Solving Ambiguity in Global Localization of Autonomous Robots

AUTHOR
Alex Goldhoorn

INTERNAL SUPERVISOR
dr. Bart de Boer

EXTERNAL SUPERVISOR
dr. Ramon López de Màntaras
(IIIA-CSIC, Bellaterra, Catalunya, Spain)

Master's Thesis
2007 - 2008

Summary

Navigation in robotics is a widely discussed problem that has been intensively investigated lately. Localization is a part of the navigation problem, this however requires some knowledge about the environment, for example a map. This work addresses the problem of multiple hypotheses of the robot location on a topological map, and presents a technique that can be used to disambiguate the position.

The solution proposed to this ambiguity problem is to first try to go to the most likely hypothesis location. There the localization procedure can be applied again with higher confidence to disambiguate the hypotheses.

In this work panorama images are used, which are 360° images of the environment. The images are made by rotating a normal camera and after that stitching the images together. Also experiments have been done with images made by a camera which points to a spherical mirror. An advantage of the later is that the whole image is acquired in one time, however the resolution is lower.

Images can be compared directly, using for example cross-correlation. However, these type of measures are affected by many problems such as illumination changes or viewpoint variation, which can completely adulterate the result. In order to mitigate this nuisances, visual invariant features have been used. In this work I have used the DoG (Difference of Gaussian) and the MSER (Maximally Stable Extremal Regions) detectors to locate interest regions in the panorama images.

The technique used to go to the most likely position (hypothesis) is homing. Homing consists in returning to a previously visited location and several methods have been proposed to this end. In this work ALV homing is used which is based on findings from experiments with an ant species. The ALV (Average Landmark Vector) is the average location of the landmarks, in this case features, as seen from the robot. The direction pointing to home can be calculated by subtracting the ALV at the home location from the ALV at the current location.

The first experiments in simulation were promising and therefore experiments with real world images were done. Images were taken in three different rooms. The direction of the home vector calculated by the ALV homing method was compared to the real home direction. The use of the MSER points was found to be significantly better than the use of DoG points.

The general and most important conclusion is that the ALV homing method works well and is robust to noise and occlusions. It therefore can be used to go to the most likely location to verify the hypothesis of the current (global) position.

Acknowledgements

I would like to thank Ramon López de Màntaras for his suppervision, his comments have been very useful for my project. I would like to thank Ricardo Toledo of the CVC for his ideas and comments. And I am thankful to Bart de Boer from the University of Groningen for his supervision.

Arnau Ramisa has helped me greatly during my project by giving ideas, comments, and starting points but we also have had good times after work. I would like to thank David Aldavert for his ideas, comments and code for the simulation. And I am glad that I could use the code of Enric Celaya to detect the landmarks in the robot laboratory. In general I would like to thank all people of the IIIA for the great time I have had there.

Finally I would like to thank my family and friends for their support, especially Sergi Consol who has been a great help during my stay in Barcelona.

Contents

Chapter 1

Introduction

The importance for men to navigate started when they wanted to go over seas. This becomes clear from the meaning of the verb *to navigate*. The definition of the verb according to *The Oxford Dictionary of English*[1] is: "plan and direct the course of a ship, aircraft, or other form of transport, especially by using instruments or maps" and "travel on a desired course after planning a route" and for animals: "find its way". It origins from the late 16th century from the word *navigare*, which is *navis*, ship, and *agere*, drive.

In robotic navigation the emphasis is on finding its way to a goal, like in the definition for animals. Insects are a good example for the essence of navigation, because they have a rather limited brain capacity due to their size. The research of the different navigation techniques of the ant species *Cataglyphis* by Wehner (1987) has been a basis for a robotic navigation research by Lambrinos et al. (1998, 2000).

Their work is used here to handle ambiguity problems in navigation. These problems occur when the navigation method has several likely hypotheses about its current location. In this work I try to cope with these problems by first going to the most likely hypothesis and from their verify this hypothesis.

To perceive the environment a panorama is used, which is a 360° image and has the big advantage of containing all the visual information around the point of view. Also in nature many animals have a wide field of view, for example the honeybee has a field of view of about 310° (Vardy (2005)).

The navigation techniques of bees have been studied by Carwright & Collet (1983), they studied how the honeybees learned and used landmarks to navigate. From this research they created the *snapshot model*. The idea of this model is to calculate the home vector, which is the vector pointing to the home position. This is done by trying to decrease the difference of the panorama at the current location with the panorama at the home location. The snapshot model was successfully tested by Lambrinos et al. (2000) with real robots in an environment with artificial landmarks.

A model which is based on the *snapshot model* is the *Average Landmark Vector model*. The Average Landmark Vector (ALV) is the average landmark location as seen from the robot. The home vector can be calculated by subtracting the ALV at the home location from the ALV at the current location. The advantages of this model are that it is a simple calculation

[1] "navigate *verb*" *The Oxford Dictionary of English* (revised edition). Ed. Catherine Soanes and Angus Stevenson. Oxford University Press, 2005. *Oxford Reference Online*. Oxford University Press.

and that only the orientation and the ALV at the home location have to be stored and not a whole image. As third advantage no matching of the landmarks has to be done; however landmarks or features still have to be extracted from the (panorama) image before the ALV can be calculated.

The ALV homing technique is used in this work to go to the most likely location, but where in most previous uses of this method landmarks are used, I make use of invariant features. The goal is to complement the global localization method by Ramisa (2006). His work is similar in spirit to the one of Tapus & Siegwart (2005), where the authors defined a fingerprint of a room as a character sequence where each character describes a certain type of feature in the relative position of the panorama. The features used were colour blobs extracted from a panoramic image, and straight angles extracted from a 2D range laser scanner lecture.

Ramisa however only used vision sensors; and the fingerprint consists on a constellation of feature points extracted from the panoramic image of the room. New fingerprints are compared by matching feature points and then by computing the essential matrix against all panoramas in the map. The panorama with the highest number of inliers is selected as the one corresponding to the current location. In his master thesis he tested several detectors and descriptors to create the features. He experimented with data from different rooms and his goal was to let the algorithm find out in which room the panorama was made, which later can be used as a global navigation method. This global navigation method showed some successful results but also several cases in which the method did not 'localize' the panorama correctly. These are the cases which are handled by the method in this work.

Ramisa chose three local feature detectors based on the work of Mikolajczyk et al. (2005): the Maximally Stable Extremal Regions (MSER; Matas et al. (2002)), the Harris-Affine and the Hessian-Affine (Mikolajczyk et al. (2005)), because of their robustness on the types of disturbances which affect the images when the robot is driving. Local features can shortly be explained as interesting points on an image that are robust to several transformations and illumination changes. Descriptors are used to characterise the detected regions. The descriptors Scale Invariant Feature Transform (SIFT; Lowe (2004)) and a variant, the Gradient Location and Orientation Histogram (GLOH) were used.

The advantages of using local features over global descriptors are that they are more robust to occlusions; and they already have demonstrated their usefulness in many tasks, including mobile robotics. A last advantage is that the features are robust to distortions from which the images, as seen from the robot, suffer. In most work about homing artificial landmarks are used, in this work however the invariant features are used, which do not have to be put on place before using the method.

Ramisa (2006) first extracts the descriptors from the panoramas which are then compared to the stored panoramas. The panorama with the highest number of matches is selected as the matching one. The essential matrix is computed and the outliers are filtered to improve the results. By calculating the essential matrix, the position of the robot can be recovered in the unit cylinder.

In my work the goal is to handle the cases in which no clear 'winner' has been found that matches an input panorama. At first the goal is to solve the ambiguity problem, in cases where a navigation method finds several possible matches such as in the work of Ramisa (2006). The research question which I try to answer is: *How can the ambiguity problem be solved in global localization?* Where I use the work of Ramisa as a global localization method, and make use of panoramas and features like he did.

The solution which is presented here focuses on using the ALV homing method to go to the most probable location at which a panorama was created. When the robot arrives at that location it can continue using the global navigation method.

Experiments with the ALV homing method were first done in simulation (Chapter 4, Goldhoorn et al. (2007*a*,*b*)) and because the results were promising, experiments were also done in the real world (Chapter 5) in an office environment.

Chapter 2

Background

The goal of this work is to handle ambiguity problem of a navigation method. Navigation is localization and mapping, which is discussed in the first section. The homing method discussed in this work uses invariant features which are explained in section 2.2. The homing method also requires panoramas (section 2.3) from which the features are extracted. Finally the ALV homing method itself is discussed.

2.1 Localization and Mapping

As cited in the introduction, the definition of navigation mentions "plan and direct the course". To plan a course there has to be some sort of knowledge of the environment, this can be in the form of a map. To be able to use such a map the robot also has to be able to localise itself on that map or with reference to some sort of coordinate system. Then the robot can localise itself and use the information of the environment to plan a route. No mapping method is used directly in this work, but the work of Ramisa (2006) is seen as a basis for that. In this work the focus is on the localisation on that map which I try to improve by creating a method that copes with ambiguity problems.

2.1.1 Localization

The easiest way to do localisation is by using the odometry sensor of the robot, which is called dead-reckoning. It uses the wheel turning to estimate the position with respect to a start location. This method however is not very trustworthy because of the errors in measurement which are created by for example wheel slipping and drifting. These errors accumulate and therefore make the calculated results useless in the longer run. To solve this, other sources of information have to be consulted (Thrun (2002)), such as cameras, sonar and laser rangefinders.

A problem which appears when using sensors is *perceptual aliasing*. When sensors have a low resolution then there is a higher probability of two locations to be indistinguishable. This problem could be tackled by using a higher resolution or by combining information from different sensors. Another problem is the *perceptual variability* which is the problem of having dynamic environments.

Cobzas & Zhang (2000) show that localization is also possible without a map, they use two panorama images (created by mosaicing, see section 2.3) from the environment and a planar

image at the current location to localise the robot with respect to one of the panoramas. They did the experiments in an office environment and made use of the planar floors which they used as an assumption in the method. Vertical lines are used as features, but the *correspondence problem* is not considered, because matching is done manually. The *correspondence problem* is the problem of determining if sensor measurements taken at different positions correspond to the same physical objects in the world. The error of their method was relatively small (5.5 cm and 3.5° for the rotation). In my work it could be interesting to localise the robot with respect to panoramas from a database to obtain a relative location in the room.

In Cobzas & Zhang (2001) a trinocular vision system is used to acquire both the intensity and the depth information from the environment. The images are divided into planar patches by using edges, and these patches are compared by using the intensity. Two or more planes of known position and orientation are required to compute the location. In the experiments the furthest point was 3 m away, the average error was 30 cm in position and 5° in orientation. Although depth is very useful, also in my work, I did not use this because it requires much more work, furthermore a trinocular vision system is expensive.

2.1.2 Mapping

Mapping is the creation of a map based on sensor information of the robot. Thrun (2002) discusses several mapping techniques. The maps created by a robot are limited (as mentioned in Thrun (1998)) by several factors: the sensors only measure a certain type of information, also their perceptual range is limited, they are sensitive to noise, and inaccuracy is caused by drift and slippage of the wheels. In general the complexity and dynamics of the environment make it impossible to maintain an exact model of the environment, and finally maps should be created real-time. Errors in measurement are even worse because they accumulate over time. Probabilistic techniques should be used to handle these errors and noise (Thrun (2000, 2002)).

Thrun (2002) mentions several problems of describing an environment, such as the amount of data, the dimensionality, the dynamic world, and choosing which way to go while creating the map. But the hardest is the *correspondence problem*.

There are two approaches to mapping (Thrun (1998)): *grid-based (metric)* and *topological*. In the metric approaches maps are evenly spaced grids in which information can be stored in each grid cell such as the presence of an obstacle. In the topological approach a map consists of nodes with information about a certain place such as a landmark or obstacle, and these nodes are connected when there is a path between the two. As Thrun (1998) mentions, advantages of the metric approach are that it is easy to build and maintain, recognition of places is non-ambiguous and view independent, and the shortest path can be calculated. These points are however not so easy for the topological approach, but this approach has other advantages: it permits efficient planning, does not require accurate determination of the robot's position, and it is a convenient representation for a problem solver. These advantages however do not apply to the metric approach.

Thrun (1998) tried to combine both approaches to get the advantages of both. He created a topological map from the metric map. Experiments with a moderate sized map showed that the efficiency of planning increased by several orders of magnitude, while the performance hardly decreased.

In this work no real mapping is done, however I use Ramisa (2006) as a basis. He made a global map by adding several panoramas to a database which were all labeled with a room

name. Although this method was not yet implemented for real navigation, it is a topological map when the links between the different rooms (in the database) have been made.

2.1.3 SLAM

When a robot has to create a map of the environment, it should also be able to localise itself on that map at the same time. This however is a chicken-and-egg problem, since the map is required for localization, and to create or expand the map, the robot has to know where it is on that map. This problem is known as Simultaneous Localization and Mapping (SLAM).

Tapus & Siegwart (2005) attempted to solve the SLAM problem by trying to recognise if the current place has been visited before. They use fingerprints to describe a place, this is a circular list of characters where each character represents a certain type feature. Two panoramas of characters are compared by using a string matching algorithm. A partially observable Markov decision process is used to do the simultaneous topological localization and mapping process. A new node is added to the map whenever there is an important change in the environment. The results were very good in indoor as well as in outdoor environments. Also a loop-closing test was performed successfully. In this test the robot has to recognise the place after haven driven a loop.

Some other approach is suggested by Booij et al. (2006*a*), who worked in an algorithm which unsupervisedly groups images of places in the human concept of *rooms*. The robot makes panorama images at several places which are assumed to be distributed uniformly over the available space. The map consists of a graph of nodes with edges between the nodes with weights representing their similarity. There is an edge between two nodes if it is possible to perform a 3D reconstruction of the local space using visual features from the two corresponding images. After this, the nodes are grouped by cutting the graph at weakly coupled places. Booij et al. (2006*b*) discusses how they prune the graph in order to optimise and speed up the processing of such a graph.

In this work I try to improve a localisation problem based on the work of Ramisa (2006), but as a next step a map should be created and maintained by the robot. The methods of Tapus et al. and Booij et al. can help in deciding when and where a panorama has to be made in order to expand the map efficiently.

2.2 Interest Points

The development of the eye has its origin about 540 million years ago, it is one of the senses which is used by numerous organisms. In robotics vision is also an often used sense. Big advantages are that a camera is relatively cheap and it gives a great amount of information. This last point however is also a disadvantage because it might be too much information, therefore the dimensionality of the data has to be reduced. A simple method to reduce dimensionality is to lower the resolution, but using images for comparison has its disadvantages.

A problem with just images is that they are difficult to compare with other images, because when another photo of the same object is made it will probably not contain the object in the same position, orientation, relative size, etc. Therefore a more robust comparison method is required.

Extracting features from an image reduces the dimensionality of the information, and it adds robustness against noise, aliasing and acquisition conditions. Local features can be defined as points or regions with a high information content, which correspond to a local

extrema function over the image. They should be resistant to image transformations, such as changes in viewpoint or in illumination. This robustness makes features very suitable to use for matching and recognition. They are resistant to partial occlusions and background clutter because they are local.

Most applications for local features require them to be compared, such as wide baseline matching, object recognition, etc. Comparing them involves the use of a local descriptor, which has to describe the local feature. This must be done in a compact way and it should be robust to illumination changes, noise and changes in the measurement region due to discontinuities or non-flat surfaces. The descriptor is calculated using the pixels near the feature point. The simplest descriptor would be the region pixels alone, however these are very sensitive to noise and illumination changes.

For a correct match the same pixels have to be used by the descriptor independently of the point of view, scale and illumination. The transformations that are most interesting for viewpoint changes are the affinity changes (Mikolajczyk et al. (2005)). Affine transformations can model viewpoint changes in a local region if the scene surface which produced the region in the image can be locally approximated by a plane. Local features which satisfy the above mentioned requirements are known as affine covariant regions (Mikolajczyk et al. (2005)). Mikolajczyk et al. (2005) tested more than ten local descriptors on affine transformations, rotation, scale changes, etc.

In this work invariant features are used because they can be extracted automatically and are robust. Therefore no external help such as artificial landmarks are required. The feature detectors MSER and DoG are used to test the homing method because they are relatively fast and are freely available. These detectors were also tested by Ramisa (2006), he however tested their descriptors, while here the detectors are used. Finally experiments with artificial landmarks were done as a comparison.

2.2.1 SIFT

The Scale-Invariant Feature Transform (SIFT, Lowe (1999, 2004)) algorithm is based on a model proposed by Edelman et al. (1997). Edelman et al. created a biologically inspired model based on complex neurons in the primary visual cortex. These neurons are activated by a gradient in a particular orientation if it appears within a small range of positions in the retina. Their hypothesis was that these complex neurons allow object recognition under small 3D rotations, which would introduce small displacements in the gradients position. Their experiments with a computational model showed an improvement in performance of more than 55% over direct correlation of the gradients.

The DoG detector Before describing regions, they first have to be detected. The first stage is to find the locations and scales that can be identified under different views. Lowe described that detecting locations that are invariant to scale changes can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as *scale space*. It has been found that under a variety of reasonable assumptions the only possible scale space kernel is the Gaussian function. Therefore the scale space of an image is the convolution of the Gaussian G and the image I:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

Lowe proposed using scale space extrema in the difference of Gaussian function convolved with the image to efficiently detect stable keypoint locations. This difference of Gaussian D can be computed from the difference of two nearby scales separated by a factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2.2}$$

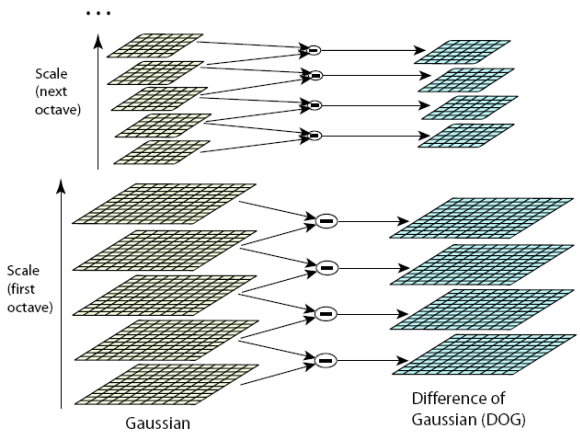


Figure 2.1: At the left the initial image is incrementally convolved with Gaussians. The adjacent image scales are subtracted to produce the DoGs, which are shown at the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated. (Taken from Lowe (2004).)

Figure 2.1 shows an efficient approach to construct D. The initial image is incrementally convolved with Gaussians to produce images separated by a constant k in scale space, shown at the left. The DoG is produced by subtracting adjacent image scales, shown at the right in Figure 2.1. The Gaussian image is resampled once a complete octave has been processed.

Local extrema of D are detected by comparing each sample point to its eight neighbours in the current image and the nine neighbours in the scale above and below. The point is selected only if it is the maximum or minimum in its neighbourhood. These points are then selected based on how sensitive they are to noise or whether they are localised along an edge. The next step is to describe the regions in order to be able to compare them.

The SIFT descriptor Although Lowe's SIFT descriptor is based on the idea proposed by Edelman et al., it has a different computational model. The image patch is divided into sixteen sub regions and a histogram of the orientations of the gradient is computed for every sub region, see Figure 2.2. A gradient can move within a sub region and still produce the same descriptor, in this way shift in position of the features is allowed.

Lowe uses a Gaussian weighting function to weigh the sample points to reduce the sensitivity to small changes in image position and to increase the importance of the centre of the measurement region. The centre points are less likely to suffer from registration errors introduced by non-planar surfaces and depth discontinuities. Trilinear interpolation is used to distribute gradient samples across adjacent bins of a histogram to avoid boundary effects in the gradient orientation.

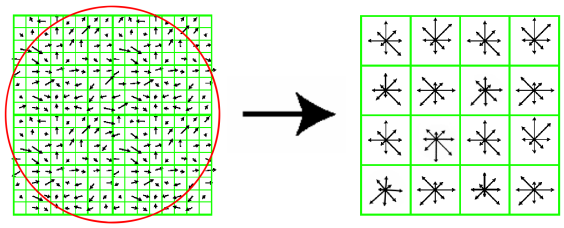


Figure 2.2: Local patches are divided in sub regions with image gradients and histogram construction by the SIFT algorithm (Lowe (2004)).

The values of the eight bins (each 45°) of each of the sixteen histograms are arranged as a vector of 128 dimensions. This vector is the descriptor which identifies a local feature. The vector is normalised to make it invariant to affine changes in intensity of the measurement region. However non-affine illumination changes such as camera saturation or a different reflectance can cause changes in the magnitude of some gradients, but it will rarely affect the orientation of the gradient. Each value of the normalised feature vector is thresholded to reduce the influence of these variations. The features can be matched by calculating the Euclidean distance between their descriptors.

2.2.2 MSER

The Maximally Stable Extremal Regions (MSER; Matas et al. (2002)) in images are regions in which the pixels have a higher or lower intensity than the pixels at the border, these are the *extremal regions*. When the intensity is changed, the regions will still be there, therefore it is *maximally stable*. Also geometric change will not change the regions. The *extremal regions* have two desirable properties: they are closed under the affine transformation of image coordinates and they are invariant to affine transformation of intensity.

The algorithm works as follows (Matas et al. (2002)). First the pixels are sorted by intensity, then the pixels are placed in the image (in decreasing or increasing order) and the list of connected components and their areas are maintained using an efficient union find algorithm. Each connected component is stored as a function of intensity. By doing intensity thresholds the parts of the function are found where no changes in the area of connected components occur, i.e. they are not merged with others. These parts are the maximally stable extremal regions.

In the next step a measurement region is defined around each MSER. This is a region of arbitrary size, which should be constructed in an affine-invariant way. These regions are used to construct the descriptor for each MSER. Having small regions increases the likeliness of them being planar, but at the same time it is less discriminative, i.e. indistinguishable from other regions. Matas et al. (2002) propose regions of different scales for every MSER. The MSERs are normalised to a circle using the second order moments of the ellipse that encloses the region.

The MSER detector was also tested by Mikolajczyk et al. (2005) and found to be one of the best in their robustness experiments. A big advantage of this method over SIFT is that it is much faster thanks to a watershed like algorithm.

Figure 2.3: An example of a landmark in the *robot laboratory*. The bar code of the box at the top is used, the lower box was used in older experiments.

2.2.3 Landmarks

Busquets (2003) used landmarks to navigate through an environment. The landmarks contain a bar code (Figure 2.3) from which a number can be extracted. The landmarks have one big bar at the top and five bars of a quarter size of the first below. Each of the five small bars represents a binary number. A bar at the left represents 0 and a bar at the right represent a 1, this makes 32 possible combinations. Since the size of the bars is known, the distance to the landmark can be calculated.

2.3 Panorama

In this work panoramas are used which are images of a 360° view. An advantage is that information from all directions is available at once. The ALV homing method (see section 2.4) requires such a view, it might be possible with a smaller view but it certainly will make it more complicated.

A relatively basic way to create a panorama consists in stitching images taken with a camera rotated around a fixed point of view until a full 360° have been covered. The next step is to stitch those images together to complete the panorama. The images should be projected onto a smooth surface such as a cylinder or sphere to avoid discontinuities or inhomogeneous sampling. A sphere is rather difficult to create because of the tilt movements that are required and the lack of a convenient data structure to store a uniformly sampled sphere.

A cylindrical representation offers some advantages, it can be created relatively easily, and it can be unrolled and stored as a rectangular image. The field of view however, is limited in the vertical direction. It is important that the optical centre is fixed, otherwise motion parallax would be introduced. However the small translations can be tolerated when the objects are far enough from the camera.

First the coordinates have to be transformed from the Cartesian system of the images to the cylindrical coordinate system:

$$\theta = \tan^{-1}\left(\frac{x}{f}\right), \quad v = \frac{y}{\sqrt{x^2 + f^2}} \tag{2.3}$$

Where (x,y) is the pixel position in the image, f the focal distance (in pixels), θ the angular position and v the height on the cylinder. The radius of the cylinder is equal to the focal length of the camera to optimise the aspect ratio of the image (Shum & Szeliski (1997)).

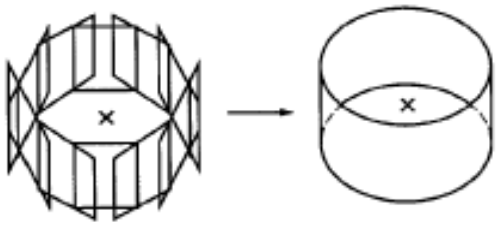


Figure 2.4: The projection from the image sequence to a cylinder.

The next step is to stitch the images (Figure 2.4), but for this the displacement vectors $\Delta t = (t_x, t_y)$ have to be calculated for each succeeding image pair. In theory t_x can be deduced from the panning angle and $t_y = 0$, however in reality this is not true due to camera twist and not perfect panning.

