# HOMING IN SCALE SPACE

DAVID CHURCHILL

# Homing in Scale Space

by

© David Churchill

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

April 2009

St. John's Newfoundland

# Abstract

Local visual homing is the process of determining the direction of movement required to return an agent to a goal location by comparing the current image with an image taken at the goal, known as the snapshot image. One way of accomplishing visual homing is by computing the correspondences between features and then analyzing the resulting flow field to determine the correct direction of motion. Typically, some strong assumptions need to be posited in order to compute the home direction from the flow field. For example, it is difficult to locally distinguish translation from rotation, so many authors assume rotation to be computable by other means (e.g. magnetic compass). We present a novel approach to visual homing using scale change information from the Scale Invariant Feature Transform (SIFT) which we use to compute landmark correspondences. The method we describe is able to determine the direction of the goal in the robot's frame of reference, irrespective of the relative 3D orientation with the goal.

# Acknowledgements

The person I want to thank first is my M.Sc. supervisor Dr. Andrew Vardy. Without your knowledge, guidance, and patience this thesis would not have been possible. Your enthusiasm and passion for research motivated me to enter the field of robotics and have kept me excited about it ever since.

Thank you to the Natural Sciences and Engineering Research Council and Memorial University for providing financial support throughout my program.

The department of Computer Science is also owed many thanks for both its academic and financial support, as well as its wonderful faculty staff. Thanks to Elaine Boone, whose superhuman organizational skills have made my academic life possible up to this point, and for putting up with my constant onslaught of questions for the past 7 years. Thanks to head of the department Dr. Wolfgang Banzhaf for his sound academic advice. To Regina Edwards, Sharon Deir, and Darlene Oliver, without whom the department would surely fall apart. Thanks to the systems administrators Nolan White, Mike Rayment, Marrian Wissink, and Paul Price to which I owe all many hours of overtime pay. Also thanks to the many faculty members who have forgotten more about computer science than I will ever know, and encouraged me to continue on and pursue an M.Sc.

Many thanks to Dr. Gary Sneddon for his patience in teaching me 100% of the statistical methods present in this thesis and also how to implement them efficiently. Thank you to Dr. David Lowe, whose SIFT keypoint scale space data was the spark that ignited my research, homing in scale space would never exist without it. Thanks to Dr. Ralf Möller, whose insight into visual homing proved invaluable for much of my results and comparison section. And to Donald Craig, who was able to help me format this document into something worth presenting.

Thank you to all the friends and colleagues I have met over the years for your support. Just because your name is not here does not mean you are not appreciated.

Last but certainly not least I would like to thank my family for encouraging me to continue with my education. Specifically I would like to thank my Mother for her constant support, for always telling me to do what I loved, and for sacrificing so much just to get me where I am today :-)

# Contents

# List of Figures

# Chapter 1

# Introduction

Visual homing is the ability of an agent to return to a goal position by comparing the currently viewed image with an image captured at the goal, known as the snapshot image. It has been shown that insects such as bees and ants have the ability to visually home and that this is a crucial component in their overall navigational strategy [6]. Visual homing is also useful for general-purpose robot navigation in situations where image data is available. In this paper we propose a new visual homing method which is far less constrained than existing methods in that it can infer the direction of translation without any estimation of the direction of rotation, thus it does not require the current and snapshot images to be captured from the same 3D orientation.

We will first introduce the terminology and concepts related to visual homing, after which we will explore the various uses for visual homing, followed by a review of existing homing methods.

## 1.1 Visual Homing Concepts

As illustrated in figure 1.1, visual homing attempts to determine the direction of movement which will take a robot from a current position **cv** to a previously visited goal location **ss** without any knowledge of the relationship between the two locations. The process of visual homing relies solely on visual data taken from images acquired at both locations. While the main goal of visual homing is to determine the direction of motion, some methods also attempt to estimate the distance between **ss** and **cv**. Since image data used typically monocular, the process of estimating distance is difficult. If an agent is able to return directly to a goal location from its current location, that location is said to be within the *catchment area* of the goal. In biologically inspired homing literature, homing success is often measured by the catchment area [5].

Despite the accuracy achieved by existing visual homing methods, the calculated homing angle and distance to the goal always include an amount of error. This error usually results in visual homing being implemented as an iterative process. While actual implementations may differ slightly, the basic algorithm is as follows:

1. Acquire and process the snapshot image **SS**
2. Acquire an image **CV** from current robot location
3. Perform visual homing, calculating an estimated home direction $\theta_{homing}$
4. Calculate an estimated distance to goal $d_{homing}$, or choose a set distance
5. Rotate by $\theta_{homing}$, travel $d_{homing}$

