by human operators. In the last few years, workers in robotic navigation have developed impressive SLAM (Simultaneous Localisation and Mapping) procedures; these allow the robot to learn the map of its current environment while localising itself in that environment. We review SLAM in Chapter 2 as well.

A second class of robotic navigation systems are topological in nature. Topological maps are representations of important and/or distinctive places in an environment, physical locations where a robot needs to make a key decision. Hallway corners and doorways are frequently represented in topological maps for indoor robot navigation. The connectedness of these important places also encodes a topological map (e.g. it is possible to navigation from one hallway corner to the next without passing through any other so-called important place). As we describe extensively in Chapter 2, topological navigation gives a qualitative answer to the question: *Where is the robot?* For example, a topological navigator might infer that it is in a particular office or moving along a particular corridor towards the billiard room. The London Tube map is a popular example of a topological map. Each Tube stop is an important place where a decision can be made. The connections between Tube stops (though not metric information like absolute distance between Tube stops) are encoded in the map.

Turning to navigation in the animal world, ethological evidence suggests that central-place foraging insects like ants and honeybees – formidable navigators – are able to navigate in natural, cluttered and dynamic environments without the need of metric maps. The desert ant *Cataglyphis fortis* for example is capable of travelling hundreds of metres from its nest in search of food and returning directly to the nest once the food has been found. Wehner [1999] reports that desert ants rely primarily on path integration (also known as dead reckoning) for long distance navigation. We can imagine an ant's foraging path as a series of path vectors, attached tip-to-tail. The negative of the sum of these vectors is a vector pointing to the ant's starting point (e.g. nest or source of food). After a tortuous food-finding excursion, ants can and do use this path integration vector to return directly to their nests (Wehner et al. [1996]). The length and orientation of each path vector in the series is noisy and this noise leads to cumulative error in the goal vector (Wehner [1999]); that is, the goal vector leads an ant to a point close to the goal but not usually coincidental with it. Behavioural evidence suggests that ants use a visual homing algorithm (discussed below) to find their nest from this nearby location (Cartwright and Collett [1983]). Judd and Collett [1998] suggest that ants create and use a vision-based topological map of positions near their nest in order to facilitate return to the nest.

We note for the sake of completeness that there has been much debate in recent years as to whether foraging insects learn a metric map representation of their environment in order to navigate (see e.g. Wehner [1999] and Bennett [1996]). Very recent evidence (Menzel et al. [2005]) supports the hypothesis that honeybees store some sort of landmark location information relative to their nest. Wehner [1999], though, argues convincingly that desert ants do not *require* a metric map of their environment in order to navigate successfully.

1.2 Visual Homing

Robotic navigation by visual homing (a type of visual servoing) is the focus of our work in this dissertation. Visual homing algorithms require no explicit quantitative localisation and thus no metric map. An agent employing a visual homing algorithm captures an image I_S (typically called the snapshot) at the goal location $S = (x_S, y_S)$ and, when later attempting to return to this location from a nearby position $C = (x_C, y_C)$, compares the current image I_C with the snapshot and infers the direction and/or distance to the goal location from the disparity between the two. We restrict ourselves to homing in two dimensions in this work though Zeil et al. [2003] demonstrate that visual homing in three dimensions is possible. We assume that no visually obvious cue marks the goal position. Otherwise the navigating agent could simply employ a beacon-aiming strategy to find the goal (Franz and Mallot [2000]).

Visual homing is a very useful navigational skill for a robot to have in its repertoire. As we indicated above, visual homing can be used in conjunction with dead reckoning to allow a robot to explore an area from a given home position and later return to that position for, for example, refueling. As we discuss in Chapter 2, visual homing has been used to solve the docking problem which requires precise positioning with respect to an object in the environment (e.g. a recharging station). We also discuss in Chapter 2 the uses of visual homing in topological navigation. Homing is often used in conjunction with a vision-based topological navigation system to move between adjacent locations in a topological map.

1.3 Image-based Visual Homing Algorithms and their Limitations

As we shall discuss in detail in Chapter 2, homing algorithms differ in the way in which image disparity is calculated. Feature-based methods segment snapshot and current images into landmarks and background. They then attempt to pair each landmark in the snapshot image with a landmark in the current image (i.e. solve the correspondence problem). Disparity is computed from the difference in bearing and/or apparent size between paired landmarks. A second class of visual homing algorithms bypass the correspondence problem, using disparities between whole images to compute homing vectors. These are known as image- or appearance-based algorithms.

Feature-based procedures require a solution to the correspondence problem as we have said. Historically, solving the correspondence problem in cluttered and dynamic environments has proved a difficult task. As we review in Chapter 2, image features computed with the recently introduced scale-invariant feature transform (SIFT) lead to reliable correspondence in such environments. This correspondence, though, often requires the comparison of hundreds or even thousands of features per image pair. In this work, we prefer to investigate a more parsimonious approach to visual homing.

The central algorithm to be explored in this dissertation relies on the empirical phenomenon, reported independently in Zeil et al. [2003] and Mitchell and Labrosse [2004], that the *difference* between two panoramic intensity images increases monotonically with the physical distance between their capture positions. Zeil et al. computed image difference with the following pixel-by-pixel root-mean-square function:

$$RMS(I_S, I_C) = \frac{1}{NM}\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}(I_C(i,j) - I_S(i,j))^2} \tag{1.1}$$

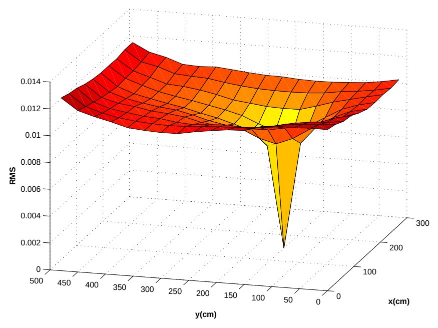
where I_S and I_C are panoramic gray-scale images with N rows and M columns. I_C is rotated to match the orientation of I_S using an external compass signal.

Equation 1.1 defines a mathematical surface, which we call a difference surface. A typical difference surface is depicted in Figure 1.1(a)). The surface's global minimum coincides with the location at which the snapshot image was taken. The surface increases in value monotonically with increasing distance from this location in all directions. Zeil's homing algorithm "Run-Down" simply moves the agent so as to minimise the RMS signal, stopping when the signal drops below a predetermined threshold. "Run-Down" is in essence an optimisation algorithm.

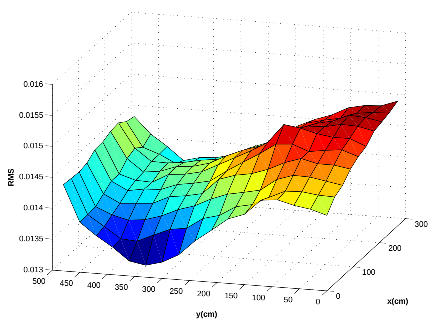
The RMS signal suffers from a serious drawback: when illumination conditions change between captures of snapshot and current images, the global minimum of the difference surface fails to coincide with the goal location (see Figure 1.1(b)).

1.4 Contributions of this Thesis

The main aim of this dissertation is to create an image-based visual homing algorithm which works robustly and efficiently in dynamic visual indoor environments. We shall investigate environments in which lighting or landmark locations change between capture of snapshot and current images. Zeil et al. [2003] measure image disparity using Equation 1.1. After a principled analysis of Equation 1.1 and its drawbacks in Chapter 3, we propose instead instead to use mutual information (MI) to gauge the similarity



(a)



(b)

Figure 1.1: Two difference surfaces formed using the RMS image difference metric. In each case, the snapshot was captured at x=150cm, y=150cm in a laboratory environment. **(a)** The snapshot was captured in the same landmark and illumination conditions as all other images. Notice the global minimum at the goal location and the absence of local minima. **(b)** Here again we use the same snapshot image as in **(a)** but the lighting source has changed in all other images. The global minimum no longer appears at the goal location. When different goal locations were used, we observed qualitatively similar disturbances in the difference surfaces formed.

between images I_S and I_C. Mutual information can be calculated with the following formula (adapted from Hill et al. [2001]):

$$MI(I_S, I_C) = \sum_i \sum_j p(i,j) log \frac{p(i,j)}{p_S(i) p_C(j)} \tag{1.2}$$

Here, $p_S(i)$ is the probability that a pixel will have intensity i (where i is typically in the range $[0,255]$) in image I_S. These probabilities are estimated from the intensity histogram of I_S; $p_C(i)$ is defined similarly for I_C. The joint probability $p(i,j)$ is the probability that the same pixel will have intensity i in I_S and intensity j in I_C. Note that in Equation 1.1 i and j range over pixel locations whereas in Equation 1.2 i and j range over intensity values. As we shall explain more fully in Chapter 3, mutual information determines how well the current image I_C predicts the snapshot image I_S. We demonstrate empirically in Chapter 3 that the use of MI leads to difference surface homing which is in many cases more robust to visual dynamism than the use of RMS.

In Chapter 4 we ask an age-old question in computer science: How do we make the computation (of image similarity with mutual information) *faster*? We investigate both parallel and serial computation of image similarity and enquire into the effects of increased computation speed on homing success.

Zeil et al. [2003] use a simple optimisation algorithm to move a homing robot so as to optimise the difference surface. We explore in Chapter 5 several other optimisation algorithms and – using novel, sensible criteria – determine which of these is best for the purpose of difference surface homing. Unlike other visual homing researchers – Zeil et al. [2003] included – we explore in Chapter 5 the effects of sensor noise on visual homing success. Robotic visual homing agents require a camera to capture a visual image. Most agents also use a compass (e.g. magnetic) to align snapshot and current images in the same external reference direction. Both of these sensors return noisy readings. In Chapter 5 we first characterise the noise probability density functions for both of these sensors. We then look at the effects of this noise on the image similarity signal. The characteristics of the noise in the image disparity signal help guide us in our choice of a good method to optimise the difference surface. We also use this noise information to create realistic homing simulations. In Chapter 6 we describe our robotic homing system and our live experiments carried out in an office environment to validate some of the results reported in previous chapters.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we shall review the various homing algorithms found in the robotics and insect ethology literature. We shall also put visual homing in the broader context of robotic navigation algorithms, reviewing general trends in this large body of research.

This chapter is organised as follows. In Section 2.2 we review current trends in robotic navigation. This includes a discussion of navigation with both metric and topological maps and algorithms robots can employ to autonomously generate these maps. We then move on to discuss visual homing in particular, beginning with a brief description of the methods used by homing researchers to capture panoramic images in Section 2.3.1. Feature-based methods are covered in Section 2.3.2. We present a critical discussion of popular solutions to the correspondence problem in Section 2.3.3. Section 2.3.4 covers image-based (also known as appearance-based) homing algorithms. We discuss in Section 2.3.5 a description of a recently discovered "visual compass" which could be useful in future visual navigation work. Finally, we discuss problems in computer vision and robotics which are closely related to visual homing in Section 2.4. Conclusions follow in Section 2.6.

We note that several of the papers we included in this work were published after we completed the bulk of our research. This is true of papers published in or after late 2006. This recent work thus had no bearing on our research goals. We included these papers here because they provide useful information about visual homing or related problems. Sometimes, as with Pons et al. [2007], we review recently published results so we can compare them with our own later in this dissertation.

2.2 Current Trends in Robotic Navigation

2.2.1 Metric Localisation

Metric localisation is the process of computing pose (position and orientation expressed in a suitable reference frame). Almost all robot localisation algorithms work by trying to match incoming sensor data with the information contained in a map. All sensors return noisy signals and this sensor noise inevitably leads to uncertainty in the robot's inferred pose. Thus, as Thrun et al. [2005] point out, it is appropriate to represent a robot's belief about its pose with a probability distribution, a function which associates a probability with every possible pose that the robot may be in at a given time.

In general, three distinct localisation problems appear in the literature (Thrun et al. [2005]): position tracking, global localisation, and the kidnapped robot problem. In position tracking problems, the initial pose of the robot in some suitable global coordinate frame is known with relatively high accuracy and precision. Position tracking algorithms are designed to update this initial pose as the robot moves through the environment. In global localisation, the robot's initial pose is unknown. The robot must infer its pose solely by matching sensor readings with map information. This is considered a harder problem than pose tracking as the robot must, at least initially, consider all of the map data when making pose inferences. The kidnapped robot problem is a difficult mix of pose tracking and global localisation. The robot *thinks* it has a good initial pose estimate but in fact its information is incorrect; it is rather in a different part of the mapped environment altogether. This state of affairs could occur because the robot was moved from a known pose without sensing the move (i.e. it has been "blindfolded" and kidnapped).

Many localisation algorithms exist in the literature. We shall review some popular and influential algorithms below. The algorithm appropriate for use in a certain situation depends on the type of map provided to the robot (if any), the type of sensor(s) the robot has to sample the environment, and the type of localisation problem the robot's user wishes to solve. We have already discussed the various localisation problems which occur in the literature. We shall cover different examples of maps and sensors in the course of our discussion of individual localisation procedures.

Perhaps the simplest pose tracking algorithm requires no map whatsoever. Mobile robots can integrate the translation and rotation commands they issue while moving to keep track of their pose over time. Since motor commands are not always carried out as desired (due to e.g. collisions with objects), most wheeled mobile robots are equipped with wheel encoders which count the number of rotations (or fractions of a rotation) undergone by each wheel while the robot is moving. They can integrate this odometric information to produce a pose estimate. Converting wheel rotations into pose change requires knowledge of the radii of the wheels (which are probably slightly different from one another) and the length of the axle(s) connecting the wheels. The more imprecise the knowledge of these dimensions, the more error introduced in the estimation of pose change from odometric sensors. Borenstein and Feng [1996] devised a method called the University of Michigan Benchmark (UMBmark) to determine these dimensions with high accuracy. UMBmark is to be applied only occasionally as Borenstein and Feng [1996] note that wheel radius and axle length probably do not change much over time. Borenstein and Feng [1996] demonstrate that the use of UMBmark can lead to an increase in pose estimation accuracy of one order of magnitude.

Imprecision in the knowledge of wheel radii and axle length will remain even if a procedure like UMBmark is employed. This imprecise knowledge leads to systematic error in pose change estimates. Integration of erroneous pose change estimates causes the robot's pose estimate to become more and more uncertain over time (Thrun et al. [2005]). The error is in fact unbounded. Also, robot wheels are subject to non-systematic errors like wheel slippage due to, for example, smooth floors and/or high-magnitude acceleration (Borenstein and Feng [1996]). These non-systematic errors will contribute to erroneous pose tracking as well. For these reasons, most pose tracking schemes use sensor data (usually) in conjunction with map information and odometric measurements in order to localise.

As we said above, Thrun et al. [2005] argue that robot localisation schemes should represent pose beliefs probabilistically. That is, a probability density function should be maintained that associates with every pose the probability that the robot currently occupies that pose. But how should this belief be updated given movement commands and sensor information? Thrun et al. [2005] demonstrate that a Bayes filter is an appropriate general solution to this problem. When a new control command is carried out, the Bayes filter specifies in general how to update the pose belief distribution using the robot's state transition probability distribution. This transition distribution describes the probability of a robot being in a new state (i.e. pose) given that it was in a particular prior state and a particular motion command was issued. The new pose distribution is frequently called the prediction. After the prediction is computed, the robot typically takes sensor readings in its new pose. These sensor readings should influence the belief that the robot is in a given pose. The Bayes filter algorithm gives a general form for the computation which updates the prediction given a measurement. This measurement update requires a measurement probability distribution which gives the probability that the robot receives a particular sensor reading given that it is in a particular state. The resulting final belief is a posterior probability distribution given a control command and subsequent sensor measurement. Thrun et al. [2005] show that for general pose distributions, no closed form solution exists for computing predictions and measurement updates. The extended Kalman filter which we describe next can be derived from the Bayes filter by assuming that pose beliefs are normally distributed. We shall see that this assumption, though, renders the extended Kalman filter most appropriate for pose tracking (at least in basic implementations) rather than full global localisation.

2.2.1.1 The Extended Kalman Filter

The extended Kalman filter (EKF) is quite a popular solution to the pose tracking problem according to Thrun et al. [2005]. We use an extended Kalman filter to augment our robot tracking system described in Chapter 6. Before discussing EKF and its use in pose tracking, we will first cover its close cousin the Kalman filter (KF). The KF is a recursive linear state estimator. A state in the context of the localisation problem is the robot's pose. The KF treats state as a random variable with a Gaussian distribution. The Gaussian distribution is a unimodal probability density function parametrised by a mean vector and covariance matrix. The mean coincides with the peak of the distribution and the covariance matrix indicates the spread of the distribution. The KF estimates the current state of the system given the prior state, the control command that brought the system from the prior state to its current state, and sensor measurements of the system in the current state. The KF is described as recursive because it estimates the current state from the previous state only, not the entire state history of the system. We shall discuss the linear aspect of the Kalman filter below.

The Kalman filter requires two functions to be defined: a state transition function and a measurement prediction function. The state transition function predicts what the current state of the system will be when a given control signal is applied to the previous state. In the context of robotic pose tracking, the control signal is a movement command (or odometric information resulting from a movement command) applied to change the pose of the robot. The measurement prediction function takes as input the predicted current state of the system (i.e. the most recent output from the state transition function) and predicts what the sensor measurement of the system will be in this state. State transition and system measurement are assumed to be stochastic processes in problems to which the KF is applied. After all, if state transition were not a stochastic process and the initial state of the system were known, then the state transition function alone could be used to track the evolving state of the system; the KF would not be required. The KF requires knowledge of the randomness involved in state transition and sensor measurement. The uncertainty of each is considered an added random variable drawn from a zero mean Gaussian distribution; the covariance matrix of each distribution is defined by the user of the KF.

In robotic pose tracking using variants of the Kalman filter, sensor measurements often take the form of the range and/or bearing of one or more landmarks as sensed from the robot. Thrun et al. [2005] present a Kalman filter-based localisation algorithm which uses both range and bearing measurements to landmarks. Range and bearing information can be extracted from, for example, stereo camera images. Durrant-Whyte [1994] shows how to track pose using range-only (e.g. acoustic beacon receivers) and bearing-only sensors (e.g. non-stereo cameras).

The measurement prediction function requires a map of the environment in which the robot is operating. Thrun et al. [2005] points out that in much published work the map consists of a list of important features or landmarks in the environment. Associated with each landmark is a location in map-based coordinates and what Thrun et al. [2005] calls a feature signature, essentially a description of that landmark for use in landmark recognition. The measurement prediction function takes the robot's predicted pose and the map and computes the range and/or bearing to a selected set of landmarks. Predicting range and/or bearing to point landmarks on a map from a given pose expressed in map coordinates is a relatively simple calculation involving basic trigonometry. Appropriate formulae can be found in Thrun et al. [2005].

The measurement prediction function above assumes that the so-called data association problem has been solved. The data association problem involves matching an environmental feature whose signature is gleaned from sensor data with a landmark stored in the map (Thrun et al. [2005]). When artificial landmarks are used, the solution to this problem is often trivial. Each satellite in the global positioning system broadcasts its identity along with data used to find the satellite's range. Durrant-Whyte [1994] describes reflective beacons which can be identified by the unique "bar codes" laid down in reflective tape on their surface. Data association is more difficult for naturally occurring features. We shall discuss techniques for feature identification later in this review. For now, we shall assume unless otherwise noted that a reliable solution to the data association problem is employed.

Once the state transition and measurement prediction functions have been defined, the Kalman filter can be applied. The user of the filter must first provide an initial belief about the state of the system. This belief is of course expressed as a normal distribution. In the context of robot pose tracking, this initial distribution describes an estimate of the robot's initial pose in map-based coordinates as well as the user's uncertainty about this initial pose. In the implementation of Thrun et al. [2005], the KF waits until a control signal is applied to change the state of the system and a measurement of the system is taken in this new state. This control signal and measurement are passed to the KF. The KF first predicts the state of the system given the prior state estimate and the control signal applied. This prediction – like all beliefs about the state of the system in the context of the KF – is expressed as a normal distribution. The mean of the predicted distribution is simply that predicted by the state transition function assuming no error in the application of the control signal. The covariance matrix of the predicted state is a function of the covariance matrix of the previous state estimate and the zero-mean Gaussian distribution expressing the uncertainty in the state transition we described earlier. Obviously, larger state transition uncertainties will lead to larger uncertainties in the predicted current state.

The KF next corrects this system state prediction using predicted and actual sensor measurements. The algorithm first computes the difference between the predicted and actual sensor measurements, a variable often called the innovation. The innovation gives a measure of the difference between the true and predicted current states of the system (Durrant-Whyte [1994]). If the innovation is relatively large (and measurement uncertainty is relatively small), then the predicted state is probably quite different from the actual state and so the predicted state must be drastically altered. The mean of the belief about the true state of the system is computed as the sum of the predicted state and the innovation weighted by a value known as the Kalman gain.

The Kalman gain is a matrix whose value depends on the zero mean Gaussian noise associated with state transition and system measurement described above. Elements of the gain matrix are generally larger if the uncertainty in state transition given a control signal is larger than the uncertainty in system measurements (Durrant-Whyte [1994]). Thus, when sensor uncertainty is relatively large, less weight is given to the innovation in the computation of the current state estimate. When transition uncertainty is relatively large, the Kalman gain ensures that the predicted state of the system plays a relatively small role in the current state estimate.

The result of the weighted sum described above is a belief about the current state of the system (e.g. the robot's pose in the context of pose tracking) expressed as a normal distribution.

The Kalman filter requires that the state transition and measurement prediction functions be linear in their input variables. This is because both functions transform the normally distributed random variable which represents the robot's pose belief. When a Gaussian random variable is transformed by a linear function, the result is another Gaussian random variable; this cannot be guaranteed if the transforming function is nonlinear. The Kalman filter depends on the system state remaining a Gaussian distribution (Thrun et al. [2005]). Unfortunately, this linearity condition is often violated in robot pose tracking problems. The state transition for example usually involves some trigonometric functions as can be seen in Durrant-Whyte [1994]. So too, range predictions use the Euclidean distance formula, a nonlinear function.

Due to this nonlinearity, the extended Kalman filter (EKF), rather than the Kalman filter, is often used to solve robot pose tracking problems. The EKF is quite similar to the KF. The big difference is that the former allows the state transition and measurement prediction functions to be nonlinear. The EKF requires that these functions be linearised about their state input. This linearisation is accomplished by estimating each function with a first order Taylor series approximation. This approximation is essentially the tangent to the nonlinear function at the point of the Gaussian mean vector. The success of the extended Kalman filter for pose tracking depends on how well this linear Taylor series approximates the underlying nonlinear function.

The extended Kalman filter is an attractive solution to the pose tracking problem because it provides an efficiently computable closed-form solution to the recursive update of pose beliefs (Thrun et al. [2005]). It is possible to derive this closed-form solution because the robot's belief about its pose is represented by the Gaussian distribution. The computation time of the EKF is $O(k^2.4 + n^2)$ when efficient matrix operations are employed (Thrun et al. [2005]) where k is the size of the output of the measurement prediction function and n is number of state variables to compute. In this case of planar pose tracking, $n = 3$. As we discuss below, though, it may not always be advisable to represent pose beliefs with Gaussian distributions.

The EKF tracker works by linearising the robot's motion and measurement models as discussed above. This is done so that the belief output by the EKF remains a Gaussian distribution after each iteration of the algorithm. The unscented Kalman filter

(UKF) provides another way to solve this problem. The UKF represents the Gaussian pose belief distribution as a set of $2n+1$ sigma points where n is the dimensionality of the Gaussian. In the case of the pose tracking problem, n is equal to three when the robot is moving on a plane since the robot's pose is composed of a two-coordinate location value and a heading value. The first sigma point is located at the mean. The other sigma points are distributed symmetrically about the mean. With knowledge of the sigma points, it is easy to recover the mean and covariance matrix which describes the Gaussian distribution which the sigma points represent. In the UKF, the sigma points are used as arguments to the (generally) nonlinear motion and measurement models. The output of these models is a new set of sigma points which can be used to compute the mean and covariance matrix of new Gaussian distributions. Thrun et al. [2005] demonstrate that the sigma point method is generally more accurate than the approximation provided by the Taylor series linearisation employed by the EKF.

The EKF for pose tracking as discussed above assumes a reliable solution to the data association problem. Thrun et al. [2005] in fact note on page 230 that "a single false correspondence can derail the [EKF] tracker by inducing an entire stream of localization and correspondence errors." The multi-hypothesis tracker (MHT) was introduced to overcome this brittleness in the EKF tracker. The MHT represents the robot's belief about its pose with a weighted mixture of multiple Gaussians. Each Gaussian results from a unique sequence of feature correspondence assignments processed by an EKF-like algorithm. The weight associated with a given Gaussian indicates the likelihood that it represents the robot's true pose. This weight can be used to prune unlikely pose beliefs, thus reducing the computational requirements of the algorithm. The MHT, unlike the EKF tracker, can be used to solve global localisation problems by providing an initial set of pose beliefs which covers the environment approximately uniformly.

2.2.1.2 Monte Carlo Localisation

The extended Kalman filter and unscented Kalman filter are appropriate for pose tracking not global localisation. This is because they represent the robot's belief about its pose with a unimodal Gaussian distribution. Monte Carlo Localisation (MCL) abandons the use of the Gaussian distribution to represent pose beliefs. Instead, the pose distribution is represented by a finite set of N weighted particles. Each particle coincides with a particular robot pose. A particle's weight (also called its importance factor) is proportional to the probability of receiving the current sensor data given that the pose represented by that particle is correct; more on this conditional probability below. MCL is a non-parametric algorithm in the sense that it does not try to estimate belief with a parametrised form like the Gaussian distribution.

The first step in the MCL algorithm involves choosing the initial set of particles. If MCL is used to solve a global localisation problem, the initial particle set can be drawn randomly from a uniform distribution of poses over all permissible locations and orientations (Thrun et al. [2001]). Each particle weight is set to $\frac{1}{N}$ where N is the total number of particles. This weighting reflects complete initial uncertainty in the pose of the robot. If there is some knowledge about the initial pose of the robot, this can be used to influence the initial sampling of particles. In fact, MCL can be used to solve both pose tracking and global localisation problems.

Once the initial particle set is created, the MCL algorithm in Thrun et al. [2005] updates the set each time the robot moves and collects sensor data. After a move is made, each particle in the set is updated given the movement command or odometric information which results from the move. This update requires knowledge of the robot's motion model: the probability distribution of the robot's current pose given its previous pose and the motor command applied at the previous pose. Thrun et al. [2001] point out that in order to make the pose updates it is sufficient to have a function which, given a prior pose and motor command, returns a randomly generated pose drawn according to the motion model. A closed-form expression of the motion model is not required. Once the particle poses have been updated, the set of unweighted particles is typically called the proposal distribution (Thrun et al. [2001]). The importance factor of each particle in the proposal distribution is then calculated from the sensor model: the probability of the current sensor reading assuming that the hypothetical particle reflects the robot's true current pose. This probability computation requires a map of the robot's environment.

Once the N hypothetical particles and their weights are computed, the MCL algorithm undertakes a resampling process (also known as importance sampling). In this step, N particles are selected from the hypothetical set of particles described above. A particle is selected at random with a probability proportional to its weight. Particles are drawn with replacement so a particle can, and often is, drawn more than once. After resampling, the set of particles is an approximation of the posterior probability distribution of robot poses given all movement commands issued and sensor data collected so far.

One might wonder why resampling is done at all as it leads to duplicate particles appearing in the particle set. If resampling were not done, then poses with quite low probability of being the true robot pose would be represented in the particle set. These unlikely particles would be processed for each iteration of the MCL algorithm, almost certainly wasting computational effort.

Thrun et al. [2005] note that choosing the value of N, the number of particles in the sample set, is something of an art. If N is too small, then there is a risk that no particle in the initial set will represent a pose near enough to the true initial pose of the robot. In general N grows exponentially with the number of state variables being estimated. A large value of N will come at a computational cost as each iteration of the MCL algorithm carries out $O(N)$ operations. Care must be taken by the human operator to choose a value of N which is neither too big nor too small for the localisation problem at hand.

We saw above how MCL can be used to solve pose tracking and global localisation problems. The kidnapped robot problem can be tackled with MCL by periodically injecting random poses into the particle set. If the robot has not been kidnapped, then these particles will quickly die out in the course of importance sampling. If the robot has been kidnapped, then hopefully one of the newly injected particles will be similar to the pose of the kidnapped robot. Sophisticated techniques exist for choosing how many particles to inject and for calculating the pose distribution from which the particles should be drawn. We consider the description of these techniques outside the scope of this review. The interested reader can find more information about these methods in Thrun et al. [2005].

Because it is easy to understand and applicable to many localisation problems in robotics, Thrun et al. [2005] on page 250 call MCL "one of the most popular localization algorithms in robotics." MCL works quite well in practice. Dellaert et al. [1999b] demonstrated that global localisation with a particle filter was able to localise a robot moving through the Smithsonian museum on a 2 kilometre journey. The museum was empty at the time so the environment remained relatively static. The robot used a laser range scanner and was equipped with an occupancy grid map of the environment. We shall below discuss some MCL localisation approaches employing visual sensors.

2.2.1.3 Range Scanning and Map Matching

The localisation solutions we have seen so far typically use relatively sparse metric maps populated by important features (i.e. landmarks) in the environment of interest. A more dense map is also used: the occupancy grid. According to DeSouza and Kak

[2002], the most commonly used type of metric map in visual robotic navigation is the occupancy grid introduced by Moravec and Elfes [1985]. An occupancy grid is typically a two-dimensional array of cells where each cell represents a unique area of physical space in an environment. In the simplest form of the occupancy grid, each cell stores a binary value indicating whether the space associated with that cell is occupied or empty (Thrun et al. [2005]). In the formulation of Moravec and Elfes [1985], cells store a value between -1 and 1. A negative cell value indicates that the cell is probably empty, where the probability of emptiness is equal to the absolute value of the number stored in the cell. Likewise, a positive cell value indicates a belief that the cell is probably occupied. Cell value zero indicates that no information has been gathered about the physical space represented by the cell. This probabilistic formulation is required because occupancy is inferred from a robot's noisy sensor readings (sonar in the case of Moravec and Elfes [1985]). We shall discuss robot map-making in more detail later in this review.

Range scan matching employs a local occupancy grid for the purpose of pose tracking. Range scan matching for pose tracking was introduced in Lu and Milios [1994] and fleshed out in Lu and Milios [1997]. Lu and Milios [1997] define a range scan as a set of robot-object distances in a panoramic 2D slice of the environment. Each element of a scan is a point consisting of the range from the scanner to the sensed object and the angle (in ego-centric terms) that the sensor was in when that range value was captured. Lu and Milios [1997] use a laser range finding sensor to generate range scans. The scan matching algorithm aims to solve the following question: if a reference scan S_{ref} is captured while the robot is in a pose P_{ref} and sometime later another scan S_{new} is captured while the robot is in a different pose P_{new}, can we determine the change in pose from P_{ref} and P_{new} given odometric information and the change between scans S_{ref} and S_{new}? It is assumed in Lu and Milios [1997] that the robot is moving in a two-dimensional environment. This in fact is quite a similar problem to that tackled in visual homing and the solutions of Lu and Milios [1997] are reminiscent of the image-based solutions to visual homing described below (particularly image warping).

Lu and Milios [1997] provide two solutions to the range scan matching problem. Both algorithms begin by transforming (rotating and translating) the reference scan into an approximation of the new scan using the rotation and translation provided by odometric information. We shall in the following discussion refer to this transformed reference scan as simply "the reference scan" for the sake of brevity.

One approach to scan matching devised by Lu and Milios [1997] is based on the iterative closest point (ICP) algorithm of Besl and McKay [1992]. The algorithm stems from the fact that if the pose change from P_{ref} to P_{new} were known, then the reference scan could be transformed (rotated and translated) to match the new scan. Discounting sensor noise, the range of a particular point in the transformed reference scan would be identical to the range of the corresponding point in the new scan. We should note here that a point P'_i in the reference scan corresponds to a point P_i in the new scan when they both result from the laser reflecting off of the same physical location in the real world. Following this logic, Lu and Milios [1997] define the distance between the transformed reference scan and the new scan as the function $E_{dist}(\omega, T) = \sum i = 1^n |R_\omega P'_i + T - P_i|^2$. P'_i is one of the n points in the range scan S_{ref} and P_i is the supposed corresponding point in S_{new}. R_ω is the rotation matrix formed from the rotation angle ω. T is a translation vector. The function E_{dist} assumes point correspondences can be determined. Solving the correspondence problem reliably is not an easy task and we will discuss it in the context of image data in the visual homing literature review. Relatively simple approximate solutions to the correspondence problem used by Lu and Milios [1997] will be discussed below. The correct values ω and T (i.e. those which correspond to the change from P_{ref} to P_{new}) are considered to be those which minimise E_{dist}. Lu and Milios [1997] state that closed-form solutions can be derived to determine values of ω and T which minimise E_{dist} given a set of correspondences. Once the values of ω and T which minimise E_{dist} have been found, the scan S_{new} is transformed (rotated and translated) with these values. The process to find a pose transform to minimise E_{dist} is repeated with S_{ref} and the transformed S_{new}. This iteration continues until the change in E_{dist} from one iteration to the next falls under a certain threshold value set by the user. Lu and Milios [1997] report that between 15 and 20 iterations are required to compute good pose change estimates in their experiments.

Lu and Milios [1997] describe two complementary and relatively simple rules for quickly (in linear time) establishing correspondences between points in the new and reference scans. The closest-point rule was used by Besl and McKay [1992] to solve – at least approximately – the correspondence problem. The closest-point rule assumes that for each point P_i in the new scan, the corresponding point in the reference scan is that closest in Euclidean distance to P_i. Lu and Milios [1997] also define the matching-range-point rule to identify corresponding points. This rule is based on the observation that if the pose change is pure rotation, then the range values of P_i and P'_i will be equal (discounting noise). This equality will hold approximately if the translation element of the sought-for pose change is small. The matching-range-point rule additionally assumes that the rotational element of the pose change is bounded and fairly small. This rule therefore looks for a point P_i in the new scan whose range value is most similar to the range of P'_i within a bounded angular window around θ_i, the laser orientation at which P'_i was captured.

Lu and Milios [1997] found that the closest-point correspondence rule gives good scan matching results when the pose change from P_{ref} and P_{new} is dominated by translation; the matching-range-point rule is applicable for pose changes with small translation but relatively large rotation. With this observation in mind, Lu and Milios [1997] suggest the iterative dual correspondence (IDC) algorithm. In each iteration of the incremental scan matching algorithm, IDC creates two sets of correspondences between S_{ref} and S_{new}, one with the closest point rule and the other with the matching-range-point rule. Each of these correspondence sets will lead to a distinct rotation and translation estimate. The reported rotation estimate for the current iteration is that garnered using the matching-range-point rule. The reported translation estimate for the current iteration is that garnered using the closest-point rule.

The other solution of Lu and Milios [1997] to the scan matching problem is called the rotation search/least-squares method. As the name suggests, this method presents a two-tiered solution to the problem of finding pose change. The top tier consists of a search for the rotational element ω of the pose change. Given a candidate rotation ω, Lu and Milios [1997] derive an efficient method – discussed below – to compute the translation T for the assumed rotation ω that best explains the change between the new and reference scans. Lu and Milios [1997] point out that a distance measure between new and reference scans (when the reference scan is transformed using a candidate rotation ω and its associated translation T) is at a minimum for the true rotation between the reference and new poses. The distance function smoothly increases as the difference between ω and the true rotation increases. Thus Lu and Milios [1997] use a golden section minimisation method to find the true rotation between poses. This minimisation is reminiscent of the method used in visual compassing discussed in Section 2.3.5 to find rotational pose changes from image data.

For each candidate rotation ω, the rotation search/least-squares method uses a least-squares function to efficiently determine the best translation T for the assumed rotation. The first step in the translation-finding method is to rotate the new scan by the current guess of the value of ω. If ω is correct, then a point P_i in the new scan differs from the corresponding point P'_i only by the unknown translational element of the pose change. Unfortunately, this correspondence is unknown. Lu and Milios [1997] use the following method to solve the correspondence problem: for each point P_i in the rotated S_{new}, a corresponding point P_i' in S_{ref} is selected which lies on the intersection of S_{ref} and the line from the origin of the scans through P_i. The set of all correspondences is fed into a least-squares formula which is a function of ω (assumed to be known) and T. The authors use a closed-form equation to efficiently determine T from this least-squares formula.

Lu and Milios [1997] reported that the rotation search/least-squares and IDC scan matching algorithms described above are complementary. The former can deal with relatively large error in pose change estimates (due to noisy odometry) but is not very accurate. IDC is accurate but may not converge if the initial pose change estimate is not well known. Lu and Milios [1997] use the two algorithms therefore in sequence.

Lu and Milios [1997] note that – at the time of writing their paper – their noisy sensor data “makes it very difficult to reliably define or extract features” which could be used to solve the correspondence problem using a feature-based approach. This is why the authors take the whole-scan approach to scan matching described above. We shall discuss in more detail in our discussion of visual homing the problems with and some current solutions to feature-based correspondence.

Related to scan matching, map matching algorithms use occupancy maps to solve local and global localisation problems. We find a map matching example in Schiele and Crowley [1994]. The essential problem is matching a local ego-centric occupancy grid map of the environment (built on-the-fly using current range scan readings) to a particular portion of a global map. In Schiele and Crowley [1994] line segments are extracted from local and global maps using a Hough transform. The authors then find the robot pose which causes the local set of lines to correlate best with the global set of lines. This is essentially a data registration procedure.

2.2.2 Topological Navigation

A map is a model of the environment in which the robot is navigating. Broadly, two types of maps are employed by robotics researchers: metric and topological (Thrun [2002]). We have already seen examples of metric maps above (e.g. the occupancy grid). A metric map stores geometrically accurate information about the environment. A topological map on the other hand consists of a list of important places in the environment and usually some information on how the robot can travel between these places (like the distance between nodes). Thrun [2002] points out that the distinction between metric and topological maps is somewhat artificial, as topological maps often contain some geometric information as well. Also, the definition of nodes and links between nodes varies somewhat from author to author as we shall see below. Topological maps are often represented as graphs in which nodes represent places and arcs the connections between places. The London Underground map is a canonical example of a topological map. The map describes how to travel from one tube stop (i.e. node) to another, not the geometric relationship (i.e. distance) between tube stops.

In the spatial semantic hierarchy of Kuipers [2000], a topological map consists of a set of locally distinctive places linked by distinctive paths. A place is considered to be distinct in Kuipers [2000] if sensor readings indicate that it is sufficiently different from nearby places already included in a topological map. This distinctiveness allows the identified place to be easily revisited using an algorithm (e.g. visual homing) which attempts to move the robot so as to maximise local distinctiveness.

Gaspar et al. [2000] argue that topological maps are useful for robot navigation in relatively large-scale environments. Topological maps are typically much sparser than geometric maps of the same environment. Topological maps are appropriate when precise positioning along a path between nodes is not required. They are, in the words of Gaspar et al. [2000] (p. 890) a relatively "long-distance/low-precision" navigation solution. Booij et al. [2007] point out that in traversing between two (perhaps non-adjacent) nodes in a topological map, the robot will not necessarily take the shortest navigable path in physical space between the two mapped locations. Metric mapping schemes are generally able to provide shortest path information. Despite these limitations we shall see below that topological navigation provides impressive behaviour in several tests.

We shall first review the topological location tracking system described in Ulrich and Nourbakhsh [2000]. Though relatively simple, this work highlights several common problems in topological navigation. The navigation results are quite impressive; the robot is consistently localised in three large indoor and one large outdoor environment. The work won the Best Vision Paper Award at the IEEE International Conference on Robotics and Automation in 2000. In Ulrich and Nourbakhsh [2000] each node of the topological map of an environment represents a distinctive place as judged by the robot's human operator. Two nodes are connected if the robot is able to move between the places the nodes represent. Each distinctive place is represented by a sequence of panoramic colour reference images associated with that place by the human operator.

The main problem solved by Ulrich and Nourbakhsh [2000] is that of place recognition, a problem quite similar to visual homing. As the robot moves through a mapped environment starting from a known initial location, it must determine which map node it is in by finding the best match between the current image provided by its panoramic image and the set of map references images. Typically, the current image must be compared with several hundred reference images for a place recognition decision. Ulrich and Nourbakhsh [2000] take an appearance-based (as opposed to feature-based) approach to this problem. To make image comparison efficient, each image (both current and reference) is represented by a set of colour histograms. Not only can histograms be compared efficiently, they are also invariant to imager rotation and require little memory for storage. Rotational invariance is important because it allows reference and current images to be safely compared without knowing the relative orientations of the imager when reference and current images were captured. In other words, a compass is not required for localisation. After experimenting with several formulae to measure the distance between two histograms, Ulrich and Nourbakhsh [2000] identified the Jeffrey divergence as most suitable for their needs. As we shall see below, efficient image storage and comparison are problems common to most vision-based topological navigation schemes. Ulrich and Nourbakhsh [2000] found in their experiments that place recognition was successful between 87.5 and 97.7 percent of the time. They note that augmenting their algorithm to deal with place recognition under varying illumination would be valuable.

Gaspar et al. [2000] also present a topological navigation system based on visual imaging. Each map node corresponds to a place in an indoor environment where a special action may be taken like going through a door or turning a corner. Links correspond to parts of the environment where it is relatively easy for the robot to move from one node to another (i.e. corridors). Each node is associated with a single omnidirectional 128*x*128 grayscale image used to identify the location corresponding to that node. Along with each link is stored a sequence of images that the robot should experience while moving along that link.

The robot in Gaspar et al. [2000] performs a place recognition query to determine its position in the topological map described above. This map consists of a “very large” (p. 893) number of reference images representing an office environment (an image is captured every 50cm in the mapping process employed by Gaspar et al. [2000]). To reduce the computational effort required to compare a current image with so many reference images, Gaspar et al. [2000] reduce all images with a technique known as

Principal Components Analysis (PCA). PCA seeks to project n-dimensional data (images in Gaspar et al. [2000] have 128*x*128 dimensions) into an m-dimensional space (where *m* is typically much smaller than *n*) such that the projection retains most of the variance of the original. The *m* dimensional space is embedded in the higher *n* dimensional space. The lower dimensional space is defined by *m* so-called principal components which are essentially mutually orthogonal *n* element vectors. The first principal component is the direction of maximal variation through the original data; the second principal component is orthogonal to the first and is in the direction which maximises variation when the original data set is projected onto the plane formed by the first and second components; the remainder of the principal components are defined similarly. Principal components are the eigenvectors (called eigenimages in this context) of the covariance matrix formed with all of the reference images. Gaspar et al. [2000] then select the 10-12 eigenimages with the highest eigenvalues to form the low-dimensional space into which each input image is projected. References images are projected in this way in a pre-processing step. Each current image is projected into the same low-dimensional space before comparison with the reference images is made. Gaspar et al. [2000] use a correlation-based measure to compare projected current and reference images. This low-dimensional representation not only makes image comparison more efficient (10-12 pixels are compared rather than 128*x*128), it also means that the storage requirements of the map are dramatically reduced (as in Ulrich and Nourbakhsh [2000]). Gaspar et al. [2000] imply that their image similarity measure is not robust to changes in illumination. As demonstrated by Pajdla and Hlaváč [1999], low-dimensional projected images are not invariant to viewer rotation. Gaspar et al. [2000] avoid this problem by generally orienting the robot in the same direction during localisation and map-making.

Unlike Ulrich and Nourbakhsh [2000], Gaspar et al. [2000] tackle the problem of how the robot can autonomously move between map nodes (i.e. traverse map arcs). In Ulrich and Nourbakhsh [2000], the robot was driven by a human operator through a mapped environment while autonomously solving the place recognition problem. Gaspar et al. [2000] use a visual servoing approach (we shall discuss visual servoing in more depth in Section 2.4.1) to move the robot down corridor centres. A sequence of images along each corridor captured while map-building is used to provide evidence that the robot is making satisfactory progress down a corridor.

Argyros et al. [2005] present a vision-based topological navigation system somewhat different from that of Gaspar et al. [2000]. We note that similar though preliminary work was published by the same research group in Argyros et al. [2001]. Argyros et al. [2005] are essentially interested in a long-range homing problem. That is, the robot has an important home position (e.g. a docking station) to which it should return after a meandering journey to carry out some mission. This journey may be quite long-range, moving the robot through several different rooms (Argyros et al. [2005] are interested indoor navigation). The assumption is that the journey is too long for standard visual homing approaches to work successfully. Setting off from its home position, the robot records a series of panoramic images along its path. The system also tracks corner-point features in these images, noting for each image the ego-centric bearing of each tracked corner. When the robot is done with its mission, the system selects a sequence of recorded images as homing targets. Argyros et al. [2005] call the locations of these target images Milestone Positions (MPs). The robot then utilises a visual homing strategy to hop from the current MP to the previous MP in the sequence. The homing strategy is quite similar to the Snapshot Model discussed in Section 2.3.2.1. The solution to the feature correspondence problem presented by the Snapshot Model is trivial here; the bearings of corner point features are known in the images collected at MPs and can be easily tracked while homing to a target MP. Since corner point feature bearings are associated with each image, no external compass information is required to aid in solving the correspondence problem, as is the case with the original Snapshot Model. In work similar to Argyros et al. [2005], Smith et al. [2006] use the average landmark vector (ALV) homing algorithm (see Section 2.3.2.4) rather than the Snapshot Model to guide the robot between MPs spaced along a predefined route.

Goedemé et al. [2005b] present a topological navigation algorithm somewhat similar to that of Argyros et al. [2005]. Goedemé et al. [2005b] tackle the problem of following a path defined by a pre-collected sequence of non-localised images. Image capture locations are separated by between 1 and 3 metres. The authors use a visual homing algorithm to move the robot from one image to the next in the sequence. Visual homing is split into a two-stage process. In the first stage, feature correspondences are established between the current and snapshot images (here we use the terminology to describe images frequently employed in the visual homing literature). Goedemé et al. [2005b] employ SIFT features; we describe SIFT features in more detail in Section 2.3.3.6. Given the set of computed feature correspondences, Goedemé et al. [2005b] then use epipolar geometry to compute the movements required to move the robot from its current pose to the snapshot pose. Visual homing using epipolar geometry is discussed in Section 2.3.2.5. The system also infers the three-dimensional position of each feature and stores it in a local metric map whose origin is defined by the starting location of the current homing process. In the second stage, the robot moves given the homing vector generated in stage one. As the robot moves, the visual features used to generate the home vector are tracked in successive images. The robot is also localised in the local map. If a feature is lost due to occlusion, then it is artificially placed in the image given knowledge of the three-dimensional location of the feature computed earlier.

Unlike Argyros et al. [2005] and Goedemé et al. [2005b], Labrosse [2007] uses an image-based rather than feature-based visual homing algorithm to follow a route along which images have been collected previously. He in fact uses the difference surface-based homing method introduced in Binding and Labrosse [2006] and described in Section 2.3.4.2. Labrosse [2007] notes that image-based visual homing algorithms are attractive because they are more computationally efficient than feature-based methods which require feature extraction and correspondence.

2.2.2.1 Image-Based Localisation

A group of papers on so-called image-based localisation employs maps which in our view straddle the boundary between metric and topological representations of the environment. In image-based localisation, maps consist of nodes representing distinctive places in the environment (as with topological navigation). Each node is typically represented by a panoramic image taken from that place. In common with metric maps, each node is precisely localised in a Cartesian coordinate frame. As we shall see, image-based localisation techniques share many of the same problems seen in topological visual navigation: efficient storage of a large database of images and efficient comparison of a current image with the database for place recognition.

Crowley and Pourraz [2001], like Gaspar et al. [2000], use PCA to reduce the dimensionality of the set of reference images forming the map used for localisation. In this case, though, the set of reference images is well-localised and is used for metric pose estimation (with motion constrained to a two-dimensional plane) rather than for topological navigation (as is the case with Gaspar et al. [2000]). Crowley and Pourraz [2001] use perspective rather than panoramic images in their work. Before making pose estimations, they collect a grid of reference images; at each grid point, images from several camera orientations are collected, approximating one panoramic image. Crowley and Pourraz [2001] point out that the set of reference images are equivalent to points on a nonlinear manifold embedded in the (high-dimensional) space whose dimension is equal to the number of pixels in each image. Assuming the mapped environment is visually static, this so-called appearance manifold itself is three-dimensional; each point on the manifold represents an image taken while the imager is in a particular three-dimensional pose. Given this manifold, one could, given an image, uniquely determine the associated imager pose (assuming no perceptual aliasing in the mapped environment). This observation forms the basis of several image-based localisation techniques. Unfortunately, the appearance manifold cannot be known precisely for real-world environments. Crowley and Pourraz [2001] estimate the appearance manifold of a mapped environment using the set of reference images described above.

In order to estimate the pose of a new image given the set of reference images, the authors attempt to find the reference image with the highest correlation with the new image. This involves a brute-force search over the whole reference set. To reduce the computational complexity of this operation, the dimensionality of the reference set is reduced using PCA as mentioned above. The set of K basis eigenimages used to reduce each image in the reference set is also used to reduce the new image. A Euclidean distance measure is then used to measure the similarity between the new and reference images. The authors note that Euclidean distance in eigenspace is a good approximation to the correlation measure in the original space of non-reduced images. The pose of the reference image most similar to the new image is taken to be the pose of the new image. Crowley and Pourraz [2001] further improve the efficiency of the search by storing the reduced reference images in a tree structure; images in the same portion of the manifold are stored in the same tree leaf. A K-ary search is used to quickly find the leaf with images most similar to the reduced version of the new image.

The above localisation approach suffers from the fact that the pose estimation error depends on the density with which the reference set is sampled. A dense sampling is required for very accurate pose estimates. Of course, larger reference sets required more computational effort to perform brute-force searches. Crowley and Pourraz [2001] tried to circumvent this problem by interpolating the appearance manifold formed by a relatively sparse reference set. It was reported that such interpolation resulted in “only minimal loss in [localisation] precision.” (p. 748)

The image-based localisation method of Crowley and Pourraz [2001] assumes visually static environments. They write that their scheme is not robust in the face of illumination conditions (particularly illuminant direction) which change after the reference image set is captured. Image-based localisation with PCA in visually dynamic environments will be discussed below.

Related to the work of Crowley and Pourraz [2001], Yang et al. [2007] show that a local appearance manifold can be linearised and this linear approximation can be used to track the pose of a moving imager. The imager in Yang et al. [2007] is free to take any pose in three dimensional space in a static world. Yang et al. [2007] show how to approximate the local appearance manifold using seven images which are proximate on the manifold. They construct a special imaging sensor consisting of four adjacent cameras which is capable of simultaneously taking the seven required images at known relative poses. Once the local manifold approximation is computed, the system waits for the camera to move to a pose and take a new image. Given the new image and the manifold approximation, the pose change can be computed as the solution to a set of linear equations.

Pajdla and Hlaváč [1999] tackle the problem of finding an image representation which is invariant to imager rotations for the purposes of image-based localisation. We saw why rotational invariance was important for image-based localisation when discussing Ulrich and Nourbakhsh [2000]. Pajdla and Hlaváč [1999] introduced the zero phase representation (ZPR) of panoramic images and showed that ZPR images are rotationally invariant. In Pajdla and Hlaváč [1999] panoramic images are formed with a digital camera viewing a convex mirror (as described in Section 2.3.1). Pajdla and Hlaváč [1999] map the image of the mirror (a circle) from polar to rectangular coordinates, "unwrapping" the mirror image to an equivalent rectangular image. A shift in the columns of this unwrapped image is equivalent to a rotation of the imager. The ZPR procedure finds, for a particular image, the columnwise shift required so that the lowest frequency element of the Fourier transform of the image has a phase value of zero. Pajdla and Hlaváč [1999] shows that two panoramic images taken in different imager orientations but from the same position will have nearly identical ZPR representations (assuming illumination and object locations remain static).

Jogan and Leonardis [1999] extend the work of Pajdla and Hlaváč [1999] by reducing the dimensionality of ZPR images using principal components analysis (PCA is described above). Jogan and Leonardis [1999] show that image-based localisation with reduced ZPR images should have approximately the same place recognition success rate as when all images (current and reference) are captured in the same compass orientation. Jogan and Leonardis [1999] also show that a competing image representation which is invariant to rotation – the autocorrelation image (Aihara et al. [1998]) – is inferior for the purposes of image-based localisation.

Jogan and Leonardis [2000] argue that using PCA to compress images results in image comparisons which are not robust to object movement between capture of reference and current images. They attempt to solve this problem by randomly subsampling a current image. The *k* pixels chosen in the subsample are those which can be generated with little error from the eigenvectors associated with the reference images. The subimage produced thus tends to include pixel locations which are not affected by objects moving in the environment. Jogan and Leonardis [2000] showed that fairly accurate image-based localisation is possible in environments with moving objects using their scheme in place.

Jogan et al. [2002] point out that PCA-based compression is also sensitive to illumination changes between current and reference (i.e. map) images for the purpose of place recognition. Jogan et al. [2002] point out that gradient-based filters when applied to images make these images more robust to illumination changes for the purpose of similarity measurement. These gradient-based filters basically highlight edges in images. Jogan et al. [2002] show how gradient-based filters can be applied to create eigenimages which are resilient to illumination changes. When doing image-based localisation, the current input image must be filtered with the same set of gradient-based filters as the eigenimages. It is demonstrated that localisation is vastly improved when this method is used in an indoor environment with dynamic illumination.

Menegatti et al. [2004] point out that many approaches to image-based localisation like Jogan and Leonardis [1999] would not be reliable for global localisation in environments suffering from perceptual aliasing in the visual mode. Perceptual aliasing occurs when two or more node locations appear similar enough to one another so that a place recognition system cannot tell them apart using only image comparisons. A corridor environment with repeated identical doorways would present a perceptual aliasing problem for a robot using vision alone. The authors thus augment their image-based localisation algorithm with a Monte Carlo localisation algorithm (see above for a general overview of MCL).

Before describing their version of MCL, we shall look at how Menegatti et al. [2004] perform image compression and comparison. Menegatti et al. [2004] initially capture grayscale panoramic images of the environment for both map-making and later localisation. Using a method presented earlier in Ishiguro and Tsuji [1996], Menegatti et al. [2004] use the discrete Fourier transform (DFT) to compress panoramic images. The discrete Fourier transform (Fisher et al. [1996]) of each row of each image is first computed. The magnitudes corresponding to the 15 lowest frequencies for each row are stored. Using the DFT, Menegatti et al. [2004] convert a 512*x*80 pixel grayscale panoramic image into a 15*x*80*x*2 element representation, reducing storage requirements by a factor of about 17. Not only are the transformed images small, the Fourier transform renders them rotationally invariant so that robot heading need not be taken into account when comparing reference and current images (as is the case in Ulrich and Nourbakhsh [2000]). Menegatti et al. [2004] compute image similarity by summing the absolute difference between the corresponding Fourier magnitudes of current and reference images. This is similar to a sum-of-squared differences image similarity measure.

The MCL algorithm used by Menegatti et al. [2004] follows the broad outline described above. We shall merely highlight some peculiarities here. Each particle represents an hypothetical robot location in map coordinates (not, as in several other MCL applications, a full pose as reference images are rotationally invariant). A particle's weight is related to the image similarities between the current image and the set of reference images deemed to be close (in terms of Euclidean distance) to the location of the particle. The algorithm is shown to be able to solve global localisation and kidnapped robot problems in an environment in which perceptual aliasing is present.

The work of Menegatti et al. [2004] is similar to the influential work of Dellaert et al. [1999a]. A difference between the two is that Dellaert et al. [1999a] use a dense map of upward facing images, essentially a visual mosaic of the ceiling of the mapped environment (a museum).

2.2.3 Autonomous Mapping

As we saw above, most localisation schemes require a map of the environment. Much work in recent years has gone into crafting algorithms to allow robots to create maps autonomously. As visual homing concerns localisation rather than map-making, we shall only briefly review recent trends in robotic map-making. Autonomous map-making presents an immediate problem: in order for the robot to map the environment beyond its immediate perceptual range, it needs to move and collect sensor readings while in various poses. To incorporate those readings into a global map, the robot needs to localise itself. But to localise itself, the robot (at least in most implementations) requires a map as we saw above. This chicken-and-egg problem is solved by a simultaneous localisation and mapping (SLAM) algorithm. SLAM is also sometimes called concurrent mapping and localisation (CML) in the literature.

As with localisation algorithms, successful SLAM methods are probabilistic in nature (Thrun [2002]). This is because the sensors (e.g. cameras, laser range finders, sonar) that robots use to acquire information about the environment for map-making return noisy data. So too, the technology used to track the robot's pose over time (e.g. wheel encoders and inertial navigation units) provide uncertain estimates of this pose. As with localisation, the current belief about the state of the world is represented by a probability distribution. In SLAM, the state consists of both the robot's current pose and (in one form of the algorithm) the locations of all observed landmarks (Durrant-Whyte and Bailey [2006]). These states variables are expressed in a suitable global reference frame. Robot pose and landmark locations are estimated jointly and their probability distribution is conditioned on all robot control commands and sensor observations made up to the current time.

Just as with probabilistic localisation, the current state distribution of the SLAM variables is updated recursively using some variant of the Bayes filter. Just as with localisation, the extended Kalman filter (EKF) is a popular realisation of the Bayes filter for use in SLAM (Durrant-Whyte and Bailey [2006]). To remind the reader: the EKF algorithm assumes that the state variables have a joint normal distribution. In addition, robot motion and sensor measurements are assumed to suffer from noise which has a zero-mean normal distribution. A complete solution to SLAM using the extended Kalman filter can be found in Dissanayake et al. [2001].

SLAM using the EKF has both benefits and drawbacks. A naive solution to SLAM using EKF requires $O(K^2)$ computations per map update (Thrun [2002]) where K is the number of landmarks in the map. Maps of large-scale environments can contain thousands of landmarks (Durrant-Whyte and Bailey [2006]) so these computational requirements are non-trivial. SLAM with the EKF is also very sensitive to errors in data association (Durrant-Whyte and Bailey [2006]). A big advantage of the Kalman filter approach to SLAM is that it maintains not only a state estimate but also a measure of the uncertainty about the state. This uncertainty can be used to judge how useful the estimated map is for navigation (Thrun [2002]). An optimised real-time EKF-SLAM algorithm has been successfully applied in a large-scale (i.e. tens of metres) outdoor environment (Guivant and Nebot). Other impressive results for EKF-SLAM in indoor, outdoor and undersea environments exist in the literature (Durrant-Whyte and Bailey [2006]). SLAM is often applied to robotics with range-measuring sensors. An impressive example of EKF-SLAM with a single moving camera is given in Davison [2003].

The above discussion of SLAM assumed that metric maps of the environment were being created. The autonomous creation of topological maps has been studied extensively as well. We shall review some impressive implementations of vision-based topological map-making. Franz et al. [1998a] present an autonomous topological map-making system. The robot moves to explore a space. If the current view is sufficiently different from all images stored in the map (or the map is empty), the current view is stored in a new node in the map. Image similarity is measured with a simple correlation measure. Image storage and comparison are made more efficient by considering only panoramic horizon line images; see Section 2.3.1 for more information on horizon lines. The newly created node is connected by an arc to the previously created node in the map. The system as described is capable of learning linear chains of snapshot locations. It may be the case that the new node is geometrically near other nodes in the map, not just the previously created node. To determine whether this is the case, the similarity between the new node's image and all existing map images is checked. If sufficient similarity is detected, then the robot attempts to home from the current location to the proposed neighbour. Franz et al. [1998a] use the image warping algorithm to home (see Section 2.3.4.1). If homing is successful, then a new arc is added in the map between the nodes in question.

Košecká et al. [2003] call their topological map a location graph. Each node in the graph represents an area of the environment (an office building) with a similar appearance over different poses. During an initial mapping phase, a robot moves through this environment (controlled by a human operator) and captures a grayscale perspective image approximately every two metres. The similarity between successive images is computed. In order to assess image similarity, Košecká et al. [2003] first extract the edges from each grayscale input image. They use edge images because they are relatively invariant to illumination changes in the environment. Košecká et al. [2003] next compute edge orientation and produce an edge orientation histogram. The similarity between two images is then measured by the similarity between their gradient orientation histograms. Košecká et al. [2003] use the χ^2 measure to rate the divergence between the histogram distributions. They find that χ^2 allows for robust discrimination between images in a real-world dynamic indoor office environment. Images collected during the mapping phase are automatically clustered using the χ^2 similarity measure. These clusters define the nodes of the location graph. Several images may be present in each cluster. Once the topological map is finalised, place recognition is achieved by using the χ^2 formula to measure the distance between a new image taken in an unknown pose and the set of reference images. Košecká and Li [2004] compare edge orientation histograms and SIFT features for the purposes of place recognition; they found that the latter provided better recognition performance, being more robust in the face of viewpoint change.

A very recent topological mapping system is presented in Booij et al. [2007]. Unlike the work of Košecká et al. [2003], Booij et al. [2007] associate a single image taken from a particular pose in the environment with each node in the topological map. The "length" of an arc connecting two nodes indicates how similar their associated images are. The authors note that short links indicate that the locations the nodes represent are likely to be geometrically close. As in Košecká and Li [2004], Booij et al. [2007] extract SIFT features to measure the similarity between two images. Booij et al. [2007] rate the similarity between two images using the number of corresponding SIFT features between the two images.

Booij et al. [2007] make an interesting statement comparing the relative merits of vision-based geometric SLAM and topological mapping: "The difficulty of [geometric SLAM] solutions is that the 3D positions of the landmarks are used while the camera only provides bearing information. Also the number of landmarks grows when the environment becomes larger, making it impossible to maintain a consistent state and covariance estimate." (p. 2) The work of Davison [2003] (mentioned above) suggests that the first statement is untrue; this author showed that EKF-SLAM with a monocular camera providing only landmark bearing information can work well. The second statement indicates a real advantage of topological over metric mapping.

2.3 Visual Homing

As we discussed in Chapter 1, all visual homing algorithms are somewhat similar. Each requires a homing agent to capture an image I_S at goal position S. When seeking to return to S from a nearby position C, the agent captures image I_C and uses the discrepancy between I_S and I_C to infer the homing vector $\vec{H}$. I_C and I_S are usually two-dimensional panoramic intensity images, though as we shall see in the following discussion this is not always the case. We shall assume that a monocular vision system is used. The orientation of the agent at S is typically different than its orientation at C. Most homing schemes (though not all; see e.g. Section 2.3.4.1) require that I_C is rotated to account for this orientation difference. Orientation correction is sometimes done with the aid of an external compass reference.

Since no metric landmark information is used, the homing vector $\vec{H}$ is often inaccurate. The agent thus moves by some distance (either fixed or calculated based on current sensor information) in the direction of $\vec{H}$ and repeats the process described above. The algorithm terminates when the discrepancy between I_S and I_C (or some value related to this discrepancy) falls below a certain threshold (again either fixed or calculated while homing).

Broadly, the discrepancy between I_S and I_C can be calculated in one of two ways. Some methods extract salient features (which we also call landmarks in this work) from I_C. These features must be identified in I_S (i.e. the correspondence problem must be solved). As the navigating agent has a monocular vision system, the change in feature bearing and/or apparent size is used to infer the homing vector. Landmark range is difficult to estimate with monocular vision and not often used. We shall not consider homing with stereo vision (e.g. Sturzl and Mallot [2002]). Since consistent feature correspondence is often difficult and computationally intensive, so-called image-based or appearance-based visual homing methods have been developed which compare the entirety of I_C and I_S to produce image discrepancy. We are interested in an image-based method in this work.

2.3.1 Panoramic Imaging

Most of the homing algorithms implemented in simulation or on actual robots capture panoramic images of their environments. These images are useful because they provide visual information which is not dependent on the orientation of the agent, as would a single field-of-view perspective camera. With a few exceptions which we shall discuss below robotic homers capture panoramic pictures with a CCD (Charge Coupled Device) camera imaging a hemisphere, cone, paraboloid or hyperboloid with a reflective surface. The merits of various mirror shapes are discussed in Nayar [1997]. The mirror reflects light from 360 degrees horizontally and 90 degrees or more vertically, forming a panoramic (though not completely omnidirectional) image of the environment. A schematic mirror and camera rig is shown in cross-section in Figure 2.1; the mirror is hyperboloid in shape.

The image of an actual hyperboloid mirror is shown in Figure 2.2(a). The mirror rig is situated in a laboratory environment. Figure 2.2(a) contains an image of the panoramic mirror *and* the rig supporting the mirror and camera. Homing researchers typically mask out the portions of the image which do not correspond to objects in the environment, as these do not contain useful landmark information. The mirror rig is

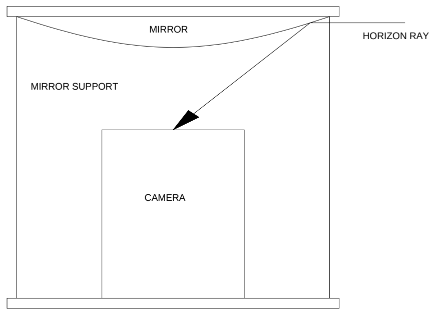


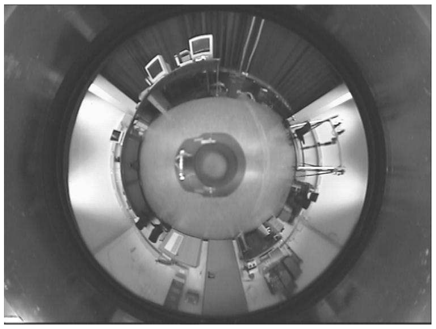
Figure 2.1: Schematic of hyperboloid mirror (in cross-section) imaged by a CCD perspective camera.

also masked for this reason. A mask for Figure 2.2(a) is shown in Figure 2.2(b) and the masked version is given in Figure 2.2(c).

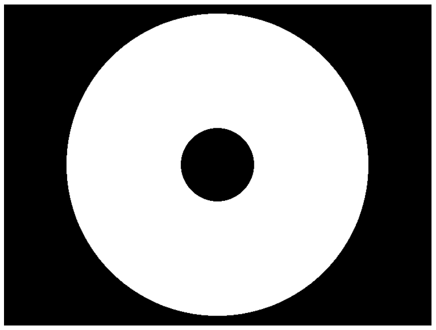
Some homing algorithms (see e.g. Zeil et al. [2003]) simply use the image of the mirror itself, suitably masked, (i.e. Figure 2.2(c)) when computing image disparity. Other algorithms (e.g. Franz et al. [1998b]) extract the so-called horizon circle from the panoramic mirror image; a sample horizon circle is drawn in black in Figure 2.3(a). The horizon circle is that portion of the mirror which reflects incoming rays of light perpendicular to the long axis of the mirror rig through the camera's aperture; such a ray is depicted in Figure 2.1. If the geometry of the mirror and the distance between the camera and the mirror are known, the radius of the horizon circle can be calculated with precision. A useful property of the horizon circle is that objects imaged in the horizon circle when the rig is in one position will also be imaged from a different position so long as both positions are on the same plane and the orientation of the rig's long axis stays the same.

Since the mirror rig can change in orientation and height slightly while the agent moves, workers typically extract an horizon annulus from the image of the mirror; see Figure 2.3(a). The inner and outer radii of the annulus are coloured in white, enclosing the black horizon circle. The extracted annulus (Figure 2.3(b)) is then averaged column-wise to estimate the intensity signal of the horizon circle (Figure 2.3(c)).

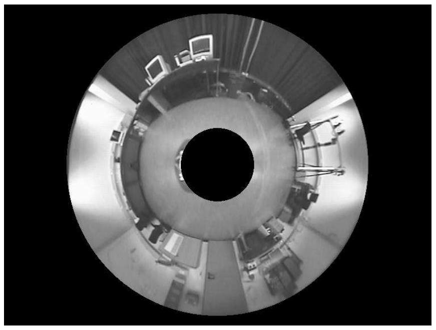
Rather than use the camera and mirror combination described above, some homing robots are equipped with a single perspective CCD camera and take multiple images while the robot rotates 360 degrees in place. The images are stitched together with special software, forming a rough panoramic image of the environment. Weber et al. [1998] describe such a system. This method for capturing panoramic images is more time-consuming – due to the camera rotation and stitching of multiple images – than



(a)

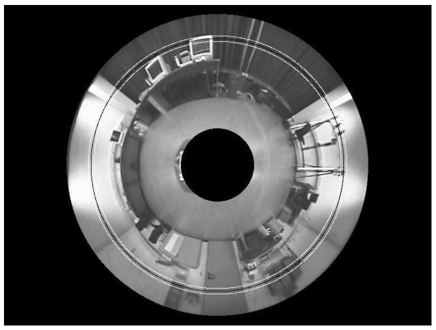


(b)



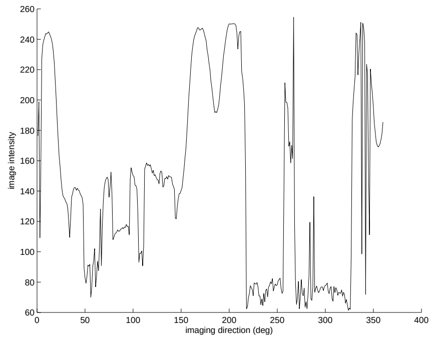
(c)

Figure 2.2: **(a)** Image of hyperboloid panoramic mirror situated in a laboratory environment. The image was download from http://www.ti.uni-bielefeld.de/html/research/avardy/index.html. **(b)** Mask to remove non-mirror portions of the image shown in (a). **(c)** Masked version of the image shown in (a).



(a)

(b)



(c)

Figure 2.3: **(a)** Image of hyperboloid panoramic mirror situated in a laboratory environment. The horizon circle for this mirror and camera combination is drawn in black. The horizon annulus – inner and outer radii drawn in white – encloses the horizon circle. **(b)** Extracted horizon annulus. **(c)** Horizon circle generated by averaging the horizon annulus column-wise.

that described previously in this section.

The Ladybug spherical digital video camera (www.ptgrey.com) offers an alternative method of capturing panoramic images than those described above. The Ladybug is composed of six colour CCD image sensors: five equally spaced along a horizon ring and one pointing vertically upwards. This camera configuration allows the system to image 75 percent of the full sphere. The manufacturers of the Ladybug have provided a software development kit which facilitates image acquisition, camera configuration and image stitching. Bradley et al. [2005] have demonstrated the Ladybug's effectiveness in real-time acquisition of panoramic images.