Also features (see section 2.2) can be used to estimate the translation between two images. The advantages of this method is the robustness to several image transformations such as illumination changes, noise and zoom. When the translations have been calculated then the images can be stitched to produce a panorama. This is what has been done in this work because the sources (camera, pan tilt unit and algorithms (from Ramisa (2006))) were already available. An example of such a panorama created by stitching is shown in Figure 2.5. As can be seen there are still small distortions due to not perfect shifting of the images. The difference in intensity is because of the automatic camera gain.

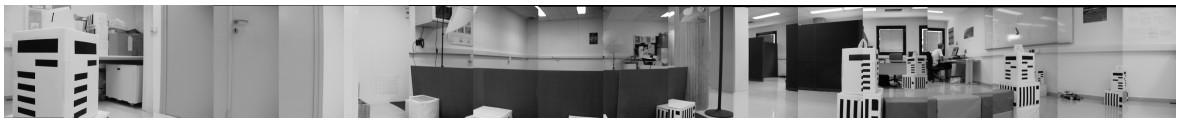


Figure 2.5: An example of a panorama image created by stitching several images together. The image is made in the robot laboratory (panorama 119, see Figure 5.6).

Another way to acquire panoramas is by using an omnidirectional camera. There are two approaches to do this, first by using a fish-eye lens, which is a wide-angle lens that takes an extremely wide, hemispherical image; and secondly by using a conventional camera pointed to a hyperbolic mirror above it. These methods have some clear advantages such as the speed of creation and no images have to be stitched. A disadvantage is the lower resolution. Vardy (2005) used a parabolic mirror for his image database from which two examples are shown in Figure 5.11.

Another alternative to acquire panoramas is using a camera ring. This is a ring of synchronised cameras and has as advantages the high speed of acquiring the image and the angle of rotation between each image pair is fixed and known. A disadvantages of this method is that the price of the whole system is more expensive because more cameras are used, and therefore also more data is generated.

2.4 Average Landmark Vector (ALV)

The snapshot model has been used to explain the insect navigation for about two decades (Carwright & Collet (1983) cited in Lambrinos et al. (1998)), this model assumes that a panoramic image of the target location is created and stored by the animal. When an insect wants to go back to the stored position it uses a matching mechanism to compare the current retinal image to the stored panorama. Lambrinos et al. (1998, 2000) suggest the Average Landmark Vector model as a better way to explain the comparison. This model assumes that the animal stores an average landmark vector instead of a snapshot. Landmarks can be (simple) features like edges. The direction to the destination is the difference of the ALV at the destination and the ALV at the current location.

Lambrinos et al. (2000) used the work of Wehner (1987) to study how an ant species which lives in the Sahara, the Cataglyphis, navigates. It cannot use pheromones (hormones which are dropped by the ants in order to 'define' a path, these hormones can be detected by other ants) to find their way back, because the pheromones evaporate in the desert. Three main strategies were found to be used by the Cataglyphis: path integration, visual piloting and systematic search. As visual plotting technique Lambrinos et al. (2000) discuss the snapshot model and later on a simplification, the ALV model. This latest model will be explained now.

The Average Landmark Vector is the average of the landmark vectors:

$$ALV(F, \overrightarrow{x}) = \frac{1}{n} \sum_{i=0}^{n} \overrightarrow{F_i} \tag{2.4}$$

Where F is a n by 3 matrix (or n by 2 when only two dimensions are used) containing the positions of the n features, and $\overrightarrow{x}$ is the current position. F_i is the i^{th} landmark position vector. In this equation the F contains the global feature positions to explain and proof the homing technique. This is the robot centred version, but it is made world centred by subtracting the current position ($\overrightarrow{x}$) to easily proof that the homing technique works :

$$ALV(F, \overrightarrow{x}) = \frac{1}{n} \sum_{i=0}^{i=n} \overrightarrow{F_i} - \overrightarrow{x} \tag{2.5}$$

To differentiate between the world coordinate system and the (self centred) coordinate system of the robot, the home vector is defined as follows:

$$homing(F, \overrightarrow{x}, \overrightarrow{d}) = ALV(F, \overrightarrow{x}) - ALV(F, \overrightarrow{d}) \tag{2.6}$$

Where $\overrightarrow{x}$ is the current location of the robot and $\overrightarrow{d}$ the destination. When the ALV functions are substituted by Eqn. 2.5 than $\overrightarrow{d} - \overrightarrow{x}$ remains, which exactly is the home vector.

Figure 2.6 shows an example of the calculation of the home vector. To simplify the image only the average landmark (the gray square) is shown and not all landmarks in the world. Note

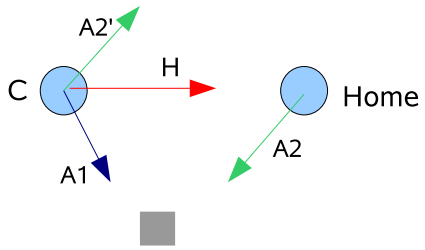


Figure 2.6: The calculation of the home vector. Both ALVs (A_1 and A_2) point to the average feature position, which is drawn as a gray block. The home vector (H) is calculated by subtracting the ALV at the destination location (A_2) from the ALV at the current location (A_1). This subtraction is shown, by the addition of the reverse vector, A_2', to A_1. The robots are aligned in this example.

that the (average) landmark in the world differs from the landmark positions seen from the robot. In this example it is also assumed that the depth of the landmarks is known. The ALVs are calculated for the current (C) and the *Home* position, this are A_1 and A_2 respectively. The home vector (H) is calculated by subtracting the ALV at the destination position (A_2) from the ALV at the current position (A_1). This can be done by adding the opposite vector of A_2, shown as A_2', to A_1, and results in the home vector H which points to the destination location.

ALV homing does not work when the ALV at the current location and at the goal location are the same (after correction for orientation differences), because this results in a zero vector. An exceptional theoretical case in which this could happen is when the ALV point, the current location and the goal location are on one line, in practice however this is very unlikely. To let the robot move anyway in such situations a random vector could be used to move the robot a small distance, and then continue the homing procedure.

After a discussion of related work to homing, experiments with the ALV homing method are being discussed in combination with invariant features (section 2.2).

Chapter 3

Related work on Navigation and Homing

A great amount of literature has been written about the subject of robot navigation, which I will not discuss thoroughly. The focus of this work is on using a homing method to deal with situations in which the robot is not sure about its current location.

Two types of homing can be distinguished: associative homing and local homing. In the first case the environment is sampled by recording the position and view of each location. In the second case homing is performed by only using information from the current and target location. Associative homing is performed by first matching an image from the current location with all the stored images. The location from the matched image should be the location of the robot at that moment, and with that information it can go to its home. Local homing only uses the information from the current position and goal position.

An example of associative homing is Nelson's Tinytown (Nelson (1991)). A robot arm with camera swept over a miniature town to create a total of 1200 images. Each image association with the home vector was directed toward the goal. It was successful for a major part of the town.

A large disadvantage of associative homing is the time it takes to create an image database, nor is their any evidence that insects or other animals sample their environment in such a way. The use of local homing does not require such a large amount of information and is also found to be biological plausible (Franz et al. (1998); Lambrinos et al. (2000); Carwright & Collet (1983)).

Most homing methods, like the ALV method used in this work, are inspired on biological examples therefore I will start by discussing those.

3.1 Biologically Inspired Examples

The subject of *visual homing* is thoroughly discussed in the work of Vardy (2005), he tries to find visual homing techniques which are *competitive* in performance and yet *biological plausible*. For the first condition he uses the method of Franz et al. (1998) as comparison and he refers to their method as *warping* (this work will be discussed later on in section 3.3). Secondly he defines *biological plausible* as "the likelihood that the same mechanism is being used in both insect and robot"; as a condition the method should adhere to the constraints of retinotopic processing.

Retinotopic processing is a concept of the brain structures that process visual information. These structures can be seen as forming a map and this is because of the way the neurons are connected. Each of these neurons processes a partly overlapping (with its neighbours) visual field. This was first found by Hubel & Wiesel (1962) who did experiments with a cat, and it was also found in the optic lobe of the bee.

In the work of Vardy, the methods which do not meet the constraints of retinotopic processing are considered less biologically plausible, however they cannot be ruled out unless they require much more computational power than a typical insect brain. To test the different methods he first made a database of images with a robot and a camera pointed to a hyperbolic mirror. Several data sets were acquired under different conditions in two rooms (see Appendix D for an explanation of the contents of the data sets).

3.1.1 The Snapshot model

Carwright & Collet (1983) and Franz et al. (1998) subdivide insect navigation according to the different strategies and modalities used. The most important are path integration (i.e. dead reckoning) and the use of visual landmarks.

Carwright & Collet (1983) created the snapshot model from experiments done with honeybees. They trained the bees in a special room with a food source (sucrose) surrounded by 40 cm high cylinders (and 4 cm in diameter); other visual cues were eliminated as much as possible. When the food source was removed and landmarks modified, then the bees would search at the place where bearing and apparent size of landmarks were most similar to the snapshot image. It was found that using only the frames of the landmarks was enough, from which can be concluded that the edges are most important. When the bees were trained with three landmarks surrounding the food source, then restoring the bearing was found to be most important. After rotating the landmarks 45° or more (around its centre) the bees were not able to learn the location anymore, this means that the insects store the coordinates with respect to an external compass.

Carwright & Collet (1983) developed five computer models to investigate and mimic the navigational techniques of the honeybee. These models were tested in a two-dimensional simulation with circular black landmarks. The agents perceived the world as a binary one-dimensional panorama. All the five models employ bearing of landmarks, matched between the snapshot and the current image; the models differ in the features (edges or regions), the use of a compass and the use of correcting vectors.

To return to home, the features in the current image should be brought closer to the features in the snapshot, these are called pairings. Two vectors are created for each pairing. The radial vector points to the feature if it has shrunk in the current image and it points in the opposite direction if the feature has grown. The tangential vector is directed to the left or right to match the direction of the movement of the feature.

The results of the fifth model were the closest to the experimental results from the bees. This model uses regions as features, a compass to compensate changes in orientation and it combines the tangential and radial vector to correct each pairing. The home vector is obtained by summing all radial and tangential vectors.

3.1.2 Using the *Cataglyphis* ant as an example

Lambrinos et al. (2000) studied the navigational strategies of the *Cataglyphis*, an ant species

that lives in the Sahara. They used the work of Wehner (1987), who studied this ant species thoroughly during more than two decades of field work. Three main strategies were found to be used by the *Cataglyphis*: path integration, visual piloting and systematic search. Lambrinos et al. (2000) created the *Sahabot 2* robot to investigate the first two techniques.

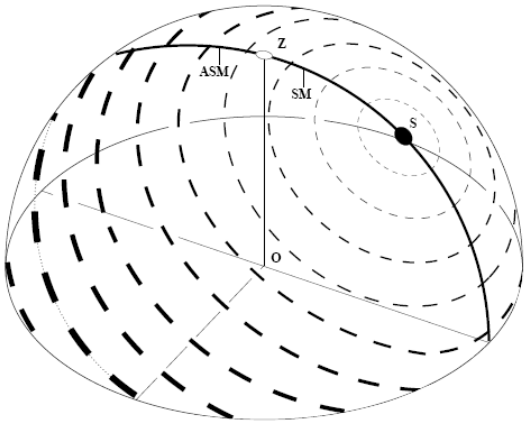


Figure 3.1: The pattern of polarization of the sky as seen from point O. Orientation and width of the bars depict the direction and degree of polarization, respectively. A prominent property of the pattern is a symmetry line running through sun (S) and zenith (Z), called *solar meridian* (SM) on the side of the sun and *anti-solar meridian* (ASM) on the opposite side (Lambrinos et al. (2000)). (Figure taken from Lambrinos et al. (2000)).

Path integration is the technique of continuously updating a home vector taking the robot movement into account. The easiest method is to use data from the wheel encoders, this does work for short distances, but for longer distances it suffers greatly from accumulating errors. The path integration method by Lambrinos et al. is based on compass information which is gained from the polarisation pattern of the sky. This polarisation occurs because the light goes through molecules in the atmosphere. The polarisation is greatest on light that is perpendicular to the sunlight rays (see Figure 3.1). In insects neurons have been found which are sensitive to the polarisation. They have compound eyes which exist out of a few to thousands of ommatidias, which are small photoreceptors that can distinguish brightness and colour. Their visual resolution however is low, but their temporal resolution is so high that even some insects can detect the polarisation of light. Generally their field of view is also larger, the honeybee for example has a field of view of about 310°.

For the robot POL (polarisation) sensors were created, three were placed on the robot *Sahabot 2* which were adjusted to be sensitive to 0°, 60° and 120° respectively. It is possible to derive the orientation (with respect to the solar azimuth) by using the polarised-light compass, but compensation for the movement of the sun during the day is required; also the season and geographical latitude have an effect. Experiments showed that the use of the POL-compass was significantly better than using proprioception.

The path integration process introduces errors, therefore the ants also use visual landmarks. In the case that no known landmarks are recognised, a systematic search is done.

Experiments with bees and ants have shown that animals store a visual snapshot of the environment. A direction can be derived by comparing snapshots. In the model of

Carwright & Collet (1983) unit lengths for the vectors were used, however for navigation the length is important to prevent driving too far and to be able to find out if the target is reached. The *proportional vector model* uses the difference in size of the projections of the landmarks to derive the vector length.

The robot had a camera with conical mirror above it to create a panoramic view. Alignment of the current and target snapshot is vital for the method. The POL-compass of the robot was used because ants also use compass information.

A second variant is the *difference vector model* which does not have radial vectors and replaces the tangential vectors by secant vectors, which are calculated by taking the difference of the vector pointing to a region in the snapshot image with the vector in the current image pointing to the same (i.e. matched) region. This model is thought to be quite similar to the proportional vector model.

Experiments were performed in Tunisia, the same place were experiments with the Cataglyphis were done. The robot moved in an area of about 5 m $\times$ 6 m. The deviation from the home position was between 7 and 20 cm.

Finally Lambrinos et al. (2000) also discuss the ALV homing method. An advantage of this method is that only the ALV and the orientation at the target location have to be stored. They however did not do any experiments with this method because it is too similar to the snapshot model according to them. They extracted the landmarks from a one-dimensional image in the same manner as the snapshot model. It however is less expensive in time and less complex because it does not require matching and only the ALV of the home location has to be stored. Lambrinos et al. (2000) showed that this model however generates home vectors identical to the difference vector model for cases when the landmarks are correctly paired. Furthermore they suggest that the ALV model implicitly establishes the correct pairing.

3.1.3 Augmented Navigational Algorithm

One limitation of the ALV homing method is that it is a local method. As soon as the environment has changed too much (i.e. too many previously seen landmarks have disappeared and/or too many previously not seen landmarks have appeared), the method does not work (as well) any more. Smith et al. (2006) created the *augmented navigational algorithm* to tackle this problem. It consists of three components: foraging, visual route learning and visual navigation. First it forages, by moving randomly and in the mean time keeping a vector to the nest, using path integration. When the food source has been found and (partly) consumed, the robot starts the visual route learning. First it stores the global home vector and then at each time step the ALV is calculated and compared to the previous ALV. The ALV is stored when the difference is significant. This process is continued and thereby creating waypoints to the food source. When it returns to the food source, it uses the previously stored home vector at the food location to find its direction back. It uses the same technique to create the waypoints. When the waypoints for both trajectories (to and form nest) have been stored then the robot uses the waypoints to navigate. It starts by navigating to the first waypoint, and the next waypoint is used as home when the difference between the current ALV and the ALV of the waypoint is small enough. This continues until the destination has been reached.

Experiments were done in a two dimensional simulated environment with 25 cylinders as landmarks and a nest and food source. The augmented ALV method worked better for larger landmarks. The normal ALV method did not work in most cases because not all landmarks

were visible, with the exception of a few cases where the nest and food source were randomly placed in the same 'visual locale'. The method performed worse when the landmarks were smaller, because there is a higher probability that there is no landmark visible. Also when there are less landmarks, the probability of perceptual aliasing is higher.

Using waypoints improved the homing method significantly, however Smith et al. (2006) were not the first to use waypoints to 'upgrade' the homing method to a navigating method. It has been done before by Hong et al. (1991), however not using the ALV method, as will be discussed later this chapter.

This augmented ALV method could be used as a local navigation method itself. In my work however the focus lies on returning to a previously visited location to verify the hypothesis of the current location of a more global navigation method.

3.1.4 The ALV in an analogue circuit

Möller (1999, 2000) implemented the ALV model in a robot with purely analog hardware. Convergence of the ALV model is shown when the landmarks stay visible and there are at least two landmarks which are not on one line with the goal. According to Möller the analog components share properties with biological systems such as the lack of a global clock, parallelism and continuous value signals. The experimental results were good, but it was not clear that they could be performed in more natural environments. According to Möller the model could be implemented in an insect brain.

Möller et al. (2001) applied the ALV model in an office environment using the basic Lambrinos et al. model, but first they extracted features. These are extracted from one-dimensional images using the adaptive threshold scheme. It worked reasonably well within the test environment but not better than warping.

Möller et al. (1999) created a neural snapshot model using a neural architecture, and to prevent making it too complex it employs the proportional vector model and uses edges as features. The neural model consists of several layers of processing rings where the neurons in each layer connect locally, except for the inner layer which has more convergent connections. The outermost layer has the one-dimensional panorama image as input. The performance of the neural model was comparable to the original snapshot model, but their experiment showed that it is biological plausible. However the model is not directly usable in the real world because of the simplistic feature detection.

Using only one-dimensional images results in the loss of a great amount of information. Vardy and Möller (Vardy (2005); Vardy & Möller (2005)) calculate the home based on the shift of landmarks. If the correspondence vector is correct, then the home is correct too, but they assume the first not to be true. It would be possible to use a technique such as RANSAC (RANdom SAmple Consensus), which is an algorithm to estimate parameters of a mathematical model from a set of observed data which contains outliers. A simpler way is to average all the home vectors calculated for each matched landmark pair by which the errors should cancel each other out.

3.1.5 Learning to home

Hafner (2001) used Hebbian learning to do visual homing. A multi-layer-perception with one input layer for both the snapshots and an output layer of only the home vector was enough. This learning method was used to simplify it and to make it more biological plausible. The

method turned out to be related to the ALV homing method. An advantage of the learned method is that no feature extraction is required.

Experiments were performed with the mobile robot *Samurai* with a USB camera and a convex mirror attached to the robot. Only the intensity values a few degrees above and below the horizon are used. First that part of the image is transformed to its polar mapping, then a low pass filter is applied to reduce noise. Also a normalised low resolution vector of the place image is available to the neural system.

First tests were performed offline with recorded images on an 8×10 matrix (140 cm $\times$ 180 cm). The robot was also equipped with a compass, differential steering and wheel encoders. The compass value was fine tuned by using camera data, because the magnetic compass does not work well inside buildings. The average angle error was smaller than 90° in more than 92% of the training cases, and smaller than 45° in more than 69%. Performance in the real world during on-line learning had been tested by recording the angle errors. Hafner mentions that the performance of her method lies between the snapshot model and the ALV model. A big advantage however is that no feature extraction or landmark detection is required.

Centre-of-Mass ALV Vardy (2005) refers to Hafner's method as the Centre-of-Mass ALV (COMALV), because the ALV is directed to the centre of mass of the image. A one-dimensional image is created by summing the intensity of each pixel per column, this result is then low-pass filtered and down-sampled. The ALV is then calculated as follows:

$$ALV_{COM} = I(\theta) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{3.1}$$

where $I(\theta)$ is the value of the one-dimensional image at index θ. In comparison with the previously discussed ALV (section 2.4), this method does not need any landmarks or features, it can use more or less the raw image after some pre-processing. For each index θ a vector is calculated with a length of $I(\theta)$, and this value is the sum of the intensity values in a column. The places in the image with the highest intensities, i.e. the centre-of-mass, will thus have the longest vectors.

Obstacle avoidance A not always mentioned point in navigation is the obstacle avoidance. This is an essential behaviour of an autonomous robot. Hafner (2003) shows that this behaviour actually emerges from the ALV homing method. When a robot gets closer to an object the landmarks behind it get occluded (see Figure 3.2), therefore those landmarks will not be used in the calculation of the ALV. This can be seen as a repulsive vector because the vector(s) to the occluded landmarks are now 'subtracted' from the ALV with respect to the previous ALV in which they were visible.

3.1.6 The turn-back-and-look behaviour

Bees and ants make learning flights and walks to determine the distance of landmarks to a goal. This finding is used by Lehrer & Bianco (2000) to select distinctive landmarks. They found that incorporating a biologically inspired turn-back-and-look behaviour to select distinctive landmarks increased the efficiency and success rate of robot homing within an unmodified office environment.

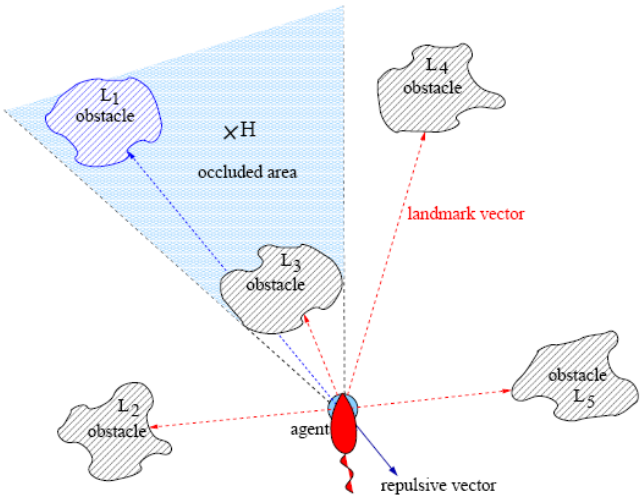


Figure 3.2: Due to the occlusion of obstacle L1 by L3 the vector to L1 is not used in the ALV and therefore can be seen as a repulsive vector. (Figure taken from Hafner (2003).)

This behaviour could be used to search for a good location from which distinctive landmarks are visible. There a panorama could be created that represents the room on the global map.

3.2 Image based homing

To compute the home vector using features requires correspondence vectors in most visual homing algorithms. A correspondence vector for a certain feature starts at the location of the feature on the snapshot and ends at (points to) the location of that feature in the current image. This vector is in the image plane.

3.2.1 The first homing algorithm

One of the first homing algorithms was created by Hong et al. (1991). They use a one-dimensional intensity image and make a signature of the location. Characteristic points are computed as well-separated points with high estimated first derivatives. The matches in the current image are scored by correlation of the surrounding windows. The algorithm requires a consistent orientation, it was tested along seventeen waypoints in an office environment. It was able to retrace its route using the waypoints.

Hong et al. (1991) use one-dimensional images which is simple nor do they use depth information. Their idea is to move the current view in such a way that the landmarks shift towards the matched landmarks in the snapshot. For each landmark a home vector is calculated which are then averaged. Franz et al. (1998) argue that their method relies on the *isotropic distance assumption* which says that the frequency and distance of landmarks are assumed to be independent of the viewing direction. If this assumption is true, then all error components will be cancelled out. Also they show that the method of Hong et al. never has an error of more than 90° which means that the agent will converge to its home nevertheless.

Röfer (1997) created a wheelchair robot and uses the same technique as Hong et al. (1991), but he uses three colour channels. A spherical mirror is used to generate panorama images where after it uses matching. Estimation techniques are used to detect changes in orientation of the robot between the current and home image. In an experiment the robot is taught to drive a certain path with the help of waypoints. During driving it continuously takes images and determines the driving direction. The robot was able to retrace the learned routes with reasonable accuracy.

Weber et al. (1999) used variations of Hong et al. (1991) and wanted to search for the best landmark correspondence algorithm. They tested several algorithms from simple close match pairing algorithms to exhaustive searching. They found that the advanced algorithms are not necessary because the accuracy is most important when the agent is close to its goal, but then corresponding is easier therefore complex algorithms are not required.

Gourichon et al. (2002) use colour information for matching like Röfer (1997) and they use the same correspondence process as Weber et al. (1999). Two-dimensional regions of the same colour are used as sectors to match the current and target image. They use a custom biologically inspired visual perception chip. A dynamic programming algorithm takes all coloured regions into account, but they only capture partial panoramas of 200°. Their results with a robot in an office environment were moderate.

The most important conclusion from this section is from Weber et al. (1999) that an advanced algorithm is not necessary to perform homing.

3.2.2 Difference function

Zeil et al. (2003) wanted to use the difference between images to determine the home vector in outdoor environments. They use the root mean square (RMS) difference between the current and target image and this difference was found to increase monotonically with the distance in most cases. Not only was the difference found to be increased smoothly with translation but also with rotation.

The RMS difference between two images showed a minimum when they had the same orientation. The maximum tested distance between images with different orientation was 1.22 m. The agent would first perform one-dimensional gradient descent in rotation to align itself with the snapshot image. Then it could compute the exploratory movements in the direction of the greatest descent.