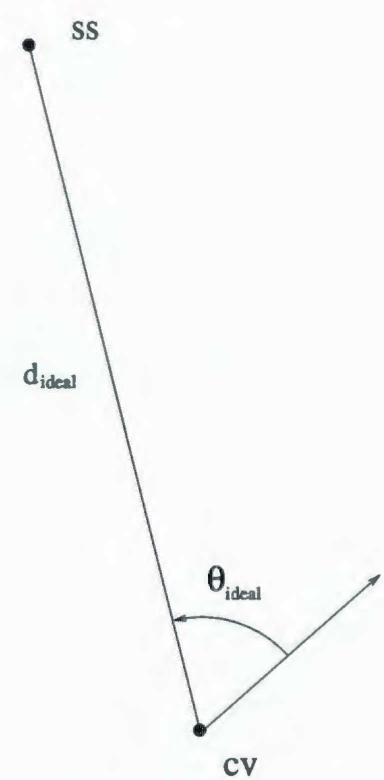



Figure 1.1: An illustration of the concept of visual homing. An agent (usually a robot), which is originally at a snapshot (goal) location **ss** is displaced and rotated to some current location **cv** (with pose given by the vector at location **cv**). Given an image taken at each location, visual homing attempts to determine the actual homing angle $\theta_{ideal}$ from **cv** to ss. In most cases, the robot pose at **ss** as well as the actual distance $d_{ideal}$ from **ss** to **cv** are unknown.

6. If the agent has reached the goal we have succeeded, otherwise return to step [2.]

In order to maximize the probability that the goal location can be seen from any other location within a given environment, many visual homing methods (including ours) utilize panoramic images. Encompassing a complete 360° view around the robot, these images afford us the best possible monocular view of our surroundings. A very important note about panoramic images is the existence of what is known as an *image horizon.* If a panoramic camera is attached to an agent which undergoes planar movement, there exists a horizontal line (usually in the center of that image) in which features do not undergo vertical translation. In a panoramic image which undergoes translation, features contract towards the direction of movement and expand away from the direction of movement. We can also define the image horizon as the locus of points on an image where the foci of expansion and contraction may be found, under the current motion constraints. For planar movement, this corresponds to a line of points along the center of the image (when image is taken parallel to the plane of movement). Concepts surrounding the foci of expansion and contraction will be further explored in chapter 1.3.1.

## 1.2 Uses for Visual Homing

Visual homing has various uses in the field of autonomous robotic navigation. Local visual homing deals with direct homing to a goal location which has a sufficient number of features in common with the robot's current location. If direct homing is possible to a goal from a given current location, that location is said to be within the goal's *catchment area*. Long-range homing however deals with situations in which the final goal location may either be obstructed, or not within the catchment area of the current location. In order to accomplish long range homing, Argyros et al [2] perform an initial exploration of the route, constructing a 'prior path' consisting of visual feature information. Selecting 'Milestone Positions' (MPs) by exploiting the information in its visual memory, the method attempts to select those MPs which guarantee the success of its local control strategy between consecutive MPs. Visual homing is used as the local control strategy between MPs which enables the larger scale learned path navigation to be successful. In [36], Vardy's biologically inspired approach uses visual homing as a means to navigate between intermediate goal positions in order to perform long-range visual homing. This method also starts by manually driving along the route it wishes to learn prior to autonomous navigation. At various points along the route, snapshot images are stored along with the associated odometric data. The approximate direction between successive snapshots is computed and stored in what is known as an odometry motion vector. Since odometry data alone suffers greatly from cumulative error, it is not sufficient information for the robot to accurately reproduce the entire route. A two-stage process is used for navigation between two intermediary route steps. First, the robot travels in the direction of the odometry motion vector with a distance slightly less than that of the vector's magnitude (to reduce the likelyhood of overshooting the goal). Next, visual homing is used to guide the robot to the stored snapshot of its intermediary goal, likely reducing errors caused by odometry. Once the estimated distance to the goal is within a given threshold, the process repeats until the final goal is reached.

Another use for visual homing is for travelling between the nodes of a topological map. Many robotic systems that are capable of navigation in unknown environments use topological maps (in the form of graphs) as their representation of the environment. The nodes of these graphs represent discrete locations in the environment. The *view graph model* proposed by Franz et al [10] utilizes a graph in which nodes represent discrete locations within the environment. Edges are said to be traversable in the graph if the nodes they connect are within each other's catchment area. Hubner et al [15] extend this idea by embedding the graph into three dimensional pose space, augmenting traversable edges with a vector describing the change in pose between nodes. They also enhance the local control strategy, combining obstacle avoidance, path integration, scene based homing, as well as topological localization [8].

## 1.3 Types of Visual Homing

Existing methods for visual based homing can be classified as either holistic or correspondence [27].