The Ladybug unfortunately suffers from some drawbacks. The most serious of these is the high price of the system: currently 11,950 USD. This amount exceeds our project budget. Also worrying is that the system's software development kit was compiled for Windows XP; we intend to use the Linux operating system in our work as we have much more programming experience with this OS. We are concerned as well with the likelihood that one or more of the Ladybug's six cameras will fail. We would probably not have the expertise to replace a broken camera so the system would have to be sent back to the manufacturer or to our in-house technicians for repair, necessitating a delay in our research. The mirror-based panoramic imagers described above, on the other hand, often use a Webcam which, if broken, can be easily and cheaply replaced.

2.3.2 Feature-Based Homing

In order to operate successfully, feature-based homing algorithms must extract the same features from I_S and I_C (the feature-extraction problem). Each feature from I_S must then be paired with a feature from I_C (the correspondence problem). The feature-extraction and correspondence problems are difficult to solve in unadulterated, cluttered environments in real-time. Landmark appearance changes with viewpoint.

In the following sections, we describe a number of feature-based homing algorithms. In each case, we discuss not only how the homing vector is generated but also how the feature extraction and correspondence problems are solved.

Some published work on homing is focused on parsimonious feature extraction and correspondence, rather than the calculation of the home vector. We review this literature in Section 2.3.3. Our main interest in this dissertation is on image-based homing algorithms which of course do not use correspondence algorithms. We review correspondence algorithms because many of them provide metrics to rate the similarity between potential feature pairs. These similarity measures could be used by image-based methods to measure whole-image discrepancy. So too, successful whole-image similarity measures could be used in feature correspondence algorithms.

2.3.2.1 The Snapshot Model

Cartwright and Collett [1983] present an early solution to the visual homing problem. The paper is an ethological study of homing behaviour in honeybees and seeks to answer the following questions: “First, what do bees learn about the spatial layout of landmarks [around a food source]? Secondly, how might this information help them reach their destination?” The authors gathered evidence to support the hypothesis that the bee stores a largely unprocessed panoramic snapshot at the goal position and compares this stored image to its current view to compute a homing vector. This has come to be known as the Snapshot Model and has inspired many of the feature-based visual homing techniques described below.

Honeybees are thought to align snapshot and current views in some external directional reference (e.g. the Earth’s magnetic field), after which features in the snapshot view and current view are extracted and matched. Feature extraction in Cartwright and Collett [1983] was simple, as both simulation and experimental setups consisted of black cylindrical landmarks against a white featureless background. Once extracted, each landmark in the current view is paired with the landmark closest in bearing in the snapshot image.

Two vectors, one radial and the other tangential, are associated with each feature pair (see Figure 2.4). The radial vector is parallel to the bearing of the snapshot feature; the tangential vector is perpendicular to the radial vector. In the original formulation of the Snapshot Model, radial and tangential vectors were of unit length, although workers who adapted this model for robotics relaxed the unit-vector restriction, with interesting results (see Section 2.3.2.3 below). The direction of the radial vector is chosen to move the agent so as to reduce the discrepancy in apparent size between paired features. The direction of the tangential vector is chosen to move the agent so as to reduce the discrepancy in bearing between paired features. The radial and tangential vectors for all feature pairs are averaged to produce a homing vector.

According to Cartwright and Collett [1983] the Snapshot Model (when run in a simulator) mimicked bee behaviour in environments with one and three landmarks. The model also performs comparably when landmarks are changed (in size, distance, and/or orientation) between training and testing and in “complex” environments of

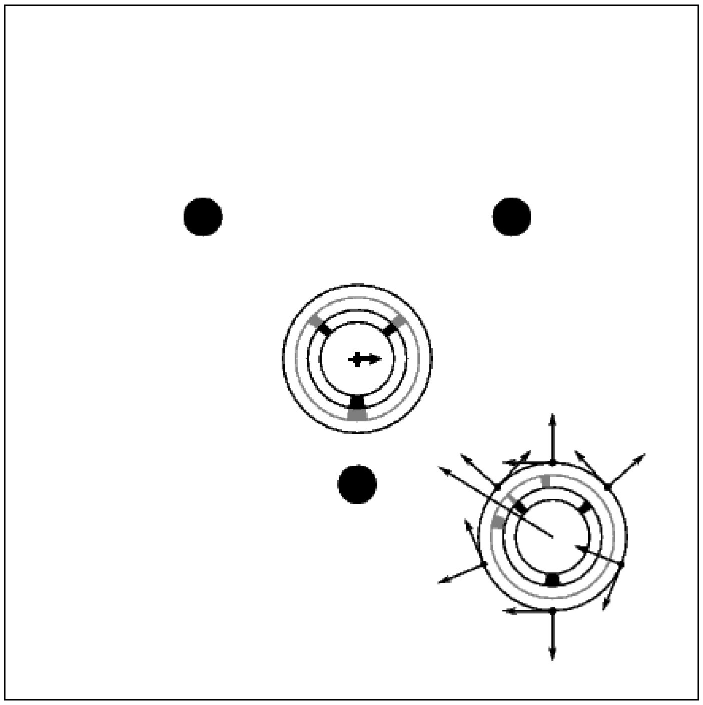


Figure 2.4: Illustration of Cartwright and Collett's Snapshot Model (from Lambrinos et al. [2000]). The three dark circles are landmarks. The agent's home position is at the $+$. The agent's current position is in the lower-right corner of the image. The agent's inner-ring represents the view as captured at the home position; the outer ring represents the current view. Radial and tangential vectors are attached to each inner-ring sector, as described in the text.

eight or nine landmarks. Distant landmarks, though, cause a problem as they offer little useful information about apparent size or bearing but increase the chance of mismatch between snapshot and current image. The authors suggest that bees might filter distant landmarks, a finding supported by Cheng et al. [1987].

Franz et al. [1998b] demonstrate that the Snapshot Model and related algorithms assume that landmarks are isotropically distributed around the snapshot position. In other words, "frequency and distance of landmarks are assumed to be independent of the viewing direction." (p. 3) If this assumption holds true and feature correspondences are computed correctly, Franz et al. [1998b] prove that the Snapshot Model is guaranteed to guide the robot directly home. If the environment is non-isotropic and correspondences are computed correctly, the Snapshot Model will still guide the robot home but on a path that spirals into the snapshot location.

The recent work reported in Bekris et al. [2004] has potentially interesting implications on the methodology of ethological visual homing experiments (though the authors are primarily interested in robotic homing). They explore homing in a simulated environment with three landmarks: L_1, L_2 and L_3. They assume the landmarks are consistently extracted from snapshot and current images and that landmark correspondences can always be found. Their homing algorithm calculates the disparity between I_S and I_C given the change in angular separation between L_1 and L_2; L_1 and L_3; and L_2 and L_3.

To this point, Bekris et al. [2004] is fairly uninteresting: the algorithm therein is quite similar to the Snapshot Model and the chosen experimental environment is unrealistic, having so few landmarks. The authors then, though, derive a method to predict, for a given goal position and landmark configuration, the set of start positions from which the goal is reachable (i.e. the catchment area). Though the authors do not discuss this, their catchment predictor could be used by insect ethologists to determine if their homing algorithm is biologically plausible. It is probable that similar catchment predictors could be derived for alternate homing schemes as well.

2.3.2.2 Hong's Homing Algorithm

Though Hong et al. [1991] makes no reference to the Snapshot Model, Hong's home-vector computation method is quite similar to that of Cartwright and Collett. In Hong's method, as in the Snapshot Model, tangential vectors are associated with each feature pair. In the Snapshot Model, though, these tangential vectors are perpendicular to the bearing of the snapshot feature; in Hong's method, there are two tangential vectors per feature pair, one perpendicular to the bearing of the snapshot feature and the other perpendicular to the bearing of the current feature. Hong does not employ radial vectors (as his features are merely points with no angular size).

Hong's homing method was tested on a mobile robot in an unmodified laboratory environment. Only a few trials were performed, most of which were successful. The robot maintained a constant orientation during its journey so no compass information was required to rotationally align snapshot and current images.

2.3.2.3 The Proportional Vector Model

Introduced in Lambrinos et al. [2000], the Proportional Vector Model (PVM) is a modification of the Snapshot Model. The unit tangential and radial vectors associated with each feature pair in the Snapshot Model are replaced by non-unit vectors in the same directions. The length of the radial vectors are proportional to the difference in apparent size between paired features. The length of the tangential vectors are proportional to the difference in bearing between paired features. These changes cause the length of the homing vector to be proportional to the distance of the agent from the goal.

Lambrinos et al. [2000] use the length of the homing vector to control the robot's speed. The robot moves relatively quickly when far from the goal and slows near the goal, reducing the chance of overshoot. The robot can also use the length of the homing vector (called "disparity" in Lambrinos et al. [2000]) as a stopping condition.

Lambrinos et al. [2000] reports that the PVM was tested with a mobile robot in the salt pan flats of Tunisia. The mobile robot uses a polarised light compass based on that of desert ant *Cataglyphis*. Lambrinos et. al. wanted to test the system in the ant's natural environment, hence the exotic locale. Artificial, high-contrast landmarks (large, black cylinders) were used to define the goal position. Eight homing runs were performed in this environment, beginning at different distances (2 or 4 meters) and directions from the goal. Each homing run brought the agent to within a few centimetres of the goal position.

2.3.2.4 The Average Landmark Vector Model

The Average Landmark Vector (ALV) Model was introduced in Lambrinos et al. [2000]. As in the Snapshot Model, features are extracted from I_S. The average of all feature bearings is calculated and stored as a unit vector. The agent stores this average landmark vector ($\vec{ALV}_S$) rather than I_S. When homing from C, I_C is captured and ($\vec{ALV}_C$)

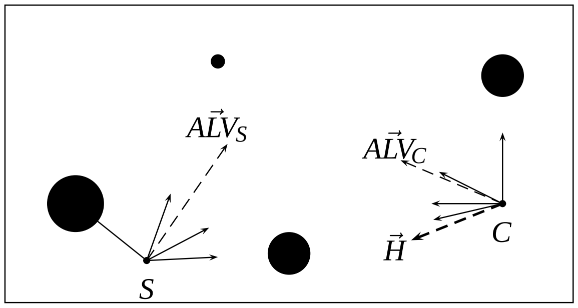


Figure 2.5: Illustration of Average Landmark Vector computation. See Section 2.3.2.4 for details.

is calculated in the same manner as ($\vec{ALV}_S$). If the agent's orientation is different in positions C and S, the algorithm counterrotates $\vec{ALV}_C$ accordingly (with the aid of a compass). The agent then moves in the direction $\vec{H} = \vec{ALV}_C - \vec{ALV}_S$ by some fixed distance. Figure 2.5 depicts the various vectors involved in the computation of $\vec{H}$ for a simple two-dimensional environment with four circular landmarks (black circles).

The ALV Model offers a number of advantages over the Snapshot Model. The agent stores the vector $\vec{ALV}_S$, rather than the entire snapshot image I_S. It is more computationally efficient to rotate the vector $\vec{ALV}_C$ than the image I_C. More importantly, there is no need to solve the correspondence problem in the computation of the ALV. Note, though, that the same features in I_S and I_C must still be extracted in order for the best average landmark vector to be produced. The authors do not analyse the effects on homing success of inconsistent feature extraction.

Lambrinos et al. [2000] alter their experimental environment so that feature extraction is trivial. Lambrinos and co-workers placed four large black cylinders around the goal location to serve as landmarks; these cylinders were in high contrast to the bright featureless Saharan desert landscape in which the experiments were carried out. I_C and I_S were panoramic intensity images; landmarks were detected with a simple thresholding operation using a preset intensity cut-off.

The ALV Model was also tested in a cluttered, unadulterated indoor office environment (Möller et al. [2001]). Here, feature extraction is necessarily a more complex process. As before, a panoramic intensity image is captured at the current location. The one-dimensional horizon image is extracted as described in Section 2.3.1. Sharp changes in the horizon image (which correspond to image edges) are used as features. The resulting ALV algorithm is able to home from some start positions in the office environment. No comparison is made with other homing algorithms, nor does the paper explore environmental change (e.g. lighting change or landmark movement) between captures of I_S and I_C.

Interestingly, Möller [2000] demonstrates that the homing vectors $\vec{H}$ produced by the ALV algorithm are the gradients of a surface which has a global maximum at the goal location. Thus, homing with the ALV algorithm is akin to optimisation with a gradient ascent algorithm. There is thus an unexpected link between ALV, a feature-based algorithm, and the explicit gradient descent homing algorithm described in Section 2.3.4.2.

2.3.2.5 Surfing on the Epipoles

Basri et al. [1999] describe a precise, compassless geometric solution to the visual homing problem. The authors call it “surfing on the epipoles.” The term “epipole” comes from epipolar geometry, which relates camera motion parameters in a stereo view to the coordinates of imaged points in each view; see Figure 2.6 for more information. Unlike most robotic homing researchers, Basri et al. [1999] use a perspective rather than a panoramic camera.

The primary result of epipolar geometry is that p_1 and p_2 (which are defined in Figure 2.6) are related by the equation $p_1^T E p_2 = 0$ where E – the so-called essential matrix – is a $3x3$ matrix whose elements are related to the translation and rotation undertaken by the camera. Basri et al. [1999] use an algorithm crafted by Longuet-Higgins [1981] to solve the essential matrix for a particular rotation and translation; this algorithm requires eight or more points of correspondence in current and snapshot images and is particularly easy to implement (Hartley [1997]). Basri et al. [1999] mention other algorithms for recovering the essential matrix which require fewer correspondences. The translation vector $\vec{T}$ (which is along the line from COP_2 to e_2, hence the name of the algorithm) is easily determined given E. The translation vector is actually known only up to a multiplicative constant. The constant is determined by moving the camera a known direction and distance and taking a third image. The movement of feature points in the intervening move is used to solve for the constant. The rotation parameters can be found from E and $\vec{T}$.

Epipole surfing offers a number of advantages over the feature-based visual homing algorithms described above. With the capture of just two “current” images, the surfing algorithm gives (assuming the correspondence problem is solved) the true home vector. The algorithms above output an estimate to the home vector and require several iterations to reach home. Surfing does not require a compass to align images to a single

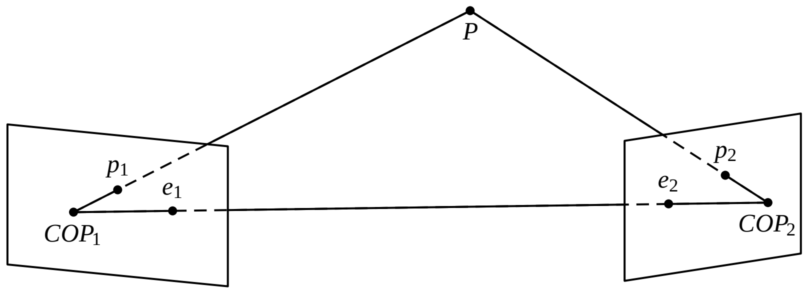


Figure 2.6: A perspective camera with centre of projection COP_1 initially images a point in space P. The image of P on the image plane is p_1 (a two-dimensional image point). The camera then translates and rotates so the image of P on the image plane is at p_2; the camera's centre of projection is now at COP_2. The points COP_1, COP_2 and P define the epipolar plane. Points e_1 and e_2 are called epipoles. Epipolar geometry provides the translation and rotation undertaken by the camera causing the image of P to move from p_1 to p_2. See text for details.

reference direction.

Svoboda and Pajdla [2002] characterise the epipolar geometry for catadioptric cameras. Svoboda et al. [1998] show how to exploit the epipolar geometry of catadioptric cameras to solve the homing problem. Their method requires at least 8 corresponding points between snapshot and current images to determine the essential matrix. Svoboda and Pajdla [2002] then show how to compute translational and rotational elements of the pose change from current to snapshot poses given the essential matrix.

2.3.3 Feature Extraction and Correspondence Algorithms

Some papers are primarily concerned with solving the feature extraction and/or correspondence problem, using the resulting landmark pairs with the Snapshot Model or some other tried-and-true feature-based visual homing algorithm. We review these efforts below.

2.3.3.1 Weber et. al.

Weber et al. [1998] are interested in solutions to the correspondence problem in environments like the one depicted in Figure 2.4 (i.e. black cylindrical landmarks in an otherwise featureless environment). Landmarks are visually indistinguishable so no attempt at object recognition is possible. Weber et al. store the bearings of landmark centres as viewed at the snapshot location. It is not clear whether the authors use an external compass to correct for orientation difference between I_S and I_C but we assume that they do.

Weber et al. [1998] describes attempts to calculate a one-to-one matching between snapshot and current features. The authors assume that the number of features in snapshot and current views is equal (by no means a valid assumption in all situations). They test seven different correspondence algorithms $h1$ - $h7$ in simulation.

- $h1$ is an exhaustive search for landmark correspondence; that is, all possible sets of pairings are considered and the pairing set with the smallest squared error in bearing difference is chosen. As there are $n!$ sets of pairings (where n is the number of landmarks in snapshot or current view), $h1$ requires $O(n!)$ time. Realising that $h1$ is infeasible for large numbers of landmarks, the authors test non-exhaustive solutions $h2$-$h7$.

- $h2$ matches each snapshot feature with the closest (in bearing) unpaired current feature; this greedy algorithm requires $O(n^2)$ steps.

- $h3$ assumes that landmark order is invariant in snapshot and current views; that is, if snapshot feature i is matched with current feature j, then snapshot feature $i+1$ must be matched with current feature $j+1$. This assumption reduces the number of possible sets of pairings to n^2 so the algorithm is $O(n^2)$.

- $h4$ initially corresponds the i^{th} feature in the snapshot with the i^{th} feature in the current view. Adjacent pairings can be swapped bubble-sort style if this leads to a decrease in mean-squared error. This method too is $O(n^2)$.

- Algorithm $h5$ is a hybrid of $h2$ and $h4$. The output produced by $h2$ is if possible improved by $h4$. As $h2$ and $h4$ are $O(n^2)$, $h5$ is $O(n^2)$ as well.

- In method $h6$, each landmark in the snapshot maintains a list of landmarks in the current view with which it would prefer to be paired, and vice-versa. (The method used to establish these preferences is not described.) A landmark in the snapshot view is paired with the landmark closest in bearing in the current view which it prefers and which prefers it.

- Method $h7$ operates like $h2$ except that a many-to-one pairing between snapshot and current features is allowed. That is, one feature in the current view may be paired with many features in the snapshot view. This landmark correspondence method is identical to that used in Cartwright and Collett [1983].

Algorithms $h1$ - $h7$ are first compared in a simulated world in which all landmarks are always visible, regardless of viewing distance. They find that homing failure (defined as the inability of the agent to reach within a certain distance from home) is lowest when using $h1$ or $h3$; homing using the other correspondence methods yields higher failure rates. The relative success of $h3$ remains steady with increasing numbers of landmarks. Home vectors are computed using a variant of the Snapshot Model.

The authors next consider a slightly more realistic simulation, in which the simulated agent has a "perceptual horizon" beyond which it cannot sense landmarks. In this type of environment, there is no clear winner among methods $h1$ - $h7$ and no useful conclusions are drawn.

Usefully, the authors demonstrate that their algorithms have an inherent tendency to avoid collisions with landmarks.

The take-home message of the work of Weber et al. [1998] is that solving the correspondence problem for a non-trivial number of landmarks is a computationally expensive task.

2.3.3.2 Gourichon et. al.

Unlike previous approaches, Gourichon et al. [2002] define locations with one-dimensional *coloured* panoramic snapshots. Colours are defined in the HSV (Hue, Saturation and Value) system.[1] The HSV colour space is useful in landmark identification in that object hues are somewhat independent of current lighting conditions.

Images are segmented into regions of approximately equal colour (landmarks). Potential region pairs are scored on their difference in average hue, average saturation, average luminosity and azimuth centre. A dynamic programming algorithm [2] is employed to find the set of matched features which maximises the sum of individual match scores. Each snapshot region can be paired with only one region in the current view. It is not clear whether every snapshot region must be paired.

Like algorithms $h2$ - $h6$ above, the dynamic programming solution is $O(n^2)$ in time (where n is the number of landmarks in the snapshot view). Like $h1$, the dynamic programming approach is global in that every region in the snapshot view is

[1]HSV is a colour space in which a particular colour is represented by three measures: Hue, Saturation and Value. Hue is the dominant wavelength of the light perceived. Saturation is the ratio of the dominant hue to all other perceived wavelengths; if this ratio is close to zero then grey is perceived. Value, also known as luminance, is light energy emitted per unit time per solid angle in a given direction.

[2]Dynamic programming is a recursive technique in which the solutions to subproblems are computed and saved. These subproblem solutions are combined to solve larger problems. The Fibonacci sequence for example can be efficiently computed recursively in this manner.

compared with every region in the current view; local approaches compare a restricted set of regions (e.g. those that have similar bearings) and are thus more susceptible to mismatches. The authors claim that the dynamic programming approach yields larger catchment areas than any of the algorithms $h2$ - $h6$, though they don't provide data to support this. The dynamic programming approach assumes that landmark order is invariant between current and snapshot views; this assumption will be true only near the goal position.

The authors show that their correspondence algorithm is fooled in an office environment by different objects with similar hues. Despite the fact that the authors use the HSV colour space, they do not test their algorithm in conditions of dynamic illumination.

2.3.3.3 Bianco et. al.

Lehrer and Bianco [2000] describe a novel addition to feature matching research which employs active vision near the goal. The approach mimics bee behaviour in looking at landmarks near the goal from a range of positions around the goal before departure. The algorithm rates landmarks on their visual reliability, averaged over these different views. Only highly reliable landmarks are memorised and used later in the homing process. Catchment areas are shown to increase when using this scheme to home in an office environment. Lehrer and Bianco [2000] employ colour, two-dimensional images.

The approach does not fully mimic bee behaviour, however. Actual bees (and wasps) learn about landmarks as they approach a goal as well as when they leave it. So too, bees learn about landmarks over the course of multiple arrivals at and departures from the goal position (Zeil et al. [1996]). Nonetheless, we consider this an interesting and useful addition to the literature. We note though that the algorithm of Lehrer and Bianco [2000] requires copious extra effort on the part of the homing robot in order to identify visually reliable landmarks.

2.3.3.4 Gaussier et. al.

Gaussier et al. [2000] outline a feature extraction technique in which "the visual system focuses on corners and/or end of lines" in an almost complete panoramic image. The surrounding area of these distinctive points is also captured: "[f]or each selected focus point, a 32 x 32 pixels [sic] local view is built by averaging the 148 x 288 pixels of the corresponding panoramic image part.” The averaging process is not described in detail.

Once feature points and their neighbourhoods have been extracted from current and snapshot views, the correspondence problem is solved as follows: a root mean square pixel-by-pixel difference is computed between each local view in the current image and every local view extracted from the snapshot image. “The best corresponding views” are used in the computation of home vectors. The authors do not define “best” but one can imagine a thresholding process is employed.

2.3.3.5 Rizzi et. al.

Rizzi et al. [2001] describe a solution to the feature extraction and correspondence problem using a Fourier transform to compute frequency information for each feature; “Visual Reference” (VR) is used instead of “feature” or “landmark,” but the terms are synonymous.

VRs are extracted from an image using a computationally intensive region growing technique. Marshall [1997] explains that region growing starts by choosing a seed pixel, compares it to neighbouring pixels, adds these neighbours to the region if they are similar (by some measure) to the seed and repeats the process on these similar neighbours. The region stops growing when no more pixels adjacent to the region are similar enough to the seed pixel to be added to the region; at this point, a seed for a new region is chosen and the region is grown as described above. The algorithm continues to choose seeds until all pixels belong to some region. Unfortunately, Rizzi et al. [2001] fails to define a similarity measure; nor does the paper describe how seed pixels are chosen.

Not all regions are considered viable VRs; only those VRs with desirable “area, perimeter regularity and chromatic saturation” are selected. The paper is vague on how perimeter regularity is measured and on what values of area, regularity and saturation indicate usable VRs. They do not reveal if their VR selection technique is tailored to their experimental environment.

The feature matching process is as follows: A Fourier-Mellon transform is computed for each VR in an image. The transform gives translation- and scale-invariant information about a particular VR. VRs in current and snapshot views are matched by comparing their Fourier-Mellon transforms.

As we noted above, solutions to the correspondence problem which measure the similarity of extracted features can be applied to the whole-image comparison required by image-based homing. We shall see for example in Section 2.3.4.1 that the Fourier Transform for image comparison introduced by Rizzi et al. [2001] is used to hasten an image-based homing technique.

2.3.3.6 SIFT Features

Pons et al. [2007] use the scale-invariant feature transform (SIFT) algorithm to extract features from snapshot and current images for the purposes of feature-based homing. SIFT features were introduced by Lowe [2004]. SIFT features are demonstrably invariant to translation, scaling and rotation in images. They are also highly distinguishable from one another and somewhat invariant to illumination changes. These properties make them suitable for the purposes of homing, where a particular image feature changes appearance due to change in imager pose between current and snapshot locations.

The first step in the SIFT algorithm is to find candidate feature locations – keypoints – in an input image. Keypoints are defined as scale-space local maxima and minima. To find keypoints, the input image is convolved with Gaussian functions with a range of scales (i.e. standard deviations). The scale relates to the level of blurring of the image. The difference between the Gaussian-blurred images for each pair of successive scales is computed. Keypoints are then identified at the local optima of neighbouring difference images. Each keypoint is associated with a particular image location and scale. Next, the local intensity gradient orientation at each keypoint location is computed. The intensity gradient orientation of neighboring pixels around a keypoint are computed (relative to the keypoint's orientation) as well. Knowlege of keypoint orientation provides rotational invariance to the associated feature. These local intensity gradient orientations make up a scale-, location- and rotationally-invariant set of 128 features for each keypoint.

To find feature correspondences between snapshot and current images, Pons et al. [2007] use the matching scheme outlined in Lowe [2004]. Lowe [2004] argues that a naive approach would be to find the snapshot feature which is closest in Euclidean distance (in feature space) to the candidate feature in the current image. The current candidate feature, though, may have no legitimate match in the set of snapshot features. Lowe [2004] found that using a simple distance threshold does not effectively weed out these false matches. Instead, Lowe [2004] suggests computing the ratio of the closest distance to the second closest distance. A relatively small ratio value indicates that the matched snapshot feature is highly distinct.

Pons et al. [2007] use the homing algorithm described in Vardy and Möller [2005] for their experiments. This homing schemes assumes that snapshot and current images were captured in the same orientation. This assumption does not generally hold in the experiments of Pons et al. [2007]. To align the orientation of the current image with that of the snapshot image, Pons et al. [2007] use a type of visual compass. We describe the notion of the visual compass in detail in Section 2.3.5. Pons et al. [2007] demonstrate that their method is successful in both indoor and outdoor static and dynamic environments. We note that to account for changes in illumination and object position at the snapshot location, Pons et al. [2007] collected several images at the snapshot location at different times throughout the day and extracted SIFT features from each representative image. The aggregate set of SIFT features is used for homing.

SIFT features have also been used in vision-based simultaneous localisation and mapping systems (e.g. Se et al. [2001]).

Several authors have made attempts to speed-up SIFT feature computation and correspondence. Bay et al. [2006] suggest speeded-up robust features (SURF) as an alternative to SIFT features. As the name suggests, these authors claim that computation and comparison of SURFs is faster than that of SIFT features. The SURF algorithm achieves quicker computation by using clever appoximations in some of the SIFT steps. Ledwich and Williams [2004] remove the rotational information from the SIFT feature vector noting that this information is not needed for camera-equipped robots constrained to move on a plane. Ke and Sukthankar [2004] reduced the dimensionality of the SIFT feature vector by applying PCA to the intensity gradient information around each keypoint.

2.3.4 Image-Based Homing

Since efficient and consistent feature extraction and correspondence is difficult in cluttered, unadulterated environments, researchers have developed homing methods which calculate homing vectors with whole images, rather than landmark sets extracted from images. These methods are described below. We note that image-based methods are sometimes known as appearance-based in the literature.

2.3.4.1 Image Warping

Image warping (Franz et al. [1998b]) is aptly named, as will soon become clear. For each calculation of $\vec{H}$, the set of all pose (position and orientation) changes between S

and C are considered. Given a candidate pose change, I_S is transformed to the image that the agent would capture at the location defined by the pose change assuming that all imaged landmarks are an equal distance from S. Each warped I_S is compared with I_C with a pixel-by-pixel root-mean-square measure. The pose change associated with the warped I_S most similar to I_C is used to determine the current home vector. If after following the home vector for a pre-determined distance the agent is not home, the process is repeated. Since image warping is a computationally costly process repeated many times for each home vector computation, Franz et al. operate on horizon images; this implies that image warping is only suitable when the homing agent is travelling on a plane (i.e. indoors). Note that because the algorithm searches over orientation changes between S and C as well as position changes, image warping does not require an external compass reference to align I_S and I_C.

Some visual homing researchers (e.g. Möller [2002] and Vardy and Oppacher [2004]) consider image warping to be the most reliable visual homing method for indoor use. In comparative tests, image warping often produces more reliable homing directions and larger catchment areas than any other visual homing method reviewed here.

Image warping's most obvious drawback is its brute force nature. That is, the algorithm arrives at an optimal solution by checking every possible pose change rather than using a more efficient process like gradient descent. Unfortunately, the parameter space searched during image warping has many local minima on which gradient descent could halt prematurely (Franz et al. [1998b]).

Image warping is also biologically implausible (Möller [2002]). It is unlikely that a small-brained insect could carry out the large number of sequential operations required by the algorithm in real-time. A more easily parallelizable algorithm is more plausible.

Some advances in image warping have recently been published. Sturzl and Mallot [2006] – like some systems in image-based localisation reviewed earlier – compress current and snapshot images using the discrete Fourier transform. They then show how to perform image warping as described above with these reduced images. Möller [2008] shows how to perform image warping on two-dimensional (rather than horizon line) images relatively efficiently.

2.3.4.2 Homing by Optimising on the Difference Surface

As we outlined in Chapter 1, Zeil et al. [2003] found that the difference between two panoramic intensity images increases monotonically with the physical distance between their capture positions. We described in the following paragraphs how this phenomenon can be exploited for visual homing.

Zeil et al. [2003] describe an algorithm which they call “Run-Down” to move the homing agent so as to optimise the difference surface. The agent moves forward, periodically sampling the difference surface. This forward movement continues while difference surface samples consistently decrease (implying that the agent is nearing the snapshot location). When the current sample is greater than the previous, the agent turns ninety degrees counter-clockwise. Upon completing the turn, the agent again moves forward and samples the difference surface periodically. The agent turns again when another increase in samples is detected. The agent continues in this manner until the current difference surface value falls below a preset threshold, at which time the agent stops, believing itself to be home. This algorithm causes the agent to “spiral in” to the snapshot location on a sometimes quite tortuous path.

Another way to optimise on the difference surface is to compute the gradient (the direction of greatest increase) of the surface at C and move in the opposite direction (the direction of greatest decrease) for a certain distance. If the robot is not then at S, the gradient at the new location is computed. Zeil et al. [2003] call this homing method “Triangular.” To compute the gradient at C, an image I_C is captured as usual at C; images I_D and I_E are captured at two nearby orthogonal locations D and E. The gradient is approximated by the vector $(\frac{RMS(I_S,I_D)-RMS(I_S,I_C)}{dist(D,C)}, \frac{RMS(I_S,I_E)-RMS(I_S,IC)}{dist(E,C)})$ where $dist(D,C)$ is the Euclidean distance between C and D (measured by dead reckoning); $dist(E,C)$ is the distance between C and E.

Möller and Vardy [2006] present a significant improvement to gradient-based optimisation on the difference surface. The authors note that in estimating the gradient of the difference surface, Zeil’s “Triangular” algorithm directs the homing agent to capture images at three proximate locations: C and two other positions nearby. They argue that I_C can be warped to approximate the images that would have been captured at the two locations near C. The assumption of equal landmark distance first made in Franz et al. [1998b] is employed to warp I_C. A similar algorithm was reported in Binding and Labrosse [2006].

Möller and Vardy [2006] demonstrate empirically that the gradients generated from warped images I_C are often at least as accurate as gradients computed with “Triangular.” The authors note, though, that the distance between C and the two nearby points at which images are captured for gradient computation with “Triangular” is probably too great in their experiments. Smaller step sizes, they argue, would have produced better gradients. The gradients from warped images simulate image sampling at locations an infinitesimal distance from C.

Möller et al. [2007] report an improvement on the homing algorithm presented in Möller and Vardy [2006]. They note that in environments with anisotropic landmark distributions, the difference surface is distorted, losing the radial symmetry that it has in environments with isotropic landmark distributions. Möller et al. [2007] show that the gradient of these distorted difference surfaces often fails to point directly to the snapshot location. This divergence between the true home direction and the gradient direction causes a homing robot to take a curved path to the snapshot location. The authors address this problem by using Newton's method to move the robot so as to optimise the difference surface. The use of Newton's method requires knowledge of the Hessian matrix of the difference surface. The Hessian at a point on a function is the square matrix containing all of the second-order partial derivatives of the function at that point. Since the difference surface is – at least in Möller et al. [2007] – a function of two independent variables (i.e. the robot's position on a plane), the Hessian in this case contains four elements: the second-order partial spatial derivatives of the difference surface. The version of Newton's method derived by Möller et al. [2007] requires only the Hessian of the difference surface at the goal location. The Hessian elements thus are only computed once, immediately after the robot captures the snapshot image. Möller et al. [2007] showed in tests in several real-world indoor anisotropic environments that the home vectors produced by Newton's method are more accurate than those produced with the method described in Möller and Vardy [2006]. In isotropic environments, the two methods produced similar home vectors.

In very recently published work, Sturzl and Zeil [2007] demonstrated that the shape of the difference surface (called an image difference function in Sturzl and Zeil [2007]) is dependent on the structure of the imaged environment. In particular, the difference surface becomes wider as the mean distance between the imager and imaged objects increases. The catchment area of a difference surface-based homing algorithm presumably increases as well, though Sturzl and Zeil [2007] carried out no homing runs to confirm this. Sturzl and Zeil [2007] also investigated image preprocessing steps to make difference surfaces robust to illumination changes. They report that (p. 519) "image processing operations – like subtracting the local mean, difference-of-Gaussian filtering and local contrast normalization – make difference functions robust against changes in illumination and the spurious effects of shadows." We stress that the work presented in Sturzl and Zeil [2007] was published after we completed the work described in this dissertation and did not have a bearing on our research.

Baccou and Jouvencel [2002] report a homing algorithm which uses non-visual input. We discuss the algorithm here because the problem solved is in essence quite similar to that described in Zeil et al. [2003].

The problem is this: an autonomous underwater vehicle (AUV) is to navigate to a single radio beacon. The AUV infers its distance from the beacon given the time-of-flight of the signal emitted by the beacon. In Zeil's homing algorithm, the disparity between I_S and I_C is a function of the Euclidean distance between S and C; here, the time-of-flight of the radio signal is a function of the distance between the AUV and the beacon. While Baccou and Jouvencel [2002] have a function relating AUV-beacon distance and time-of-flight, there is no knowledge of such a function in Zeil's scheme.

The authors use an extended Kalman filter to estimate the vehicle's location with respect to the beacon. See Section 2.2.1 for more information on the use of the EKF for localisation. The filter requires a good initial estimate of the vehicle's position in a coordinate system with the beacon at the origin. The authors generate this estimate by driving the vehicle in a circle around its starting point, capturing a number of beacon-range measurements along the way. A non-linear least squares optimisation algorithm is used to find the initial vehicle position which is most consistent with the series of range estimates.

This initial estimate is fed to the extended Kalman filter which, if all goes well, will improve the estimation of the moving vehicle's position given odometric readings paired with ongoing beacon-range measurements. The authors found that the error in the position estimate decreases most rapidly when the vehicle is made to travel in a circle centred at the beacon. This movement scheme is obviously at odds with the goal of navigating to the beacon.

The authors do not consider the optimisation methods described in earlier in this section.

2.3.4.3 Visual Homing with Optic Flow Techniques

Vardy and Möller [2005] apply optic flow techniques to solve the homing problem. Optic flow is the perceived movement of objects caused by viewer rotation and/or translation. In the context of visual homing, moving from S to C causes a particular imaged point $I_S(x,y)$ to move to $I_C(x',y')$ (assuming no inter-object occlusion has occurred). The optic flow displacement vector associated with this imaged point is $(x'-x, y'-y)$.

Using straightforward trigonometry, Vardy and Möller [2005] demonstrated that the home vector can be calculated given the optic flow displacement vector associated with one imaged point. The solution requires that the image is two-dimensional; both the vertical and horizontal components of the displacement vector are required in the calculation. In addition, the robot is constrained to travel in a single plane. If the homing robot is in a different orientation when capturing I_S than when capturing I_C, the latter image must be rotated to account for this difference before optic flow calculations can be carried out. Since individual displacement vectors are typically noisy, the authors generate displacement vectors across the entirety of I_S and average the resulting home vectors. Vardy and Möller [2005] demonstrated that if correspondence errors are uncorrelated then the home vectors which result from these faulty correspondences will tend to cancel each other out when summed together. Thus, only home vectors resulting from correct correspondences will tend to affect the final, average home vector.

But how are the displacement vectors computed? Vardy and Möller [2005] adapted several methods from the optic flow literature, testing each in turn. The BlockMatch methods segments I_S into a number of subimages. The best match for each subimage is found in I_C using a brute-force search in the region surrounding the subimage. Another displacement vector computation method – FirstOrder – estimated the displacement from the intensity gradient at each pixel in I_C. This has the obvious advantage over BlockMatch that no search is required to compute the displacement vector.

In tests in a few indoor environments, Vardy and Möller [2005] showed that their optic flow techniques often surpassed image warping in homing success. They used two criteria for comparison: average angular error and return ratio. Angular error is the difference – at a particular location C and for a particular snapshot location S – between the computed home vector and actual home vector. This measure is averaged over a large number of pairings of S and C. The return ratio for a particular snapshot location S is the percentage of successful homing runs to S from a grid of surrounding starting locations. Their test were carried out in visually static and dynamic indoor environments. Lighting or landmark locations changed between captures of snapshot and current images in their dynamic environments. Their optic flow algorithms showed resilience to this dynamism.

2.3.5 The Visual Compass

As we detailed above, the robot may be in a different orientation at S than at C. In order to meaningfully compare I_S and I_C, I_C must be rotated to account for this orientation difference. Most of the homing methods described above require an external compass reference to determine the orientation difference. The obvious choice of compass is magnetic. Unfortunately, magnetic compasses are notoriously noisy in indoor environments as the geomagnetic signal is distorted by electrical equipment. Both Zeil et al. [2003] and Labrosse [2004] propose the so-called visual compass as an alternative. These authors found that as I_C is rotated over 360 degrees (either by physically rotating the imager or by shifting the pixels of I_C to simulate such a rotation) the RMS difference between I_S and the rotated I_C often attains the global minimum at or near the orientation at which I_S was captured. This is more likely to be true the nearer C is to S. At locations C relatively far from S, the desired orientation is at or near a local RMS minimum. For these reasons, the visual compass – according to Binding and Labrosse [2006] – must initially be seeded with the agent's current orientation in the compass reference frame used when capturing I_S. The visual compass can then be used to update the agent's orientation as it navigates. The initial agent orientation is provided by a human operator, a significant drawback for autonomous robotics.

2.4 Tasks Related to Visual Homing

2.4.1 Visual Servoing

Hutchinson et al. [1996] define visual servoing as follows: "the task in visual servoing is to use visual information to control the pose of the robot's end-effector relative to a target object or a set of target features. The task can also be defined for mobile robots, where it becomes the control of the vehicle's pose with respect to some landmarks." (p. 651) Hutchinson et al. [1996] and Chaumette and Hutchinson [2006] distinguish between two types of visual servoing: position-based and image-based. The former attempts to explicitly estimate the pose change between current and target poses using image features and some knowledge of the three-dimensional structure of the environment. In image-based visual servoing, the control commands are deduced directly from image features. Visual homing as we defined it above, then, can be seen as a type of image-based visual servoing. Usher et al. [2002] for example generate an alternative to the ALV homing algorithm described in Section 2.3.2.4 using both feature bearing *and* range and call it a servoing algorithm. It should benefit our understanding of the current state-of-the-art in visual homing to review some major results in image-based visual servoing.

Chaumette and Hutchinson [2006] note that most visual servoing schemes essentially attempt to move a camera so as to minimise a time-dependent error function of the form $e(t) = s(t) - s*$. In image-based visual servoing, $s(t)$ is a set of image feature parameters (e.g. the pixel locations of corner points) in a camera's current view and $s*$ is the set of parameters for the same features in the target view. Hutchinson et al. [1996] notes that $s*$ can be acquired via a "teach by showing approach in which the robot is moved to the goal position and the corresponding image is used to compute a vector of desired image feature parameters" (p. 661). This is the approach adopted in visual homing algorithms. The computation of $e(t)$ typically requires a solution to the correspondence problem (the servoing solution of Usher et al. [2002] is a notable exception). We shall return to feature correspondence in visual servoing a bit later.

How can a robot be instructed to move so as to minimise $e(t)$? Chaumette and Hutchinson [2006] argue that most image-based visual servoing schemes adopt the same general approach in the design of a controller. This controller sets the velocity (translational and rotational) of a camera-equipped servoing robot. The controller exploits the fact that the derivative of $e(t)$ with respect to time is a function of the product of the robot velocity and the so-called interaction matrix (also known as the feature Jacobian in the literature). One can solve this equation for velocity to obtain an expression for the desired velocity of the robot as the product of the inverse of the interaction matrix and the derivative of the $e(t)$.

Unfortunately in practice one cannot generally know the exact values in the interaction matrix or by extension its inverse (Chaumette and Hutchinson [2006]). The form of the interaction matrix in general depends on the feature parameters chosen and the distance of imaged objects from the camera (Chaumette and Hutchinson [2007]). For point-like features and perspective camera images, the interaction matrix depends in part on the distance of imaged objects from the camera. This range cannot be determined if a single image from a single camera is used. Thus an approximation to the interaction matrix is often employed. Espiau et al. [1992] for example use the interaction matrix valid at the goal location throughout servoing, treating it as a constant. This approximation only requires the range of feature points in the goal orientation which can be determined in a pre-processing step. Chaumette [2004] devised the form of the interaction matrix for feature parameters consisting of the moments of planar objects. Again, the depth of image features must be approximated for the moments-based matrix. Barreto et al. [2002] derived the interaction matrix for catadioptric imaging systems. Generally, the search for useful feature parameters and their associated interaction matrices is a major topic in the visual servoing literature (Chaumette and Hutchinson [2006]).

We mentioned above that computation of $e(t)$ typically requires a solution to the feature correspondence problem. Marchand and Chaumette [2004] note that "[most] of papers [sic] related to visual servoing consider very basic image processing algorithms." (p. 2) The initial correspondence between features is actually sometimes computed manually (see e.g. Hutchinson et al. [1996] and Cretual and Chaumette [2001]). Marchand and Chaumette [2004] report correspondence estimation by an estimation of the fundamental matrix. Once an initial correspondence is made, many visual servoing solutions track features as they move in images. Feature movement is of course due to parallax induced by the movement of the servoing camera. Marchand and Chaumette [2004] review feature tracking algorithms used in visual servoing. They report for example that simple edge features are tracked by searching in the image in the direction of the edge normal. Marchand and Chaumette [2004] also relate the use of image registration algorithms to find the affine transformation of image blocks. Papanikolopoulos and Smith [1995] describe the use of block-matching using a sum-of-squares similarity measure to track regions in successive images.

Deguchi and Noguchi [1996] present an appearance-based rather than feature-based approach to image-based visual servoing. They reduce the dimensionality of each captured image (including the image taken at the goal position) using principal components analysis (PCA), a technique we saw used in the context of image-based localisation. The camera images used to create the eigenvectors used in PCA reduction are collected in a preprocessing step before servoing occurs. Deguchi and Noguchi [1996] then show how visual servoing is equivalent to tracing a path along the appearance manifold made up by reduced images. They demonstrate that the parameters of the tangent plane to the appearance manifold at any given point on the manifold can be seen as an interaction matrix. Deguchi and Noguchi [1996] finally show how the change from the current pose to the goal pose can be estimated using the current interaction matrix and the difference between dimensionality-reduced current and goal images. This technique is quite similar to that of Yang et al. [2007] described earlier.

2.4.2 Docking

Robotic docking involves controlling a robot so that it achieves a desired pose with respect to a docking station. The docking station will be assumed to be static in this review. Autonomous fork-lifts and robotic geologists come to mind when robot docking is mentioned. Also, a robot arm must dock with an object before grasping it. A docking algorithm is typically employed once the dock is within the perceptual range of the robot's sensors; docking is therefore considered a relatively short-range navigation skill. Docking solutions should ideally provide highly accurate positioning of the robot relative to the dock. As we shall see below, robots equipped with visual sensors can solve the docking problem by visual homing and other visual servoing approaches. Docking in fact provides a useful application for visual homing in indoor environments. Santos-Victor and Sandini [1997] distinguish between ego-docking – where the camera is mounted on the robot – and eco-docking – where the camera is mounted on the docking station. We shall here considered ego-docking solutions only, as these are most similar to visual homing.

Wei et al. [2005] present a docking algorithm based on the average landmark vector model, a feature-based visual homing algorithm described in Section 2.3.2.4. Wei et al. [2005] augment the basic ALV algorithm to allow more precise control over the trajectory of the robot as it approaches the dock. Briefly, this control is exerted by weighting each image feature differently and using these weights in the computation of homing vectors. The solution achieves positioning accuracy on the order of 1cm in real-world tests with artificial landmarks. Though poorly written and frankly difficult to understand, Jantapremjit and Wilson [2007] seem to describe a similar solution for docking autonomous underwater vehicles.

McCarthy and Barnes [2006] describe a method to allow a robot to dock with a planar surface perpendicular to the ground plane to which the robot is constrained (i.e. a wall). The authors are primarily concerned with controlling a robot to stop as close to a wall as possible without actually touching it. In this sense, the robot is not concerned with docking with a particular point on the wall, but merely approaching it safely. McCarthy and Barnes [2006] use the optical flow resulting from moving towards the wall to measure the time-to-contact. The authors demonstrate that the divergence of the flow field can be used to compute the time-of-contact to an object along the optical axis of the camera. This computation rests on the assumption that the wall is perpendicular to the camera's optical axis. As this is rarely the case in practice, McCarthy and Barnes [2006] devise a way to estimate time-of-contact from flow field divergence when this assumption is not met. The docking wall is assumed to be sufficiently textured to provide strong optical flow patterns as the robot moves toward the wall. This assumption may prove to be the Achilles heel of the method, as walls frequently lack such texture. The work of McCarthy and Barnes [2006] is similar to the earlier work of Santos-Victor and Sandini [1997].

2.4.3 Image Registration

Visual homing is quite similar to the problem of image registration. An image registration algorithm attempts to find the function which best transforms one image of an object or scene into a second image of the same scene or object. The two images can differ due to the poses of the imager, the modalities in which the images were captured, and/or the layout of the scene among other things. Image registration algorithms seek a function which transforms pixel locations; this function is often though not always affine. While image registration algorithms search for the pixel-by-pixel transformation between two images, visual homing seeks to estimate the transformation of an *imager* from S to C given images I_S and I_C.

Hill et al. [2001] give a comprehensive review of image registration algorithms used in medical imaging applications. The paper demonstrates that image registration solutions are quite similar to many visual homing algorithms. Early registration work tried to find landmarks in the images to be aligned and used the change in pose of these landmarks to infer the overall image transformation. These algorithms had to select appropriate landmarks and match corresponding landmarks in multiple images, both difficult problems as we discussed earlier in this chapter.

More recent work in image registration has attempted to align entire images, eschewing landmark selection and correspondence issues. Similar to appearance-based homing, these image registration algorithms search for the image transformation which maximises the similarity between one image and a second transformed image of the same scene. A focus of this work has been on the similarity measure used to compare images. The use of mutual information in image registration was first reported by Viola and Wells [1995] and apparently independently discovered by Maes et al. [1997].

Registration techniques are used in metric robot localisation. We saw examples of the use of registration for scan- and map-matching in Section 2.2.1.

2.5 Mapless versus Metric Map-based Visual Homing

After reviewing both metric localisation algorithms and visual homing (plus related publications in servoing and topological navigation), it is natural to ask which approach gives better results in solving the homing problem set out in the introduction to this chapter. Goedemé et al. [2005a] seek to answer this question. In particular, these authors want to determine whether acquiring knowledge of the three-dimensional layout of landmark features (i.e. “structure”) near the snapshot location makes homing easier.

The so-called structureless homing algorithm used by Goedemé et al. [2005a] is a bearing-only variant of the Snapshot Model described in Section 2.3.2.1. Goedemé et al. [2005a] extract two types of features from their snapshot and current images. They use a variant of the SIFT features described in Section 2.3.3.6. Goedemé et al. [2005a] also extract so-called invariant column segments. The robot in Goedemé et al. [2005a] is constrained to move on the floor of an office environment. Many objects in this environment (e.g. window frames and white boards) form edges parallel to the floor plane in images. The top and bottom edges of say a window frame correspond to two sharp nearby intensity gradients in several adjacent columns of an unwrapped panoramic image of the scene. These paired sharp intensity gradients delimit a column segment from which feature data is extracted. Goedemé et al. [2005a] note that column segment features can be identified more quickly than SIFT features. Feature correspondences are established at the beginning of the homing process. After that, while the robot moves along a homing vector, features are efficiently tracked in successive current images. The home vector is continuously updated as the robot moves towards the snapshot position.

The second homing method of Goedemé et al. [2005a] builds a three-dimensional map of local features as the robot homes. The origin of the map is the point at which the robot begins the homing procedure. As before, both SIFT features and invariant column segments are extracted from current and snapshot images. These features are now, though, localised in a three-dimensional map of the local environment. Goedemé et al. [2005a] use a simultaneous localisation and mapping (SLAM) algorithm based on the extended Kalman filter (EKF) to create the local map; see Section 2.2.3 for more information on EKF-SLAM. The estimate of the snapshot location on the local map is updated along with the robot’s current pose, increasing the accuracy of the home vector estimate.

To compare the two homing methods, Goedemé et al. [2005a] place their robot (an autonomous wheelchair) between 1 and 2 metres from a single snapshot position in an indoor environment. They run an unspecified number of homing runs. Half of the runs use the so-called structureless homing method and the other half of course use the map-based method. It appears that the latter algorithm leads the robot in a more-or-less straight path to the snapshot location while the structureless method takes a slightly more meandering path. They both seem to exhibit about the same final precision, though. The map-based method, on the other hand, requires more than 100 times more processing time per home vector computation than the structureless method. On their 800 MHz processor, the map-based homing algorithm took 71 ms per home vector and the mapless algorithm, 0.44 ms. One wishes that the authors carried out more experiments, varying the home distance and the number of features extracted from images. The latter has a great bearing on the time required to perform EKF-SLAM. We also wish that the overall time-to-home had been published, as this would seem an important criteria for comparing the two methods.

We conclude from this study that so-called structureless homing algorithms are probably sufficient for visual homing and preferable when a mobile robot is equipped with a relatively primitive processor. A robot should employ a map-based solution, though, if path length must be minimised for some reason.

2.6 Conclusions

We began this review by looking into two major trends in robotic navigation: metric navigation and topological navigation. Metric localisation seeks to answer the question "Where is the robot?" quantitatively (i.e. the robot is at coordinates (3,4) with heading 90 degrees in a given reference frame). Topological navigation gives a more qualitative answer (i.e. the robot is in the office or is travelling down the hallway towards the library). There are strengths and weaknesses to both approaches, as we have discussed above.

In this review we have seen that visual servoing algorithms frequently form an important part of a robot's algorithmic repertoire in topological approaches to navigation. A servoing algorithm is typically employed to guide a robot between adjacent nodes in a topological map. Visual homing is a type of servoing and sometimes the algorithm used in a topological navigation system is explicitly called a homing algorithm. We also note that measuring image similarity plays an important role in vision-based topological navigation. For the purposes of localisation or place recognition, the similarity between the robot's current view is compared with references images stored in the topological map. For autonomous topological map-making, image similarity measures are employed to determine if the current location is perceptually distinct from previous mapped positions; if so, a new map node is typically instantiated. Topological navigation calls for the image similarity measure to be both computationally efficient and relatively invariant to visual change in the environment (due to e.g. lighting change and object movement). Efficiency is required because a newly collected image must usually be compared with many reference (i.e. map) images in order to recognise a place. Invariance to environmental change is needed because the environment may have changed since the reference images were captured. We shall return to image similarity measures below.

Focusing on the topic of visual homing, we have in this chapter outlined the most important visual homing algorithms found in the literature. These algorithms fall naturally into two categories: feature-based and image-based (also known as appearance-based). Almost all feature-based algorithms require reliable solutions to the problems of consistent feature extraction and correspondence to ensure successful operation. If these problems can be solved consistently, our review indicates that the epipole-surfing algorithm Basri et al. [1999] should be used to home. This algorithm produces an accurate home vector given just two successive "current" images and requires no external compass reference (as several homing algorithms do). Consistent feature extraction, though, is by no means an easy task. It is telling that many experiments in feature-based visual homing take place in adulterated environments with easy-to-detect artificial landmarks. An exception to this was the work of Pons et al. [2007]. These authors extracted SIFT features from snapshot and current images and used a robust matching scheme to establish feature correspondence. SIFT features are relatively invariant to changes in orientation, scale and location in images and so are appropriate for the visual homing problem. Though Pons et al. [2007] offer impressive homing results in indoor and outdoor static and dynamic environments, their algorithm seems to require thousands of feature similarity computations for each home vector calculation. Pons et al. [2007] do not offer figures on how long each home vector computation requires. Computationally efficient SIFT features (e.g. SURFs) have been proposed recently and will probably play a role in future feature-based visual homing algorithms.

As we have demonstrated, image-based visual homing problems avoid explicit feature correspondence. This approach therefore offers a potentially robust and efficient complement to feature-based homing. Image-based homing schemes infer home vectors by considering every pixel in snapshot and current images, treating each as a feature in its own right. We have reviewed three image-based homing algorithms in this chapter. Image warping requires a computationally intensive brute force search for every home vector computation. We consider this a major drawback. The optic flow-based algorithms of Vardy and Möller [2005] – though impressive – assume that the homing agent is constrained to travel on a single plane. We consider the difference surface homing algorithm pioneered by Zeil et al. [2003] to be the image-based homing algorithm which shows the most promise.

Unfortunately, as described by Zeil et al. [2003], this algorithm is sensitive to changes in lighting between capture of snapshot and current images. Also, Zeil et al. [2003] applied only simple optimisation algorithms to this task and assumed perfect compassing information in the process. There are thus open problems in difference surface-based homing which we chose to tackle at the outset of our work on this dissertation. We also seek in this work to improve the computational efficiency of the image similarity measure which lies at the heart of the difference surface homing algorithm. We do this not only to improve the efficiency of difference surface homing, we also believe that this image similarity measure could be applied to the problems of place recognition and mapping in vision-based topological navigation. As we noted above, solutions to these problems benefit from a computationally efficient image similarity measure.

Chapter 3

Building Robust Difference Surfaces

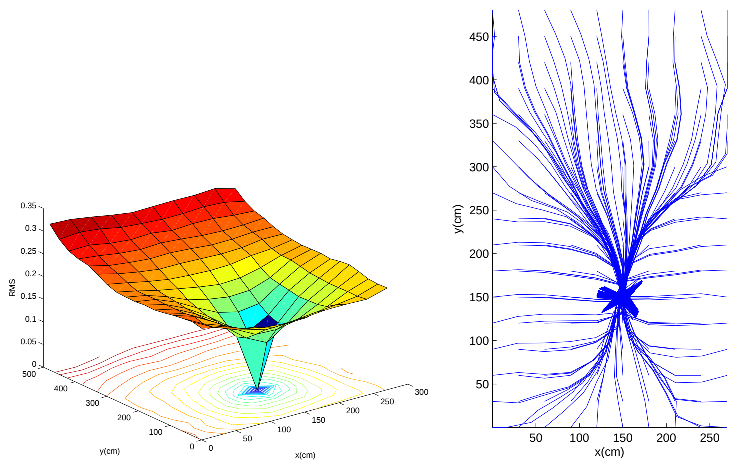
3.1 Introduction

As we discussed in Chapters 1 and 2, homing by moving an agent so as to optimise a difference surface – a technique pioneered by Zeil et al. [2003] – is quite a promising solution to the visual homing problem. We see for example in Figure 3.1(a) a difference surface formed in which I_S and all current images I_C are captured in a static laboratory environment. A number of sample homing runs using this difference surface are shown in Figure 3.1(b), each beginning at a different point on the laboratory floor. The agent uses a gradient descent algorithm, described more fully in Section 3.2.1 to move so as to optimise the difference surface. All homing runs are successful (i.e. end within 30cm of the reference location).

Zeil et al. [2003] identified a significant problem with difference surface homing: when I_S and I_C are captured in different illumination conditions, the chances of homing successfully decrease, often dramatically. We see an example of this in Figure 3.2. This is obviously an issue in indoor environments – the focus of our study in this dissertation – where human occupants often change lighting conditions to suit their needs. Of course, illumination changes – sometimes frequently – outdoors too.

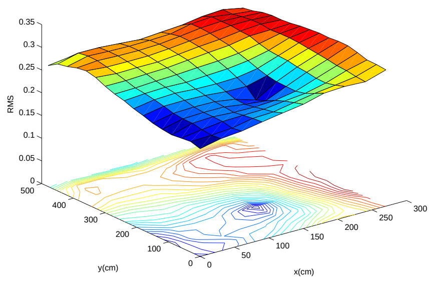
This chapter will focus in part on building difference surfaces in a laboratory environment which are robust to changes to illumination. There are two obvious remedies:

- Transform reference and current image intensities to minimise the effects of dynamic illumination (using e.g. histogram equalisation).
- Use image similarity measures other than the root-mean-square.

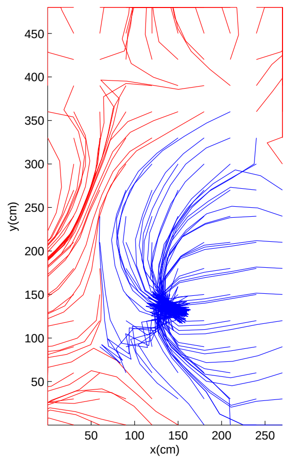


(a) (b)

Figure 3.1: **(a)** A difference surface formed using the RMS image similarity measure defined by Equation 3.1. The reference image was captured at x=150cm, y=150cm in a laboratory environment. The reference image was captured in the same landmark and illumination conditions as all other images. Notice the global minimum at the reference location and the absence of local minima. **(b)** Here we illustrate a number of homing runs using the difference surface in (a). Each homing run starts at a grid point on the laboratory floor. The simulated agent moves so as to optimise the difference surface using a gradient descent algorithm. All homing runs are successful (i.e. end within 30cm of the reference location).



(a)



(b)

Figure 3.2: **(a)** A difference surface formed using the RMS image similarity measure defined by Equation 3.1. As in Figure 3.1 the reference image was captured at x=150cm, y=150cm in a laboratory environment. In this example, though, the reference image was captured in a different illumination condition than all current images I_C. Notice that a local difference surface minimum coincides with the reference location but other local optima have appeared. **(b)** Here we illustrate a number of homing runs using the difference surface in (a). Each homing run starts at a grid point on the laboratory floor. The simulated agent moves so as to optimise the difference surface using a gradient descent algorithm. Successful homing runs (i.e. those ending within 30cm of the reference location) are shown in blue and homing failures are shown in red. There are a significant number of homing failures, unlike in Figure 3.1(b).

While these approaches are not mutually exclusive, we chose in this work to follow the second approach only. We find difference surface homing attractive because, among other reasons, the image similarity measurement involved is algorithmically simple and computationally efficient. To heavily preprocess I_S and/or I_C for the purposes of illumination correction would compromise this efficiency. Also, we could find no principled way of choosing which intensity transformations to apply and in what order. We chose therefore to focus on alternative image similarity measures in this work. We chose these alternatives based on careful and systematic analysis of the problems with difference surface homing using *RMS* to measure image similarity.

The locations of objects in human-populated environments are subject to frequent change. So along with dynamic illumination, we are also interested in this chapter on the effect of the movement of imaged objects between captures of I_S and I_C. This is a question not specifically addressed by Zeil et al. [2003].

This chapter is organised as follows. In Section 3.2 we describe difference surface homing experiments and their results with image similarity measured using *RMS*. This section also contains a detailed analysis of the drawbacks of *RMS* as an image similarity measure for use in visual homing in dynamic environments. This analysis leads us to an alternative similarity measure: the covariance. In Section 3.3 we repeat the experiments of Section 3.2 using covariance to assess image similarity. Our analysis of the limitations of covariance leads us to propose mutual image information as a third similarity measure. We experiment with mutual information as we did with *RMS* and covariance and analyse its strengths and weaknesses. We draw conclusions from our experimental results in Section 3.7. Section 3.8 describes future work.

3.2 Exploring the Root-Mean-Square Image Similarity Measure

3.2.1 Experiments and Results

We define the *RMS* image similarity between grayscale images I_S and I_C as

$$RMS(I_S, I_C) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(I_C(i) - I_S(i))^2} \tag{3.1}$$

Both I_S and I_C have N pixels.

We shall use Andrew Vardy's image database in our experiments. The database is

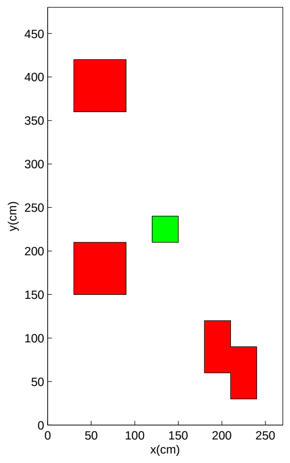


Figure 3.3: Schematic showing the dimensions of the experimental area represented by Vardy's image data sets (see text for details). Location of plant is indicated by a green polygon. Chairs are represented by red polygons.

available for download at http://www.ti.uni-bielefeld.de/html/research/avardy/index.html and is described in detail in Vardy and Möller [2005]. The database consists of a number of image sets. Each set contains 170 colour panoramic images captured every 30cm on a 2.7m x 4.8m horizontal grid on a laboratory floor. All images were captured with the camera in the same compass direction. Landmark layout and/or illumination conditions differ from set to set as described in Table 3.1.

We show a sample image from each data set in Figure 3.4. As in much other work on visual robotic navigation (e.g. Menegatti et al. [2004] and Hong et al. [1991]), the panoramic images were made by pointing a camera at a three dimensional (in this case hyperbolic) mirror. The mirror appears as a large circle occupying most of each image in Figure 3.4. The rig supporting the mirror and camera is imaged outside this circle; the robotic platform carrying the mirror is imaged at the centre of the circle. This extraneous information must be removed from each image before the image is used for navigation; we created the mask depicted in Figure 3.5(a) to do so.

As stated above, we want to measure the success of visual homing with *RMS* difference surfaces in a laboratory environment in both static and dynamic visual conditions. To this end, we created a number of difference surface sets. Each difference surface set is defined by the source of snapshot and current images. For example, one set of difference surfaces used the "Original" images for both I_S and I_C; another set took I_S from "Winlit" and all I_C from "Original", thus simulating dynamic illumination. The complete list of data set pairings is given in Table 3.2. We used the pairings along

Original	All overhead lights are on. No obstructive objects have been placed on the floor of the experimental area.
Winlit	Only the bank of lights near the curtained window (upper half of the image) are switched on. Those near the door are off.
Doorlit	Only the bank of lights near the closed door are switched on. Those above the curtained window are off.
Arboreal	All overhead lights are on. A plant has been placed in the centre of the experimental area (solid green polygon in Figure 3.3).
Chairs1	All overhead lights are on. Three office chairs have been placed along the walls of the laboratory, out of the experimental area.
Chairs2	All overhead lights are on. The three chairs in “Chairs1” have been moved to the experimental area. The chairs are represented by solid red polygons in Figure 3.3.

Table 3.1: Description of each of the image data sets used in this work.

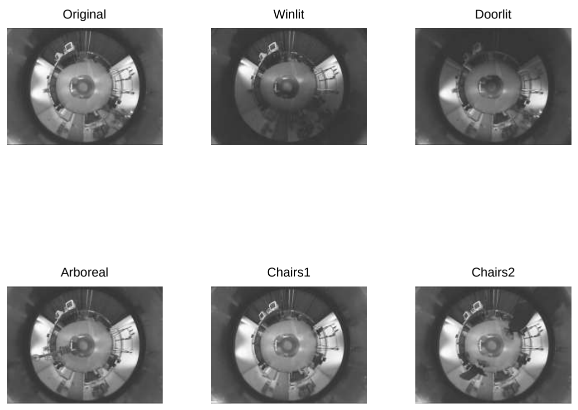


Figure 3.4: Sample from each of the Vardy image data sets used in this work. See text for details.

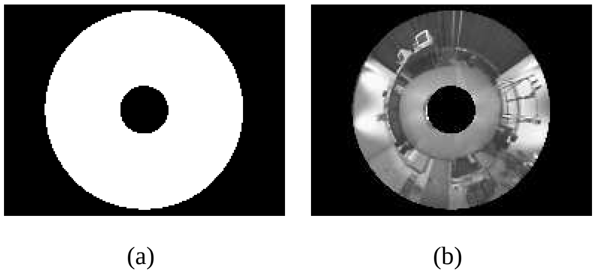


Figure 3.5: **(a)** Mask used to remove sections of Vardy images not corresponding to panoramic mirror. **(b)** Example of masked Vardy image. Every Vardy image used in this work is similarly masked.

		Current image data set					
		Original	**Winlit**	**Doorlit**	**Arboreal**	**Chairs 1**	**Chairs 2**
Snapshot set	**Original**	✓	✓	✓	✓	-	-
	Winlit	✓	✓	✓	-	-	-
	Doorlit	✓	✓	✓	-	-	-
	Arboreal	✓	-	-	✓	-	-
	Chairs 1	-	-	-	-	✓	✓
	Chairs 2	-	-	-	-	✓	✓

Table 3.2: Source of snapshot and current images for the various difference surfaces used in this work. A ✓indicates that the pairing was used to create a set of difference surfaces.

the diagonal of the table to create difference surfaces reflecting static environments. The off-diagonal pairings involving the “Winlit” and “Doorlit” sets simulate a laboratory environment in which illumination is non-constant. The off-diagonal pairings involving the “Arboreal” set simulate an environment in which there is change in the location of a relatively unobtrusive landmark. The pairings $(Chairs1, Chairs2)$ and $(Chairs2, Chairs1)$ yield difference surfaces reflecting the movement of more prominent objects in the environment.

Every difference surface set consists of nineteen surfaces corresponding to the nineteen different snapshot locations shown in Figure 3.6. We chose these locations because they are fairly uniformly distributed around the experimental area while not being inside a chair (in the “Chairs2” set) or the plant (in the “Arboreal” set). If we had used all 170 grid points as snapshot locations, our simulated homing experiments

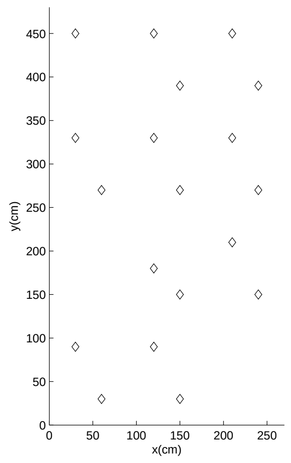


Figure 3.6: The set of nineteen snapshot locations used in this work. A diamond indicates a snapshot location.

would have required several months of computation time.

To rate the success of homing on the difference surfaces described above, we use the criterion defined in Vardy and Möller [2005]: the average return ratio. The return ratio (*RR*) is, for a particular difference surface, the ratio of successful homing runs to the total number of homing runs. Homing runs are initiated from each of the 169 non-snapshot locations (fewer if I_C is drawn from the “Chairs2” or “Arboreal” sets) in the experimental area. The average return ratio ($\overline{RR}$) is the mean return ratio for all nineteen difference surfaces in a difference surface set. The average return ratio provides a fairly good summary of the ease or difficulty of homing in particular conditions (static, dynamic illumination, or dynamic landmark locations).

The agent uses the gradient of the difference surface to home. The gradient of a two-dimensional function f (like the difference surface) at a particular point (x,y) is the vector of the partial derivatives of the function at (x,y) and is typically labelled $\nabla f(x,y)$.[1] The gradient at (x,y) points in the direction of greatest increase of the function at this point (Kleitman [2005]). To home, the agent assesses the gradient of the difference surface at its start location, moves by 30cm in the direction opposite the gradient (since the agent must move so as to *minimise* the *RMS* difference surface in order to home), and recalculates the gradient at its new position. The process of gradient calculation is described below. The agent continues in this manner until one of a set of stopping criteria is met; these criteria are discussed below.

[1]The gradient can be calculated for a function of more than two dimensions but this is not relevant to our task.

We use Matlab's **gradient** command to compute the gradient of the difference surface at each step. This command estimates the true gradient at a point (x,y) using the following two-sided differencing equation:

$$\nabla f(x,y) \approx \begin{bmatrix} \frac{f(x+h,y)-f(x-h,y)}{2h} \\ \frac{f(x,y+h)-f(x,y-h)}{2h} \end{bmatrix} \tag{3.2}$$

where h is the separation between grid points, in our case 30cm. In order to estimate the gradient at (x,y), Equation 3.2 implies that the homing agent must assess the difference surface at the four end points of a cross centred on (x,y). Equation 3.2 is undefined at the boundaries of the function. At these boundaries, a similar one-sided differencing equation is employed. We note that our gradient descent algorithm will typically find the local difference surface minimum nearest the start location. We explore alternative optimisation techniques in Chapter 5.

It will often be the case that during homing we must estimate the gradient of the difference surface at a non-grid point (i.e. a location in our experimental area at which no image was captured). We first thought of using Matlab's **interp2** function to interpolate the difference surface gradient (or rather, to interpolate each component of the difference surface gradient in turn as **interp2** is designed to interpolate two-dimensional scalar-valued functions, not vector-valued functions). Matlab's **interp2** interpolates a two-dimensional function given a matrix of samples of that function. **interp2** is capable of constructing new data points with a nearest neighbour, bilinear, bicubic or spline interpolation. Unfortunately, when drawing current images from the "Chairs2" or "Arboreal" data sets, the difference surface value for those grid points that are occupied by, respectively, a chair or plant will be undefined leading to undefined gradients. We found that **interp2** produces an undefined answer when presented with one or more sampled function values which are undefined (i.e. Not-a-Number in Matlab's scripting language). Thus we were forced to abandon **interp2**.