The computational expenses are low, and since it works with outdoor images, it appears to be biologically plausible. An advantage is that no landmarks have to be identified.

Vardy (2005) applied SSD to the images in his database and found that gradient descent is possible in some parts of the database, but depending on the distance with the snapshot and the degree of rotation.

Zeil et al. (2003) tested two methods for homing with gradient descent, using images with constant orientation. The first method starts by moving the camera in an arbitrary orientation. When the RMS difference between the snapshot and current image begins to increase, the movement direction is rotated 90° counterclockwise. Then the camera moves until the RMS difference values increase again. In the second method the robot makes two exploitative moves to compare the RMS values for three locations. It then moves in the direction of minimal RMS. Both methods were able to return within 5 cm of the snapshot in 90% of the cases.

Vardy tested two variants of the second method on the image database. The *GradDescent* method performed best. It first makes four exploratory moves and then sums the four weighted

vectors of each exploratory move, where the weight is given by the sum of squared difference (SSD, not the RMS) in that direction. The sum of these vectors should point in the direction of the greatest ascent. The opposite direction should therefore be the greatest descent, and therefore the way the agent has to travel. This GradDescent method however performed worse than warping and COMALV.

3.3 Warping

Most homing methods implicitly assume an *isotropic landmark distribution*, meaning that the landmarks are evenly distributed around the robot. Franz et al. (1998) show that violating this assumption is not disastrous. In their proof they use the technique of Hong et al. (1991) with a constant orientation to show that the error should not exceed 90° in a static environment. Furthermore the correspondence problem is assumed to be solved and there are more then two non-aligned landmarks. With an error smaller than 90° the distance to the home position should decrease monotonically.

The warping method operates by searching and comparing a warped version of the current image with the snapshot at the home location. They try to match the viewpoint of the snapshot image, this is done by searching through the parameter space for the warped image that best matches the stored snapshot image. This is not very hard because the images are one-dimensional.

Their experiments were done in a 118 cm × 102 cm area with toy houses as landmarks. A modified Khepera robot was used with a camera pointing to a conical mirror, which gave a panoramic view. The robot performed robustly up to an average distance of 15 cm from the home position. In an experiment in an office environment the robot performed successful up to 2 m away from the home. They discuss two homing algorithms: one with a *constant distance assumption* and one with the *isotropic distance assumption*. The accuracy of the first increased, but then decreased until a distance of 15 cm, the later stayed more or less constant.

Vardy (2005) refers to the method of Franz et al. (1998) as *warping*. According to Vardy the advantage of the warping method is that it is solid mathematically underpinned, has a good performance and the ability to estimate the orientation along with the home direction.

Vardy discusses the biological plausibility of the three methods and mentions that warping involves search, which does not satisfy his retinotopic processing requirement, but since it works with one-dimensional images, it is possible to do it real-time.

Vardy did experiments with warping, he used his image database as input for the experiments. The images were transformed to one-dimensional images by first extracting a fixed angular distance above and below the horizon and then summing the pixel intensity for each column. As a next step the image should be low-pass filtered and down sampled.

Vardy (2005) deduced the formula using flow fields and the equal distance assumption:

$$\delta = \arctan\left(\frac{\rho\sin(\theta-\alpha)}{1-\rho\cos(\theta-\alpha)}\right) - \psi \tag{3.2}$$

Where angle δ is the flow and represents the shift of a landmark on the panorama; θ is the angle of the landmark on the panorama of the current position. The mapping is done by trying to find the warped image S' using the current image C:

$$S'(\theta) = C(\theta+\delta) \tag{3.3}$$

Several values for the parameters α, ρ and ψ should be tried to find a S' which closely resembles the snapshot of the goal location. The home direction can be calculated when δ is known.

The results of Vardy's experiments were good, but warping performed worse near the boundaries. This however can be explained by the *equal distance assumption* which is violated near the boundaries and corners. Franz et al. (1998) do argue however that if the distance to the home is (much) smaller than the distance to the landmark, then the error remains small and it will converge to its home. Near the boundary however, the distance to the landmark will be small and therefore the error might be larger. However warping remains relatively robust to many changes.

Stürzl & Mallot (2002) used the warping technique, but instead of intensity images, they used range images. For this they used a special panoramic stereo mirror which encodes the approximate distance of viewed objects. This method was also tested on images from a robot in an arena with toy houses. The original warping method was still superior, but this variant performed better in situations with changing illumination.

3.3.1 Vardy's findings

Vardy (2005) tried several methods before arriving to a method which is competitive to warping and biological plausible. Two methods were found to be better than warping: *FirstOrder* and *SecondOrder*. The first employs first-order derivatives to estimate the direction of feature translation. The last employs second-order derivatives. They both assume that image brightness is conserved. They employ a Taylor approximation to determine the direction of feature translation, but this is only valid for small feature translations.

FirstOrder performed very good on the *original* image set, *SecondOrder* less but was consistent over all sets. Both techniques could be combined by using the average and thereby resulting in a more robust method. Vardy also proposes an hybrid approach which uses the difference method of Zeil et al. (2003).

With supervised learning an algorithm could be optimized, as has been done by Hafner (2001) to arrive at COMALV. He also suggested to integrate the different homing techniques such as path integration, recognition-based recall of stored vectors and the use of landmarks en route.

From the results of the experiments with this homing method Vardy concluded that there is a visual homing method that is competitive in performance, yet adheres to the retinotopic constraint. Two other important conclusions of his work are that rich image descriptors are not necessary for visual homing and that bad correspondences can be tolerated (up to about 10%), while it was previously thought that filtering was crucial.

3.4 Comparison

In this chapter several homing techniques have been discussed with each their advantages. Before discussing the experiments done, I first show a comparison of these methods in Table 3.1.

Only one associative homing method has been discussed (by Nelson (1991)), but the disadvantage of this method is that first images of several locations have to be stored in combination with the home vector. After this the image of the current location has to be compared to the images in the database. This is first of all slow in creation as well in use and secondly it does not seem to be biologically plausible.

Several methods make use of matching like Hong et al. (1991), others make use of the snapshot model or the ALV model. The used landmarks or features also differ. Some use cylinders as landmarks (such as Lambrinos et al. (2000)), others toy houses (Franz et al. (1998)), and others use features (like Möller et al. (1999)) but some use other information from the images such as the intensity (for example Hafner (2001)).

As a homing method the ALV is used in this work, because it is simple yet robust. In contrast with Lambrinos et al. (2000) for example, no artificial landmarks were used, but invariant features. These features are robust in the sense that they are found on more or less the same place even when the point of view or light conditions have changed. Furthermore the homing method discussed here is also meant to be an extension to a navigation method which makes use of such features (as discussed in Ramisa (2006)).

reference	method	landmarks / features	orientation	experiment
Carwright & Collet (1983)	snapshot model	cylinders	compass	honeybees and simulation
Franz et al. (1998)	warping	toy house	method estimates it only near goal	118 cm × 102 cm area with toy houses
Gourichon et al. (2002)	snapshot model	2D regions of same colour	constant	office environment
Hafner (2001)	Hebbian learning, COMALV	image intensity	compass (with camera data)	offline 140 cm × 180 cm field
Hong et al. (1991)	matching	well-separated points	constant	hallway, 17 waypoints
Lambrinos et al. (1998, 2000)	snapshot and ALV model	cylinders	polarisation	desert
Lehrer & Bianco (2000)	turn-back-and-look behaviour	comparing templates	calculated by the method	office environment
Möller (1999, 2000)	ALV model (analogue)	black A4	(electronic) compass	1 m × 1 m white room
Möller et al. (1999)	neural snapshot model	edges as features	constant*	simulation
Möller et al. (2001)	ALV model	thresholded 1D image	constant	office environment, grid of 3 m × 4.5 m
Nelson (1991)	associative	edge patterns	constant	Tinytown
Röfer (1997)	matching	correlation matrix	estimation techniques	waypoints
Smith et al. (2006)	Augmented Navigational Algorithm (ALV)	cylinders	compass* (simulated)	simulation
Stürzl & Mallot (2002)	warping	toy house	constant*	toy house arena
Vardy & Möller (2005)	warping, snapshot model	none, images used	compass (online)	online and offline
Weber et al. (1999)	matching	features (cylinders in experiment)	compass*	simulation and real world
Zeil et al. (2003)	RMS different	RMS difference	constant / RMS	outdoor

Table 3.1: This table shows some properties of the discussed homing methods. The landmarks / features and orientation columns show the information used in the experiments.

*These papers did not explicitly mention if any orientation compensation was used, and so yes what kind.

Chapter 4

Simulation Experiments

In most researches discussed in Chapter 3 in which the ALV was used to perform homing (section 2.4), landmarks were used. However in this research not artificial landmarks but invariant features are used. To find out how well this works, the experiments are first done in simulation.

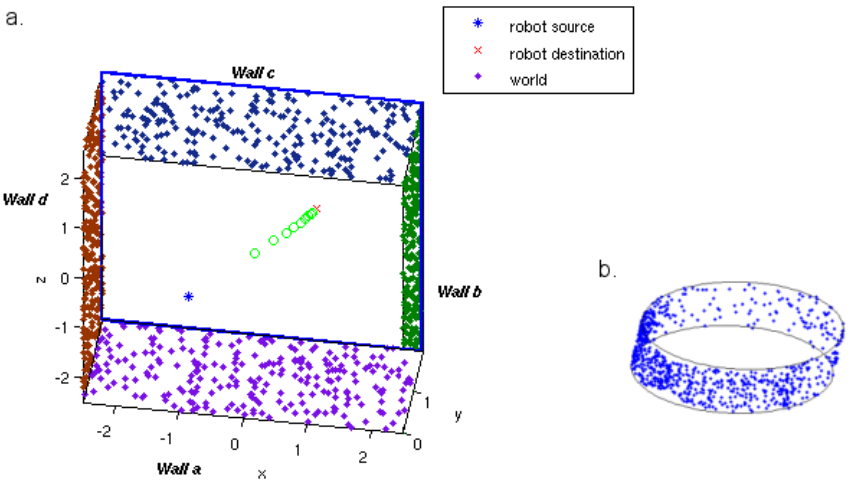


Figure 4.1: The world with uniformly randomly spread feature points.

4.1 The Simulated World

Figure 4.1a shows a world in the simulation. In the simulation I wanted to test how well the *ALV method* worked with a great amount of invariant features, under different conditions. This is comparable to using feature detectors such as DoG or MSER in the real world (see section 2.2).

The robot in the simulation uses an omnidirectional view of the environment, therefore it uses a cylindrical projection of the environment at its current location, an example of such a projection is shown in Figure 4.1b. The radius of this cylinder is 1, therefore the robot does

Table 4.1: The tested parameter values. The values in bold are the default values. The values in italic are the values which have been tested in a second experiment.

Parameter	Tested values
wall length (m)	3, **5**, 10, 15, 20, 30, 50
standard deviation of the Gaussian noise added to the feature position	**0**, 0.001, 0.01, 0.05, 0.1, 0.5, 1
number of features per world	20, 40, 100, 500, **1000**, 5000, 10000
number of added fake features	**0**, 1, 5, 10, 20, 50, *500, 700, 900, 950, 980, 990, 995*
number of removed features	**0**, 1, 5, 10, 20, 50 *500, 700, 900, 950, 980, 990, 995*
maximum random rotation angle of the projected panorama *(rad)*	**0**, $\frac{1}{10}\pi$, $\frac{2}{10}\pi$, $\frac{3}{10}\pi$, ..., π

not have any depth information of the features. Depth could be calculated by using views from two or more locations, but this also adds extra computational work which is not wanted here. The cylindrical projection contains the vectors to all the features, i.e. the location of the features as seen from the robot. These vectors are used to calculate the ALV as in Eqn. 2.4. The next step is to calculate the home vector (Eqn. 2.6).

The process of calculating the home vector is continued until the robot is close enough to its end position. Three other restrictions are set:

1. the maximum number of steps which the robot is allowed to take to reach its goal;
2. not seeing any features more than five times in a row;
3. and not travelling a too long distance.

If one of these criteria is not met, the attempt is considered to be a failure. These criteria are explained in more detail in the next section.

The homing simulation algorithm which was created in Matlab is shown in Appendix A. The parameter `map` in the algorithm contains the location of all objects in the world. This `map` parameter is only used for the simulation to calculate the projection. The `AcquireCylinder` function projects the features that are visible at the current location of the robot to an image plane (an unrolled panorama), thereby using the `map` which contains all the objects in the world.

The simulation environment can contain several objects. These objects are only generated as being feature points. The world can be generated by adding several objects to it and thereby giving their location, height, and the number of features it should contain. In the simulation only the xz-plane is used for moving, so the robot will not change its y-direction. The camera is set 1 m from the ground (at $y = 1.0$). Which features are visible to the robot depends on their closeness and their y-position (height). The robot has a limited view in the height but it has a 360° view in the xz-plane. To make the simulation more realistic, Gaussian noise was added to the positions of the features before projecting them. Also the possibility of features to disappear or fake features to appear was added, i.e. occlusions.

The effects of several parameters have been investigated, which are listed in Table 4.1. The table also shows which values were tested. One parameter was investigated at a time. The parameters which were not investigated during that experiment were set to the default value, which is shown as the bold values in Table 4.1. Several other parameters are kept constant,

World	Description	Wall 1	Wall 2	Wall 3	Wall 4
1	All walls	x	x	x	x
2	2 sided, I	-	-	x	x
3	2 sided, II	x	-	x	-
4	1 sided	-	-	x	-

Table 4.2: The different worlds used in the simulation experiment and per world which walls are present.

such as the robot start and destination position, the shape of the world and the height of the camera. The robot was put on location (-1, 1, -1), where the second coordinate is 1 for the height of the camera. The end location is set to (1, 1, 1); when the robot is within a distance of 0.1 m (meters are used as units in the simulation to make it comparable to the real world) it is counted as having reached its target. The ideal distance is calculated by the Euclidean distance between the start and end location, which is 2.83 m.

Four slightly different worlds were created where the centre of the box is set to (0, 0) on the xz-plane. Figure 4.1 shows the first world (world 1) and the letters for each wall, it also shows the start and end location of the robot. The walls are shown as lines on the xz-plane and the robot can detect them by the features (dots in Figure 4.1). The other worlds are based on the first world, but not all the walls are present. Table 4.2 shows in which world which walls are present. The walls have a height of 2 m and a roughness of 1 mm (i.e. the deviation from a plane wall). The experiments are done with all four worlds. The locations of the features on each wall are randomised (uniformly) before each run of the virtual robot. The locations of the features are not changed during a run, in which the robot has to navigate from the start position to the destination position.

The amount of features is equal for each present wall. In the case of 1000 features per world (the default value), each wall in world 1 contained 250 features, each of the two walls in world 2 and 3 contained 500 features and in world 4 wall 3 contained all 1000 features.

The objects in the worlds (walls in this case) are seen as features by the robot. The robot is not able to see features which are too close ($<$ 0.1 m), because they are not sharp enough. Features that are projected outside the image plane (which is 500 pixels high) are not included in the projection.

4.2 The Experiments

The influence of the variables listed in Table 4.1 are investigated by the experiments which will be discussed in this section. No home vector can be calculated when there are no features present in the current position. In that case the robot should move to a location where there are features visible. At first it should try to use the previous home vector, because this vector points to the destination from its previous position. If the robot does not move too much, this should still be the case. However if the previous home vector has a length of zero then a vector is generated with a random direction and a random magnitude (but with a maximum of 0.5 m). The robot is allowed to use either the previous home vector or a random vector only five times in a row (the constant `MAX_NR_EMPTY_PROJ` in the algorithm in Appendix A) to avoid the robot from only driving randomly. If there are no projected features after these five times, the run is registered as failed.

To prevent the simulation from running endlessly, a limit on the number of steps (iterations) was added. Experimentally was found that 2000 is an appropriate limit. This limit is shown as the constant `MAX_NR_ITERATIONS` in the algorithm. As a last constraint, the robot is allowed to travel at maximum a distance ten times the ideal distance (`MAX_TRAVELED_DIST` in the algorithm).

Table 4.1 shows the different parameters which are used in the simulation and their tested values. The table shows the default values as bold. The noise parameter is the standard deviation of the Gaussian noise which is added to the position of the not yet projected features. The *add fake features* parameter is the amount of fake features which are added to the world before each projection. The positions of the 'fake features' are randomised before each projection. Unseen features can appear suddenly in the real world, which is simulated by adding 'fake features'. Features can also suddenly 'disappear', this is simulated by removing features. Features are randomly removed before each projection, but per run, the same amount of features are removed.

A test which was later done is rotating the projection a random amount (but with a fixed maximum rotation) before the ALV was calculated. The last row of Table 4.1 shows the different maximum angles which were tested.

The robot was expected to perform worse (i.e. have a lower success rate, needing more iterations and having a greater difference with the ideal distance) when the amount of disturbance is higher (a higher standard deviation of noise, adding more fake features, removing more features or having a higher rotation angle).

4.3 Analysis of the results

For each parameter, each of its values listed in Table 4.1 was tested in twenty times for all four worlds thereby using default values for the other parameters. The default value of each parameter is shown in Table 4.1 as bold.

A run is marked as successful if it does not violate one of the previously mentioned three requirements: stay within the limits of 2000 iterations, do not drive a longer distance than ten times the ideal distance and do not use the previous or a random home vector more than five times in a row. From the successful runs the difference between the driven distance and the ideal distance will be compared. At last the number of iterations which the simulated robot used to reach its goal are compared.

The results were not normally distributed as can be seen in Figure 4.2, this has been confirmed by the χ^2 Goodness of Fittest. For this reason the t-test cannot be used to compare the number of iterations and the difference with the ideal distance. Neither bootstrapping made the majority of the data sets normal. Therefore the Wilcoxon Rank Sum test (also called the Mann-Whitney U test; Wilcoxon (1945)) is used. The Rank Sum test compares the ranks of both data sets. It first unites both sets, sorts the united set and then gives each value a rank, depending on its position. Then the sum of the ranks of both sets are compared. In the discussion of the results, $\alpha = 0.05$ will be used. This means that there is a 95% chance of the medians of both data sets to be equal. One of the consequences of using ranks instead of the values themselves is that data sets which are not distributed proportionally (the steps between the data points have different lengths), are treated the same as data sets which are proportionally distributed.

The detailed results are shown in Appendix B in the tables B.1 to B.4.

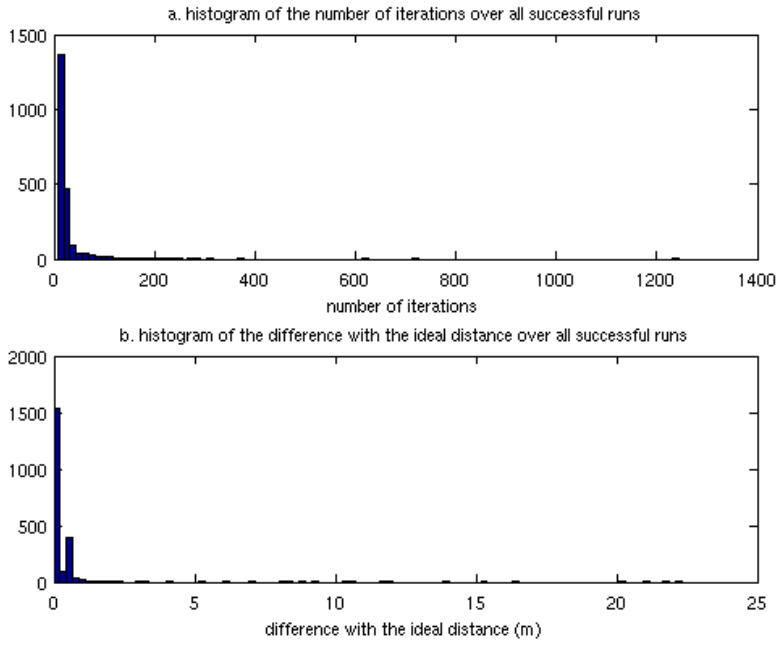


Figure 4.2: Histograms of the number of iterations (a) which the robot used to reach its goal and the difference with the ideal distance (b). Only results from successful runs were used for these histograms. Both histograms contain 100 bins.

4.3.1 Noise in the feature positions

Gaussian noise was added to the feature position to test the robustness to this kind of noise. As can be seen from Table B.1 the success rate rapidly dropped with an increasing noise standard deviation. The robot was able to reach its goal in almost all test-runs within the limitations when the noise had a standard deviation of 0.001 or less. Except for 40% of the runs in world 4 in which the test failed. All the runs completely failed with a noise level of 0.5 and higher. They failed because no features were projected onto the image plane more than five times in a row, except in world 3 where most runs failed because the maximum number of steps had been exceeded. Also for lower noise levels, the most failures were because the maximum distance travelled had been exceeded.

In a second experiment (Table B.3) the distance between the start end endpoint is increased (to 5.66 m) by changing the end and start position to (-2, 1, -2) and (2, 1, 2) respectively. In this second experiment the noise standard deviation was limited up to 0.1, because a noise level of 0.5 and higher resulted in unsuccessful runs only in the first experiment. For a noise level of 0.001 all runs succeeded except 35% in world 4.

More noise required more iterations for the robot to reach its goal (this is only significant for a noise level of 0.01 and higher). Also the travelled distance was higher with more noise (only significant for a noise level of 0.01 and 0.1).

It is difficult to compare the amount of noise to a situation in the real world. In the simulation Gaussian noise is used, because noise in nature generally is normal. The level at which noise is tolerated by the model has been found by using different levels of this noise

(accomplished by changing the standard deviation). This noise in the real world is dependent on a lot of different factors such as the sensor, environment, speed, etc., therefore it could be hard to measure, and it might be better to try the ALV method on a real robot.

4.3.2 Removing features

Occlusions of objects are simulated by randomly removing a number of features. Removing up to 50 of the 1000 features per world did not have significant any effect on the performance. Therefore this experiment was also redone, the results are shown in Table B.3. This table shows that removing 500 features (50% of all features) reduced the success rate only slightly (to 95%) in worlds 2 and 3 and to 50% in world 4. Removing 90% of the features resulted in success rates of 60%, 35%, 40% and 15% respectively for worlds 1 to 4. For 99.0% removed features 15% of runs in world 2 succeeded, 10% in world 3 and none in worlds 1 and 4. For 99.5% removed features none of the runs succeeded.

The number of iterations increased and the travelled distance increased when more features were removed, but only after at least 50% of the features were randomly removed. Although Table B.3 shows some values which are lower (for example a median of 13.5 iterations for 990 removed features) these are not significant because of the low number of successful runs (only 2 for 990 removed features) which were used to calculate this median.

The robustness to randomly removing features could be because of the uniform distribution of the feature points over the walls. Since the features which were removed were also chosen (uniformly) randomly, the resulting set of features visible to the robot is still randomly distributed in the world, therefore the ALV should not be much different.

4.3.3 Adding fake features

Previously occluded objects are simulated by adding 'fake features'. From Table B.1 can clearly be seen that adding up to 50 fake features did not have any significant effect on the performance, number of iterations nor the travelled distance. The distance between the begin and end position was increased to 5.66 m in the second experiment (Table B.3). The number of added fake features is increased to 500 until 995 fake, and to compare the results to the shorter distance, the same amounts of fake features (0 to 50) were also tested. This neither had any significant effect on the performance.

Even adding up to 100,000 fake features (10,000% of the amount of available real features) did not have any significant effect on the performance. The fake features are all uniformly randomly distributed within the *room* which is defined by the walls, with its centre at $(0, 1, 0)$. The mean position of the fake features therefore is the centre of the room, and since the ALV is the mean of all feature positions this is the centre too. Therefore the influence of the fake features is zero to really low in the centre of the room, and this influence decreases when more features are added. The robot however uses the projected features and it is not always in the centre. Therefore the features should not always be projected equally on the panorama even when the robot is close to the centre, because up to 100,000 fake features were added.

4.3.4 Number of features

Having more (reliable) features present in the world increased the performance of the robot (higher success rate, less iterations and a smaller difference with the ideal distance). The success rate was 100% for 100 and more features for world 1, for 500 and more features for

worlds 2 and 3 and 1000 and more features for world 4. The success rate for 20 features per world varied from 80% for world 1 to 50% for world 4. Having more features available increased the amount of information, and therefore it should improve the performance of the robot when it uses the features to navigate.

The number of iterations used by the robot in worlds with 20 and 40 features per world was significantly higher than worlds with 500 or more features. Also the travelled distance was higher for less features per world.

4.3.5 Room size

The length of the walls is only varied in the x direction, the length of the wall in the z direction is kept constant at 5 m. Table B.1 shows that all runs were successful when the wall in the x direction had a length of 5 or 10 m. The performance reduced when the wall in the x direction was longer, until a performance rate between 65% and 35% for the four worlds when the room was 50 m $\times$ 5 m. And almost all of the failures were because the maximum number of iterations had been exceeded. The results were worse for worlds with less walls than the world with all walls.

The number of iterations increased significantly with an increasing wall length, according to the rank sum test. Also the difference with the ideal distance increased significantly with the wall length.

There seemed to be a preference for square rooms, to verify this, another experiment was done. In this new experiment all worlds were square, so the walls in the x and z direction had the same length. For the world with only one wall, this only meant that the robot was further away (half of the wall length) from the wall, since the origin of the coordinate system is at the centre the 'room'. Table B.2 shows the results of this experiment which reveals that the success rate did not decrease with increasing wall length. Except for the smallest room (3 m $\times$ 3 m) all runs in the other rooms succeeded for each world.

The robot performed worse (lower success rate, higher number of iterations and a higher difference with the ideal distance) when one wall is much longer than the other. This can be explained by the way the features are projected. Feature pairs which are at the same distance on the wall, are projected closer to each other on the panorama when they are further away than when they are closer to the robot.

Figure 4.3 shows the projection of two features f_1 and f_2, on two panorama cylinders c_1 and c_2. The difference in the position of the projection of f_2 on both cylinders: $p_1(f_2)$ and $p_2(f_2)$ is really small as can be seen by the horizontal dotted line. Wan et al. (2004) mention the same problem: the vergence angle a is small near the antipodal direction and therefore the uncertainty of the depth is high.

The consequence of a small vergence angle is that the projections of the features on both panoramas (for example feature f_2 on panoramas c_1 and c_2 in Figure 54.3) are very close to each other (shown by the dashed line), therefore the difference between the vectors pointing to the projected features on both panoramas is very small. In the experiments there were about 1000 features equally divided over each present wall (which depended on the world, see Table 4.2). The longest wall was in the x direction, and the robot had to travel in both x and z direction to reach its goal, going from (-1, 1, -1) to (1, 1, 1). Going in the x direction was the biggest problem, because the wall in the x direction was very long, therefore the features which were far away were projected almost at the same place as has been explained by Figure 4.3.

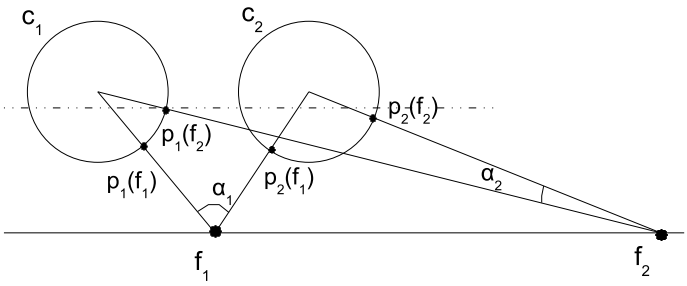


Figure 4.3: The projection of two features on two cylinders c_1 and c_2. The features f_1 and f_2 are shown and their projections on both cylinders: p_1 and p_2.

To calculate the home vector, the average landmark vectors are subtracted, but in the case of the features on the walls in the x-direction, this resulted in very short vectors. The length of the home vectors was measured to verify this. The mean length of the home vector in a room of 5 m $\times$ 5 m was 0.025 m (and a standard deviation of 0.007 m). However when one wall of the room was 50 m and the other 5 m, it resulted in very small home vectors of 10^{-6} m. For this reason a lot of runs failed because the maximum number of iterations (2000) had been exceeded.

4.3.6 Rotation of the panorama projection

Table B.4 shows the results of the experiment in which the panoramas were rotated a random angle. The *maximum* rotation angle is shown in the first column of Table B.4. As can be seen from the table, the success rate drops very fast. A maximum rotation of $\frac{\pi}{10}$ rad resulted in 75%, 70%, 90% and 15% successful runs respectively for worlds 1 to 4. A maximum angle of $\frac{2\pi}{10}$ rad only succeeded 55% of the runs in world 2. For higher maxima all runs failed with some exceptions which probably succeeded because the rotation angle was a random value maximised by the variable.

From this experiment can be concluded that the ALV homing method cannot be used when the orientation of the robot changes randomly, the orientation has to be known. For this reason, the panoramas should be aligned.

4.3.7 Results per world

As explained in the previous section, the worlds differ in the amount of walls and which walls are present. Table 4.2 showed which walls are present in each worlds. It is clear that the robot had the most successful runs in world 1, the world with all walls, and the least in world 4, the world with only one wall.

The results of the different worlds show that the distribution of the features makes a difference. The robot performed best in world 1, in which the features are spread equally over all four walls and worst in world 4 where only one wall is present. The results of worlds 2 and 3, which both have two walls present, are between world 1 and 4 in performance. However the results from the first three worlds are not significantly different. The differences of the results can be explained by the isotropic feature distribution assumption for the same reason I expect that the system performs worse in worlds 2 and 3 than in world 1. The only significant

difference between the results of world 2 and 3 is that the travelled distance is less in world 2 than in world 3. This is probably because world 2 has a wall in both the x and z direction, while world 3 has both walls in the x direction.

4.4 Conclusion

In this chapter I tried to find the robustness of using the average landmark vector to navigate a robot from one point to another. A big advantage of the ALV is its simplicity. The only requirement is that the features have to be extracted from the environment. Observations in the real world however, are almost never the same, therefore the robustness of the method has been tested by adding noise to the feature positions and by randomly adding and removing features.

The experiments discussed in this chapter were done in a simulated environment in worlds with four, two or only one wall present. These walls were visible by the features which were present on the wall. The robot was said to be successful if it found the destination point within the following three limitations: 1. the robot is not allowed to use more than 2000 steps (iterations); 2. the projection of the world should not be empty more than five times in a row (in that case either the previous home vector or a vector with random orientation and length was used); 3. the robot should travel at maximum a distance ten times the Euclidean distance between the start and destination position.

Occlusions were simulated by removing randomly chosen features before every projection. Removing 50% of the features resulted in a mean success rate of 85%. The robustness to removing features is probably explainable by the distribution of the features. In the simulation the features were uniformly distributed over each wall in the world. This distribution does not change after removing a certain amount of randomly chosen features. Experiments should be done in which the features are not distributed uniformly to confirm this.

Adding fake features, which can be thought of as previously occluded objects, resulted in no performance drop at all. Even when 100,000 fake features were added, within the bounds set by the world (width $\times$ length $\times$ height). This might also be explained by the uniform distribution of the fake features. The mean of the uniformly distributed fake features is the centre of the room, therefore it should not have any influence in or near the centre of the room. To confirm this, experiments can be done in which the fake features are not uniformly distributed.

Adding Gaussian noise to the positions of the features before each projection showed that the robot was able to reach its goal 90% of the time within the set limitations, when the noise had a standard deviation of 0.001 or less. A standard deviation of 0.5 or more resulted in only unsuccessful runs. However more information is required about the noise level of the sensor and feature detector in order to compare these results to results in the real world.

Having more (reliable) features present in the world increases the performance of the robot (higher success rate, less iterations and a smaller difference with the ideal distance). For the simulation the threshold of a success rate of 100% for the number of features is between 500 and 1000. Although having only 20 features in the world still resulted in 50% to 80% successful runs. However it has to be taken into account that these runs were without any noise and without any other disturbances.

Random rotations of the projected panorama gave very bad results as expected, therefore the panoramas (or at least the ALVs) should be aligned before calculating the home vector.

This can be accomplished by using a compass for example.

The ALV method implies an equal distance assumption of the landmarks, because no depth is used. Franz et al. (1998) also mentions the isotropic feature distribution, which can explain why the results in world 4 were worse than in the other worlds. The robot used more iterations when more features were removed, which was not significant for all number of removed features but was to be expected since the ALV every time has a different error.

From these experiments can be concluded that using the ALV for visual homing is a robust method. The next step is to try this method on a real robot. In the real world there can be problems with the 'noisiness' of the features, but this depends on which sensor is used and which feature detector.

Chapter 5

Real World Experiments

In the previous chapter it has been shown that the ALV homing method worked in simulation, therefore experiments in the real world have been done to further evaluate the method. The experiments have been done in an office environment in several rooms. This chapter first discusses the experimental setup, then the results and finishes with a discussion and conclusion.

5.1 Robot and environment

Figure 5.1: The Pioneer 2AT robot as used in my experiments. A pan tilt unit is mounted on the robot with a camera on top. It also has a laser range finder, but this was not used in the experiments.

For the experiments a Pioneer 2AT robot (Figure 5.1) is used with a pan tilt unit (PTU 46-70) mounted on it and on top of this PTU a camera (Sony DFW-VL500; focal distance of about 800 pixels). The robot has six bumpers at the front and six at the back. The robot is controlled from a laptop (Dell with an Intel Pentium M 1000MHz, 799 MHz, 760 MB RAM) placed on top of the robot. The programs are run under Microsoft Windows XP and created in Microsoft Visual Studio 6.

The experiments are done in the IIIA in an 'office environment'. All floors are flat and white and also the walls and ceilings are white. The room in which most experiments are

done, the robot laboratory, is about 10.5 m × 11.2 m. The panorama in Figure 2.5 shows the room as seen from the robot. The laboratory is divided into a part with landmarks in which experiments were done with previous navigation methods, a robot soccer field, and three working places.

5.2 Panoramas

The panoramas are made using a camera on a pan tilt unit, Figure 2.5 shows an example of such a panorama. The focal distance f is 800 pixels. Seventeen pictures are taken by turning the PTU 20° between each picture. The panoramas have a length of 5028 pixels ($f \cdot 2\pi = 800 \cdot 2\pi \approx 5027$). In the next step the feature points of each image are extracted, from which three types are tested: DoG, MSER and the landmarks.

The seventeen images created have to be stitched to get the whole panorama. The easiest way to do this is by calculating the amount of overlap between the pictures based on the rotation of the camera and the focal distance:

$$s_{\text{default}} = 2\pi f \frac{\alpha}{2\pi} = f\alpha \tag{5.1}$$

where f is the focal distance of the camera and α the angle (in radians) between two succeeding images. However due to the projection to a cylinder and small errors of the PTU, the difference in horizontal shift between the images can better be calculated, for example by using SIFT points. The SIFT points were chosen because of their stability, the great amount of features which are detected and because the program to extract SIFT features is freely available[1] for research purposes. This program is used as detector of the DoG features and at the same time it is a descriptor and therefore the features can be compared in and between images.

Each image is compared to the next image (and the last is compared to the first). They are compared by matching the features of both images, based on the descriptors. For each feature in one image the most similar in the other image is chosen as the match. This is done by finding the smallest Euclidean distance between the descriptors of the features. This list of matches is filtered by a RANSAC variant. For each match the amount of votes for the translation proposed by that match is counted. If the distance of the translation (dx_i, dy_i) calculated from, for example match 1, has at maximum an Euclidean distance of 5 pixels from another match, (dx_j, dy_j), then this is counted as a vote for match 1. The translation with the most votes is the winner. The actual translation that will be used is the average of the winner and all the translations which voted in favour of it, i.e. have an Euclidean difference of 5 pixels or less.

Three requirements are set for the translation which won:

1. There should be at least ten votes.

2. The distance should not deviate more than 25 pixels in the x direction and not more than 10 in the y direction from s_{default} (Eqn. 5.1). There is an exception for the match between the last and first image which can suffer from accumulative errors, therefore a maximum deviation of 50 pixels in the x direction and 30 in the y direction is allowed. A second problem is that the used pan tilt unit is not capable of rotating 360°, therefore there is much more space between those two images.

[1] The free SIFT detector was downloaded from `http://www.cs.ubc.ca/~lowe/keypoints/`.

3. The ratio between the distance of the closest and second closest pair should be smaller than 0.8 ("closest distance"/"second closest distance" < 0.8).

When these requirements are not met then the default value (s_{default}) from Eqn. 5.1 is used.

The next step is to project the features onto a cylinder (with radius 1). Only the x location of the features in the image is used, and since the image represents a 360° view it can be seen as the cylindrical angle coordinate. For the calculation of the ALV however, the Cartesian coordinates are required, which can be calculated using the next equations:

$$\alpha = 2\pi x/l + \pi \tag{5.2}$$

$$(x, y) = (r \sin \alpha, r \cos \alpha) \tag{5.3}$$

where x is the horizontal position of the feature, l the length of the panorama and r the radius of the cylinder, which in this case is 1.

5.3 Real-time use

To use the homing method real-time it should be able to drive. And when driving it should not collide into objects therefore some extra behaviours have to be added. When a working alignment method is available then the robot can perform online real-time homing.

5.3.1 Driving

The *ActivMedia Pioneer 2AT* has four wheels without a steering mechanism, therefore it turns by skid steering, i.e. turning the wheels at one side more than the other. When using the homing method online, the robot first turns in the correct direction before driving, i.e. rotating on place. When the robot turns and drives at the same time it will deviate too much from the home direction, especially when the angle to turn is big.

A minimum speed is required to overcome the resistance to turn the robot. In order to turn, the angle to turn also has to be big enough because the robot needs to accelerate and de-accelerate. Of course the way the robot turns depends on more factors such as: the wheels, the speed of turning, the turning angle, the floor, the battery level, etc.

Six tests were done in the IIIA robot laboratory to measure the turning error. The floor in this room is the same as in all experiments, it is a flat tiled floor.The turn speed is 5 deg/s and the angle to turn is 90°; three turns are clockwise and three anti-clockwise. The mean error was 2.90° and the standard deviation 1.87°. Although this error also depends on the turn angle, I still think that this will not cause problems for real-time use of the homing method.

5.3.2 Collision detection

In section 3.1.5 the work of Hafner (2003) was discussed, in which she showed that obstacle avoidance behaviour emerges from the ALV homing method. She however used only a few landmarks in her experiment, in contrast in my experiment where many more features (up to 1000 and more) are used. This might decrease the obstacle avoidance behaviour.

When doing real-time homing the bumpers of the robot could be used to detect a collision to be sure that the robot does not continue pushing against an object. The *Pioneer 2AT* has six bump detectors at the front and six back. When a collision occurs, i.e. one of the bumpers is activated, then the robot drives towards the opposite direction. An even better method would be to avoid collisions by detecting the objects with for example a sonar or laser sensor.

5.4 Alignment

As already mentioned before (in Chapter 4), the ALV homing technique requires the panoramas at the current and the target locations to be aligned; and because in normal situations the robot never only drives in one direction, the robot will not have a constant orientation.

The same technique which is used to stitch the images could also be used here, where the distance of the feature descriptors is compared. It searches the nearest neighbours of each feature and uses this information to calculate the amount of shift.

Another method is to use the odometry data of the robot. A known problem of this method however is that wheel skidding and noise creates a cumulative error in the position given by the Pioneer robot; and since the orientation is measured by using wheel information, this also suffers from that error.

Franz et al. (1998) tried to estimate the orientation by shifting the snapshot and current view until a minimal image distance is reached. This method however only worked well near the goal. The method which I tried is using histograms of the horizontal position of the DoG points of the panoramas.

5.4.1 Histograms

The histograms are made by dividing the panorama in n bins (columns of equal width). The amount of features is counted for each bin. This is done for both panoramas which have to be aligned. Then for every possible shift (at maximum $n-1$), the sum of the squared difference is calculated. The shift s for which this squared difference is the minimum, should be the difference in bins between the two panoramas:

$$s_{\min} = \text{argmin}_{s \in [0..n-1]} \sum_{i=0}^{n-1} (h_2[(i+s) \bmod n] - h_1[i])^2 \tag{5.4}$$

where h_1 and h_2 are the histograms of the two panoramas. Every possible shift is checked (0 to $n-1$) and from these values the shift with the lowest difference, $s_{\min}$ bins, should be the rotational difference between the two panoramas, which in degrees is $360 * s_{\min}/n$. This method is still rather simple because the difference between two neighbouring bins is very strict, Gaussian shaped bins could be used to reduce this effect.

Experiments This alignment method was tested in the laboratory by manually rotating the robot 90° each time; the lines of the floor tiles were used to measure the square angle. Figure 5.2 shows the DoG points of the four panoramas. This Figure shows clearly that the points, especially looking at the white spot (which is a white wall), shift a quarter of the image size. Figure 5.3 shows the histogram (100 bins) of the panoramas in Figure 5.2. The error[2] was $3.60° \pm 2.28$ for 100 bins and $2.82° \pm 2.27$ for 1000 bins. As a second test the histograms were smoothed by a Gaussian kernel (10 bins and $\sigma = 1$) which resulted in an error of $3.60° \pm 3.94$ for 100 bins and $3.06° \pm 2.41$ for 1000 bins. From these results no significant difference can be found, therefore for the alignment 100 bins without smoothing can be used. A second note is that there is still a small human error in positioning the robot, which makes the difference even less significant.

[2] I write "mean"±"standard deviation".

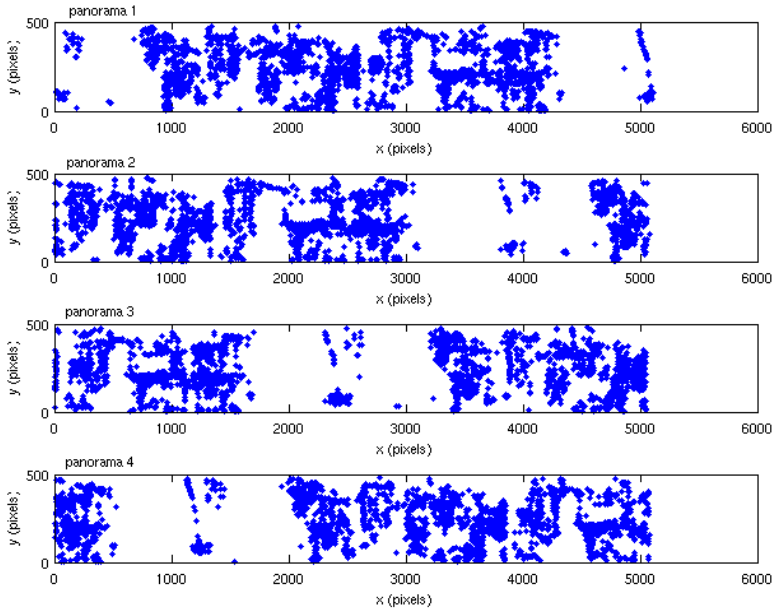


Figure 5.2: These pictures show the DoG points of panoramas 1 up to 4 (each with a width of 5028 pixels). The panoramas were created with 90° clockwise difference in orientation.

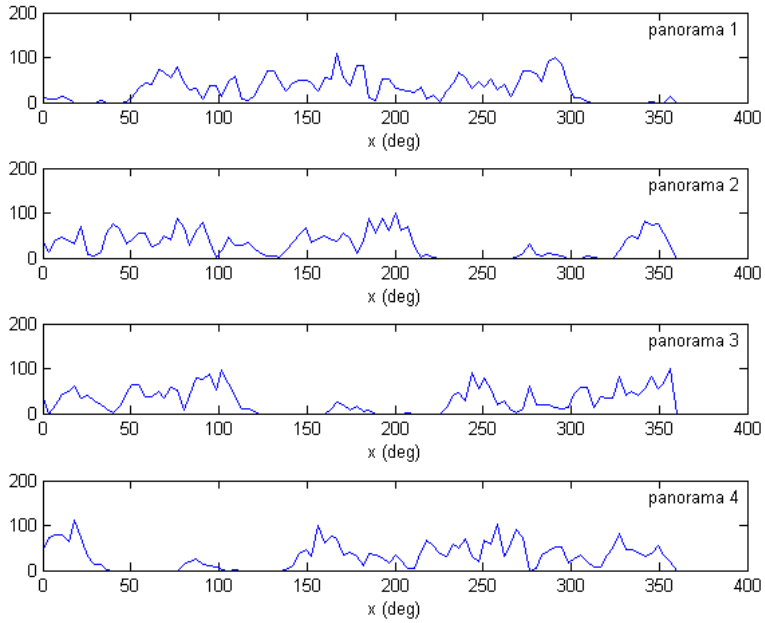


Figure 5.3: These are the histograms of the panoramas in Figure 5.2 with 100 bins.

Since the histogram alignment method works 'on place', the method should be tested with two panoramas not made on the same place, however still in the same room. Figure 5.6 shows the places on which panoramas were created in the robot laboratory; for all these combinations the alignment difference was calculated by the histogram algorithm. Figure 5.4 shows a smoothed plot of the error in the alignment for each distance. At a distance of around 100 cm the error was already $8.1° \pm 19.7$ and at 200 cm the error was $81.7° \pm 59.2$; for higher distances the results are more or less the same or worse.

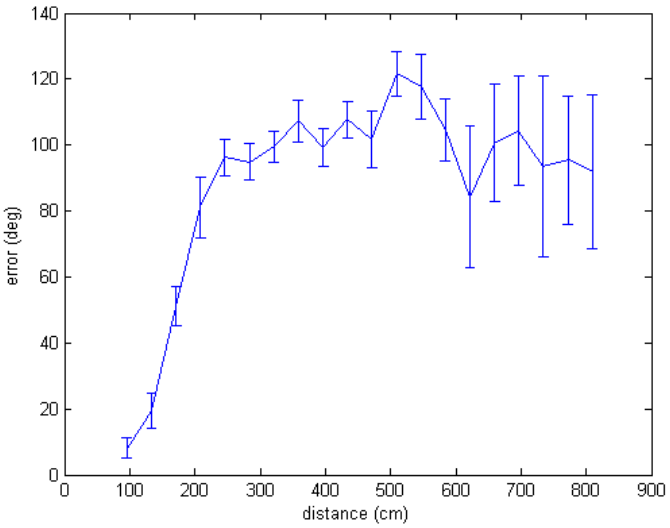


Figure 5.4: The plot shows the alignment error between two panorama against the distance between them. The data from the robot laboratory (see Figure 5.6 for the map) was used to calculate this error. Twenty bins for the distance were created in which the mean and standard error were calculated, which are shown in the plot.

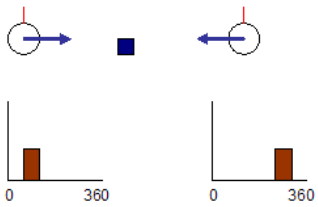


Figure 5.5: A worst case scenario for using the histogram alignment method. The blue square is the only landmark. The two circles are two positions at which a panorama was taken. The arrow is the ALV and the red line pointing upwards is the orientation of the robot. The histograms of the panoramas are shown below them.

Worst Case Scenario Figure 5.5 shows a worst case scenario for using the histogram alignment method. The blue square is the only landmark; the two circles are two positions at

which a panorama was taken. The arrow is the ALV and the red line is the orientation of the robot.

The average landmark in both cases is between the two panoramas, and therefore the histogram method will return a difference of 180°; the panoramas however already are aligned. The error in this method will of course degrade the performance of the homing method itself. In reality this worst case scenario is very unlikely, since normally the features are more distributed around the robot in the whole room.

Like the Franz et al. (1998), who also tested a shift technique to estimate the orientation, the histogram alignment results were only good when both panoramas were close. Instead of detecting the alignment (i.e. difference in orientation) from the panoramas, an external device such as a compass could be used, however no such device was available.

Since the goal of the experiments in this chapter is to test the ALV homing technique, and because the bad results of the histogram alignment method will degrade the results drastically, no alignment method is used. Therefore I chose to use only panoramas with the same orientation. For the same reason no real-time homing can be used.

5.5 Landmarks

Six landmarks are present in the *robot laboratory* (see Figure 2.3) and since the code which recognises the landmarks already was available (as discussed in section 2.2.3), these landmarks were also used in the experiments. An advantage of the landmarks is that the recognition is robust; a disadvantage however is that there are only a few. The occlusion of a few landmarks will have a relatively larger impact than when there are numerous features available, like the DoG and MSER detectors generate. However due to the stability I suspect that the ALV homing method performs better using these landmarks than using DoG or MSER points. An existing algorithm at the IIIA is used to extract the position of the landmarks from the images.

5.6 Experiments

In these experiments the positions in the room at which the robot created a panorama are measured. The orientation of the robot is kept constant for each of the three experiment rooms so that the results are not dependent on an alignment method. With the measured locations of the robot the correct (including measurement errors) home angle and distance can be calculated. These values are used to compute the error of the results with the ALV homing method.

Three types of landmarks/features are used: 1) DoG points; 2) MSER points; and 3) the landmarks. The landmarks are only used in the robot laboratory because they are only present in that place.

For each room a scaled map of the room is shown, (see figures 5.6, 5.9 and 5.10). The maps contain the biggest objects in the room such as desks (shown as rectangles) and it shows the locations at which a panorama was created. The objects are only shown to give a rough impression of the environment. The squares in Figure 5.6 show the landmarks which each have a number. The circles show the positions at which a panorama was taken and they are identified by a number. From each panorama a red line is shown which is the home vector and should point to the home location which is shown as a red circle.