We instead devised our own two-dimensional interpolation method. To determine the gradient of the difference surface at a non-grid point (x,y) we use Equation 3.2 to compute the gradient of the difference surface at the four grid points closest to (x,y); these grid points form a square around (x,y). The x- and y-components of the interpolated gradient are computed separately but the procedure is identical for each so we shall describe the computation of the x-component only. We define a set F_x which contains the x-component of each grid point gradient used in the interpolation. The cardinality of F_x will be at most four. Undefined values are removed from F_x. We found no situation in our simulations in which F_x was empty. The interpolated value of the x-component of the gradient at (x,y) – denoted $f_x(x,y)$ will be the weighted mean of the elements of F_x:

$$f_x(x,y) = \frac{\sum_{g \in F_x} g \cdot w_g}{\sum_{g \in F_x} w_g} \tag{3.3}$$

The weight w_g is inversely related to the Euclidean distance between (x,y) and the grid point (x_g,y_g) corresponding to w_g. We define a set D each of whose elements D_g is the Euclidean distance between (x,y) and the grid point (x_g,y_g). Weight w_g is defined as

$$w_g = \frac{min(D)}{D_g} \tag{3.4}$$

This interpolation procedure is relatively simple to implement, provides smooth gradient interpolation and allows us to easily ignore undefined difference surface values.

It is possible during a homing run that the agent will collide with an object. The "Chairs2" and "Arboreal" data sets were captured with prominent objects – three chairs in the former case and a plant in the later – placed within the experimental area. Also, since we felt extrapolating the difference surface outside the bounds covered by the data sets would have yielded dubious surface values, we treated the experimental area as surrounded by a transparent wall. We assume our agent has been ringed with primitive bump sensors which give it a rough indication of the direction of the obstruction when a collision occurs. The agent turns away from the obstruction until it is free to move forward at which point it continues homing. We grant that the avoidance of the imaginary walls surrounding the experimental area may slightly inflate the success rates of our homing runs since the distance the robot can travel from any snapshot location is bounded. We argue though that each of the metrics we explore below will be equally advantaged by this decision.

The agent continues a homing run until one of the following stopping criteria is satisfied:

- The number of gradient calculations exceeds 400. Four hundred steps of 30cm each is sufficient to move between any two points in the experimental area, no matter how tortuous the route.

- The agent detects that its last twenty homing steps cluster around a particular location. This clustering will occur around local difference surface optima, which frequently (though not necessarily) coincide with the snapshot location. Clustering is detected by dead reckoning, a process (described in Chapter 2) by which a moving agent tracks its pose by integrating an estimate of all angles turned and linear distances travelled (see e.g. Krantz [1996]). Real robots such as the Koala used in the live experiments described in Chapter 6 often use wheel encoders to infer the pose change resulting from a movement command. A wheel encoder measures the number of rotations (including fractional rotations) made by a given wheel. Though the Koala has four wheels in total, two per side, the left and right wheel pairs are each driven by one motor, rotating in synchrony (K-Team [2001]); thus the Koala has two wheel encoders. In both simulated and live homing runs, our agent is limited to move in sequences consisting of pure translations and pure rotations. After one of these translations or rotations has been carried out, it is easy to use the wheel encoder readings along with knowledge of the wheel radii and axle length to compute the angle turned or linear distance travelled. As we related in Chapter 2, measurement of robot motion by wheel encoders is corrupted by both systematic and non-systematic noise. Wheel slippage due to smooth floors and/or high-magnitude acceleration is a common source of non-systematic measurement noise. Systematic noise comes from imprecision in the knowledge of robot wheel radii and axle length as well as the finite precision of the wheel encoder reading. We assume that our simulated environment is dry and flat so no wheel slippage occurs (none was observed in the live experiments in Chapter 6). We inject systematic noise into our simulated agent's dead reckoning system by adding an offset to the actual distance translated or angle rotated by the agent when performing a motor command. The offset for both rotation and translation is normally distributed with zero mean and for rotation a standard deviation of 0.1 degrees and for translation a standard deviation of 0.2 cm. This noise distribution was derived by Zampoglou et al. [2006]. We note that although dead reckoning noise leads to unbounded errors in pose tracking with respect to an initial pose over time, local clustering detection should not be greatly affected by such noise.

A homing run is deemed successful if it ends within 30cm of the snapshot location.

We carried out the experiment described above. The results are given in Table 3.3.

		Current image data set					
		Original	**Winlit**	**Doorlit**	**Arboreal**	**Chairs 1**	**Chairs 2**
Snapshot set	**Original**	0.977 (0.066)	0.584 (0.398)	0.654 (0.412)	0.933 (0.083)	-	-
	Winlit	0.427 (0.164)	0.949 (0.058)	0.020 (0.059)	-	-	-
	Doorlit	0.536 (0.210)	0.037 (0.110)	0.967 (0.051)	-	-	-
	Arboreal	0.956 (0.070)	-	-	0.924 (0.075)	-	-
	Chairs 1	-	-	-	-	0.975 (0.048)	0.598 (0.117)
	Chairs 2	-	-	-	-	0.953 (0.065)	0.578 (0.106)

Table 3.3: Average return ratios for homing experiments carried out on RMS difference surfaces in static and dynamic environments. The standard deviation of the average return ratio for each data set pairing is given in brackets.

3.2.2 Discussion

We can make several observations about the results presented in Table 3.3. When I_S and I_C are taken from the same data set, difference surface homing works quite well in almost all cases. For example, when all images are taken from the "Original" data set, the average return ratio is 0.977; when snapshot and current images are drawn from the "Winlit" set, $\overline{RR}$ is reduced to the still impressive 0.949. We see a glaring exception when all images are drawn from the "Chairs2" set. We shall discuss this exception below.

As reported in Zeil et al. [2003] for outdoor scenes, difference surface homing is much less successful when illumination conditions change between capture of I_S and I_C. When for example I_S is drawn from "Original" and I_C is drawn from "Winlit" $\overline{RR}$ is 0.584. The situation is even worse – in fact quite dire – when "Doorlit" is the source of the snapshot image; $\overline{RR}$ in this case is 0.037.

Results are mixed when landmarks change positions between capture of I_S and I_C. When I_S is drawn from "Chairs1" and I_C is drawn from "Chairs2", the average return ratio is 0.598 but when the sources of the snapshot and current images are reversed, $\overline{RR}$ improves to 0.953. There seems in fact to be a general diminution of average return ratio when I_C is drawn from "Chairs2" – even when the snapshot image is also drawn from "Chairs2" as we noted above. There is a similar, though much less dramatic trend when the "Arboreal" set is used. We speculate that the presence of the large objects in the experimental area, rather than movement of imaged objects between capture of snapshot and current images, causes difference surface homing to perform less well.

We shall try to justify this speculation below.

We would like to know why homing on *RMS* difference surfaces is affected by visual dynamism as described above. As given in Equation 3.1, *RMS* is difficult to analyse. We therefore break the equation into several terms as follows:

$$MSD(I_S, I_C) = \frac{1}{N}\sum_{i=1}^{N}(I_S(i) - I_C(i))^2 \tag{3.5}$$

$$= \frac{1}{N}\sum_{i=1}^{N}[I_S(i)]^2 + \frac{1}{N}\sum_{i=1}^{N}[I_C(i)]^2 - \frac{2}{N}\sum_{i=1}^{N}I_S(i)I_C(i) \tag{3.6}$$

Note first that the square root in Equation 3.1 has been removed from Equation 3.5, transforming the *RMS* into an expression of mean squared differences (*MSD*); we have done this because the square root plays no significant role in the behaviour of *RMS* and slightly muddies our mathematical analysis.

Equation 3.6 contains three terms. These terms can be transformed into forms more amenable to analysis. Several standard textbooks on statistics (see e.g. Svenshnikov [1968]) tell us that

$$\frac{1}{N}\sum_{i=1}^{N}[I_S(i)]^2 \approx Var(I_S) + (\bar{I_S})^2 \tag{3.7}$$

$$\frac{1}{N}\sum_{i=1}^{N}[I_C(i)]^2 \approx Var(I_C) + (\bar{I_C})^2 \tag{3.8}$$

$$\frac{1}{N}\sum_{i=1}^{N}I_S(i)I_C(i) \approx Cov(I_S, I_C) + \bar{I_S}\bar{I_C} \tag{3.9}$$

where $Var(I_S)$ is the variance of the intensities in I_S, $\bar{I_S}$ is the mean intensity in I_S; $Var(I_C)$ and $\bar{I_C}$ are defined similarly. $Cov(I_S, I_C)$ is the pixelwise covariance between I_S and I_C. Since the number of pixels in I_S and I_C is large, the difference between the left and right hand sides of each of the three equations above is exceedingly small in practice.

We substitute the right hand sides of Equation 3.7, Equation 3.8 and Equation 3.9 into Equation 3.6 and perform some algebraic manipulation. The *MSD* is transformed into

$$MSD(I_S, I_C) \approx Var(I_S) + Var(I_C) - 2Cov(I_S, I_C) + (\bar{I_S} - \bar{I_C})^2 \tag{3.10}$$

From Equation 3.10 it becomes clear that in moving our homing agent so as to minimise the *RMS* between I_C and I_S, the agent is actually simultaneously

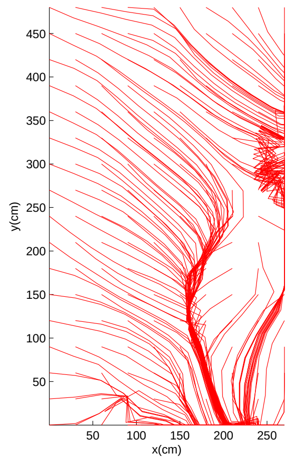


Figure 3.7: Homing paths starting from all non-goal positions for a difference surface with snapshot location at x=90cm, y=30cm. The snapshot image was drawn from the "Original" set and all current images were drawn from the "Winlit" set. No homing runs reached the goal location.

- seeking high covariance between I_S and I_C (i.e. minimising $-2Cov(I_S, I_C)$);
- seeking low variance current images (i.e. minimising $Var(I_C)$); and
- seeking equality of the mean intensities of I_S and I_C (i.e. minimising $(\overline{I_S} - \overline{I_C})^2$).

The second and third items above can cause homing errors. These errors are evident in a number of the homing experiments we performed whose results are summarised in Table 3.3.

We see the equalisation of mean intensities playing a deleterious role when for example the snapshot image is taken from the "Original" data set and all current images are drawn from the "Winlit" data set. Figure 3.7 shows the homing paths (starting from all non-goal positions) for this data set pairing when the goal location was set at x = 90cm, y = 30cm. None of the runs manages to reach the goal position, even those which begin quite close to the goal.

The influence of the term $(\overline{I_S} - \overline{I_C})^2$ in Equation 3.10 is easy to see when we consider the spatial gradient of MSD. Recall that the simulated homing agent follows the negative MSD gradient to generate the homing paths depicted in Figure 3.7. It follows directly (Kleitman [2005]) from Equation 3.10 that the negative spatial MSD gradient (denoted ∇MSD) is given by the following equation:

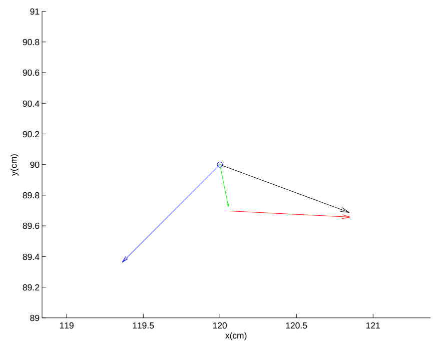


Figure 3.8: $-\nabla MSD(I_S, I_C)$ (black vector) as the vector sum of $-\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)]$ (green vector) and $-\nabla[(\overline{I_S} - \overline{I_C})^2]$ (red vector). The true home direction is represented by the blue vector.

$$-\nabla MSD(I_S, I_C) \approx -\nabla[Var(I_S)] + -\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)] - \nabla[(\overline{I_S} - \overline{I_C})^2] \tag{3.11}$$

Since I_S is constant during a homing run, $\nabla[Var(I_S)] = \vec{0}$ so

$$-\nabla MSD(I_S, I_C) \approx -\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)] - \nabla[(\overline{I_S} - \overline{I_C})^2] \tag{3.12}$$

For the purposes of our current example, it is useful to view $-\nabla MSD(I_S, I_C)$ as the vector sum of $-\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)]$ and $-\nabla[(\overline{I_S} - \overline{I_C})^2]$. With this in mind, we look at Figure 3.8. This figure shows $-\nabla MSD$ (black vector) at x=120cm, y=90cm for the homing runs depicted in Figure 3.7. We can see that $-\nabla MSD$ is oriented more than 90 degrees from the true home direction (blue vector) so any move in this direction will bring the agent farther from the goal. The error in $-\nabla MSD(I_S, I_C)$ is largely due to the influence of $-\nabla[(\overline{I_S} - \overline{I_C})^2]$ (red vector). The red vector points roughly in the direction of the window, which is the part of the experimental arena in the “Winlit” set whose average lighting intensity is most similar to that of the snapshot (which was taken from the “Original” set). When we home using $-\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)]$ (green vector) as the negative gradient direction, we meet with much more success; see Figure 3.9. The gradient error described above is qualitatively similar to many others we came across when lighting differed between snapshot and current images.

Equation 3.10 implies that the homing agent will be attracted to areas whose corresponding images have relatively low variance compared with images of nearby areas. Figure 3.10(a) shows the magnitude and gradient of the variance of all current images

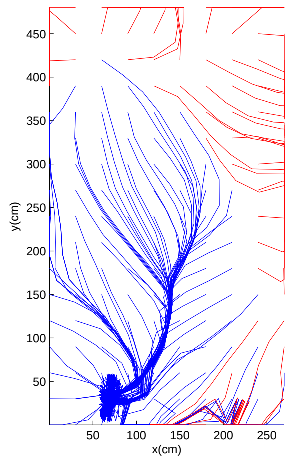
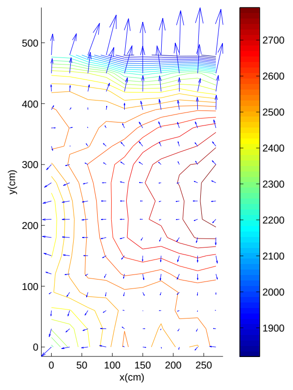


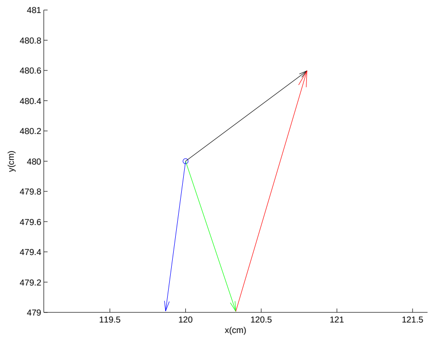
Figure 3.9: Homing runs on the difference surface described in the caption of Figure 3.7. Here, we used $-\nabla[Var(I_C)] + \nabla[2Cov(I_S, I_C)]$ as the negative gradient rather than Equation 3.12. Successful homing runs are shown in blue, unsuccessful in red. In comparison with Figure 3.7, we see that homing errors have been dramatically reduced.

from the "Original" data set. Relatively low variance values and gradients of high magnitude are evident at the top of the figure. We can see the effect of this area of rapidly changing, low variance in the example depicted in Figure 3.10(b). This figure shows $-\nabla MSD$ (black vector) at x=120cm, y=480cm. The mean intensity difference equalisation term has been removed from MSD so that its influence cannot be blamed for the homing error we shall describe. The true home vector, pointing to the snapshot location at x=60cm, y=30cm, is shown in blue. We can see that $-\nabla MSD$ is oriented more than 90 degrees from the true home direction (blue vector) so any move in this direction will bring the agent farther from the goal. The error in $-\nabla MSD(I_S, I_C)$ is largely due to the influence of $-\nabla[Var(I_C)]$ (red vector). It is clear in Figure 3.10(b) that $\nabla[2Cov(I_S, I_C)]$ (green vector) is a much better estimate of the true home direction than is $-\nabla MSD$. The gradient error described above is qualitatively similar to many others we came across when the agent came near regions whose images had relatively low variance.

The variance of current images also sometimes decreases when the agent moves toward a chair. This is because the chairs, from certain viewpoints, are large almost uniformly dark objects. When the agent moves toward a chair, the portion of an image taken up by a chair becomes larger, driving the variance of the image down. We can see this effect in Figure 3.11. Figure 3.11(a) shows I_C at x=210cm, y=180cm in the

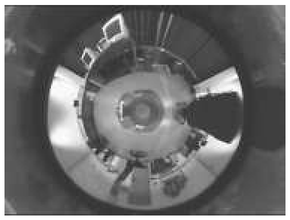


(a)

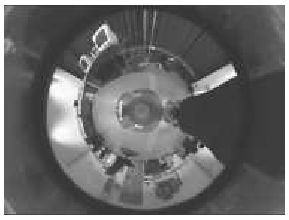


(b)

Figure 3.10: **(a)** The magnitude and gradient of the variance of all current images from the "Original" data set. The colour bar on the right indicates the magnitude of the variance. **(b)** Depiction of gradient error caused by low, rapidly changing variance; see text for details.



(a)

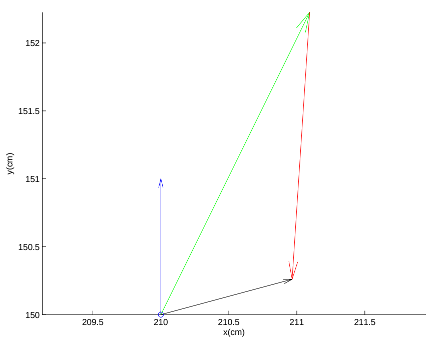


(b)

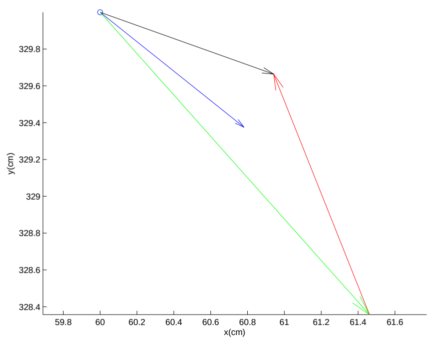
Figure 3.11: Illustration of the changing appearance of a chair as the agent moves towards it. **(a)** I_C at x=210cm, y=180cm in the "Chairs2" environment. **(b)** I_C at x=210cm, y=150cm in the "Chairs2" environment.

"Chairs2" environment; Figure 3.11(b) is I_C in the same environment but 30cm closer to the nearest chair. The image of the chair takes up more of the image in Figure 3.11(b) than in Figure 3.11(a) while the rest of the image content stays largely the same. The intensity variance decrease between the images in Figures 3.11(a) and 3.11(b) is 237; we assume this decrease is largely due to the changing appearance of the chair.

Figure 3.12(a) illustrates the effect of diminishing image variance caused by an image of a nearby chair. This figure shows $-\nabla MSD$ (black vector) at x=210cm, y=180cm; current images are taken from the "Chairs2" data set and the snapshot image was drawn from the "Chairs1" data set. The mean intensity difference equalisation term has been removed from MSD so that its influence cannot be blamed for the hom-



(a)



(b)

Figure 3.12: Two examples of MSD gradient error caused by proximate chairs; text for details.

ing error we shall describe. The true home vector, pointing to the snapshot location at x=210cm, y=450cm, is shown in blue. The black vector is $-\nabla MSD$; it clearly deviates from the true home vector. The error in $-\nabla MSD(I_S, I_C)$ is largely due to the influence of $-\nabla[Var(I_C)]$ (red vector). This red vector points toward the nearby chair. It is clear in Figure 3.12(a) that $\nabla[2Cov(I_S, I_C)]$ (green vector) is a better estimate of the true home direction than is $-\nabla MSD$. We show a similar example in Figure 3.12(b). Here we assess $-\nabla MSD$ (black vector) at x=60cm, y=330cm. The snapshot location is now at x=210cm, y=210cm. Again, the error in $-\nabla MSD(I_S, I_C)$ (black vector) is largely due to $-\nabla[Var(I_C)]$ (red vector).

It seems clear from the previous discussion that assessing the similarity between I_S and I_C with covariance rather than RMS is a more sensible approach. We determine if this is indeed the case in the next section.

3.3 Exploring the Covariance Image Similarity Measure

3.3.1 Experiments and Results

Here, we used covariance to measure the similarity between I_S and I_C where covariance (COV) is defined as:

$$COV(I_S, I_C) = \frac{1}{N}\sum_{i=1}^{N}(I_S(i) \cdot I_C(i)) - \overline{I_S} \cdot \overline{I_C} \tag{3.13}$$

N, I_S, I_C, $\overline{I_S}$ and $\overline{I_C}$ are defined as above.

		Current image data set					
		Original	**Winlit**	**Doorlit**	**Arboreal**	**Chairs 1**	**Chairs 2**
Snapshot set	**Original**	0.974 (0.061)	0.729 (0.227)	0.805 (0.321)	0.939 (0.084)	-	-
	Winlit	0.915 (0.211)	0.998 (0.007)	0.065 (0.201)	-	-	-
	Doorlit	0.668 (0.283)	0.076 (0.218)	0.994 (0.015)	-	-	-
	Arboreal	0.961 (0.069)	-	-	0.930 (0.083)	-	-
	Chairs 1	-	-	-	-	0.978 (0.043)	0.600 (0.097)
	Chairs 2	-	-	-	-	0.965 (0.069)	0.582 (0.124)

Table 3.4: Average return ratios for homing experiments carried out on *COV* difference surfaces in static and dynamic environments. The standard deviation of the average return ratio for each data set pairing is given in brackets.

To test the covariance as a potentially useful similarity measure in static and dynamic conditions, we carried out the same experiments described in Section 3.2.1, of course using Equation 3.13 to assess image similarity rather than Equation 3.1. When homing on *RMS* difference surfaces we attempted to minimise the *RMS* signal; with *COV* difference surfaces we seek to move the agent so as to *maximise* the value of the difference surface.

The results of our experiments using *COV* to measure image similarity are given in Table 3.4.

3.3.2 Discussion

For a given data set pairing, each average return ratio listed in Table 3.4 is greater than that given in Table 3.3. But are these differences statistically significant? We used McNemar's test (Sprent and Smeeton [2007]) to make this determination. McNemar's is a nonparametric test designed for nominal, paired data. Our data are paired in the sense that every homing run on an *RMS* difference surface corresponds to exactly one homing run on a *COV* difference surface. The results are nominal since each run results in either success or failure. We set the level of significance at 5%. Signficance results are given in Table 3.5.

Taken together, Tables 3.3, 3.4 and 3.5 tell us that the *COV* is sometimes a better image similarity measure than *RMS* in static conditions – in particular those conditions with non-uniform overhead lighting. *COV* always outperforms *RMS* when illumination conditions change between captures of I_S and I_C. We note, though, that *COV* results

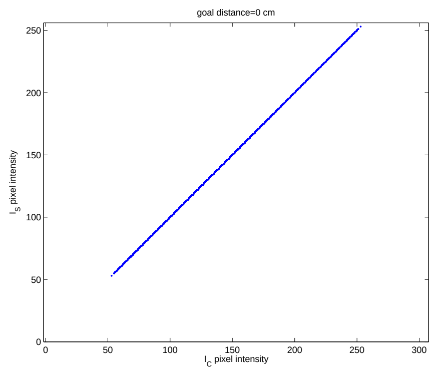
		Current image data set					
		Original	**Winlit**	**Doorlit**	**Arboreal**	**Chairs 1**	**Chairs 2**
Snapshot set	**Original**	N	Y	Y	N	-	-
	Winlit	Y	Y	Y	-	-	-
	Doorlit	Y	Y	Y	-	-	-
	Arboreal	Y	-	-	N	-	-
	Chairs 1	-	-	-	-	N	N
	Chairs 2	-	-	-	-	Y	N

Table 3.5: This table indicates whether the average return ratio given in Table 3.4 (*COV* results) is significantly different than the average return ratio given in Table 3.3 (*RMS* results) for a given data set pairing. A 'Y' indicates a statistically significant difference for a particular data set pairing; an 'N' indicates that there is not enough experimental evidence to reject the hypothesis that the average return ratios are equal. McNemar's test was used with a 5% level of significance. See text for details.

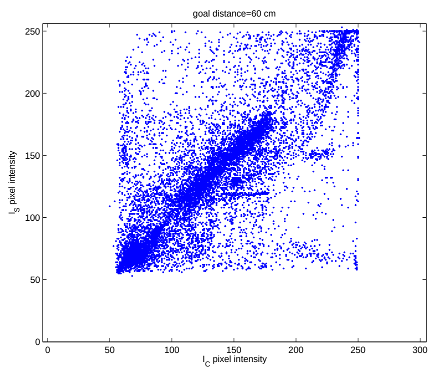
are quite poor in the face of relatively extreme illumination change, when I_S is drawn from "Doorlit" and I_C is drawn from "Winlit" or vice-versa.

COV sometimes outperforms *RMS* when objects move between capture of I_S and I_C – namely, the ("Arboreal","Original") and ("Chairs2", "Chairs1") data set pairings. The average return ratios for the two similarity measures are statistically indistinguishable when large objects are placed within the experimental area during capture of current images (i.e. when current images are drawn from the "Chairs2" or "Arboreal" sets). This last point is somewhat surprising. Given our analysis in Section 3.2.2, we expected difference surface homing with the *COV* measure to be more successful than homing with *RMS* in environments with large objects in the experimental arena. We are not as yet certain why this improvement fails to occur.

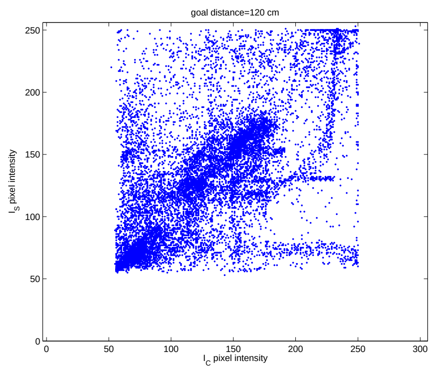
The covariance is only a trustworthy measure of the similarity between I_S given I_C when there is a linear relationship between pixel intensities in I_S and I_C. Such a linear relationship between I_S and I_C *does* exist in static conditions. Figure 3.13(a) shows a scatterplot in which we plotted the intensity of each pixel in I_C against the intensity of the corresponding pixel in I_S; intensities range from 0 to 255. Both images were taken from the "Original" data set and from the same location (x=60cm, y=270cm); in other words, I_C is identical to I_S. In this case, there is a perfectly linear relationship between I_S and I_C. As the capture position of I_C moves away from that of I_S, the strength of



(a)



(b)



(c)

Figure 3.13: **(a)** This figure depicts a scatterplot in which we plotted the intensity of each pixel in I_C against the intensity of the corresponding pixel in I_S. Both images were captured at the same location (x=60cm, y=270cm) and both were taken from the "Original" set. **(b)** Scatterplot formed as in (a) but here I_C was captured 60cm from capture position of I_S. **(c)** Scatterplot formed as in (a) but here I_C was captured 120cm from capture position of I_S.

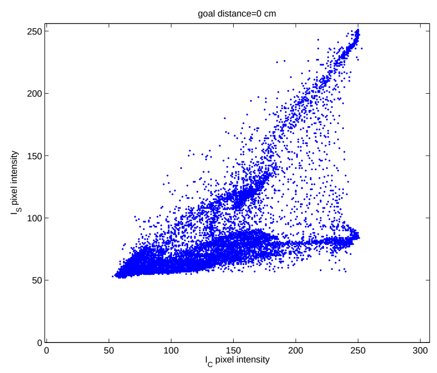


Figure 3.14: This figure depicts a scatterplot in which we plotted the intensity of each pixel in I_C against the intensity of the corresponding pixel in I_S. Both images were captured at the same location (x=60cm, y=270cm) but I_S was taken from the “Winlit” data set and I_C from the “Original” set.

the linear relationship decreases but remains clearly linear (see Figures 3.13(b) and 3.13(c)).

There ceases to be a linear relationship between pixel intensities in I_S and I_C when the two images are drawn from different data sets (i.e. in dynamic conditions). Figure 3.14 shows a pixel intensity scatterplot similar to those in Figure 3.13 in which we plotted the intensity of each pixel in I_C against the intensity of the corresponding pixel in I_S. Both images were captured at the same location (x=60cm, y=270cm) but I_S was taken from the “Winlit” data set and I_C from the “Original” set. This is clearly a nonlinear relationship. The bifurcated nature of the scatterplot is due to the fact that the portion of I_S which images the curtained window is quite similar to the same part of I_C; the rest of I_C is brighter than corresponding parts of I_S since lights above the door were turned off when I_S was taken.

3.4 Exploring the Mutual Information Image Similarity Measure

Though there isn’t a strictly linear relationship between I_S and I_C in Figure 3.14, I_S is quite *predictable* given I_C. For example, if we are told that a pixel in I_C has value 200, then we can predict with high probability that the corresponding pixel in I_S has an intensity close to either 60 or 175. This predictability can be measured with mutual image information. Mutual information as an image similarity measure was pioneered by Viola and Wells [1995] for the purposes of appearance-based image registration. Similar though apparently independent work was published by Maes et al. [1997]. We reviewed image registration and its similarity to visual homing in Chapter 2.

Mutual image information (MI) can be defined as

$$MI(I_S, I_C) = H(I_S) - H(I_S|I_C) \tag{3.14}$$

where $H(I_S)$ is the entropy of I_S and $H(I_S|I_C)$ is the conditional entropy of I_S given I_C. This definition of mutual information is adapted from the one given in Hill et al. [2001]. Entropy and conditional entropy are themselves defined as follows:

$$H(I_S) = -\sum_{a=0}^{B-1} p_S(a) log_2(p_S(a)) \tag{3.15}$$

$$H(I_S|I_C) = -\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} p_{SC}(a,b) log_2(p_{S|C}(a|b) \tag{3.16}$$

In Equation 3.15 $p_S(a)$ is the probability that a pixel will have intensity a ($0 \leq a < B$) in image I_S. In this work, $p_S(a)$ is calculated from the normalised intensity histogram of I_S. Image entropy is highest when all possible pixel values are equally likely (i.e. the pixel intensity histogram has a uniform distribution) and lowest (zero) when one pixel value is certain and the others never occur. The joint probability $p_{SC}(a,b)$ in Equation 3.16 is the probability that a given pixel in I_S has intensity a and the same pixel in I_C has value b; $p_{SC}(a,b)$ is calculated from the normalised joint intensity histogram of I_S and I_C. Finally, the conditional probability $p_{S|C}(a|b)$ is the probability that a pixel will have intensity a in I_S given that the corresponding pixel in I_C has intensity b.

It is clear from Equation 3.14 that maximising the mutual information between I_S and I_C involves minimising the conditional entropy $H(I_S|I_C)$; $H(I_S)$ is constant while homing. Analysis of Equation 3.16 tells us that $H(I_S|I_C)$ is minimal (zero) if knowing that a pixel in I_C has intensity b allows us to predict with probability 1 that the corresponding pixel in I_S has intensity a for all a and b. Conditional entropy will be much higher if intensity values in I_C are poor predictors of corresponding pixel intensities in I_S. Hence, mutual image information is a measure of how predictable I_S is given I_C.

We use the following equation to compute MI in our experiments. This form is equivalent to Equation 3.14 (Hill et al. [2001]) but is slightly less computationally intensive.

$$MI(I_S, I_C) = \sum_{a=0}^{B-1}\sum_{b=0}^{B-1} p_{SC}(a,b) log_2(\frac{p_{SC}(a,b)}{p_S(a)p_C(b)}) \tag{3.17}$$

		Current image data set					
		Original	**Winlit**	**Doorlit**	**Arboreal**	**Chairs 1**	**Chairs 2**
Snapshot set	**Original**	0.905 (0.097)	0.602 (0.238)	0.892(0.208)	0.865 (0.098)	-	-
	Winlit	0.914 (0.115)	0.979 (0.072)	0.307 (0.305)	-	-	-
	Doorlit	0.790 (0.119)	0.274 (0.297)	0.987 (0.023)	-	-	-
	Arboreal	0.892 (0.097)	-	-	0.811 (0.133)	-	-
	Chairs 1	-	-	-	-	0.832 (0.135)	0.557 (0.126)
	Chairs 2	-	-	-	-	0.807 (0.148)	0.498 (0.152)

Table 3.6: Average return ratios for homing experiments carried out on *MI* difference surfaces in static and dynamic environments. The standard deviation of the average return ratio for each data set pairing is given in brackets.

3.4.1 Experiments and Results

We repeated the experiments described in Section 3.2.1, using *MI* as the image similarity measure rather than *RMS*. The results of these experiments are given in Table 3.6.

3.4.2 Discussion

As *COV* is generally a better measure of image similarity than *RMS* for the purposes of difference surface homing, we shall compare *MI* with *COV* only. As in Section 3.3.2, we use McNemar's test to determine if there is a statistically significant difference between the average return ratios given in Table 3.4 and Table 3.6. Table 3.7 makes clear that the average return ratios of *COV* and *MI* are statistically different for all but one data set pairing, when I_S is taken from "Winlit" and all current images are taken from "Original."

MI is generally not as good a similarity measure as *COV* in our experiments. *MI* outperformed *COV* for only four out of sixteen data set pairings (see Table 3.6). *MI* beat *COV* in situations where lighting conditions changed from snapshot to current image capture. This outperformance is dramatic for severe illumination changes: when snapshot images are drawn from the "Winlit" set and current images are drawn from the "Doorlit" set, or vice-versa. In fact, homing with the *COV* measure almost always fails for these data set pairings. We also note that when *COV* does beat *MI* the difference in return ratios is often not terribly large. In fact, the mean return ratio over all data set pairings is 0.76 for *COV* and 0.75 for *MI*. Since *COV* fails almost totally in the face

		Current image data set					
		Original	Winlit	Doorlit	Arboreal	Chairs 1	Chairs 2
Snapshot set	Original	Y	Y	Y	Y	-	-
	Winlit	N	Y	Y	-	-	-
	Doorlit	Y	Y	Y	-	-	-
	Arboreal	Y	-	-	Y	-	-
	Chairs 1	-	-	-	-	Y	Y
	Chairs 2	-	-	-	-	Y	Y

Table 3.7: This table indicates whether the average return ratio given in Table 3.6 (*MI* results) is significantly different than the average return ratio given in Table 3.4 (*COV* results) for a given data set pairing. A 'Y' indicates a statistically significant difference for a particular data set pairing; an 'N' indicates that there is not enough experimental evidence to reject the hypothesis that the average return ratios are equal. McNemar's test was used with a 5% level of significance. See text for details.

of relatively extreme lighting change and *MI* is at least competitive with *COV* in many other cases, we chose to use *MI* as an image similarity measure in the remainder of this dissertation. We note that *COV* should be used to assess image similarity if it is known that illumination conditions are likely to remain static.

3.5 Run-Time Comparison of Similarity Measures

The aim of this chapter is to compare various image similarity measures for the purpose of difference surface homing in static and dynamic environments. So far, we have made this comparison on the basis of homing failure rates. It is also sensible to measure the computation time required for each of these similarity measures. After all, the operator of a homing robot would not want her machine to sit stationary for interminable seconds while struggling to compute image similarity.

The computation of *RMS* with Equation 3.1 requires the summation of N terms where N is the number of pixels in I_S (equal to the number of pixels in I_C). Each term involves two pixel-value queries (one for the current and one for the snapshot image), one subtraction and one squaring. *RMS* is therefore an $O(N)$ algorithm.

To compute the first term of Equation 3.13, N multiplication operations are required. Each multiplication involves two pixel-value queries (one for the current and one for the snapshot image). The second term of Equation 3.13 is the product of the mean intensities of I_C and I_S. We can assume that the mean intensity of I_S is precomputed before homing begins. The calculation of the mean intensity of I_C requires N pixel-value queries and N summations. Thus, in all – like *RMS* – covariance computation is an $O(N)$ algorithm. For a given value of N, though, computation of Equation 3.13 may take somewhat longer than computation of Equation 3.1 as the former requires more fundamental operations (i.e. multiplication, summation, etc.) to be carried out per summation term.

Before computing Equation 3.17, the intensity histogram of I_C, that of I_S, and their joint histogram must be generated. The histogram of the image I_S can be compiled before homing begins and therefore does not count in the computational cost of mutual information. Computation of the intensity histogram of I_C requires that an array of B elements (where B is the number of intensity levels in I_C) be initialised. Following this, each of the N pixels of I_C is read and the corresponding element in the histogram array is incremented. Histogram computation is therefore an $O(B+N)$ algorithm. By a similar argument, it is clear that generation of the aforementioned joint histogram requires $O(B^2+N)$ time. We note that if $B^2 < N$, it makes sense to compute the intensity histogram of I_C by summing the elements of each row of the joint histogram, an $O(B^2)$ operation. Once we have the necessary histograms, computation of Equation 3.17 can begin. Equation 3.17 involves the sum of B^2 terms, each of which requires three histogram queries, two multiplications, one division and one call to the logarithm function. Computation of Equation 3.17 therefore requires $O(B^2)$ operations. Taking the histogram generation into account, computation of mutual information takes $O(B^2+N)$ time.

For a given number of intensity levels and image size, mutual information computation requires more time than the calculation of covariance or *RMS*. To see what this difference means in practice, we timed each of these three similarity measures for various image sizes. In the case of *MI* we also varied the number of intensity levels. This test was carried out on a Dell Pentium 4 Optiplex 2.0 GHz computer, the same computer used to compute image similarity in our live robotic experiments described in Chapter 6. Similarity measures were computed using purpose-built Matlab functions. We created random grayscale images as input to the similarity functions. The size of these images varied from 424130 pixels (the size of the images in Vardy's database) down to 24130 pixels in steps of 50000 pixels. For all covariance and *RMS* calculations and for one set of mutual information calculation the number of intensity levels

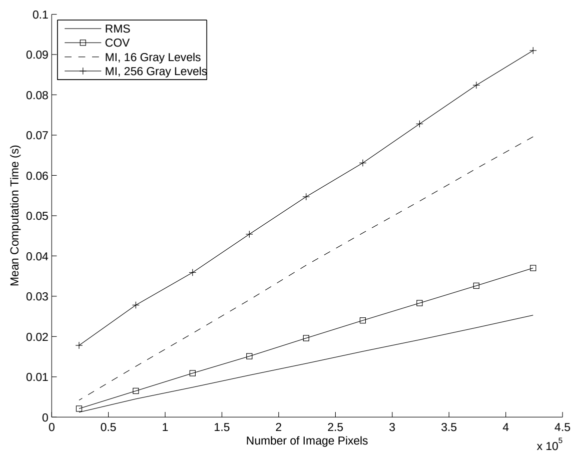


Figure 3.15: This graph depicts the results of our experiment comparing the time required to compute the image similarity measures examined in this chapter. See the text of Section 3.5 for details of the experimental procedure.

was set to 256. We carried out a second set of mutual information computations using images downsampled to 16 gray levels.

The results of this experiment are depicted in Figure 3.15. Each data point in this figure is the mean time required for 100 similarity measure calculations. The standard deviation for each data point was quite small, too small for meaningful error bars to be included in the figure. As predicted, *RMS* is the speediest similarity measure for all image sizes, followed by the covariance. For all image sizes, *MI* (with images downsampled to 16 graylevels) took more time than covariance. *MI* with images with 256 graylevels took the most time.

3.6 Comparison with Other Homing Methods

In this section we determine how the difference surface based homing algorithms described in this chapter rate against other visual homing studies which employed Vardy's image data set.

3.6.1 Vardy and Möller [2005]

Vardy and Möller [2005] were the first researchers (appropriately enough) to extensively use Vardy's image database for visual homing studies. We reviewed in Chapter 2 the novel homing algorithms developed by Vardy and Möller [2005]. Perhaps the most impressive of these algorithms in terms of consistent performance and computational complexity is the one called FirstOrder. Briefly, this algorithm computes feature correspondence between snapshot and current images using the intensity-based optic flow between these two images. More detail about FirstOrder can be found in Chapter 2.

Vardy and Möller [2005] reported that the mean return ratio when FirstOrder was used to home with all images taken from the "Original" data set was 0.9746. In tests where snapshots were taken from one data set and current images from the same or another set to simulate homing in several static and dynamic environments (the same procedure we used in this chapter), the mean return ratio for FirstOrder over all such data set pairings was 0.614. We note that Vardy and Möller [2005] used a superset of the image data sets we employed in this chapter. Unfortunately, the return ratio for individual data set pairings is not given quantitatively. Rather, in Figure 12 in Vardy and Möller [2005] return ratio values are encoded as shades of gray with white indicating a return ratio of 1 and black a return ratio of 0. Vardy and Möller [2005] state that FirstOrder had particular difficulty when lighting changed between snapshot and current images. This may be because FirstOrder assumes constant illumination between snapshot and current images. This difficulty is reflected in the qualitative results reported in Figure 12 in Vardy and Möller [2005]. Finally, we note that FirstOrder is a relatively fast homing algorithm as it requires no searching to solve the correspondence problem. Vardy and Möller [2005] reported that image processing for FirstOrder took on average 193.7ms on a Pentium 4 2GHz processor.

Before comparing FirstOrder to the algorithms we looked at in this chapter, we note that there is a procedural difference between the experiments of Vardy and Möller [2005] and our own. Vardy and Möller [2005] tuned their image pre-processing steps to optimise the performance of each homing algorithm. This tuning was done by simulating homing with the "Original" data set with different sets of pre-processing parameters. The parameters which yielded the best return ratio results were used in subsequent trials in which different data sets were used. We did no image pre-processing in our work and no extensive parameter tuning. This procedural difference renders our homing experiments more realistic and potentially more difficult than those described in Vardy and Möller [2005].

Now we compare our homing results with those of Vardy and Möller [2005]. When image similarity is computed with the *RMS* measure, we reported in Table 3.3 a mean return ratio of 0.977 (standard deviation 0.066) when both current and snapshot images are taken from the "Original" data set. When image similarity is computed with the

COV measure, we reported in Table 3.4 a mean return ratio of 0.974 (standard deviation 0.061). For the mutual information image similarity measure, Table 3.6 gives a mean return ratio of 0.905 (standard deviation 0.097). Thus for two of the three similarity measures we used in this chapter, the return ratios were about equal to that of Vardy and Möller [2005] for images taken from the “Original” data set. The return ratio of 0.905 for mutual information is lower than expected in light of other static results. We note that Table 3.6 reports that when both snapshot and current images are taken from the “Winlit” set, the mean return ratio for mutual information-based homing is 0.979 (standard deviation 0.072). Also, when both snapshot and current images are taken from the “Doorlit” set, the average return ratio is 0.987 (standard deviation 0.023). One would assume that these static environments are more difficult to home in than the “Original” environment due to non-uniform overhead lighting. Despite this, the mutual information-based homing performs quite well.

We said above that the mean return ratio for FirstOrder over all data set pairings considered in Vardy and Möller [2005] was 0.614. The mean return ratio for *RMS*-based difference surface homing over all data sets considered in this chapter is 0.69; for *COV* the value is 0.76 and for *MI* the value is 0.75. Thus our difference surface optimisation homing method provides better overall behaviour than FirstOrder for all of the image similarity measures considered.

Finally, we said above that FirstOrder requires on average 193.7ms on average to process one pair of current and snapshot images to generate a home vector. The data in Figure 3.15 indicate that the computation of image similarity can be up to ten times faster than the computation of a home vector with FirstOrder. This data was generated on a computer with the same processor speed as that used by Vardy and Möller [2005]. The reader may argue that to generate a home vector (i.e. gradient) in difference surface homing the robot has to move to three adjacent locations and carry out image similarity measurements at these three locations. Home vector computation takes longer therefore than a single image similarity computation. This is true but Möller and Vardy [2006] demonstrated that the robot need not actually move in order to generate a gradient estimate of the difference surface. Thus, home vector computation in difference surface homing is still potentially faster than home vector computation by FirstOrder.

3.6.2 Möller and Vardy [2006]

Möller and Vardy [2006] used Vardy's "Original" data set in their experiments to test a gradient-descent difference surface homing algorithm called the matched-filter descent in image distances or MFDID. The MFDID algorithm is described more fully in Chapter 2. Möller and Vardy [2006] compared the MFDID with the one-sided differencing gradient computation method we described in Section 3.2.1; Möller and Vardy [2006] called this the DID (descent in image distance) algorithm. Möller and Vardy [2006] used an image similarity measure for the DID algorithm almost identical to the root-mean-square measure employed by Zeil et al. [2003]. They computed the return ratios of DID and MFDID for the "Original" data set for all snapshot locations and report an average return ratio of 0.932 (standard deviation 0.21) for the DID algorithm and an average return ratio of 0.956 (standard deviation 0.17) for the MFDID algorithm. Clearly, in these tests, MFDID outperformed DID. As we explained in Chapter 2, MF-DID also has the virtue that – unlike DID and our own optimisation algorithms – the robot does not move in order to compute the gradient of the difference surface at a particular location.

How do these results compare with our own homing experiments using the "Original" data set? Before answering this question, we note an important procedural difference between the experiments reported in this chapter and those described in Möller and Vardy [2006]. Möller and Vardy [2006] removed high frequency image information from current and snapshot images using a Butterworth filter. They chose filter parameters which optimised the performance (as judged by return ratios) of the homing methods they explored. According to Figure 9 (bottom plot) in Möller and Vardy [2006], this optimisation process favoured the MFDID algorithm over the DID algorithm. As the homing algorithms we examined in this chapter are similar to DID, the use of the Butterworth filter may advantage the MFDID algorithm as compared to our algorithms as well. This procedural difference renders our homing experiments more realistic and potentially more difficult than those described in Möller and Vardy [2006].

Now we compare our results with those of Möller and Vardy [2006]. When image similarity is computed with the *RMS* measure, we reported in Table 3.3 a mean return ratio of 0.977 (standard deviation 0.066) when both current and snapshot images are taken from the "Original" data set. When image similarity is computed with the *COV* measure, we reported in Table 3.4 a mean return ratio of 0.974 (standard deviation 0.061). For the mutual information image similarity measure, Table 3.6 gives a mean return ratio of 0.905 (standard deviation 0.097). So our homing algorithms produce better results than both DID *and* MFDID when the *RMS* and *COV* measures are used to compute image similarity. One reason for this might be that we are using different snapshot locations than Möller and Vardy [2006] did. We believe this is not a likely explanation, though, since our snapshot locations spanned the entire experimental area just as did those of Möller and Vardy [2006]. A second more plausible explanation is that we when possible used two-sided differencing to compute difference surface gradients (see Section 3.2.1) whereas Möller and Vardy [2006] used one-sided differencing. It is quite possible that the former method produces more accurate difference surface gradients as it uses more difference surface information to compute them. This is a question we take up again in Chapter 5.

Unfortunately, difference surface homing with the mutual information image similarity measure in the "Original" data set does not perform as well as DID or MFDID. In defence of the mutual information measure, we note that Table 3.6 reports that when both snapshot and current images are taken from the "Winlit" set, the mean return ratio for mutual information-based homing is 0.979 (standard deviation 0.072). Also, when both snapshot and current images are taken from the "Doorlit" set, the average return ratio is 0.987 (standard deviation 0.023). One would assume that these static environments are more difficult to home in than the "Original" environment due to non-uniform overhead lighting. Despite this, the mutual information-based homing performs quite well.

In a follow-up to Möller and Vardy [2006], Möller et al. [2007] also used a part of Vardy's image database in their homing work. The work in Möller et al. [2007] is described in Chapter 2. Möller et al. [2007] did not use the return ratio to measure the effectiveness of their homing solution. They in fact did not carry out full homing runs at all. Möller et al. [2007] instead computed home vectors for various locations around a snapshot location. They were primarily interested in comparing the angular error between true and estimated home vectors for various difference surface optimisation algorithms. As their criteria for success are different from ours, we shall not compare our work with that of Möller et al. [2007].

3.6.3 Pons et al. [2007]

Pons et al. [2007] also used Vardy's image database in their visual homing study. We reviewed the homing algorithm of Pons et al. [2007] in Chapter 2. Briefly, these authors extracted SIFT features from current and snapshot images. Corresponding features were used as input to the homing algorithm of Vardy and Möller [2005].

We should note two important procedural differences between our work and that reported in Pons et al. [2007]. To make their algorithm more robust to changes in lighting and object location at a particular snapshot position, Pons et al. [2007] collect several images over time from that position. For their outdoor test for example, they collect images throughout a particular day (the precise number of images is left unspecified). SIFT features for a particular snapshot location are extracted from all representative snapshot images. We on the other hand use one image to represent a snapshot location. We consider our approach more realistic as it is unlikely that an autonomous robot will stay in one place for an entire day, powered on, capturing images. One could argue that the robot would leave and return to the snapshot location over the course of a day, taking a new snapshot image each time it arrives home. This approach assumes that the robot would be able to home to its snapshot location using the snapshot images already captured. This assumption is not tested by Pons et al. [2007]. A second difference between our work and that of Pons et al. [2007] is in how Vardy's environments are rendered dynamic. We use image sets captured by Vardy in the same office environment. Vardy altered the environment for each set by moving landmarks or turning overhead lights on or off. Pons et al. [2007] take a single set (presumably Vardy's "Original" set) and alter the images to simulate changes in lighting or object movement. We consider our tests therefore to be more realistic. These procedural differences make it difficult to compare our homing method with that of Pons et al. [2007]. Nonetheless, we attempt a comparison below.

Pons et al. [2007] find that their method is affected by dynamics in the environment and that its success is highly dependent on the snapshot position. For one snapshot position near the centre of Vardy's grid (at x=180cm and y=300cm in the coordinate system in Figure 3.6), they report return ratios of 1 for both relatively minor and relatively major environmental dynamics. When the snapshot location is moved to a corner (x=30cm, y=30cm), the return ratios drop to 0.88 and 0.77 respectively. Pons et al. [2007] attribute the poorer return ratios in the latter case to some non-goal positions being too far from the snapshot location for correct SIFT feature correspondence to occur.

Here we compare the homing results of Pons et al. [2007] with our own. Unfortunately, the snapshot locations we used (see Figure 3.6) did not coincide with those chosen by Pons et al. [2007]. We therefore examine homing runs for snapshot loca-

		Current image data set					
		Original	Winlit	Doorlit	Arboreal	Chairs 1	Chairs 2
Snapshot set	Original	0.976	0.824	1.000	0.906	-	-
	Winlit	0.982	1.000	0.053	-	-	-
	Doorlit	0.694	0.288	1.000	-	-	-
	Arboreal	0.953	-	-	0.900	-	-
	Chairs 1	-	-	-	-	0.912	0.688
	Chairs 2	-	-	-	-	0.929	0.659

Table 3.8: Return ratios for homing experiments carried out on MI difference surfaces in static and dynamic environments. The snapshot location for all homing runs was at x=150cm, y=270cm in the coordinate system in Figure 3.6.

		Current image data set					
		Original	Winlit	Doorlit	Arboreal	Chairs 1	Chairs 2
Snapshot set	Original	0.871	0.482	0.424	0.718	-	-
	Winlit	0.829	1.000	0.394	-	-	-
	Doorlit	0.876	0.000	0.971	-	-	-
	Arboreal	0.888	-	-	0.641	-	-
	Chairs 1	-	-	-	-	0.600	0.306
	Chairs 2	-	-	-	-	0.641	0.318

Table 3.9: Return ratios for homing experiments carried out on MI difference surfaces in static and dynamic environments. The snapshot location for all homing runs was at x=60cm, y=30cm in the coordinate system in Figure 3.6.

tions close to those used by Pons et al. [2007]. We compare our results at snapshot location x=150cm, y=270cm with the results of Pons et al. [2007] at snapshot location x=180cm, y=300cm. Both of these home positions are central in Vardy's image capture grid. The return ratios for homing to x=150cm, y=270cm are given in Table 3.8; all Vardy data set pairings examined in this chapter are represented in this table. We note that many of the return ratios in Table 3.8 are less than 1 (the return ratio reported by Pons et al. [2007] for snapshot location x=180cm, y=300cm). We stress, though, that due to the procedural differences stated above, our results are not directly comparable to those of Pons et al. [2007].

We compare our results at snapshot location x=60cm, y=30cm with the results of Pons et al. [2007] at snapshot location x=30cm, y=30cm. Both of these home positions are close to the lower-left-hand corner in Vardy's image capture grid. The return ratios for homing to x=60cm, y=30cm are given in Table 3.9; all Vardy data set pairings examined in this chapter are represented in this table. We note that several of the return ratios, including those resulting from dynamic conditions, are greater than or approximately equal to 0.88 which is the larger of the two return ratios reported by Pons et al. [2007] for homing runs to x=30cm, y=30cm. We again stress that comparison based on these results is difficult due to experimental differences. Nonetheless, the data in Table 3.9 indicates that our method may be on par – at least for relatively minor environmental dynamism – with the feature-based method of Pons et al. [2007].

3.7 Conclusions

The work of Zeil et al. [2003] suggests that visual homing by optimising *RMS* difference surfaces outdoors works well in static conditions but is compromised when the illumination conditions in which the snapshot was captured are different than those in which current images are captured. In this work we confirmed that this was also the case in an indoor, laboratory environment. Our analysis of *RMS* revealed that a homing agent will move so as to equalise the average intensities of I_C and I_S in dynamic illumination conditions. We also showed empirically that homing on *RMS* difference surfaces is robust to at least some movement of imaged objects. Our novel analysis of the *RMS* image similarity measure led us to infer that when I_C is captured near a large, monochromatic object (like a chair in the "Chairs2" data set) and C is relatively far from S, the homing agent may be attracted to this object rather than to S. We confirmed this finding – at least anecdotally – by looking at individual homing runs in our laboratory environment.

Our analysis of *RMS* in Section 3.2.2 led us to propose the covariance as an alternative image similarity measure. Judging by the average return ratios listed in Tables 3.3 and 3.4, the covariance is a better similarity measure than root-mean-square in our laboratory environment in both static and dynamic conditions. As we noted in Section 3.3.2, the covariance assumes a linear relationship between pixel intensities in I_S and I_C. This is not always the case in dynamic conditions. The mutual image information similarity measure (Section 3.4) makes no such assumption and instead more broadly measures how *predictable* I_S is given I_C.

The comparison between covariance and mutual information is somewhat ambiguous: mutual information does dramatically better than covariance in most cases where illumination changes between I_C and I_S. Mutual information does slightly less well than covariance in static environments and in environments in which the agent passes near large, monochromatic objects. Since mutual information is generally a relatively good image similarity measure and sometimes relatively very good, we will use it to measure image similarity for the purposes of difference surface homing in the rest of this dissertation.

In Section 3.5, we compared the time required to calculate each of the image similarity measures. As predicted, *RMS* was the fastest for a given image size, followed by *COV* which was in turn faster than *MI*. We note, though, that none of the similarity measures was particularly slow; none required more than 0.1 seconds for any image size used. Still, as we have argued above that *MI* provides the most robust homing performance of the three measures, we would like to find ways of speeding the computation of *MI* without degrading homing performace to a high degree. We undertake this work in the next chapter.

In Section 3.6 we compared difference surface-based visual homing with other recently published visual homing algorithms which were tested with Vardy's database. We argued that comparison was difficult because (1) other authors made certain assumptions which made their homing experiments easier than ours and (2) other authors used different parts of the Vardy database than we did. Still, our comparison indicates that the success of difference-surface based homing is on par with the other methods we examined.

3.8 Future Work

As we discussed in Section 3.1, we chose not to apply any intensity or colour transformations to either I_S or I_C. Of course, some transformations could render homing by optimising on the difference surface even more successful than reported in our results, especially when illumination conditions change between capture of I_S and and I_C. This was in fact strongly implied by Sturzl and Zeil [2007], a work we discussed in Chapter 2. These authors advocated the use of difference-of-Gaussian filtering, subtraction of image mean intensity from each image pixel, and local contrast normalisation to fight the effects of dynamic illumination.

The hue-saturation-value (HSV) colour space is reported by some researchers (see e.g. Gourichon et al. [2002]) to be more robust to illumination change than grayscale. Histogram equalisation is also sometimes used to account for dynamic illumination. A logical next step in our research is to apply these or other as yet unidentified image transformations to I_S and/or I_C before computing image similarity with mutual information.

To bolster illumination invariance, we could also find edges in both current and snapshot images by convolving each image with, say, a Sobel kernel. Edges are relatively illumination invariant features favoured by several image-based navigation schemes described in Chapter 2. We did some preliminary testing with edge-filtered images and found that they, using a difference surface approach, yielded a very small catchment area since edge overlap between current and snapshot images is minimal when the robot moves even a small distance from the goal location (i.e. 30cm). Another approach we could take is to use an image registration algorithm given edge-filtered current and snapshot images to find the pose change between the current and snapshot poses.

Sturzl and Mallot [2006] found that they could expand the catchment area of their homing routine by removing high-frequency components of current and snapshot images. These blurred images, though, lead to less precise homing performance. They suggested a multi-scale approach, using relatively blurry images at the beginning of a homing run and increasingly sharper images as the run proceeds. We could try the same approach in difference surface-based homing. We note that Zeil et al. [2003] advocated this approach but did not use it.

We have looked at three image similarity measures in this work: root-mean-square, covariance, and mutual information. There are other similarity measures described in the statistics literature which may be applicable; these include the Kendall tau rank correlation coefficient, Spearman's rank correlation coefficient (Anderson and Sclove [1986]), and pointwise mutual information. We carried out small-scale homing experiments using the first two measures to compute image similarity. We found neither Kendall nor Spearman produced better better than COV or MI. As these experiments were limited, though, we feel that more work with these alternative image similarity measures is warranted.

The image similarity measures we have experimented with require that I_S and I_C are aligned to the same external compass direction. In our work this alignment is achieved by rotating I_C. This is a drawback for two reasons: (1) a compass is required and (2) image rotation requires non-trivial computational effort. Must I_S and I_C be aligned in this way by measuring the discrepancy between the two? Not necessarily. We could, for example, simply compare the marginal intensity distributions of I_S and I_C (i.e. compare the normalised intensity histograms of these images). The Kullback-Leibler divergence (Weisstein [2007b]) is a commonly used measure of the difference between probability distributions and is defined as follows:

$$D_{KL}(I_S||I_C) = \sum_{a=0}^{B-1} p_S(a) log_2 \frac{p_S(a)}{p_C(a)} \tag{3.18}$$

$p_S(a)$ is the probability that a pixel has intensity a ($0 \leq a < B$) in I_S; $p_C(a)$ is defined similarly.

We performed the homing experiments described in Section 3.2.1 with a few of the data set pairings listed in Table 3.2 using Equation 3.18 to measure image similarity. When both snapshot and current images were taken from the "Original" data set, the average return ratio was 0.843 (standard deviation 0.212). This result is quite promising. Unfortunately, when snapshots were taken from the "Original" data set but current images were drawn from the "Winlit" set – simulating an illumination change – $\overline{RR}$ was only 0.110 (with a standard deviation of 0.172). This average return ratio was much smaller than when RMS, COV and MI were used to assess image similarity. We saw a similar degradation in average return ratio when snapshots were drawn from "Chairs1" and current images drawn from "Chairs2": $\overline{RR}$ was 0.289 with standard deviation of 0.205. It seems from this limited study that measuring the similarity between non-aligned images using the Kullback-Leibler divergence works fairly well in static environments but fails dramatically in dynamic ones. Future work in this area might involve normalising the intensity histograms of I_S and I_C before assessing their similarity or using an algorithm other than Kullback-Leibler to measure this similarity.

RMS (or something very similar) is quite often used to measure the difference between images in other image-based navigation schemes (e.g. the image warping algorithm of Franz et al. [1998b], the image-based Monte Carlo localisation algorithm of Menegatti et al. [2004], and several topological navigation algorithms). As in our work, these algorithms compare a current image with one or more images captured previously. Lighting and landmark locations might well have changed in the interim. We have demonstrated that mutual information is robust to this dynamism and so could provide a useful image similarity measure in image-based robot navigation in general.

As we discussed in Chapter 2, central-place foraging insects like ants and honeybees seem to employ a visual homing algorithm to rediscover a nest or food source. A number of visual homing algorithms, both feature-based and image-based, have been published in the robotics and insect ethology literature though which algorithm(s) these insects use is still unknown. A biologically plausible neural circuit implementing image similarity measurement using mutual information, covariance or *RMS* would lend weight to the hypothesis that insects move so as to optimise an image similarity signal in order to home. Zeil [2007] believes that a neural implementation of mutual image information is possible but no significant work on this has yet been done.

Chapter 4

Fast Computation of Mutual Image Information

4.1 Introduction

In this chapter, we explore methods to speed the computation of mutual information (MI). There are a few reasons to do this. If we can reduce the time to compute MI without drastically diminishing the ease with which a difference surface can be optimised, then an agent should arrive home faster from a given starting location. We note that it may be the case that robot movement time during homing is vastly greater than MI computation time, whether relatively fast or slow. We investigate this issue in this and the next chapter. Another reason to speed MI computation is that this image similarity measure could be used not only for homing but also for place recognition in vision-based topological navigation. Place recognition, as we discussed in Chapter 2, typically involves the comparison of a recently captured input image with a large number of reference (i.e. map) images. This high volume of image comparison is also required in image-based metric localisation (see Chapter 2) so fast computation of mutual image information may also be useful in this field.

There are two ways to compute mutual image information (MI) using the technology at our disposal: serially or in parallel. We have in our laboratory an AnaFocus EyeRIS Vision System (Castillo [2005b]) which is capable of performing a variety of operations in parallel on images stored as analogue signals. We present novel marginal and joint histogram algorithms for use in MI computation with the EyeRIS. Alternatively, we can capture images using a standard Webcam and store and process them with a serial Pentium-type computer, as is often done in visual navigation in robotics (see e.g. Franz et al. [1998b]).

The algorithms for parallel and serial MI computation suggest different ways to increase processing speed. Analysing our novel parallel algorithm, we find that a reduction in the number of intensity levels will produce a corresponding decrease in the time required to compute MI. On the other hand, the serial MI algorithm depends on both spatial image resolution and the number of image intensity levels. In our experiments, we will systematically vary the spatial and intensity resolutions of images used to compute MI difference surfaces. We shall attempt to determine whether any of our parameter settings produce difference surfaces which are likely to be unduly difficult to optimise. We shall then attempt to identify the best parameter settings and use these to time MI computation on systems described above.

This chapter is organised as follows: Section 4.2 describes the EyeRIS Vision System in detail. We also describe the Webcam and Pentium computer used for serial MI computation. In Section 4.3 we describe the algorithms used for the computation of mutual image information in serial and in parallel. Section 4.4 outlines the experiments we performed to compare the chosen methods of MI computation. Results are given in Section 4.5. We close with a discussion and conclusions in Section 4.6 and future work in Section 4.7. Related work is discussed in Section 4.8.

4.2 Materials

4.2.1 AnaFocus EyeRIS Vision System

The AnaFocus EyeRIS Vision System consists of two processors: the AnaFocus ACE16kv2 Focal Plane Processor (FPP) and the Altera NIOS II Digital Microprocessor (Castillo [2005a]).

The FPP is both an image capture system and parallel image processor. The FPP contains a rectangular grid of 128×128 photosensors. Each photosensor is coupled with a processor, connected with each of its eight neighbours in the grid. The FPP can store and process up to seven images at a time. Images are stored in analog form on the FPP, but are digitised into 256 gray levels when downloaded to the digital microprocessor's memory.

The FPP's processors are capable of performing several common image operations in parallel (see Castillo [2005c]). These include unary operations such as thresholding, global averaging and convolutions and binary logical and arithmetic (subtraction, addition) operations. All operations result in new images; this includes global averaging, which produces a constant image all of whose 128×128 values are set to the mean of the input image. The FPP is programmed in a proprietary language described in Castillo [2005c].

The digital microprocessor is responsible for the flow of program execution and manipulating the results of images processed by the FPP. The microprocessor is programmed in the C programming language. See Castillo [2005c] for more information. The microprocessor can store up to 1024 digital images of the type described above.

The EyeRIS is a very attractive system for parallel image processing. The application programming interface described in Castillo [2005c] is a comprehensive library of image manipulation functions. It is easy to incorporate these functions into control programs resident on the system's microprocessor. Geis et al. [2007] laud the EyeRIS for the "massively parallel character of the focal plane processor and the lower power consumption of the system together with a comparably small size" (p. 2930). The last two of these qualities make the system particularly appropriate for use in autonomous mobile robotics in which size and energy requirements are often important constraints.

4.2.2 Laptop and Webcam

We also compute MI with more traditional hardware than that described above: namely a Webcam providing images to a laptop. The camera is a Creative Labs Video Blaster Webcam III Model 6840 (Cre [2000]). The Webcam is capable of capturing colour images at the following resolutions: 640×480, 352×288, 320×240, 176×144, and 160×120. The camera's frame rate depends on the capture resolution; at resolution 640×480, the maximum frame rate is 24 frames per second.

Our laptop is an Acer Travelmate 313T. We chose this laptop because it is lightweight enough at 1.2kg to be carried by a Koala mobile robot. The Acer's processor speed is 266 MHz.

4.3 Computation of Mutual Image Information

Mutual information (MI) between snapshot image I_S and current image I_C is calculated with the following formula (adapted from Hill et al. [2001]):

$$MI(I_S, I_C) = \sum_i \sum_j p(i,j) log \frac{p(i,j)}{p_S(i) p_C(j)} \tag{4.1}$$

Algorithm 1 Compute histogram of intensity image serially

Require: I is intensity image with N rows, M columns and intensities $0..B-1$

Ensure: $hist[i]$ stores number of pixels with intensity i in I, $0 \leq i \leq B-1$

```
 1: procedure ComputeImageHistogramSerially($I$)
 2:     var int[0..B-1] $hist$                    ▷ Must initialise $hist$ to all zeros
 3:     for $r \leftarrow 1, N$ do
 4:         for $c \leftarrow 1, M$ do
 5:             $i \leftarrow I[r][c]$
 6:             $hist[i] \leftarrow hist[i] + 1$
 7:         end for
 8:     end for
 9:     return $hist$                             ▷ The image histogram of $I$
10: end procedure
```

Here, $p_S(i)$ is the probability that a pixel will have intensity i in image I_S; gray level intensities range from 0 to $B-1$ (B is 256 or less for our hardware). These probabilities are estimated from the intensity histogram of I_S; $p_C(j)$ is defined similarly for current image I_C. The joint probability $p(i, j)$ is the probability that the same pixel will have intensity i in I_S and intensity j in I_C. These probabilities are estimated from the normalised joint histogram of images I_S and I_C. The joint histogram of two images is a square matrix of size $B \times B$. Entry (i, j) in the joint histogram is the number of pixels in the first image with intensity i coinciding with pixels with intensity j in the second image. The logarithm in Equation 4.1 is to base 2.

Histograms are typically computed serially but with the EyeRIS system we have the option of generating them in parallel. We describe both methods below.

4.3.1 Serial Computation of Image Histograms

The algorithm for computing the intensity histogram of a grayscale image is included in almost every image processing textbook (see e.g. Fisher et al. [1996]). The pseudocode is given in Algorithm 1.

Explicit program listings for the computation of joint image histograms is somewhat difficult to find. It is easy, though, to glean the pseudocode from written description of the joint histogram (see e.g. Hill et al. [2001]). The serial joint histogram algorithm is outlined in Algorithm 2.

Algorithm 2 Compute joint histogram of two intensity images serially

```
Require: $I$, $J$ are intensity images with $N$ rows, $M$ columns and intensities $0..B-1$
Ensure: $jhist[i][j]$ stores number of coincidences of intensity $i$ in $I$ with intensity $j$ in
    $J$, $0 \leq i, j \leq B-1$
 1: procedure ComputeJointImageHistogramSerially($I$, $J$)
 2:     var int[0..B-1][0..B-1] $jhist$          ▷ Must initialise $jhist$ to all zeros
 3:     for $r \leftarrow 1, N$ do
 4:         for $c \leftarrow 1, M$ do
 5:             $i \leftarrow I[r][c]$
 6:             $j \leftarrow J[r][c]$
 7:             $jhist[i][j] \leftarrow jhist[i][j] + 1$
 8:         end for
 9:     end for
10:     return $jhist$                          ▷ The joint image histogram of $I$ and $J$
11: end procedure
```

4.3.2 Parallel Computation of Image Histograms

We devised novel single and joint histogram algorithms for use on the EyeRIS's parallel Focal Plane Processor.

Pseudocode for single image parallel histogram computation is given in Algorithm 3. The algorithm iterates B times, once for each gray level b. In each iteration, the algorithm computes a thresholded image in which only pixels with intensity value b in the original image are turned "on." The algorithm then counts the number of "on" pixels in the thresholded image and stores this count in the histogram bin for intensity b.

Figure 4.1 illustrates the thresholding process. The image to be analysed – the "original" – is in the top left-hand corner. The original consists of eight intensity levels $0..7$ and has dimensions 128×128. The original image contains eight intensity bands of equal area. The eight binary thresholded images are shown as well. Black pixels in the threshold image for intensity b indicate pixels in the original image not equal to intensity b. A histogram created from the original images will be uniform and identical; each bin will contain a count of 2048 pixels.

Algorithm 3, line 5 requires the computation of the number of 1's in the bandpass intensity thresholded image I_{Thresh}. There are at least two ways to do this with the EyeRIS's FPP. The FPP has a built-in function called AddressEventFunction (Castillo

Algorithm 3 Compute histogram of intensity image in parallel

```
Require: $I$ is intensity image with $N$ rows, $M$ columns and intensities $0..B-1$
Ensure: $hist[i]$ stores number of pixels with intensity $i$ in $I$, $0 \leq i \leq B-1$
1: procedure ComputeImageHistogramInParallel($I$)
2:     var int[0..b-1] hist                          ▷ Must initialise hist to all zeros
3:     for $i \leftarrow 0, B-1$ do
4:         $I_{thresh} \leftarrow$ bandpass threshold $I$ at intensity $i$  ▷ Produces binary image with 1's within threshold
5:         $hist[i] \leftarrow$ number of 1's in $I_{thresh}$ (see text for details)
6:     end for
7:     return hist                                    ▷ The image histogram of $I$
8: end procedure
```

[2005c]), which counts the number of white pixels in a binary image. Unfortunately, the running time of this operation – though very fast – is dependant on the number of 1's in the input image. Also, the count is capped at 4095 pixels due to memory limitations.

Alternatively, we could use the GlobalMean operator to compute the mean image intensity m of I_{Thresh}. Using special circuitry, the mean is computed almost instantaneously, regardless of image content (Castillo [2005c]). Given this mean, the number of white pixels in I_{Thresh} is $\frac{m \cdot 128^2}{255}$ (assuming white pixels are stored at intensity 255). As the GlobalMean returns an integer value in the range $[0, 255]$, this method will be somewhat inaccurate. For example, a true mean in the range [99.5, 100.5) will be rounded to 100, giving a count of 6425 white pixels. The actual number of white pixels is in the range [6393, 6457). For this reason, despite its limitations, we use the AddressEventFunction to compute the number of white pixels in I_{Thresh} rather than the GlobalMean method.

Our algorithm for parallel joint histogramming is given in Algorithm 4. The main idea is this: In order to compute the joint histogram value for intensities i and j we compute the bandpass thresholded image of I at intensity i producing binary image I_{thresh}. We similarly bandpass J at intensity j. We then compute a pixel-wise logical-and of I_{thresh} and J_{thresh}, which produces a binary image K whose pixels are only "on" at locations in which I has intensity i and J had intensity j. The count the number of "on" pixels in K is stored in the joint histogram entry at (i, j).

Figure 4.2 illustrates the process of creating the images K. Image I is given in

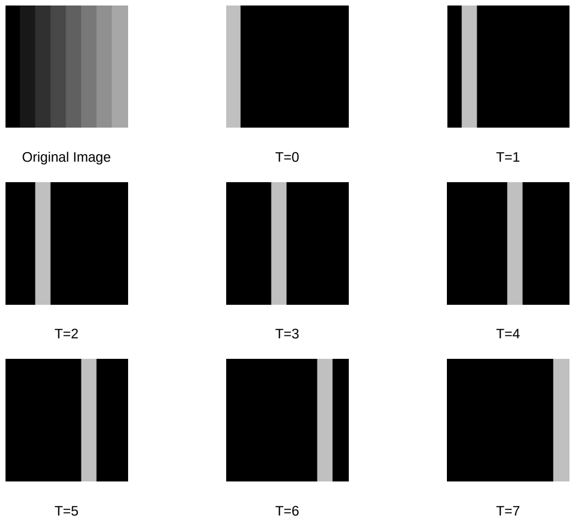


Figure 4.1: Images illustrating the thresholding process used for parallel single image histogramming. See text for more information.

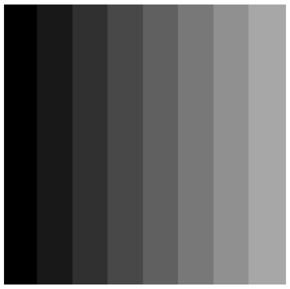
Figure 4.2(a) and J in Figure 4.2(b). Figure 4.2(c) shows the jointly thresholded image K for each value of i and j ($0 \leq i, j \leq 7$).

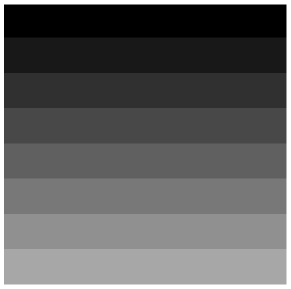
4.3.3 Time Requirements of MI Computation with Parallel and Serial Histogramming

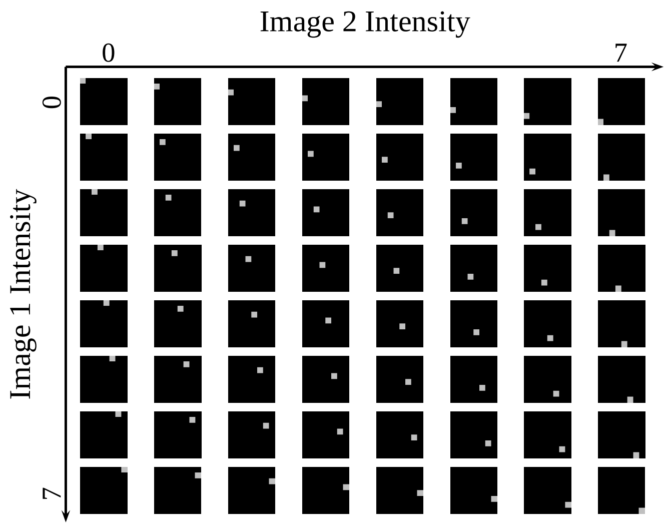
It is clear from Equation 4.1 that, once image intensity probabilities have been generated from image histograms, the computation of mutual information requires $O(B^2)$ time, where B is as before the number of image gray levels. Conversion of histogram counts to intensity probabilities also requires $O(B^2)$ operations. This analysis holds regardless of the method used to compute the histograms.

It follows from Algorithms 1 and 2 that serial computation of both single and joint image histograms requires $O(NM)$ operations, where each image has N rows and M columns. Thus, serial computation of mutual information requires a total of $O(B^2 + NM)$ steps.

Parallel computation of a single image histogram, on the other hand, requires $O(B)$ steps, as the loop in Algorithm 3 iterates B times. Generation of the joint histogram in parallel requires $O(B^2)$ steps, owing to the doubly-nested loop structure of Algorithm 4. Parallel computation of mutual information requires a total of $O(B^2)$ steps. Note that we assume that the instructions on lines 4 and 6-8 of Algorithm 4 require unit time (independent of the size of the input). As we shall discuss later, this is the case on


(a) Image 1


(b) Image 2



(c) Jointly Thresholded Images

Figure 4.2: Illustration of the parallel joint histogramming process. See text for details.

Algorithm 4 Compute joint histogram of two intensity images in parallel

```
Require: $I$, $J$ are intensity images with $N$ rows, $M$ columns and intensities $0..B-1$
Ensure: $jhist[i][j]$ stores number of coincidences of intensity $i$ in $I$ with intensity $j$ in
    $J$, $0 \leq i, j \leq B-1$
 1: procedure ComputeJointImageHistogramSerially($I$, $J$)
 2:     var int[0..B-1][0..B-1] $jhist$                    ▷ Must initialise $jhist$ to all zeros
 3:     for $i \leftarrow 0, B-1$ do
 4:         $I_{thresh} \leftarrow$ bandpass threshold $I$ at intensity $i$
 5:         for $j \leftarrow 0, B-1$ do
 6:             $J_{thresh} \leftarrow$ bandpass threshold $J$ at intensity $j$
 7:             $K \leftarrow I_{thresh} \wedge J_{thresh}$         ▷ $\wedge$ denotes logical pixel-wise "and"
 8:             $jhist[i][j] \leftarrow$ number of 1's in $K$
 9:         end for
10:     end for
11:     return $jhist$                          ▷ The joint image histogram of $I$ and $J$
12: end procedure
```

the EyeRIS platform but may not be so with all parallel image processing systems.

4.4 Experiments

4.4.1 Viable MI difference surfaces with reduced images

Given the analysis presented in Section 4.3.3 it is evident that the time required for serial histogram computation depends on both spatial and intensity resolution. A crucial question then is how much can we reduce spatial and/or intensity resolution without producing MI difference surfaces which are unduly difficult to optimise? We attempt to answer this question with experiments using Vardy's image data set, described in the previous chapter.

As indicated in Section 4.2 our Webcam is capable of producing images at various resolutions. We reproduce an approximate subset of these resolutions (352×288, 176×144, and 160×120) by scaling Vardy's images to, respectively, 50%, 25% and 20% of their original size. Additionally, we scale Vardy's images to 17% their original size to approximate images captured by the EyeRIS system.

Image scaling by a factor of $\frac{1}{n}$ is a two-step process. In the first stage, the original

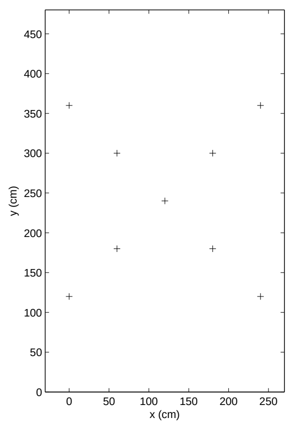


Figure 4.3: Snapshot coordinates used in experiments described in Section 4.4.

image is convoluted with an $n \times n$ averaging filter each of whose entries is $\frac{1}{n^2}$. The convolved image is then sampled at every n^{th} row and, within that row, every n^{th} column. Though our Webcam's resolution reduction algorithm is unpublished, some reports (see e.g. Filippov [2006]) suggest typical Webcams mimic this process to produce low-resolution images.

For each of the four spatial resolutions above, we varied the number of gray levels, using 256, 128, 64, 32 and 8 in turn. A simple linear binning process was used when necessary to reduce gray levels from the original 256. For example, to reduce to 8 gray levels, intensities in the range [0,31] in the original image are mapped to intensity 0, intensities in [32,63] are mapped to intensity 1, and so forth.

To form MI difference surfaces, we fixed snapshot locations at the nine coordinates indicated in Figure 4.3; this set was designed to be small while covering most of the experimental environment. For a given gray level and spatial resolution, we computed three difference surfaces for each snapshot location. They differ as follows:

1. Both snapshot and current images were taken from the "Original" set.

2. The snapshot was taken from the "Original" set and the current images were taken from the "Winlit" set.

3. The snapshot was taken from the "Original" set and the current images were taken from the "Chairs" set.

The first set of difference surfaces models a static environment, the second set models a dynamic environment with changing lighting and the third models a dynamic environment with changing object locations.

We created 675 difference surfaces in total. Given so many surfaces to assess, we could not simulate homing runs from all possible starting points for each surface as we did in the previous chapter. We therefore assessed return ratios for all 225 static difference surfaces by homing from thirty-two uniformly distributed starting locations on each surface. These homing simulations use the software described in the previous chapter.

To measure ease of optimisation on difference surfaces created in dynamic and static environments, we shall use the following criteria; these are less computationally intensive than the return ratio:

4.4.1.1 Global maximum at reference location

When homing, we shall assume that the reference location occurs at the global maximum of the difference surface. Thus it is crucial that for each difference surface the reference location coincides with the global maximum.

4.4.1.2 Local maxima

Commonly used optimisation algorithms guarantee – at best – the identification of the maximum nearest the starting location (see e.g. Adby and Dempster [1974]). Thus, local maxima away from the reference location should be rare and easy to distinguish from the global maximum. A local maximum is defined as a position whose value is greater than all immediate neighbours.

4.4.1.3 Meaningful gradients

Some optimisation schemes are gradient-based (again see Adby and Dempster [1974]); they attempt to determine and follow the local gradient (when maximising) or negative gradient (when minimising). Assuming we can accurately approximate function gradients, these gradients will only provide useful information if the angular divergence between the true home direction and the current local gradient is small. We therefore measure this angular divergence.

4.4.1.4 Monotonic difference surfaces

So-called direct search (Adby and Dempster [1974]) optimisation methods employ raw function values rather than local gradients (see e.g. the RunDown method in Zeil et al. [2003]). These algorithms assume that – when maximising – an increase in function value indicates a decrease in distance to a local maximum. We measure the probability that a unit move (i.e. 30cm) in a random direction will provide useful information. A move towards the reference location should correspond to an increase when using the MI metric. So too, a move away from the reference location should correspond to a decrease when using the MI metric. Below, we frequently refer to this as the "good moves" criterion.

4.4.2 Timing Experiments

Here, we determine the speed with which mutual information is calculated on the platforms described in Section 4.2 using the range of viable resolutions and gray levels identified in the previous experiments. We will look in particular at computation in parallel using the EyeRIS and computation in serial on our Acer laptop.

4.5 Results

4.5.1 Viable MI difference surfaces with reduced images

We use the criteria established in the previous section to judge the viability of the difference surfaces produced here. In all experiments, the global maximum of the difference surface coincided with the goal location, regardless of reduction of spatial and/or intensity resolution.

Table 4.1 gives the results of our homing experiments in static conditions with images reduced spatially and/or in number of gray levels. It is clear that these reductions have little effect on the success – as measured by return ratio – of homing in Vardy's laboratory environment.

Table 4.2 gives the mean number of local maxima for all spatial and intensity reductions for difference surfaces in which current and snapshot images were taken from the "Original" data set. Local maxima seem to decrease with decreasing numbers of gray levels. The maxima tend to increase with decreasing spatial resolution. As only nine difference surfaces were created for each pairing of spatial resolution and intensity resolution, there is not enough data to determine if the local maxima reported in Table 4.2 are significantly different.

The non-snapshot maxima reported in Table 4.2 may cause problems during homing. If an agent begins homing close to one of these spurious maxima, it may well be attracted to it in the homing process and even eventually halt at it. We remind the reader

	Num. Gray Levels				
Scale Down To	**256**	**128**	**64**	**32**	**8**
100%	0.97	0.97	0.97	0.95	0.98
50%	0.97	0.97	0.97	0.95	0.98
25%	0.98	0.98	0.97	0.97	0.97
20%	0.97	0.97	0.97	0.96	0.97
17%	0.97	0.98	0.97	0.95	0.97

Table 4.1: Return ratios for homing on MI difference surfaces formed using Vardy's images scaled to various spatial and/or intensity resolutions. Both snapshot and current images were taken from the "Original" data set.

	Num. Gray Levels				
Scale Down To	**256**	**128**	**64**	**32**	**8**
100%	1.9	1.9	1.9	1.8	1.0
50%	1.9	2.0	1.9	1.9	0.9
25%	1.7	1.9	1.9	1.7	1.3
20%	2.6	2.1	1.8	1.8	1.4
17%	2.2	2.4	2.6	2.1	1.4

Table 4.2: Mean number of local maxima (in addition to the true global maximum) in MI difference surfaces formed using Vardy's images scaled to various spatial and/or intensity resolutions. Both snapshot and current images were taken from the "Original" data set.

that the stopping criterion used by our homing algorithm cannot distinguish between local and global difference surface maxima. This is a problem frequently reported in the optimisation literature. We experiment with one method of avoiding non-snapshot maxima in the next chapter. We also give an extensive discussion of another method – simulated annealing – in the future work section of the next chapter.

Table 4.3 gives the mean number of local maxima for all spatial and intensity reductions for difference surfaces in which current images were taken from the "Winlit" set and snapshot images were taken from the "Original" set. A comparison of Table 4.2 and Table 4.3 indicates – in accord with the results of the previous chapter – that dynamic illumination leads to more difference surface local maxima than are seen in static conditions, making homing more difficult. Here, unfortunately, reduction of

	Num. Gray Levels				
Scale Down To	**256**	**128**	**64**	**32**	**8**
100%	3.2	3.8	3.7	3.6	3.6
50%	2.9	3.1	4.3	4.0	3.9
25%	3.2	3.3	3.7	4.1	4.6
20%	3.2	3.2	3.7	4.2	4.6
17%	3.7	3.9	3.6	4.3	4.3

Table 4.3: Mean number of local maxima in MI difference surfaces formed using Vardy's images scaled to various spatial and/or intensity resolutions. Snapshot were taken from the "Original" data set and current images were taken from the "Winlit" set.

	Num. Gray Levels				
Scale Down To	**256**	**128**	**64**	**32**	**8**
100%	3.6	3.9	3.8	4.1	3.6
50%	3.6	3.7	4.0	4.2	3.3
25%	4.0	4.3	4.2	4.2	3.3
20%	3.4	4.1	4.2	4.3	3.7
17%	3.9	4.4	4.9	4.0	3.6

Table 4.4: Mean number of local maxima in MI difference surfaces formed using Vardy's images scaled to various spatial and/or intensity resolutions. Snapshot were taken from the "Original" data set and current images were taken from the "Chairs" set.

spatial and/or intensity resolution often yields more (though not dramatically more) local maxima than we saw in difference surfaces formed with unreduced images.

Table 4.4 gives the mean number of local maxima for all spatial and intensity reductions for difference surfaces in which current images were taken from the "Chairs" set and snapshot images were taken from the "Original" set. There seems to be no discernible relationship between number of intensity levels and number local maxima. As spatial resolution decreases, number of local maxima tend to increase as in Tables 4.2 and 4.3; again this increase does not seem dramatic.

We next looked at the standard deviation of angular discrepancy between difference surface gradient and true home direction. Figure 4.4 illustrates this divergence for snapshot and current images taken from the "Original" data set. Images were scaled down by 50% and various gray level reductions were used. The curve in blue illustrates

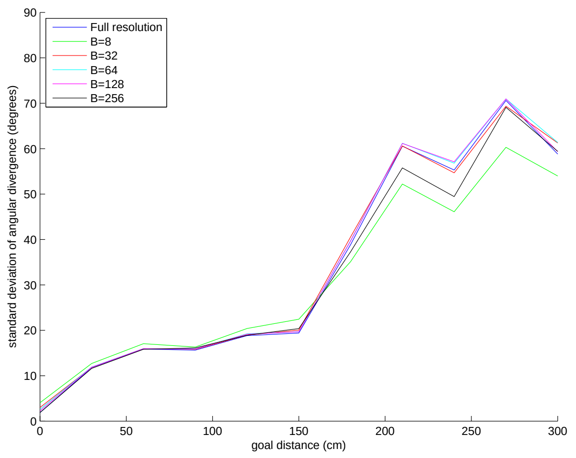


Figure 4.4: Angular discrepancy between difference surface gradient and true home direction as a function of goal distance. Snapshot and current images were taken from the “Original” data set. Images were scaled down by 50% and various gray level reductions were used. The curve in blue illustrates the standard deviation of divergence for difference surfaces produced from unreduced images.

the standard deviation of divergence for difference surfaces produced from unreduced images. As can be seen, there is no dramatic difference in this criteria for the various gray level reductions shown. Spatial reductions of images to 25%, 20% and 17% of original size yield almost identical results.

When current images are taken from either the “Winlit” or “Chairs” data set, we see largely the same results as reported in the previous paragraph. We do note, though, that the maximal and minimal gray level settings tend to produce slightly worse behaviour when spatial reduction is fairly large, as seen in Figure 4.5.

As with the divergence criterion, the “good moves” criterion (defined in Section 4.4.1.4) exhibits little sensitivity to spatial and/or gray level reduction in static environments. So too, in dynamic scenes the “good moves” criterion slightly favours mid-level gray level reductions for large spatial reductions.

We close this section showing a few representative difference surfaces. Rather than present fully three-dimensional difference surfaces (e.g. Figure 4.6(a)) in this section, we represent each difference surface with three transects (e.g. Figures 4.6(b), 4.6(c) and 4.6(d)). Each transect intersects the goal location; the first (Figure 4.6(b)) has slope infinity in the coordinate system of Figure 4.3; the second (Figure 4.6(c)) has slope 1; and the third (Figure 4.6(d)) has slope -1. This kind of difference surface plot makes it easier to see the effects of gray level and/or spatial reduction. The negative

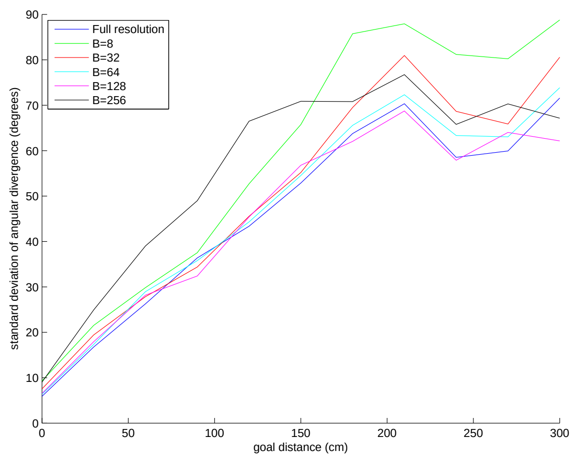
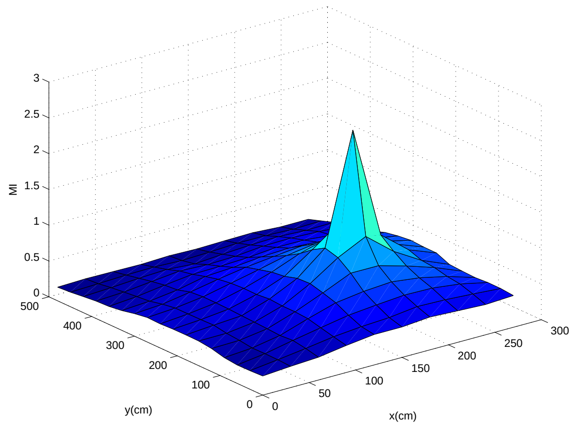


Figure 4.5: Angular discrepancy between difference surface gradient and true home direction as a function of goal distance. Snapshot images were taken from the "Original" data set and current images were taken from the "Winlit" data set. Images were scaled down to 25% of original size and various gray level reductions were used. The curve in blue illustrates the standard deviation of divergence for difference surfaces produced from unreduced images.

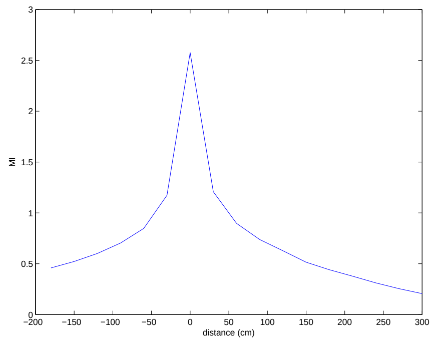
distances in the transect plots indicate positions with x-coordinates less than that of the goal location.

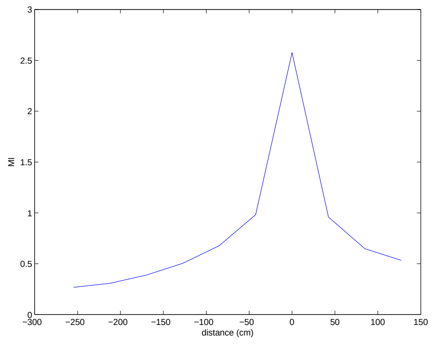
Figure 4.7 depicts six difference surfaces captured at a single snapshot location (x=180cm, y=180cm); both snapshot and current images were taken from the "Original" data set. Figures 4.7(a), 4.7(b) and 4.7(c) each display a single transect from each of the six difference surfaces; the transects are defined as in Figure 4.6. The difference surface in blue was created with images whose spatial and intensity resolution was unreduced; we provide this for comparison. The four other difference surfaces – in green, red, cyan, and magenta – used images which were reduced to – respectively – 128, 64, 32, and 8 gray levels (spatial resolution was unchanged). As can be seen in the figure, reduction of gray levels results in a scaling-down of MI surface values near the snapshot location and a constant shift of MI surface values relatively far from the snapshot location. We see qualitatively similar results at other snapshot locations, for other spatial resolutions, and in the dynamic environments we experimented with.

Figure 4.8 depicts six difference surfaces captured at a single snapshot location (x=180cm, y=180cm); both snapshot and current images were taken from the "Original" data set. Figures 4.8(a), 4.8(b) and 4.8(c) each display a single transect from each of the six difference surfaces; the transects are defined as in Figure 4.6. The difference

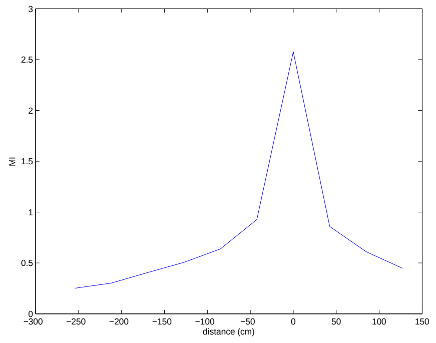


(a)





(b) (c)



(d)

Figure 4.6: Difference surfaces Section 4.5 are presented not as fully three-dimensional (e.g. **a**). We rather depict three transects (e.g. **b**, **c** and **d**), each intersecting the goal location. See text for more details.

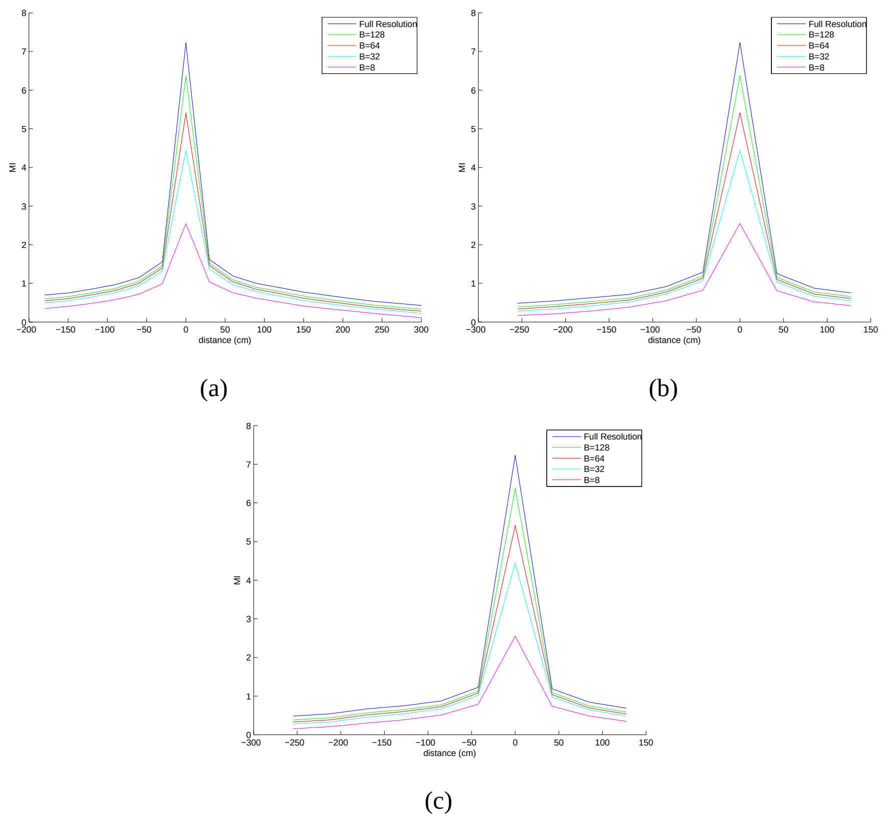
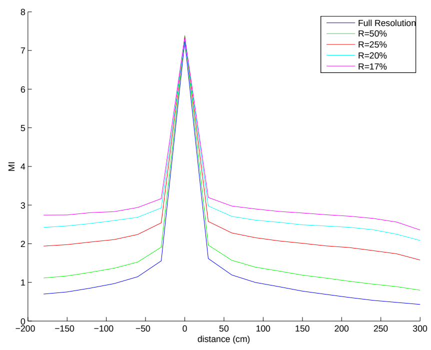
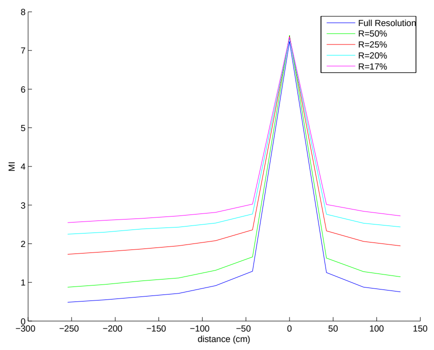


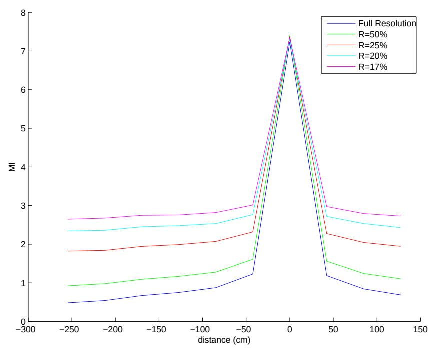
Figure 4.7: Transects of six difference surfaces produced at snapshot location x=180cm, y=180cm. Snapshot and current images were taken from the “Original” data set. See text for details of how the six difference surfaces were created.



(a)



(b)



(c)

Figure 4.8: Transects of six difference surfaces produced at snapshot location x=180cm, y=180cm. Snapshot and current images were taken from the “Original” data set. See text for details of how the six difference surfaces were created.

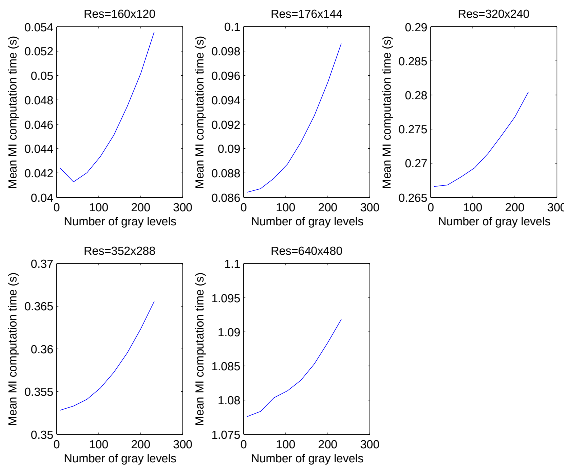


Figure 4.9: Mean times for serial MI computation on our laptop for various spatial and gray scale resolutions. Note that the vertical scale on each subplot is different. See Section 4.5.2 for details.

surface in blue was created with images whose spatial and intensity resolution was unreduced; we provide this for comparison. The four other difference surfaces – in green, red, cyan, and magenta – used images which were scaled down to – respectively – 50%, 25%, 20%, and 17% of their original size (number of gray levels remained constant at 256). As can be seen in the figure, spatial resolution reduction leaves the MI value at the snapshot location almost unchanged. There is a constant shift of MI surface values relatively far from the snapshot location. We see qualitatively similar results at other snapshot locations, for other spatial resolutions, and in the dynamic environments we experimented with.

4.5.2 Timing Experiments

Figure 4.9 shows the mean time (over 20 trials) required to compute MI on the Acer laptop for various spatial resolutions and gray levels. The standard deviation of computation times on the laptop is exceedingly small, so we do not show it.

It proved difficult to alter the number of gray levels used in parallel computation of MI on the EyeRIS. Given the results of the previous section, we fixed the number of gray levels at 16 for our parallel timing experiment. Mean MI computation time over 20 trials with 16 gray levels was 0.098 seconds with a standard deviation of 0.0656 seconds. Image resolution was 128*x*128 pixels, the native size of all EyeRIS images. This mean time is approximately equal to the mean computation time on the laptop at

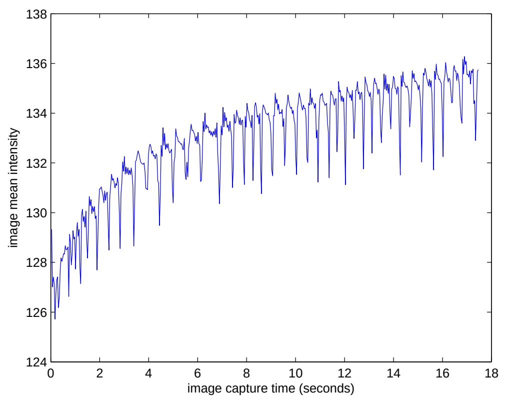


Figure 4.10: Mean intensity of images of constant well-lit scene captured over time with the EyeRIS system.

the lowest spatial resolution.

4.5.3 Noise in Image Capture on EyeRIS

As the EyeRIS system is in a fairly early stage of development, we wanted to determine whether it produces any unreported bugs when capturing images and computing mutual information. In an early test, we programmed the EyeRIS system to capture 500 images, one every 35 ms (the default exposure time); all overhead lights were turned on. A troubling trend soon became apparent. As depicted in Figure 4.10, the mean image intensity over time increases approximately logarithmically, with periodic sharp dips. This contrasts with Webcam images captured over time, which are corrupted by zero mean Gaussian noise.

We looked at the values of individual pixels in successive EyeRIS images and found that the dips evident in Figure 4.10 are caused by a reduction in the majority of image pixels. This decrease is invariably followed by an increase of pixel value in one of the next few images. The mean intensity increase is usually slightly higher than the preceding mean intensity decrease. Between dips, successive images differ by noise with a slightly positive mean and approximately Gaussian distribution. We have brought this problem to the manufacturer of the EyeRIS but they have as yet offered no solution.

Our laboratory is lit by flourescent lights. It may be the case that the dips in mean image intensity in Figure 4.10 are caused by the flicker – the periodic dimming – of these lamps. The flicker of computer monitors near the EyeRIS may contribute to this phenomenon as well. To determine whether the dips are caused by these light sources,

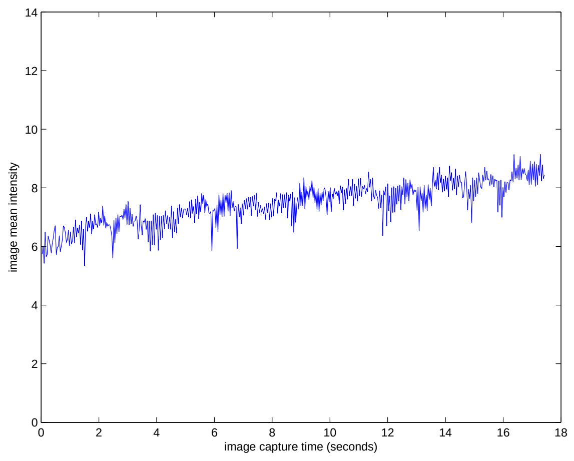


Figure 4.11: Mean intensity of images of constant scene captured over time with the EyeRIS system. All flourescent lights and nearby computer monitors in the laboratory were turned off.

we as before programmed the EyeRIS system to capture 500 images, one every 35 ms. This time, though, the images were taken with all flourescent lights and nearby computer monitors turned off. The only illumination came from natural light flowing through our laboratory's windows. Figure 4.11 depicts the mean intensity of the 500 images we captured successively in this experiment. We have adjusted the range of the y-axis of Figure 4.11 to cover fourteen gray levels, more than are needed to display the data; this is to make comparison with Figure 4.10 – whose y-axis also spans fourteen gray levels – easier. The mean image intensities depicted in Figure 4.11 are smaller than those in Figure 4.10 since the images used to create the former figure were taken in much darker conditions. About ten sudden sharp dips in mean intensity level are evident in Figure 4.11, far fewer than the number of dips in Figure 4.10. The majority of dips in Figure 4.10 represent a change in mean intensity of about three gray levels whereas the dips in Figure 4.11 are generally of smaller magnitude – about one gray level. Interestingly, there are about ten dips in Figure 4.10 which represent a change in mean intensity of about one gray level, matching the pattern seen in Figure 4.11. Our conclusion is that the lab's flourescent lighting causes the majority of dips in mean image intensity seen in Figure 4.10. There may be another periodically flickering source of illumination – dimmer than the flourescents and as yet unidentified – in our laboratory which was still active when capturing the images used to make Figure 4.11.

An important question is this: does the noise described above disrupt MI difference surfaces, making them difficult to optimise? In order to find out, we performed the following experiment: We first computed a difference surface transect using images from Vardy's "Original" data set. The snapshot was the image at x=150 cm, y=0 cm. We then simulated a homing run along a line to this snapshot location, beginning at x=150 cm, y=330 cm. As expected, this homing run produces a graph in which MI increases monotonically with decreasing goal distance, reaching a maximum at the goal (see Figure 4.12, dashed line).

In order to determine the effect of EyeRIS noise on the MI signal, we computed the difference surfaces transect described above but this time corrupted Vardy's images with this noise. We corrupted each image by adding a random integer (allowed to be position or negative) to each pixel value in the image. The noise value for each pixel was drawn randomly from a probability density function generated from some of the 500 noisy EyeRIS images we described above. To generate this probability density function we assumed that the Vardy images along the transect were captured at 1 second intervals. We also assumed, somewhat arbitrarily, that movement along the transect began 5 seconds after starting to acquire EyeRIS images. We identified the EyeRIS image whose capture time corresponded best with the assigned capture time of the current Vardy image. The EyeRIS images were captured every 35ms. Thus, the second Vardy image in the transect, for example, was matched with the 172nd EyeRIS image and the third Vardy image in the transect was matched with the 201st EyeRIS image. To compute the noise probability density function for the third Vardy image, we subtracted the 172nd EyeRIS image from the 201st and created a histogram from the pixel differences. The noise probability density functions for the other Vardy images in the transect were computed similarly. The difference surface transect which resulted from the corrupted Vardy images is shown in Figure 4.12 (solid line). The MI signal generated from corrupted images exhibits a much less pronounced maximum at the goal location than does the uncorrupted MI signal. We consider this MI signal, in fact, to be unusable for homing purposes. We repeated this experiment with several (ten) combinations of snapshot location and starting location using Vardy's "Original" data set. The result in each case was qualitatively similar to that described above.

4.6 Discussion and Conclusions

The goal of this chapter was to provide methods to speed the computation of mutual image information without reducing the effectiveness of difference surface-based visual homing. We proposed a novel method of computing mutual image information

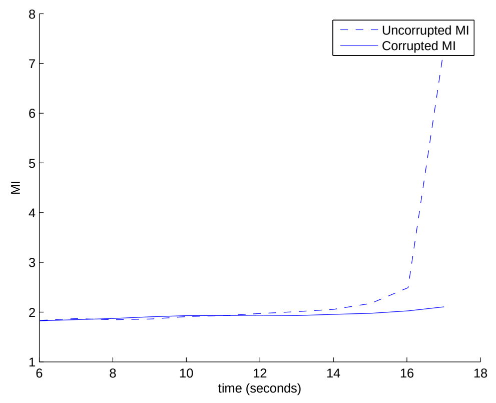


Figure 4.12: Solid line: transect of homing run using images from Vardy's "Original" data set. Dashed line: transect homing run using same images as before, but corrupted by noise of the type seen in Figure 4.10.

with the help of a parallel image processing device: the AnaFocus EyeRIS Vision System. We showed that this algorithm computes mutual image information in $O(B^2)$ operations, where B is the number of intensity levels in the input images. We then demonstrated that reduction of gray levels (to a minimum of 8) does not have a negative impact on homing performance in a laboratory environment in both static and dynamic conditions.

Unfortunately, it is clear from the work described in Section 4.5.3 that the noise in the EyeRIS's image capture process renders this device unusable for our homing studies. We demonstrated that this noise is in part due to the fluorescent lighting used to illuminate our laboratory environment.

As the EyeRIS system proved to be non-viable for our purposes, we attempted to speed the serial computation of mutual information. We did so by reducing the spatial resolution and/or number of intensity levels in snapshot and current images. Serial computation of mutual image information takes $O(B^2 + NM)$ steps where each input image has N rows and M columns. We demonstrated that a reduction in spatial and/or intensity levels in our input image has little discernible negative effect on homing performance in a laboratory environment in both static and dynamic conditions. We considered quite a large range of spatial and intensity level reductions, as described in Section 4.4.

We compared the time required to compute mutual information with parallel and serial algorithms in Section 4.5.2. We computed mutual information using the Eye-

RIS using input images of 128*x*128 pixels subsampled to 16 gray levels. On average computation took 0.098 seconds. Serial computation with slightly larger images with 16 gray levels took on average about 0.042 seconds, less than half the time. The serial setup is therefore faster than the parallel (at least with input images reduced as described above) and is not subject to the debilitating image capture noise described in Section 4.5.3. In "live" homing experiments, therefore, we feel justified in using our laptop (or when possible a faster desktop computer) – rather than the EyeRIS – to compute mutual image information with reduced images.

But what image size and gray level resolution should we use when computing mutual image information serially? There is no one combination of image size and gray level cardinality that yields consistently superior performance by any of the measures of comparison considered in Section 4.4.1. The results reported in Section 4.4.1 lead us to conclude that for difference surface homing in static and dynamic conditions, images with a spatial resolution of 176*x*144 pixels (20 percent of the original size of the Vardy images) downsampled to 128 gray levels will consistently yield relatively good homing performance. Mutual information with images of this size will take on average 0.09 seconds to compute serially using our laptop, approximately the same amount of time required by the parallel algorithm (albeit with smaller images and fewer gray levels).

We believe that homing with reduced images is successful due to the effect that such reduction has on the mutual information signal. We saw in Figure 4.7 that reduction of gray levels results in a scaling-down of MI surface values near the snapshot location and a constant shift of MI surface values relatively far from the snapshot location. Even for quite drastic reduction in the number of gray levels the difference surface retains a global maximum at the snapshot location. For a location somewhat distant from the snapshot location, the gradient of the difference surface for one gray level setting is approximately equal to the gradient of the surface at the same location for another gray level setting. These qualities lead difference surface homing results which are largely unaffected by the number of gray levels in input images.

The observations detailed above are based on experimental results. We would like to provide some analytical support for these results. We first look at the change in MI due to gray level reduction when I_C is captured at the snapshot location. If I_C is identical to I_S at this location (i.e. no image capture noise, environment is static), then $MI(I_S, I_C) = MI(I_S, I_S)$. It follows from Equation 4.1 that

$$MI(I_S, I_S) = -\sum_{a=0}^{B-1} p_S(a) lg(p_S(a)) = H(I_S) \tag{4.2}$$

where $p_S(a)$ is the probability that a pixel will have intensity a ($0 \leq a < B$) in image I_S; lg is shorthand for a logarithm with base 2. Values of a for which $p_S(a) = 0$ are ignored in the calculation of Equation 4.2. As indicated, $MI(I_S, I_S)$ is simply the entropy of I_S (denoted $H(I_S)$). In the remainder of this discussion, we shall drop the subscripts attached to probabilities (e.g. $p_S(a)$ becomes $p(a)$).

When the number of gray levels in I_S is halved, $H(I_S)$ becomes

$$H_{reduced}(I_S) = -\sum_{\substack{a=0 \\ by2}}^{B-1} [p(a) + p(a+1)]\, lg\, [(p(a) + p(a+1)] \tag{4.3}$$

which can be rewritten as

$$H_{reduced}(I_S) = -\sum_{\substack{a=0 \\ by2}}^{B-1} p(a) lg\, [(p(a) + p(a+1)] + p(a+1) lg\, [(p(a) + p(a+1)] \tag{4.4}$$

So too, Equation 4.2 can be rewritten as

$$H(I_S) = -\sum_{\substack{a=0 \\ by2}}^{B-1} [p(a) lg(p(a)) + p(a+1) lg(p(a+1))] \tag{4.5}$$

Each term in $-p(a) lg\, [(p(a) + p(a+1)]$ in Equation 4.4 corresponds to a single term $-p(a) lg(p(a))$ in Equation 4.5. Since $0 < p(a), p(a+1) < 1$, $-p(a) lg\, [(p(a) + p(a+1)] < -p(a) lg(p(a))$ for all a. Thus, as we saw in Figure 4.7, gray level reduction causes a decrease in MI value at the snapshot location. The scale of MI reduction is dependent on the distribution of intensities in I_S. If, for example, the distribution of intensities is uniform, then $p(a) = \frac{1}{B}$ for all a. Equation 4.5 becomes

$$H(I_S) = -\sum_{\substack{a=0 \\ by2}}^{B-1} \left[\frac{1}{B} lg \frac{1}{B} + \frac{1}{B} lg \frac{1}{B} \right] = -\frac{B}{2} \frac{2}{B} lg \left[\frac{1}{B} \right] = -lg \left[\frac{1}{B} \right] = lg(B) \tag{4.6}$$

and Equation 4.3 becomes

$$H_{reduced}(I_S) = -\sum_{\substack{a=0 \\ by2}}^{B-1} \frac{2}{B} lg \left[\frac{2}{B} \right] = -\frac{B}{2} \frac{2}{B} lg \left[\frac{2}{B} \right] = -lg \left[\frac{2}{B} \right] = lg(B) - 1 \tag{4.7}$$

Thus, when I_S has a uniform intensity distribution, halving the number of intensity levels leads to a reduction of 1 in the MI signal at the snapshot location.

We saw in Figure 4.7 that reduction of gray levels results in a constant shift of MI surface values relatively far from the snapshot location. We shall provide some analytical support for this observation. We define four mutual image information readings:

- $MI_1 = MI(I_S, I_{C1})$ is the mutual information reading at a position $\vec{x}_1$ relatively far from the snapshot location; I_{C1} denotes the image captured at $\vec{x}_1$ and – as usual – I_S is the image captured at the snapshot location.
- $MI_2 = MI(I_S, I_{C2})$ is the mutual information reading at a position $\vec{x}_2$ close to $\vec{x}_1$; I_{C2} denotes the image captured at $\vec{x}_2$.
- $MI_{1,reduced}$ is the mutual information reading at $\vec{x}_1$ with the number of gray levels in I_S and I_{C1} cut in half.
- $MI_{2,reduced}$ is the mutual information reading at $\vec{x}_2$ with the number of gray levels in I_S and I_{C2} cut in half.

To support our empirical findings, we would like to show that $MI_1 - MI_{1,reduced} \approx MI_2 - MI_{2,reduced}$.

It is convenient to express mutual image information in the form (from Hill et al. [2001])

$$MI(I_S, I_C) = H(I_S) + H(I_C) - H(I_S, I_C) \tag{4.8}$$

where $H(I_C)$ is the entropy of I_C and $H(I_S, I_C)$ is the joint entropy between the two images (which we shall define later).

Equation 4.8 makes clear that demonstrating that $MI_1 - MI_{1,reduced} \approx MI_2 - MI_{2,reduced}$ is equivalent to showing that

$$\begin{aligned}&[H(I_S) + H(I_{C1}) - H(I_S, I_{C1})] - \left[H(I_{S,reduced}) + H(I_{C1,reduced}) - H(I_{S,reduced}, I_{C1,reduced})\right] \approx \\ &\quad [H(I_S) + H(I_{C2}) - H(I_S, I_{C2})] - \left[H(I_{S,reduced}) + H(I_{C2,reduced}) - H(I_{S,reduced}, I_{C2,reduced})\right]\end{aligned} \tag{4.9}$$

We shall assume that, since $\vec{x}_2$ near to $\vec{x}_1$, $H(I_{C1}) = H(I_{C2})$ and $H(I_{C1,reduced}) = H(I_{C2,reduced})$. This assumption is valid if neither $\vec{x}_2$ nor $\vec{x}_1$ is close to an imaged object and the

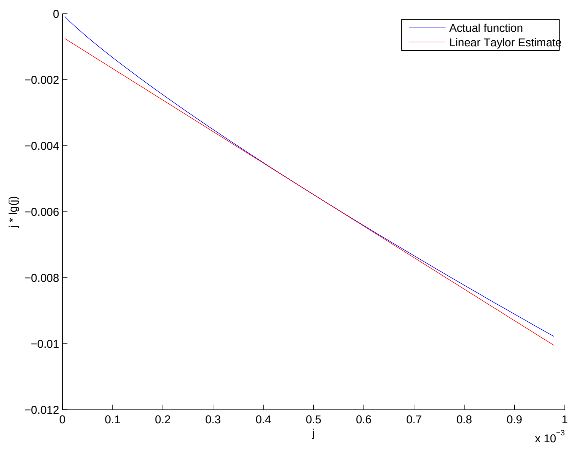


Figure 4.13: Graph (in blue) of j versus $j \cdot lg j$ for small positive values of j. The linear Taylor series estimate of $j \cdot lg j$ is shown in red.

visual environment is static. Given these assumptions, in order to show that $MI_1 - MI_{1,reduced} \approx MI_2 - MI_{2,reduced}$ it suffices to demonstrate that

$$-H(I_S, I_{C1}) + H(I_{S,reduced}, I_{C1,reduced}) \approx -H(I_S, I_{C2}) + H(I_{S,reduced}, I_{C2,reduced}) \quad (4.10)$$

Joint entropy is defined as

$$H(I_S, I_C) = -\sum_{a=0}^{B-1} \sum_{b=0}^{B-1} p(a,b) lg\left(p(a,b)\right) \quad (4.11)$$

where $p(a,b)$ is the probability that a given pixel in I_S has intensity a and the same pixel in I_C has value b. Terms with $p(a,b) = 0$ are ignored in the calculation of the joint entropy.

When the number of gray levels in I_S and I_C is halved, Equation 4.11 becomes

$$H(I_{S,reduced}, I_{C,reduced}) = -\sum_{\substack{a=0 \\ by2}}^{B-1} \sum_{\substack{b=0 \\ by2}}^{B-1} [p(a,b) + p(a,b+1) + p(a+1,b) + p(a+1,b+1)] \cdot \\ lg\left[p(a,b) + p(a,b+1) + p(a+1,b) + p(a+1,b+1)\right] \quad (4.12)$$

We found it difficult to show that Equation 4.10 is true using joint entropy as expressed in Equations 4.11 and 4.12. Equations 4.11 and 4.12 consist of a number of terms of the form $j \cdot lg j$, where j is a particular joint probability. We shall replace each of these terms with its Taylor polynomial approximation. According to Taylor's theorem (Burden and Faires [1993]), any function which is n-times differentiable can be approximated by a polynomial of (up to) degree n whose terms are dictated by the theorem. The Taylor polynomial is designed to coincide with the approximated function for one value η of the function's dependent variables. The approximation becomes less exact as the distance from η increases. Since $\vec{x}$ is far from the snapshot location, each joint probability j will be a positive real number very near zero (j could in fact be equal to zero but these values are ignored in the computation of joint entropy). As can be seen in Figure 4.13, the graph of $j \cdot lg j$ is almost linear when j is near zero (a typical range of joint probability values was used to create this graph). Thus, we feel justified in using a Taylor polynomial of degree one to approximate $j \cdot lg j$; this Taylor polynomial is

$$j \cdot lg j \approx \eta lg(\eta) + lg(e\eta)(j - \eta) \tag{4.13}$$

where e is the base of the natural logarithm and η is a small positive real number. We shall use the following more convenient form of Equation 4.13

$$j \cdot lg j \approx j lg(e\eta) - \eta lg(e) \tag{4.14}$$

Using Equation 4.14 we can rewrite Equation 4.11:

$$\begin{aligned} H(I_S, I_C) &\approx -\sum_{a=0}^{B-1}\sum_{b=0}^{B-1}[p(a,b) lg(e\eta) - \eta lg(e)] \\ &\approx -lg(e\eta)\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} p(a,b) + \eta lg(e)\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} 1 \end{aligned} \tag{4.15}$$

Of course, $\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} p(a,b) = 1$. According to the definition of joint entropy given earlier in this section, terms of Equation 4.15 with $p(a,b) = 0$ are ignored in the calculation of the joint entropy. Thus, $\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} 1$ is a count of the number of non-zero values of $p(a,b)$ for all a and b. As $p(a,b)$ will sometimes be zero for particular values of a and b, this count is sometimes less than B^2. We shall let $\sum_{a=0}^{B-1}\sum_{b=0}^{B-1} 1$ equal K where $K \leq B^2$. Thus Equation 4.15 becomes

$$H(I_S, I_C) \approx -lg(e\eta) + K\eta lg(e) \tag{4.16}$$

We can substitute the right-hand side of Equation 4.16 into Equation 4.10:

$$-K_1 \eta lg(e) + K_{1,reduced} \eta lg(e) \approx -K_2 \eta lg(e) + K_{2,reduced} \eta lg(e) \tag{4.17}$$

This equality will hold if $K_{1,reduced} - K_1 \approx K_{2,reduced} - K_2$; we see that empirically this is often the case. For example, in one instance using Vardy's data set, $K_{1,reduced} - K_1 = -18188$ and $K_{2,reduced} - K_2 = -18275$; in this case $\vec{x_1}$ and $\vec{x_2}$ were separated by 30cm. We saw quantitatively similar results for other samples of Vardy's data set.

We showed in this chapter that we could speed the computation of mutual information without reduction in the success of difference surface homing. It could be the case, though, that the time taken to home is dominated by robot movement rather than difference surface evaluation. In the next chapter we carry out detailed simulations of difference surface homing with various optimisation algorithms; mutual information is used to compute image similarity. The results of about 5000 simulations in static and dynamic conditions indicate that about 2.3 percent of homing time is on average spent doing difference surface evaluations when using the gradient-based optimisation algorithm employed in the previous chapter. The robot spends the rest of the time making pose changes. Other optimisation algorithms (both gradient-based and not) yield similar results.

The work done in this chapter is still of some value, though. The simulated robot was made to move rather slowly (8.0 cm per sec) on the assumption that in our live experiments the robot will move slowly in order to avoid wheel slippage. If the robot is made to move faster, function evaluation time plays a larger role in overall homing time. Also, speedy image histogram and mutual information computation is of general interest as these algorithms are widely applicable in image processing tasks (see e.g. Shahbahrami et al. [2008] and Li [2005]).

4.7 Future Work

The obvious next step in using the EyeRIS is to eliminate the noise in image capture depicted in Figure 4.10. We have contacted the manufacturer to make them aware of the problem. They have as yet offered no solution.

Authors have recently reported attempts to compute image histograms (see e.g. Scheuermann and Hensley [2007]) and mutual image information (see e.g. Shams and Barnes [2007]) with graphics processing units (GPUs). GPUs are chips which were developed to facilitate the transformation and rendering of computer graphics primitives like polygons on personal computers. The development of the GPU has been driven by the flourishing computer gaming industry whose products are of course often extremely graphics-intense. GPUs transform and render quickly by performing common vector operations in parallel. As many algorithms, not just those in computer graphics, rely on vector operations, researchers have in the last few years seen the potential for extremely fast general-purpose computation with GPUs (GPGPU) beyond mere rendering (Luebke et al. [2006]).

The main problem with mutual information calculation with a GPU is in image histogramming. Shams and Barnes [2007] show that histogramming runs into trouble when two or more different GPU subprocessors, working in parallel, attempt to increment the same histogram bin at the same time. These authors overcome this problem by dividing the GPU into blocks of subprocessors. Each block is devoted to computing the joint histogram of a unique subset of the input images. Each subprocessor in a particular block increments a unique subset of histogram bins. Once each block has computed the joint histogram for the subset of the input for which it is responsible, the joint histograms are combined to produce the complete joint histogram for the input images. This melding process is described by Shams and Barnes [2007] as efficient but is not laid out in detail.

A field-programmable gate array (FPGA) is a device consisting of thousands of reconfigurable hardware logic blocks (Li [2005]). Each logic block can be configured by a programmer to execute a relatively simple function (i.e. a logic gate, decoder, etc.). The programmer can also define the connections between logic blocks. In this way, an FPGA can be used to create complex integrated circuits by a programmer "in the field" rather than by a chip manufacturer.

FPGAs have in the past few years received interest from researchers in image processing and vision-based robotics (see e.g. Draper et al. [2000], Shahbahrami et al. [2008] and Anderson et al. [2005]). With their array of logic blocks, FPGAs lend themselves to parallel implementations image processing routines. Efforts have been made to parallelise the computation of image histograms using FPGAs (see e.g. Li [2005] and Shahbahrami et al. [2008]). Histogram computation is, as we have previously explained, a crucial step in the calculation of mutual image information. Shahbahrami et al. [2008] note that the major challenge in the parallel computation of an image histogram with an FPGA is memory collision. That is, if all image pixels are read simultaneously to compute a histogram, it is likely that at least two pixels will have the same intensity value. Each subprocessor handling this pixel value will try to increment the associated histogram bin at the same time. Shahbahrami et al. [2008] solve this problem by dividing images in half and computing in serial the histogram of the half-images at the same time on separate processors.

It would be interesting to pit our parallel algorithm for mutual information computation against an FPGA-based or GPU-based MI computation. The FPGA-based MI computation in Li [2005] is tested on 3D images so we cannot easily compare the reported timing results to ours. Nor can we compare our results directly with those reported by Shams and Barnes [2007] as these authors do not report the time required for mutual information computation of individual images. Thus, we would likely have to implement the mutual image information algorithms described in Li [2005] and Shams and Barnes [2007] before a comparison with our work could be made.

Other parallel image histogram algorithms exist, but are designed for systems different than the EyeRIS and somewhat unlike FPGAs. Jenq and Sahni [1992] describes an efficient algorithm to be used on a reconfigurable mesh parallel processor. Like the EyeRIS's FPP, a reconfigurable mesh consists of a grid of locally connected processors. As the name suggests, the connections of the reconfigurable mesh can be opened or closed during program execution. Given an $N \times N$ image with B gray levels, the histogram is computed in $O(\sqrt{B}\, log_{\sqrt{B}}(\frac{N}{\sqrt{B}}))$ steps if $B < N$ and $O(\sqrt{B})$ for $B \geq N$.

4.8 Related Work

After creating our parallel single-image histogramming algorithm (Algorithm 3), we found a similar algorithm in Braunl et al. [2001]. The algorithms differ in how band-pass intensity images are created for each intensity and in how the number of band-passed pixels are counted. We could find no parallel joint histogramming algorithm in the literature.

We could find only one instance of visual homing with parallel computation (Möller [2000]). In this work, Mőller describes a purpose-built analog circuit which computes a homing vector with the average landmark vector scheme. He demonstrates that the system works in simple arena consisting of black cylinders on a white background. The system was not tested in more complex environments.

Chapter 5

Optimising the Difference Surface

5.1 Introduction

In Chapter 2, we described two methods which Zeil et al. [2003] used to move the homing agent so as to optimise the difference surface: "Run-Down" and "Triangular." These may not be the best solutions to this problem. This chapter investigates ways of optimising the difference surface in order to home. Unlike Zeil et al. [2003] (and to our knowledge all other visual homing researchers), we consider visual homing under the influence of realistic sensor noise.

5.2 Problem Definition

We assume that the agent is travelling on a planar surface while homing. Without loss of generality, we define the starting point of the agent's homing run to be the origin of a local two-dimensional Cartesian coordinate system. The agent's initial orientation defines the x-axis of this coordinate system; the y-axis is in the orthogonal direction in the plane on which the agent travels.

With this coordinate system in mind, we can make the following useful definitions. We define $I(\vec{x})$ to be the panoramic intensity image capturable at position $\vec{x} = (x, y)$. The value of the difference surface at $\vec{x}$ – the signal measured by the homing agent – will be defined to be $f(\vec{x})$. The function f is synonymous with what we have called the difference surface in previous chapters.

The process of optimisation, as we shall see in the following sections, is an iterative one. The agent moves through a sequence of points between its starting position (at the origin of our coordinate system) and the snapshot location. We label the k_{th} point in this sequence $\vec{x}_k$ $(1 \leq k \leq N)$. We let $\vec{x}_S$ be the location of the snapshot.

The problem is to move the agent to search for global optimum $\vec{x}^*$ of f. We assume that $\vec{x}^*$ is equal to $\vec{x}_S$; the results reported in Chapter 3 indicate that this assumption is valid quite often in both static and dynamic environments for difference surfaces formed using the mutual information image similarity measure. To be precise, this assumption held true for 291 of the 304 *MI* difference surfaces (or 96%) used in the experiments in Chapter 3. The 13 difference surfaces which did not meet this assumption were formed with dramatic lighting change between snapshot and current images (i.e. the snapshot image was drawn from the "Winlit" set and current images were drawn from the "Doorlit" set or vice-versa).

5.3 Optimisation Algorithms

The literature provides many algorithms to optimise a function (see e.g. Adby and Dempster [1974]). Some are designed for linear functions, others for non-linear, some work only for functions with integer domains, others for real-value inputs. Which algorithms are right for us? Our optimisation problem has a number of qualities which will help us narrow down the search for an appropriate algorithm.

1. Our moving agent can use path integration and/or dead-reckoning to estimate $\vec{x}_k$. These techniques suffer from cumulative errors. Thus as homing proceeds, our agent will have an increasingly vague estimate of $\vec{x}_k$.

2. Unlike many optimisation problems found in the literature, in order to measure f at two distinct points $\vec{x}$ and $\vec{x}'$, the robot must travel from $\vec{x}$ to $\vec{x}'$.

3. Since the robot we use is non-holonomic, it must rotate in order to change its direction of travel. This rotation of course takes time.

4. Rotation and translation of the robot are noisy processes. That is, the robot executes each requested motor command with some level of imprecision.

5. Given our empirical study of the difference surface in previous chapters, f is clearly nonlinear. Unfortunately, we do not know the functional form of f. Note that in prior published work we have made some attempt to identify this form, at least in the case of *RMS* difference surfaces (see Szenher [2005b]).

6. The evaluation of the f at $\vec{x}$ is corrupted by sensor noise, as we will show in Sections 5.4.1 and 5.4.2.

7. The gradient of f at $\vec{x}$ is not directly available. If $f(\vec{x})$ were a known function – e.g. $3x^2 + 6x + 4y^3$ – then we could compute its gradient; the parameterised gradient of our example function is $[6x+6 \;\; 12y^2]^T$. If we knew that $x = 4$ and $y = 5$, we could compute the gradient at this point to be $[30 \;\; 300]^T$. Unfortunately, while homing we know neither the precise form of f nor the value of $\vec{x}$.

Given the considerations in the above list, as well as the problem statement in Section 5.2, the literature indicates that stochastic optimisation algorithms are most appropriate for the problem at hand. According to Spall [2003], stochastic optimisation algorithms apply when there is noise in the measurement of the function to be optimised and/or the direction in which to search during optimisation is (at least sometimes) chosen with some randomness. Both are certainly the case here (see list items (4) and (6) above). As we shall see there exist some useful so-called direct search stochastic optimisers which require no function gradient information. Other stochastic optimisers provide ways of approximating the gradient. Below, we describe the stochastic optimisers suggested by Spall [2003] which we have chosen to use.

5.3.1 Stochastic Optimisation

The gradient of a function at a point is a vector pointing in the direction of greatest function increase at that point. If we knew the gradient of f at $\vec{x_k}$ we could home by moving in the direction of that gradient for a certain distance, reassess the gradient at the new location and continue in this manner until home. As discussed above, we cannot directly measure the gradient f at a given location. Spall [2003] suggests a number of methods to estimate the local gradient from a few local function measurements.

The such first method – two-sided finite differencing (2FDSA) – is defined by the following equation:

$$\vec{g}(x_k, y_k) = \begin{bmatrix} \frac{f(x_k+c_k, y_k) - f(x_k-c_k, y_k)}{2c_k} \\ \frac{f(x_k, y_k+c_k) - f(x_k, y_k-c_k)}{2c_k} \end{bmatrix} \tag{5.1}$$

$\vec{g}(x_k, y_k)$ denotes the estimate of the gradient of f (i.e. the difference surface) at the k^{th} step in the homing algorithm. To evaluate Equation 5.1, the value of f at four points – $(x_k + c_k, y_k)$, $(x_k - c_k, y_k)$, $(x_k, y_k + c_k)$ and $(x_k, y_k - c_k)$ – must be computed. The

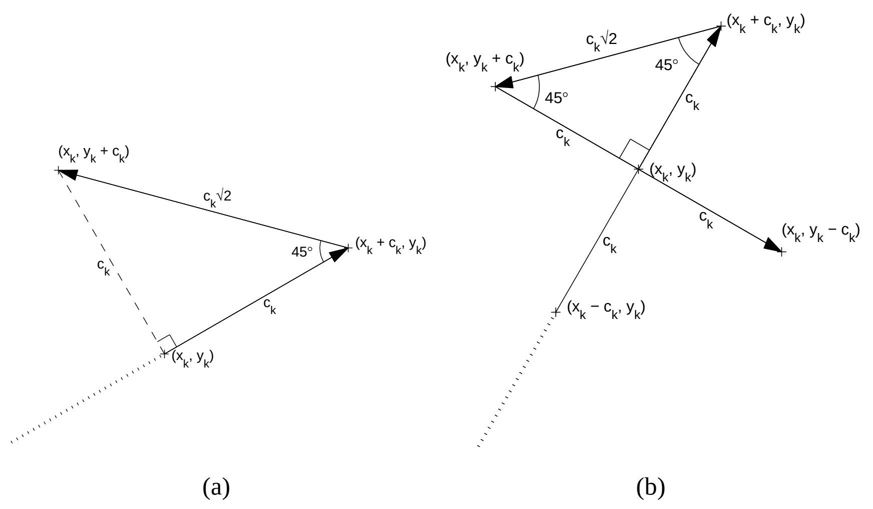


Figure 5.1: **(a)** A skematic of the moves made by our model agent to gather the information required to calculated Equation 5.2. The moves required to calculate Equation 5.1 are given in **(b)**.

points $(x_k + c_k, y_k)$ and $(x_k - c_k, y_k)$ form a line segment of length $2c_k$ with (x_k, y_k) at its midpoint. The value we assign c_k will be discussed below. This line segment can be oriented in any direction in the plane on which the agent is moving. The points $(x_k, y_k + c_k)$ and $(x_k, y_k - c_k)$ also form a line segment of length $2c_k$ with (x_k, y_k) at its midpoint. This line segment must be orthogonal to the first. The geometric relationship between these four points is depicted in Figure 5.1(b). The x-component of $\vec{g}(x_k, y_k)$ is proportional to the difference between the value of f at $(x_k + c_k, y_k)$ and $(x_k - c_k, y_k)$. The y-component of $\vec{g}(x_k, y_k)$ is proportional to the difference between the value of f at $(x_k, y_k + c_k)$ and $(x_k, y_k - c_k)$. Note that the estimate of the gradient of f given in Equation 5.1 follows directly from the definition of the gradient as being the vector of partial derivatives of f (Kleitman [2005]).

Spall [2003] suggests a less time-consuming alternative to the gradient estimate given in Equation 5.1. The one-sided difference procedure (1FDSA) is described by the following equation:

$$\vec{g}(x_k, y_k) = \begin{bmatrix} \frac{f(x_k + c_k, y_k) - f(x_k, y_k)}{c_k} \\ \frac{f(x_k, y_k + c_k) - f(x_k, y_k)}{c_k} \end{bmatrix} \tag{5.2}$$

Equation 5.2 requires that the homing robot visit three adjacent points and carry out a single function evaluation (i.e. MI calculation) at each of these points. When estimating the gradient of f with Equation 5.1, the robot visits four points and evaluates four *MI* image differences. There are several possible sequences of movement commands that the robot can use to visit these points. We have hand-choreographed these moves to minimise the movement time required to carry out 1FDSA and 2FDSA. Figure 5.1(a) depicts the actions our homing agent makes in order to calculate the difference surface gradient at (x_k, y_k) using Equation 5.2. The agent moves from the location of the previous gradient estimate (x_{k-1}, y_{k-1}) along the dotted line emanating from the lower left-hand corner of the figure to (x_k, y_k). When it reaches (x_k, y_k), the agent carries out a function evaluation. It then moves by c_k distance units (e.g. cm) in its current heading to $(x_k + c_k, y_k)$. Another function evaluation is carried out. The agent must now move to $(x_k, y_k + c_k)$ and carry out a function evaluation there. It could move back to (x_k, y_k), turn 90 degrees counter-clockwise[1] and move c_k units to $(x_k, y_k + c_k)$. It takes less time, though – starting from $(x_k + c_k, y_k)$ – to rotate 45 degrees clockwise and move in reverse by $\sqrt{2}c_k$ units to $(x_k, y_k + c_k)$; this is what the agent does. The agent uses dead-reckoning to estimate all distances travelled and angles turned. At this point, the agent has the information necessary to estimate the gradient of the difference surface at (x_k, y_k) using Equation 5.2.

Once the gradient has been calculated, the agent should move from (x_k, y_k) in the direction of the gradient (θ) by a distance a_k to reach (x_{k+1}, y_{k+1}). The value we assign a_k is discussed below. Unfortunately, the agent is currently sitting at $(x_k, y_k + c_k)$. It would be inefficient to travel back to (x_k, y_k) and from there move to (x_{k+1}, y_{k+1}). We instead use simple trigonometry to infer the distance (a'_k) and direction (θ') to (x_{k+1}, y_{k+1}) from $(x_k, y_k + c_k)$. These are given by the following equations:

$$a'_k = \sqrt{a_k^2 \cos^2\theta + (c_k - a_k \sin\theta)^2} \tag{5.3}$$

$$\theta' = atan2(a_k \sin\theta - c_k, a_k \cos\theta) \tag{5.4}$$

The function $atan2(y, x)$ is the so-called two-argument inverse tangent (Weisstein [2007a]); it returns the counterclockwise angle between the x-axis and the vector $[x\ y]^T$ and, unlike the standard inverse tangent function, it is valid for all four quadrants of the Cartesian plane.

The set of moves used to gather the difference surface values required to calculate Equation 5.1 is depicted in Figure 5.1(b). As in Figure 5.1(a), the agent moves along the dotted line starting from the lower left-hand corner of the figure toward (x_k, y_k). In this case, though, the agent stops c_k units from (x_k, y_k) to sample the difference

[1]Our simulated agent – like the Koala robot we shall use in our live robotic experiments described in Chapter 6 – is non-holonomic.

surface at $(x_k - c_k, y_k)$. It then moves ahead by $2c_k$ units to $(x_k + c_k, y_k)$; once there, it again samples the difference surface. To move to $(x_k, y_k + c_k)$, the agent as above rotates by 45 degrees clockwise and moves in reverse by $\sqrt{2}c_k$ units. After sampling the function at $(x_k, y_k + c_k)$ the agent rotates 45 degrees clockwise and moves forward by $2c_k$ units to measure the difference surface at $(x_k, y_k - c_k)$. Using the movement commands we just described, 1FDSA is approximately 2 times faster than 2FDSA. This movement speed-up may counterbalance the assumed reduction in the accuracy of 1FDSA's gradient estimate; we shall attempt to determine if this is the case in our experiments, described below.

Once the gradient has been estimated, the homing agent moves from its current location $\vec{x_k}$ in the direction of this gradient and by a distance a_k to the next point in the optimisation process, $\vec{x_{k+1}}$.

Spall [2003] suggests that the choice of the evolution of gains a_k and c_k are crucial to the success of the stochastic optimisation algorithm. Several authors (e.g. Spall [2003] and Cole-Rhodes et al. [2003]) use the following equations to compute gain sequences:

$$a_k = \frac{a}{(k+1+A)^{\alpha}} \tag{5.5}$$

$$c_k = \frac{c}{(k+1)^{\gamma}} \tag{5.6}$$

These equations of course depend on the user-defined values of a, c, A, α, and γ. As above, the independent variable k is the current iteration of the homing algorithm. Spall [2003] suggests that the value of these parameters is the deciding factor in the success or failure of the finite difference stochastic optimisation algorithm. For example, an overly large value of a will cause the optimisation algorithm to behave wildly in early iterations. If α or A is too large, the algorithm will move very slowly towards an optimum. If c is not large enough, the gradient estimate will be overcome by function noise.

Cole-Rhodes et al. [2003] limit the parameters of Equations 5.5 and 5.6 to the following: $a, c > 0$, $A \geq 0$, $0 < \gamma < \alpha < 1$. Spall [2003] provides a proof that if a, c, α, γ, the function to be minimised and the measurement noise meet certain criteria then one can guarantee that a stochastic optimisation algorithm utilising a finite-difference gradient will converge to an optimum value. Unfortunately, our function does not seem to meet the criteria. Still, Equations 5.5 and 5.6 provide useful forms for our gain sequences which we shall use in our experiments. We discuss determination of good values for a, c, A, α, and γ in Section 5.4.

5.3.2 Rejected Optimisation Algorithms

Spall and other authors suggest a number of optimisation algorithms which we have rejected. We discuss these algorithms, and why we chose to reject them, briefly in this section.