Like in the simulation only the direction of a feature is known, and not its distance, therefore the home vector will not contain distance information either. The home angle calculated by the homing method is compared to the 'correct' home angle which is calculated by geometry.

$$\theta_{\text{difference}}(h_{\text{homing}}, h_{\text{correct}}) = \min\left(|h_{\text{correct}} - h_{\text{homing}}|; 360 - |h_{\text{correct}} - h_{\text{homing}}|\right) \tag{5.5}$$

All angles are in degrees and counter-clockwise; h_{correct} is the homing direction calculated by using the positions (geometry), and h_{homing} is computed by the homing method. To find out how well the method works for each room and each type of features, all the panorama positions per data set are used. For each data set (the *square room*, the *robot laboratory*, and a *corridor*) all the locations at which a panorama was created are used to calculate the home vector to each of the other locations. From the error calculated with Eqn. 5.5 for each combination of panorama pair in one room, the mean, median, standard deviation and a score are calculated. The score is between 0 and 1 where 1 is best; it is calculated by using the proportion of the maximum error:

$$s = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1; i \neq j}^{n} \theta_{\text{difference}}(\text{home}(P_i, P_j), \angle(P_i, P_j))}{180n(n-1)} \tag{5.6}$$

where n is the number of panoramas in the set and P the set of panoramas. The divisor is the sum of the difference of the home angle calculated by the ALV homing method and by geometry. This difference, i.e. error, is calculated for each panorama pair, which in total are $n(n-1)$ pairs. The sum of errors is divided by that factor to get an average, and to normalise the score between 0 and 1 it is also divided by the maximum error which is 180°.

In the following sections the results of the data sets of the three rooms will be discussed. A list of all results is shown in Appendix C; Table C.1 and Table C.2 show the results sorted by data set and Table C.3 shows all results sorted by score.

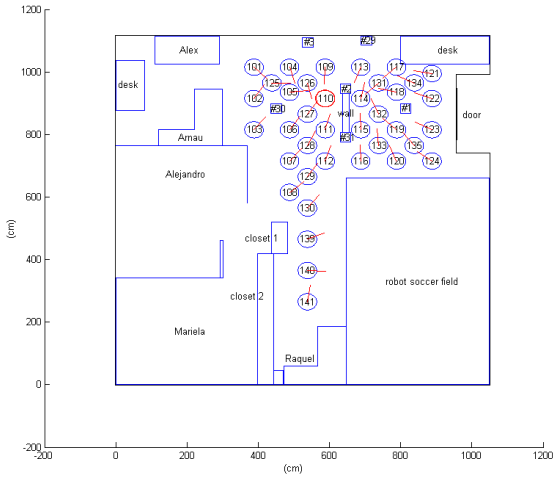
5.7 Results

The experiments were done in three different rooms of different sizes: the *robot laboratory* (10.5 m × 11.2 m), the *square room* (4.0 m × 3.4 m) and the *corridor* (2.2 m × 22.5 m).

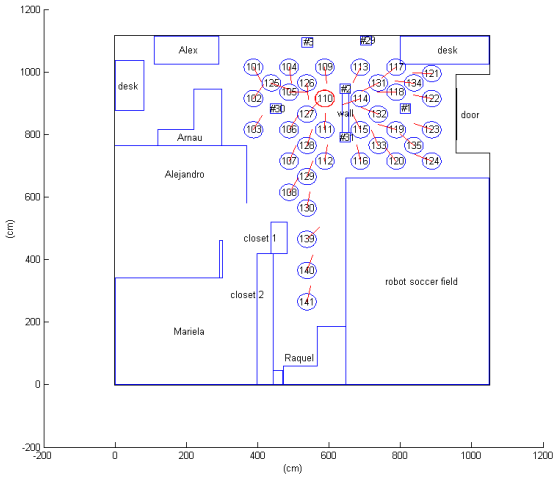
When calculating the home vector between two points, for example a and b, the home vector from a to b will always point in opposite direction (180° difference) of the home vector from b to a. This can easily be deducted from the ALV homing formula (Eqn. 2.6): since $a - b = -(b - a)$. This means that these are dependent values and therefore only one of them is used in the analysis.

5.7.1 Robot laboratory

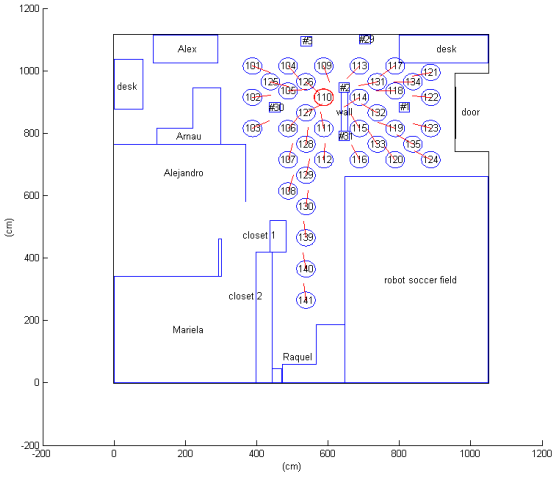
Most panoramas, 38 in total, were made in the *robot laboratory*, a big room of about 10.5 m × 11.2 m. Only the half is really used for this experiment because the other part is filled with working places and the robot soccer field as can be seen in Figure 5.6. An example of a panorama in this room is shown in Figure 2.5. Figure 5.6 shows the home vectors when the home is panorama 110, and it can be seen that the best results are obtained by using landmarks (Figure 5.6c). The six artificial landmarks[3] are shown as numbered rectangles on the map.



(a) Homing using DoG points.

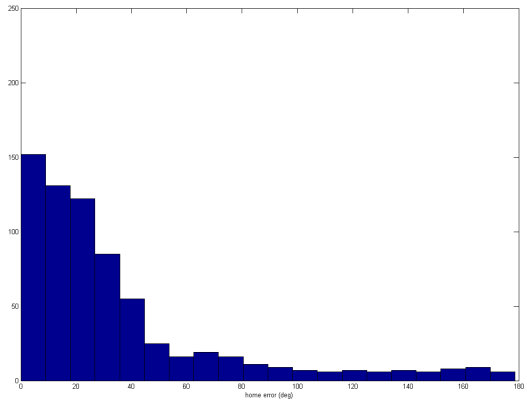


(b) Homing using MSER points.

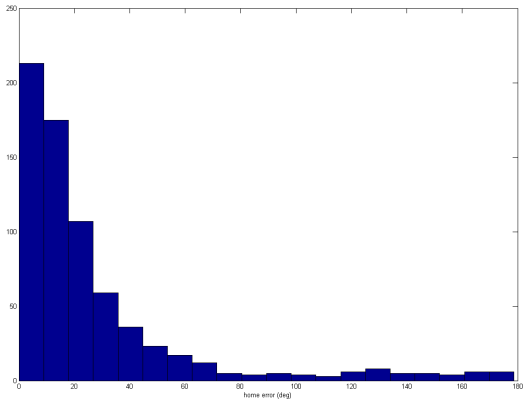


(c) Homing using landmarks.

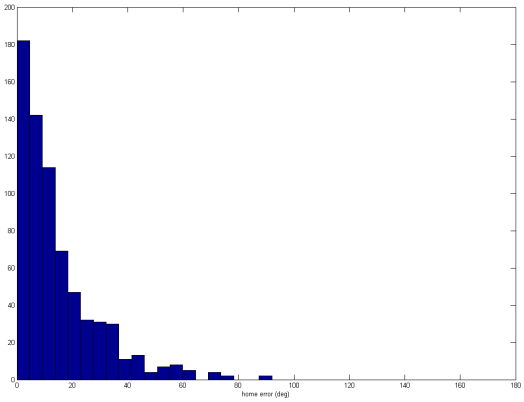
Figure 5.6: Homing to panorama 110 in the *robot laboratory* using DoG points (a), MSER points (b) and the landmarks (c).



(a) Home error using DoG points.



(b) Home error using MSER points.



(c) Home error using landmarks.

Figure 5.7: The histograms (with 20 bins) show the error of the home angle in the *robot laboratory*, for DoG points (a), MSER points (b) and the landmarks (c).

Figure 5.7 shows the distribution of the difference between the calculated and the correct home angle, i.e. the error. The home vectors have an error equal to or lower than 90° in 89.3% of the cases when DoG points were used, 92.6% for MSER points and 99.6% when the landmarks were used. An error of 10° or less was obtained in 22.6% of the cases for DoG points, 32.7% for MSER points and 64.3% for landmarks.

Table 5.1 shows the results per type of feature point used. The homing errors for the three methods are all significantly different ($p<0.001$) according to the rank sum test, and the t-test after bootstrapping ($n = 1000$). From this can be concluded that the homing method worked best with the artificial landmarks, as expected, and worst with DoG points.

[3] I use the word "landmarks" to refer to the artificial landmarks and not to the (MSER or DoG) feature points.

	DoG	MSER	Landmarks
Mean error	35.60°	27.84°	14.88°
Median error	22.85°	16.03°	10.17°
Standard deviation	36.67°	35.51°	14.86°
Score	0.8022	0.8454	0.9173
Best home	117	117	110

Table 5.1: The homing error using the panoramas from the *robot laboratory*. The 'best home' field shows the number of the panorama (see Figure 5.6 for the numbers in the *robot laboratory*), which when chosen as home, resulted in the lowest average error.

5.7.2 Square room

The square room is 4.0 m × 3.4 m big, and is a part of the robot laboratory, but it is isolated for the experiment. Figure 5.8 shows a panorama made in this room; its map is shown in Figure 5.9, this is the part were Mariela worked which is visible in the laboratory map (Figure 5.6).

Figure 5.8: Panorama 137 from the *square room*.

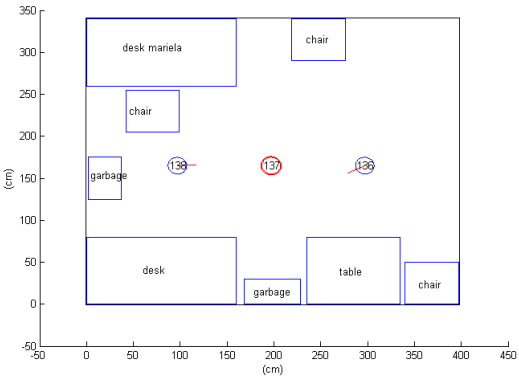
Figure 5.9 shows the map of the room and the home vectors to panorama 137. Table 5.2 shows the statistics of the homing method using DoG and MSER. When MSER points were used the error was lower than when DoG points were used, but this is not significant (confirmed by the rank sum test and the t-test) and it must be noted that only three panoramas were created in this room.

	DoG	MSER
Mean error	13.78°	9.65°
Median error	12.00°	12.03°
Standard deviation	11.31°	7.84°
Score	0.9234	0.9464
Best home	138	138

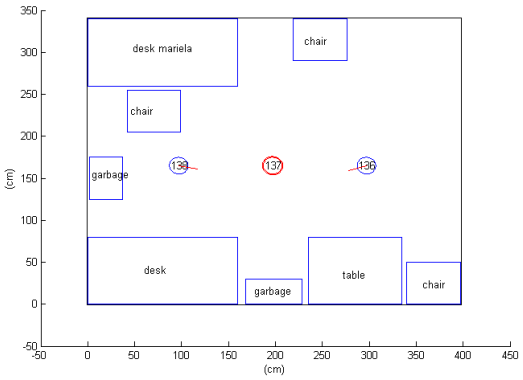
Table 5.2: The error of the homing method using the panoramas which were made in the *square room*.

5.7.3 Corridor

Although the simulation showed that the ALV homing method works better in 'square' rooms, I wanted to find out what the impact of a not square room in the real world would be on the method. A corridor was chosen for that reason as last experiment room. It is a normal corridor and the part in which the robot drove is 2.2 m wide and about 22.5 m long. Figure



(a) Homing using DoG points.

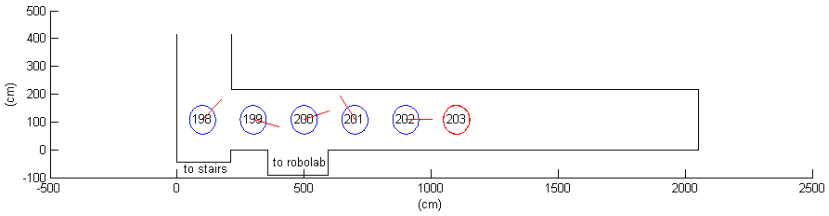


(b) Homing using MSER points.

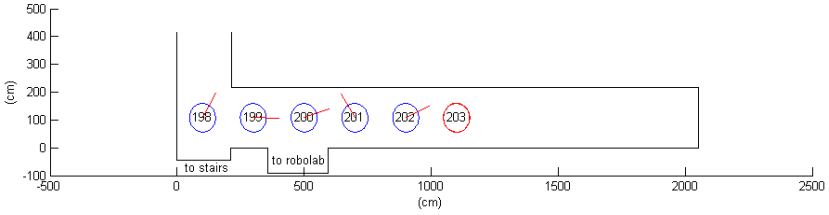
Figure 5.9: Homing to panorama 137 in the *square room* using DoG points (a) and MSER points (b).

5.10 shows the map of the corridor, and it shows that at the left the corridor is going up on the map, but this part was not used in the experiment and therefore not shown. Figure 5.12 shows the panoramas made in the *corridor*.

In Figure 5.10 the home vectors to panorama 203 are shown. An error of 90° or less was obtained in 73.3% of the cases for both feature types, an error of 10° or less was only obtained in one case (6.7%). Table 5.3 shows the average error of this data set; the differences between the results with DoG and MSER are not significant.



(a) Homing using DoG points.



(b) Homing using MSER points.

Figure 5.10: Homing to panorama 203 in the *corridor* using (a) DoG points and (b) MSER points.

	DoG	MSER
Mean error	56.26°	52.67°
Median error	44.58°	35.71°
Standard deviation	43.64°	44.90°
Score	0.6874	0.7074
Best home	203	200

Table 5.3: The average error of the homing method in the *corridor* for the different feature types.

5.8 Vardy's Image Database

As a last test I used the image database of Vardy[4] which he discussed and used in his thesis (Vardy (2005)).

Vardy (2005) did experiments with several homing techniques. He created an image database with images from the hall and the robot laboratory of the Bielefeld University. He created six data sets of the laboratory and two of the hall, all under slightly different conditions (such as the amount of light and added objects; see Appendix D for more details). The images were created in a grid; in the robotic laboratory 10×17 images were created with 30 cm between them (horizontally and vertically); in the hall 10×21 images were created per data set with 50 cm between them. The images were created with an *ImagingSource DFK 4303* camera which pointed towards an hyperbolic mirror. Figure 5.11 shows a panorama from the *original* data set (created in the robot laboratory) and one panorama from the *hall1* data set.

Although the panorama images were made in a different way as the panorama images mentioned before, they still can be used to perform homing. It is even slightly easier because the features do not have to be projected on a cylinder, but some pre-processing is still required.

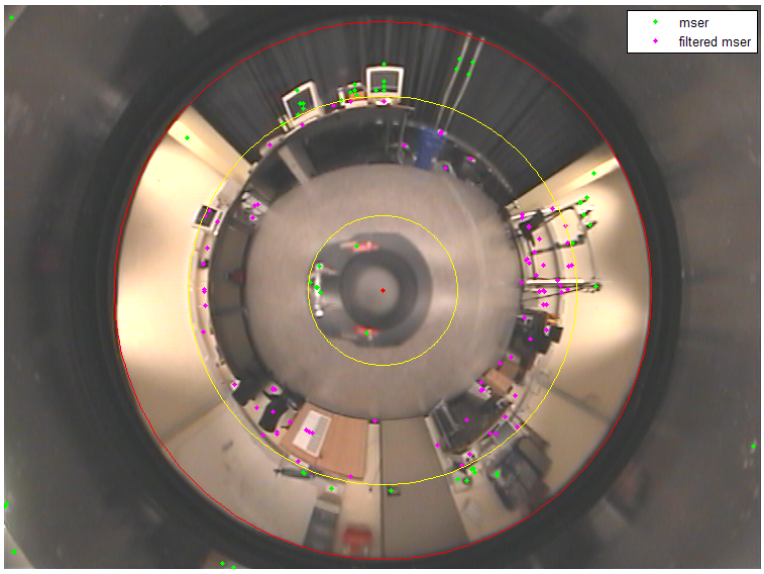
First of all the features should be extracted from the image, this is shown in Figure 5.11. As can be seen in this figure the image also contains non relevant parts which lay outside the mirror. Like in previous researches the used view is decreased to a certain amount of degrees above and below the horizon, which is the line between the centre of the spherical mirror and the outer circle of the mirror. Only features which fall in this area are used for the homing method.

The vector of a feature has its origin in the image centre (shown as the red dot in Figure 5.11) and points to the feature point. These vectors have to be normalised to 1 before calculating the ALV, because the length of the vectors only shows the distance in pixels on the image. After this the ALVs and the home vector can be calculated as described in section 2.4.

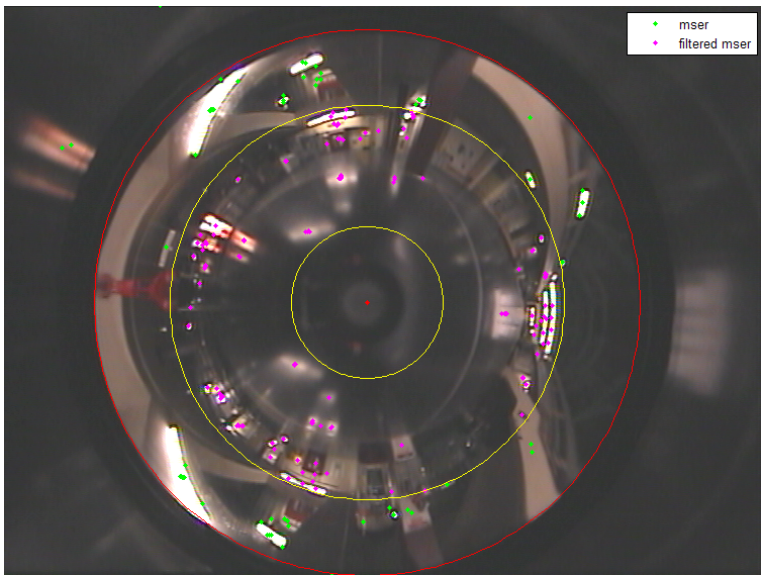
Table C.2 in Appendix C shows a list of the results of the homing experiments with all Vardy's data sets. For each data set of Vardy the column type shows the degrees under and above the horizon which were used. Table C.3 shows the results with all data sets sorted by score, including the data sets discussed earlier in this chapter.

As can be seen from the table the scores vary from 0.85 to 0.3 and the home angle error from $28.2° \pm 27.6$ to $126.0° \pm 43.3$. The results are worse than the results with the previously discussed data sets, but it must be noticed that Vardy's data sets contain more samples.

[4] Vardy's *Panoramic Image Database* is available at `http://www.ti.uni-bielefeld.de/html/research/avardy/index.html`.



(a) Image 4_8 from Vardy's image set *original*.



(b) Image 4_9 from Vardy's image set *hall1*.

Figure 5.11: Two panoramas from Vardy's image database. The outer red circle shows the border of the parabolic mirror, the two inner yellow circles show the 20° line above and below the horizon. The points show the location of the MSER features; the filtered features are the ones between the yellow circles.

From the results can be seen that a wider view angle gives better results. When MSER points were used the use of a view angle of 15° (above and below the horizon) worked significantly better than a lower angle ($p<0.001$, t-test and rank sum test). For all data sets except for *doorlit* and *hall1* the best view angle was 20°. This is also the case when DoG points were used, except for the data sets *day*, *hall2* and *screen*. In the data set *day* the difference was not significant enough; using a view angle of 5° had the best results in the sets *hall2* ($p<0.001$, rank sum test) and *screen* ($p<0.05$, rank sum test) when DoG points were used.

It is also clear from Table C.3 that the performance is better when using the MSER detector than the DoG detector. This difference is significant for all data sets with a view of more than 5° above and below the horizon (using the t-test and rank sum test; $p<0.001$). It also can be seen from the table that the best of the IIIA sets are all above the data sets of Vardy, however this is only significant for the *robot laboratory*. Comparing the results of Vardy's data set with the results of the IIIA data set is difficult because of several reasons. First of all there are two big differences between them: the environments and the way the panoramas are made. The rooms could be assumed to be quite similar since they both are flat 'office like' with several desks, chairs and computers, but the landmarks present in the *robot laboratory* are not present in Vardy's rooms for example.

5.9 Discussion

When the results of the different rooms are compared then it can be seen that the ALV homing method worked better in both the *square room* and the *robot laboratory* than the *corridor*. There is no significant difference between the *square room* and the *robot laboratory*. This difference might be explained by the previously found conclusion, in the simulated experiment (Chapter 4), that the method works better in more square rooms. This is due to the equal distance assumption.There is however another reason why the homing method had worse results in the *corridor*.

Corridor result The panoramas created in the *corridor* (Figure 5.12) show that there are several 'disturbing' factors on which numerous MSER feature points were found. Panorama 198 is the only panorama with a corridor at the left, therefore the MSER detector finds considerable more features at the left than the other panoramas which have a white wall at the left. In panoramas 200 and 201 a door with blinds is visible, and the MSER detector also found a great amount of features on these blinds; in panoramas 199 and 200 the robot laboratory is visible through an open door which again has many interesting points. Figure 5.10a and 5.10b confirm this, because here the home direction from panoramas 199, 200 and 202 to 203 were good, but from panoramas 198 and 201 really bad. The reason for this is that in panoramas 198 and 201 the most MSER points are focused on one side, in panorama 198 at the left corridor and in panorama 201 at the left door blinds. In the other panoramas the features are more or less spread the same as in the target, panorama 203. The good results of the other panoramas might be explained by the lack of features due to the white walls. Although only the MSER detector was mentioned here, the DoG detector generated even more feature points, but with more or less a same distribution.

In Table C.3 can be seen that the best *corridor* of the IIIA data sets is at rank 25, but this is below the best of the data sets *robot lab* and *square room*.

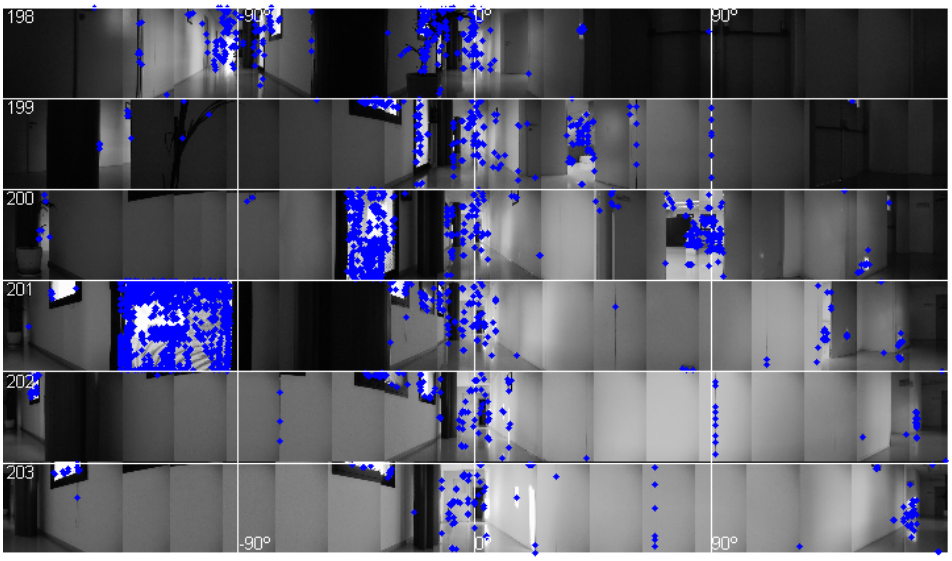


Figure 5.12: All the panoramas made in the corridor. The blue dots are MSER points.

Upper and lower part In an attempt to improve the results the view of the image was limited to only the lower half of panorama image. This part contains objects which are closer to the robot and therefore decrease the size of the visible world, for this reason a room may look more square. Experiments were done using only the features at the lower half of the panorama image. The results discussed in this section are also listed in tables C.1 and C.3 in the appendix.

In the *robot laboratory* using only the lower half of the panorama resulted in a lower error than using all features of the panorama ($p<0.001$ with the t-test and the rank sum test for both DoG and MSER). For the other rooms there was no significant difference in performance. Also here the best results were when the MSER detector was used ($p<0.005$ for the *robot laboratory* and *corridor*) except for the *square room* where DoG was the best detector ($p<0.001$, rank sum test).

Also the use of only the upper half part of the panorama was tested, but these results were significantly worse than using the whole panorama for the *robot laboratory* ($p<0.001$, t-test and rank sum test). There was no significant difference in the *square room* and *corridor*.

Depth When only the position of the features on the panorama are used then only the direction of the home vector can be used, but when the distance to the features is available, then a more precise distance to the home can be calculated. The landmark homing method already provides us with a distance to each landmark, therefore the distance to the home could be calculated more precisely.