Spall [2003] describes a third method (apart from 1FDSA and 2FDSA) to estimate the gradient: simultaneous perturbation (SP). SP randomly selects a new direction from a given distribution (usually a Bernoulli distribution) and moves in that direction for c_k units. The function is measured at the beginning and end of the move. The difference in the two measurements is used to estimate the magnitude of the projection of the gradient in the chosen direction. The algorithm uses this estimated magnitude to decide how far to move in the chosen direction. SP requires only two function measurements per iteration. Unlike 1FDSA and 2FDSA, the number of function evaluations required per iteration remains fixed regardless of the number of independent variables. This limit in function evaluations comes at a cost of a relatively poor gradient estimate. In preliminary tests, the disadvantage of SP outweighed the advantage, so we chose not to use it in the experiments described below. SP may prove useful, though, if homing in three dimensions. In three dimensions, 2FDSA requires six function evaluations to estimate the difference surface gradient; SP still requires just two.

The Nelder-Mead (Spall [2003]) algorithm is a popular optimisation algorithm when only noisy function measurements (rather than noisy or non-noisy function gradients) are available. When optimising in two dimensions, Nelder-Mead (NM) samples the function at the vertices of an (initially random) triangle; the triangle is called a simplex. The vertex with the worst (i.e. least optimal) function value is identified and this vertex is reflected through the line connecting the other two vertices. If the function value at this new vertex is an improvement, another reflection if performed. If not, we shrink the simplex and evaluate the function at the vertices of the new vertex. We reject NM for homing because the path the algorithm generates is quite tortuous; as noted above, rotating and translating the homing robot takes time.

We discussed in Chapter 2 the use of the extended Kalman filter by Baccou and Jouvencel [2002] to solve a navigation problem essentially equivalent to difference surface homing. Though the Kalman filter is not an optimisation algorithm, we still discuss it in this section because we ultimately decided to reject it for use in difference surface homing. The extended Kalman filter allowed Baccou and Jouvencel [2002] to infer the location of their robot with respect to the home position using a succession of noisy estimates of their distance from home. We cannot use this approach in our work because our homing robot has no knowledge of the function relating difference surface values to home distance. It is concievable that the homing agent could learn such a function on its first outward trip from the snapshot location. We implemented this idea, though, and met with very little success. The problem was that the function learned by the agent as it left the snapshot location was invalid during homing in dynamic conditions. Another problem with the Kalman filter approach is the homing agent has no knowledge of its initial location with respect to the snapshot location; a rough intial position estimate is required by the Kalman filter. Using a particle filter (see e.g. Menegatti et al. [2004] and Fox et al. [2001]) instead of the Kalman filter may solve this problem.

Some authors (see e.g. Lizotte [2005]) recommend performing a regression to estimate the underlying function f from noisy samples. Given an estimate of f from regression, we could use well-known techniques from calculus to find the value of $\vec{x}$ which optimises the estimate of f derived from the regression. Standard regression techniques require knowledge of the underlying form of the function being sampled; we do not know the functional form of f. This is not necessarily a problem, since we could interpolate the difference surface – with a method like Gaussian process regression (Lizotte et al. [2007]) – using noisy difference surface samples and then search over all interpolated values to find an optimum. This approach is problematic because the difference surface samples must be paired with the values of $\vec{x_C}$ at which they were collected. According to item (1) in the above list, we have an increasingly noisy estimate of $\vec{x_C}$ as the robot moves from one sampling location to the next. For these reasons, we choose not to use any form of regression to optimise the difference surface.

In Zampoglou et al. [2006], we investigated the use of a biologically inspired optimisation algorithm for difference surface homing. The algorithm was based on chemotaxis behaviour of the nematode worm as described by Feree and Lockery [1999]. The algorithm takes a difference surface reading $f(\vec{x_k})$ at the agent's current location $\vec{x_k}$, moves forward by a distance dependent on $f(\vec{x_k})$ to $x_{\vec{k+1}}$ and takes another surface reading. The agent then turns in direct proportion to $f(x_{\vec{k+1}})$ and to $f(\vec{x_k}) - f(x_{\vec{k+1}})$; the turn is also subject to a constant bias. The constants of proportionality as well as the bias term are free parameters. We used a genetic algorithm in simulation to find values for these parameters which optimised homing success on a number of training difference surfaces (created using Vardy's image data sets). Image similarity was measured with *RMS* in this work since, when we carried out these experiments we were unaware of mutual image information. We found that the parameters evolved in simulation caused the robot to turn excessively in simulated test trials. Also, the parameters evolved in simulation were not effective when applied to "live" robotic homing. For these reasons, we chose not to pursue this method of optimisation in this chapter.

5.4 Experiments in Simulation

Here, we will describe simulation experiments we undertook to determine which of the optimisation algorithms described in Section 5.3 works best to solve the problem of visual homing by optimising on difference surfaces. To make the simulation as realistic as possible, we investigated the noise injected into the mutual image information signal by our noisy Webcam and compass sensors; see Section 5.4.1 and 5.4.2 respectively. We use the same robot movement noise model as was employed in the experiments in Chapter 3. As we are using Vardy's image data sets in our simulations, mutual image information is only available at grid points spaced at 30cm intervals on a 3m x 4.5m planar area. We determine in Section 5.4.3 the best way to interpolate the difference surface in the presence of the aforementioned signal noise.

5.4.1 Webcam Capture Noise

We wanted to characterise the noise present in images captured by our Webcam. To do so, we took 28 grayscale images of a static scene (see Figure 5.2).

We consider the intensity of each pixel to be a random variable. To establish the expected value at each pixel, we computed the mean intensity value at each pixel, forming a mean image. We shall assume that this mean image is a good estimate of the expected image of the static scene.

To estimate the distribution of intensity noise, we subtracted the mean image from each of our 28 test images and used the resulting intensity differences to produce a histogram, shown in Figure 5.3. The distribution of intensity noise from the mean is approximately Gaussian (skewness = -0.0352, kurtosis = 3.2556) with a standard deviation of 0.9443 and mean of zero.

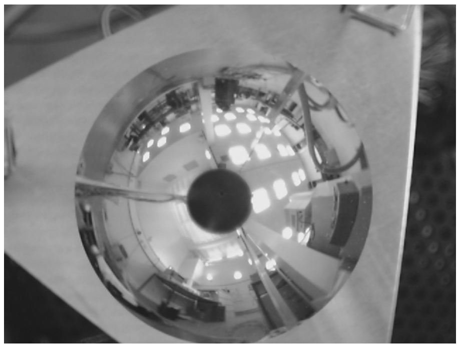


Figure 5.2: Image of static scene used in Webcam noise test. Image is of our panoramic mirror.

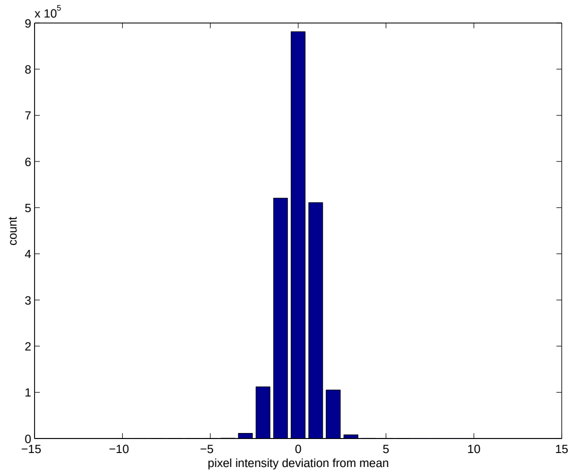


Figure 5.3: Histogram of intensity deviations from the mean (i.e. intensity noise) in our test images.

Distance from snapshot (cm)	$MI_{Noiseless}$	mean MI_{Noisy}	std. dev. MI_{Noisy}
0	7.1648	5.3204	0.0023
30	1.4825	1.4758	0.0009
60	1.0444	1.0444	0.0009
90	0.8808	0.8806	0.0011
120	0.7787	0.7788	0.0008
150	0.6973	0.6970	0.0009
180	0.6307	0.6307	0.0009
210	0.5958	0.5962	0.0008
240	0.5463	0.5455	0.0009
270	0.5026	0.5006	0.0009
300	0.4833	0.4821	0.0008
330	0.4339	0.4336	0.0009
360	0.4039	0.4040	0.0009
390	0.3730	0.3731	0.0008
420	0.3495	0.3476	0.0010

Table 5.1: Comparison of mutual information values computed from noiseless and noisy current images. The intensity of each image pixel was corrupted with zero-mean Gaussian noise. See text for details.

Autocorrelation of each noise image indicates that intensity noise is spatially independent. We do not have enough data to determine whether pixel intensity noise is correlated over time, but visual inspection suggests that temporal correlation is low.

We saw in Chapter 4 that certain image capture noise has quite a deleterious effect on *MI* difference surfaces. We wonder if this is true of the zero-mean Gaussian white noise described here. To find out, we computed an *MI* transect using images from the “Original” data set beginning at x=120cm, y=60cm and moving parallel to the y-axis. We computed the *MI* signal with noiseless images (assuming Vardy’s images are noiseless) and then corrupted each current image 100 times with a different noise matrix each time. Elements of the noise matrix were drawn independently from $N(0, 0.9443^2)$. The plot of this data is rather difficult to interpret so we instead list it in Table 5.1.

As we can see in Table 5.1, $MI_{Noiseless}$ decreases monotonically as distance from the snapshot location increases (as expected). This is also true of the mean MI_{Noisy}

signal. The mean MI_{Noisy} signal is less than the $MI_{Noiseless}$ signal at and near (i.e. 30cm away from) the snapshot location. This makes sense: I_C is a good predictor of I_S near the snapshot location. Adding noise to I_C in this case will therefore cause relatively large decreases in MI. Away from the snapshot location, the mean value of MI_{Noisy} is approximately equal to $MI_{Noiseless}$. The standard deviation of the noisy signal is relatively small at all data points. This indicates that any single noisy MI value is likely to be close to the mean. Thus, it is probably enough to capture one current image I_C and use the MI valued derived from it rather than the mean MI value computed from several images captured at the same location.

We would like to transform the distribution of intensity noise described above into a distribution of mutual information given noisy current images. Such a distribution of MI noise would be quite useful in quickly injecting realistic noise into our homing simulations. Unfortunately, we were unable to do this, though we did derive some aspects of such a noise distribution for RMS difference signals (see Appendix A). Below we discuss other methods to simulate homing with noisy sensors.

5.4.2 Compass Noise

The visual homing system requires an estimate of the robot's orientation in some external reference frame. This is because the robot almost certainly has a different orientation at S than it does at C. I_C must be rotated in software to account for this orientation difference, otherwise measuring the similarity between I_S and I_C would be meaningless. The orientation estimate will be somewhat noisy in real-world experiments. In this section we investigate the effects of compass noise on the MI signal.

We originally intended to use a digital magnetic compass in our homing experiments. As we describe in Section 6.2.1, our digital compass was overcome by environmental noise in the indoor environment in which we carried out experiments. We therefore had to rely on our tracking system to provide directional information. As we discuss in Section 6.2.2, we at first sought to use a tether tracking system to track the robot during experiments and provide "live" directional information. This tether tracker proved unsuitable and we created a visual tracker to replace it. The experiments in this section, however, assume the use of the tether tracker. The noise characteristics of the direction estimates provided by the tether and visual trackers are similar.

The tracker provides position information in a reference frame defined by the location of the tether bases. We can in principle use the difference between the current

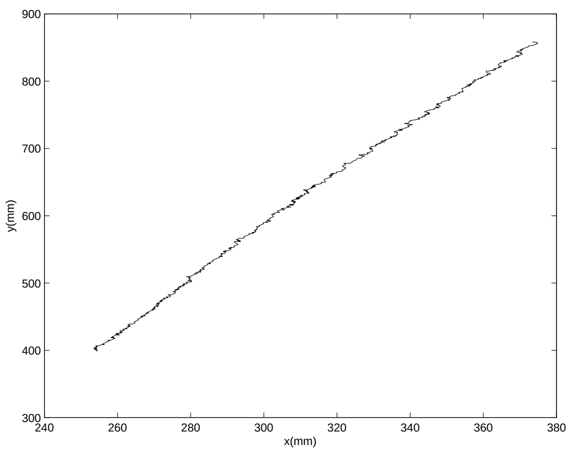


Figure 5.4: Plot of locations recorded by our tracking system. Robot moved in a straight line from lower left to upper right.

position estimate and the previous position estimate to infer the agent's direction of travel. In practice, these estimates are somewhat noisy, as illustrated in Figure 5.4. Thus, we shall estimate travel direction by fitting a line to the previous N samples, where N is to be determined.

In order to determine the noise characteristics of the compass signal provided by the tracker, we drove a tethered agent along fourteen straight tracks on our laboratory floor. The agent moved at constant speed of about $1\frac{cm}{sec}$. At this speed, position estimates were generated on average every 0.85mm. The shortest track was roughly 10cm and the longest, 50cm. Track directions varied.

For each track, we took the agent's true direction to be the slope of the best fitting line through the set of track points. We computed the agent's estimate of its current direction at every 50^{th} data point, using the previous 80 data points. The value of N=80 was empirically determined to provide a small standard deviation in the local direction estimate.

We calculated the distribution of difference between the agent's actual direction of travel and its current local estimate; see Figure 5.5. This distribution has kurtosis equal to 3.36 so we conclude that it is Gaussian. The standard deviation is 0.86 degrees. As is clear in the plot, the distribution is skewed slightly positive. We believe that this was caused by an error in calibrating the tether tracking system (see Section 6.2.2 for information on tracker calibration). For simulation purposes, we shall assume that generally the distribution of compassing error is zero mean Gaussian with the reported standard deviation. Since successive direction estimates use overlapping data sets,

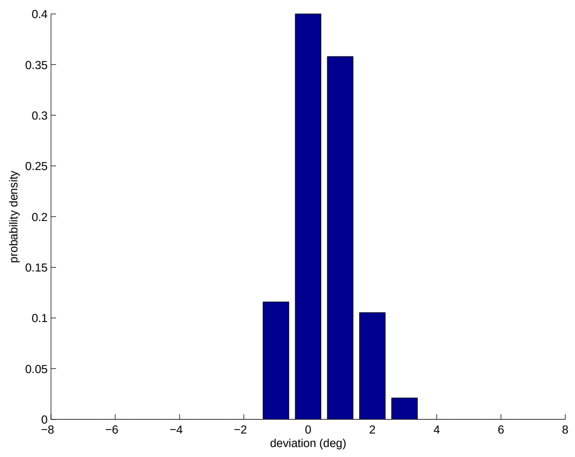


Figure 5.5: Distribution of difference between actual direction and estimated direction using tether tracker to infer direction.

there is a small correlation between errors in successive estimates. We shall ignore this in simulation.

As with Webcam noise, we are interested in the effect of a noisy direction estimate on the *MI* signal. We computed an *MI* transect using images from the “Original” data set beginning at x=120cm, y=60cm and moving parallel to the y-axis. We computed the *MI* signal with noiseless images (assuming Vardy’s images are noiseless) and then corrupted each current image 100 times with a different rotation error each time. Compass errors were drawn independently from $N(0,0.86^2)$. We list the results of this experiment in Table 5.2.

Compass noise seems to have a greater effect on the *MI* signal near the snapshot location than does Webcam noise. The standard deviation of the noisy signal is such that some compass readings may mask the global maximum at the snapshot location. Care should be taken when nearing the goal that the compass signal is accurate.

As with the Webcam noise, we were unable to completely characterise the effects of compass noise on the *MI* signal. We describe how compass noise is injected into our homing simulation below.

5.4.3 Interpolating the Difference Surface

We will use Vardy’s image data sets to form difference surfaces with which to simulate homing. We wish to include the measurement noise identified in the Sections 5.4.1 and 5.4.2 to make the simulation as realistic as possible. As has been described pre-

Distance from snapshot (cm)	$MI_{Noiseless}$	mean MI_{Noisy}	std. dev. MI_{Noisy}
0	7.1648	3.4089	1.6460
30	1.4825	1.4696	0.0197
60	1.0444	1.0464	0.0060
90	0.8808	0.8795	0.0052
120	0.7787	0.7754	0.0085
150	0.6973	0.6937	0.0104
180	0.6307	0.6311	0.0071
210	0.5958	0.5947	0.0045
240	0.5463	0.5464	0.0016
270	0.5026	0.5029	0.0020
300	0.4833	0.4825	0.0038
330	0.4339	0.4333	0.0042
360	0.4039	0.4030	0.0018
390	0.3730	0.3707	0.0024
420	0.3495	0.3508	0.0010

Table 5.2: Comparison of mutual information values computed from noiseless and noisy current images. Current images were rotated by a random amount to simulate compass noise. See text for details.

viously, Vardy's images were sampled every 30cm on a regular 3.0m$\times$5.4m planar grid. What should we do when, during the simulation, the agent wishes to measure the difference surface value at a non-grid point $\vec{x}$ which has been corrupted by this measurement noise?

Though we know the noise characteristics of the compass and Webcam, we were unable to derive a distribution for noise in the MI signal resulting from imprecision in these sensors. Our only recourse, then, is, to corrupt an image or images at grid points close to $\vec{x}$ with compass and Webcam noise and use corresponding signal(s) to estimate the noisy MI signal at $\vec{x}$. We thought of a number of ways to do this:

1. Corrupt the image at the grid point closest to $\vec{x}$ with compass and Webcam noise. Use the MI value calculated with this corrupted image as the best estimate of the MI value at $\vec{x}$ corrupted by the same noise.

2. Corrupt the four images at grid points surrounding $\vec{x}$ with identical compass and Webcam noise. Compute the MI value for each of these corrupted images. Take the noisy MI value at $\vec{x}$ to be the weighted sum of the four surrounding MI values; weight each according to its distance from $\vec{x}$ and normalise so that the weights sum to 1.

3. Corrupt the four images at grid points surrounding $\vec{x}$ with identical compass and Webcam noise. Compute the weighted sum of the images (assign weights, again, according to distance from $\vec{x}$ and ensure that weights sum to 1). Compute the MI value of this composite, noisy image and use this as the estimate of the MI value at $\vec{x}$.

4. Compute the weighted sum of the four images surrounding $\vec{x}$. Corrupt the composite image with Webcam and compass noise. Compute the MI value of this noisy image and use this as the estimate of the MI value at $\vec{x}$.

We note that the first interpolation method described above is somewhat unrealistic, as it yields a "stepped," highly discontinuous difference surface, akin to an Egyptian pyramid. We include the method, though, because it is significantly faster than the other three.

How do we choose which of these interpolation methods to use? We do so by looking at positions $\vec{x}$ for which we do have image data. We selected snapshot images from the "Original" data set at the locations shown in Figure 5.6. For each snapshot, we

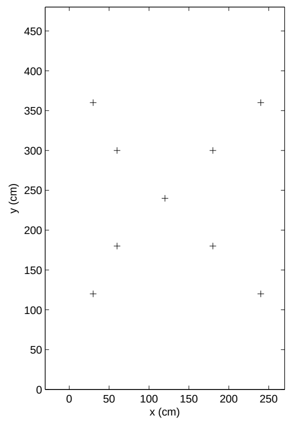


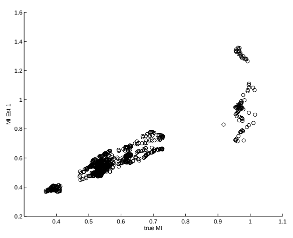
Figure 5.6: Location of snapshots used in experiment to determine the best way to interpolate the difference surface when compass and Webcam are corrupted by noise.

randomly selected 100 current images from the "Original" data set. For each current image I_C, we randomly generated a Webcam noise matrix and compass error from the distributions described in Sections 5.4.1 and 5.4.2. We corrupted I_C with this noise and computed the mutual information between the current snapshot and the corrupted image I_C; we shall call this MI_{true}. We corrupted the images at grid points directly north, east, south and west of I_C with the same noise values. We then estimated MI_{true} using each of the four methods described above producing MI_{est1}, MI_{est2}, MI_{est3}, MI_{est4} for each MI_{true}. Weights, when required, were set at 0.25 as all image captured locations were an equal distance from the location of I_C.

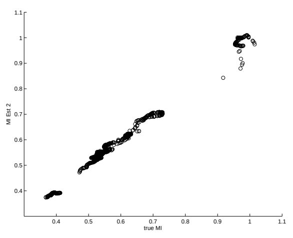
The above procedure gave us 900 data points with which to judge the efficacy of each method. The scatter plots of MI_{true} versus MI_{est1}, MI_{true} versus MI_{est2}, and so forth are given in Figure 5.7. The relationship in each case is clearly linear so we computed the correlation coefficient between the values of MI_{true} and MI_{est1}, between MI_{true} and MI_{est2}, etc. These correlation coefficients are given in Table 5.3. Though each relationship is quite strong, Method 2 yields the highest coefficient. The probability of getting the reported correlation by chance is very close to zero in each case. We also looked at the standard deviation of the difference between M_{true} and each estimate. Method 2 yields the smallest standard deviation. Thus, we shall use Method 2 to interpolate the difference surface for MI values corrupted by sensor noise.

Dynamic environments had little effect on this result.

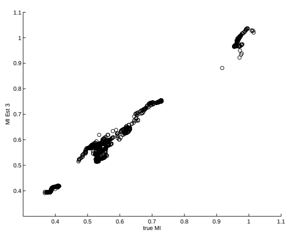
Method 2 above involves manipulating images of the environment in order to generate noisy interpolated MI values. It would probably be more efficient to draw a noisy MI value from a probability density function (p.d.f.) whose distribution is based on



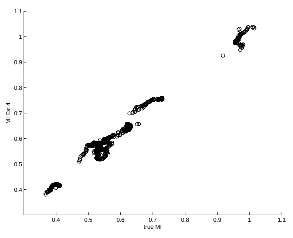
(a)



(b)



(c)



(d)

Figure 5.7: Scatterplots indicating relationship between MI_{true} and MI_{est} for each MI difference surface interpolation method described in Section 5.4.3.

	Method 1	Method 2	Method 3	Method 4
r	0.9114	0.9961	0.9904	0.9902

Table 5.3: Correlation coefficients for the four scatterplots shown in Figure 5.7.

the noise distributions of Webcam and compass measurements. But is it possible to determine this probability density function without considering the content of current and/or snapshot images?

We investigated this question in simulation, as it was easier to change the structure of a simulated environment than a real one. The simulated environment is a two-dimensional world enclosed by a circular wall which is segmented into arcs of equal length. Each arc is painted a particular shade of gray. The length of each arc and the shades which they are painted are controlled by the user. These shaded arcs provide the visual information required for homing. No other landmarks are present. The agent can take a panoramic image (a one-dimensional array of 360 gray-scale elements) from any position within the circular enclosure.

To demonstrate that the noise in the MI signal which results from compass noise is dependant on the structure of the environment, we first create a world in which each painted segment of the circular enclosure has an arc length of five degrees. We use nine shades of gray – ranging from 0 to 8 – to paint the arcs. The first arc – at a bearing of zero and running counterclockwise – is painted with shade zero, the next is painted with shade one, and so on. When a particular arc is painted with shade eight, the next will be painted with shade zero. We let both S and C be located at the centre of the circle. The agent has a noisy compass whose signal is used to rotate current images to account for orientation changes between S and C. The compass noise has a simple distribution: it is constantly three degrees. The $MI_{Noiseless}$ value in this case is 3.1699 and the value of MI_{Noisy} is 2.1990, a difference of 0.9710. We then increase the length of each painted arc to ten degrees. In this case, the $MI_{Noiseless}$ value is again 3.1699 but MI_{Noisy} is 2.2886, a difference of 0.8813. Finally, we increase arc length to 20 degrees and find that $MI_{Noiseless}$ remains 3.1699, MI_{Noisy} has risen to 2.5601, and the difference between them is 0.6098.

We see clear, explicable trends in these results. As arc length increases, $MI_{Noiseless}$ remains constant. This is because, since I_C is identical to I_S when compass noise is ignored, $MI_{Noiseless}$ is equal to the entropy of I_C (or equivalently the entropy of I_S). In each of the three cases above, I_C has a uniform intensity histogram of nine elements. Thus, $MI_{Noiseless}$ is unchanged in each of these three cases. More germaine to the topic at hand, we also see above that when arc length increases, MI_{Noisy} also increases, though always remaining less than $MI_{Noiseless}$. When arc length of each segment is 5 degrees, there are 72 objects visible in I_S and I_C. A three degree compass error will result in $3 \cdot 72 = 216$ pixel disagreements between I_C^{Noisy} and I_S. But when object arc length is 10 degrees, I_C contains 36 objects so a three degree compass error will result in only $3 \cdot 36 = 108$ pixel disagreements. This reduction in pixel disagreements means that I_C^{Noisy} is a better predictor of I_S in the latter case than in the former, leading to an increase in MI_{Noisy} in the latter case. The trend continues when object arc length is increased to 20 degrees. MI_{Noisy} is always less than $MI_{Noiseless}$ because the compass noise always causes some drop in the ability of I_C^{Noisy} to predict I_S.

Generalising from this result, we hypothesise that – in real world homing – mutual information noise (i.e. the difference between $MI_{Noiseless}$ and MI_{Noisy}) for a given compass error will depend on the distribution of the apparent sizes of imaged objects. Images containing few, relatively large objects will exhibit less mutual information noise than those with objects of lesser apparent size. Apparent object sizes in turn depend on the structure of the environment in which homing occurs. Since mutual information noise caused by compassing noise depends on scene structure, one cannot simulate noise in mutual information without attending to the images used to compute the mutual information.

5.4.4 Stopping Criteria

In static environments in which there is no sensor noise, it is easy to determine when to stop homing. The agent could simply remember the difference surface value at the snapshot location at the beginning of its outbound journey and stop homing when the absolute difference between the current surface value and this stored “home” value falls below some (small) threshold.

Spall [2004] suggests that deciding when to stop a stochastic optimiser is much more difficult. Certain of our findings lead us to agree. In Chapter 3 we reported that environmental change after the snapshot image is captured diminishes the global optimum at the snapshot location. In Sections 5.4.2 and 5.4.1 we found that sensor noise – particularly compass noise – injects variability into the MI signal which is particularly acute at the snapshot location. Ideally, we would use the current image and knowledge of sensor noise to predict the modified difference surface value at the snapshot location; unfortunately, we found no method to do this. Instead, we must rely on the current different surface value and perhaps other comparisons between the current and snapshot images to derive stopping criteria. These criteria must be as robust as possible to changes in the environment and to sensor noise. They must be applicable without knowledge of how the environment has changed and/or current sensor noise values.

We explored several different stopping criteria of our own invention. The interested reader can refer to Appendix B to find out more about these criteria and the experiments we carried out to measure their viability. Unfortunately, the more successful ones required the agent to gather information about the range of difference surfaces found in its environment in both static and dynamic conditions. This learning would require a lot of time and effort before homing could begin; we did not think this was feasible.

For both simulated and live homing runs, we employed the stopping criterion used in our experiments in Chapter 3. Optimisation halts when several successive values of $\vec{x}_k$ cluster around a point. Such clustering indicates that this point is a local optimum. This criterion was suggested by Spall [2003]. The clustering criterion can be used without any knowledge of the environment in which homing is taking place and can be implemented in conjunction with every optimisation algorithm studied in this chapter. Of course, this criteria will cause the homing agent to halt at any difference surface optimum, whether it coincides with the snapshot location or not. We shall also halt homing runs after a large amount of simulated time (900 seconds) has elapsed without the clustering criterion having been met. The simulation of homing time is discussed in Section 5.4.5.1.

5.4.5 Experiments

We wanted to determine which of the optimisation algorithms described in Section 5.3 works best in static and dynamic indoor environments. We used Vardy's image data sets to provide the images for our simulations. Please refer to Chapter 3 for a full description of these data sets.

As in the experiments in Chapter 3, we drew snapshot and current images from the same data set in order to simulate static conditions and from different data sets in order to simulate dynamic conditions. We used the "Original", "Winlit" and "Chairs" data sets here. We paired these data sets to create two static and three dynamic environments: ("Original", "Original"), ("Original", "Winlit"), ("Winlit", "Original"), ("Winlit", "Winlit"), ("Chairs", "Original").

For each data set pairing above, we fixed nineteen snapshot locations; these are depicted as black diamonds in Figure 5.8. Note that a few of these snapshot locations are slightly different than the ones used in Chapter 3; this is because we are no longer using the "Arboreal" data set so do not have to worry about a snapshot location being

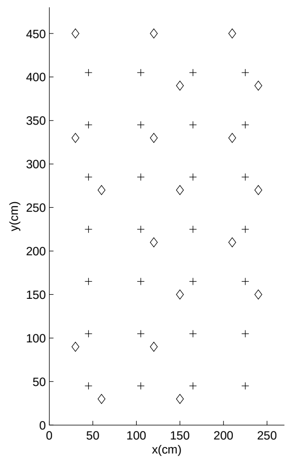


Figure 5.8: The set of nineteen snapshot locations and twenty-eight starting locations used in our experiments. A diamond indicates a snapshot location. A plus indicates the location of the start of a homing run.

in the interior of the plant in that data set.

For each combination of data set pairing and snapshot location, we homed starting from a set of twenty-eight starting locations; these are depicted as black plusses in Figure 5.8. Note that we skipped a homing run if current images were drawn from the "Chairs" data set and the starting location fell within the interior of a chair.

We explore fewer data set pairings and starting locations here than we did in Chapter 3 because the addition of sensor noise in our simulation increases the time required to compute mutual information, thus causing simulated homing runs to take too long to run. We computed 100 mutual information calculations with injected sensor noise on our desktop computer and found that a calculation took on average 0.16 seconds. This value takes into account image reading, calculation and addition of image noise, image masking, histogramming and entropy calculations. In the experiments of Chapter 3, we used 16 data set pairings and 19 snapshot locations, yielding a total of 304 difference surfaces on which we simulated homing. For each difference surface, 169 homing runs were carried out, each starting from a different non-snapshot grid point. In total, therefore, the experiments in Chapter 3 involved 51376 homing runs. If we had used the same number of data set pairings and starting locations in the experiments described in this chapter, we would have carried out 513760 homing runs as we consider a total of ten optimisation algorithms in this chapter. We found that – in the experiments we did carry out here – 74 mutual information calculations on average were required per homing run. Thus, to process the mutual information calculations for 513760 homing runs would have taken approximately 6082900 seconds, or 70.4 days of computation time. This is a low estimate for the total time these experiments would have taken as we have only accounted for the calculation of mutual information. Though homing simulations in this chapter are mostly taken up with MI calculation, other factors (like keeping track of the simulated agent's pose, saving files, etc.) come into play as well. We consider something more than 70.4 days of computation time too long to wait for a set of experiments to terminate.

For each combination of data set pairing, snapshot location and starting location, we carried out a homing run using each of the optimisation algorithms described in Section 5.3.

We also homed with Zeil's "Run-Down" algorithm (Zeil et al. [2003]) to determine if any of our suggested optimisers is better than that which currently exists in the visual homing literature. "Run-Down" is known in the optimisation literature as one-direction-at-a-time search (see e.g. Adby and Dempster [1974]). "Run-Down" works as follows: the agent travels in the direction it is currently facing, periodically measuring the function to be optimised. The distance c between samples is an algorithm parameter and is unchanging during optimisation. When the current function value measurement is less than the previous one, the agent turns ninety degrees to the left or right (it does not matter which, as long as the same direction is consistently taken). The agent moves by c in this new direction and measures the function. If this new measurement is an improvement, the agent continues to move as before; if not, the agent turns 180 degrees and moves in this direction, sampling the function every c units, until the function ceases to improve. It then turns ninety degrees in either direction and repeats the process described above. The agent executes this algorithm until stopping criteria is met. "Run-down" was not specifically designed for use with noisy function values. The distance between samples is typically chosen by a human operator to be large enough to overcome measurement noise but small enough so that the algorithm halts within a reasonable distance from the optimum.

A drawback of the "Run-Down" algorithm is that it takes a rather tortuous route to the snapshot location while homing. A drawback to the gradient-based optimisation methods we described above (i.e. 1FDSA) is that they may, depending on the value of a_k, move along the estimated gradient a relatively long distance before resampling the difference surface. If the gradient estimate is poor, then the agent could move a significant distance before realising its mistake. If the gradient estimate is good, the agent might still move too far in that direction, overshooting the snapshot location. We shall try merging these algorithms into a novel fourth algorithm: travel in the direction of the estimated gradient, but instead of moving for a fixed distance, sample the difference surface periodically and recalculate the gradient when a decrease in successive difference surface values is detected. This hybrid algorithm may decrease the overall distance travelled by the agent during a homing run. On the other hand, it may require more difference surface evaluations than “Run-Down” and may be more susceptible to stalling in non-goal optima than the other gradient-based stochastic optimisers we used in our experiments.

5.4.5.1 Criterion for Comparison of Optimisation Algorithms

Those who study optimisation algorithms often base their comparison of competing algorithms on the number of function evaluations required to come within a certain small distance of an optimum value (Spall [2003]). As we noted in Section 5.3, the homing robot must translate and rotate in order to move from one difference surface evaluation point to the next. Hence, the time required to undertake these motor commands as well as the time taken to actually compute mutual image information must be taken into account in any comparison of algorithms. One of our criteria for algorithm comparison is therefore the total time T_{Total} taken for a homing run, as given by the following formula

$$T_{Total} = \frac{\theta_{Total}}{speed_{rotation}} + \frac{D_{Total}}{speed_{translation}} + \frac{F_{Total}}{speed_{computation}} \tag{5.7}$$

where θ_{Total} is the sum of all angles turned by the homing agent during a homing run; $speed_{rotation}$ is the speed of the agent’s rotation; D_{Total} is the total linear distance travelled during a homing run; $speed_{translation}$ is the speed at which the agent translates; F_{Total} is the total number of mutual image information computations undertaken during a homing run; and $speed_{computation}$ is the time required for each MI computation. For our simulation experiments, we assume our robot translates at a speed of 8.0 cm/sec and rotates at a rate of 28.65 degrees/sec. These are reasonable estimates of the speed our actual Koala Silver mobile robot moves at in “live” homing runs (see Chapter 6). We assume that one mutual image information evaluation takes 0.1 seconds, again a reasonable estimate of the actual time required for this operation in “live” runs. T_{Total} is measured in seconds. Equation 5.7 will not only allow us to compare different optimisation algorithms, it will also give us a concrete sense of the actual time required to complete a homing run in real-world experiments. We only take successful homing runs (i.e. those halting within 30cm of the snapshot location) into account when using this criteria.

We shall also compare optimisation algorithms using the return ratio measure defined in Chapter 3.

5.4.5.2 Setting Finite Difference Gain Parameters

Spall [2003] outlines a method to automatically set the parameters a, c, α, γ and A of Equations 5.5 and 5.6. This approach assumes that the standard deviation of the function value noise is independent of the location at which the function is evaluated. From Sections 5.4.2 and 5.4.1 we know this not to be the case; noise variance increases dramatically when mutual information is evaluated near the snapshot location. It seems therefore that we have to set the values of a, c, α, γ and A manually. We did not want these parameter settings to result from extensive domain knowledge; after all, the homing robot cannot "practice" homing to several difference snapshot locations from several different starting points, adjusting gains as it goes, before undertaking a "real" homing task. We therefore observed homing runs to just two snapshot locations from a small number of starting locations, taking all images from Vardy's "Original" data set. We tried several different settings of a, c, α, γ and A for each of these homing runs. We chose the parameter set which provided the best homing runs as measured by Equation 5.7. We note too few trials were undertaken to claim that one parameter set provided statistically significantly better results than another.

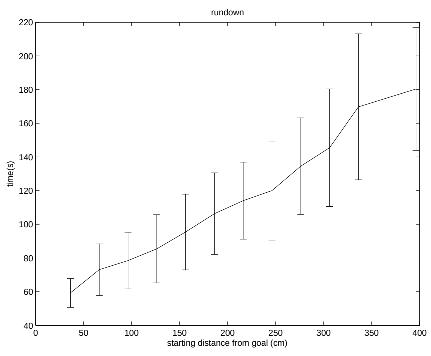
For one-sided finite differencing, we let $c = 15cm$, $a = 100cm$, $A = 0$, $\alpha = 1$ and $\gamma = 0$. We found that any value of α much less than one caused the values of a_k to decay too slowly; the homing agent often overshot the snapshot location several times before stopping at it. We set a lower limit of a_k at 15cm to prevent the agent from moving negligible distances in the direction of the gradient. A constant value of $c_k = 15cm$ seemed to suffice for gradient estimation both near and far from the snapshot location so we let $\gamma = 0$. We use the same gain parameters for two-sided differencing. Using the same procedure, we chose the step size c of "Run-Down" to be 15cm.

5.4.6 Results and Discussion

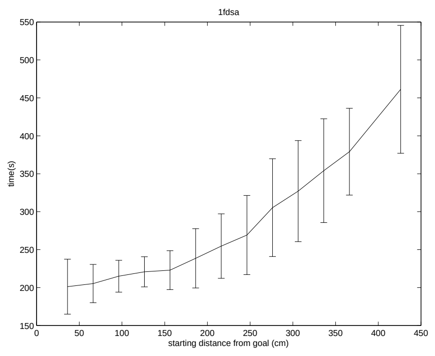
Given the results reported in Table 5.4, it seems that homing with noisy sensors is a more difficult problem than homing with noise-free sensors. Though different starting locations and – in some cases – different snapshot locations were used in the exper-

Optimisation Method	Average Return Ratio
Run-Down	0.807
1FDSA	0.836
2FDSA	0.905
Hybrid Run-Down/1FDSA	0.722

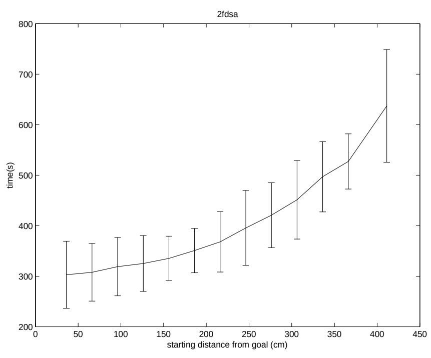
Table 5.4: Average return ratios for static environments for each optimisation algorithm considered in the chapter. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test (see Chapter 3 for details of this significance test).



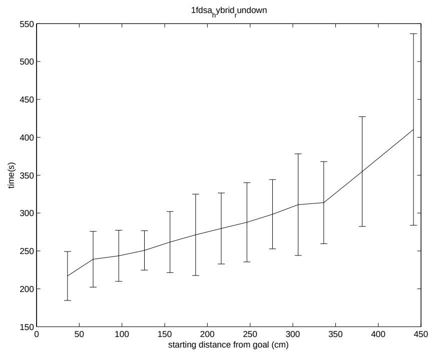
(a) Run-Down



(b) 1FDSA



(c) 2FDSA



(d) Hybrid

Figure 5.9: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. Experiments were conducted in static environments. The error bars indicate the standard deviation from the mean homing time.

Optimisation Method	Average Return Ratio
Run-Down	0.504
1FDSA	0.567
2FDSA	0.641
Hybrid Run-Down/1FDSA	0.580

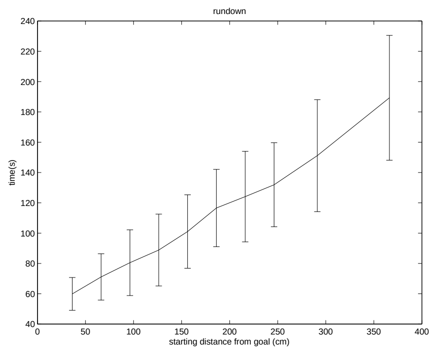
Table 5.5: Average return ratios for dynamic lighting environments for each optimisation algorithm considered in the chapter. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.

Optimisation Method	Average Return Ratio
Run-Down	0.801
1FDSA	0.798
2FDSA	0.833
Hybrid Run-Down/1FDSA	0.695

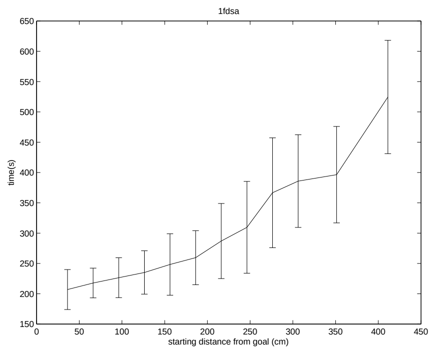
Table 5.6: Average return ratios for a moving landmark environment for each optimisation algorithm considered in the chapter. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.

iments reported in Chapter 3 making direct comparison difficult, the return ratios for homing in static environments were generally higher in Chapter 3 than those reported in Table 5.4. It is clear that gradient ascent homing using gradients estimated by two-sided finite differencing is superior in static environments. Unfortunately, Figure 5.9 indicates that – as we expected – two-sided finite differencing takes much more time than 1FDSA. Gradient ascent with one-sided finite differencing exhibits the second best return ratio in static conditions (see Table 5.4).

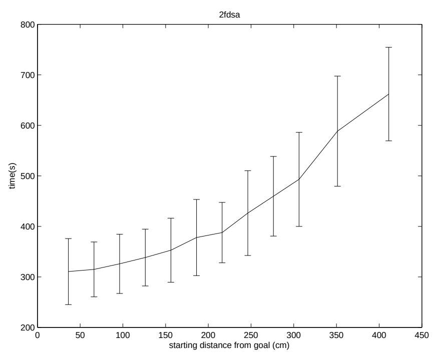
The trends we saw in static conditions are generally repeated in simulated environments with dynamic illumination. Table 5.5 indicates that gradient ascent with gradient computation with two-sided finite differencing has the highest return ratio. Gradient estimation with one-sided differencing is second-best. Again, according to Figure 5.10, the relative success of 2FDSA comes at a cost of greater mean homing times. We note that no optimisation algorithm performs particularly well in the face of dynamic illumination. Almost all failed runs become stuck in local optima which do not coincide with the snapshot location. We shall deal with this problem below.



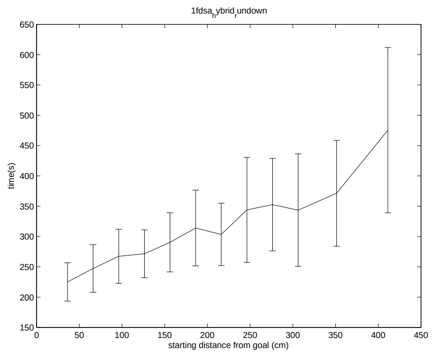
(a) Run-Down



(b) 1FDSA



(c) 2FDSA

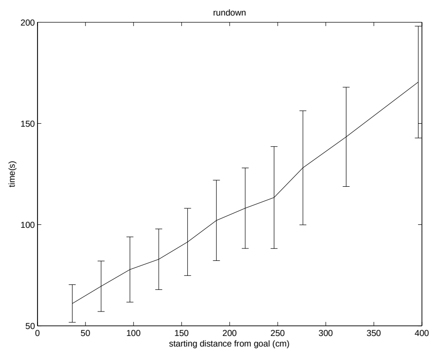


(d) Hybrid

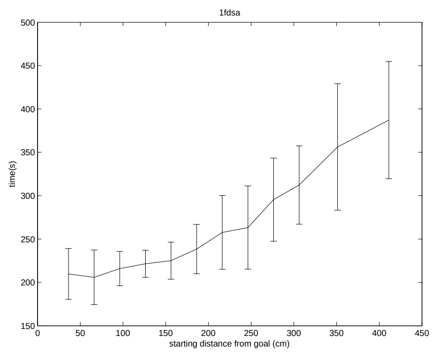
Figure 5.10: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. Experiments were conducted in dynamic illumination environments. The error bars indicate the standard deviation from the mean homing time.

Optimisation Method	**Average Return Ratio**
Run-Down	0.685
1FDSA	0.721
2FDSA	0.785
Hybrid Run-Down/1FDSA	0.660

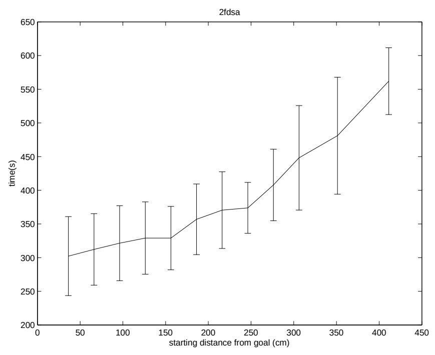
Table 5.7: Average return ratios for all (static and dynamic) environments for each optimisation algorithm considered in the chapter. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.



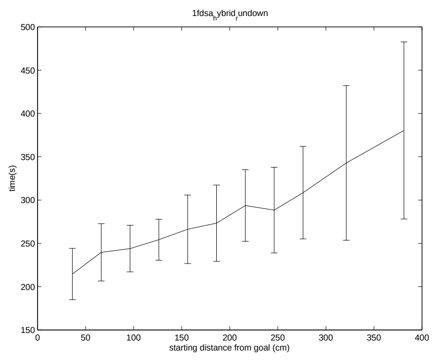
(a) Run-Down



(b) 1FDSA

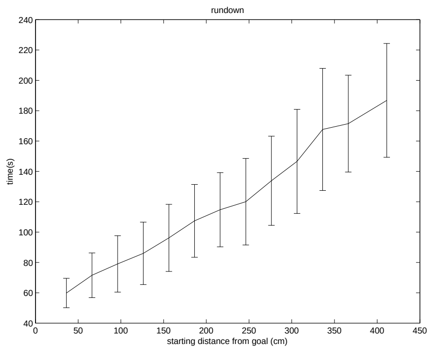


(c) 2FDSA

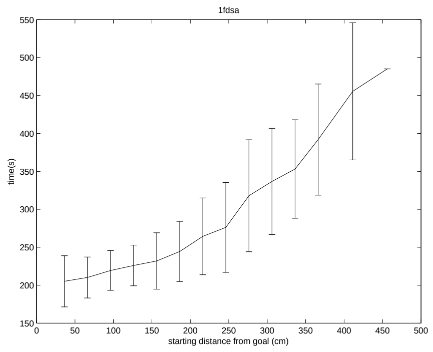


(d) Hybrid

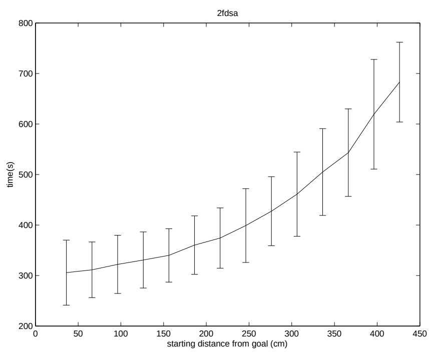
Figure 5.11: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. Experiments were conducted in an environment in which landmark locations changed between captures of snapshot and current images. The error bars indicate the standard deviation from the mean homing time.



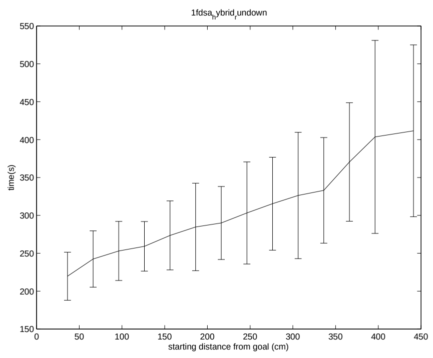
(a) Run-Down



(b) 1FDSA



(c) 2FDSA



(d) Hybrid

Figure 5.12: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. All difference surface pairings were taken into account. The error bars indicate the standard deviation from the mean homing time.

5.4.7 Experiments: Avoiding Local Maxima

In looking at individual homing runs in the previous section, it is clear that the overwhelming reason for homing failure is that the agent gets trapped in difference surface maxima that do not coincide with the snapshot location. Few homing runs “time out.” In accord with the observations given in Chapter 3, the function values at these non-goal maxima are almost always much less than the function value at the snapshot location for a particular difference surface. Therefore, in this section, we want to determine if the non-goal maxima can be distinguished from the difference surface maximum at the snapshot location. In a cursory examination of a number difference surfaces (both static and dynamic), we saw that all non-goal maxima had difference surface values less than 0.75 and all snapshot maxima were greater than 0.75.

We augmented each optimisation algorithm to detect and escape misleading difference surface optima as follows: when the algorithm’s stopping criterion is triggered, check if the current or previous difference surface samples is less than 0.75. If this is the case, then escape this assumed non-snapshot optimum by selecting a random value uniformly from the range $[0, 360]$, rotating by that amount and translating by one metre. The optimisation algorithm then resumes from this new spot.

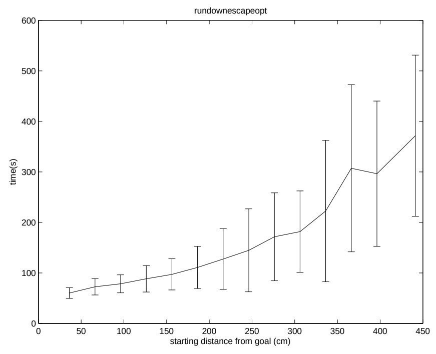
5.4.8 Results and Discussion: Avoiding Local Maxima

The detection of non-snapshot optima clearly has a salutary effect on homing in many cases. Compare, for example, Tables 5.4 and 5.8. All return ratios are dramatically higher in Table 5.8 than in Table 5.4; the differences are statistically significant, too, according to McNemar’s test with 95% probability. This increased success seems to come at the cost, though, of higher mean homing times; compare Figures 5.9 and 5.13. This increase in mean time is due to those homing runs which failed without non-snapshot optimum detection and which subsequently succeed because of it. These relatively time-consuming homing runs are not counted in the means reported in Figures 5.9 but do affect the means in Figure 5.13.

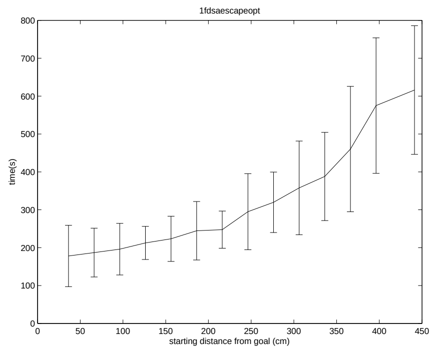
We see as well that homing is more likely to succeed in dynamic lighting conditions when we attempt to detect and avoid non-snapshot optima. Compare Tables 5.5 and 5.9. We noticed that almost all homing failures reported in Table 5.9 result from the algorithm reaching the maximum number of iterations; that is, no optimum is ever located. This happens because in these cases the function value at the snapshot location is less than 0.75, our crudely determined threshold for classifying difference surface

Optimisation Method	**Average Return Ratio**
Run-Down	0.826
1FDSA	0.974
2FDSA	0.979
Hybrid Run-Down/1FDSA	0.911

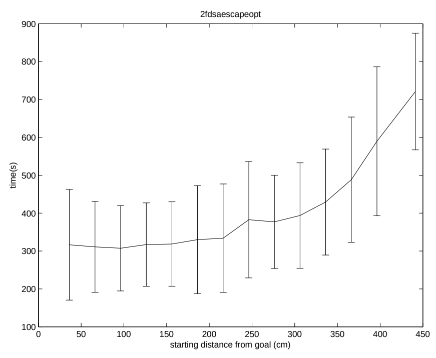
Table 5.8: Average return ratios for static environments for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.



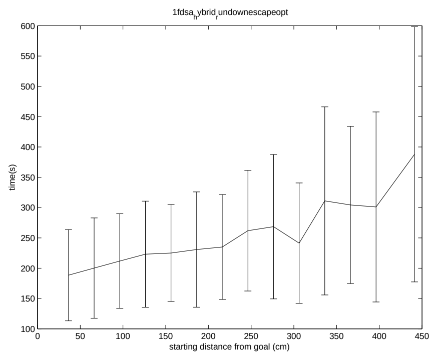
(a) Run-Down



(b) 1FDSA



(c) 2FDSA

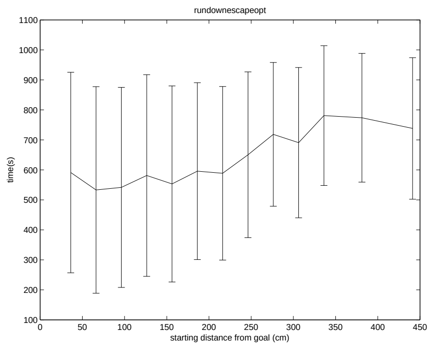


(d) Hybrid

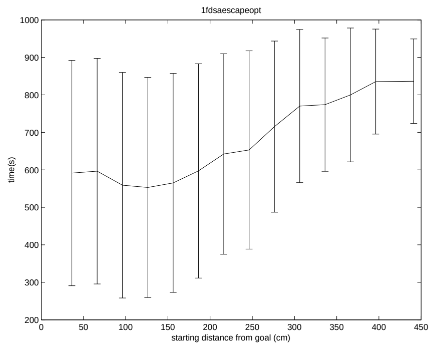
Figure 5.13: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. The algorithms are augmented with a method to escape non-snapshot optima. Experiments were conducted in static environments. The error bars indicate the standard deviation from the mean homing time.

Optimisation Method	Average Return Ratio
Run-Down	0.618
1FDSA	0.829
2FDSA	0.779
Hybrid Run-Down/1FDSA	0.814

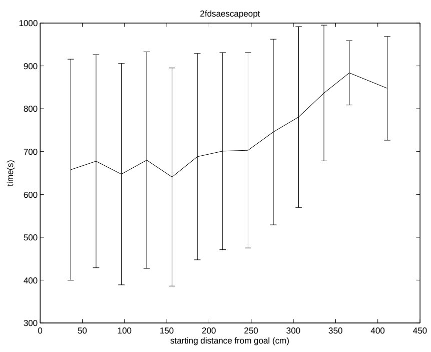
Table 5.9: Average return ratios for dynamic lighting environments for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.



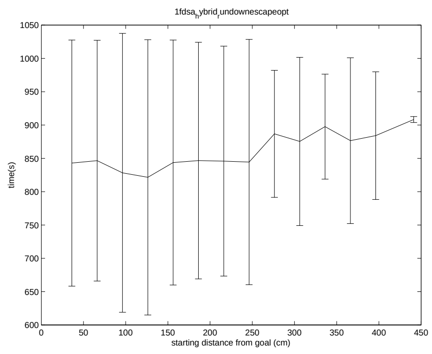
(a) Run-Down



(b) 1FDSA



(c) 2FDSA



(d) Hybrid

Figure 5.14: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. The algorithms are augmented with a method to escape non-snapshot optima. Experiments were conducted in dynamic illumination environments. The error bars indicate the standard deviation from the mean homing time.

Optimisation Method	Average Return Ratio
Run-Down	0.974
1FDSA	0.979
2FDSA	0.976
Hybrid Run-Down/1FDSA	0.921

Table 5.10: Average return ratios for a moving landmark environment for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. All pairs of average return ratios – except Run-Down, 1FDSA and 1FDSA, 2FDSA – are significantly different with 95% probability according to McNemar's test.

Optimisation Method	Average Return Ratio
Run-Down	0.812
1FDSA	0.917
2FDSA	0.898
Hybrid Run-Down/1FDSA	0.874

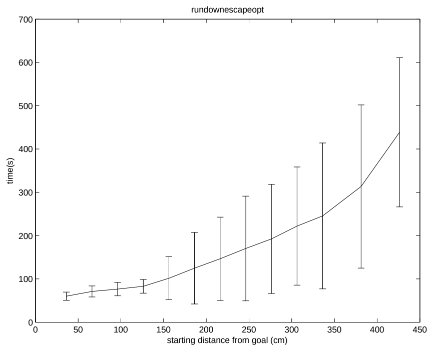
Table 5.11: Average return ratios for all (static and dynamic) environments for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. All pairs of average return ratios are significantly different with 95% probability according to McNemar's test.

optima. The illumination change causes the mutual image information between I_S and I_C at the snapshot location to be quite low, though an optimum still usually exists at the snapshot location as made clear in Chapter 3. There is probably no threshold value which will rid us totally of these false negatives while allowing us to detect with high probability non-snapshot optima.

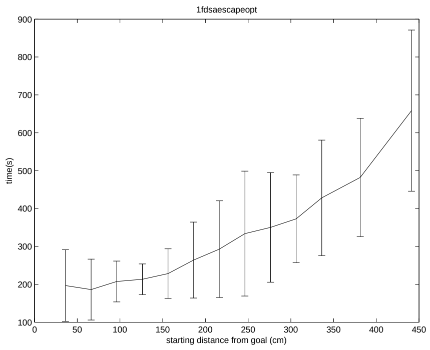
Surprisingly – unlike in past experiments – the hybrid algorithm is among the top performing optimisation algorithms in Table 5.9. We are not as yet certain why the hybrid algorithm performs relatively well in this experiment.

5.4.9 Experiments: Learning Gains

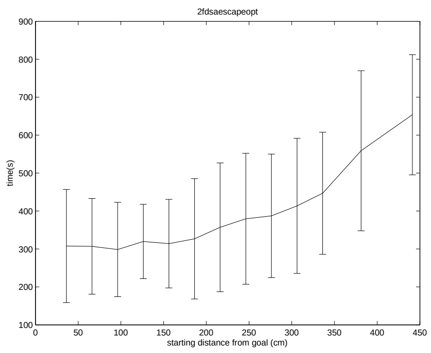
Spall [2003] suggests that the choice of gain series a_k is crucial in determining the success or failure of gradient ascent optimisation with gradients estimated with one-



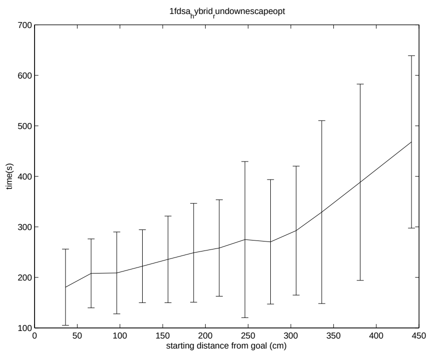
(a) Run-Down



(b) 1FDSA

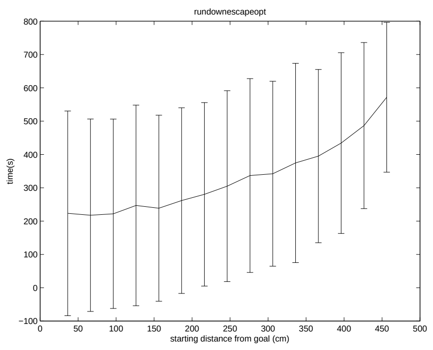


(c) 2FDSA

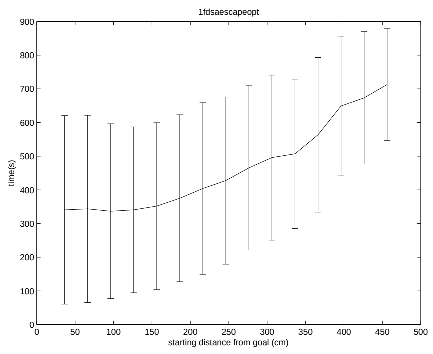


(d) Hybrid

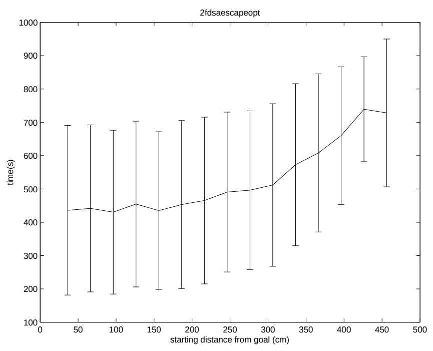
Figure 5.15: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. The algorithms are augmented with a method to escape non-snapshot optima. Experiments were conducted in an environment in which landmark locations changed between captures of snapshot and current images. The error bars indicate the standard deviation from the mean homing time.



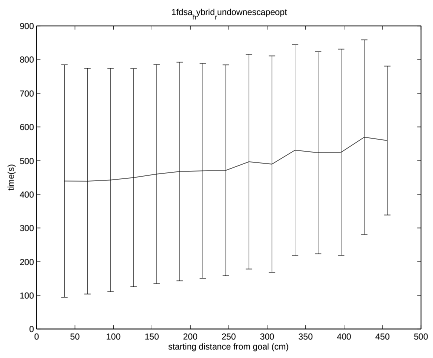
(a) Run-Down



(b) 1FDSA



(c) 2FDSA



(d) Hybrid

Figure 5.16: Mean homing time as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. The algorithms are augmented with a method to escape non-snapshot optima. All difference surface pairings were taken into account. The error bars indicate the standard deviation from the mean homing time.

or two-sided finite differencing. We wondered if we could find a more intelligent way of setting these gains than the rather limited trial-and-error procedure described in Section 5.4.5.2.

The nature of the homing problem demands the agent visits the snapshot location and travels away from it at least once before attempting to find it again by homing. We wondered if useful information about gains could be learned on this outbound path. It seems that ants (Judd and Collett [1998]) and honeybees (Lehrer and Bianco [2000]) learn something about the environment near the snapshot location for the purposes of homing while travelling away from it. A relatively simple though useful piece of information would be a mapping from mutual information to goal distance d. We could use this mapping to replace a_k as an estimate of the distance to travel in the direction of an estimated gradient. We published a similar idea in Szenher [2005a], though this work pertained to difference surfaces generated with the *RMS* metric.

After some trial and error, we found that Equation 5.8 gives a reasonable fit to goal distance d as a function of noisy mutual information values.

$$d = \frac{1}{B} tan(\frac{MI - C}{-A}) \tag{5.8}$$

Here, d is the agent's distance to the snapshot location and MI is the current mutual information value. A, B and C are free parameters which define the shape of Equation 5.8. C can be seen as the maximum mutual information signal encountered; A and B are empirically typically positive values close to zero. We have no theoretical basis for choosing Equation 5.8 to map MI value to goal distance. This functional form was empirically the best among many that we experimented with.

An example of the application of Equation 5.8 is given in Figure 5.17. We used Vardy's "Original" data set to generate the noisy MI data in Figure 5.17 (plotted in the figure as pluses). We simulated a robot leaving the snapshot location at x=60cm, y=30cm. It moved in the direction of the y-axis and sampled the noisy MI signal once every 10cm until hitting the boundary of the data set. Goal distance information was inferred by dead-reckoning. We used this data as input to a nonlinear function fitting algorithm. The algorithm seeks values for A, B and C which minimise the squared difference between the data and Equation 5.8. We used Matlab's **fminsearch** routine to perform this minimisation; **fminsearch** employs the Nelder-Mead optimisation procedure described in Section 5.3.2. The search begins at $A = 0.03$, $B = 0.05$ and $C = 1.5$; we noticed that most good fits had parameter values close to these settings. The solid line in Figure 5.17 represents the version of Equation 5.8 which best fits the data.

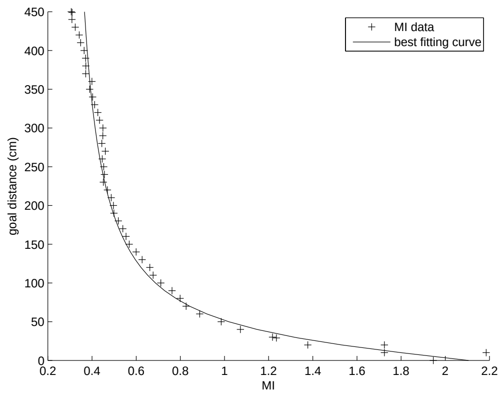


Figure 5.17: The pluses are (noisy MI, goal distance) data pairs to which we would like to fit Equation 5.8. See text for details about how this data was generated. The solid line plots the instantiation of Equation 5.8 which best fits the data: $d = \frac{1}{0.0268} tan(\frac{MI-2.1054}{-0.0204})$.

We repeated the experiments described in Section 5.4.5 using the 1FDSA and 2FDSA gradient estimation algorithms. Instead of setting a_k as described in Section 5.4.5.2, the agent learns parameters for Equation 5.8 for each snapshot location in the manner described above. When homing, the agent uses the learned function to transform a noisy MI reading into a distance to travel in the estimated gradient direction. The agent continues to avoid non-snapshot difference suface maxima as in the previous set of experiments. Results are given below.

5.4.10 Results and Discussion: Learning Gains

Table 5.12 shows that the learning of gains increases the probability that homing in static conditions will be successful. As we can see in Figure 5.18 this learning also speeds homing dramatically. The same observations can be made about learned gains in environments in which the locations of landmarks change between capture of snapshot and current images (see Table 5.14 and Figure 5.20).

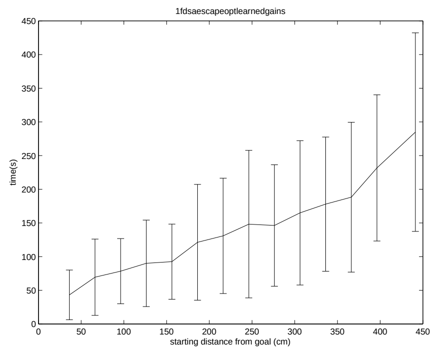
A quite different result is evident when illumination changes between capture of snapshot and current images (Table 5.13). The reason for these abysmal return ratios is that the mutual information at a given location is often much less when I_C and I_S are captured under different illumination conditions than when they are taken in the same illumination conditions. This reduction in mutual information leads Equation 5.8 to overestimate (often by a large amount) the agent's current distance to the snapshot

Optimisation Method	**Average Return Ratio**
1FDSA	0.974
2FDSA	0.992

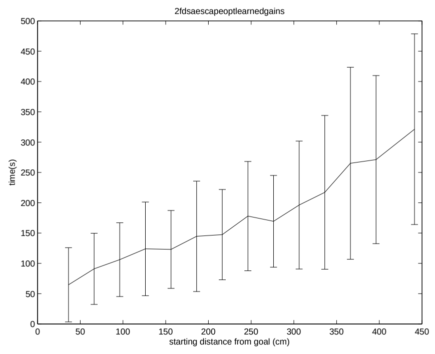
Table 5.12: Average return ratios for static environments for each 1FDSA and 2FDSA. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. The return ratios are not significantly different with 95% probability according to McNemar's test.

location. For example, suppose that I_S is captured at x=150cm, y=150cm and drawn from the "Original" data set. When I_C is captured at the nearby location x=150cm, y=120cm and also drawn from the "Original" data set, the value of $MI(I_S,I_C)$ is 1.59. When I_C is captured x=150cm, y=120cm but drawn from the "Winlit" data set, the value of $MI(I_S,I_C)$ is reduced to 0.72. In the former case, Equation 5.8 predicts that the goal is 32cm away from x=150cm, y=120cm, quite a good prediction. In the later case, Equation 5.8 predicts that the goal is 149cm away from the current location. Though the estimated gradient points towards the goal location in our example, the agent in the later case wildly overshoots the goal location, moving past it by more than one meter. This example is qualitatively similar to numerous other examples we examined.

If successive gradient estimates generally point towards the goal, then the overestimates of the agent's goal distance described above will cause the agent to move back and forth over the goal location but never in a tight enough cluster for the clustering stopping criterion discussed in Section 5.4.4 to be invoked. If a particular goal distance overestimate causes the agent to move far from the goal, then the gradient estimate at this relatively distant location will likely not point towards the goal. Thus, the agent will begin to take large steps in essentially random directions. The agent will come close to the goal again only by chance. In either case (whether the gradient points towards the goal or not), the homing run is likely to stop because the time-out criterion in Section 5.4.4 is met, not because the goal location is detected. We stop homing runs after more than 900 simulated seconds have elapsed. This is why we see in Figure 5.19 that the mean homing time for both 1FDSA and 2FDSA algorithms is about 900 seconds, regardless of starting distance from the goal.



(a) 1FDSA



(b) 2FDSA

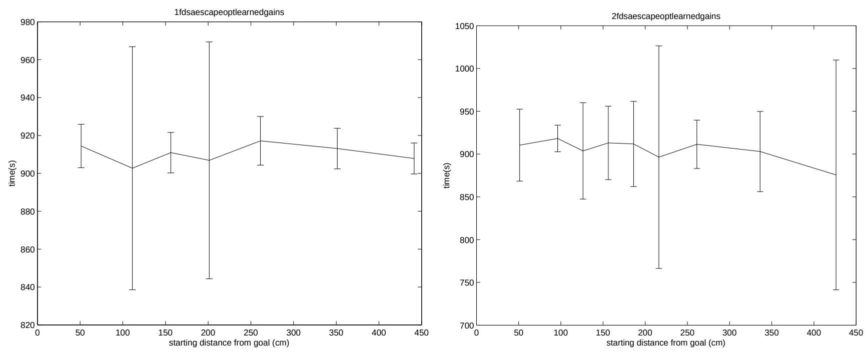
Figure 5.18: Mean homing time as a function of starting distance from the snapshot location for optimisation methods 1FDSA and 2FDSA. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. Experiments were conducted in static environments. The error bars indicate the standard deviation from the mean homing time.

Optimisation Method	**Average Return Ratio**
1FDSA	0.116
2FDSA	0.138

Table 5.13: Average return ratios for dynamic lighting environments for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. The return ratios are significantly different with 95% probability according to McNemar's test.

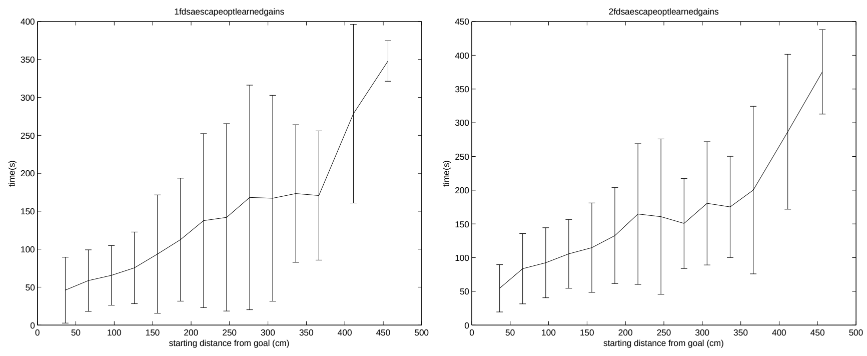
Optimisation Method	**Average Return Ratio**
1FDSA	0.986
2FDSA	0.993

Table 5.14: Average return ratios for a moving landmark environment for each optimisation algorithm considered in the chapter. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. The return ratios are significantly different with 95% probability according to McNemar's test.



(a) 1FDSA (b) 2FDSA

Figure 5.19: Mean homing time as a function of starting distance from the snapshot location for optimisation methods 1FDSA and 2FDSA. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. Experiments were conducted in dynamic illumination environments. The error bars indicate the standard deviation from the mean homing time.



(a) 1FDSA (b) 2FDSA

Figure 5.20: Mean homing time as a function of starting distance from the snapshot location for optimisation methods 1FDSA and 2FDSA. The algorithms are augmented with a method to escape non-snapshot optima. The distance to move in a gradient direction is learned. Experiments were conducted in an environment in which landmark locations changed between captures of snapshot and current images. The error bars indicate the standard deviation from the mean homing time.

5.5 Conclusions

In this chapter, we set out to identify appropriate algorithms for moving the homing agent so as to optimise a difference surface in both static and visually dynamic environments. In Section 5.3, we described homing algorithms which were appropriate to the task at hand: stochastic optimisation algorithms. We described in Section 5.3.2 a number of popular optimisation algorithms which are inappropriate for difference surface homing for a variety of reasons. It is hoped that the discussion of these inappropriate algorithms will constrain future visual homing researchers.

The optimisation algorithms which were ultimately selected were compared in simulated homing runs. A novel comparison criteria is defined in Section 5.4.5.1 which estimates the total time taken for the robot to carry out a homing run. To make the simulated homing runs as realistic as possible, the noise in mutual information due to sensor (imager and compass) noise was investigated in Sections 5.4.1 and 5.4.2. We found that realistic compass noise had a much more deleterious effect on the *MI* signal than did realistic imager noise. Both noise sources were present in our simulations. Sensor noise is rarely considered in robotic homing studies. Among the many visual homing papers reviewed in Chapter 2 only Möller et al. [2007] looked at the degradation of home vector precision as a function of compass noise. They found unsurprisingly that home vector precision decreases with increasing compass error. Möller et al. [2007] report that this problem can be alleviated by removing high-frequency components of current and snapshot images.

In our simulated homing experiments we compared the stochastic optimisers (1FDSA and 2FDSA) described in Section 5.3 with the "Run-Down" algorithm used for difference surface homing by Zeil et al. [2003]. Also considered was a novel hybrid between "Run-Down" and 1FDSA which was intended to overcome the drawbacks of each constituent algorithm. Though 2FDSA consistently outperformed the other optimisation algorithms in terms of homing success rate, its method of gradient estimation is quite time-consuming. The 1FDSA algorithm was consistently second-best in terms of homing success and consistently yields dramatically lower total mean homing time than 2FDSA. We thus conclude that 1FDSA is the best algorithm of those that we experimented with to use for difference surface homing.

The link between stochastic optimisation and difference surface homing which we forged here was very useful. Our investigation of the stochastic optimisation literature led us to methods for effectively choosing the gains which are an integral part of the

1FDSA and 2FDSA algorithms.

5.6 Future Work

Central place foraging insects often home to the same location – e.g. their nest – over and over again. Some applications – like docking and recharging at an inconspicuous wall mains – may require a homing robot to do something similar. Can a homing robot autonomously improve homing performance over multiple homing runs to a single snapshot location (perhaps in a dynamic environment)? In the context of the work done in this chapter, this may mean updating the values of a, c, A, α, and γ in Equations 5.5 and 5.6 over multiple homing runs.

Work related to this problem from which we may take inspiration has already appeared in the literature. Martinez-Marin and Duckett [2005] used a fast reinforcement learning algorithm to train a mobile robot to dock with a bottle sitting on an otherwise empty tabletop. Training was done online and it took less than an hour for the robot to learn a workable mapping of states to actions for this task. Weber et al. [2003] presented similar work, though the training was done in simulation. In these papers, the possible actions to take in a particular state formed a discrete, finite set. In the problem described above, the actions are related to continuous variables a, c, A, α, and γ. This difference may cause our problem to be more difficult to solve by reinforcement learning.

In addition to those described in Section 5.3, another optimisation scheme recommended by Spall [2003] is simulated annealing (SA). SA is as we shall see designed to avoid stalling in local optima relatively early in an optimisation process. Empirically, SA seems to work well when the function to be optimised is corrupted by noise. Like the optimisation schemes we used in our experiments, a simulated annealing algorithm selects a sequence of points in the search space. An SA algorithm may choose the successor to a point $\vec{x_k}$ at random, using gradient information or with some other criteria. If the function value at the successor $\vec{x_{k+1}}$ to $\vec{x_k}$ is better than (e.g. greater than) that at $\vec{x_k}$, then the new point is accepted; that is, the new point becomes the base for further explorations of the search space. If the function value at $\vec{x_{k+1}}$ is worse than that at $\vec{x_k}$, $\vec{x_{k+1}}$ may still be accepted with a certain probability. The probability depends on the current “temperature” of the annealing process; the higher the temperature, the greater the probability of accepting a “bad” move. The temperature is initially relatively high in early stages of the optimisation process and is reduced as k increases. The rate of cooling is controlled by the annealing schedule, a monotonically decreasing function of k. The success of simulated annealing is notoriously highly dependent on the choice of the initial temperature and on the annealing schedule Spall [2003]. We did not use SA here because we assumed a lot of human parameter setting would be required to make SA successful, limiting the autonomy of a homing robot. This assumption may not be valid though; SA should be explored in future.

We included realistic sensor noise in our simulation experiments to make these simulations as realistic as possible. We did not, though, inquire into how much sensor noise the homing process could tolerate before homing becomes difficult or impossible. We would like to do so in future. In particular, we would carry out homing trials using Vardy's "Original" data set using various snapshot and starting locations. For a particular snapshot and starting location, we would simulate a number of homing runs, each time increasing the Webcam and/or compass noise. We would measure homing success using the average return ratio criterion.

In our implementation in Section 5.4, estimation of the difference surface gradient at $\vec{x_C}$ using Equation 5.2 or Equation 5.1 requires the homing agent to move to two or more positions adjacent to $\vec{x_C}$ and evaluate mutual image information at each of these positions. Both the agent movement and MI evaluation take time. Möller and Vardy [2006] demonstrated that the difference surface gradient at $\vec{x_C}$ – when image similarity is measured with RMS – can be estimated without explicitly moving the agent from $\vec{x_C}$. These workers demonstrated that image $I(\vec{x'_C})$ can be inferred from $I(\vec{x_C})$ if $\vec{x'_C}$ is near $\vec{x_C}$. The inference depends on the assumption that all imaged objects are at an equal distance from $\vec{x_C}$; an assumption inspired by and similar to that made by the image warping algorithm (Franz et al. [1998b]). The image similarity between $I(\vec{x_C})$ and the snapshot and two or more warped images and the snapshot can be used to estimate the gradient. The creation of warped image $I(\vec{x'_C})$ takes significant computational effort, though, and multiple warped images must be created for each gradient estimate.

We believe the difference surface gradient can be estimated without agent motion about $\vec{x_C}$ and *without image warping*. Before discussing the method we introduce some new notation: let $MI_S(x,y)$ be the mutual image information between a snapshot image and the image captured at (x,y); the snapshot location is at the origin of the coordinate system in which (x,y) is defined. Our proposed method calls for the agent to capture two additional images near the snapshot after having stored the snapshot itself, one at $-c$ units from the snapshot location along the x-axis and the other at $-c$ units from the snapshot location in the y-direction. These images need be captured only once,

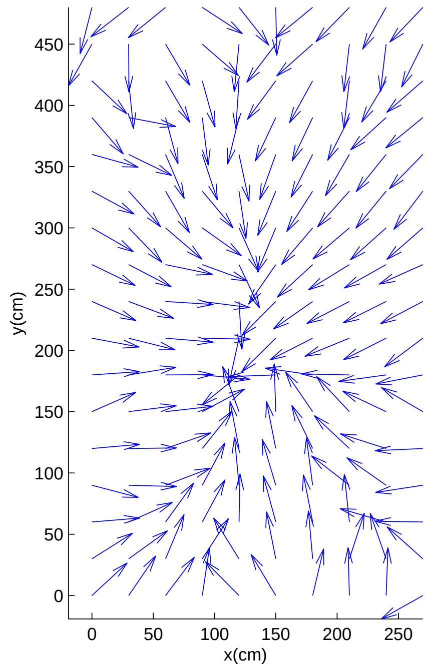


Figure 5.21: Difference surface gradient directions (shown as unit vectors) estimated with Equation 5.9. We set c at 60cm. The snapshot is located at x=120cm, y=180cm. All images were taken from Vardy's "Original" data set.

probably as the agent leaves the snapshot location for the first time. We shall call these images $I(\vec{x_{S_x}})$ and $I(\vec{x_{S_y}})$. The gradient at $\vec{x}$ is then estimated with the following equation

$$\mathbf{g}(x,y) = \left[\; \frac{MI_{S_x}(x+c,y) - MI_S(x,y)}{c} \; \frac{MI_{S_y}(x,y+c) - MI_S(x,y)}{c} \; \right] \tag{5.9}$$

Note that the agent does not move from its current location in order to compute the gradient with this formula. We compare this with the the one-sided difference gradient estimated of Equation 5.2 rewritten using our new notation

$$\mathbf{g}(x,y) = \left[\; \frac{MI_S(x+c,y) - MI_S(x,y)}{c} \; \frac{MI_S(x,y+c) - MI_S(x,y)}{c} \; \right] \tag{5.10}$$

Equations 5.9 and 5.10 yield the same answer if $MI_{S_x}(x+c,y) = MI_S(x+c,y)$ and $MI_{S_y}(x,y+c) = MI_S(x,y+c)$. We assume these equalities hold if c is relatively small.

We estimated difference surface gradients using Equation 5.9 using images from Vardy's data set. The resulting gradient vector directions (shown as unit vectors) are shown in Figure 5.21. The mean deviation between the gradients depicted in Figure 5.21 and the true home vector was approximately zero degrees (-1.7 degrees) with a standard deviation of 24.2 degrees. By contrast, the mean deviation between the gradients computed with Equation 5.10 was -0.5 degrees with a standard deviation of 29.9 degrees. Though gradient directions produced by our proposed method are on average better than those produced by Equation 5.10, we cannot claim our proposed method is generally better; more experimentation must be done. The mean deviation between the gradients computed with each method was 20.9 degrees with standard deviation 18.6 degrees, indicating that Equations 5.9 and 5.10 produce gradient vectors in roughly the same direction.

Chapter 6

Robotic Experiments

6.1 Introduction

Much of our work in the previous chapters was done in simulation using Vardy's image data sets, described Chapter 3. We would like to replicate our reported findings in a different environment to lend credence to the idea that difference surface homing is generally applicable. To this end, we carried out a number of "live" robotic trials in our laboratory environment. We describe the robot and imaging system we assembled for these experiments in Section 6.2.1. We created a visual tracking system – described in Section 6.2.2 – to estimate the robot's pose (position and orientation) during experiments. Pose information is required for both post-experiment analysis and to provide the robot's current orientation to its homing algorithm. Our experiments and their results are detailed in Section 6.3; conclusions follow in Section 6.4. We discuss future work in Section 6.5.

6.2 Materials

6.2.1 Description of Robot

We used a Koala Silver Edition mobile robot in our live experiments (see Figure 6.1). To capture panoramic images of the environment, the robot was equipped with a Creative Labs CT4840 Video Blaster Webcam imaging a panoramic mirror. The Webcam's gain control mechanism was turned off. Since the Webcam's gain control algorithms are unpublished, having gain control active would have affected our experiments in ways which we could not predict or account for.

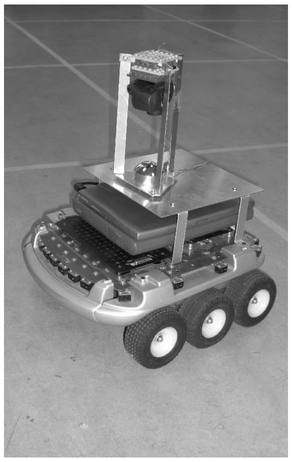


Figure 6.1: Image of Koala Silver mobile robot mounted with Acer 313T laptop computer and panoramic imaging rig used in our “live” homing experiments.

The Koala also carried a small Acer 313T laptop computer. We chose this laptop because it is one of the few affordable laptops which weighs less than one kilogram, a requirement as the Koala’s maximum weight capacity is about 3kg. The laptop computes with an Intel Pentium 266 MHz processor and has 32MB of RAM. The laptop’s Lithium Ion battery allows for about two hours of off-mains use. We installed SlackWare Linux 11.0 running the 2.6 Linux kernel on the laptop.

We originally intended that the Koala equipped with the Acer laptop would home completely autonomously. The laptop would run a program to capture panoramic images from the Webcam, compute mutual image information, generate appropriate homing vectors, and cache data for future analysis. Since, though, the tracking system required us to manually interact with panoramic images on our desktop computer during homing (see Section 6.2.2), we had to change this plan.

We could have connected our Webcam directly to our desktop computer via a USB (Universal Serial Bus) extension cord of up to 5 metres. This cord, though, would have appeared in many of the images captured by the Webcam, essentially playing the role of a moving landmark and thereby making homing more difficult. The cord would have also become tangled in the robot’s wheels during homing, requiring frequent human intervention.

In the final system, the Acer laptop was equipped with a Cisco Aironet 340 wireless network card, allowing communication with the University’s wireless network. We installed OpenVPN on the laptop to establish a link between the wireless network and our Desktop-based network account. Images captured with the Koala’s panoramic imaging rig were sent via the wireless network to our desktop computer (a Dell Pentium 4 Optiplex). Image similarity was calculated on the desktop computer. Wireless image transmission took about 0.2 seconds per image. Images were captured at a resolution of 320x240 pixels and stored in the jpeg (Joint Photographic Experts Group) format as this format compresses images for relatively fast wireless transmission.

Difference surface homing requires an estimate of the robot's orientation in some external reference frame. This is because the robot almost certainly has a different orientation at S than it does at C. I_C must be rotated in software to account for this orientation difference otherwise measuring the similarity between I_S and I_C with mutual image information would be meaningless.

We originally intended to use a digital magnetic compass in our homing experiments to measure orientation. Digital magnetic compasses with serial and/or USB (universal serial bus) connectability are typically quite expensive (see e.g. http://www.oceanserver-store.com/compass.html). We were fortunate to find an inexpensive model manufactured by Silicon Laboratories, the F350. This compass provides a tilt-corrected azimuthal compass signal, as well as the tilt of the compass and the current temperature. We wrote a daemon program in C to continually sample and decode the compass signal; we shall make this code available online.

Unfortunately, we eventually found that the F350 is unsuitable for our needs. The signal is highly sensitive to magnetic interference, which is common indoors. Even when we attempted to shield the compass from such noise, the azimuth reading exhibited a large standard deviation when travelling on a straight path; the mean error was non-zero as well. The tilt signal is quite reliable indoors, though, as is the temperature.

Until we can afford a more sophisticated digital compass, we must fall back on our tracking system (see Section 6.2.2) for directional information. This as we shall see limits the autonomy of the robot, but we are left with little choice.

6.2.2 Tracking System

We needed to track the robot during homing to provide a record of the robot's movement for post-experiment analysis. As our magnetic compass failed to work indoors (see Section 6.2.1), we also used our tracking software to provide an estimate of the robot's orientation at each homing step so that I_C could be rotated to account for changes in robot orientation between S and C.

We originally intended to track the movement of the robot during a homing run

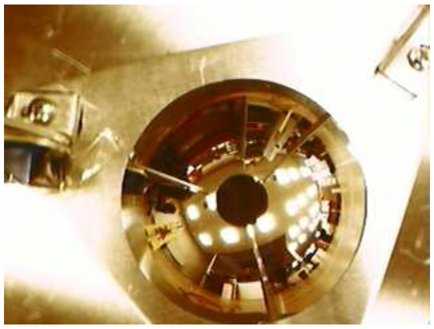


Figure 6.2: Sample image from our panoramic imaging system used by our tracking system.

with a tether tracking system designed by Robert MacGregor, senior technical officer in the School of Informatics at the University of Edinburgh. This system consists of up to four base stations, each of which holds a spool of fishing wire. The end of each length of wire is attached to a point on the robot. If the robot moves away from a base station the wire unwinds. Movement towards the base station will cause the wire to respool, as the base station pulls the wire towards it with a small, constant force. Each base station maintains an accurate and precise estimate of the length of wire currently unspooled (i.e. the distance of the robot from the base station). In an initial calibration phase, the position of each base station (in a coordinate system with a particular station at the origin) is estimated. Given these positions and the distance of the robot to each base station, simple trigonometry can be employed to infer the robot's position in the aforementioned coordinate system. Two base station locations/robot-distance measurements are required for this calculation. If more than two base stations are available, several estimates (one for every possible pair of base stations) of the robot's position are made and the average position is reported. The robot's orientation can be estimated by calculating the best-fitting line through several successive position estimates as the robot moves in a particular direction.

As we mentioned earlier, the tether tracking system proved unsuitable for our purposes. The fishing wire must be tethered to the highest point on the robot otherwise the wire becomes tangled on the robot when the robot rotates. The highest point on our robot is the top of the panoramic mirror rig. The constant pulling force of the base stations causes the rig to bend, distorting the captured image in ways that are difficult to correct for in software. The pulling force also causes a torque around the centre of mass of the robot, frequently causing the robot to topple over.

We designed and built a visual tracking system to replace the tether tracker. The visual tracker relies on a surveying technique called resection (McCaw [1918]) to determine the robot's current position (x_R, y_R). Resection takes as input the ego-centric bearing of three landmarks whose positions in some suitable reference frame are known. The resection algorithm itself is a rather involved trigonometric procedure; we refer the reader to McCaw [1918]) for details. Once the agent's position is fixed, its bearing θ (i.e. rotation counter-clockwise from the x-axis of the coordinate system in which the landmark locations are defined) can be computed using the known location of one of the landmarks (x_L, y_L). We use the following formula to compute the robot's bearing:

$$\theta = tan^{-1}\left(\frac{y_L - y_R}{x_L - x_R}\right) \tag{6.1}$$

Using Equation 6.1 we can compute three bearing estimates using the three different landmark locations used in the resection process. Multiple bearing estimates are averaged using the circular mean (Batschelet [1983]) formula.

As resection requires the locations of recognised landmarks, we placed six landmarks in our experimental laboratory environment. Our colleagues – for an experiment of their own – marked out a 4m x 5m grid on our laboratory floor. The grid was marked in masking tape spaced at one metre intervals. One can see a portion of this grid in Figure 6.1. We placed the six landmarks along the periphery of the grid, on intersection points of the grid. In this way, we were easily able to localise these landmarks.

We used images taken with the robot's panoramic imaging rig to determine the robot-centric bearing of these landmarks. See Figure 6.2 for a sample image used by our tracking system. We chose landmarks that were visually inconspicuous and, when possible, part of a typical laboratory so that it could not be claimed that we adulterated the laboratory environment to improve the performance of our homing robot. One landmark was, for example, a waste basket and another was a poster cylinder standing on end. It proved difficult to automatically identify these landmarks in our images. Thus, we had to perform the identification manually. As we described in Section 6.2.1, each time a pose estimate was required, the robot's on-board computer captured the current panoramic image and sent it wirelessly to our desktop computer. Our tracking program displayed the image on screen and we mouse-clicked on each visible landmark.

Given their locations in pixel-coordinates, the tracking program inferred the robot-centric bearing of each selected landmark. As we mentioned earlier, the resection algorithm requires three landmark bearings to calculate the robot's position. If more than three landmark are visible, the tracking program uses the RANSAC (RANdom

SAmple Consensus) algorithm (Fischler and Bolles [1981]) to determine the subset of landmark bearings which gives the most consistent set of position estimates. The mean of these position estimates is taken to be the robot's true position. The mean position is then used to compute the agent's bearing using Equation 6.1. We use an extended Kalman filter (Welch and Bishop [2006]) to further filter out noise in the pose estimate. The filter modifies the position estimate produced by the RANSAC method by taking into account the tracker's previous position estimate and the robot movement commands issued at this previous location.

As the panoramic imaging rig is somewhat unstable, the location of the panoramic mirror in the images provided by the Webcam changes slightly over time. At the beginning of each homing run, the tracker finds the centre and boundary of the imaged mirror using the Hough transform (Fisher et al. [1996]). As the Hough transform is computationally expensive, we do not reestimate the location of the mirror in the Webcam image during a homing run. The assumption that the relationship between the mirror and the Webcam does not change during a homing run seems to be valid in most cases. We also use the mirror location information to mask out non-mirror image segments when computing mutual image information.

We wanted to determine how well our visual tracker estimates the robot's pose. We first tested the tracker in simulation. The simulated environmental area contained six landmarks, represented by the blue pluses in Figure 6.3. This landmark arrangement was quite similar to the one we used in our "live" robotic trials. The agent started at position x=200, y=200 in this arena, oriented at 45 degrees. One-hundred movement steps were simulated. In each step, the agent made a random decision to move forward by 10 distance units or to rotate by 90 degrees counterclockwise. Forward movement occurred with 85% probability. After each movement, the agent estimated its pose using the method described above. The pose estimate requires the agent to measure the bearing of each sensed landmark. To mirror real-world conditions, we added random noise to these bearing measurements. Each noise sample was drawn independently from a zero mean Gaussian distribution with a standard deviation of 1.5 degrees.

The result of one run of our simulation is shown in Figure 6.3. The solid black line in this figure indicates the robot's true path and the red line our visual tracker's estimate of the robot's position. The position estimate had approximately zero mean error (0.16 distance units in the x-direction and 0.05 distance units in the y-direction). The standard deviation of the position estimate was small: 1.56 distance units in the x-direction and 1.75 distance units in the y-direction. The robot's estimated bearing is

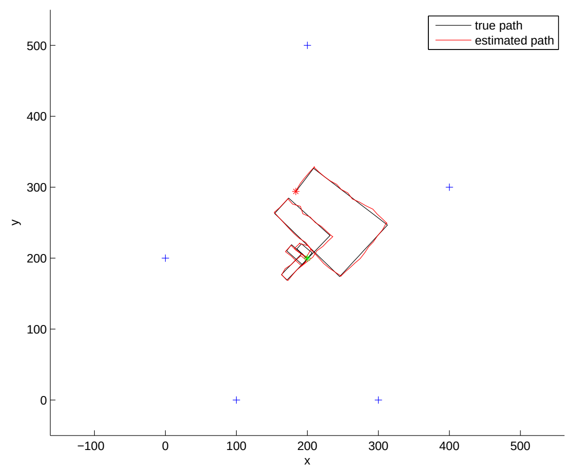


Figure 6.3: Result of a simulated test of our visual tracking system. The start position of the simulated agent's path is depicted as a green star, the end position a red star. The true path of the simulated agent is shown as a solid black line. The agent's estimated location is shown as a red line. Landmarks are depicted as blue pluses.

not shown in Figure 6.3. The mean bearing error was -0.09 degrees with a standard deviation of 0.66 degrees. The distribution of the bearing error was approximately Gaussian, with kurtotis = 2.89 and skewness = -0.17. We re-ran the simulation several times; the results were similar in every run.

We next tested the visual tracker in our laboratory environment. We could not assess the tracker's ability to estimate position since, using the coarse grid laid out on our laboratory floor, we had only a rough sense of the agent's true position. To measure the tracker's ability to estimate bearing, we drove the robot along a straight 2.5m track on the laboratory floor. The robot stopped approximately every 5cm (as measured by dead reckoning), at which point a pose estimate was made. The estimated position of each data capture point is shown in Figure 6.4. We drove the robot along two other 2.5m tracks in different parts of our laboratory and gathered 50 equally spaced pose estimates along these tracks as well. We combined the data of the three tracks and found that the standard angular deviation from the mean is 0.71 degrees, broadly in line with our simulation results. The skewness is 0.09 (small, as in the simulation) but, unlike in the simulation, the kurtosis of the data is 6.54, indicating a non-Gaussian distribution with a relatively high peak at the mean.

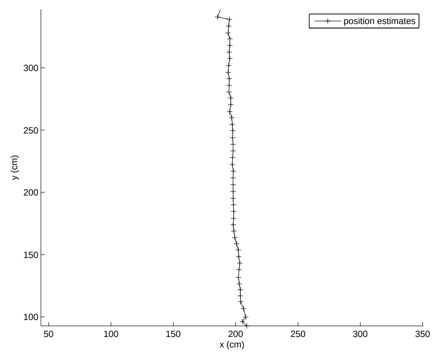


Figure 6.4: Result of a test of our visual tracking system in our laboratory environment. We drove the robot along a straight 2.5m track, taking pose estimates every 5cm (for a total of 50 estimates). The position estimates are depicted as black crosses.

6.3 Experiments and Results

In our first experiment, we wanted to determine which image similarity measure – *RMS* or *MI* – yields better homing performance in static environments. We measured homing performance using the criteria used in previous chapters: total homing time and return ratio. In this experiment, our robot homed starting from either one or two metres away (approximately) from the snapshot location. All overhead lights in our laboratory were turned on during capture of both snapshot and current images, resulting in constant illumination over the entire experimental area. We selected snapshot locations which were uniformly distributed in our test environment and starting locations uniformly distributed (when possible) around the snapshot positions. For every pairing of start location and snapshot location, we performed two homing runs, one using *RMS* to measure image similarity and the other using *MI* to measure image similarity. For each similarity measure, we performed eight one-metre (i.e. starting one metre from the snapshot location) and twelve two metre tests, for a total of forty homing runs. We used the “Run-Down” algorithm to home in these experiments, setting the interval between difference surface samples at 15cm for the one-metre experiments and 25cm for the two-metre experiments. These inter-sample distances were determined empirically after observing a few unrecorded homing runs at various starting distances in our environment.

The results of this experiment are summarised in Table 6.1. In the one-metre experiments, homing with *RMS* and *MI* was roughly equivalent, though *MI* had a higher return ratio. The disparity between the return ratio in the two-metre tests was greater,

Similarity Measure	F_{Total}	D_{Total} (cm)	θ_{Total} (deg)	T_{Total} (s)	RR
	Goal Distance $\approx$ 1 metre				
RMS	28.4 [13.3]	426.0 [200.0]	1188.0 [373.3]	97.6 [39.2]	0.75
MI	27.1 [7.4]	407.1 [111.1]	1015.7 [253.0]	89.1 [23.0]	0.88
	Goal Distance $\approx$ 2 metres				
RMS	22.6 [2.7]	565.6 [66.7]	888.8 [169.7]	104.0 [14.3]	0.75
MI	28.8 [12.0]	720.5 [301.2]	1251.8 [654.5]	136.6 [60.2]	0.92

Table 6.1: Summary of results for homing runs comparing MI and RMS image similarity measures in static conditions. Mean values over all homing runs are reported, with standard deviations in square brackets. Only successful homing runs were used to calculate the summary statistics other than, of course, the return ratio. In this table F_{Total} is the total number of image similarity computations undertaken during a homing run; D_{Total} is the total linear distance travelled during a homing run; θ_{Total} stands for the sum of all angles turned by the homing agent during a homing run; T_{Total} is the total time taken during a homing run; and RR is the return ratio. As in our simulations in Chapter 5, the robot translated at 8 cm/sec and rotated at about 29 degrees/sec. A single computation of MI and RMS took roughly the same amount of time, about 0.1 seconds. When calculating T_{Total}, we ignored the time required to transmit images over the wireless network and the time required to estimate the robot's pose using the visual tracker as these are implementation dependent.

with *MI* reaching the goal 92% of the time. As might be expected, the total time required to home in the *MI* experiments increases with increasing starting distance from the goal. Since the sample interval is greater for the two-metre tests, the increase in mean function evaluations and mean total translation did not increase dramatically. Strangely F_{Total} and θ_{Total} actually decrease in the two-metre *RMS* experiments as compared to the one-metre *RMS* experiments. We believe that this is because in several of the one-metre *RMS* homing runs the robot became temporarily trapped in local optima before finally reaching the goal.

In our second experiment, we sought to determine which similarity measure – *RMS* or *MI* – yields better homing performance in an environment in which lighting changed between snapshot and current image capture. As above, the robot starts homing from either one or two metres away (approximately) from the snapshot location. We selected snapshot locations which were uniformly distributed in our test environment and starting locations uniformly distributed (when possible) around the snapshot positions. For every pairing of start location and snapshot location, we performed two homing runs, one using *RMS* to measure image similarity and the other using *MI* to measure image similarity. All snapshot images were captured with all overhead lights turned on. Current images were captured with half of the overhead lights directly above the experimental area turned off; in some experiments we turned the left bank of lights off and in others we turned the right bank of lights off when capturing current images.

For each similarity measure, we performed eight one-metre and eight two-metre tests, for a total of thirty-two homing runs. We used the “Run-Down” algorithm to home in these experiments, setting the interval between difference surface samples at 15cm for the one-metre experiments and 25cm for the two-metre experiments. These inter-sample distances were determined empirically as above.

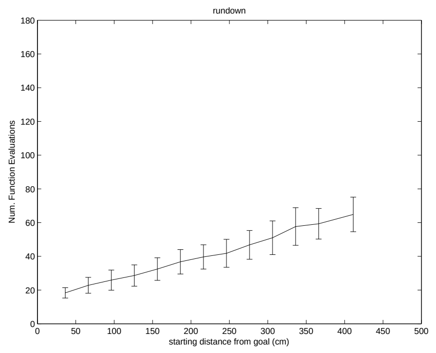
The results of our dynamic illumination experiments are reported in Table 6.2. As measured with the return ratio *RR*, homing using *MI* to measure image similarity is clearly more robust than when using *RMS*. This agrees with the results given in Chapter 3. In the two-metre tests, homing with *RMS* was only successful 38% of the time compared with a 75% return rate for *MI*. We observed that while homing using *RMS*, a robot starting in a relatively unlit portion of the environment would move toward the more brightly lit part of the arena rather than towards the snapshot location. This observation jibes with our finding – reported in Chapter 3 – that *RMS* difference surface homing will cause the homing agent to move so as to equalise the mean intensities of the current and snapshot images. The other criteria listed in Table 6.2 indicate that suc-

Similarity Measure	F_{Total}	D_{Total} (cm)	θ_{Total} (deg)	T_{Total} (s)	RR
	Goal Distance $\approx$ 1 metre				
RMS	21.6 [3.2]	540.0 [80.2]	864.0 [225.9]	99.8 [17.7]	0.62
MI	19.7 [6.5]	492.9 [161.8]	810.0 [207.8]	91.9 [27.7]	0.88
	Goal Distance $\approx$ 2 metres				
RMS	23.0 [4.6]	575.0 [114.6]	840.0 [137.5]	103.5 [19.6]	0.38
MI	24.2 [6.0]	604.2 [151.2]	870.0 [245.9]	108.3 [27.6]	0.75

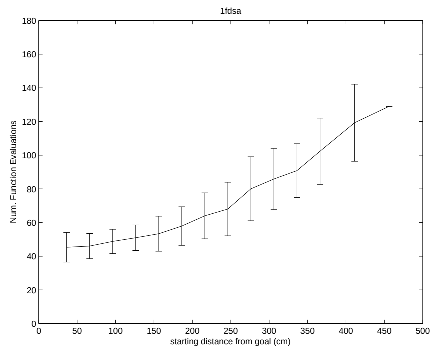
Table 6.2: Summary of results for homing runs comparing *MI* and *RMS* image similarity measures in dynamic illumination conditions. Mean values over all homing runs are reported, with standard deviations in square brackets. Only successful homing runs were used to calculate the summary statistics other than the return ratio *RR*. For definitions of F_{Total}, D_{Total}, θ_{Total}, and T_{Total} see the caption of Table 6.1.

cessful homing with *MI* takes roughly the same amount of time as successful homing with *RMS*.

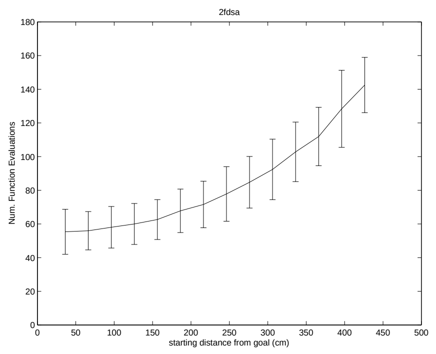
Given the relatively inferior return ratio of “Run-Down” reported in Table 5.7, one may be surprised that we used this algorithm for difference surface optimisation in the experiments described above. We wanted each “live” homing run to be as speedy as possible since the battery life of the Koala and the Acer laptop are quite limited, both lasting for a little over an hour of constant use. Once drained, each battery requires several hours to recharge. In trial homing runs, the majority of the time (about 90 percent) was spent making the semi-manual pose estimation described in Section 6.2.2. We performed a pose estimate each time the difference surface was sampled because the orientation of the robot is required to compute image similarity. We chose “Run-Down” to compare *MI* and *RMS* difference surfaces in the experiments above because “Run-Down” – as demonstrated in Figure 6.5 – required fewer difference surface evaluations as a function of starting distance than 1FDSA, 2FDSA or Hybrid in our simulation experiments. The difference in function evaluations is often dramatic; for example, Figure 6.5 indicates that “Run-Down” always requires fewer and often less than half the number of difference surface evaluations on average than 1FDSA for any given starting distance. Though the performance of “Run-Down” is relatively poor in Table 5.7, in absolute terms its mean return ratio was only a few percentage points worse than 1FDSA. For this reason – and to make our live experiments as fast as possible – we chose to use “Run-Down” in the experiments described above.



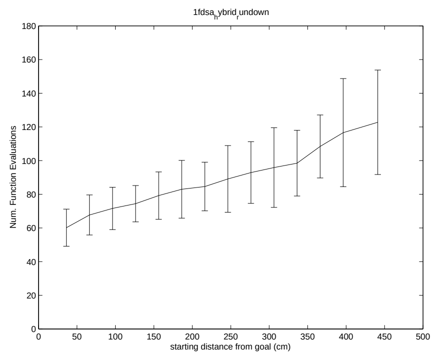
(a) Run-Down



(b) 1FDSA



(c) 2FDSA



(d) Hybrid

Figure 6.5: Mean number of difference surface evaluations as a function of starting distance from the snapshot location for optimisation methods Run-Down, 1FDSA, 2FDSA and Hybrid. The error bars indicate the standard deviation from the mean number of difference surface evaluations. All simulated homing runs used to make Figure 5.12 in Chapter 5 were used to create these graphs.

Opt. Algorithm	F_{Total}	D_{Total} (cm)	θ_{Total} (deg)	T_{Total} (s)	RR
	Goal Distance ≈ 1 metre				
RunDown	27.1 [7.4]	407.1 [111.1]	1015.7 [253.0]	89.1 [23.0]	0.88
1FDSA	24.4 [9.1]	983.9 [329.8]	1258.9 [394.6]	169.4 [55.7]	1.00
	Goal Distance ≈ 2 metres				
RunDown	29.8 [13.1]	743.8 [328.6]	1158.8 [687.8]	136.4 [65.8]	1.00
1FDSA	21.0 [0.0]	859.3 [0.0]	1236.8 [116.1]	152.7 [4.1]	0.75

Table 6.3: Summary of results for homing runs comparing “Run-Down” and gradient descent with one-sided finite differencing (1FDSA) in a static environment. Mutual information was used to measure image similarity. Mean values over all homing runs are reported, with standard deviations in square brackets. Only successful homing runs were used to calculate the summary statistics other than the return ratio RR. For definitions of F_{Total}, D_{Total}, θ_{Total}, and T_{Total} see the caption of Table 6.1.

In our final experiment, we sought to determine whether “Run-Down” or gradient descent with one-sided finite differencing (1FDSA) was the better optimisation algorithm. As above, the homing robot started from either one or two metres away (approximately) from the snapshot location. We selected snapshot locations which were uniformly distributed in our test environment and starting locations uniformly distributed (when possible) around the snapshot positions. For every pairing of start location and snapshot location, we performed two homing runs, one using “Run-Down” to home and the other using 1FDSA. All snapshot and current images were captured with all overhead lights turned on. Image similarity was computed with mutual information.

For each optimisation algorithm, we performed eight one-metre and eight two-metre tests, for a total of thirty-two homing runs. For the “Run-Down” algorithm, we set the interval between difference surface samples at 15cm for the one-metre experiments and 25cm for the two-metre experiments. These inter-sample distances were determined empirically as above. When homing with 1FDSA, we set c_k at 25cm for all k and $a_k = \frac{100}{k+1} cm$. These gains were empirically determined in the same way that we fixed the “Run-Down”’s inter-sample interval. To gather the difference surface samples required to make a gradient estimate with 1FDSA, we used the sequence of movement commands described in Section 5.3.1.

As we can see in Table 6.3, 1FDSA is more successful at a starting distance of one metre than is “Run-Down.” The 1FDSA algorithm, though, takes dramatically more time to home than does “Run-Down.” This disparity is due to the movement that the robot has to undertake in order to compute each gradient estimate. Interestingly, 1FDSA requires less time to home at two meters than at one meter; we as yet do not have an explanation for this. 1FDSA at two meters is still slower than “Run-Down,” though.

The fact that – as reported in Table 6.3 – “Run-Down” performs better than 1FDSA at 2 metres runs counter to our expectations given the simulation results of Chapter 5. We examined in detail the records of the two homing runs in which 1FDSA fails and “Run-Down” succeeds. In one case, the 1FDSA algorithm guides the agent closer to the snapshot location after every gradient estimate as is desired. Unfortunately, in the later part of the homing run, the agent’s distance from the snapshot locaction decreases only slightly (by a few centimetres) in each iteration of the optimisation algorithm since it moves in a zig-zag fashion towards the goal. This behaviour is interpreted as clustering around a fictitious difference surface optimum so the agent halts at about 50cm from the snapshot location. The clustering stopping criterion fails in this instance. In the case of the second 1FDSA failure, the initial gradient estimate is fairly poor; the agent’s one metre move in this direction brings it only slightly closer (1.8 metres) to the goal position than the starting distance of 2 metres. The new location is a local optimum so the agent spends the remainder of the homing run meandering near this location. This example highlights a weakness with the 1FDSA algorithm: if at least one of the first few gradient estimates is poor, then the agent will move only slightly nearer or indeed away from the goal location, potentially to a location which is more difficult to home from than the starting location. “Run-Down” does not make such large steps so turns towards the snapshot location before finding this local optimum.

Some of the standard deviations for two-metre 1FDSA reported in Table 6.3 are zero. This may seem odd at first glance. The reason for it is that the homing algorithm checks if the “clustering” stopping criterion is true starting on the seventh iteration of the 1FDSA algorithm. In all of our two-metre 1FDSA tests, the stopping criterion happened to be true on reaching the seventh iteration of the algorithm. The number of difference surface evaluations and the linear distance travelled (as measured by dead reckoning) is directly proportional to the number of iterations of the 1FDSA algorithm. Thus, these values were equal to 21.0 and 859.3cm, respectively, in all successful two-metre 1FDSA runs.

6.4 Conclusions

Some results reported in Section 6.3 are broadly in agreement with our simulation experiments described in previous chapters. In static conditions, visual homing using *RMS* to measure image similarity performs roughly as well as when *MI* is used to measure similarity. When illumination changes between the capture of the snapshot and current images, homing using *MI* is dramatically more robust. We observed that several of the *RMS* homing runs in dynamic lighting fail in the manner predicted in Chapter 3: the robot moves so as to equalise the mean intensity of snapshot and current images, rather than towards the snapshot location. Unlike in our simulation work, the choice between optimisation algorithms – specifically "Run-Down" and 1FDSA – yields ambiguous results. The 1FDSA algorithm is successful 100% of the time when homing starts 1 metre from the snapshot location but is less robust than "Run-Down" when starting from 2 meters. 1FDSA consistently takes more time to home than does "Run-Down" due to the expensive gradient estimation carried out by the former algorithm.

Homing success rates reported in this chapter are generally on par with those garnered from simulated experiments and reported on in Chapter 5. We found though that in our "live" experiments the robot reliably homed from up to 2 metres from the snapshot locations but in our simulated experiments – using Vardy's image data set – the homing agent could reach home from a greater starting distance (up to 4.5 metres). There are a few factors which could account for this discrepancy. We may not have accounted for all sources of sensor noise in our simulated experiments. In particular, as we noted above, the Webcam image of the mirror moves over time due to physical instability in our panoramic imaging rig. We corrected for this movement at the beginning of every homing run in "live" experiments but it might have been better to apply the correction *during* homing runs as well (though this would have required significant computational effort). Our panoramic mirror was upturned, reflecting large parts of our laboratory's ceiling whereas Vardy's mirror was downturned so that floor of his laboratory took up a large part of the images which he captured. Whereas the image of the floor changes very little as the agent moves, the image of the ceiling (with its lights and repeated tile pattern) changes more rapidly. Relatively rapid change in image content leads to relatively rapid decrease in mutual image information as the agent leaves the snapshot location. Finally, of course, the laboratory environment in which Vardy captured his images was different than the laboratory in which we captured ours. If

Vardy's environment contained imaged objects which were larger and farther from the robot than those in our environment, mutual information would generally be usable over a wider area in Vardy's lab.

6.5 Future Work

We described in this chapter the difficulty we had in attaining a global compass reference to match the orientations of current and snapshot images. We ended up using our tracking software to provide an orientation signal to the mobile robot. This of course limits the autonomy of the robot.

There are a few alternatives to our compassing solution which we could try in future. We could use the zero phase representation (ZPR) of images described in Chapter 2. The ZPR uses phase information from the discrete Fourier transform of images to rotate any image taken in a particular environment to a single canonical orientation. We could also use angle histograms (Hsieh et al. [1997]) to align current and snapshot images. In computing an angle histogram, snapshot and current images would first be edge-filtered and the orientation of each edge in each image would be computed. Then, the orientation of each edge in the current image would be compared to the orientation of all edges in the snapshot image. Each such comparison yields an angular difference which would be used to increment the corresponding bin in the angle histogram. At the end of this process, the bin with the most elements yields an estimate of the orientation difference between current and snapshot images. This technique might be particularly useful in our laboratory as the ceiling has distinct edges whose orientation in our panoramic images is highly dependent on the orientation of the robot. These two alternatives to compassing – ZPR and angle histograms – would of course have to be tested in the context of difference surface-based visual homing before being used.

We could also use the visual compass described in Chapter 2 to rotationally align current and snapshot images. We in fact did some preliminary testing to determine whether this was a viable solution for visual homing by difference surface optimisation. The visual compass seemed to sometimes introduce local optima at locations relatively far from the snapshot location. These local optima would of course attract a nearby homing robot, steering it away from the true snapshot location. It may benefit us to take another look at the visual compass, though.

If all of the above methods fail to provide reliable information to align the orientations of snapshot and current images, we could try equipping the Koala with a gyroscope. The gyroscope would augment our dead reckoning system to track the robot's orientation over time. Due to systematic and non-systematic odometric errors (described in Chapter 2), the robot's dead reckoning system is insufficient to perform this tracking. A gyroscope measures the angular velocity of an accelerating system. The gyroscope's output can be integrated over time to estimate the robot's orientation with respect to its initial orientation. As in Roumeliotis and Bekey [1997], an extended Kalman filter can be used to fuse wheel encoder information with measurements from the robot's gyroscope to boost the accuracy of the robot's orientation estimate. See Chapter 2 for more information on the extended Kalman filter.

Chapter 7

Conclusions and Key Future Work

In this dissertation we have investigated a computationally efficient and robust algorithm for visual homing in dynamic indoor environments. As we discussed in Chapter 2 visual homing in general is a useful visual servoing technique. Visual homing is frequently employed to guide a robot between adjacent nodes in a vision-based topological navigation system (see e.g. Argyros et al. [2005]). Visual homing is an appropriate approach for these systems because purely topological vision-based representations of an environment do not contain explicit metric information about that environment (i.e. landmark locations). As we first set out in Chapter 1 visual homing allows for navigation without explicit knowledge of the location of landmarks in a global coordinate system. We saw in Chapter 2 that visual homing is also used to solving the docking problem in which a robot must be guided to a precise pose with respect to an object in the environment for the purpose of, say, recharging or grasping.

We argued in Chapter 2 that image-based (as opposed to feature-based) visual homing is a worthwhile approach to visual navigation. Image-based approaches eschew feature selection, extraction and correspondence. Many navigation algorithms require consistently successful solutions to these difficult problems in order to operate. Though SIFT features (see Chapter 2 for details) offer a powerful tool for feature extraction and correspondence, we in this dissertation opt to investigate a more parsimonious approach to visual homing.

When we began work on this dissertation, there were two image-based visual homing algorithms to be found in the literature: image warping (Franz et al. [1998b]) and homing by difference surface optimisation (Zeil et al. [2003]). Image warping uses a computationally intensive brute-force search to infer home vectors. Difference surface homing, on the other hand, is quite computationally efficient and algorithmically simple. Unfortunately, Zeil et al. [2003] found that difference surface homing using a root-mean-square (*RMS*) formula to measure image similarity is not robust in visually dynamic environments. As we shall argue here, our contribution to the field of visual homing has been to render difference surface homing more robust to visual dynamism; to push the algorithm to limits not considered by Zeil; and to pioneer methods to speed up the computation required by difference surface homing without degradation of homing success. Along the way we have discovered novel and fruitful links between visual homing and other bodies of literature.

Zeil et al. [2003] identified limitations to *RMS* as an image similarity measure used for difference surface homing. In Chapter 3, we provided novel empirical and analytical proof of these limitations. We demonstrated for the first time that difference surface homing with *RMS* also works well in a static indoor laboratory environment. In agreement with Zeil et al. [2003], we showed empirically that difference surface homing with *RMS* is not robust when the snapshot image I_S and current image I_C are captured in different lighting conditions. In a novel mathematical analysis of the root-mean-square measure, we demonstrated that moving the homing agent so as to minimise *RMS* between current and snapshot images is equivalent to simultaneously

- seeking high covariance between I_S and I_C (i.e. minimising $-2Cov(I_S, I_C)$);
- seeking low variance current images (i.e. minimising $Var(I_C)$); and
- seeking equality of the mean intensities of I_S and I_C (i.e. minimising $(\bar{I_S} - \bar{I_C})^2$).

We argued that the second two items above often lead the homing agent to difference surface minima which do not coincide with the snapshot location (i.e. false positives).

We demonstrated in a principled way in Chapter 3 that there are better alternatives to *RMS* for measuring image similarity for the purpose of difference surface homing. Our novel analysis of the root-mean-square measure predicted that the covariance term of the *RMS* equation would yield more robust difference surfaces in dynamic environments than the *RMS* itself. We confirmed this prediction with simulated homing runs using Vardy's image data sets. We argued that the covariance is only a trustworthy measure of the similarity between I_S given I_C when there is a linear relationship between intensities in I_S and I_C at corresponding pixel locations. We demonstrated that such a linear relationship between I_S and I_C *does* exist in static conditions but is not always present in dynamic situations; we gave an example in which illumination change between capture of I_S and I_C leads to a nonlinear relationship in pixel intensities. Mutual image information (*MI*) assumes no such linear relationship, only that pixel intensities in I_C are good *predictors* of corresponding pixels I_S. Mutual image information was indeed superior to covariance for the purposes of difference surface homing in several dynamic environments.

As we argued in Chapter 3, mutual image information as an image similarity measure may have applications outside the narrow confines of difference surface homing. *RMS* (or something very similar) is quite often used to measure the difference between images in other image-based navigation schemes (e.g. the image warping algorithm of Franz et al. [1998b] and image-based Monte Carlo localisation [Menegatti et al. [2004]]). As in our work, these algorithms compare a current image with images captured previously. Lighting and landmark locations might well have changed in the interim. We have demonstrated that mutual image information is robust to this dynamism and so could provide a useful image similarity measure in image-based robot navigation in general. Thus this dissertation may well be of interest to many workers in visual robotic navigation and machine vision, not just those focused on visual homing.

Compared to other image-based visual homing algorithms – particularly image warping (see Franz et al. [1998b] and Chapter 2) – difference surface homing is a computationally efficient homing algorithm. One of our interests in this work was to increase this efficiency without drastically diminishing the ease with which a difference surface can be optimised. We did so using techniques which no other researchers in visual homing have yet explored. In Chapter 4 we explored computation of mutual information using both serial and parallel processors. An EyeRIS parallel image processing device was available in our laboratory for our use. We created novel parallel histogramming algorithms for use on the EyeRIS and demonstrated that computation of mutual information with these algorithms requires $O(B^2)$ operations where B is the number of intensity levels in the input images.

Unfortunately, noise in the EyeRIS's image capture process renders images unusable for difference surface homing. We therefore explored methods to speed the serial computation of mutual information. Serial computation of mutual image information takes $O(B^2 + NM)$ steps where each input image has N rows and M columns. We demonstrated that a reduction (sometimes quite drastic) in spatial and/or intensity levels in our input image has little or no discernible effect on homing performance in a laboratory environment in both static and dynamic conditions. We used a number of novel criteria to assess or in some cases infer homing success; more information on these criteria can be found in Chapter 4. We found that the time required for serial computation of mutual information for low resolution images was of the same order of magnitude as the time required for parallel computation with the EyeRIS. We thus felt justified in abandoning the EyeRIS for the time being and computing mutual information serially in the remainder of our dissertation work.

We presented empirical evidence that reduction in the number of image intensity levels results in a scaling-down of MI surface values near the snapshot location and a constant shift of MI surface values relatively far from the snapshot location. The reduction of the spatial resolution of input images had a similar though not identical effect on MI surface values. We speculated that homing with reduced images is successful due to the effect that such reduction has on the mutual information signal. At the end of Chapter 4 we provided novel analytical support for the observed reduction in MI values in response to gray level reduction both at and relatively far from the snapshot location.

Unlike most researchers in robotic homing, we demonstrated that our homing algorithms operate successfully in both realistic simulation and "live" robotic trials. We introduced realistic sensor noise in our simulations, identifying noise in both the image capture and compassing systems and determining the resulting distribution of noise in the mutual information signal. We scanned the optimisation literature to identify a set of appropriate algorithms to guide the homing robot to maximise the difference surface. Of these algorithms, we found that a gradient ascent with one-sided gradient estimation (1FDSA) is the best algorithm for the task at hand as judged by our novel evaluation criteria.

For our "live" homing trials (described in Chapter 6), we constructed a mobile robot capable of visual homing and designed and built a visual tracking system to infer the robot's pose (position and orientation) during experiments. The pose information provided by the tracker was important for the interpretation of experiments; we also used it to align snapshot and current images to a single global compass direction. Without such image rotation, calculation of mutual image information would have been meaningless.

The results reported in Chapter 6 are sometimes in line with those garnered from simulation. We found – when snapshot and current images are captured in different illumination conditions – that the mutual information image similarity measure often leads to more robust difference surface homing than the root-mean-square image similarity measure does. Unlike in our simulation work, the choice between optimisation algorithms – specifically "Run-Down" and 1FDSA – is ambiguous. The 1FDSA algorithm is successful 100% of the time when homing starts 1 metre from the snapshot location but is less robust than “Run-Down” when starting from 2 meters. 1FDSA consistently takes more time to home than does “Run-Down” due to the expensive gradient estimation carried out by the former algorithm.

The 1FDSA algorithm is expensive because it requires the homing agent to move in order estimate the difference surface gradient at a particular location. We outlined a method in Chapter 5 which allows the homing agent to estimate the difference surface gradient at a location without moving from that location. Our proposed method calls for the agent to capture two additional images near the snapshot after having stored the snapshot itself, one at $-c$ centimetres from the snapshot location in a randomly chosen direction and the other at $-c$ centimetres from the snapshot location in a direction orthogonal to the randomly chosen direction. These images need be captured only once, probably as the agent leaves the snapshot location for the first time. We showed in Chapter 5 how the image similarity between each of these three snapshot images and the current image I_C could be used to estimate the difference surface gradient at the current image capture point. In a pilot study, we demonstrated the success of this method in a static laboratory environment. The gradients of a sample difference surface computed with this method are shown in Figure 5.21. More work should be done with this algorithm to demonstrate its usability in other environments.

As we relate in Chapter 6, our “live” homing trials demonstrated to us that accurate compassing indoors is difficult. The image similarity measures we experimented with in Chapter 3 require that I_S and I_C are aligned to the same external compass direction. Must I_S and I_C be aligned in this way before measuring the similarity between the two? Not necessarily. We showed in a pilot study described in Chapter 3 that the marginal intensity distributions of I_S and I_C (i.e. the normalised intensity histograms of these images) could be compared with the Kullback-Leibler divergence (Weisstein [2007b]). The intensity histogram of an image I_C is of course left unchanged when I_C is rotated. Using this image similarity measure we demonstrated reasonably successful difference surface homing in a static laboratory environment. Results in dynamic conditions were poor, though. More work must be done with the Kullback-Leibler divergence similarity measure to improve performance in dynamic situations.

We close this chapter with a discussion of the overall strategy – a novel strategy in the visual homing community – we employed to make progress in work on this dissertation. In essence we attempted to find connections between visual homing and other bodies of literature whenever possible and to bring successful, appropriate algorithms from these fields to bear on difference surface visual homing.

In the latter part of Chapter 3 we argued that visual homing is quite similar to the problem of image registration. An image registration algorithm attempts to find the function which best transforms one image of an object or scene into a second image of the same scene or object. The two images can differ due to the poses of the imager, the modalities in which the images were captured, and/or the layout of the scene among other things. The function sought is a transformation of pixel locations, often though not always affine. While image registration algorithms search for the pixel-by-pixel transformation between two images, visual homing seeks to estimate the transformation of an *imager* from S to C given images I_S and I_C. Hill et al. [2001]) demonstrate that mutual image information (MI) can be used to assess image similarity for the purposes of image registration. MI works well as an image similarity measure in registration even when the two images to be registered differ significantly in for instance lighting and modality. The kinship between image registration and visual homing – and the success of MI in the former – led us to apply MI to measure image similarity in visual homing with, as Chapter 3 demonstrates, beneficial results.

Our study in Chapter 5 of noise in the mutual information signal led us to the body of literature known as stochastic optimisation. This literature gave us several optimisation algorithms to experiment with which have proven track records in optimising noisy functions. As we mentioned above, we found that among these algorithms gradient ascent using one-sided differencing to compute the gradient was the best choice according to our novel evaluation criteria. The stochastic optimisation literature provided useful advice on choosing realistic gains for the 1FDSA algorithm.

In sum, difference-surface based visual homing is computationally efficient method which we have shown to be competitive with other recently published homing algorithms in dynamic indoor environments. Our novel mutual information image similarity measure renders difference surface-based homing relatively robust to dramatic illumination changes in the environment. We have shown that mutual information can be computed efficiently with little loss of homing in homing success. This computational efficiency means that mutual information may well find a role in place recognition problems for topological navigation in which a current image must be compared with many reference images.

References

P.R. Adby and M. A. H. Dempster. *Introduction to Optimization Methods*. Chapman and Hall, 1974.

H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localisation using omnidirectional images. In *Proc. of the 14th International Conference on Pattern Recognition*, pages 1799–1803, 1998.

J. Anderson, D. Lee, and J. Archibald. FPGA implementation of vision algorithms for small autonomous robots. volume 6006, pages 401–411, Boston, MA, USA, 2005.

T. W. Anderson and S. L. Sclove. *The Statistical Analysis of Data*. The Scientific Press, Palo Alto, CA, USA, 1986.

A. A. Argyros, K. Bekris, and S. C. Orphanoudakis. Robot homing based on corner tracking in a sequence of panoramic images. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 3–10, 2001.

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

P. Baccou and B. Jouvencel. Homing and navigation using one transponder for auv, post-processing comparisons results with long base-line navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4004–4009, Washington, DC, USA, 2002.

Joao P. Barreto, Frederick Martin, and Radu Horaud. Visual servoing/tracking using central catadioptric images. In *8th International Symposium on Experimental Robotics*, Bombay, India, 2002.

Ronen Basri, Ehud Rivlin, and Ilan Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 33(2):117–137, 1999.

Edward Batschelet. *Circular Statistics in Biology*, pages 7–10. Academic Press, 1983.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision*, 2006.

K. E. Bekris, A. A. Argyros, and L. E. Kavraki. Angle-based methods for mobile robot navigation: reaching the entire plane. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2373–2378, New Orleans, LA, USA, 2004.

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

P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

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

O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Image based navigation using a topological map. In *Proceedings of the 13th Annual Conference of the Advanced School for Computing and Imaging*, The Netherlands, 2007.

J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(5):869–880, 1996.

D. Bradley, A. Brunton, M. Fiala, and G. Roth. Image-based navigation in real environments using panoramas. In *Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, Ottawa, Canada, 2005.

T. Braunl, S. Feyrer, W. Rapf, and M. Reinhardt. *Parallel Image Processing*. Springer, 2001.

R. L. Burden and J. D. Faires. *Numerical Analysis*. PWS-Kent, 1993.

B.A. Cartwright and T.S. Collett. Landmark learning in bees. *Journal of Comparative Physiology*, 151:521–543, 1983.

S. M. Castillo. EyeRIS vision system evaluation kit: Hardware description. 2005a.

S. M. Castillo. EyeRIS vision system evaluation kit: Development platform introduction. 2005b.

S. M. Castillo. EyeRIS vision system evaluation kit: Programmer's guide. 2005c.

F. Chaumette. Image moments: A general and useful set of features for visual servoing. *IEEE Transactions on Robotics and Automation*, 20(4):713–723, 2004.

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

F. Chaumette and S. Hutchinson. Visual servo control part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, 2007.

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

A. A. Cole-Rhodes, K. L. Johnson, J. LeMoigne, and I. Zavorin. Multiresolution registration of remote sensing imagery by optimization of mutual information using a stochastic gradient. *IEEE Transactions on Image Processing*, 12(12):1495–1511, 2003.

User's Guide: Creative Video Blaster WebCam 3. Creative Labs, 2000.

A. Cretual and F. Chaumette. Visual servoing based on image motion. *The International Journal of Robotics Research*, 20(11):857–877, 2001.

J. L. Crowley and F. Pourraz. Continuity properties of the appearance manifold for mobile robot position estimation. *Image and Vision Computing*, 19:741–752, 2001.

A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the 2003 IEEE International Conference on Computer Vision*, pages 1403–1410, 2003.

K. Deguchi and T. Noguchi. Visual servoing using eigenspace method and dynamic calculation of interaction matrices. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 302–306, 1996.

F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR '99)*, Fort Collins, CO, USA, 1999a.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*, 1999b.

G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csobra. A solution to the simultaneous localisation and mapping (SLAM) problem. *IEEE Trans. Robotics and Automation*, 17(3):229–241, 2001.

B. Draper, W. Najar, W. Bohm, J. Hammes, B. Rinker, C. Ross, M. Chawathe, and J. Bins. Compiling and optimizing image processing algorithms for FPGAs. 2000.

H. Durrant-Whyte. Where am I? a tutorial on mobile vehicle localization. *Industrial Robot*, 21(2):11–17, 1994.

H. Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *IEEE Robotics and Automation Magazine*, 13(2): 99–110, 2006.

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

T. Feree and S. Lockery. Computational rules for chemotaxis in the nematode c. elegans. *Journal of Computational Neuroscience*, 6:263–277, 1999.

Andrey Filippov. An alternative to the correlated double sampling (cds) and binning in ccd readout. http://www3.elphel.com/articles/CDS.html, 2006.

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

R. Fisher, S. Perkins, A. Walker, and E. Wolfart. *Hypermedia Image Processing Reference*. John Wiley and Sons, Ltd., 1996.

Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert. Particle filters for mobile robot localization. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, New York, 2001. Springer.

M.O. Franz and H.A. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133–153, 2000.

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

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

J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omni-directional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000.

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

G. Geis, F. Gollas, and R. Tetzlaff. On the implementation of cellular wave computing methods by hardware learning. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2930–2933, 2007.

Toon Goedemé, T. Tuytelaars, L. Van Gool, D. Vanhooydonck, R. Demeester, and M. Nuttin. Is structure needed for omnidirectional visual homing? In *Proceedings of the 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Espoo, Finland, 2005a.

Toon Goedemé, T. Tuytelaars, G. Vanacker, M. Nuttin, and L. Van Gool. Omnidirectional sparse visual path following with occlusion-robust feature tracking. In *Proceedings of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras Location*, Beijing, China, 2005b.

S. Gourichon, J.A. Meyer, and P. Pirim. Using colored snapshots for short-range guidance in mobile robots. *International Journal of Robotics and Automation : Special Issue on Biologically Inspired Robotics*, 17(4):154–162, 2002.

J.E. Guivant and E.M. Nebot. *IEEE Trans. Robotics and Automation*, 17(3):242–257.

Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

D. L. G. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes. Medical image registration. *Physics in Medicine and Biology*, 46:1–45, 2001.

J. Hong, X. Tan, B. Pinnette, R. Weiss, and E.M. Riseman. Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 620–625, Sacremento, CA, USA, 1991.

J. W. Hsieh, H. Y. M. Liao, K. C. Fan, M. T. Ko, and Y. P. Hung. Image registration using a new edge-based approach. *Computer Vision and Image Understanding*, 67 (2):112–130, 1997.

S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

H. Ishiguro and S. Tsuji. Image-based memory of environment. In *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS'96)*, pages 634–639, 1996.

P. Jantapremjit and P. Wilson. Optimal control and guidance for homing and docking tasks using an autonomous underwater vehicle. In *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, pages 243–248, Harbin, China, 2007.

J.F. Jenq and S. Sahni. Histogramming on a reconfigurable mesh computer. In *Proc. 6th IEEE International Parallel Processing Symposium*, pages 425–432, 1992.

M. Jogan, A. Leonardis, H. Wildenauer, and H. Bischof. Mobile robot localization under varying illumination. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 741–744, 2002.

Matjaz Jogan and Ales Leonardis. Robust localization using panoramic view-based recognition. In *Proceedings 15th International Conference on Pattern Recognition*, pages 136–139, 2000.

Matjaz Jogan and Ales Leonardis. Panoramic eigenimages for spatial localisation. In *Computer Analysis of Images and Patterns*, pages 558–567, 1999.

S.P.D. Judd and T.S. Collett. Multiple stored views and landmark guidance in ants. *Nature*, 392:710–714, 1998.

K-Team. Koala silver user manual. 2001.

Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of Computer Vision and Pattern Recognition*, pages 506–513, 2004.

D. J. Kleitman. *Calculus with Applications*. The Massachusettes Institute of Technology, MA, USA, 2005.

J. Košecká and F. Li. Vision based topological markov localization. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 1481–1486, New Orleans, LA, USA, 2004.

J. Košecká, L. Zhou, P. Barber, and Z. Duric. Qualitative image based localization in indoor environments. In *Proceedings of the 2003 IEEE Computer Society on Computer Vision and Pattern Recognition (CVPR '03)*, pages 3–8, 2003.

D. Krantz. Non-uniform dead-reckoning position estimate updates. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 2061–2066, 1996.

B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

F. Labrosse. Visual compass. pages 85–92, University of Essex, Colchester, UK, 2004.

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

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

L. Ledwich and S. Williams. Reduced SIFT features for image retrieval and indoor localisation. In *Proc. of the Australian Conference on Robotics and Automation*, Canberra, Australia, 2004.

M. Lehrer and G. Bianco. The turn-back-and-look behaviour: bee versus robot. *Biological Cybernetics*, 83:211–229, 2000.

J. J. Leonard, P. M. Newman, R. J. Rikoski, J. Neira, and J. D. Tardos. Towards robust data association and feature modeling for concurrent mapping and localization. Lorne, Victoria, Australia, 2001.

Jianchun Li. *Design of an FPGA-Based Computing Platform for Real-Time 3D Medical Imaging*. PhD thesis, Case Western Reserve University, 2005.

D. Lizotte. Noisy global function optimization. www.cs.ualberta.ca/ dale/cmput670/NoisyFunOpt.ppt, 2005.

D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with gaussian process regression. Hyderabad, India, 2007.

H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

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

F. Lu and E. E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. pages 935–938, 1994.

F. Lu and E. E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

David Luebke, Mark Harris, Naga Govindaraju, Aaron Lefohn, Mike Houston, John Owens, Mark Segal, Matthew Papakipos, and Ian Buck. GPGPU: general-purpose computation on graphics hardware. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 208, New York, NY, USA, 2006. ACM. ISBN 0-7695-2700-0.

F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *IEEE Transactions on Medical Imaging*, 16(2):187–198, 1997.

E. Marchand and F. Chaumette. Features tracking for visual servoing purposes. In D. Kragic and H. Christensen, editors, *Proceedings of the Conference on Advances in Robot Vision*, Sendai, Japan, 2004.

D. Marshall. Region growing. $http://www.cs.cf.ac.uk/Dave/Vision\_lecture/node35.html$, 1997.

J. K. Martin and D. S. Hirschberg. Small sample statistics for classification error rates ii: Confidence intervals and significance tests. Technical Report 96-22, Irvine, CA 92697-3425, July 1996.

T. Martinez-Marin and T. Duckett. Fast reinforcement learning for vision-guided mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

C. McCarthy and N. Barnes. A robust docking strategy for a mobile robot using flow field divergence. In *Proceedings of the 2006 International Conference on Intelligent Robots and Systems*, Beijing, China, 2006.

G. T. McCaw. Resection in survey. *The Geographical Journal*, 52(2):105–123, 1918.

E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based monte-carlo localisation with omnidirectional images. *Robotics and Autonomous Systems*, 48 (1):17–30, 2004.

R. Menzel, U. Greggers, A. Smith, S. Berger, R. Brandt, S. Brunke, G. Bundrock, S. Hülse, T. Plümpe, F. Schaupp, E. Schüttler, S. Stach, J. Stindt, N. Stollhoff, and S. Watzl. Honey bees navigate according to a map-like spatial memory. *Proceedings of the National Academy of Sciences of the United States of America*, 102(8):3040–3045, 2005.

T. Mitchell and Frederic Labrosse. Visual homing: a purely appearance-based approach. In *Proceedings of TAROS (Towards Autonomous Robotic Systems)*, The University of Essex, UK, 2004.

R. Möller. Insect visual homing strategies in a robot with analog processing. *Biological Cybernetics*, 83:231–243, 2000.

R. Möller. A biorobotics approach to the study of insect visual homing strategies, 2002.

R. Möller. Local visual homing by warping of two-dimensional images. *Robotics and Autonomous Systems*, 2008. to appear.

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

R. Möller, D. Lambrinos, T. Roggendorf, R. Pfeifer, and R. Wehner. *Insect Strategies of Visual Homing in Mobile Robots*, chapter 3, pages 37–66. The MIT Press, Cambridge, Massachusetts, 2001.

R. Möller, A. Vardy, S. Kreft, and S. Ruswisch. Visual homing in environments with anisotropic landmark distribution. *Autonomous Robots*, 23(3):231–245, 2007.

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

S. K. Nayar. Omnidirectional video camera. In *Proceedings of DARPA Image Understanding Workshop*, New Orleans, USA, 1997.

T. Pajdla and V. Hlaváč. Zero phase representation of panoramic images for image based localization. In *Proceedings of the 8th Annual CAIP*, pages 550–557, 1999.

N. Papanikolopoulos and C. Smith. Computer vision issues during eye-in-hand robotic tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2989–2994, 1995.

J. Saez Pons, W. Hübner, H. Dahmen, and H. A. Mallot. Vision-based robot homing in dynamic environments. In *Proceedings of the 13th Annual International Conference on Robotics and Applications*, Würzburg, Germany, 2007.

A. Rizzi, D. Duina, and R. Cassinis. A novel visual landmark matching for a biologically inspired homing. *Pattern Recognition Letters*, 22:1371–1378, 2001.

S. I. Roumeliotis and G. A. Bekey. An extended kalman filter for frequent local and infrequent global sensor data fusion. In *Proceedings of the International Symposium on Intelligent Systems and Advanced Manufacturing*, 1997.

José Santos-Victor and Giulio Sandini. Visual behaviors for docking. *Computer Vision and Image Understanding: CVIU*, 67(3):223–238, 1997.

T. Scheuermann and J. Hensley. Efficient histogram generation using scattering on GPUs. pages 33–37, Seattle, Washington, USA, 2007.

B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 1994.

S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

A. Shahbahrami, J. Hur, B. Juurlink, and S. Wong. FPGA implementation of parallel histogram computation. Goteborg, Sweden, 2008.

R. Shams and N. Barnes. Speeding up mutual information computation using nvidia cuda hardware. pages 555–560, Glenelg, Australia, 2007.

L. Smith, A. Philippides, and P. Husbands. Navigation in large-scale environments using an augmented model of visual homing. pages 251–262, 2006.

J. C. Spall. *Handbook of Computational Statistics*, chapter Stochastic Optimization, pages 169–199. Springer, Heidelberg, 2004.

J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, Hoboken, NJ, USA, 2003.

P. Sprent and N. Smeeton. *Applied Nonparametric Statistical Methods*, pages 133–135. Chapman and Hall, 2007.

W. Sturzl and H. A. Mallot. Vision-based homing with a panoramic stereo sensor. In *Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 620–628, 2002.

W. Sturzl and H. A. Mallot. Efficient visual homing based on fourier transformed panoramic images. *Robotics and Autonomous Systems*, 54(4):300–313, 2006.

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

A.A. Svenshnikov. *Problems in Probability Theory, Mathematical Statistics and Theory of Random Functions*, page 85. W.B. Saunders Company, Philadelphia, USA, 1968.

T. Svoboda and T. Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision*, 49(1):23–37, 2002.

T. Svoboda, T. Pajdla, and V. Hlavac. Motion estimation using central panoramic cameras. 1998. IEEE International Conference on Intelligent Vehicles.

M. Szenher. Visual homing with learned goal distance information. In K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte, editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment*, pages 223–229, 2005a.

M. Szenher. Visual homing in natural environments. In U. Nehmzow, C. Melhuish, and M. Witkowski, editors, *Proceedings of Towards Autonomous Robotic Systems*, pages 221–226, 2005b.

S. Thrun. Robotic mapping: a survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, Pittsburgh, PA 15213, February 2002.

S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141, 2001.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, USA, 2005.

I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *IEEE International Conference on Robotics and Automation*, pages 1023–1029, San Francisco, CA, USA, 2000.

K. Usher, P. Corke, and P. Ridley. Sensing for visual homing. In *Australasian Conference on Robotics and Automation*, pages 37–42, 2002.

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

A. Vardy and F. Oppacher. A scale invariant neural feature detector for visual homing. In *Proceedings of the Workshop on Neurobotics, German Conference on Artificial Intelligence*, 2004.

P. Viola and W. M. Wells. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1995.

C. Weber, S. Wermter, and A. Zochios. Robot docking with neural vision and reinforcement. In *Proceedings of the IROS-2003 Workshop on Robot Programming by Demonstration*, Las Vegas, NV, USA, 2003.

K. Weber, S. Venkatesh, and M. Srinivasan. Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97, 1998.

R. Wehner. Large-scale navigation: the insect case. *Lecture Notes in Computer Science*, 1661:1–20, 1999.

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

R. Wei, D. Austin, and R. Mahony. Biomimetic application of desert ant visual navigation for mobile robot docking with weighted landmarks. *International Journal of Intelligent Systems Technologies and Applications*, 1(1–2):174–190, 2005.

E. W. Weisstein. Sample variance distribution. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/SampleVarianceDistribution.html, 2003.

E. W. Weisstein. Inverse tangent. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/InverseTangent.html, 2007a.

E. W. Weisstein. Relative entropy. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/RelativeEntropy.html, 2007b.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 2006.

H. Yang, M. Pollefeys, G. Welch, J. Frahm, and A. Ilie. Differential camera tracking through linearizing the local appearance manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

M. Zampoglou, M. Szenher, and B. Webb. Adaptation of controllers for image-based homing. *Adaptive Behavior*, 14(4):381–399, 2006.