5.10 Conclusion

Also with the real robot experiments the ALV homing method had a very positive outcome. The best results were obtained with the panoramas from the *square room*, however this is not significant because only three panoramas were made in this room. The results from the *corridor* were worst, as expected. In the simulation already was found that the performance of the homing method is better in square rooms than in rooms with big differences in the width and length. The problem of long rooms such as a corridor is that the projections of the features onto a panorama are closer to each other the further they are away from the robot (see Figure 4.3).

Looking at the difference in performance of using DoG and MSER points it can be concluded that the use of MSER features significantly outperforms the use of DoG points. The artificial landmarks in the *robot laboratory* were used to find out how well the method worked in comparison with invariant features. The results with the artificial landmarks were significantly better than using invariant features, the error was about 7° less than using MSER points (only the lower half of the panorama).

Normally one should expect the homing method to work worse when the distance between the current location and the home is lower, but this relation could not be found. This might be because the room is too small or because objects occlude a big part of the field. Further work would be needed if there is any relation between the distance and error.

An attempt to improve the results was done by trying to make the rooms, such as the corridor, more square by only using the lower half of the panorama, because then the closer objects are more prominent. This however had no significant improvement in the *corridor*, and neither in the *square room*. Only in the *robot laboratory* there was a significant lower error ($p<0.001$).

The images of Vardy's data sets (Vardy (2005)) were also used to test the ALV homing method. The panoramas are made by a camera pointing to a parabolic mirror. The advantage of this is that the panorama is made in one step, instead of several steps for the method I used. Secondly the resolution is lower, which is good for the processing speed but it may reduce the performance. In practice however the performance of these sets was not much worse than the results of the IIIA data sets. From these images also SIFT and MSER points are extracted which are used to calculate the ALV. It was found that using almost the whole image (20° above and below the horizon) resulted in the best performance.

The scores (with 1 being best and 0 begin worst) of the IIIA data sets varied from 0.67 to 0.96, whereas the results of Vardy's data sets varied from 0.30 to 0.85 (see Table C.3). Looking at the best parameters however, such as using the lower half of the panorama for the IIIA data sets and using a view angle of 20° above and below the horizon of Vardy's data, then the scores of the IIIA data sets vary from 0.73 to 0.96 and the scores of Vardy's data sets from 0.67 to 0.85. This shows that the method performs almost as well in the different rooms and with the different types of panoramas, and thereby confirms the robustness of the method.

Finally some comparison to other work can be made, however in most works other error measurements are used such as the distance at which it arrives to its home. In this work however no such experiments have been done yet. Hafner (2001) also did experiments in an office environment in a grid. After off-line learning the average error was smaller than 90° in 92% of the cases and smaller than 45° in more than 69%. This is comparable to the results in the *robot laboratory* for the DoG features, and the results for using MSER feature points

were even better.

The experiments by Franz et al. (1998) were done in a 118 cm $\times$ 110 cm environment but the catchment area was relatively smaller than the catchment area of the IIIA data sets when looking at the maps of the *square room* and *robot laboratory*. Their algorithm performed robustly up to an average distance of 15 cm. They also mention experiments done in an office environment in which the algorithm performed robustly until about 2 m.

Chapter 6

Conclusion and Future Work

In this work I have proposed a method to complement the global localization system of Ramisa (2006), where feature regions computed in a panoramic image were used to characterise a room. Specifically here the problem of having various valid hypotheses is addressed.

When the robot has several likely hypotheses about its current location (i.e. room) then homing can be used to return to the position where the most likely panorama from the database was made. If the hypothesis with the highest probability was correct, then the panorama at that location should be more similar to the panorama from the database, making it the significantly best hypothesis. In the case that an incorrect panorama was chosen from the database, then the same steps should be taken as before in order to find out in which room the robot is.

Although there are several methods to do homing such as the 1D method of Hong et al. (1991), warping (Franz et al. (1998)) or snapshots (Lambrinos et al. (2000)), I used the *ALV homing* method (Lambrinos et al. (1998, 2000)) mainly because of its simplicity.

In order to evaluate the proposed method, I first conducted experiments using a simulated environment and later tested it in a real world scenario. The real world experiments were done with panoramas created in three different rooms. The locations at which the robot made the panoramas were measured manually and used to calculate the real homing directions, which were used to verify the homing method. The orientation of the robot was kept constant because no good working alignment method was available.

The panoramas were created using a normal camera on a pan tilt unit which rotated to get images from different directions. These images were combined to create a panorama. From these images features were extracted which were used by the homing method. Two types of invariant feature detectors were tested: Lowe's Difference of Gaussians extrema (DoG, Lowe (1999, 2004)) and Maximally Stable Extremal Regions (MSER, Matas et al. (2002)). Only the horizontal locations of the features were used, i.e. the cylindrical angle, and not height, nor depth. The ALV is calculated by getting the average location of all the feature positions.

ALV homing was found to be a good working method, and furthermore it is a simple method and therefore it is computationally cheap and relatively fast. It is small in the memory requirements because only the ALV and (relative) orientation of the home location have to be stored. However the method performed worse in rooms where the proportion of the room width and length differ greatly. This has been explained by the way the features are projected on the panorama and by the *equal distance assumption* of the features (Franz et al. (1998)).

Vardy (2005) discusses biologically based homing methods in his thesis and also did experiments with several of them. In his work he used panoramas that were made with a camera pointed to a parabolic mirror. The advantage of creating a panorama like this is the speed of creation. Whereas I first had to retrieve images from several angles and then stitch them to create a high resolution panorama. In order to compare both methods of panorama acquisition I performed additional experiments using Vardy's data sets.

When the results of IIIA data sets and Vardy's data sets are compared we can see that the ALV homing method performs slightly better on the IIIA data sets, but the difference is very low. There might be several reasons to explain this, such as the difference in resolution, the camera or the environment. For these reasons it cannot be concluded that having a panorama with a higher resolution made by rotating a camera is much better for ALV homing than using a camera and a parabolic mirror, in contrary the faster creation of the parabolic panoramas might be more important than the slightly better performance.

The robustness of the whole method in this case is also thanks to the feature detectors MSER and DoG, whereby the use of MSER points was significantly best. Also Mikolajczyk et al. (2005) confirmed that MSER is one of the better feature detectors. An important advantage of using invariant features is that no artificial landmarks are required, which makes the robot more autonomous.

The main goal of this work was to solve the ambiguity problem in the localization method. Here it is shown that it is possible to do this with the ALV homing method. Experiments also confirmed that the method is robust.

6.1 Future work

Although experiments with the ALV homing method were successful, no real-time experiments could be done because no good alignment method was available. Some other method to retrieve information about the orientation is for example a compass. Hafner (2001) however mentioned that a magnetic compass does not work very well inside buildings, therefore she used camera information to compensate for that. Lambrinos et al. (2000) used a polarised-light compass which worked good, but it needs sunlight from all directions above it and glass depolarises the light, therefore it cannot be used inside buildings. Vardy (2005) proposed to use the coherence of flow fields as an indicator of correct orientation. Zeil et al. (2003) suggest to use the difference between images to align them by using one-dimensional gradient descent.

For the homing method to work in real-time the panoramas should be created faster and therefore the use of a camera and parabolic mirror is a good option. This however should also be tested with the used global navigation method, such as from Ramisa (2006).

The ALV method works better when the home is relatively near. To cover larger distances with the ALV homing method, the augmented model of Smith et al. (2006) could be used, in which a sequence of home positions is stored. However there has to be another navigation or search behaviour that finds its way to a destination, e.g. a food source, the first time. Smith et al. (2006) did this by using random search, and by keeping a home vector to be able to navigate backward.

Finally the ALV homing method could be improved by using depth information, because then not only the direction but also the distance to the home can be estimated. This could

be done by using stereo images. But using two cameras also means doing twice the processing and the calculation of the depth, therefore it will be at least twice as slow as when only one camera is used.

The ALV homing method can be used as an addition to a navigation technique, not only to improve localisation ambiguity but also to perform more local navigation. The subsumption architecture (Brooks & Connell (1986)) could be used to combine the local and global navigation behaviours, the obstacle avoidance and other behaviours.

Appendix A

ALV Homing Algorithm

This appendix lists the algorithm that uses the ALV homing algorithm to return to a previously visited location. This algorithm is made for a simulated world, but the main part could also be used on a real robot.

The parameter `map` is the virtual world and contains the locations of all points in the world. These points can be invariant features or objects such as landmarks. The `AcquireCylinder` function projects the features that are visible at the current location of the robot to an image plane, thereby using the `map`. At first the projected features at the destination are projected on a cylinder. If there are no points visible from that position then obviously it is not possible to use the ALV homing method. If there are points visible then the ALV of the home location can be computed after the Cartesian coordinates of the points have been calculated.

Next the same procedure is repeated for the current location to retrieve the ALV. With both ALVs the home vector can be calculated which then is used to move the robot. This is continued until the robot is within a minimum distance of the goal or if one of the limitations has been exceeded: maximum number of iterations, maximum number of empty projections or the maximum distance.

```
function go(map, startPos, endPos)
idealDistance = EuclideanDistance(startPos, endPos);
% create the panorama from the end location (as a flat image)
cylinder = AcquireCylinder(map, endPos);
if isEmpty(cylinder)
        % no features can be seen from the end location, therefore exit
        return;
end
% calculate the positions of the features by back projecting
% them to a 3D cylinder
bp = BackProjection(cylinder);
% the ALV at the destination
endALV = calculateALV(bp);
% initialize variables
currentPos = startPos;
travelled = 0;
nrIterations = 0;
```

```
nrEmptyProj = 0;

while (EuclideanDistance(currentPos, endPos) > minimumDistance And
                nrIterations < MAX_NR_ITERATIONS And
                nrEmptyProj <= MAX_NR_EMPTY_PROJ And
                travelled <= MAX_TRAVELED_DIST)

        % get the panorama at the current locations
        % (returns a matrix with a list of the feature positions
        % on the panorama image)
        cylinder = AcquireCylinder(map, currentPos);

        if isEmpty(cylinder)
                % no projected features
                if nrIterations > 0 And Length(previousALV) > 0
                        currentALV = previousALV;
                else
                        % (with a max length of 0.5)
                        currentALV = 0.5 * randomVector();
                end
                % increase nr empty projections
                nrEmptyProj = nrEmptyProj + 1;
        else
                % back project to 3D cylinder
                % (to cartesian coordinates)
                bp = BackProjection(cylinder);
                % calculate the ALV at the current location
                currentALV = calculateALV(bp);
                % reset nr of empty projections
                nrEmptyProj = 0;
        end

        % calculate the direction to go to
        home = currentALV - endALV;
        % increase the traveled distance with the length of 'home'
        travelled = travelled +  EuclideanDistance(currentALV, endALV);
        % move
        currentPos = currentPos + home;
        % increase nr of iterations
        nrIterations = nrIterations + 1;

        previousALV = currentALV;
end
```

Appendix B

Simulation Results

This appendix shows the detailed results of the simulation (Chapter 4). Table B.1 shows the results of the first experiment. After analysing the results of the wall length another experiment was done with square rooms, these results are shown in Table B.2. The second experiment (Table B.3) was done to further analyse the effects of noise and occlusions. Finally Table B.4 shows the results of experiments where the projected panorama was rotated randomly.

World 1	*successful*	*unsuccessful*			*number of runs*		*number of iterations (successful)*			*difference with ideal (successful)*		
		max. iter.	*max. proj*	*max. dist.*	*total*	*successful*	*mean*	*median*	*std. dev.*	*mean*	*median*	*std. dev.*
x wall len. (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	10.45	10.00	0.69	0.06	0.06	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.20	12.00	0.52	0.00	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	34.25	33.50	3.80	0.21	0.21	0.02
15	100.00%	0.00%	0.00%	0.00%	20	20	61.20	58.00	9.56	0.36	0.35	0.04
20	100.00%	0.00%	0.00%	0.00%	20	20	95.95	86.50	25.01	0.50	0.52	0.06
30	55.00%	45.00%	0.00%	0.00%	20	11	197.45	178.00	60.89	0.67	0.69	0.08
50	50.00%	50.00%	0.00%	0.00%	20	10	286.20	234.00	158.38	0.79	0.77	0.20
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	11.90	12.00	0.55	0.00	0.00	0.00
0.001	100.00%	0.00%	0.00%	0.00%	20	20	43.65	12.50	136.41	0.84	0.05	3.42
0.010	40.00%	0.00%	0.00%	60.00%	20	8	50.38	34.00	44.56	4.64	2.87	4.77
0.050	10.00%	0.00%	5.00%	85.00%	20	2	39.50	39.50	37.48	9.33	9.33	10.10
0.100	10.00%	0.00%	10.00%	80.00%	20	2	17.00	17.00	4.24	4.28	4.28	1.43
0.500	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
1.000	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
nr. of features												
20	80.00%	15.00%	0.00%	5.00%	20	16	13.56	14.00	3.54	0.09	0.09	0.08
40	95.00%	5.00%	0.00%	0.00%	20	19	13.53	14.00	2.55	0.06	0.05	0.06
100	100.00%	0.00%	0.00%	0.00%	20	20	12.30	12.00	1.66	0.02	0.02	0.02
500	100.00%	0.00%	0.00%	0.00%	20	20	11.70	12.00	0.73	0.00	0.00	0.00
1000	100.00%	0.00%	0.00%	0.00%	20	20	12.25	12.00	0.44	0.00	0.00	0.00
5000	100.00%	0.00%	0.00%	0.00%	20	20	11.90	12.00	0.31	0.00	0.00	0.00
10000	100.00%	0.00%	0.00%	0.00%	20	20	12.00	12.00	0.00	0.00	0.00	0.00
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	11.85	12.00	0.67	0.00	0.00	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	12.10	12.00	0.79	0.00	0.00	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	11.80	12.00	0.70	0.00	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	11.85	12.00	0.59	0.00	0.00	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	11.85	12.00	0.67	0.00	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	11.80	12.00	0.52	0.00	0.00	0.00
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	11.90	12.00	0.45	0.00	0.00	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	12.00	12.00	0.65	0.00	0.00	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	11.80	12.00	0.62	0.00	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	11.95	12.00	0.69	0.00	0.00	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	12.15	12.00	0.75	0.00	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	12.40	12.50	0.82	0.01	0.01	0.01

Table B.1: Results of the first experiment for world 1.

World 2	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
x wall len. (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	8.85	9.00	0.49	0.01	0.01	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	12.60	12.50	0.94	0.01	0.01	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	29.45	28.50	5.34	0.09	0.09	0.02
15	70.00%	30.00%	0.00%	0.00%	20	14	55.93	54.50	16.21	0.13	0.13	0.03
20	75.00%	25.00%	0.00%	0.00%	20	15	70.80	63.00	28.89	0.14	0.12	0.04
30	70.00%	30.00%	0.00%	0.00%	20	14	81.64	73.00	46.83	0.08	0.08	0.05
50	65.00%	35.00%	0.00%	0.00%	20	13	55.85	40.00	34.52	0.07	0.02	0.14
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.40	12.00	0.68	0.01	0.00	0.00
0.001	100.00%	0.00%	0.00%	0.00%	20	20	81.55	14.50	274.28	1.21	0.07	4.68
0.010	40.00%	0.00%	0.00%	60.00%	20	8	44.25	23.00	49.55	3.08	1.57	4.19
0.050	0.00%	0.00%	25.00%	75.00%	20	0	-	-	-	-	-	-
0.100	10.00%	0.00%	45.00%	45.00%	20	2	72.50	72.50	7.78	19.37	19.37	4.17
0.500	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
1.000	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
nr. of features												
20	70.00%	25.00%	0.00%	5.00%	20	14	16.64	17.00	5.85	0.08	0.03	0.13
40	90.00%	10.00%	0.00%	0.00%	20	18	15.28	16.00	3.43	0.04	0.04	0.03
100	95.00%	5.00%	0.00%	0.00%	20	19	14.26	13.00	4.93	0.02	0.02	0.01
500	100.00%	0.00%	0.00%	0.00%	20	20	12.60	12.50	0.82	0.01	0.01	0.00
1000	100.00%	0.00%	0.00%	0.00%	20	20	12.45	12.50	1.00	0.01	0.00	0.00
5000	100.00%	0.00%	0.00%	0.00%	20	20	12.50	12.50	0.51	0.01	0.01	0.00
10000	100.00%	0.00%	0.00%	0.00%	20	20	12.55	13.00	0.51	0.01	0.01	0.00
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.30	12.00	0.73	0.01	0.01	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.81	0.01	0.01	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	12.55	13.00	0.94	0.01	0.01	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	12.60	13.00	0.82	0.01	0.01	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.93	0.01	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	12.85	13.00	0.88	0.01	0.01	0.00
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.65	12.50	0.88	0.01	0.01	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	12.70	13.00	0.80	0.01	0.01	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	1.18	0.01	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	12.65	13.00	0.88	0.01	0.01	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	12.10	12.00	0.97	0.01	0.01	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	12.85	12.00	1.63	0.01	0.01	0.01

World 3	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
x wall len. (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	11.85	12.00	0.75	0.03	0.02	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.75	13.00	0.97	0.02	0.02	0.01
10	100.00%	0.00%	0.00%	0.00%	20	20	18.55	18.00	1.70	0.07	0.06	0.02
15	100.00%	0.00%	0.00%	0.00%	20	20	29.70	30.00	3.56	0.21	0.21	0.04
20	90.00%	10.00%	0.00%	0.00%	20	18	44.22	42.00	9.53	0.37	0.38	0.06
30	55.00%	45.00%	0.00%	0.00%	20	11	74.82	64.00	24.57	0.50	0.47	0.09
50	45.00%	55.00%	0.00%	0.00%	20	9	147.22	139.00	47.78	0.74	0.73	0.11
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.10	12.00	0.64	0.02	0.02	0.01
0.001	100.00%	0.00%	0.00%	0.00%	20	20	13.45	12.00	3.17	0.11	0.09	0.08
0.010	45.00%	0.00%	0.00%	55.00%	20	9	65.56	72.00	48.22	6.30	6.95	5.02
0.050	5.00%	0.00%	0.00%	95.00%	20	1	10.00	10.00	0.00	0.62	0.62	0.00
0.100	0.00%	0.00%	25.00%	75.00%	20	0	-	-	-	-	-	-
0.500	0.00%	0.00%	25.00%	75.00%	20	0	-	-	-	-	-	-
1.000	0.00%	0.00%	25.00%	75.00%	20	0	-	-	-	-	-	-
nr. of features												
20	80.00%	10.00%	0.00%	10.00%	20	16	18.63	20.00	6.57	0.15	0.10	0.16
40	85.00%	10.00%	0.00%	5.00%	20	17	14.71	12.00	5.55	0.11	0.08	0.11
100	100.00%	0.00%	0.00%	0.00%	20	20	13.50	12.00	3.75	0.04	0.04	0.02
500	100.00%	0.00%	0.00%	0.00%	20	20	12.25	12.00	1.12	0.02	0.02	0.01
1000	100.00%	0.00%	0.00%	0.00%	20	20	12.50	12.50	1.00	0.02	0.02	0.01
5000	100.00%	0.00%	0.00%	0.00%	20	20	12.20	12.00	0.41	0.02	0.02	0.00
10000	100.00%	0.00%	0.00%	0.00%	20	20	12.15	12.00	0.37	0.02	0.02	0.00
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.70	13.00	0.86	0.02	0.02	0.01
1	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.67	0.02	0.02	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.67	0.02	0.02	0.01
10	100.00%	0.00%	0.00%	0.00%	20	20	12.50	12.00	0.89	0.02	0.02	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	12.40	12.00	0.75	0.02	0.02	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	12.70	13.00	0.98	0.02	0.02	0.01
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.50	12.00	0.95	0.02	0.02	0.01
1	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.81	0.02	0.02	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.65	13.00	0.75	0.02	0.02	0.01
10	100.00%	0.00%	0.00%	0.00%	20	20	12.15	12.00	0.88	0.02	0.02	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	12.15	12.00	0.93	0.02	0.02	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	12.50	12.50	1.15	0.02	0.02	0.01

Table B.1: Results of the first experiment for worlds 2 and 3.

World 4	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
x wall len. (m)												
3	40.00%	50.00%	10.00%	0.00%	20	8	49.75	48.00	5.78	0.94	0.95	0.03
5	100.00%	0.00%	0.00%	0.00%	20	20	22.65	23.00	0.75	0.58	0.58	0.04
10	100.00%	0.00%	0.00%	0.00%	20	20	19.70	19.50	2.15	0.02	0.02	0.01
15	95.00%	5.00%	0.00%	0.00%	20	19	36.89	35.00	10.21	0.08	0.08	0.04
20	80.00%	20.00%	0.00%	0.00%	20	16	50.00	45.50	16.03	0.17	0.17	0.07
30	55.00%	45.00%	0.00%	0.00%	20	11	92.09	71.00	58.24	0.31	0.29	0.20
50	35.00%	65.00%	0.00%	0.00%	20	7	77.71	62.00	32.98	0.31	0.27	0.21
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	22.70	22.50	1.22	0.57	0.57	0.04
0.001	60.00%	0.00%	0.00%	40.00%	20	12	34.75	25.50	20.03	0.76	0.61	0.41
0.010	15.00%	0.00%	0.00%	85.00%	20	3	91.67	25.00	118.95	7.84	1.01	11.97
0.050	5.00%	0.00%	25.00%	70.00%	20	1	106.00	106.00	0.00	20.25	20.25	0.00
0.100	15.00%	0.00%	35.00%	50.00%	20	3	35.67	37.00	4.16	9.80	9.19	3.93
0.500	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
1.000	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
nr. of features												
20	50.00%	45.00%	0.00%	5.00%	20	10	27.10	23.00	7.81	0.73	0.63	0.45
40	45.00%	55.00%	0.00%	0.00%	20	9	27.67	27.00	7.48	0.67	0.61	0.21
100	55.00%	45.00%	0.00%	0.00%	20	11	23.82	23.00	4.31	0.59	0.55	0.19
500	95.00%	5.00%	0.00%	0.00%	20	19	22.26	22.00	1.56	0.56	0.57	0.06
1000	100.00%	0.00%	0.00%	0.00%	20	20	22.85	23.00	1.14	0.58	0.57	0.04
5000	100.00%	0.00%	0.00%	0.00%	20	20	22.55	23.00	0.51	0.57	0.58	0.02
10000	100.00%	0.00%	0.00%	0.00%	20	20	22.55	23.00	0.51	0.57	0.57	0.02
fake features												
0	95.00%	5.00%	0.00%	0.00%	20	19	23.21	23.00	1.13	0.58	0.59	0.04
1	100.00%	0.00%	0.00%	0.00%	20	20	22.95	23.00	1.19	0.60	0.60	0.04
5	85.00%	15.00%	0.00%	0.00%	20	17	23.12	23.00	1.11	0.59	0.59	0.04
10	100.00%	0.00%	0.00%	0.00%	20	20	22.90	23.00	1.17	0.58	0.57	0.04
20	95.00%	5.00%	0.00%	0.00%	20	19	23.26	23.00	1.56	0.58	0.59	0.05
50	100.00%	0.00%	0.00%	0.00%	20	20	22.95	23.00	1.64	0.59	0.59	0.06
removed feat.												
0	95.00%	5.00%	0.00%	0.00%	20	19	22.68	23.00	1.34	0.57	0.58	0.05
1	100.00%	0.00%	0.00%	0.00%	20	20	22.85	23.00	1.18	0.57	0.57	0.04
5	100.00%	0.00%	0.00%	0.00%	20	20	23.00	22.00	2.22	0.58	0.57	0.05
10	90.00%	10.00%	0.00%	0.00%	20	18	22.83	23.00	1.34	0.59	0.57	0.04
20	100.00%	0.00%	0.00%	0.00%	20	20	23.70	23.00	3.25	0.59	0.58	0.05
50	95.00%	5.00%	0.00%	0.00%	20	19	23.32	23.00	1.11	0.60	0.61	0.04

Table B.1: Results of the first experiment for world 4.

World 1	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
wall length (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	7.05	7.00	0.39	0.01	0.01	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.20	12.00	0.52	0.00	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	37.20	37.00	0.70	0.00	0.00	0.00
15	100.00%	0.00%	0.00%	0.00%	20	20	56.60	56.50	1.57	0.00	0.00	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	75.25	75.00	1.94	0.00	0.00	0.00
30	100.00%	0.00%	0.00%	0.00%	20	20	113.80	113.50	2.53	0.00	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	190.05	189.50	3.49	0.00	0.00	0.00

World 2	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
wall length (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	11.00	11.00	1.08	0.07	0.07	0.01
5	100.00%	0.00%	0.00%	0.00%	20	20	12.60	12.50	0.94	0.01	0.01	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	36.60	37.00	0.82	0.00	0.00	0.00
15	100.00%	0.00%	0.00%	0.00%	20	20	56.70	57.00	1.34	0.00	0.00	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	76.20	76.00	0.95	0.00	0.00	0.00
30	100.00%	0.00%	0.00%	0.00%	20	20	113.20	113.00	2.67	0.00	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	189.50	189.50	4.01	0.00	0.00	0.00

World 3	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
wall length (m)												
3	100.00%	0.00%	0.00%	0.00%	20	20	8.00	8.00	0.56	0.05	0.05	0.02
5	100.00%	0.00%	0.00%	0.00%	20	20	12.75	13.00	0.97	0.02	0.02	0.01
10	100.00%	0.00%	0.00%	0.00%	20	20	80.25	80.50	1.89	0.38	0.39	0.01
15	100.00%	0.00%	0.00%	0.00%	20	20	124.25	125.00	2.75	0.38	0.38	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	168.55	168.00	3.09	0.39	0.39	0.01
30	100.00%	0.00%	0.00%	0.00%	20	20	257.15	256.00	5.38	0.38	0.38	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	427.75	427.50	10.34	0.38	0.38	0.01

World 4	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
wall length (m)												
3	80.00%	20.00%	0.00%	0.00%	20	16	12.44	12.00	1.15	0.52	0.51	0.05
5	100.00%	0.00%	0.00%	0.00%	20	20	22.65	23.00	0.75	0.58	0.58	0.04
10	100.00%	0.00%	0.00%	0.00%	20	20	65.00	65.00	1.45	0.41	0.41	0.01
15	100.00%	0.00%	0.00%	0.00%	20	20	106.50	107.00	2.21	0.40	0.40	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	149.25	149.50	4.53	0.40	0.39	0.02
30	100.00%	0.00%	0.00%	0.00%	20	20	235.75	236.00	4.29	0.39	0.39	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	405.30	406.00	7.29	0.38	0.38	0.01

Table B.2: Results of the *square room* experiment.

World 1	*successful*	*unsuccessful*			*number of runs*		*number of iterations (successful)*			*difference with ideal (successful)*		
		max. iter.	*max. proj*	*max. dist.*	*total*	*successful*	*mean*	*median*	*std. dev.*	*mean*	*median*	*std. dev.*
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	15.40	16.00	0.94	0.00	0.00	0.00
0.0001	100.00%	0.00%	0.00%	0.00%	20	20	15.60	16.00	0.68	0.01	0.01	0.00
0.0005	100.00%	0.00%	0.00%	0.00%	20	20	15.55	16.00	1.39	0.02	0.02	0.01
0.001	100.00%	0.00%	0.00%	0.00%	20	20	16.40	15.00	3.08	0.07	0.05	0.06
0.005	75.00%	0.00%	0.00%	25.00%	20	15	97.87	20.00	200.49	5.01	0.51	11.41
0.010	60.00%	0.00%	0.00%	40.00%	20	12	114.00	26.00	163.93	9.18	1.38	14.18
0.050	0.00%	0.00%	25.00%	75.00%	20	0	-	-	-	-	-	-
0.100	5.00%	0.00%	50.00%	45.00%	20	1	11.00	11.00	0.00	1.02	1.02	0.00
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	15.60	16.00	0.82	0.01	0.01	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	15.60	16.00	0.60	0.01	0.01	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	15.35	15.00	0.81	0.01	0.01	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	15.25	15.00	0.64	0.01	0.01	0.00
20	100.00%	0.00%	0.00%	0.00%	20	20	15.10	15.00	0.45	0.01	0.00	0.00
50	100.00%	0.00%	0.00%	0.00%	20	20	15.35	15.00	0.93	0.01	0.00	0.00
500	100.00%	0.00%	0.00%	0.00%	20	20	15.30	15.00	0.92	0.01	0.00	0.00
700	100.00%	0.00%	0.00%	0.00%	20	20	15.30	15.00	0.66	0.01	0.00	0.00
900	100.00%	0.00%	0.00%	0.00%	20	20	15.45	15.00	0.83	0.01	0.01	0.00
950	100.00%	0.00%	0.00%	0.00%	20	20	15.20	15.00	0.77	0.00	0.00	0.00
980	100.00%	0.00%	0.00%	0.00%	20	20	15.45	15.00	0.69	0.01	0.00	0.00
990	100.00%	0.00%	0.00%	0.00%	20	20	15.15	15.00	0.81	0.01	0.00	0.00
995	100.00%	0.00%	0.00%	0.00%	20	20	15.00	15.00	0.73	0.01	0.01	0.00
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	15.05	15.00	0.76	0.01	0.01	0.00
1	100.00%	0.00%	0.00%	0.00%	20	20	15.65	16.00	0.93	0.01	0.00	0.00
5	100.00%	0.00%	0.00%	0.00%	20	20	15.00	15.00	0.86	0.01	0.00	0.00
10	100.00%	0.00%	0.00%	0.00%	20	20	15.25	15.00	0.97	0.01	0.01	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	15.40	15.50	0.68	0.01	0.01	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	15.40	15.00	0.82	0.01	0.01	0.01
500	100.00%	0.00%	0.00%	0.00%	20	20	17.85	16.00	6.63	0.11	0.07	0.13
700	90.00%	0.00%	0.00%	10.00%	20	18	18.28	16.50	7.58	0.22	0.18	0.25
900	85.00%	0.00%	0.00%	15.00%	20	17	57.47	18.00	107.42	3.45	0.58	7.88
950	60.00%	0.00%	0.00%	40.00%	20	12	139.08	83.50	145.07	14.10	7.86	15.73
980	10.00%	0.00%	0.00%	90.00%	20	2	37.50	37.50	33.23	4.86	4.86	5.61
990	0.00%	0.00%	15.00%	85.00%	20	0	-	-	-	-	-	-
995	0.00%	0.00%	45.00%	55.00%	20	0	-	-	-	-	-	-

World 2	*successful*	*unsuccessful*			*number of runs*		*number of iterations (successful)*			*difference with ideal (successful)*		
		max. iter.	*max. proj*	*max. dist.*	*total*	*successful*	*mean*	*median*	*std. dev.*	*mean*	*median*	*std. dev.*
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	28.95	28.00	3.58	0.17	0.17	0.02
0.0001	100.00%	0.00%	0.00%	0.00%	20	20	28.25	28.00	3.37	0.17	0.18	0.02
0.0005	100.00%	0.00%	0.00%	0.00%	20	20	32.75	30.50	8.16	0.24	0.22	0.06
0.001	100.00%	0.00%	0.00%	0.00%	20	20	42.70	30.00	47.47	0.47	0.29	0.69
0.005	80.00%	0.00%	5.00%	15.00%	20	16	38.06	27.00	20.41	1.11	0.76	0.78
0.010	50.00%	0.00%	10.00%	40.00%	20	10	68.80	41.50	96.40	3.71	2.01	6.28
0.050	5.00%	0.00%	75.00%	20.00%	20	1	179.00	179.00	0.00	26.59	26.59	0.00
0.100	15.00%	0.00%	75.00%	10.00%	20	3	11.33	11.00	3.51	0.90	1.18	0.58
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	28.30	28.00	2.66	0.17	0.17	0.01
1	100.00%	0.00%	0.00%	0.00%	20	20	29.20	28.00	3.00	0.17	0.17	0.02
5	100.00%	0.00%	0.00%	0.00%	20	20	29.90	30.00	3.04	0.17	0.17	0.02
10	100.00%	0.00%	0.00%	0.00%	20	20	29.10	28.50	3.84	0.17	0.17	0.02
20	100.00%	0.00%	0.00%	0.00%	20	20	28.40	29.00	2.95	0.17	0.17	0.01
50	100.00%	0.00%	0.00%	0.00%	20	20	29.95	30.00	3.62	0.17	0.17	0.02
500	100.00%	0.00%	0.00%	0.00%	20	20	29.65	30.00	3.67	0.17	0.17	0.01
700	100.00%	0.00%	0.00%	0.00%	20	20	30.55	30.00	4.11	0.17	0.17	0.01
900	100.00%	0.00%	0.00%	0.00%	20	20	29.20	28.00	3.90	0.17	0.17	0.02
950	100.00%	0.00%	0.00%	0.00%	20	20	28.35	28.00	3.03	0.17	0.16	0.02
980	100.00%	0.00%	0.00%	0.00%	20	20	28.85	28.00	3.59	0.17	0.17	0.02
990	100.00%	0.00%	0.00%	0.00%	20	20	29.20	29.00	2.75	0.17	0.17	0.02
995	100.00%	0.00%	0.00%	0.00%	20	20	29.45	29.50	4.24	0.17	0.17	0.02
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	28.20	28.50	3.79	0.17	0.17	0.01
1	100.00%	0.00%	0.00%	0.00%	20	20	28.85	28.00	3.20	0.17	0.17	0.02
5	100.00%	0.00%	0.00%	0.00%	20	20	29.00	28.00	3.64	0.17	0.17	0.01
10	100.00%	0.00%	0.00%	0.00%	20	20	29.15	29.00	4.15	0.17	0.17	0.01
20	100.00%	0.00%	0.00%	0.00%	20	20	30.95	30.00	4.51	0.18	0.18	0.02
50	100.00%	0.00%	0.00%	0.00%	20	20	28.95	28.50	4.36	0.18	0.18	0.02
500	95.00%	5.00%	0.00%	0.00%	20	19	43.74	32.00	31.75	0.52	0.34	0.52
700	70.00%	0.00%	0.00%	30.00%	20	14	40.86	27.00	42.11	0.73	0.38	1.06
900	45.00%	0.00%	10.00%	45.00%	20	9	184.89	34.00	303.75	9.59	1.41	16.77
950	35.00%	0.00%	20.00%	45.00%	20	7	73.29	37.00	92.27	5.15	2.20	8.40
980	35.00%	0.00%	40.00%	25.00%	20	7	70.71	25.00	90.17	7.47	2.15	10.60
990	15.00%	0.00%	65.00%	20.00%	20	3	30.67	24.00	21.78	4.01	2.02	4.08
995	0.00%	0.00%	85.00%	15.00%	20	0	-	-	-	-	-	-

Table B.3: Results of the second experiment for worlds 1 and 2.

World 3	*successful*	*unsuccessful*			*number of runs*		*number of iterations (successful)*			*difference with ideal (successful)*		
		max. iter.	*max. proj*	*max. dist.*	*total*	*successful*	*mean*	*median*	*std. dev.*	*mean*	*median*	*std. dev.*
noise												
0	100.00%	0.00%	0.00%	0.00%	20	20	18.80	19.00	0.89	0.15	0.15	0.03
0.0001	100.00%	0.00%	0.00%	0.00%	20	20	18.35	18.00	1.04	0.14	0.13	0.02
0.0005	100.00%	0.00%	0.00%	0.00%	20	20	19.25	19.00	1.86	0.20	0.20	0.05
0.001	100.00%	0.00%	0.00%	0.00%	20	20	25.45	20.00	26.18	0.42	0.28	0.57
0.005	85.00%	0.00%	5.00%	10.00%	20	17	90.35	25.00	147.22	4.43	0.90	7.62
0.010	65.00%	0.00%	0.00%	35.00%	20	13	150.38	48.00	186.84	11.52	2.90	15.01
0.050	10.00%	0.00%	35.00%	55.00%	20	2	24.00	24.00	7.07	3.50	3.50	1.44
0.100	5.00%	0.00%	35.00%	60.00%	20	1	42.00	42.00	0.00	11.00	11.00	0.00
fake features												
0	100.00%	0.00%	0.00%	0.00%	20	20	19.05	19.00	1.00	0.13	0.12	0.03
1	100.00%	0.00%	0.00%	0.00%	20	20	17.95	18.00	0.94	0.13	0.13	0.03
5	100.00%	0.00%	0.00%	0.00%	20	20	18.45	19.00	0.83	0.13	0.13	0.02
10	100.00%	0.00%	0.00%	0.00%	20	20	18.60	19.00	1.19	0.13	0.13	0.03
20	100.00%	0.00%	0.00%	0.00%	20	20	19.05	19.00	0.76	0.14	0.13	0.03
50	100.00%	0.00%	0.00%	0.00%	20	20	18.80	19.00	1.40	0.13	0.13	0.02
500	100.00%	0.00%	0.00%	0.00%	20	20	18.70	19.00	0.80	0.13	0.12	0.02
700	100.00%	0.00%	0.00%	0.00%	20	20	18.60	18.50	1.19	0.12	0.12	0.03
900	100.00%	0.00%	0.00%	0.00%	20	20	18.20	18.00	1.01	0.12	0.12	0.02
950	100.00%	0.00%	0.00%	0.00%	20	20	18.40	18.00	1.05	0.13	0.14	0.03
980	100.00%	0.00%	0.00%	0.00%	20	20	18.70	19.00	1.17	0.13	0.13	0.03
990	100.00%	0.00%	0.00%	0.00%	20	20	19.00	19.00	0.92	0.14	0.13	0.02
995	100.00%	0.00%	0.00%	0.00%	20	20	18.45	18.00	0.94	0.12	0.12	0.02
removed feat.												
0	100.00%	0.00%	0.00%	0.00%	20	20	18.55	18.00	0.94	0.13	0.13	0.02
1	100.00%	0.00%	0.00%	0.00%	20	20	18.60	19.00	1.23	0.14	0.14	0.02
5	100.00%	0.00%	0.00%	0.00%	20	20	18.50	19.00	0.89	0.13	0.12	0.03
10	100.00%	0.00%	0.00%	0.00%	20	20	18.80	19.00	0.89	0.14	0.14	0.03
20	100.00%	0.00%	0.00%	0.00%	20	20	18.80	19.00	0.95	0.14	0.14	0.03
50	100.00%	0.00%	0.00%	0.00%	20	20	18.45	18.50	1.05	0.14	0.14	0.02
500	95.00%	5.00%	0.00%	0.00%	20	19	36.00	20.00	65.28	0.66	0.26	1.65
700	95.00%	0.00%	0.00%	5.00%	20	19	22.89	19.00	13.62	0.50	0.38	0.52
900	70.00%	0.00%	0.00%	30.00%	20	14	146.86	32.50	201.71	10.14	1.83	14.87
950	40.00%	0.00%	10.00%	50.00%	20	8	33.63	22.50	25.73	2.48	1.27	2.66
980	5.00%	0.00%	40.00%	55.00%	20	1	30.00	30.00	0.00	2.69	2.69	0.00
990	10.00%	0.00%	30.00%	60.00%	20	2	13.50	13.50	0.71	1.49	1.49	0.16
995	0.00%	0.00%	50.00%	50.00%	20	0	-	-	-	-	-	-

World 4	*successful*	*unsuccessful*			*number of runs*		*number of iterations (successful)*			*difference with ideal (successful)*		
		max. iter.	*max. proj*	*max. dist.*	*total*	*successful*	*mean*	*median*	*std. dev.*	*mean*	*median*	*std. dev.*
noise												
0	95.00%	5.00%	0.00%	0.00%	20	19	19.95	20.00	2.25	0.65	0.65	0.07
0.0001	80.00%	20.00%	0.00%	0.00%	20	16	36.63	21.00	60.05	0.80	0.69	0.48
0.0005	85.00%	15.00%	0.00%	0.00%	20	17	101.06	20.00	229.66	2.20	0.72	4.20
0.001	65.00%	0.00%	5.00%	30.00%	20	13	27.23	22.00	15.23	0.95	0.72	0.55
0.005	25.00%	0.00%	40.00%	35.00%	20	5	53.20	42.00	41.40	5.11	2.68	5.09
0.010	10.00%	0.00%	45.00%	45.00%	20	2	58.00	58.00	38.18	10.33	10.33	9.07
0.050	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-
0.100	0.00%	0.00%	95.00%	5.00%	20	0	-	-	-	-	-	-
fake features												
0	80.00%	20.00%	0.00%	0.00%	20	16	18.94	19.00	2.17	0.60	0.61	0.07
1	95.00%	5.00%	0.00%	0.00%	20	19	20.89	21.00	2.92	0.66	0.67	0.10
5	100.00%	0.00%	0.00%	0.00%	20	20	20.35	19.00	5.31	0.64	0.63	0.10
10	90.00%	10.00%	0.00%	0.00%	20	18	21.78	20.00	7.46	0.68	0.66	0.08
20	65.00%	35.00%	0.00%	0.00%	20	13	19.69	19.00	4.55	0.62	0.62	0.09
50	80.00%	20.00%	0.00%	0.00%	20	16	19.69	19.00	2.36	0.65	0.64	0.11
500	95.00%	5.00%	0.00%	0.00%	20	19	20.58	20.00	3.82	0.68	0.67	0.11
700	85.00%	15.00%	0.00%	0.00%	20	17	20.88	19.00	4.48	0.63	0.65	0.10
900	100.00%	0.00%	0.00%	0.00%	20	20	19.70	18.50	4.51	0.63	0.62	0.10
950	70.00%	30.00%	0.00%	0.00%	20	14	21.07	20.00	3.50	0.68	0.68	0.08
980	90.00%	10.00%	0.00%	0.00%	20	18	22.22	19.00	12.14	0.64	0.62	0.09
990	90.00%	10.00%	0.00%	0.00%	20	18	19.50	20.00	3.11	0.63	0.65	0.12
995	80.00%	20.00%	0.00%	0.00%	20	16	21.19	19.00	6.44	0.66	0.64	0.08
removed feat.												
0	85.00%	15.00%	0.00%	0.00%	20	17	20.35	20.00	3.66	0.64	0.66	0.11
1	85.00%	15.00%	0.00%	0.00%	20	17	19.06	18.00	3.25	0.60	0.60	0.11
5	90.00%	10.00%	0.00%	0.00%	20	18	20.67	20.00	3.12	0.65	0.63	0.10
10	75.00%	25.00%	0.00%	0.00%	20	15	22.80	20.00	9.99	0.66	0.65	0.06
20	85.00%	15.00%	0.00%	0.00%	20	17	20.59	19.00	4.84	0.67	0.62	0.11
50	80.00%	20.00%	0.00%	0.00%	20	16	20.75	19.50	5.07	0.66	0.66	0.14
500	50.00%	10.00%	0.00%	40.00%	20	10	69.20	24.50	137.29	2.14	0.82	4.14
700	40.00%	0.00%	15.00%	45.00%	20	8	39.25	28.50	33.33	1.59	1.14	1.44
900	30.00%	0.00%	20.00%	50.00%	20	6	90.83	32.00	119.60	7.51	1.89	11.14
950	15.00%	0.00%	60.00%	25.00%	20	3	130.33	61.00	149.58	12.70	4.99	15.68
980	10.00%	0.00%	75.00%	15.00%	20	2	196.50	196.50	57.28	49.91	49.91	0.56
990	0.00%	0.00%	90.00%	10.00%	20	0	-	-	-	-	-	-
995	0.00%	0.00%	100.00%	0.00%	20	0	-	-	-	-	-	-

Table B.3: Results of the second experiment for worlds 3 and 4.

World 1	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
rotate (rad)												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.35	12.00	0.67	0.00	0.00	0.00
pi/10	75.00%	0.00%	0.00%	25.00%	20	15	214.93	152.00	183.62	7.12	5.18	6.28
2pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
3pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
4pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
5pi/10	5.00%	0.00%	0.00%	95.00%	20	1	7.00	7.00	0.00	0.39	0.39	0.00
6pi/10	5.00%	0.00%	0.00%	95.00%	20	1	7.00	7.00	0.00	0.36	0.36	0.00
7pi/10	0.00%	0.00%	0.00%	1	20	0	-	-	-	-	-	-
8pi/80	5.00%	0.00%	0.00%	95.00%	20	1	7.00	7.00	0.00	0.56	0.56	0.00
9pi/10	20.00%	0.00%	0.00%	80.00%	20	4	6.25	6.00	0.50	0.21	0.18	0.09
pi	10.00%	0.00%	0.00%	90.00%	20	2	8.00	8.00	1.41	0.29	0.29	0.10

World 2	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
rotate (rad)												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.45	12.50	0.60	0.01	0.01	0.00
pi/10	70.00%	0.00%	0.00%	30.00%	20	14	24.07	7.00	48.34	1.12	0.08	3.05
2pi/10	55.00%	0.00%	0.00%	45.00%	20	11	6.18	6.00	0.60	0.10	0.06	0.07
3pi/10	15.00%	0.00%	0.00%	85.00%	20	3	6.00	6.00	1.00	0.14	0.13	0.06
4pi/10	5.00%	0.00%	0.00%	95.00%	20	1	5.00	5.00	0.00	0.14	0.14	0.00
5pi/10	0.00%	0.00%	5.00%	95.00%	20	0	-	-	-	-	-	-
6pi/10	0.00%	0.00%	10.00%	90.00%	20	0	-	-	-	-	-	-
7pi/10	0.00%	0.00%	20.00%	80.00%	20	0	-	-	-	-	-	-
8pi/80	0.00%	0.00%	45.00%	55.00%	20	0	-	-	-	-	-	-
9pi/10	0.00%	0.00%	45.00%	55.00%	20	0	-	-	-	-	-	-
pi	0.00%	0.00%	70.00%	30.00%	20	0	-	-	-	-	-	-

World 3	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
rotate (rad)												
0	100.00%	0.00%	0.00%	0.00%	20	20	12.25	12.00	0.44	0.02	0.02	0.01
pi/10	90.00%	0.00%	0.00%	10.00%	20	18	192.67	162.50	185.78	6.42	5.53	6.53
2pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
3pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
4pi/10	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
5pi/10	5.00%	0.00%	0.00%	95.00%	20	1	7.00	7.00	0.00	0.37	0.37	0.00
6pi/10	25.00%	0.00%	0.00%	75.00%	20	5	6.80	7.00	0.45	0.35	0.39	0.13
7pi/10	15.00%	0.00%	5.00%	80.00%	20	3	7.33	7.00	0.58	0.49	0.49	0.10
8pi/80	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-
9pi/10	0.00%	0.00%	10.00%	90.00%	20	0	-	-	-	-	-	-
pi	0.00%	0.00%	0.00%	100.00%	20	0	-	-	-	-	-	-

World 4	successful	unsuccessful			number of runs		number of iterations (successful)			difference with ideal (successful)		
		max. iter.	max. proj	max. dist.	total	successful	mean	median	std. dev.	mean	median	std. dev.
rotate (rad)												
0	100.00%	0.00%	0.00%	0.00%	20	20	23.00	23.00	1.75	0.57	0.57	0.06
pi/10	15.00%	0.00%	0.00%	85.00%	20	3	243.33	219.00	107.58	16.47	15.30	6.83
2pi/10	0.00%	0.00%	30.00%	70.00%	20	0	-	-	-	-	-	-
3pi/10	0.00%	0.00%	95.00%	5.00%	20	0	-	-	-	-	-	-
4pi/10	0.00%	0.00%	80.00%	20.00%	20	0	-	-	-	-	-	-
5pi/10	0.00%	0.00%	90.00%	10.00%	20	0	-	-	-	-	-	-
6pi/10	0.00%	0.00%	75.00%	25.00%	20	0	-	-	-	-	-	-
7pi/10	0.00%	0.00%	95.00%	5.00%	20	0	-	-	-	-	-	-
8pi/80	0.00%	0.00%	90.00%	10.00%	20	0	-	-	-	-	-	-
9pi/10	0.00%	0.00%	95.00%	5.00%	20	0	-	-	-	-	-	-
pi	0.00%	0.00%	95.00%	5.00%	20	0	-	-	-	-	-	-

Table B.4: Results of the third experiment.

Appendix C

Real World Results

#	dataset	type	features	mean	median	std.dev.	score	best home	n
25	corridor	not filtered	MSER	52.66	35.71	44.89	0.7074	200	6
29	corridor	lower half	MSER	48.83	39.02	41.63	0.7287	203	6
31	corridor	upper half	MSER	59.15	42.56	46.02	0.6714	199	6
32	corridor	not filtered	DoG	56.26	44.58	43.64	0.6874	203	6
33	corridor	lower half	DoG	56.45	49.69	42.39	0.6864	203	6
38	corridor	upper half	DoG	57.08	38.19	45.65	0.6829	203	6
4	robot lab	not filtered	Landmarks	14.88	10.16	14.86	0.9173	110	38
8	robot lab	not filtered	MSER	27.84	16.03	35.51	0.8454	117	38
10	robot lab	lower half	MSER	21.96	11.09	30.05	0.8780	117	38
11	robot lab	upper half	MSER	50.62	39.33	42.47	0.7188	117	38
16	robot lab	not filtered	DoG	35.60	22.85	38.67	0.8022	117	38
27	robot lab	lower half	DoG	26.90	13.05	34.74	0.8506	117	38
30	robot lab	upper half	DoG	56.14	45.77	43.84	0.6881	117	38
1	square room	not filtered	MSER	9.65	12.03	7.84	0.9464	138	3
2	square room	lower half	MSER	20.62	25.27	8.49	0.8855	138	3
3	square room	upper half	MSER	6.83	4.10	5.33	0.9621	138	3
5	square room	not filtered	DoG	13.78	12.00	11.31	0.9234	138	3
6	square room	lower half	DoG	14.94	14.52	10.75	0.9170	138	3
7	square room	upper half	DoG	20.91	18.96	6.64	0.8838	138	3

Table C.1: This table shows the results of the IIIA data sets as discussed in Chapter 5. The type column shows which part of the panorama has been used: all features (*not filtered*), only the features at the *lower half* of the panorama or only at the *upper half* (see section 5.9). The features column shows which features have been used to perform homing: DoG or MSER, or artificial *landmarks* which were only available in the *robot laboratory* (see Chapter 5 and section 2.2.3). The next three columns: mean, median and std. dev. (standard deviation) show information about the direction error of the home vector in degrees. The calculation of the score is shown in Eqn. 5.6; 1 being best and 0 being worst. The 'best home' column shows the ID of the location of the home whereto the mean error is smallest. Finally the n column shows the number of samples, i.e. different panoramas, for the data set. The number in the first column shows the rank of the score in the results list of all data sets (Table C.3).