J. Zeil. Personal communication. 2007.

J. Zeil, A. Kelber, and R. Voss. Structure and function of learning flights in bees and wasps. *Journal of Experimental Biology*, 199:245–252, 1996.

J. Zeil, M. Hofmann, and J. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.

Appendix A

RMS Distribution Due to Webcam Noise

Here, we derive the distribution of the *RMS* signal when the current image I_C is corrupted by zero-mean Gaussian white noise. This is the Webcam capture noise described in Section 5.4.1.

RMS is closely related to the mean squared differences (*MSD*) measure, which we use here for the sake of mathematical simplicity[1]:

$$MSD(I_S, I_C) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (I_S(i,j) - I_C(i,j))^2 \tag{A.1}$$

Here, as usual, I_S is our snapshot image and I_C is our current image. Each is an intensity image with N rows and M columns.

As stated above, each pixel (i,j) in I_C is corrupted by a noise $\varepsilon_{i,j}$ drawn from a normal mean Gaussian distribution with standard deviation σ: $\varepsilon_{i,j} \sim N(0, \sigma^2)$. This noise is spatially and temporally uncorrelated. We shall call the noisy current image I_C^{Noisy}.

The following relates $MSD(I_S, I_C)$ to $MSD(I_S, I_C^{Noisy})$:

[1]Other authors have replaced *RMS* with *MSD*; see for example Möller and Vardy [2006].

$$\begin{aligned}
MSD(I_S, I_C^{Noisy}) &= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[I_S(i,j) - (I_C(i,j) + \varepsilon_{i,j})]^2 \\
&= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[(I_S(i,j) - I_C(i,j))^2 + 2\varepsilon_{i,j}(I_S(i,j) - I_C(i,j)) + \varepsilon_{i,j}^2] \\
&= \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[I_S(i,j) - I_C(i,j)]^2 + \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}[\varepsilon_{i,j}(I_S(i,j) - I_C(i,j))] + \\
&\quad \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\varepsilon_{i,j}^2 \qquad \text{(A.2)}
\end{aligned}$$

The first term in Equation A.2 above is simply Equation A.1. The third term is the sample variance of the noise distribution. This term has an expected value of σ^2; it's distribution is, according to Weisstein [2003], a Pearson type III distribution. As the noise has a mean of zero, the second term is an estimate of (twice the) covariance between the signal and the difference image $I_S - I_C$. The noise is independent of the difference image. Thus, this second term will be very close to zero. We shall assume it is zero.

From this discussion we conclude that corrupting I_C with zero-mean Gaussian white noise leads to a positive shift in *MSD* value with expected value σ^2 with a Pearson type III distribution. To test this, we corrupted an image I_C with Gaussian white noise with variance 25 1000 times. The mean difference between the *MSD* with noisy images and the "true" *MSD* was 24.1. When we reduced the intensity noise variance to 16, the mean difference became 15.4. When we reduced the noise variance to 9, the mean difference became 8.7. In all cases the individual differences were positive as predicted. It seems as though the mean difference is consistently slightly less than the intensity noise variance (our mathematical analysis predicted equality). This discrepancy might be due to the fact that we limit noisy intensity values to be in the range $[0, 255]$ when injected noise brings an intensity value outside this range. This thresholding in effect reduces the variance of the noise by a small amount.

We next wanted to determine whether the distribution of differences was indeed a Pearson type III distribution as predicted. We first looked at one of our empirically generated probability distributions (for intensity noise with variance 9) and saw a resemblance to the gamma distribution (which is a Pearson type III distribution). We then used Matlab's **gamfit** function to calculate maximum likelihood estimates of the shape and scale parameters for our empirical distribution (assuming that it is a gamma distribution). Finally, we used Matlab's **chi2gof** function to determine whether the estimated shape and scale parameters yield a gamma distribution which is a statistically "good" fit for our empirical distribution. Unfortunately, the **chi2gof** function indicated that the null hypothesis – that the empirical distribution is a gamma distribution – should be rejected. A five percent significance level was used. We repeated this procedure with other members of the family of Pearson type III distributions but failed to find an appropriate distribution.

Appendix B

Optimisation Stopping Criteria

As we discussed in Section 5.4.4, it is difficult to know when our homing algorithm has reached its goal. We propose the following stopping criteria to aid in making this decision:

1. Stop when the current MI difference surface sample exceeds a predetermined threshold. Given the discussion in Section 5.4.4, this criteria is unlikely to work in all situations. It's advantage is that every stochastic optimisation algorithm described in Section 5.3 makes use of difference surface values so this criteria is applicable regardless of the method of optimisation.

2. Stop when the magnitude of the current estimate of the gradient of the difference surface exceeds a predetermined value. Spall [2003] suggests that gradient estimates are less susceptible to noise than are value estimates so this criteria may prove better than (1). Of course, we can only apply this criteria when using an optimisation algorithm which estimates the local gradient.

3. Section 5.4.2 suggests that a small compass error near the snapshot location will produce a relatively noisy estimate of the local MI value. To mitigate this problem, we could store many versions of the reference image, rotated by different amounts from the true reference image. We could then compute the MI value of the current image (counterrotated using the current compass estimate) and each of these reference images. We could use the maximum MI value as the best estimate of the current MI value. Unfortunately, this algorithm requires several MI computations (one for each reference image) for each iteration of the optimisation routine. For fast homing, this is unacceptable.

Instead of the above procedure, we shall compute the pixel-wise **mean** of the rotated reference images above. This image average need only be computed once, at the start of the agent's homing run. We shall then compute the mutual information between the current and mean reference image. Optimisation stops when this MI estimate exceeds a predetermined value.

4. Preliminary tests with Vardy's images indicate that the mutual information between binary, edge-filtered reference and current images is close to zero away from the snapshot location and significantly greater than zero at (and most likely) near the snapshot location. This may only occur in indoor environments where there is strong edge information. Thus we can stop homing when the MI value between edge-filtered snapshot and current images exceeds a predetermined threshold.

   Computing an edge-filtered current image at each homing step adds overhead. This overhead may outweigh the benefit of using this stopping criteria. We shall attempt to determine this in tests below.

5. Compass noise affects the MI signal of edge-filtered images as well. The averaging technique used in criterion (3) seems not to work for edge-filtered images for reasons that we do not fully understand. Instead, we precompute seven edge-filtered snapshot images rotated by -3, -2, -1, 0, 1, 2, and 3 degrees from the snapshot orientation. We assume that these edge-filtered images have few edge pixels compared to the total number of pixels. A sparse representation, then, is justified in which we store only the locations of edges in each image. Instead of computing the mutual information between these edge-filtered snapshots and the edge-filtered current image, we simply count the number of edge pixels in the current image which coincide with the edge pixels in each snapshot image. We then normalise this count by the total number of edge pixels E in a snapshot image (all snapshots will have the same number of edge pixels as they differ only in rotation). The largest normalised count is used in this stopping criterion. We halt optimisation when the maximum normalised count exceeds a precomputed threshold.

6. The relatively large standard deviation of MI value near the snapshot location due to relatively small compass errors could be used to our advantage. Specifically, we could precompute three snapshot images rotated by -1, 0, and 1 degrees

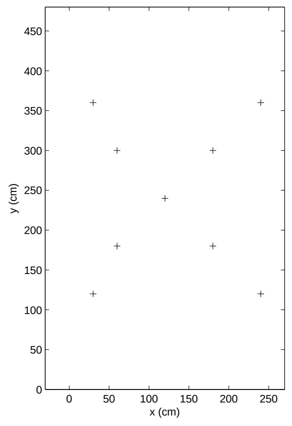


Figure B.1: Location of snapshots used in experiment to determine good stopping criteria for homing.

from the snapshot orientation (I_S^{-1}, I_S^0 and I_S^{+1}) and compute the mutual information between the current image and each of these three snapshot images. We then let this criteria equal:

$$max(\frac{|MI(I_S^{-1},I_C) - MI(I_S^0,I_C)|}{MI(I_S^0,I_C)}, \frac{|MI(I_S^{+1},I_C) - MI(I_S^0,I_C)|}{MI(I_S^0,I_C)}) \qquad \text{(B.1)}$$

Optimisation halts when this criterion exceeds a precomputed threshold.

In order to evaluate the above stopping criteria in both static and dynamic situations, we drew images from three of Vardy's data sets: "Original", "Winlit" and "Chairs." Following from the results given in Chapter 4, all images were reduced to one-quarter their original size and the original 256 intensity levels were linearly mapped to 64.

We first drew both snapshot and current images from the "Original" data set. Snapshot locations are as shown in Figure B.1. One-hundred trials were conducted for each snapshot location. In twenty-five of these trials, the current location coincided with the snapshot location. In the other 75% of the trials, the current location was selected at random from the 169 non-snapshot grid locations. The current image was corrupted with the sensor noise described in Sections 5.4.1 and 5.4.2. We computed the value of each of the six criteria above for the snapshot and current images chosen for the current trial.

We repeated the process described in the previous paragraph for the eight other possible data set pairings. This yielded a total of 8100 data points to examine. Twenty-five percent of these data points (i.e. 2025) were at snapshot locations and the rest (i.e.

6075) were at non-snapshot locations. Each data point was labelled as a snapshot or non-snapshot location.

We used this labelled data to determine which of the six stopping criterion or set of criteria is best. Early experiments indicated that criteria (6) was particularly ineffective, for reasons we do not fully understand. We decided to omit this criterion from further testing. The remaining five criteria can be formed into 31 unique combinations consisting of between 1 and 5 individual criteria. Each criteria relies on a predetermined threshold. For each of the 31 criteria sets, 3-fold cross validation was performed on a randomly shuffled version of the data set. Using the MATLABAresenal machine learning package (see http://finalfantasyxi.inf.cs.cmu.edu/MATLABArsenal/MATLABArsenal.htm), we trained a linear perceptron classifier to identify the data in the training set as either a snapshot location or non-snapshot location using the current set of criteria.

To evaluate the criteria, we first looked at classification error rates; that is, the ratio false positives and negatives to the number of test points. We obviously seek criteria with low classification error rates. We also observe that false positives – that is, when the agent mistakenly believes itself to be home – are much more dangerous than false negatives. A false positive will cause the agent halt prematurely while a false negative will result in the agent passing through the snapshot location without stopping. In the later case, the optimisation algorithm is likely to bring the agent quickly back to the snapshot location. The agent will likely meander around the snapshot location until sensor noise allows for a correct classification. With this in mind, we list both classification errors and the rates of false positives in Table B.1.

In order to determine whether a particular set of criteria in Table B.1 is better than another based on error rates, we need to know whether the given error rates are significantly different from one another. Martin and Hirschberg [1996] suggests that the standard test for significant difference between classification error rates is given in Anderson and Sclove [1986]. The test reported in Anderson and Sclove [1986] indicates whether proportions drawn from two independent samples of a data set are statistically similar. Our samples are indeed independent, as the test data was randomly shuffled. Results of this test of significance are given for all pairs of sets of stopping criteria in Table B.2. A 95% confidence level was used. Note this this data is split columnwise into multiple successive tables so that no table is wider than a single page.

When using direct search algorithms (in which no gradient estimate is made), Table B.1 suggests that using criteria 1, 4 and 5 in concert is best. These yield the lowest classification error when gradients are unavailable. In fact, the classification error re-

Stopping Criteria	**Classification Error**	**False Positive Rate**
1	12.2%	4.0%
2	10.3%	2.6%
3	12.0%	0.9%
4	5.9%	3.1%
5	5.1%	3.5%
4+5	3.9%	2.1%
3+5	2.7%	0.6%
3+4	6.0%	0.5%
2+5	3.7%	1.4%
2+4	5.0%	3.8%
2+3	12.4%	6.5%
1+5	4.3%	3.3%
1+4	5.2%	2.6%
1+3	6.5%	2.5%
1+2	13.1%	6.0%
3+4+5	2.1%	1.1%
2+4+5	2.9%	0.8%
2+3+5	2.5%	1.0%
2+3+4	4.6%	2.1%
1+4+5	2.0%	1.1%
1+3+5	2.5%	1.7%
1+3+4	7.2%	0.1%
1+2+5	3.7%	1.0%
1+2+4	4.3%	1.7%
1+2+3	9.7%	6.8%
1+2+3+4	4.7%	2.8%
1+2+3+5	2.4%	1.3%
1+2+4+5	2.3%	0.8%
1+3+4+5	3.1%	1.4%
2+3+4+5	2.2%	1.7%
1+2+3+4+5	1.9%	0.4%

Table B.1: Classification error rates for given stopping criteria using a linear perceptron classifier.

	1	2	3	4	5	4+5	3+5	3+4	2+5	2+4	2+3	1+5	1+4	1+3	1+2	3+4+5
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-
5	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
4 + 5	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
3 + 5	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-
3 + 4	Y	Y	Y	N	Y	Y	Y	-	-	-	-	-	-	-	-	-
2 + 5	Y	Y	Y	Y	Y	N	Y	Y	-	-	-	-	-	-	-	-
2 + 4	Y	Y	Y	Y	N	Y	Y	Y	Y	-	-	-	-	-	-	-
2 + 3	N	Y	N	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-
1 + 5	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y	-	-	-	-	-
1 + 4	Y	Y	Y	N	N	Y	Y	Y	Y	N	Y	Y	-	-	-	-
1 + 3	Y	Y	Y	N	Y	Y	Y	N	Y	Y	Y	Y	Y	-	-	-
1 + 2	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	-	-
3 + 4 + 5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
2 + 4 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
2 + 3 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
2 + 3 + 4	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	N	N	Y	Y	Y
1 + 4 + 5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
1 + 3 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
1 + 3 + 4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
1 + 2 + 5	Y	Y	Y	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
1 + 2 + 4	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
1 + 2 + 3	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
1 + 2 + 3 + 4	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	N	N	Y	Y	Y
1 + 2 + 3 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
1 + 2 + 4 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
1 + 3 + 4 + 5	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
2 + 3 + 4 + 5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
1 + 2 + 3 + 4 + 5	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N

Table B.2: Cell (i, j) in this table indicates whether the error rates for stopping criteria given in rows i and j in Table B.1 are significantly different with 95% confidence. A ”Y” indicates signficant difference and an ”N” signals no significant difference. This table is symmetric about the diagonal so only data below the diagonal are given. This table is split columnwise into several tables (see below) in order to fit on a single page.

	2+4+5	2+3+5	2+3+4	1+4+5	1+3+5	1+3+4	1+2+5	1+2+4	1+2+3	1+2+3+4	1+2+3+5	1+2+4+5	1+3+4+5	2+3+4+5	1+2+3+4+5
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3 + 4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1 + 4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1 + 3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1 + 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3 + 4 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 4 + 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 3 + 5	N	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2 + 3 + 4	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-
1 + 4 + 5	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-
1 + 3 + 5	N	N	Y	N	-	-	-	-	-	-	-	-	-	-	-
1 + 3 + 4	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-
1 + 2 + 5	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-
1 + 2 + 4	Y	Y	N	Y	Y	Y	N	-	-	-	-	-	-	-	-
1 + 2 + 3	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-
1 + 2 + 3 + 4	Y	Y	N	Y	Y	Y	Y	N	Y	-	-	-	-	-	-
1 + 2 + 3 + 5	N	N	Y	N	N	Y	Y	Y	Y	Y	-	-	-	-	-
1 + 2 + 4 + 5	Y	N	Y	N	N	Y	Y	Y	Y	Y	N	-	-	-	-
1 + 3 + 4 + 5	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
2 + 3 + 4 + 5	Y	N	Y	N	N	Y	Y	Y	Y	Y	N	N	Y	-	-
1 + 2 + 3 + 4 + 5	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	N	Y	N	-

Table B.3: Continuation of Table B.2.

sulting from the use of this set of criteria is not significantly different than the best overall classification error (when all criteria are used together). The rate of false positives when using criteria 1, 4 and 5 is quite low too.

The data in Tables B.1 and B.2 suggests that, when difference surface gradients are estimated, criteria 2 and 5 taken together yield quite good performance. The classification errors are not quite as low as when criteria 1, 4 and 5 are applied, but criteria 2 and 5 requires less computation.

In the final incarnation of our difference surface homing algorithms, we chose not to use the stopping criteria described in this appendix. This is because each criterion requires a threshold to be set. The homing robot would have to sample several images at different locations in its environment to determine an appropriate value for each threshold. We felt that this sampling led us away from the problem we wished to focus on, namely difference surface homing. The work described here may well be of benefit to future visual homing applications.

Appendix C

Published Papers

I published four papers on visual homing in the course of work on this dissertation. These papers in chronological order are:

- M. Szenher. Visual homing with learned goal distance information. In K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte, editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment*, pages 223-229, 2005.
- M. Szenher. Visual homing in natural environments. In U. Nehmzow, C. Melhuish, and M. Witkowski, editors, *Proceedings of Towards Autonomous Robotic Systems*, pages 221-226, 2005.
- M. Zampoglou, M. Szenher, and B. Webb. Adaptation of controllers for image-based homing. *Adaptive Behavior*, 14(4):381-399, 2006.
- Szenher, M. (to appear). Navigation by image-based visual homing. In J. R. Rabunal, J. Dorado and A. Pazos (Eds.), *Encyclopedia of Artificial Intelligence*. Hershey, PA, USA: Information Science Reference.

The text of these papers is included in this appendix.

Three of the four papers describe work which was wholly my own and were written entirely by me (with welcome comments from my advisor, peers and reviewers). The paper titled “Adaptation of controllers for image-based homing” describes work primarily done by Marcus Zampoglou, an MSc student at the University of Edinburgh, under the supervision of myself and Dr. Barbara Webb. Zampoglou wrote most of the paper and performed the experiments described therein. The original idea (biologically-based optimisation of a difference surface) put forth in the paper was mine. I constructed most of the hardware and software used in Zampoglou used in his experiments.

Visual Homing with Learned Goal Distance Information

Matthew Szenher

Institute of Perception Action and Behaviour, University of Edinburgh, Room 1106, James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, UK
M.Szenher@sms.ed.ac.uk

Summary. Visual homing is a navigational technique allowing an agent to return to a location which it has visited previously, using visual information alone. Traditional visual homing techniques eschew goal distance information. We show that, using our novel algorithm, such information is relatively easy to acquire. We then show that the use of such goal distance information reduces the computational cost of homing.

1 Introduction

An agent employing visual homing begins at some target location at which it stores some visual information (frequently referred to as a snapshot image). The agent then meanders, perhaps completing certain tasks, until it is time to return to the target from its current location. The agent captures some visual information at the current location, compares this information to that stored at the target and uses the discrepancy to generate a homing vector. The homing vector generally leads the agent closer to the target, but not directly to it. Visual homing is an iterative process.

Most homing schemes estimate the bearing of, and not the distance from, the target position with respect to the agent's current location. The agent merely takes a unit step in the estimated direction of the target location before computing another such estimate (see e.g. [2]). Other homing schemes, such as the average landmark vector model [5] use the difference between information garnered from current and snapshot images to infer "some information" about the distance to the goal. This information correlates with but is not normally equal to the current distance to the target position. The information is used to slow the agent near the goal to avoid overshoot and serves as a stopping threshold.

In this paper, we employ a simple method to learn a novel map between the image difference of current and snapshot views and the spatial distance of the current and target positions. We use this estimated distance to augment some existing homing algorithms. We show that this additional information reduces the number of computational steps required to home without loss of homing reliability.

2 Related Research

Visual homing algorithms fall into two categories: feature-based or image-based. Feature-based homing schemes attempt to segment images into foreground (features or landmarks) and background. A homing vector is inferred from the difference in bearing and (sometimes) apparent size between paired features in current and target images. Image-based homing algorithms use the whole-image difference between snapshot and current views to generate a homing vector.

Examples of feature-based visual homing algorithms include the snapshot model [2], the average landmark vector model [6], Hong's homing scheme [4] and Basri et al.'s epipole surfing algorithm [1]. Feature-based homing schemes require that the same landmarks are detected in both snapshot and current images, which are often captured at dramatically different viewpoints. This proved a problem in our test environment, so we did not use any feature-based visual homing schemes in the experiments described below. Instead, we employed the following two image-based visual homing algorithms reported in the literature.

2.1 Image Warping

Image warping [3] is a much-cited image-based visual homing algorithm. The algorithm makes the often unrealistic assumption that all landmarks are at an equal distance from the goal position. The algorithm iterates through all possible pose changes between goal and current positions and, using the equal distance assumption, warps the current image to a candidate snapshot. The candidate snapshot most similar to the actual snapshot yields a homing vector whose length is proportional to (but, because of the equal distance assumption) not equal to the agent's actual distance to the goal location. This brute-force algorithm has been shown to work well in practice (see, e.g., [8]).

2.2 Homing by Gradient Descent

Zeil et al. in [10] demonstrated that image difference between a snapshot image and images taken nearby is highly correlated with the spatial distance between the locations at which the images were captured. They defined image difference as the root-mean-square difference over all pixels (RMS). When RMS values are calculated at all points on the plane with respect to a particular reference image, the result is an RMS difference surface (see Figure 1).

In novel work, we confirmed that the results of Zeil et al. hold for a laboratory environment (see Figure 1). Our data (provided by Andrew Vardy and used in [8]) was collected in a laboratory whose periphery was lined with computers, posters, curtains, plants, etc. There were no obstructions in the experimental area itself (the floor). Using a small Webcam imaging a hyperbolic mirror, 170 panoramic colour images were captured at 30 cm intervals in a 3m x 5.1m area on the flat laboratory floor. The area was illuminated by overhead, fluorescent lights. All images are aligned with an external compass reference.

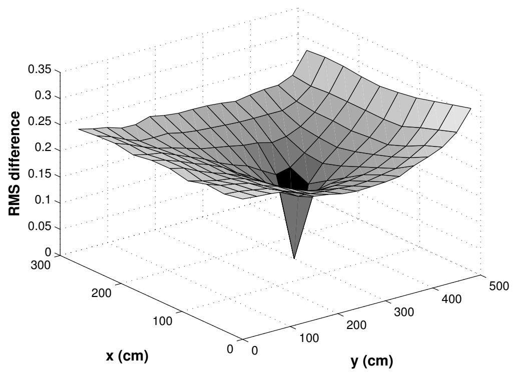


Fig. 1. RMS difference surface with respect to reference point x = 150cm, y = 300cm in our laboratory environment.

To home, the agent can calculate RMS values at three proximate, noncolinear points in the environment. The agent can then, using these three values, make a planar estimate of the local RMS difference surface and calculate the gradient of this plane. The gradient serves as the local homing vector. The length of the estimated gradient is rarely equal to the agent's distance from the goal. The agent can continue this process until the RMS falls below a certain threshold, near zero. Zeil et al. call this method "Triangular."

3 Learning the Local Distance Map

Unlike previous approaches to visual homing, we will use the estimated spatial distance of the agent from the goal position while performing a homing run. In this section, we will give a novel algorithm which a homing agent can use to learn such a distance estimate.

After viewing a number of RMS surfaces similar to that shown in Figure 1, we realised that most take a characteristic shape: the RMS value increases monotonically from the target position; the magnitude of the gradient of the RMS surface near the reference position is relatively large and decreases as the spatial distance from the reference location increases. Zeil et al. reported that this monotonic property holds for up to three meters outdoors. In our laboratory environment, the RMS surface increased monotonically for 4.5 meters from the target location in some cases.

In previous work ([7]), we found that the power law as a function of distance to the goal provides a good fit to the characteristic RMS surface. This power function is

$$RMS = Ad^B \tag{1}$$

where A and B are constants and d is the Euclidean distance from the current location to the reference location. Given A, B and an RMS value at a particular location, we can use the inverse of Equation 1 to estimate goal distance. We show in Section 4 that this simple metric information can render visual homing dramatically more efficient.

We chose to employ the least squares fitting approach described in [9] to determine A and B. A set of n training values RMS_i are collected on the agent's first outward journey from the goal position. The agent's path integration system (we assume our agent possesses such a sensor) provides the associated values d_i. On first leaving the goal, path integration is fairly accurate.

4 Experiments and Results

All experiments described below used the laboratory data set described in Section 2 and were carried out offline.

4.1 Distance Estimation

The purpose of this experiment was to measure the reliability of the inverse of Equation 1 in the estimation of distance from the goal. We iterated through all 170 images, choosing each to be the reference image in turn. For each reference image, we calculated parameters A and B as described in [9]. Twenty locations in a Gaussian distribution (with standard deviation 60cm) around the reference were selected at random as training points; in unpublished work, we found that this sampling regime worked well. Having calculated A and B, we used the inverse of Equation 1 to estimate the distance from every other (non-training) point in the data set to the current reference point. This yielded 150 test points of varying distances from the goal location. We repeated this process using every other point in the data set as the reference point in turn, resulting in a set of 25500 estimated distances. Results are reported in Figure 2; they indicate that in our laboratory environment the inverse of Equation 1 is a fairly accurate predictor of spatial distance from the goal.

4.2 Using Distance Estimation in Visual Homing

Our aim in this project was to show that existing visual homing algorithms could benefit from learned goal distance information without loss of reliability. The most obvious benefit would be in decreased homing path lengths but we found no significant change in path lengths with the addition of goal distance information. We instead looked at reduction in pixel operations during homing. All homing algorithms are iterative processes, in which new homing vectors are calculated at each iteration. Thus, the number of iterations required for a homing run multiplied by the number of pixel operations required in each iteration dictates the total number of pixel operations carried

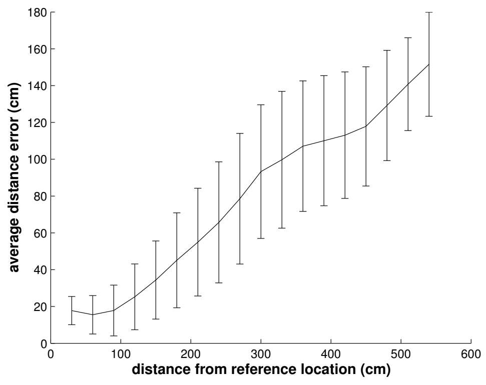
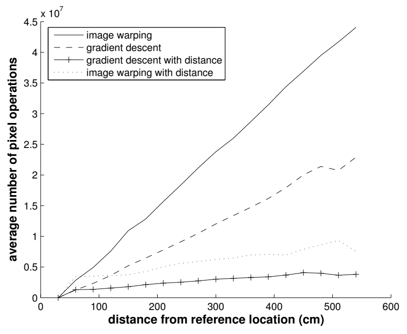


Fig. 2. Average error in distance estimate as a function of distance from the goal position. Error bars indicate standard deviation of distance estimation.

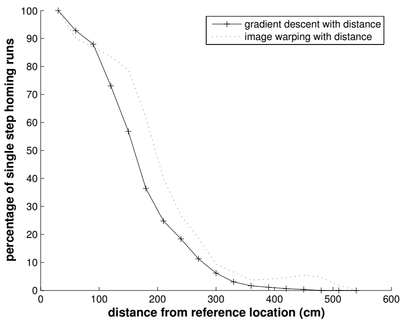
out in a homing run. Gradient descent homing requires only $O(nm)$ pixel operations in each iteration where n and m are the dimensions of the snapshot and current images. Our images were 752×564 pixels. Image warping, a brute force algorithm, requires $O(nmp)$ pixel operations in each iteration, where n and m are as before and p is the number of candidate agent poses considered (3600 in our case). Figure 3(a) shows a graph of the average total number of pixel operations required by each homing algorithm as a function of goal distance. Clearly, the addition of goal distance information drastically reduced computation time.

Did this savings in computation time affect homing reliability? To find out, we computed the mean return ratio for each algorithm in our laboratory environment. Return ratio is a measure of homing performance first introduced in [8]. Each position in the environment is taken in turn as the goal position. The return ratio for that goal position is the percentage of other locations from which homing is successful. Return ratio is akin to what other visual homing researchers call catchment area size (e.g. [3]). We found that the mean return ratio for image warping was 0.88 (s.d. 0.26) while the mean return ratio for image warping augmented with distance was 0.91 (s.d. 0.20). For gradient descent homing, the mean return ratio was 0.85 (s.d. 0.22); when augmented with goal distance information, the mean return ratio for the algorithm was 0.84 (s.d. 0.22). Application of the one-tailed Wilcoxon Signed-Rank Matched-Pair Test tells us that, with 95% confidence, the differences in return ratio means between unaugmented and augmented homing algorithms is significant.

As a final indication of the benefit of using goal distance information in homing, we show in Figure 3(b) that image warping or gradient descent augmented with goal distance information allow the agent to home in just one iteration beginning from relatively long distances from the goal. In contrast,



(a)



(b)

Fig. 3. **a**: Average number of pixel operations as a function of distance. **b**: Ratio of single step homing runs to all successful homing runs as a function of distance from the goal location for homing algorithms augmented with distance information.

unaugmented image warping and gradient descent only take the agent home in one step when the agent begins homing one unit (30 cm in our setup) from the goal location.

We repeated the above experiments with data sets captured in the same laboratory environment under a number of different lighting conditions. We arrived at largely the same results, so long as the snapshot and current images compared during the homing process were captured under the same illumination conditions (e.g. constant illuminant spectrum, intensity and location). Visual homing is not robust when illumination conditions change in the period between snapshot capture and current image capture.

5 Conclusions and Future Work

Our aim in this project was to show that visual homing augmented with goal distance information is more computationally efficient than unaugmented visual homing, with no loss of reliability. We demonstrated that indeed two image-based visual homing algorithms – image warping and homing by gradient descent – which employed goal distance information required dramatically less computation time than their unaugmented counterparts. The inclusion of distance information improves homing reliability in the case of image warping and only slightly reduces it in the case of homing by gradient descent.

Many extensions to this project are possible. We will certainly test our augmented homing schemes in environments different than the laboratory described in this paper. For example, [3] reports that the performance of image warping is degraded when there are obstacles between the goal location and the current location. We will also try to tackle visual homing in environments with non-static lighting and/or moving landmarks. We also wish to determine whether our results translate to online robot homing.

References

1. Ronen Basri, Ehud Rivlin, and Ilan Shimshoni. Visual homing: Surfing on the epipoles. In *ICCV*, pages 863–869, 1998.
2. B.A. Cartwright and T.S. Collett. Landmark learning in bees. *Journal of Comparative Physiology*, 151:521–543, 1983.
3. M.O. Franz, B. Schőlkopf, H.A. Mallot, and H.H. Bűlthoff. Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, 79:191–202, 1998.
4. J. Hong, X. Tan, B. Pinnette, R. Weiss, and E.M. Riseman. Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 620–625, Sacremento, CA, USA, 1991.
5. D. Lambrinos, R. Mőller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30:39–64, 2000.
6. R. Mőller, D. Lambrinos, T. Roggendorf, R. Pfeifer, and R. Wehner. *Insect Strategies of Visual Homing in Mobile Robots*, chapter 3. The MIT Press, Cambridge, Massachusetts, 2001.
7. M. Szenher. Learning local goal distance information. In *Proceedings of the Ninth International Conference on Cognitive and Neural Systems*, 2005.
8. A. Vardy and F. Oppacher. A scale invariant neural feature detector for visual homing. In *Proceedings of the Workshop on Neurobotics, German Conference on Artificial Intelligence*, 2004.
9. E. W. Weisstein. Least squares fitting – power law. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/ LeastSquaresFitting-PowerLaw.html, 2005.
10. J. Zeil, M. Hofmann, and J. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.

Visual Homing in Natural Environments

Matthew Szenher
Institute of Perception Action and Behaviour
School of Informatics
University of Edinburgh
James Clerk Maxwell Building
The King's Buildings
Mayfield Road, Edinburgh, EH9 3JZ
M.Szenher@inf.ed.ac.uk

Abstract

In the past few years, a number of researchers (e.g. Zeil et al. (2003)) have found that, in a variety of natural environments, the difference between a reference intensity image and images captured nearby increases monotonically with distance from the reference location. The image difference as a function of spatial distance typically takes the form depicted in Figure 1(a) which we call a difference surface. Zeil et al. devised a novel visual homing algorithm which took advantage of the characteristic shape of difference surfaces. We propose in this paper that the shape of the difference surface springs from a particular quality of natural scenes. Our hypothesis leads to a mathematical description of difference surfaces. We use our derived function to make predictions about the efficacy of visual homing in various environments and under different conditions.

1 Introduction

Consistently successful autonomous navigation in outdoor environments remains a Holy Grail in robotics. We believe that a step towards this goal was taken in Zeil et al. (2003). This paper reported that, in static outdoor environments, the difference between two images (image difference is defined in Section 1.2) increases with the distance between the positions at which the images were captured. If an image is captured at a reference location and compared with images taken nearby, the plot of difference values as a function of image capture location with respect to the reference typically forms a surface like the one shown in Figure 1(a) (up to about 3m from the reference). We call this function a difference surface. Mitchell and Labrosse (2004) apparently independently discovered this phenomenon. Our purpose in this paper is to determine why difference surfaces take this characteristic shape in natural environments[1]. We believe that a particular quality of images of natural environments, discussed in Sections 2, leads to these characteristic difference surfaces.

Zeil et al. (2003) and Mitchell and Labrosse (2004) took advantage of the typical form of the difference surface to construct very simple, robust navigation algorithms which guide a mobile robot autonomously to the reference location in outdoor environments. Their navigation algorithms are a type of visual homing algorithms; visual homing is surveyed in Section 1.1. The homing algorithm of Zeil et al. is described in particular in Section 1.2. Our hypothesis, described in Section 2 leads to a parameterized mathematical model for difference surfaces in natural environments which is derived in Section 3. In Section 5 we use our parameterized difference surface model to make predictions about the relative ease with which homing should occur in different types of natural environments.

1.1 Visual Homing

Visual homing is a navigational technique allowing an agent to return to a location which it has visited previously. The agent begins at a target location at which it stores some visual information (frequently referred to as a snapshot image). The agent then meanders, perhaps completing certain tasks, until it is time to return to the target from its current location. The agent captures some visual information at the current location, compares this information to that stored at the target and uses the discrepancy to generate a homing vector. The homing vector generally leads the agent closer to the target, but not directly to it. Visual homing is an iterative process.

[1]We could not find an explicit definition of "natural environments" in the relevant literature. In the spirit of Huang and Mumford (1999), we define "natural environments" in this paper to be those environments not specifically designed for experimental purposes. Natural environments can contain natural and/or manmade structures.

Visual homing algorithms fall into two categories: feature-based or image-based. Feature-based homing schemes attempt to segment images into foreground (features or landmarks) and background. A homing vector is inferred from the difference in bearing and (sometimes) apparent size between paired features in current and snapshot images. Images must be aligned with an external compass reference or captured while the agent is in a fixed orientation in order for paired feature bearing differences to have meaning. Examples of feature-based visual homing algorithms include the snapshot model (Cartwright and Collett, 1983), the average landmark vector model (Mőller et al., 2001), Hong's homing scheme (Hong et al., 1991) and Basri et al.'s epipole surfing algorithm (Basri et al., 1998). Feature-based homing schemes require that the same landmarks are detected in both snapshot and current images, which are often captured at dramatically different viewpoints. With the exception of the average landmark vector model, feature-based homing schemes must then pair each landmark detected in the snapshot with a landmark extracted from the current image (i.e. the correspondence problem must be solved). Landmark correspondence and consistent detection are difficult in outdoor environments as they are often cluttered, have complex depth structure, are subject to highly dynamic lighting conditions, and contain many objects which change appearance with viewpoint (i.e. trees and flat rocks).

We believe image-based homing algorithms are preferable to feature-based schemes as the former make no attempt to extract landmarks from images, nor do image-based schemes require a solution to the correspondence problem; they instead use whole-image differences between snapshot and current views to generate a homing vector. Image warping (Franz et al., 1998) is a popular image-based homing scheme, considered by many to be the state of the art in visual homing (see e.g. Vardy and Oppacher (2004)). The image warping algorithm iterates through all possible values of agent rotation and translation which could have lead the agent from the goal pose to its current pose. For each set of parameters, the algorithm warps the current image into a candidate snapshot image, assuming that all imaged points are at an equal distance from the goal position. The parameters which yield the candidate snapshot most similar to the true snapshot are used to home. In Zeil et al. (2003), an image-based homing algorithm computationally less demanding than image warping was proposed; it is described below.

1.2 Difference Surfaces and Gradient Descent Homing

The notion of the difference between two images was introduced in Section 1. Zeil et al. (2003) defined image difference as the root-mean-square difference over all pixels (RMS), as follows

$$RMS(I_1, I_2) = \sqrt{\frac{\sum_{i=1}^{n}(I_1(i) - I_2(i))^2}{n}} \qquad (1)$$

where I_1 and I_2 are intensity images stored in arrays of n elements; both images are aligned with an external compass reference. (Note: some readers may know Equation 1 as the square root of a sum of squared differences (SSD).) When RMS values are calculated at all points on a plane with respect to a particular reference image, the result is a difference surface (e.g. Figure 1(a)). The surface's minimum value (0) coincides with the reference location.

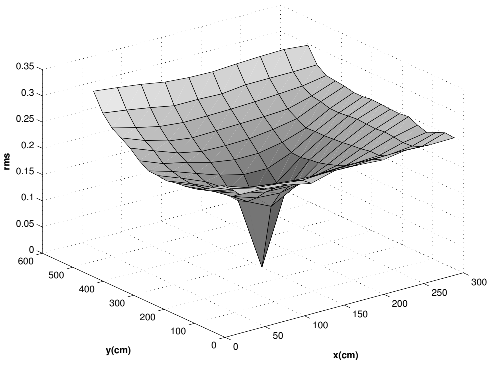
We confirmed that the results of Zeil et al. hold for a laboratory environment (Figure 1). Our data (provided by Andrew Vardy and used in Vardy and Oppacher (2004)) was collected in a laboratory whose periphery was lined with computers, posters, curtains, plants, etc. There were no obstructions in the experimental area itself (the floor). Using a small Webcam imaging a hyperbolic mirror, panoramic colour images were captured at 30 cm intervals in a 3m x 5.1m area on the flat laboratory floor. The area was illuminated by overhead, fluorescent lights. All images are aligned with an external compass reference.

Zeil et al. demonstrated empirically that a visual homing algorithm they called "Run-Down" successfully guides an agent back to a goal position. As usual, the agent captures a snapshot I_S at the goal position and meanders for a time before "deciding" to return to the reference. The agent chooses a random direction and moves in a straight line in this direction, capturing images periodically, first I_{C_0}, then I_{C_1}, then I_{C_2}, etc. After each image capture, the agent calculates $RMS(I_S, I_{C_i})$. The agent continues along this line until $RMS(I_S, I_{C_i}) < RMS(I_S, I_{C_{i+1}})$. At this point, the agent executes a 90° turn to the right and repeats the process. Given the shape of the difference surface, this 90° turn will be almost parallel to the gradient of the surface at the current position. The agent can sample RMS values in both directions on this new line to determine the direction of RMS decrease. The agent stops when the current value of $RMS(I_S, I_{C_i})$ falls below a certain threshold close to zero. In this way, the agent spirals in to the goal position.

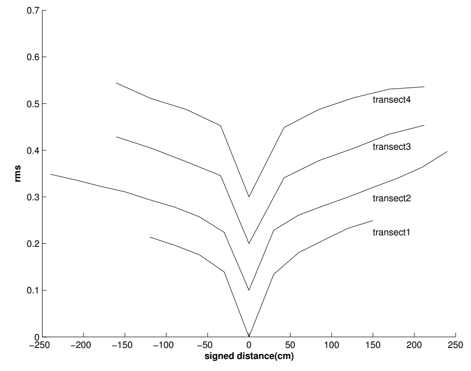
2 Pixel Correlations in Images of Natural Scenes

Ruderman (1997) reported an empirically derived property of images of natural scenes: that the expected value of the square of the intensity difference D between pixels separated by a visual angle x is

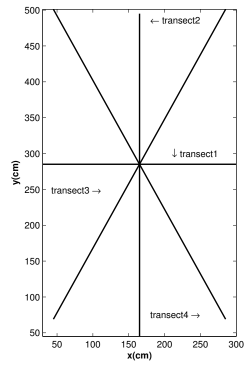
$$D(x) = D_1 - D_2 x^{-\eta} \qquad (2)$$



(a)



(b)



(c)

Figure 1: **a**: The difference surface for the reference image at x=150cm, y=270cm. Images were collected in the laboratory environment described in Section 1.2. **b**: Cross-sections of the difference surface shown in **(a)** along the transects shown in **(c)**. Each successive cross-section in **(b)** is shifted vertically by 0.1 units to improve readability. The signed distances in **(b)** indicate whether a particular point is to the left (negative) or to the right (positive) of the reference point.

where D_1, D_2 are η are constants for a given natural image derived from a power law least squares fitting of image data. Ruderman found mathematically that D_2 and η must have the same sign. This restriction accords with our intuition as it causes Equation 2 to be monotonically increasing with distance.

We hypothesize that the quality of natural images embodied in Equation 2 leads to difference surfaces of the form depicted in Figure 1(a).

That the difference between pixel intensities increases monotonically with distance within natural images is not surprising. The most likely explanation is that the world is composed of objects which frequently have constant or smoothly changing luminance across their surfaces. That the monotonic increase takes the form of a power law is more surprising. The interested reader should see Ruderman (1997) for a detailed explanation.

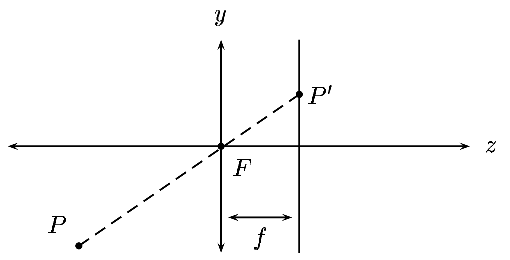


Figure 2: Our model imaging system consists of a focal point F at the origin of our coordinate system and an image plane at distance f from the y-axis along the z-axis. The x-axis of the coordinate system is orthogonal to the page. The origin of the image plane is at the intersection of the image plane with the z-axis. A point P at (P_x, P_y, P_z) is projected onto image plane point P' at $(\frac{P_x f}{P_z}, \frac{P_y f}{P_z})$ (a perspective projection).

3 From Pixel Correlation to RMS Surfaces

In order to apply the hypothesis given in Equation 2 above to our study of difference surfaces in natural environments, we must translate agent motion to corresponding pixel motion, otherwise known as optic flow. Our first step is to construct a model of the agent's imaging system; this is given in Figure 2. Using this camera model, we can deduce optic flow due to agent movement as follows: A point P in space at (P_x, P_y, P_z) will project to point $(\frac{P_x f}{P_z}, \frac{P_y f}{P_z})$ on the image plane. If our agent moves in direction $\vec{v} = (v_x, v_y, v_z)$ (a unit vector) by a distance d then the same point P will be projected on to the image plane at $(\frac{(P_x - dv_x)f}{P_z - dv_z}, \frac{(P_y - dv_y)f}{P_z - dv_z})$. Thus, the distance x (in pixel units) that P will have moved in the image due to corresponding movement by the agent $d\vec{v}$ is

$$x = \|\frac{P_x f}{P_z} - \frac{(P_x - dv_x)f}{(P_z - dv_z)}, \frac{P_y f}{P_z} - \frac{(P_y - dv_y)f}{(P_z - dv_z)}\| \quad (3)$$

We assume that the agent has not moved to a position where P is behind the image plane or is obstructed by another object.

In the foregoing calculations, we will assume that the agent moves parallel to the x-y plane (i.e. $v_z = 0$). Given this assumption, Equation 3 reduces to

$$x = \frac{fd}{P_z} \quad (4)$$

The assumption that $v_z = 0$ implies that points in the environment remain at a constant distance from the image plane (and so imaged objects will not change in apparent size) as the agent moves. This is generally an unrealistic assumption unless the agent is moving down a hallway or parallel to a forest edge. We hope to deal with the more complex case in future work.

Equation 4 allows us to study the expected value of Equation 1 using Equation 2 as an agent moves through a natural environment. We will actually look at the square of Equation 1 ($SRMS$) since it is more amenable to mathematical analysis.

Suppose our agent takes a snapshot image I_S, meanders for a time and starts homing at a distance d in direction $\vec{v}$ from the goal, at which it captures an image I_C. The expected $SRMS$ value in this situation is

$$\langle SRMS(I_S, I_C)\rangle = \langle \frac{\sum_{i=1}^{n}(I_S(i) - I_C(i))^2}{n}\rangle \quad (5)$$

Since the expected value is a linear operator,

$$\langle SRMS(I_S, I_C)\rangle = \frac{1}{n}\sum_{i=1}^{n}\langle (I_S(i) - I_C(i))^2\rangle \quad (6)$$

Moving a distance d in direction $\vec{v}$ will cause pixels in I_S to shift by amount x to form I_C, where x and d are related by Equation 4. The expected value of the square of the intensity difference between $I_S(i)$ and $I_C(i)$ is therefore given by Equation 2. Thus

$$\langle SRMS(I_S, I_C)\rangle = \frac{1}{n}\sum_{i=1}^{n}(D_1 - D_2 x_i^{-\eta}) \quad (7)$$

By the commutativity of addition and the definition of expected value, Equation 7 becomes

$$\langle SRMS(I_S, I_C)\rangle = D_1 - D_2\langle x^{-\eta}\rangle \quad (8)$$

Finally, according to Equation 4 and the linearity of the expected value operator, Equation 8 becomes

$$\langle SRMS(I_S, I_C)\rangle = D_1 - D_2(fd)^{-\eta}\langle P_z^{\eta}\rangle \quad (9)$$

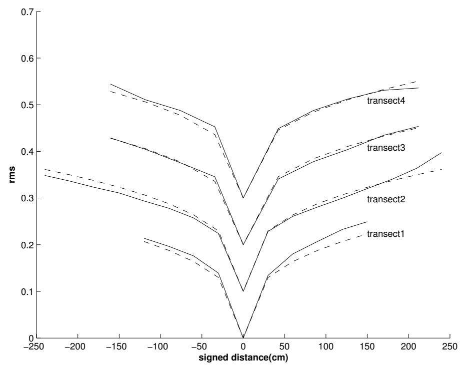


Figure 3: Equation 9 fit to the difference surface transects depicted in Figure 1(b). Solid lines are the observed transects and dashed lines are best-fitting curves. Transects are offset vertically for readability.

$\langle SRMS(I_S, I_C)\rangle$ is a power law in d, with constants D_1, D_2, f, η, and $\langle P_z^{\eta}\rangle$. Equation 9 is our parameterized difference surface model. If our hypothesis is correct, real-world difference surfaces should, on average, take this form.

4 Testing the Hypothesis

To test our hypothesis, we sought to determine if Equation 9 could be used to fit observed difference surfaces. The observed surfaces we used were generated from the image library described in Section 1.2. In each of a number of experiments, we chose a particular location on the image grid as our goal location. We then chose as test points transects similar to those shown in Figure 1(c). We chose 20 training points in a Gaussian distribution with standard deviation 60cm around the reference point; no test points, save the reference point, were used as training points. We then used a power law least squares fitting procedure to determine best fitting values for D_2 and η; D_1 was fixed at 0. Figure 3 shows the result of one such experiment, with reference location at x=150cm, y=270cm as in Figure 1. The best value for D_2 in this experiment was 0.0414 and the value of η was 0.3364. The fit to the test points is remarkably close, with root mean square error of 0.0104. We repeated this experiment at 50 different reference locations in the image library. The mean values of D_2 and η were found to be 0.0111 and 0.0805, respectivley. The average root mean square error between best-fitting curve and observed transects was found to be 0.0083 with a standard error of 0.0019. The close match between difference surface transects generated from Equation 9 – which springs from our hypothesis – and observed transects lends credence to our hypothesis.

5 Predictions

Equation 9 allows us to make predictions about the ease of gradient descent homing in environments characterised by D_1, D_2 and η. As noted before, D_2 and η always have the same sign (except when η is zero, in which case Equation 2 takes a different form which does not include D_2). For positive D_2 and η

$$\lim_{d \to \infty} D_1 - D_2(fd)^{-\eta}\langle P_z^{-\eta}\rangle = \infty \quad (10)$$

But when D_2 and η negative

$$\lim_{d \to \infty} D_1 - D_2(fd)^{-\eta}\langle P_z^{-\eta}\rangle = D_1 \quad (11)$$

Thus, for positive D_2 and η we expect successive difference values calculated during the homing process to be generally more distinct relatively far from the goal than in environments with negative D_2 and η. Homing may thus be less affected by noise in difference measurements in environments with positive D_2 and η, thus facilitating the homing process. Unfortunately, a search of the literature revealed no mapping between type of environment (e.g. forest, seaside, Martian) to signs of η and D_2 so we cannot as yet make a prediction as to which environments are more suited to visual homing by gradient descent.

Ruderman (1997) provided convincing evidence that η (and perhaps D_1 and D_2 as well) in Equation 2 are largely unaffected by any image calibration/filtering which does not spatially decorrelate object images. If this is indeed the case, Equation 9 should be unaffected by such calibration as well. Some experiments reported in Zeil et al. (2003) make us doubt this invariance in difference surfaces. Zeil et al. found that sudden changes in illumination and/or movement of objects near the agent degraded the difference surface so that it was no longer unimodal and/or that the minimum moved from the reference location. The disconnect between the calibration invariance of Equation 2 and the sensitivity of actual difference surfaces to environmental conditions must be investigated.

Equation 9, our parameterized difference surface model, depends on Equation 2. We believe that the most likely explanation for Equation 2 is that natural images are generally composed of objects with constant or smoothly changing luminance. But some portions of some natural images lack this kind of structure. For example, a uniformly overcast sky contains almost no objects and grassy ground contains only small objects. We predict that visual homing will be more successful if information-poor regions of images such as these are detected and ignored.

6 Conclusions and Future Work

We have argued that difference surfaces take the characteristic form exemplified by Figure 1(a) because natural images exhibit the quality embodied in Equation 2. From this hypothesis we derived a parameterized equation which fits observed difference surfaces very closely. We made predictions about the efficacy of homing under various conditions given this difference surface model.

Our future work will begin with testing the predictions we made in Section 5. Specifically, we will seek to determine whether environments which exhibit positive values of D_2 and η are more amenable to visual homing by gradient descent than environments which exhibit negative values of these parameters. We will also seek to determine whether the detection and deletion of sky and ground regions of images will improve gradient-descent homing.

We would like to collect image libraries like the one described in Section 1.2 in outdoor environments. After doing so, we will repeat the experiment described in Section 4 using this data and, hopefully, provide further validation of our hypothesis. Another interesting test would involve estimating the values of D_1, D_2 and η for a particular environment from a number of still images of that environment, as Ruderman did. We could then plug these parameter values into Equation 9 and compare the model difference surfaces obtained with observed difference surfaces.

We have modelled difference surfaces with the assumption that environmental conditions are static. That is, we have supposed that the illuminant is unchanging and the locations of objects in the environment are fixed. This is obviously not the case in real outdoor environments and, as noted in Section 5, Zeil et al. (2003) reported that environmental dynamism alters the shape of the difference surface so that homing to the goal position is no longer possible. Zeil et al. also reported that the difference surface with respect to a particular reference point frequently regains its characteristic shape after such a period of deformation. We would like to study the long term temporal statistics of natural scenes to determine the variability of these statistics over the period of hours or days (the operational lifespan of a mobile robot). To our knowledge, temporal statistics of natural scenes have only been examined in the short term (on the order of a few seconds; see e.g. Dong and Atick (1995)).

Observed difference surfaces in static or dynamic environments will be subject to noise from a number of sources: measurement error, object occlusions, illumination changes, movement of shadows, movement of nearby objects due to wind and/or human intervention, etc. There are optimization techniques which purport to effectively find the minima of noisy multivariate functions. One such algorithm, compass search (see e.g. Kolda et al. (2003)), samples the function at a set distance in the four cardinal directions from a particular point. This multi-directional sampling is more time-consuming but potentially more robust than Zeil et al.'s uni-directional "Run-Down" algorithm. We hope to use compass search to home in outdoor environments.

Finally, we firmly believe that it is possible to prove that "Run-Down", when applied to a function of the form of Equation 9 (or some slightly modified version of this function), is guaranteed to halt within a certain distance from the goal position. Zeil et al. did not provide such a proof. In the literature on function optimization, "Run-Down" is classed as a line search optimization technique. Kolda et al. (2003) describes necessary and sufficient conditions on the search direction and step size required for line search techniques to converge to an optimum. We believe it is possible to prove that "Run-Down" meets those conditions.

Acknowledgments

I would like to thank Andrew Vardy, PhD candidate at the School of Computer Science, Carleton University, for providing the image library described in Section 1.2. Thanks also to Barbara Webb – my PhD supervisor – and Jan Wessnitzer for providing insightful comments on early drafts of this paper.

References

Basri, R., Rivlin, E., and Shimshoni, I. (1998). Visual homing: Surfing on the epipoles. In *ICCV*, pages 863–869.

Cartwright, B. and Collett, T. (1983). Landmark learning in bees. *Journal of Comparative Physiology*, 151:521–543.

Dong, D. and Atick, J. (1995). Statistics of natural time-varying images. *Network Computation in Neural Systems*, 6(3):345–358.

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

Hong, J., Tan, X., Pinnette, B., Weiss, R., and Riseman, E. (1991). Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 620–625, Sacremento, CA, USA.

Huang, J. and Mumford, D. (1999). Statistics of natural images and models. In *Computer Vision and Pattern Recognition-Volume 1*, page 1541, Fort Collins, CO, USA.

Kolda, T., Lewis, R., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *Society for Industrial and Applied Mathematics Review*, 45(3):385–482.

Mőller, R., Lambrinos, D., Roggendorf, T., Pfeifer, R., and Wehner, R. (2001). *Insect Strategies of Visual Homing in Mobile Robots*, chapter 3. The MIT Press, Cambridge, Massachusetts.

Mitchell, T. and Labrosse, F. (2004). Visual homing: a purely appearance-based approach. In *Proceedings of TAROS (Towards Autonomous Robotic Systems)*, The University of Essex, UK.

Ruderman, D. (1997). Origins of scaling in natural images. *Vision Res.*, 37(23):3385–3398.

Vardy, A. and Oppacher, F. (2004). A scale invariant neural feature detector for visual homing. In *Proceedings of the Workshop on Neurobotics, German Conference on Artificial Intelligence.*

Zeil, J., Hofmann, M., and Chahl, J. (2003). Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469.

Adaptation of Controllers for Image-Based Homing

Markos Zampoglou, Matthew Szenher and Barbara Webb

Adaptive Behavior 2006; 14; 381
DOI: 10.1177/1059712306072338

Adaptive Behavior

Adaptation of Controllers for Image-Based Homing

Markos Zampoglou[1], Matthew Szenher[2], Barbara Webb[2]
[1]*Department of Applied Informatics, University of Macedonia*
[2]*Institute of Perception Action and Behaviour, School of Informatics, University of Edinburgh*

Visual homing is a short-range robot navigation method which can lead an agent to a position with accuracy, provided that the majority of the scene visible from the home position is also visible from the current robot position. Recently Zeil, Hoffmann and Chahl (2003) showed that a simple calculation—the root mean square (RMS) difference between the current image and the home image—produces a monotonic function leading to the home position for natural images. In this article we propose a gradient descent algorithm based on *Caenorhabditis elegans* chemotaxis (Ferree & Lockery, 1999) for homing with the RMS signal. The parameters for this algorithm are evolved for a simulated agent, and the resulting homing behavior compared with alternative algorithms in simulation and using a real robot. A simulated agent using this algorithm in an environment constructed from real world images homes efficiently and shows generalization to variations in lighting and changes in the scene. In the real robot this algorithm is affected by noise resulting from imperfect sensors, and alternative algorithms appear more robust. However, the best performing algorithm for unchanging environments, image warping (Franz, Schölkopf, Mallot, & Bülthoff, 1998), is completely disabled by scene changes that do not affect algorithms utilizing the RMS difference.

Keywords homing · taxis · image difference

1 Background

1.1 Introduction

The problem of visual homing can be stated as follows. An agent, equipped with a monocular camera, is located at a point S. The goal is for the agent to move to a home position C, using a home image taken from location C and the current visual information from the camera (Figure 1) to determine its movement. Visual homing is a short-range robot navigation method which can lead an agent to a position with accuracy, provided the majority of the scene visible from the home position is also visible from the current robot position.

Most approaches to visual homing use an omnidirectional image of the environment (Franz & Mallot, 2000), typically obtained by using a spherical, parabolic or conical mirror located above a camera pointing upwards. Assuming movement takes place on a flat surface, there is a circle on the mirror, centerd on its central vertical axis, where the angle between the line of sight and the surface normal of the mirror is equal to the angle between the ground plane and the mirror surface. For points projected on this circle, the line of sight is reflected in a direction parallel to the ground plane. For object whose projections fall on this circle, these projections will change their bearing in the circle as the agent moves, but will never leave this

Correspondence to: Barbara Webb, IPAB JCMB, Mayfield Road, Edinburgh EH9 3JZ, UK. *E-mail*: bwebb@inf.ed.ac.uk
Tel.: +44 131 651 3453; *Fax*: +44 131 651 3435

Copyright © 2006 International Society for Adaptive Behavior
(2006), Vol 14(4): 381–399.
DOI: 10.1177/1059712306072338

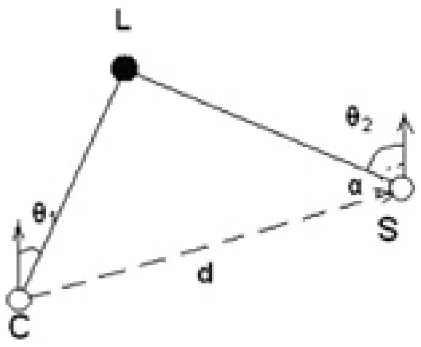


Figure 1 The homing task is to move the agent from S in the direction of C. In the simplest case, visual information is the relative location of the projection landmark L; this allows the home direction to be calculated from simple geometry.

circle (Figure 2). The visual information used in many approaches is the one-dimensional image created by unfolding this circle into a line.

Existing visual homing methods can be divided into two groups: feature-based methods, where specific features are extracted from the images and matched between the current and home images, and image-based methods, where the image pixel values are used directly. Feature-based methods include the snapshot model (Cartwright & Collett, 1983; Hong, Tan, Pinnette, Weiss, & Riseman, 1992; Weber, Venkatech, & Srinivasan, 1998), and the average landmark vector method (Lambrinos, Möller, Labhart, Pfeifer, & Wehner, 2000; Möller, 2000), while the most popular image-based method is image warping (Franz et al., 1998).

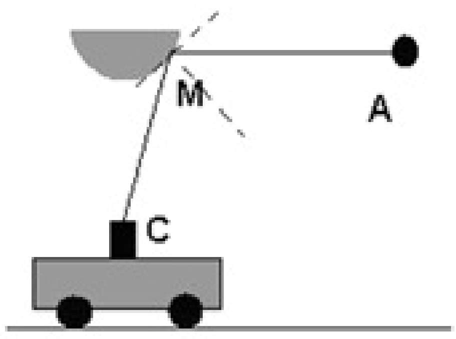


Figure 2 Capturing a horizon line using a spherical mirror mounted above a camera. If MA is parallel to the ground plane, the projection of A will change its bearing with respect to a circle centerd on the vertical axis of the mirror, but will never move out of this circle.

The image warping algorithm works under the assumption that all landmarks are situated at equal distances from the home location. Using a grayscale one-dimensional horizon image taken from the current location, the model calculates all possible deformations of the image line, for all possible rotations and movement vectors. The agent then moves according to the movement vector that produces the image most closely resembling the home image. The measure of resemblance suggested by Franz et al. is the dot product between the two (normalized) image lines. Image warping is a very effective and robust algorithm, even in cases where the equal distance assumption does not hold. Its major disadvantage, however, is its computational cost, since it has to perform a full search at every homing step.

1.2 The Root Mean Square (RMS) Difference Surface

Recently, Zeil et al. (2003) studied the properties of outdoor scenes for the purpose of image homing, using a panoramic camera in the natural habitat of a wasp that is known to locate its nest by visual homing. They captured a home position image, and similar images from various positions in 3-D space, at different times of day. Instead of extracting the horizon line, the whole panoramic images were unfolded, and then each image was compared with the home image using the root mean square difference function. That is, the difference between the current image and the home image was determined by:

$$RMS = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(I_C(i,j)-I_S(i,j))^2}{M\times N}}, \qquad (1)$$

where $I_C(i, j)$ is the intensity function for pixel (i, j) in the current image, $I_S(i, j)$ is the intensity function for pixel (i, j) in the home image, and $M \times N$ are the dimensions of the images. This function applies to greyscale images, but can be generalized for color images.

Zeil et al. estimated the RMS difference for a grid of camera positions (x, y, z) in a given area, compared with a home image taken from the centre of the area. In our work, however, the camera remains at a constant height z. The 3-dimensional function resulting from the (x, y) pairs and the corresponding RMS values will be referred to from now on as the RMS difference surface.

The resulting difference surface was unimodal in both dimensions, for a certain area around the home position. That is, in dimensions x and y, movement towards the home position resulted in gradual reduction of the difference, reaching a global minimum in the target location, with no local minima whatsoever. The size of the area for which this held true (the "catchment area") was up to 3 square meters for the specific experiment (the term "catchment area" in visual homing refers to the area that contains all the start positions from which an agent can home).

The same characteristic applied to the dimension z, and also for rotation. Regardless of the relative position of the agent with respect to the goal position, the RMS difference was reduced as the current height approached the height at which the home image was taken, and was also reduced as the current orientation approached the orientation of the home image. In each case, the RMS difference reached a global minimum when the current and home coordinates matched. Labrosse (2004) did an extensive study of this property of the image difference and its potential use as a compass.

1.3 Homing as Gradient Descent

As Zeil et al. suggest, it is possible to take advantage of the unimodal property of the catchment area to perform visual homing. The agent can determine its orientation by turning until the minimum RMS difference is encountered and move towards the home location by descending towards the point that gives the minimum RMS difference. Homing can thus be regarded as a gradient descent problem.

Function minimization by gradient descent (also known as steepest descent) in general works by determining the negative of the gradient of the function at the current position and moving in this direction until a line-minimum is reached (Weisstein, 2006). The algorithm recalculates the negative of the gradient at the new position (which is orthogonal to the current direction of movement) and minimizes the function along this new direction. This process is repeated until the magnitude of the gradient is very near zero, indicating proximity to a local minimum.

However, if we are trying to home by having a robot descend the surface formed by the RMS calculation for various positions, the only information available to the robot is the current input and the sequence of the inputs so far. This means that we cannot easily calculate the gradient of the function at a given position. The only higher-order information we have is an estimate of the local derivative of the function in the direction that the agent previously moved, which can be extracted through the difference between the current and the previous input. Based on this, Zeil et al. (2003) proposed the RunDown algorithm, which is a variation of the gradient descent algorithm, adjusted for the particularities of the problem. The agent moves in a random direction for a programmer-determined step length. If the RMS difference decreases, the agent moves forward again. Otherwise, the agent turns 90 degrees in one direction and moves forward again. If the step size is sufficiently small, the agent will effectively move in one direction until the function in that direction reaches its minimum value, then turn in a direction orthogonal to the previous one and repeat, thus achieving a good approximation of gradient descent. However, in this method, the resulting path lengths are large relative to the initial distance of the agent from the goal.

1.4 Taxis as Gradient Descent

In an attempt to find a more efficient way to take advantage of the properties of the RMS difference surface for the purpose of visual homing, we turned towards a biologically inspired approach. Chemotaxis is a process used by some organisms to locate sources of food. Ferree and Lockery (1999) define it as "oriented movement in response to a chemical gradient." In effect, during chemotaxis, an organism performs gradient ascent from an area of low concentration of a chemical substance, to the global maximum of concentration. Organisms such as *Caenorhabditis elegans* perform chemotaxis using a single sensor to detect only the current concentration of the chemical substance, and thus solve an analogous problem to a robot trying to home using only the current RMS difference. In this sense, *C. elegans* chemotaxis is a gradient ascent method more suitable for robot homing than the various mathematical models, since it takes into account the particularities of the real world (single sensor, need for a minimal path length). By imitating the chemotaxis mechanism, in this case for gradient descent instead of ascent, the robot should be able to home.

In Ferree and Lockery (1999), the neural network that controls the taxis behavior of *C. elegans* is studied, and the underlying computational rules are extracted. It is demonstrated that the behavior of *C. elegans* during taxis can be modeled by

$$\frac{d\theta}{dt} = \Omega_{bias} + z_0 C(t) + z_1 \frac{dC}{dt}, \qquad (2)$$

where $d\theta/dt$ is the turning rate (i.e., the angle turned at a time step), Ω_{bias}, z_0, and z_1 are constant parameters, $C(t)$ the input at time t, and dC/dt the temporal gradient, which can be estimated as

$$\frac{dC}{dt} = C(t) - C(t-1). \qquad (3)$$

Function (2) can be used to produce gradient descent on an input surface, using only local information, provided the function parameters are properly selected. The agent can thus home by moving forward some distance at each time step, and then turning according to the output of (2). In most forms of gradient descent or ascent, a decreasing step size is desirable, so that the agent can approach the goal rapidly and then slow down to locate it with the desired accuracy. This can be implemented for gradient descent by making the forward movement at each time step proportional to $C(t)$, that is, $C(t)$ multiplied by some constant U.

To use this Taxis algorithm it is necessary to find appropriate parameter values of Ω_{bias}, z_0, z_1 and U. In this article, we used an evolutionary strategy to evolve these parameters, using a simulated agent homing on RMS difference surfaces calculated from previously collected images in a robot lab environment consisting of an empty space surrounded by desks with desktop computers on them. We then tested the generality of the evolved algorithms on different surfaces and compare their performance to the RunDown algorithm. We then compared the Taxis algorithm, RunDown and image warping on a physical robot performing homing in our lab.

2 Methods

2.1 Environment for Simulated Homing

A number of image datasets (available at http://www.ti.uni-bielefeld.de/html/research/avardy/index.html), courtesy of Andrew Vardy (Memorial University, St. John's) were used to build a simulated environment in which the agent would move. These sets were omnidirectional images of the same 3 × 5.1 m area (an office room), and consisted of three sets with the same pictures captured under different illumination levels ("Original", "Twilight", "Night"), and another set where a number of chairs had been added to the room ("Chairs") (Vardy & Möller, 2005). The images corresponded to a real world grid with 30 cm distances between every two consecutive images.

For a controller to be robust, we have to ensure that homing can be achieved not only when the current illumination conditions are identical to the conditions under which the home image was captured, but also when there are significant differences between the two. That is, the agent would ideally be able to home in twilight using a home image taken in full daylight. To create these conditions for the simulated agent, all possible RMS surfaces were formed, by comparing a home image to the image set it was drawn from (e.g., a home image from the "Original" set, compared with all the images in the "Original" dataset) and comparing a home image with a different image set (e.g., a home image from the "Original" dataset, subtracted from all the images of the "Night" dataset). Of the total 16 surfaces formed, 9 surfaces were chosen for use in simulation. They are listed in Table 1. To normalize the values we adopted the technique used by Zeil et al. The pixels of the home image were shuffled, so that a new random image was created. That image had the

Table 1 The surfaces on which homing was tested. The surfaces marked * were used for evolving the controllers.

Surface	Home image from	Current images from
O_O*	"Original"	"Original"
O_T	"Original"	"Twilight"
O_N*	"Original"	"Night"
O_C*	"Original"	"Chairs"
N_O*	"Night"	"Original"
T_N	"Twilight"	"Night"
C_O*	"Chairs"	"Original"
C_C*	"Chairs"	"Chairs"
N_C	"Night"	"Chairs"

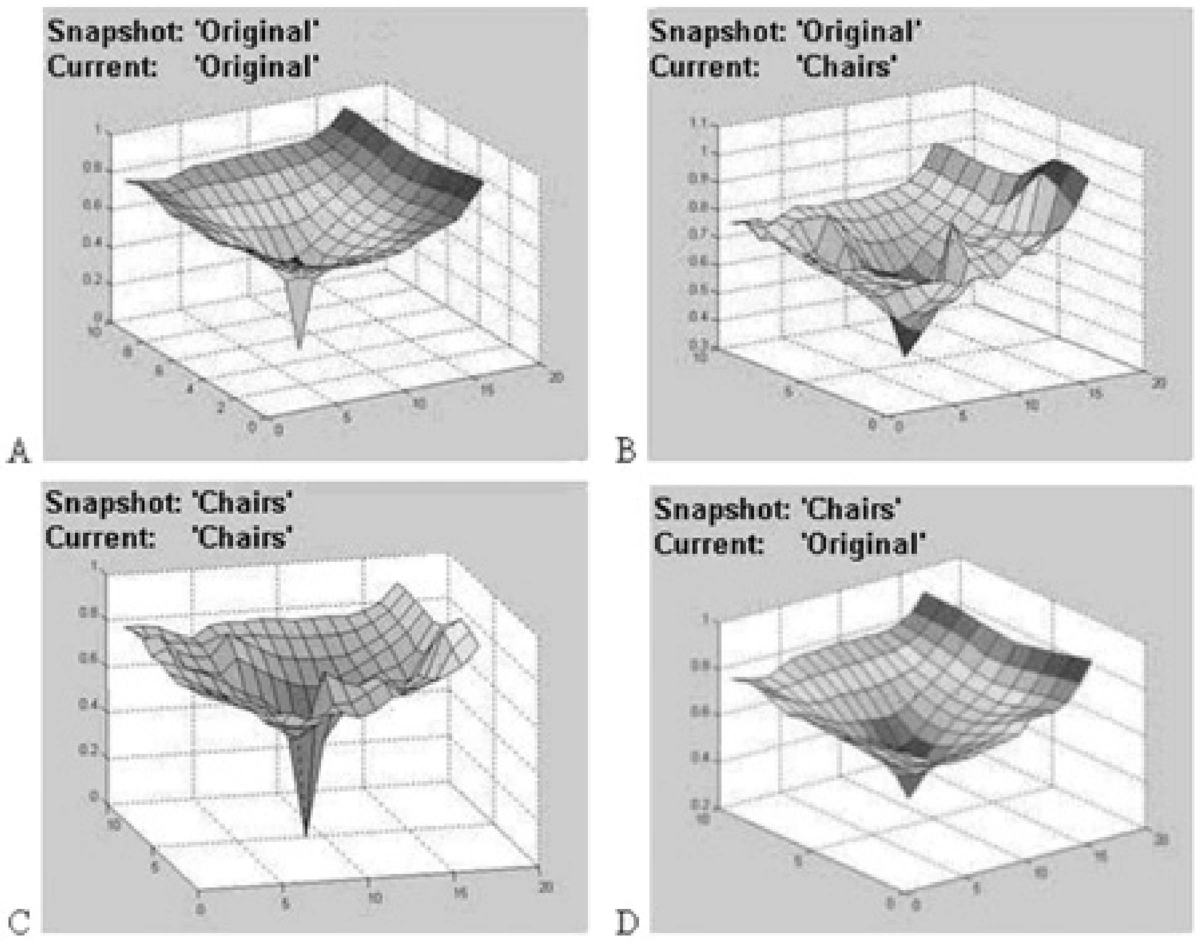


Figure 3 Four characteristic root mean square (RMS) difference surfaces. A) RMS difference between a home image from the “Original” set and every image in the “Original” set. B) RMS difference between the same home image and the “Chairs” set. C) RMS difference between a home image from the “Chairs” set and the “Chairs” set. D) RMS difference between a home image from “Chairs” and the “Original” set. Each unit in the axes corresponds to 0.3 m.

same pixel value distribution as the home image. On the other hand, because of the change in the position of each pixel, the RMS difference between the original and the randomized image gave a good estimate of the maximum RMS difference for that particular home position. All RMS differences were then normalized through division by that maximum value.

The fundamental property of the global minimum appearing at the home position held true for all surfaces (Figure 3A). Excluding the “Chairs” dataset, the unimodality of the surface also held true. In all the surfaces where the “Chairs” dataset was used as the current image set, although the general cusped form was retained, a number of local optima always appeared (Figure 3B, 3C). At this point, it interesting to note that, when using a home image from the “Chairs” set and current images from the same set, local minima and maxima still appear (Figure 3C), while when using a home image from the “Chairs” set, and the current images from “Original” (Figure 3D), no local optima can be seen. This suggests that local optima appear not when there are differences between the home image and current image set, but when objects (chairs) are close to the camera, so that moving the agent causes significant deformations that disrupt the catchment area.

2.2 Optimization Through Evolution

Having built the surfaces around which the simulated agent would move, we wanted to optimize the four parameters (Ω_{bias}, z_0, z_1, and U) of the Taxis algorithm described in Equation (2). Individuals were encoded by four real numbers, and, as a result, a high mutation probability (0.6) was necessary, to ensure that a large range of values was covered. Mutation involved changing each parameter by a random value drawn from a uniform distribution, ranging from –0.05 to 0.05 for the first three parameters and from –0.0175 to 0.0175 for U. One point crossover was used with the fixed probability 0.6. When crossover took place, a crossover point was randomly chosen in one of the three possible positions, and the parameters of the two individuals chosen for crossover exchanged place. The selection method used was stochastic universal sampling (Mitchell, 1998), and the population size was 200.

The fitness was evaluated by running each individual for a maximum of 500 time steps on six different surfaces (out of the nine possible home image–current image combinations) and seeing if, and in how many steps, an individual could reach the home position. At each simulation time step, the agent reads the current normalized RMS difference from the grid position corresponding to its actual position. As the original data only provides values at 30 cm intervals on the grid, the value at intermediate positions was interpolated from the nearest data values. The agent then estimates the new turning angle using Equation (2) and moves forward by $C(t) \times \text{U}$. A run was considered successful when the agent's distance from the goal was less than a threshold D (= 0.1 m). The fitness score was *500–number of steps to reach home*, with failure to reach home within 500 steps scored as 1. To reduce dependence of the evolved solutions on details of the simulated environment, we applied Jakobi's "radical envelope of noise" methodology (Jakobi, 1997).

According to this methodology, when evolving a robot controller in a simulated world for real-world application there are three steps to be taken:

(1) Identify a base set of robot–real-world-environment interactions, which have a basis in reality, and explicitly separate them from the simulation implementation aspects.
(2) Ensure that the implementation aspects vary significantly from evaluation to evaluation, so that the controller learns to ignore them and focus on the base set (i.e., ensure the simulation is *base set exclusive*).
(3) Ensure that every aspect of the base set varies slightly from evaluation to evaluation so that the controller is forced to cope with the differences between the simulation and the real world that result from noise (i.e., ensure the simulation is *base set robust*).

In our case, the base set consisted of only two interactions: the RMS difference input from the world, and the robot's movement in it.

The second step was implemented partly by the use of the different surfaces, as mentioned above, and also by using randomly chosen start positions for homing from the circle of locations a distance R (= 1 meter) from the home position. For the third step, Gaussian noise was added to the RMS value, the agent's turning angle and the displacement at each time step. The final parameters of the noise distribution were taken from the actual Koala robot (Figure 4) that was to be used for the real-world experiments, and were N(0, 0.002) for the RMS input and N(0, 0.0005) for both the turning angle (in radians) and the displacement (in the simulation 1 distance unit = 0.3 meters), where N(μ, σ) indicates a normal random distribution with mean μ and standard deviation σ.

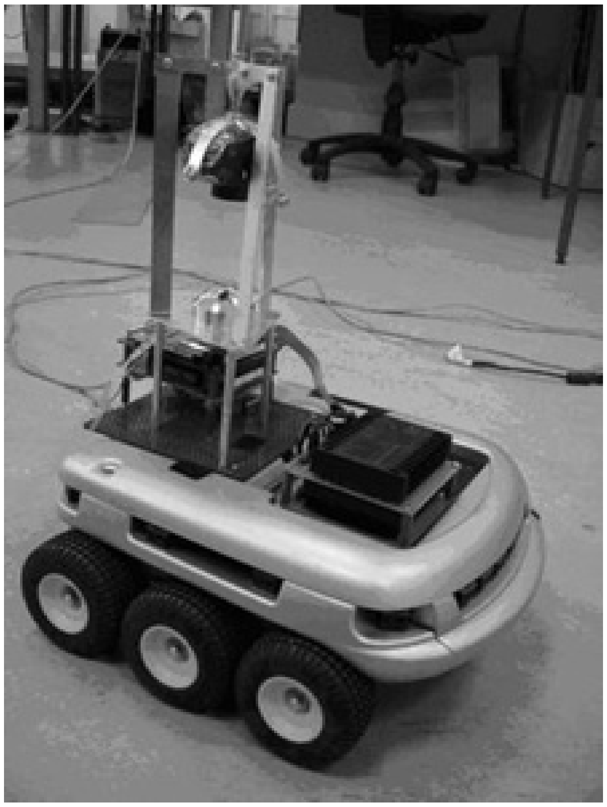


Figure 4 The Koala robot used in the experiments.

The RunDown algorithm was also implemented for the simulated agent. In this, the agent moves a fixed distance in an initial random direction, and if the RMS difference decreases, continues in that direction. If it increases, on the next step the agent moves in a direction 90 degrees to the right of the previous direction. One alteration was made with respect to the implementation described in Zeil et al. (2003). This was to use a variable step size, that is, $C(t) \times U$, rather than a fixed step size, to make the results more directly comparable to the Taxis algorithm. Both algorithms were then tested on all nine surfaces, and compared for homing success rate, time-steps required, total distance traveled, total turning, and final homing precision.

2.3 Robot Implementation

A Koala robot was used for real-world evaluation comparison of the Taxis, RunDown and image warping algorithms. Images were captured using a Creative Labs webcam pointing downward at a parabolic mirror manufactured by Kugler. The camera was supported on a rig consisting of three narrow pillars, which were visible in the image but did not appear to significantly affect any of the algorithms. The webcam connected via a USB port to a Dell Inspiron 7500 laptop with an Intel Pentium III processor, running Mandrake Linux 8.2. All image processing, calculations and generation of movement commands were programmed in C on the laptop. Motor commands were sent via a serial link to the built-in robot microprocessor for execution. The robot proved very consistent and accurate in its turning angle and movement distance, well within the noise parameters used in the simulation.

As well as the Taxis and RunDown algorithms, the image warping algorithm was also implemented on the robot, as follows: Given rotation ψ, home vector angle α and the ratio ρ of the agent's distance from the home position d to the home position's distance from the landmarks R, that is, $\rho = d/R$, a pixel located at angle θ in the current horizon image will be displaced by

$$\delta = \arctan\left(\frac{\rho\sin(\theta-\alpha)}{1-\rho\cos(\theta-\alpha)}\right) - \psi . \tag{4}$$

By shifting all pixels in the current image by their respective δ values, we build an approximation of the image, had the agent moved and rotated according to α and ψ, and had ρ been a good estimate of the actual agent and landmark distances. Searching over all ψ, α and ρ, the best match to the home image is found, and the corresponding home vector used to move the robot for a fixed distance. The termination of homing is signaled when the home vector changes by > 170 degrees, which indicates the robot has just passed over the home position. The only free parameter is the step size, which was set to 0.25 meters.

It proved necessary to rerun the parameter optimization procedure for the Taxis algorithm on the real robot, as the RMS differences measured by the robot differed in several ways from the surfaces from the Vardy dataset (see Section 3.3). We explored a fast method for obtaining an estimate of the surface properties that could be used for optimization, which does not require capturing a complete image grid. The robot captured a home image, and then captured a line of RMS differences by moving outwards in one direction and capturing images at equal distances. The surface was then created by assuming symmetry, that is, rotating this line of values through 360 degrees. Since a Taxis controller only has to adapt its parameters to the steepness and value range of a surface, it was assumed that any particularities of an area (e.g., local optima) could be ignored. By capturing the RMS difference line for each of the four directions and evolving a controller that could home on all four corresponding (artificial) surfaces, the controller was optimized for the properties (steepness, value range) of the given area. Initially, four surfaces were built in this way, for a particular home position and movement in four different directions, and the genetic algorithm (GA) was run on all four using the simulated agent. Since, in the original simulation, it was possible to evolve controllers on certain surfaces that were able to home on other, similar ones, it was assumed that a similar result could be achieved for the real-world controllers. However, this was not the case. The optimal controller for one surface was often unable to home on another one. However, the similarities between the principal characteristics of the surfaces ensured that the final population of 200 individuals from the GA run on the four initially sampled surfaces included at least some individuals that could home on other, similar, areas. Thus, every time the environment changed, the RMS surface for the particular environmental setup was again sampled for each of the four directions, and each of the 200 individuals was reevaluated on it in simulation. The best individual was then used to set the parameters for the evaluation of homing on the robot.

3 Results

3.1 Results of Evolving the Simulated Agent

The GA was able to come up with an efficient controller in about 250 generations. The controller parameters from three different runs of the GA appear in Table 2.

It is possible to directly interpret these parameters, following the discussion of Ferree and Lockery (1999) who used the same function to control chemotaxis of simulated nematodes. Recall that the control rule is:

$$\frac{d\theta}{dt} = \Omega_{\text{bias}} + z_0 C(t) + z_1(C(t) - C(t-1)), \quad (5)$$

where $d\theta/dt$ is the turning rate (i.e., the angle turned at a time step), and $C(t)$ the input at time t. After turning the agent moves forward by the distance $U \times C(t)$.

In all three controllers the bias Ω_{bias} and the first-order term z_1 had the same sign. This meant that, for a certain value range of $C(t) - C(t-1) < 0$, the bias and the first-order term cancel each other out, resulting in a small or zero turning rate when the gradient is decreasing. As the values moved out of that range, the agent is forced to turn in steeper angles. As Ferree and Lockery point out, the role of the first-order term causes a behavior called *klinotaxis*, described as “a change in turning rate in response to the spatial gradient of a stimulus field.”

In a typical RMS surface, the gradient is not constant, but grows steeper as we approach the home position. Near the home position, the RMS values are small, and it is klinotaxis that plays the most important role in navigation. However, when the agent is away from the home position, the gradient is small, and the RMS values are high. In that case, it is the zero-order term z_0 that plays the most important role, changing the turning rate proportionally to the input itself. This behavior is defined as *klinokinesis*: “A change in turning rate in response to the scalar value of a stimulus field” (Ferree & Lockery, 1999). The sign of the zero-order term can be either positive or negative, without seeming to follow any particular pattern. This is because the input is always positive, and, since we are dealing with angles, reducing an angle towards 0 is the same as increasing it towards 360 degrees. As the agent moves towards the home position, klinotaxis becomes more and more important, while the role of klinokinesis is reduced to simply adjusting the effects of the first-order term. Note that $C(t) - C(t-1)$ is generally much smaller than $C(t)$ itself, hence z_1 is correspondingly larger than z_0.

3.2 Evaluating the Controllers in Simulation

In order to evaluate the homing abilities of the controllers, we performed 1,000 runs of each controller on each of the nine surfaces, including the six on which the controller was evolved, and the three not used in evolution. The RunDown algorithm was also tested on the same surfaces for comparison. The controllers were evaluated with respect to five different measures: the homing success rate; the mean number of steps; the total distance traveled; the total angle turned; and the homing precision achieved. In these evaluation runs, the agent was considered to have homed when the RMS difference became smaller than a preset threshold (= 0.42).

A sample run of each controller appears in Figure 5. The homing success rate for each controller appears in Table 3. There is in fact one common factor for the sur-

Table 2 Evolved parameters for three different runs of the GA.

Controller	Ω_{bias}	z_0	z_1	U (in 0.3 m)
1	0.6490	0.9485	44.6288	0.6744
2	1.3821	–0.7951	37.0456	0.6492
3	1.3250	–0.7987	36.1076	0.9403

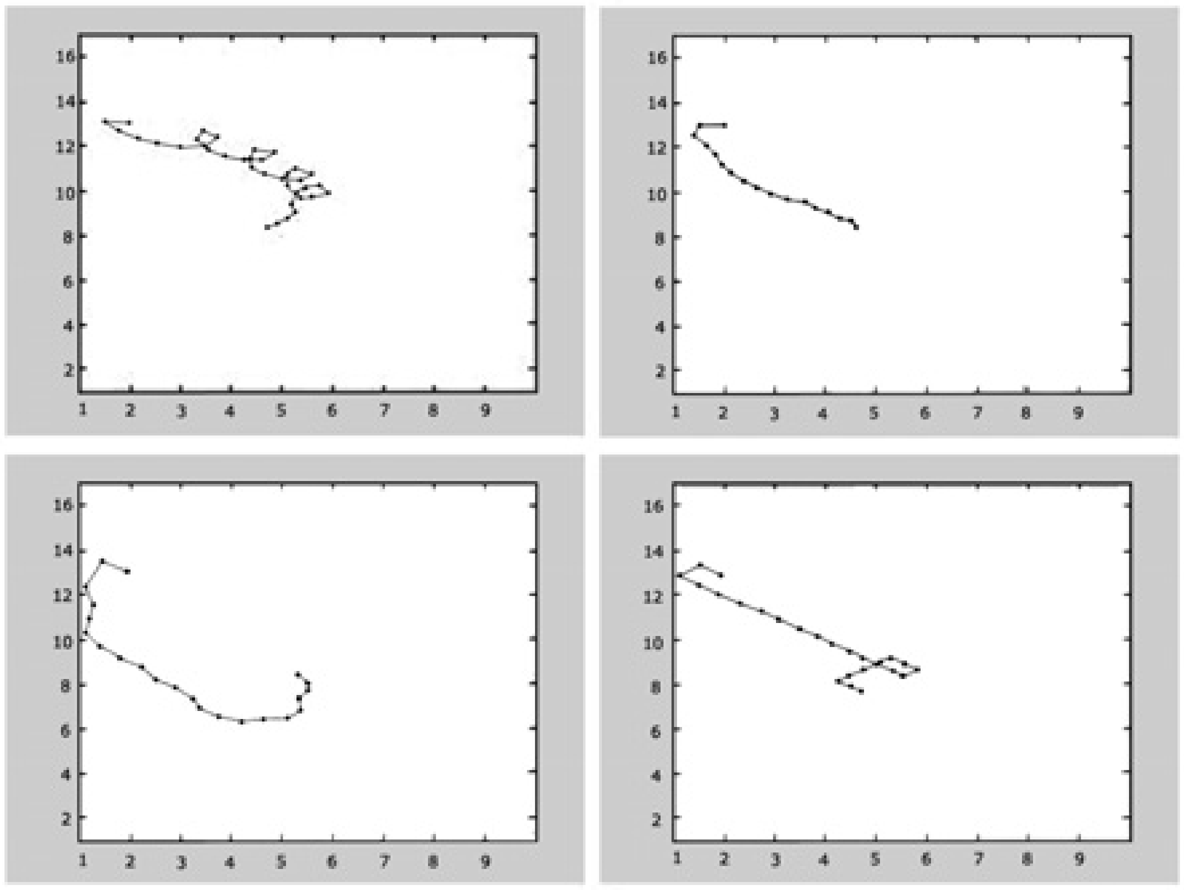


Figure 5 A sample run with each controller on surface N_O. Upper left: Controller #1. Upper right: Controller #2. Bottom left: Controller #3. Bottom right: RunDown.

Table 3 The success rate for each controller and each surface, 1,000 runs per surface.

	Controller				
Surface	1	2	3	Overall	RunDown
O_O	100%	100%	100%	100%	100%
O_T	100%	100%	100%	100%	100%
O_N	100%	100%	100%	100%	100%
O_C	90.6%	94.7%	84.9%	90.07%	78.6%
N_O	100%	100%	100%	100%	100%
T_N	100%	100%	100%	100%	100%
C_O	100%	100%	100%	100%	100%
C_C	95.4%	96.9%	83.3%	91.87%	78.2%
N_C	91.8%	95.7%	89.8%	92.43%	77.4%

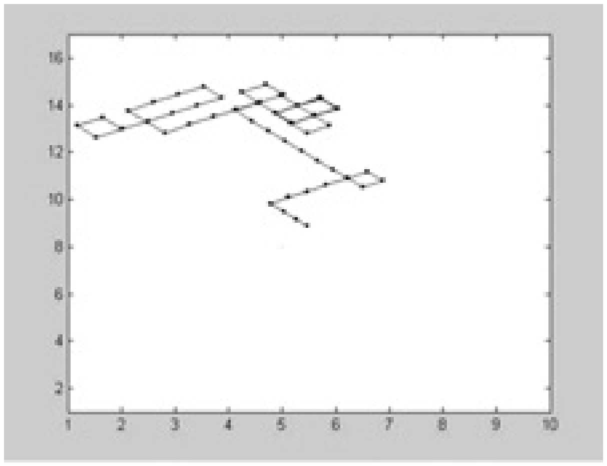


Figure 6 A RunDown homing run on surface C_C. In this run, it took 68 steps to home.

faces where performance is relatively low: they use as the current image set, the "Chairs" dataset. This implies that the true difficulty for homing comes when local optima appear, as a result of objects that are lying near the catchment area and can come too close to the camera. This is a common problem for all homing algorithms, but in the RMS difference approach it takes the form of local minima or maxima in the surface.

Another observation on the success rates is the relatively poor performance of RunDown on the surfaces that contain local optima. One reason for this is the "rigid" behavior of RunDown. Unlike Taxis, where at each step the turning rate is completely different, and thus a local optimum usually simply causes a small divergence from the course, RunDown often gets trapped, moving around a local minimum in square patterns (Figure 6).

Figure 7 shows the mean number of steps taken by each controller to home on each surface. It is apparent that there is no direct link between the surfaces used to train the controllers, and the steps needed to home. This is desirable, since it indicates that the controllers were not optimized for the specific surfaces used for the evolution, but were instead optimized for homing on surfaces which carry the general properties of these surfaces. We can see that RunDown is in most cases worse than Controller #3, slightly worse than the average of the three Taxis controllers, and better than the other two controllers. The exception is surface N_C, where RunDown faces serious problems. Surface N_C has the most extreme local optima, and RunDown's inability to deal with them is apparent.

A similar pattern emerges from Figure 8, which shows the mean distance traveled per homing run. However, in this case, RunDown is slightly worse than all the other controllers. The superiority of the Taxis algorithms in the distance traveled is countered by their higher cost in turning, as shown in Figure 9. This is to be expected, since the Taxis controllers turn at least a little at every time step, while the RunDown algorithm often covers large distances by moving straight ahead. This is also reflected in the higher variances in

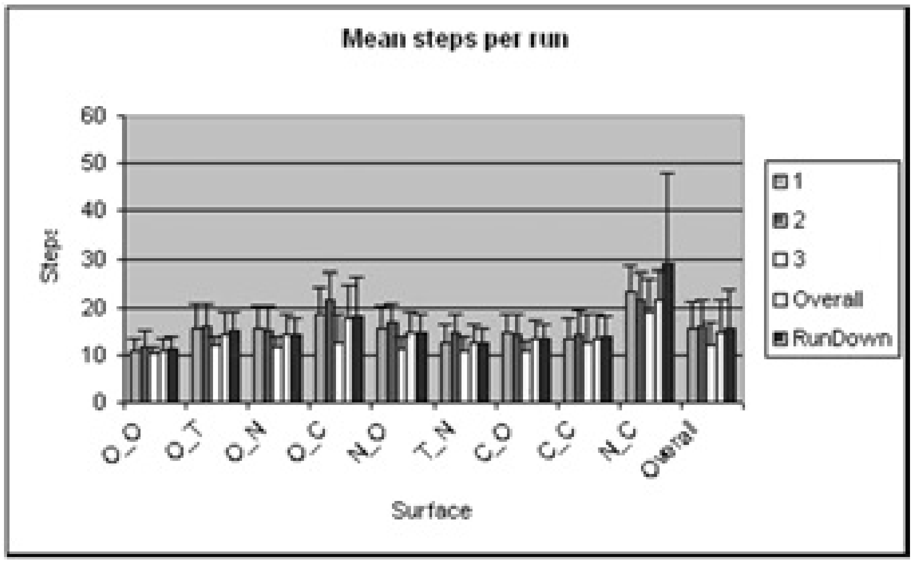


Figure 7 The mean number of steps taken to reach home by the different algorithms on each surface.

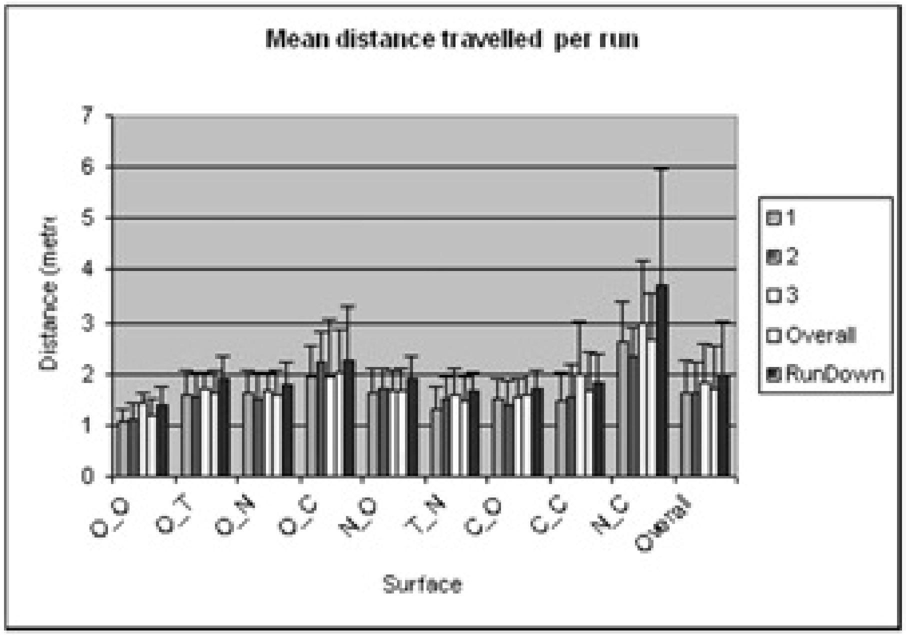


Figure 8 The mean distance traveled to reach home by the different algorithms for each surface.

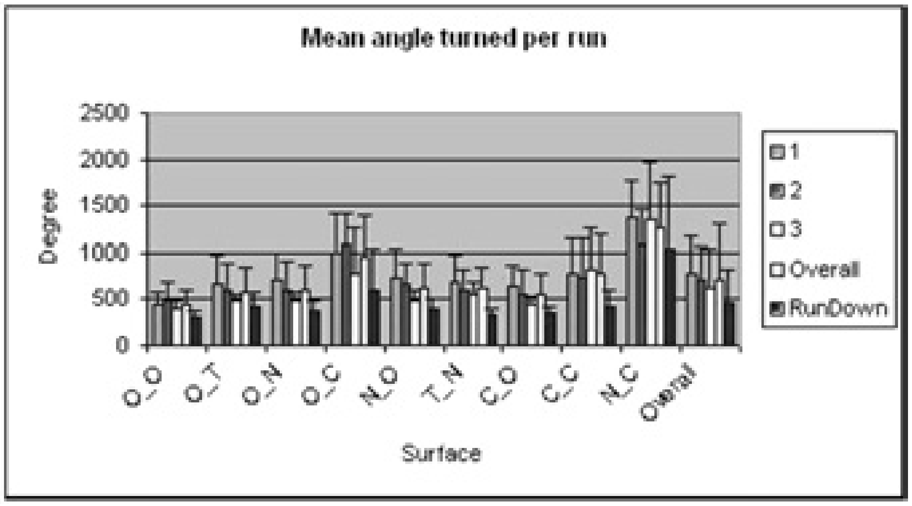


Figure 9 The mean angle turned during homing by the different algorithms for each surface.

the turning rates of the Taxis controllers. The Taxis algorithm is not as predictable in its behavior as Run-Down. On a surface that is not completely symmetrical, the course of the agent depends heavily on the start position. Even noise plays a role in leading to a different course at each run. This unpredictability is not necessarily a disadvantage, however, since it increases the robustness of homing in the presence of local optima.

As for homing precision, it is strictly determined by three parameters: the controller's step length; the RMS values near the home position; and the RMS threshold, which is fixed across all runs. This leads to homing precision being more or less the same in every homing run for the same surface and the same controller, which is reflected in the extremely low variance in the data in Figure 10. However, comparing the overall homing pre-

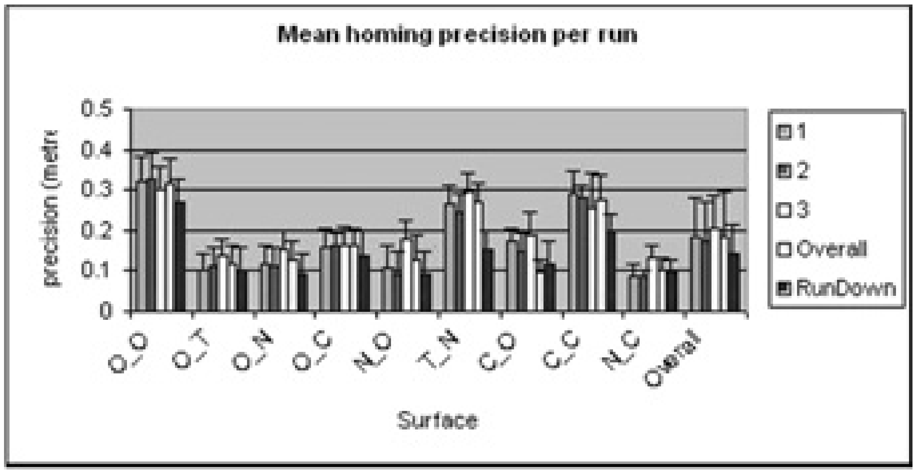


Figure 10 The mean homing accuracy of the different algorithms for each surface.

cision on different surfaces, we see that there are clear differences between them. This results from the termination condition of the Taxis algorithm. Some surfaces had a very low value at the home position (O_O and C_C actually had zero), so the RMS difference goes below the threshold while the agent is still quite far away from the goal. Ideally, the threshold would be adapted to the surface, but in the real world, this will not be possible, since we cannot know in advance the conditions under which the agent might have to home.

In summary, the Taxis controllers are on average as good as RunDown, if not better, for most criteria, excluding the total angle turned. However, in the presence of local optima on the RMS difference surface, the performance of RunDown deteriorates significantly, while Taxis homing is able to maintain a relatively high performance.

3.3 Comparing the Vardy Surfaces and the Robot Lab Surfaces

Our initial attempts to use the evolved controllers from the simulated agent on the robot were very unsuccessful, which led us to study more closely the shape of the RMS difference surface for our specific camera-mirror setup and environment. The RMS surface was estimated by taking single rows of equally distanced (30 cm) images with the Koala robot. For previous surfaces, it was assumed that all images were taken under the same orientation. As for any given position the minimum RMS difference appears when the home image and current orientation match, it is possible to estimate the RMS difference for that orientation by capturing an image at a random orientation, modeling rotation by shifting the unfolded image, and keeping the minimum RMS difference from all possible shifts.

In studying the resulting RMS difference surfaces, a number of observations were made:

(1) The fundamental properties of the RMS surface were present in all cases (i.e., global minimum at the home position, and gradual increase in the RMS difference as the robot moves away from home). The RMS difference increase generally looks as a quadratic function of the distance from home, although the exact properties of the function are only now beginning to be studied (Szenher, 2005).
(2) The RMS surface was clearly not as steep as in the Vardy dataset. Steepness was affected by changes in the illumination levels, decreasing as the illumination decreased, but even in the case of maximum possible lighting, the value range was significantly narrower, and the slope smaller (Figure 11).
(3) The introduction of new objects, or movement of old ones either after taking the home image or throughout the experiment, did very little to disturb the RMS surface.
(4) Moving the robot too close to an object led to a local maximum on the RMS surface. This is because, as the area covered by the object on the image increased, it covered a large part of it and resulted in large differences. As the robot moved on and

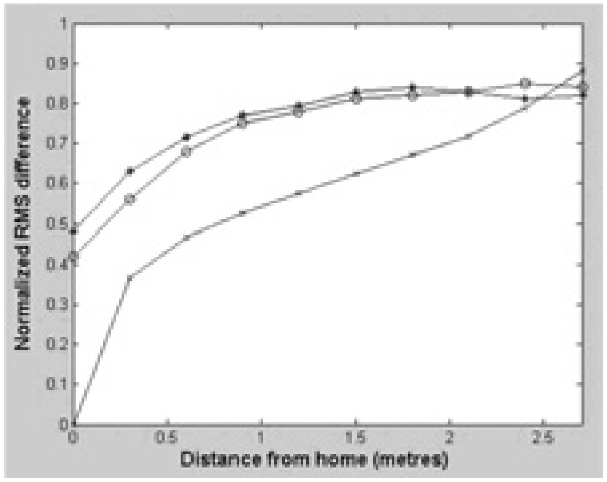


Figure 11 RMS values for different situations: X: The RMS difference for the "Original" Vardy dataset. O: RMS differences taken from the lab area under bright illumination. Stars: RMS differences taken from the same area under low illumination.

away from the object, the landmarks reappeared and led to smaller RMS differences (Figure 12).

(5) The radius of the catchment area reached up to 2.7m, and never came below 1.5m (Figure 11). Again, the radius was affected by the illumination levels, decreasing as the illumination decreased, but also from the presence of nearby objects.

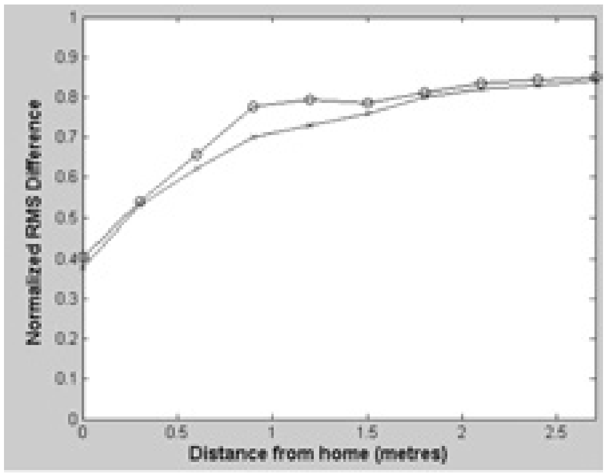


Figure 12 The change in the RMS surface when the agent happens to pass too close to an object, resulting in occlusion of most other features. X: No object. O: A small cardboard box next to the agent's route at 1 m from the home position.

The main reason for the reduced slope appeared to be noise introduced by rotation, although it is also possible that differences in the environment or the image capture method used contributed to the problem. The assumption that rotation can be modeled by shifting the unfolded image by a number of columns requires that the camera is perfectly orthogonal to the panoramic mirror, and the position of the mirror center in the panoramic image (i.e., before unfolding) is known with accuracy. Our camera rig was less accurate in this respect than those used by Zeil et al. and Vardy. Capturing an image from the home position with the same orientation as the original home image gave a minimum normalized RMS difference of 0.01 to 0.1. Rotating the robot by 180 degrees to capture the image, and then shifting the image columns by 180 degrees (which should compensate for the rotation) gave a minimum normalized RMS difference of 0.4 to 0.5 (Figure 13). In general, it can be said that the RMS difference increased as the current orientation moved up to 10–15 degrees away from the home image orientation, and then remained similar for further rotations. Thus, the RMS difference will be overestimated when the current image is not taken at the same orientation as the home image (Figure 14). Moreover, when the orientations *are* aligned, the RMS difference will be notice-

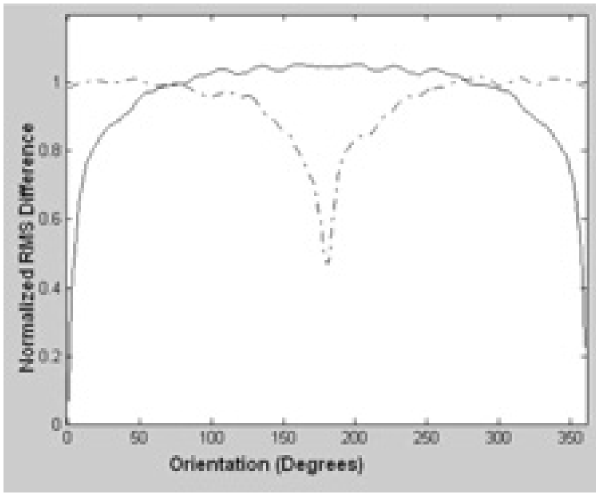


Figure 13 The RMS difference for all theoretical rotations of two images, taken from the home position with different orientations. Continuous line: Current agent orientation matches the home image orientation, producing a minimum of 0.08 at 0º. Interrupted line: Current agent orientation 180º from home image orientation, producing a minimum of 0.48 at 180º.

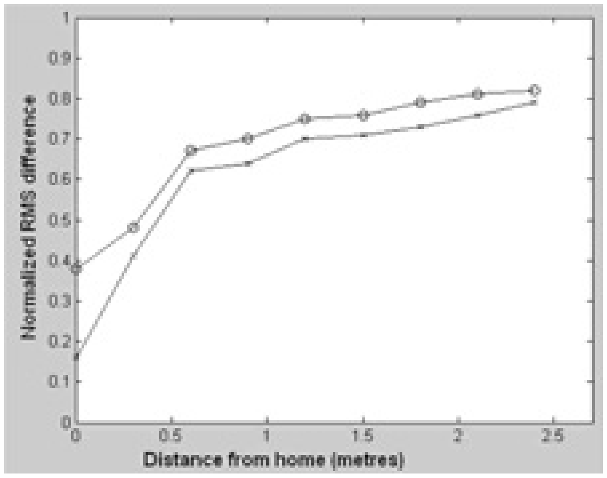


Figure 14 The RMS difference for the same route and home position, with the home image captured at different orientations. X: Matching home image and current orientations. O: 90º angle between home image and current orientations.

ably smaller, which can effectively cause a transient minimum that affects the homing process. This problem could not be effectively modeled by Gaussian noise, hence the controllers evolved for the simulated agent could not deal with it. We therefore had to repeat the GA search for parameters on surfaces estimated from the robot data as described earlier. This produced moderately successful controllers as described below.

3.4 Comparison of Taxis, RunDown and Image Warping in the Physical Robot

The performance of the three algorithms was compared using the same measures as in the simulation. The first tests used a well-lit, unchanging environment, and consisted of two sets of ten runs for each algorithm: Ten starting from different positions 1 meter away from the home position, and another ten runs starting from 2 meters away. Three different home positions were used across the trials.

The first observation is that, unlike the simulation, the Taxis algorithm performs worse than the RunDown algorithm (Tables 4 and 5), particularly for the longer distance where there is a high failure rate

Table 4 The results for 10 short-range homing runs (1 meter away), in a clear, bright area with no movement (numbers in brackets indicate the standard deviation).

Algorithm	Taxis	RunDown	Warping
% success	90%	80%	100%
# of steps	20.67 (45.11)	13.75 (29.43)	6.8 (6.76)
Distance traveled	4.211 (1.5)	2.525 (1.02)	1.776 (0.518)
Angle turned	1316.41 (346180)	360 (72900)	316.04 (105180)
Homing precision	0.4290 (0.08)	0.179 (0.01)	0.2876 (0.02)

Table 5 The results for 10 long-range (2 meters away) homing runs in a clear, bright area with no movement (numbers in brackets indicate the standard deviation).

Algorithm	Taxis	RunDown	Warping
% success	40%	90%	60%
# of steps	22.25 (7.6875)	23.56 (112.025)	16.33 (18.56)
Distance traveled	5.988 (3.87)	4.489 (4.9)	3.814 (1.148)
Angle turned	1559.3 (150550)	580 (218000)	999.30 (246560)
Homing precision	0.4180 (0.046)	0.252 (0.016)	0.2647 (0.02)

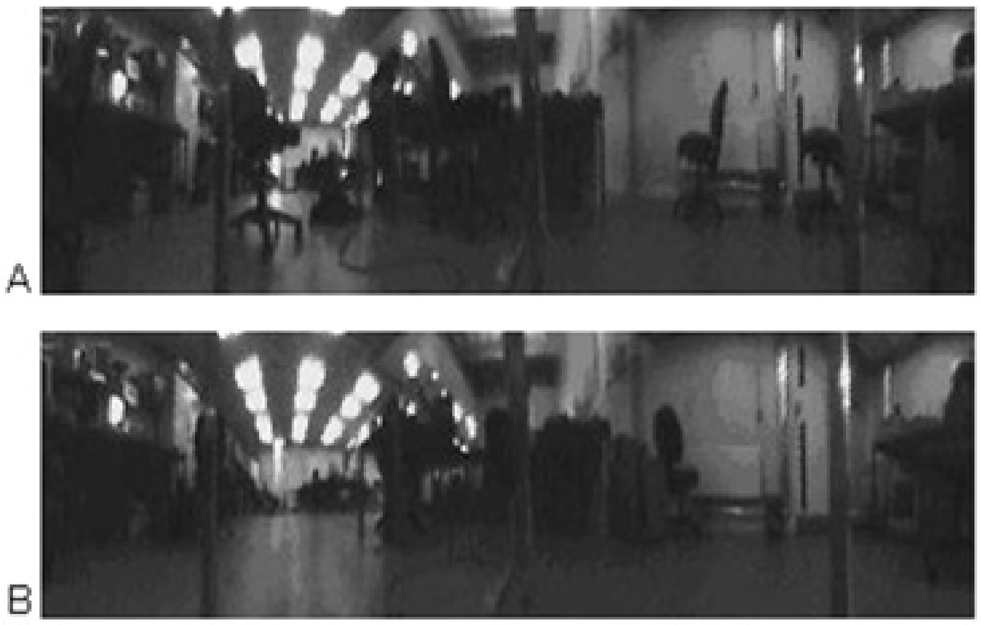


Figure 15 The moved landmarks experiment. A: The unfolded home image. B: An unfolded image taken from the home position after the changes in the scene.

(60%), but also at the shorter distance where the number of steps, distance traveled, angle turned and homing precision are all worse for the Taxis algorithm. The poor performance of the Taxis is likely to be due to noise as described above. The behavior of an agent during RunDown homing only depends on the sign of the gradient, while Taxis homing also depends on the actual RMS difference values. This means that it is significantly more sensitive to noise than RunDown. It should be pointed out, however, that the Vardy images used in the simulation were also captured in the real world, but the quality of the equipment reduced the noise to a level that did not disturb Taxis homing. It therefore seems plausible that, with better equipment (where the image is not distorted in rotation), the noise could be reduced enough for the Taxis algorithm to home more effectively than seen here.

The real-world implementation of the image warping algorithm, on the other hand, performed better than both the real and the simulated versions of both the Taxis algorithm and RunDown over shorter distances. This means that the Taxis algorithm is in fact not in a position to compete with image warping on these measures, even if the real-world conditions were ideal. However, note that this result is for a static environment. Our next test examined homing by each algorithm when gradual changes were made to the environment with respect to the stored home image.

Obviously, sufficient change in the environment will render the home image position unrecognizable, and homing will fail for any algorithm. On the other hand, all algorithms demonstrate a level of tolerance to minor changes in the environment. The question, then, is which algorithm is the first to break down if we gradually increase the number of changes in the scene? In the example of the changes shown in Figure 15, it was observed that the image warping algorithm was unable to home. In fact, its perceived home position had been relocated from position H in Figure 16 where the robot lies, to the “trap” position marked with T. The movement of the objects had changed the horizon line enough for image warping to find the best match in that position, instead of the home position. When the trap point was not visible from the start position, but the actual home position was, the agent would initially move to the true home position, bringing the trap point within homing range, and the agent would then go on and home on the trap. Thus, the image warping algorithm was rendered completely useless.

As for the RMS-based algorithms, the most significant problem was the appearance of a local minimum at the position marked with M in Figure 16. This

Table 6 The results for 10 homing runs after changing the scene.

Algorithm	Taxis	RunDown
% success	80%	90%
# of steps	22.75 (57.68)	13.33 (4.44)
Distance traveled	5.077 (3.676)	1.639 (0.099)
Angle turned	1670.38 (592560)	209.99 (7200)
Homing precision	0.220 (0.013)	0.188 (0.006)

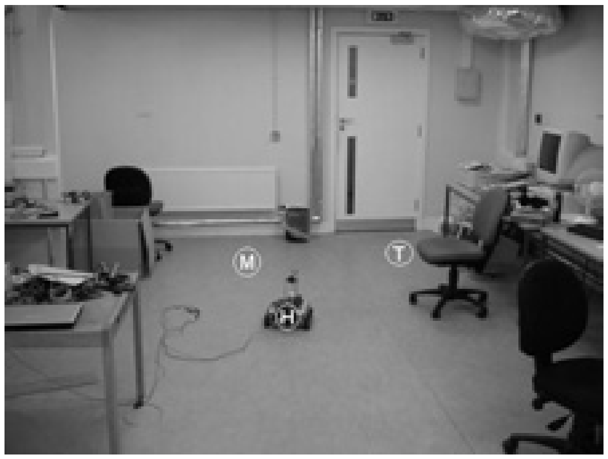


Figure 16 The moved chairs scene. H: The home position. M: The local minimum. T: The perceived home position for image warping.

in some way resembles the way in which image warping was misguided. There is a fundamental difference though, in that the local minimum was not as deep as the global minimum, which remained on the home position. This is because the RMS surface is calculated using visual information not only from the horizon line, but instead from the entire scene, including floor and ceiling patterns. This means that, even when the landmarks happen to move in such a way as to misguide image warping, the RMS surface needs much more radical changes in the scene to lose the global minimum.

The results for the homing runs in this environment are given in Table 6. It can be seen that no significant changes in the behavior of the algorithms appeared, suggesting that, if the agent does not get trapped in the local minimum, it remains completely unaffected by the changes in the environment. The RunDown algorithm still outperformed the Taxis algorithm.

4 Discussion

A new algorithm for image-based homing was proposed. It combines the properties of the RMS pixel difference between the home image and any image in the catchment area (Zeil et al., 2003), and a computational model for *C. elegans* chemotaxis (Ferree & Lockery, 1998). Parameters for this algorithm were optimized using an evolutionary strategy for a simulated agent homing in an environment built from image data gathered in the real world. For the simulated agent, the evolved Taxis controllers were more efficient than the RunDown algorithm proposed by Zeil et al. and performed significantly better in situations where objects close to the agent caused local minima in the RMS difference surface. However, in homing by the real robot the Taxis algorithm suffered from noise, and performed worse than the RunDown algorithm. When there was no change in the scene between capture of the home image and the homing run, both RMS-based algorithms performed worse than image warping over short distances, although the success of image warping was reduced at larger distances. However, in the more realistic situation of some changes occurring, the image warping algorithm was much less robust, becoming entirely mistaken about the home location. The RMS-based algorithms were relatively unaffected.

The main difference between the RMS-based controllers was observed when the starting distance was larger. It would be reasonable to assume that, for a Taxis controller adapted for a particular surface, the catchment area extends as long as the RMS difference keeps increasing. In the experiments in a bright, static environment, the catchment area often extended as far as 2.7 meters. It is clear that both image warping and the Taxis algorithm have problems homing beyond 2 meters, while RunDown seems relatively reliable at those distances. As this problem was not observed for the simulated agent, it suggests that the first two algorithms are more affected by the real world noise introduced by less-than-perfect calibration of the camera–mirror position. On the other hand, as the simulated agent was using real-world data, it appears reasonable to assume that with a better camera–mirror rig, this noise level could be reduced to a degree that would make both algorithms competitive with RunDown over these longer distances, and possibly more efficient.

We can also compare the computational cost of the algorithms, which is due to the image comparisons required. To estimate the RMS difference at each step, there is a comparison under all 360 possible orientations, and each comparison takes $360 \times H \times 3$ calculations, where H the image height, 360 the image width and the three calculations are the subtraction, square and addition. Furthermore, to shift an image by N columns, it takes $2 \times N + 360 - N = 360 + N$ operations. Thus for all orientations (0 to 359), a total of $360 + 362 + \ldots + 719 = 149{,}220$ operations are required. In our implementation, the image height H was 70, making for 27,216,000 operations for all comparisons, and 27,410,220 operations in total.

Image warping compares $n_\alpha \times n_\rho \times n_\psi$ images, where n_α, n_ρ, and n_ψ are the number of increments for α, ρ, and ψ respectively, and compares them using 360×2 calculations for the normalization and 360×2 calculations for the comparison (multiplication and summation). In order to build each image, image warping transforms the current image using Equation (4) for each pixel. The total number of operations necessary, including the auxiliary ones (modulus by 360, adding to the current pixel position, since δ only signifies the pixel shift) is 14. In our implementation, it is $n_\alpha = 36$, $n_\rho = 36$, and $n_\psi = 10$ for the first recursion, and $n_\alpha = 3$, $n_\rho = 3$, and $n_\psi = 10$ for the second recursion. This adds up to a total of 84,564,000 operations, which means that image warping requires 3.01 times more operations to come up with a home vector.

Note, however, that our implementation of image warping used a recursive implementation to search for the best values, starting with a search increment of 10 degrees for α and ψ, and reducing it at each recursion. Similarly, the increment for ρ was initially 0.2, and was reduced at each subsequent recursion. The current RMS implementation performs a full search, since the behavior of the RMS values with rotation was not known beforehand. It should be straightforward to implement RMS with a recursive search, with an initial step of 10 degrees, as used for image warping. This should reduce the number of calculations for RMS algorithms by an order of magnitude, increasing the advantage over Image Warping to a factor of about 30 times faster.

There are other function minimization techniques described in the mathematics literature that—like RunDown and Taxis—minimize a function without explicit calculation of the gradient. Downhill simplex and Powell's method are two popular examples (Press, Flannery, Teukolsky, & Vetterling, 1992). Downhill simplex evaluates the function at the points of a triangle (the "simplex" of the title). The initial triangle is determined by the robot's initial pose. The algorithm then usually reflects the vertex with largest function value about the line running through the other two vertices, forming a new triangle. In this way, downhill simplex marches towards the nearest function minimum. The simplex can adapt itself to functional values with long, narrow valleys, which typically cause problems with other minimization schemes. Powell's method, like RunDown, minimizes the function along two alternating directions. Unlike RunDown, though, Powell's method adapts these directions during optimization to take advantage of the local topography of the function. We may in future compare the efficacy of RunDown and Taxis with that of downhill simplex and Powell's method to solve the problem at hand.

In conclusion, it was demonstrated that the proposed Taxis algorithm is a more efficient homing algorithm than RunDown, but less robust to noise introduced by rotation of our camera–mirror rig on the real robot. The reduced computational complexity of the RMS algorithms and tolerance to changes in the scene make them possible competitors to image warping, which is widely recognized as the most successful visual homing algorithm to date, and the one most commonly used for comparison (Vardy & Möller, 2005). It also seems more plausible that an RMS-based algorithm could be neurally implemented, but as yet any possible mechanisms for storage and comparison of images in insect brains are a matter of speculation.

Acknowledgments

This work was partly supported by the European Commission under project FP6-2003-IST2-004690 SPARK.

References

Cartwright, B. A., & Collett, T. S. (1983). Landmark learning in bees. *Journal of Computational Physiology A*, *151*, 521–543.

Ferree, T., & Lockery, S. (1999). Computational rules for chemotaxis in the nematode *C. elegans. Journal of Computational Neuroscience*, *6*, 263–277.

Franz, M. O., & Mallot, H. A. (2000). Biomimetic robot navigation. *Robotics and Autonomous Systems*, *30*, 133–153.

Franz, M. O., Schölkopf, B., Mallot, H. A., & Bülthoff, H. H. (1998). Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, *79*, 191–202.

Hong, J., Tan, X., Pinnette, B., Weiss, R., & Riseman, E. (1992). Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation* (pp. 620–625). Sacramento, CA.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior, 6*, 325–368.

Labrosse, F. (2004). Visual compass. In U. Nehmzow & C. Melhuish (Eds.), *Proceedings of Towards Autonomous Robotic Systems 2004* (pp. 85–92). Colchester, UK: Springer-Verlag.

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

Mitchell, M. (1998). *An Introduction to Genetic Algorithms* (pp. 155–179). Cambridge, MA: MIT Press.

Möller, R. (2000). Insect visual homing strategies in a robot with analog processing. *Biological Cybernetics*, *83*, 231–243.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1992). *Numerical recipes in C* (pp. 394–444). Cambridge, UK: Cambridge University Press.

Szenher, M. (2005). Visual homing in natural environments. In U. Nehmzow, C. Melhuish, & M. Witkowski (Eds.), *Proceedings of Towards Autonomous Robotic Systems 2005* (pp. 221–226). London, UK.

Vardy, A., & Möller, R. (2005). Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, *17*, 47–89.

Weber, K., Venkatesh, S., & Srinivasan, M. V. (1998). An insect-based approach to robotic homing. In *Proceedings of the 14th International Conference on Pattern Recognition* (pp. 297–299). Brisbane, Australia: The IEEE Press.

Weisstein, E. (2006). *Method of steepest descent*. From Mathworld—A Wolfram Web Resource: http://mathworld.wolfram.com/MethodofSteepestDescent.html

Zeil, J., Hoffmann, M., & Chahl, J. S. (2003). Catchment areas of panoramic home images in outdoor scenes. *Journal of the Optical Society of America A*, *20*, 450–469.

About the Authors

Markos Zampoglou is currently a Ph.D. student in the Department of Applied Informatics, University of Macedonia, Thessaloniki. He received his B.Sc. in applied informatics in the University of Macedonia in 2004 and a M.Sc. in artificial intelligence in the University of Edinburgh in 2005. His current research involves machine vision and content-based video retrieval. *Address:* Department of Applied Informatics, University of Macedonia, Egnatia 156, 54006 Thessaloniki, Greece. *E-mail*: markzampoglou@yahoo.gr

Matthew Szenher is currently a Ph.D. student in the Institute of Perception, Action and Behaviour at the School of Informatics, University of Edinburgh. He received his Sc.B. in computer science from Brown University in 1995. His current research involves robotic homing in dynamic environments. *Address*: Institute of Perception Action and Behaviour (IPAB), School of Informatics, University of Edinburgh, Room 1.1106, James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ, UK.
E-mail: M.Szenher@sms.ed.ac.uk.

Barbara Webb received a B.Sc. in psychology from the University of Sydney in 1987 and a Ph.D. in artificial intelligence from the University of Edinburgh in 1993. She has held lectureships at the University of Nottingham and the University of Stirling, and is currently a Reader in Informatics at the University of Edinburgh. Her main research interest is robot modeling of insect sensorimotor systems.

Navigation by Image-based Visual Homing

Matthew Szenher, M.Szenher@sms.ed.ac.uk
Institute of Perception Action and Behaviour (IPAB)
School of Informatics, University of Edinburgh, UK

1. INTRODUCTION

Almost all autonomous robots need to navigate. We define navigation as do Franz & Mallot (2000): "Navigation is the process of determining and maintaining a course or trajectory to a goal location" (p. 134). We allow that this definition may be more restrictive than some readers are used to - it does not for example include problems like obstacle avoidance and position tracking - but it suits our purposes here.

Most algorithms published in the robotics literature localise in order to navigate (see e.g. Leonard & Durrant-Whyte (1991a)). That is, they determine their own location and the position of the goal in some suitable coordinate system. This approach is problematic for several reasons. Localisation requires a map of available landmarks (i.e. a list of landmark locations in some suitable coordinate system) and a description of those landmarks. In early work, the human operator provided the robot with a map of its environment. Researchers have recently, though, developed simultaneous localisation and mapping (SLAM) algorithms which allow robots to learn environmental maps while navigating (Leonard & Durrant-Whyte (1991b)). Of course, autonomous SLAM algorithms must choose which landmarks to map and sense these landmarks from a variety of different positions and orientations. Given a map, the robot has to associate sensed landmarks with those on the map. This data association problem is difficult in cluttered real-world environments and is an area of active research.

We describe in this chapter an alternative approach to navigation called visual homing which makes no explicit attempt to localise and thus requires no landmark map. There are broadly two types of visual homing algorithms: feature-based and image-based. The feature-based algorithms, as the name implies, attempt to extract the same features from multiple images and use the change in corresponding features to navigate. Feature correspondence is - like data association - a difficult, open problem in real-world environments. We argue that image-based homing algorithms, which provide navigation information based on whole-image comparisons, are more suitable for real-world environments in contemporary robotics.

2. BACKGROUND

Visual homing algorithms make no attempt to localise in order to navigate. No map is therefore required. Instead, an image I_S (usually called a snapshot for historical reasons) is captured at a goal location $S = (x_S , y_S)$. Note that though S is defined as a point on a plane, most homing algorithms can be easily extended to three dimensions (see e.g. Zeil et al. (2003)) . When a homing robot seeks to return to S from a nearby position $C = (x_C , y_C)$, it takes an image I_C and compares it with I_S. The home vector $\boldsymbol{H} = C - S$ is inferred from the disparity between I_S and I_C (vectors are in upper case and bold in this work). The robot's orientation at C and S is often different; if this is the case, image disparity is meaningful only if I_C is rotated to account for this difference. Visual homing algorithms differ in how this disparity is computed.

Visual homing is an iterative process. The home vector $\boldsymbol{H}$ is frequently inaccurate, leading the robot closer to the goal position but not directly to it. If $\boldsymbol{H}$ does not take the robot to the goal, another image I_C is taken at the robot's new position and the process is repeated.

The images I_S and I_C are typically panoramic grayscale images. Panoramic images are useful because, for a given location *(x,y)* they contain the same image information regardless of the robot's orientation. Most researchers use a camera imaging a hemispheric, conical or paraboloid mirror to create these images (see e.g. Nayar (1997)).

Some visual homing algorithms extract features from I_S and I_C and use these to compute image disparity. Alternatively, disparity can be computed from entire images, essentially treating each pixel as a viable feature. Both feature-based and image-based visual homing algorithms are discussed below.

3. FEATURE-BASED VISUAL HOMING

Feature-based visual homing methods segment I_S and I_C into features and background (the feature extraction problem). Each identified feature in the snapshot is then usually paired with a feature in I_C (the correspondence problem). The home vector is inferred from - depending on the algorithm - the change in the bearing and/or apparent size of the paired features. In order for feature-based homing algorithms to work properly, they must reliably solve the feature extraction and correspondence problems.

The Snapshot Model (Cartwright & Collett (1983)) - the first visual homing algorithm to appear in the literature and the source of the term "snapshot" to describe the goal image - matches each snapshot feature with the current feature closest in bearing (after both images are rotated to the same external compass orientation). Features in (Cartwright & Collett (1983)) were black cylinders in an otherwise empty environment. Two unit vectors, one radial and the other tangential, are associated with each feature pair. The radial vector is parallel to the bearing of the snapshot feature; the tangential vector is perpendicular to the radial vector. The direction of the radial vector is chosen to move the agent so as to reduce the discrepancy in apparent size between paired features. The direction of the tangential vector is chosen to move the agent so as to reduce the discrepancy in bearing between paired features. The radial and tangential vectors for all feature pairs are averaged to produce a homing vector. The Snapshot Model was devised to explain the behaviour of nest-seeking honeybees but has inspired several robotic visual homing algorithms.

One such algorithm is the Average Landmark Vector (ALV) Model (Möller et al. (2001)). The ALV Model, like the Snapshot Model, extracts features from both I_C and I_S. The ALV Model, though, does not explicitly solve the correspondence problem. Instead, given features extracted from I_S , the algorithm computes and stores a unit vector $\boldsymbol{ALV_S}$ in the direction of the mean bearing to all features as seen from *S*. At *C*, the algorithm extracts features from I_C and computes their mean bearing, encoded in the unit vector $\boldsymbol{ALV_C}$. The home vector **H** is defined as $\boldsymbol{ALV_C} - \boldsymbol{ALV_S}$. Figure 1 illustrates home vector computation for a simple environment with four easily discernible landmarks.

Several other interesting feature-based homing algorithms can be found in the literature. Unfortunately, space constraints prevent us from reviewing them here. Two algorithms of note are: visual homing by "surfing the epipoles" (Basri et al. (1998) and the Proportional Vector Model (Lambrinos et al. (2000)).

The Snapshot and ALV Models were tested by their creators in environments in which features contrasted highly with background and so were easy to extract. How is feature extraction and correspondence solved in real-world cluttered environments? One method is described in Gourichon et al. (2002). The authors use images converted to the HSV (Hue-Saturation-Value) colour space which is reported to be more resilient to illumination change than RGB. Features are defined as image regions of approximately equal colour (identified using a computationally expensive region-growing technique). Potential feature pairs are scored on their difference in average hue, average saturation, average intensity and bearing. The algorithm searches for a set of pairings which maximise the sum of individual match scores. The pairing scheme requires *$O(n^2)$* pair-score computations. The algorithm is sometimes fooled by features with similar colours (specifically, pairing a blue chair in the snapshot image with a blue door in the current image). Gourichon et al. did not explore environments with changing lighting conditions.

Several other methods feature extraction and correspondence algorithms appear in the literature; see e.g. Rizzi et al. (2001), Lehrer & Bianco (2000) and Gaussier et al. (2000). Many of these suffer from some of the same problems as the algorithm of Gourichon et al. described above. The appearance of several competing feature extraction and correspondence algorithms in recent publications indicates that these are open and difficult problems; this is why we are advocating image-based homing in this chapter.

4. IMAGE-BASED VISUAL HOMING

Feature-based visual homing algorithms require consistent feature extraction and correspondence over a variety of viewing positions. Both of these are still open problems in computer vision. Existing solutions are often computationally intensive. Image-based visual homing algorithms avoid these problems altogether. They infer image disparity from entire images; no pixel is disregarded. We believe that these algorithms present a more viable option for real-world, real-time robotics.

Three image-based visual homing algorithms have been published so far; we describe these below.

4.1 Image Warping

The image warping algorithm (Franz et al. (1998)) asks the following question: When the robot is at *C* in some unknown orientation, what change in orientation and position is required to transform I_C into I_S? The robot needs to know the distance to all imaged objects in I_S to answer this question precisely. Not having this information, the image warping algorithm makes the assumption that all objects are at an equal (though unknown) distance from *S*. The algorithm searches for the values of position and orientation change which minimises the mean-square error between a transformed I_C and I_S . Since the mean square error function is rife with local minima, the authors resort to a brute force search over all permissible values of position and orientation change.

Unlikely as the equal distance assumption is, the algorithm frequently results in quite accurate values for $\boldsymbol{H}$. Unlike most visual homing schemes, image warping requires no external compass reference. Unfortunately, the brute force search for the homing vector and the large number of transformations of I_C carried out during this search make image warping quite computationally expensive.

4.2 Homing with Optic Flow Techniques

When an imaging system moves from *S* to *C* , the image of a particular point in space moves from $I_S(x,y)$ to $I_C(x',y')$. This movement is called optic flow and $(x-x', y-y')$ is the so called pixel displacement vector. Vardy & Möller (2005) demonstrate that the home vector $\boldsymbol{H}$ can be inferred from a single displacement vector so long as the navigating robot is constrained to move on a single plane. So too, several noisy displacement vectors can be combined to estimate $\boldsymbol{H}$.

Vardy & Möller (2005) describe a number of methods, adapted from the optic flow literature, to estimate the displacement vector. One of the most successful methods – BlockMatch - segments the snapshot image into several equal-sized subimages. The algorithm then does a brute force search of a subset of I_C to find the best match for each subimage. A displacement vector is computed from the centre of each subimage to the centre of its match pair in I_C.

A less computationally intensive algorithm estimates the displacement vector from the intensity gradient at each pixel in I_C . The intensity gradient at a particular pixel can be computed straightforwardly from intensities surrounding that pixel. No brute-force search is required.

In comparative tests, Vardy & Möller demonstrated that their optic flow based methods perform consistently better than image warping in several unadulterated indoor environments. A drawback to the optic flow homing methods is that the robot is constrained to move on a single plane. The authors do not provide a way to extend their algorithm to three dimensional visual homing.

4.3 Surfing the Difference Surface

Zeil et al. (2003) describe a property of natural scenes which can be exploited for visual homing: as the Euclidean distance between *S* and *C* increases, the pixel-by-pixel root mean square (RMS) difference between I_S and I_C increases smoothly and monotonically. Labrosse and Mitchell discovered this phenomenon as well; see Mitchell & Labrosse (2004). Zeil et al. reported that the increase in the RMS signal was discernible from noise up to about three meters from *S* in their outdoor test environment; they call this region the catchment area.

RMS, when evaluated at locations in a subset of the plane surrounding *S*, forms a mathematical surface, the difference surface. A sample difference surface is shown in Figure 2(a) (see caption for details).

Zeil et al. describe a simple algorithm to home using the RMS difference surface. Their "Run-Down" algorithm directs the robot to move in its current direction while periodically sampling the RMS signal. When the current sample is greater than the previous, the robot is made to stop and turn ninety degrees (clockwise or counter-clockwise, it does not matter). It then repeats the process in this new direction. The agent stops when the RMS signal falls below a pre-determined threshold. We have explored a biologically inspired difference surface homing method which was more successful than "Run-Down" in certain situations (Zampoglou et al. (2006)).

Unlike the optic flow methods described in Section 4.2, visual homing by optimising the difference surface is easily extensible to three dimensions (Zeil et al. (2003)).

Unfortunately, when lighting conditions change between capture of I_S and I_C , the minimum of the RMS difference surface often fails to coincide with *S*, making homing impossible (Figure 2(b)).

5. FUTURE TRENDS

No work has yet been published comparing the efficacy of the image-based homing algorithms described in Sections 4.2 and 4.3. This would seem the logical next step for image-based homing researchers. As we mentioned in Section 4.3, the difference surface is disrupted by changes in lighting between captures of I_S and I_C. This problem obviously demands a solution and is a focus of our current research. Finally, it would be interesting to compare standard map-based navigation algorithms with the image-based visual homing methods presented here.

6. CONCLUSION

Visual homing algorithms - unlike most of the navigation algorithms found in the robotics literature - do not require a detailed map of their environment. This is because they make no attempt to explicitly infer their location with respect to the goal. These algorithms instead infer the home vector from the discrepancy between a stored snapshot image taken at the goal position and an image captured at their current location.

We reviewed two types of visual homing algorithms: feature-based and image-based. We argued that image-based algorithms are preferable because they make no attempt to solve the tough problems of consistent feature extraction and correspondence - solutions to which feature-based algorithms demand. Of the three image-based algorithms reviewed, image warping is probably not practicable due to the computationally demanding brute force search required. Work is required to determine which of the two remaining image-based algorithms is more effective for robot homing in real-world environments.

REFERENCES

Basri, R., Rivlin, E., & Shimshoni, I. (1998). Visual homing: Surfing on the epipoles. In *The Proceedings of the Sixth International Conference on Computer Vision* (p. 863-869).

Cartwright, B., & Collett, T. (1983). Landmark learning in bees. *Journal of Comparative Physiology*, 151, 521-543.

Franz, M., & Mallot, H. (2000). Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30, 133-153.

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

Gaussier, P., Joulain, C., Banquet, J., Leprêtre, S., & Revel, A. (2000). The visual homing problem: an example of robotics/biology cross fertilization. R*obotics and Autonomous Systems*, 30, 155-180.

Gourichon, S., Meyer, J., & Pirim, P. (2002). Using colored snapshots for short-range guidance in mobile robots. *International Journal of Robotics and Automation: Special Issue on Biologically Inspired Robotics*, 17 (4), 154-162.

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

Lehrer, M., & Bianco, G. (2000). The turn-back-and-look behaviour: bee versus robot. *Biological Cybernetics*, 83, 211-229.

Leonard, J. J., & Durrant-Whyte, H. F. (1991a). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7 (3), 376-382.

Leonard, J. J., & Durrant-Whyte, H. F. (1991b). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems* (pp. 1442-1447). Osaka, Japan.

Möller, R., Lambrinos, D., Roggendorf, T., Pfeifer, R., & Wehner, R. (2001). Insect strategies of visual homing in mobile robots. In B. Webb & T. R. Consi (Eds.), *Biorobotics: Methods and Applications* (pp. 37-66). The MIT Press, Cambridge, Massachusetts.

Mitchell, T., & Labrosse, F. (2004). Visual homing: a purely appearance-based approach. In *Proceedings of TAROS (Towards Autonomous Robotic Systems)*. The University of Essex, UK.

Nayar, S. K. (1997). Omnidirectional video camera. In *Proceedings of DARPA image understanding workshop*. New Orleans, USA.

Rizzi, A., Duina, D., & Cassinis, R. (2001). A novel visual landmark matching for a biologically inspired homing. *Pattern Recognition Letters*, 22, 1371-1378.

Vardy, A., & Möller, R. (2005). Biologically plausible visual homing methods based on optical flow techniques. *Connection Science, Special Issue: Navigation*, 17 (1-2), 47-89.

Zampoglou, M., Szenher, M., & Webb, B. (2006). Adaptation of controllers for image-based homing. *Adaptive Behavior*, 14 (4), 381-399.

Zeil, J., Hofmann, M., & Chahl, J. (2003). Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20 (3), 450-469.

TERMS AND DEFINITIONS

Catchment Area: The area from which a goal location is reachable using a particular navigation algorithm.

Correspondence Problem: The problem of pairing an imaged feature extracted from one image with the same imaged feature extracted from a second image. The images may have been taken from different locations, changing the appearance of the features.

Image-based Visual Homing: Visual homing (see definition below) in which the home vector is estimated from the whole-image disparity between snapshot and current images. No feature extraction or correspondence is required.

Feature Extraction Problem: The problem of extracting the same imaged features from two images taken from (potentially) different locations.

Navigation: The process of determining and maintaining a course or trajectory to a goal location.

Optic Flow: The perceived movement of objects due to viewer translation and/or rotation.

Snapshot Image: In the visual homing literature, this is the image captured at the goal location.

Visual Homing: A method of navigating in which the relative location of the goal is inferred by comparing an image taken at the goal with the current image. No landmark map is required.

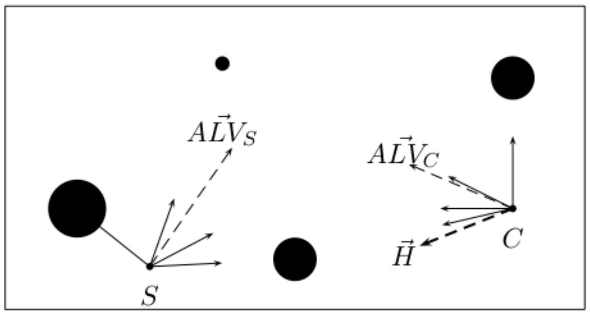
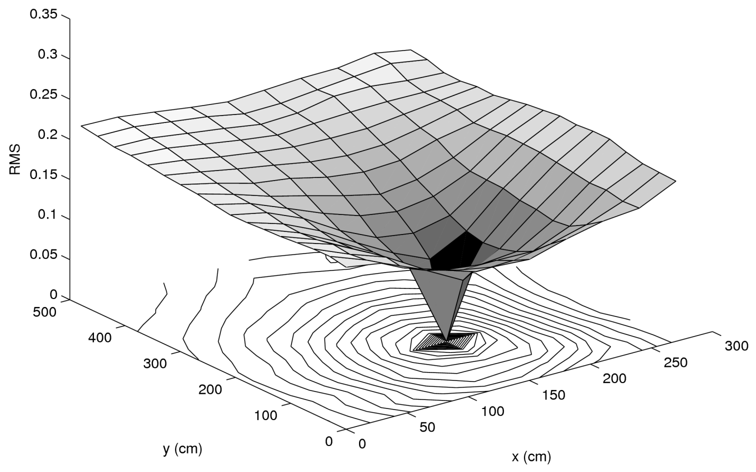
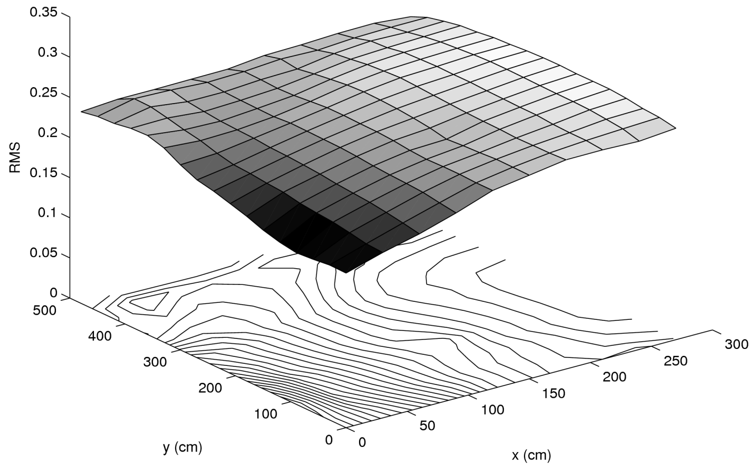


Figure 1. Illustration of Average Landmark Vector computation. See Section 3 for details.



(a)



(b)

Figure 2. Two difference surfaces formed using the RMS image similarity measure. Both the surfaces and their contours are shown. In each case, the snapshot I_S was captured at x=150cm, y=150cm in a laboratory environment. (a) The snapshot was captured in the same illumination conditions as all other images. Notice the global minimum at the goal location and the absence of local minima. (b) Here we use the same snapshot image as in (a) but the lighting source has changed in all other images. The global minimum no longer appears at the goal location. When different goal locations were used, we observed qualitatively similar disturbances in the difference surfaces formed. The images used were taken from a database provided by Andrew Vardy which is described in Vardy & Möller (2005).