#	dataset	type	features	mean	median	std.dev.	score	best home	n
14	arboreal	5	MSER	112.73	122.76	48.51	0.3737	14	170
17	arboreal	10	MSER	49.97	35.14	44.80	0.7224	153	170
26	arboreal	15	MSER	37.83	25.31	37.20	0.7898	17	170
66	arboreal	20	MSER	34.89	23.39	35.49	0.8061	50	170
75	arboreal	5	DoG	100.55	104.77	51.59	0.4414	135	170
80	arboreal	10	DoG	99.07	101.86	51.64	0.4496	135	170
81	arboreal	15	DoG	95.20	95.05	51.65	0.4711	4	170
92	arboreal	20	DoG	90.33	88.88	50.27	0.4981	4	170
37	chairs	5	MSER	116.47	126.06	46.16	0.3529	14	170
53	chairs	10	MSER	84.42	82.22	50.74	0.5310	14	170
57	chairs	15	MSER	79.61	72.79	51.91	0.5577	16	170
59	chairs	20	MSER	58.92	45.23	47.55	0.6726	84	170
65	chairs	5	DoG	93.51	92.00	50.99	0.4805	5	170
73	chairs	10	DoG	93.46	92.24	51.88	0.4808	4	170
74	chairs	15	DoG	90.33	87.09	51.16	0.4981	4	170
93	chairs	20	DoG	86.15	82.29	49.48	0.5214	3	170
9	day	5	MSER	72.67	62.94	50.31	0.5963	169	170
18	day	10	MSER	62.50	51.30	47.17	0.6528	17	170
41	day	15	MSER	39.78	29.30	36.49	0.7790	17	170
48	day	20	MSER	26.18	18.73	27.62	0.8545	17	170
69	day	5	DoG	93.10	94.15	51.72	0.4828	152	170
70	day	10	DoG	92.99	94.81	51.31	0.4834	152	170
71	day	15	DoG	93.00	94.20	50.95	0.4833	135	170
72	day	20	DoG	93.05	93.13	50.42	0.4830	135	170
13	doorlit	5	MSER	102.79	110.41	51.90	0.4290	14	170
15	doorlit	10	MSER	48.45	33.95	43.90	0.7308	14	170
24	doorlit	15	MSER	30.69	19.35	33.38	0.8295	15	170
54	doorlit	20	MSER	35.41	21.27	38.52	0.8033	50	170
55	doorlit	5	DoG	80.15	75.59	52.15	0.5547	169	170
56	doorlit	10	DoG	83.79	81.13	50.27	0.5345	169	170
58	doorlit	15	DoG	84.44	81.34	49.75	0.5309	152	170
86	doorlit	20	DoG	83.07	80.38	49.17	0.5385	151	170
19	hall1	5	MSER	62.69	53.82	44.78	0.6517	39	200
22	hall1	10	MSER	45.81	35.12	39.05	0.7455	41	200
35	hall1	15	MSER	42.61	31.55	38.32	0.7633	159	200
42	hall1	20	MSER	58.49	48.71	44.53	0.6751	99	200
51	hall1	5	DoG	108.85	109.20	46.09	0.3953	80	200
64	hall1	10	DoG	101.86	99.59	44.55	0.4341	40	200
83	hall1	15	DoG	89.27	85.64	45.78	0.5041	0	200
91	hall1	20	DoG	77.16	71.90	45.43	0.5713	0	200
39	hall2	5	MSER	68.57	57.22	49.19	0.6190	19	200
40	hall2	10	MSER	72.72	62.01	50.68	0.5960	19	200
45	hall2	15	MSER	61.00	50.90	45.30	0.6611	18	200
49	hall2	20	MSER	59.51	49.20	43.09	0.6694	18	200
94	hall2	5	DoG	116.47	130.14	49.96	0.3529	198	200
96	hall2	10	DoG	124.42	137.90	45.31	0.3088	199	200
98	hall2	15	DoG	125.99	137.57	43.33	0.3000	61	200
99	hall2	20	DoG	122.09	132.21	44.38	0.3217	20	200
20	original	5	MSER	118.50	128.27	44.93	0.3416	153	170
36	original	10	MSER	72.09	62.67	50.85	0.5995	14	170
46	original	15	MSER	58.53	45.11	47.56	0.6748	17	170
67	original	20	MSER	43.18	32.23	38.49	0.7601	50	170
78	original	5	DoG	103.19	108.47	50.36	0.4267	50	170
84	original	10	DoG	101.98	105.32	50.19	0.4335	4	170
88	original	15	DoG	97.93	97.66	49.81	0.4559	4	170
95	original	20	DoG	91.36	88.78	49.70	0.4924	3	170
12	screen	5	MSER	66.71	52.61	50.03	0.6294	102	170
21	screen	10	MSER	63.43	54.30	45.02	0.6476	153	170
43	screen	15	MSER	45.71	33.75	40.43	0.7461	0	170
44	screen	20	MSER	28.64	18.42	31.04	0.8409	95	170
60	screen	5	DoG	86.87	85.63	51.82	0.5174	135	170
61	screen	10	DoG	88.76	88.36	52.02	0.5069	135	170
62	screen	15	DoG	88.42	85.67	51.75	0.5088	135	170
63	screen	20	DoG	89.25	86.89	51.88	0.5041	3	170

Table C.2: This table shows the results of the homing method using Vardy's data set (see sections 5.8 and 5.8 and Appendix D for an explanation of the contents of the data sets). The type column shows the number of degrees above and below the horizon of the image which were used. See Table C.1 for the explanation of the other columns.

#	dataset	type	features	mean	median	std.dev.	score	best home	n
23	twilight	5	MSER	122.21	133.26	44.17	0.3211	136	170
34	twilight	10	MSER	73.55	65.58	51.23	0.5914	18	170
50	twilight	15	MSER	57.63	44.59	46.70	0.6798	153	170
68	twilight	20	MSER	46.21	34.90	39.72	0.7433	50	170
76	twilight	5	DoG	102.16	105.37	49.95	0.4324	6	170
82	twilight	10	DoG	101.25	102.97	50.17	0.4375	6	170
85	twilight	15	DoG	96.44	96.35	50.29	0.4642	5	170
97	twilight	20	DoG	91.66	89.34	49.59	0.4908	5	170
28	winlit	5	MSER	97.11	103.58	54.36	0.4605	136	170
47	winlit	10	MSER	78.69	72.12	51.96	0.5628	16	170
52	winlit	15	MSER	72.39	63.58	50.84	0.5978	50	170
77	winlit	20	MSER	52.39	39.58	44.07	0.7089	50	170
79	winlit	5	DoG	103.01	109.13	46.78	0.4277	134	170
87	winlit	10	DoG	104.93	109.14	46.08	0.4171	135	170
89	winlit	15	DoG	103.28	105.54	45.33	0.4262	34	170
90	winlit	20	DoG	98.86	99.65	44.11	0.4508	0	170

Table C.2: (continued)

#	dataset	type	features	mean	median	std.dev.	score	best home	n
1	square room	upper half	MSER	6,83	4,10	5,33	0,9621	138	3
2	square room	not filtered	MSER	9,65	12,03	7,84	0,9464	138	3
3	square room	not filtered	DoG	13,78	12,00	11,31	0,9234	138	3
4	robot lab	not filtered	Landmarks	14,88	10,16	14,86	0,9173	110	38
5	square room	lower half	DoG	14,94	14,52	10,75	0,9170	138	3
6	square room	lower half	MSER	20,62	25,27	8,49	0,8855	138	3
7	square room	upper half	DoG	20,91	18,96	6,64	0,8838	138	3
8	robot lab	lower half	MSER	21,96	11,09	30,05	0,8780	117	38
9	day	20	MSER	26,18	18,73	27,62	0,8545	17	170
10	robot lab	lower half	DoG	26,90	13,05	34,74	0,8506	117	38
11	robot lab	not filtered	MSER	27,84	16,03	35,51	0,8454	117	38
12	screen	20	MSER	28,64	18,42	31,04	0,8409	95	170
13	doorlit	15	MSER	30,69	19,35	33,38	0,8295	15	170
14	arboreal	20	MSER	34,89	23,39	35,49	0,8061	50	170
15	doorlit	20	MSER	35,41	21,27	38,52	0,8033	50	170
16	robot lab	not filtered	DoG	35,60	22,85	38,67	0,8022	117	38
17	arboreal	15	MSER	37,83	25,31	37,20	0,7898	17	170
18	day	15	MSER	39,78	29,30	36,49	0,7790	17	170
19	hall1	15	MSER	42,61	31,55	38,32	0,7633	159	200
20	original	20	MSER	43,18	32,23	38,49	0,7601	50	170
21	screen	15	MSER	45,71	33,75	40,43	0,7461	0	170
22	hall1	10	MSER	45,81	35,12	39,05	0,7455	41	200
23	twilight	20	MSER	46,21	34,90	39,72	0,7433	50	170
24	doorlit	10	MSER	48,45	33,95	43,90	0,7308	14	170
25	corridor	lower half	MSER	48,83	39,02	41,63	0,7287	203	6
26	arboreal	10	MSER	49,97	35,14	44,80	0,7224	153	170
27	robot lab	upper half	MSER	50,62	39,33	42,47	0,7188	117	38
28	winlit	20	MSER	52,39	39,58	44,07	0,7089	50	170
29	corridor	not filtered	MSER	52,66	35,71	44,89	0,7074	200	6
30	robot lab	upper half	DoG	56,14	45,77	43,84	0,6881	117	38
31	corridor	not filtered	DoG	56,26	44,58	43,64	0,6874	203	6
32	corridor	lower half	DoG	56,45	49,69	42,39	0,6864	203	6
33	corridor	upper half	DoG	57,08	38,19	45,65	0,6829	203	6
34	twilight	15	MSER	57,63	44,59	46,70	0,6798	153	170
35	hall1	20	MSER	58,49	48,71	44,53	0,6751	99	200
36	original	15	MSER	58,53	45,11	47,56	0,6748	17	170
37	chairs	20	MSER	58,92	45,23	47,55	0,6726	84	170
38	corridor	upper half	MSER	59,15	42,56	46,02	0,6714	199	6

Table C.3: This table shows the results of all data and contains the same information as tables C.1 and C.2, however the data in this table is sorted by score. Rows in light gray are the results of the IIIA data sets and the other rows are the results of Vardy's data sets. See tables C.1 and C.2 for an explanation of the other columns.

#	dataset	type	features	mean	median	std.dev.	score	best home	n
39	hall2	20	MSER	59,51	49,20	43,09	0,6694	18	200
40	hall2	15	MSER	61,00	50,90	45,30	0,6611	18	200
41	day	10	MSER	62,50	51,30	47,17	0,6528	17	170
42	hall1	5	MSER	62,69	53,82	44,78	0,6517	39	200
43	screen	10	MSER	63,43	54,30	45,02	0,6476	153	170
44	screen	5	MSER	66,71	52,61	50,03	0,6294	102	170
45	hall2	5	MSER	68,57	57,22	49,19	0,6190	19	200
46	original	10	MSER	72,09	62,67	50,85	0,5995	14	170
47	winlit	15	MSER	72,39	63,58	50,84	0,5978	50	170
48	day	5	MSER	72,67	62,94	50,31	0,5963	169	170
49	hall2	10	MSER	72,72	62,01	50,68	0,5960	19	200
50	twilight	10	MSER	73,55	65,58	51,23	0,5914	18	170
51	hall1	20	DoG	77,16	71,90	45,43	0,5713	0	200
52	winlit	10	MSER	78,69	72,12	51,96	0,5628	16	170
53	chairs	15	MSER	79,61	72,79	51,91	0,5577	16	170
54	doorlit	5	DoG	80,15	75,59	52,15	0,5547	169	170
55	doorlit	20	DoG	83,07	80,38	49,17	0,5385	151	170
56	doorlit	10	DoG	83,79	81,13	50,27	0,5345	169	170
57	chairs	10	MSER	84,42	82,22	50,74	0,5310	14	170
58	doorlit	15	DoG	84,44	81,34	49,75	0,5309	152	170
59	chairs	20	DoG	86,15	82,29	49,48	0,5214	3	170
60	screen	5	DoG	86,87	85,63	51,82	0,5174	135	170
61	screen	15	DoG	88,42	85,67	51,75	0,5088	135	170
62	screen	10	DoG	88,76	88,36	52,02	0,5069	135	170
63	screen	20	DoG	89,25	86,89	51,88	0,5041	3	170
64	hall1	15	DoG	89,27	85,64	45,78	0,5041	0	200
65	chairs	15	DoG	90,33	87,09	51,16	0,4981	4	170
66	arboreal	20	DoG	90,33	88,88	50,27	0,4981	4	170
67	original	20	DoG	91,36	88,78	49,70	0,4924	3	170
68	twilight	20	DoG	91,66	89,34	49,59	0,4908	5	170
69	day	10	DoG	92,99	94,81	51,31	0,4834	152	170
70	day	15	DoG	93,00	94,20	50,95	0,4833	135	170
71	day	20	DoG	93,05	93,13	50,42	0,4830	135	170
72	day	5	DoG	93,10	94,15	51,72	0,4828	152	170
73	chairs	10	DoG	93,46	92,24	51,88	0,4808	4	170
74	chairs	5	DoG	93,51	92,00	50,99	0,4805	5	170
75	arboreal	15	DoG	95,20	95,05	51,65	0,4711	4	170
76	twilight	15	DoG	96,44	96,35	50,29	0,4642	5	170
77	winlit	5	MSER	97,11	103,58	54,36	0,4605	136	170
78	original	15	DoG	97,93	97,66	49,81	0,4559	4	170
79	winlit	20	DoG	98,86	99,65	44,11	0,4508	0	170
80	arboreal	10	DoG	99,07	101,86	51,64	0,4496	135	170
81	arboreal	5	DoG	100,55	104,77	51,59	0,4414	135	170
82	twilight	10	DoG	101,25	102,97	50,17	0,4375	6	170
83	hall1	10	DoG	101,86	99,59	44,55	0,4341	40	200
84	original	10	DoG	101,98	105,32	50,19	0,4335	4	170
85	twilight	5	DoG	102,16	105,37	49,95	0,4324	6	170
86	doorlit	5	MSER	102,79	110,41	51,90	0,4290	14	170
87	winlit	5	DoG	103,01	109,13	46,78	0,4277	134	170
88	original	5	DoG	103,19	108,47	50,36	0,4267	50	170
89	winlit	15	DoG	103,28	105,54	45,33	0,4262	34	170
90	winlit	10	DoG	104,93	109,14	46,08	0,4171	135	170
91	hall1	5	DoG	108,85	109,20	46,09	0,3953	80	200
92	arboreal	5	MSER	112,73	122,76	48,51	0,3737	14	170
93	chairs	5	MSER	116,47	126,06	46,16	0,3529	14	170
94	hall2	5	DoG	116,47	130,14	49,96	0,3529	198	200
95	original	5	MSER	118,50	128,27	44,93	0,3416	153	170
96	hall2	20	DoG	122,09	132,21	44,38	0,3217	20	200
97	twilight	5	MSER	122,21	133,26	44,17	0,3211	136	170
98	hall2	10	DoG	124,42	137,90	45,31	0,3088	199	200
99	hall2	15	DoG	125,99	137,57	43,33	0,3000	61	200

Table C.3: (continued)

Appendix D

Vardy's Image Database

Vardy created a database of images (Vardy (2005); Vardy & Möller (2005)) which he used to test several homing methods, and is available at
`http://www.ti.uni-bielefeld.de/html/research/avardy/index.html`. Section 5.8 discusses the results of the ALV homing method which is used in this work with images from his database.

The images were taken in the robot laboratory and the hall of the Bielefeld University. They were created in a grid of 10 × 17 images in the robotic laboratory with 30 cm between them (horizontally and vertically); and in the main hall 10 × 21 images were created per data set with 50 cm between them.

An *ImagingSource DFK 4303* camera with a large wide-view hyperbolic mirror from Accowle Ltd. was used. The images were captured at the camera's highest resolution (752 × 564 pixels). Two examples of such images are shown in Figure 5.11.

Several data sets were created under different conditions, six in the robot laboratory and two in the main hall. Table D.1 shows the description of the data sets which are taken from Vardy & Möller (2005).

Name	Description
`original`	The standard or default condition of the room. Images were collected with both overhead fluorescent light bars on, the curtains and door closed, and no extraneous objects present.
`chairs`	Three additional office chairs were positioned within the capture grid.
`arboreal`	A tall (3 m) indoor plant was added to the centre of the capture grid.
`screen`	A projection screen (height 2.55 m) was added to the centre of the capture grid. The screen ran parallel to the long-axis of the room.
`day`	Images were collected with curtains open in full daylight.
`twilight`	Curtains and door open. Image collection began at 18:34 on 13 March 2004 just after sunset. At this time, the room was still receiving plenty of daylight. The last image was captured at 19:14. The sky outside was quite dark by this time.
`doorlit`	The light bar near the window was switched off.
`winlit`	The light bar near the door was switched off.
`hall1`	Collection area was adjacent to a dining area with two large open areas extending off in opposite directions and the large main hall extending out opposite the dining area. Overhead structures are visible and appear in significantly different positions within the collected images.
`hall2`	Collection area was near a message board at approximately the centre of the hall.

Table D.1: This table shows the descriptions of the different data sets , the first six were made in the robot laboratory and the last two in the main hall (Vardy & Möller (2005)).

Bibliography

Booij, O., Zivkovic, Z. & Kröse, B. (2006*a*), From images to rooms, *in* 'Proc. IROS Workshop From Sensors to Human Spatial Concepts', IEEE, pp. 53–58.

Booij, O., Zivkovic, Z. & Kröse, B. (2006*b*), Sparse appearance based modeling for robot localization, *in* 'Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems', IEEE, pp. 1510–1515.

Brooks, R. A. & Connell, J. H. (1986), Asynchronous distributed control system for a mobile robot, *in* 'Proceedings of SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering', Cambridge, MA, pp. 77–84.

Busquets, D. (2003), A Multiagent Approach to Qualitative Navigation in Robotics, PhD dissertation, Universitat Politècnica de Catalunya.

Carwright, B. A. & Collet, T. S. (1983), 'Landmark learning in bees: Experiments and models', *Journal of Comparative Physiology* **151**, 521–543.

Cobzas, D. & Zhang, H. (2000), 2d robot localization with image-based panoramic models using vertical line features, *in* 'Vision Interface 2000', pp. 211–216.

Cobzas, D. & Zhang, H. (2001), Mobile robot localization using planar patches and a stereo panoramic model, *in* 'Vision Interface 2001', pp. 94–99.

Edelman, S., Intrator, N. & Poggio, T. (1997), Complex cells and object recognition, unpublished manuscript, University of Cornell.

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.

Goldhoorn, A., Ramisa, A., de Mántaras, R. L. & Toledo, R. (2007*a*), Robot homing simulations using the average landmark vector method, Technical Report RR-IIIA-2007-03, IIIA-CSIC, Bellaterra.

Goldhoorn, A., Ramisa, A., de Mántaras, R. L. & Toledo, R. (2007*b*), Using the average landmark vector method for robot homing, *in* '19th International Conference of the ACIA', Vol. 163 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 331–338.

Gourichon, S., Meyer, J.-A. & Pirim, P. (2002), 'Using coloured snapshots for short-range guidance in mobile robots', *International Journal of Robotics and Automation* **17**(4), 154–162.

Hafner, V. V. (2001), 'Adaptive homing: Robotic exploration tours', *Adaptive Behavior* **9**(3/4), 131–141.

Hafner, V. V. (2003), Agent-environment interaction in visual homing, *in* S. Pierre, M. Barbeau & E. Kranakis, eds, 'Embodied Artificial Intelligence', Vol. 2865 of *Lecture Notes in Computer Science*, Springer, pp. 180–185.

Hong, J., Tan, X., Pinette, B., Weiss, R. & Riseman, E. (1991), Image-based homing, *in* 'Proceedings of the 1991 IEEE International Conference on Robotics and Automation', pp. 620 – 625.

Hubel, D. & Wiesel, T. (1962), 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *Journal of Physiology* **160**, 106–154.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R. & Wehner, R. (2000), 'A mobile robot employing insect strategies for navigation', *Robotics and Autonomous Systems* **30**(1-2), 39–64.

Lambrinos, D., Möller, R., Pfeifer, R. & Wehner, R. (1998), Landmark navigation without snapshots: the average landmark vector model, *in* N. Elsner & R. Wehner, eds, 'Proc 26th Göttingen Neurobiology Conference', Thieme-Verlag.

Lehrer, M. & Bianco, G. (2000), 'The turn-back-and-look behaviour: bee versus robot', *Biological Cybernetics* **83**(3), 211–290.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, *in* 'ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2', IEEE Computer Society, Washington, DC, USA, pp. 1150–1157.

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

Matas, J., Chum, O., Martin, U. & Pajdla, T. (2002), Robust wide baseline stereo from maximally stable extremal regions, *in* P. L. Rosin & D. Marshall, eds, 'Proceedings of the British Machine Vision Conference', Vol. 1, BMVA, London, UK, pp. 384–393.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. & Gool, L. (2005), 'A comparison of affine region detectors', *International Journal of Computer Vision* **65**, 43–72(30).

Möller, R. (1999), 'Visual homing in analog hardware', *International Journal of Neural Systems* **9**(5), 383–389.

Möller, R. (2000), 'Insect visual homing strategies in a robot with analog processing', *Biological Cybernetics, special issue: Navigation in Biological and Artificial Systems* **83**, 231–243.

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', AAAI Press / MIT Press, pp. 37–66.

Möller, R., Maris, M. & Lambrinos, D. (1999), 'A neural model of landmark navigation in insects', *Neurocomputing* **26-27**, 801–808.

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

Ramisa, A. (2006), Qualitative navigation using panoramas, Master's project, Universitat Autònoma de Barcelona.

Röfer, T. (1997), 'Controlling a wheelchair with image-based homing', *Spatial Reasoning in Mobile Robots and Animals, AISB-97 Workshop* pp. 66–75.

Shum, H. & Szeliski, R. (1997), Panoramic image mosaics, Technical Report MSR-TR-97-23, Microsoft Research.

Smith, L., Philippides, A. & Husbands, P. (2006), Navigation in large-scale environments using an augmented model of visual homing., *in* S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino & D. Parisi, eds, 'SAB', Vol. 4095 of *Lecture Notes in Computer Science*, pp. 251–262.

Stürzl, W. & Mallot, H. A. (2002), 'Vision-based homing with a panoramic stereo sensor', *Lecture Notes in Computer Science* **2525**, 620–628.

Tapus, A. & Siegwart, R. (2005), Incremental robot mapping with fingerprints of places, *in* 'Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', Edmonton, Canada, pp. 2429–2434.

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

Thrun, S. (2000), 'Probabilistic algorithms in robotics', *AI Magazine* **21**(4), 93–109.

Thrun, S. (2002), Robotic mapping: A survey, Technical report, Carnegie Mellon University.

Vardy, A. (2005), Biologically Plausible Methods for Robot Visual Homing, PhD dissertation, Carleton University.

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

Wan, A. S. K., Siu, A. M. K., Lau, R. W. H. & Ngo, C.-W. (2004), A robust method for recovering geometric proxy from multiple panoramic images, *in* 'ICIP', pp. 1369–1372.

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

Wehner, R. (1987), 'Spatial organization of foraging behavior in individually searching desert ants, cataglyphis (sahara desert) and ocymyrmex (namib desert)', *Experientia. Supplementum* **54**, 15–42.

Wilcoxon, F. (1945), 'Individual comparisons by ranking methods', *Biometrics Bulletin* **1**, 80–83.

Zeil, J., Hofmann, M. & Chahl, J. (2003), 'Catchment areas of panoramic snapshots in outdoor scenes', *JOSA-A* **20**(3), 450–469.