### 1.3.1 Holistic Methods

Holistic methods rely on comparisons between images as a whole. An example of a holistic method is the method of Zeil et al. who posit a simple distance metric between images and implement homing as gradient descent in the space of this distance metric [41]. This method, while elegant in its simplicity, relies on the existence of a monotonic relationship between image distance and spatial distance. It also requires small exploratory movements of the robot in order to determine the gradient of the image distance function. Möller and Vardy described an alternative method based on

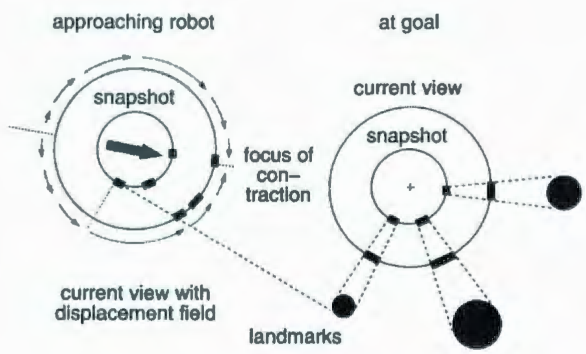



Figure 1.2: As the robot moves from the goal location, the surrounding landmarks seem to have displaced according to the arrows in the current view (left). If the robot moves in such a way that these displacements are minimized, it makes its way back to the goal. [4]. The warping method attempts to simulate these displacements by distorting the image based on several movement parameters [9].

gradient descent that removes the need for exploratory movements prior to computing a home vector [27].

Another holistic method is the so-called *warping method* of Franz et al. [9] which searches for the parameters of motion which make the warped snapshot image most similar to the current image. A warped snapshot image is generated by transforming the snapshot image *as if* the robot had actually moved according to the given motion parameters. To make this transformation tractable the assumption is made that all objects are equidistant from the goal. Given this assumption, the resulting flow fields $\delta(\theta)$ have the following form (derived from figure 1.3):

$$\delta(\theta) \quad = \quad \arctan\left(\frac{\rho\sin(\theta-\alpha)}{1-\rho\cos(\theta-\alpha)}\right)-\psi \tag{1.1}$$

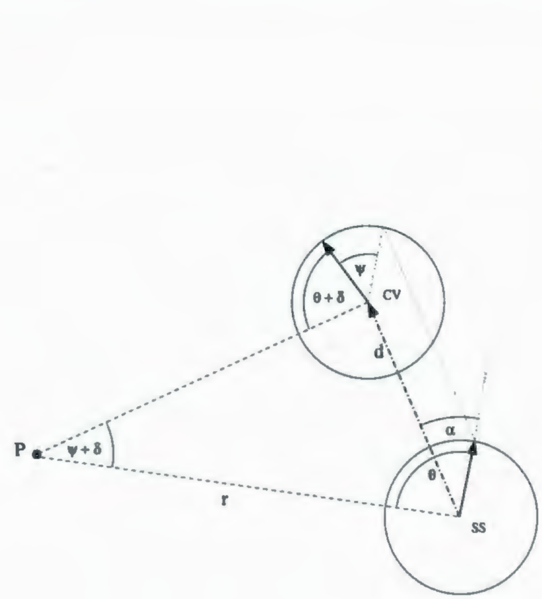



Figure 1.3: The direction of an image feature point P taken by a panoramic camera is denoted by angle $\theta$. We then displace the camera in direction $\alpha$ by distance $d$, and finally change its orientation by $\psi$. [9].

where $\theta$ is the position of a feature in the goal image, $\alpha$ is the direction the robot has moved away from the goal, $\psi$ is the change in sensor orientation, and $\rho$ is the ratio between the average landmark distance and the true distance to the goal. The snapshot is then warped by iterating over all possible values of our movement parameters ($\alpha$,$\psi$,$\rho$) in order to produce an image which matches the current snapshot image. When an image is found to be a suitable match, the direction $\alpha + \pi$ is chosen as the homing direction. The algorithm for this is as follows [9]:

```
WHILE image distance to snapshot > 0 {
   FOR all values of ψ, α, ρ DO {
      compute displacement field from equation (1.1)
      // note that warped snapshots are pre-computed
      distort snapshot with displacement field
      compute image distance to current view
```

```
    }
    select parameter set with closest match
    drive in direction α + π
}
```

Despite the clearly unrealistic nature of the assumption that all landmarks are of equal distance from the snapshot, the warping method has been found to perform robustly in various indoor environments [26]. In this paper we utilize the warping method to benchmark the performance of our algorithm.

### 1.3.2 Correspondence Methods

Correspondence based homing methods utilize feature detection and matching algorithms to form a set of correspondence vectors between the snapshot and current images. These vectors give the shift of the features in image space, known as the image flow field, as seen in figure 1.4. The flow field formed by these correspondence vectors is then interpreted to yield the direction of motion. In visual homing, these flow fields can comprise both robot translation as well as rotation. The separation of these two components of motion can be quite difficult. To illustrate this point, we will examine the image flow fields presented in figure 1.4. In the top image, the features which lie on the horizon undergo purely horizontal shifting under pure translation. If we were to add rotation of $\alpha$ to this image, it would have the effect of shifting all features horizontally by $\frac{\alpha w}{2\pi}$ pixels, where $w$ is the image width. If $\alpha$ is known, we simply subtract the rotation component from the vectors to obtain the translation component. If $\alpha$ is unknown, determining how much shifting to attribute to rotation

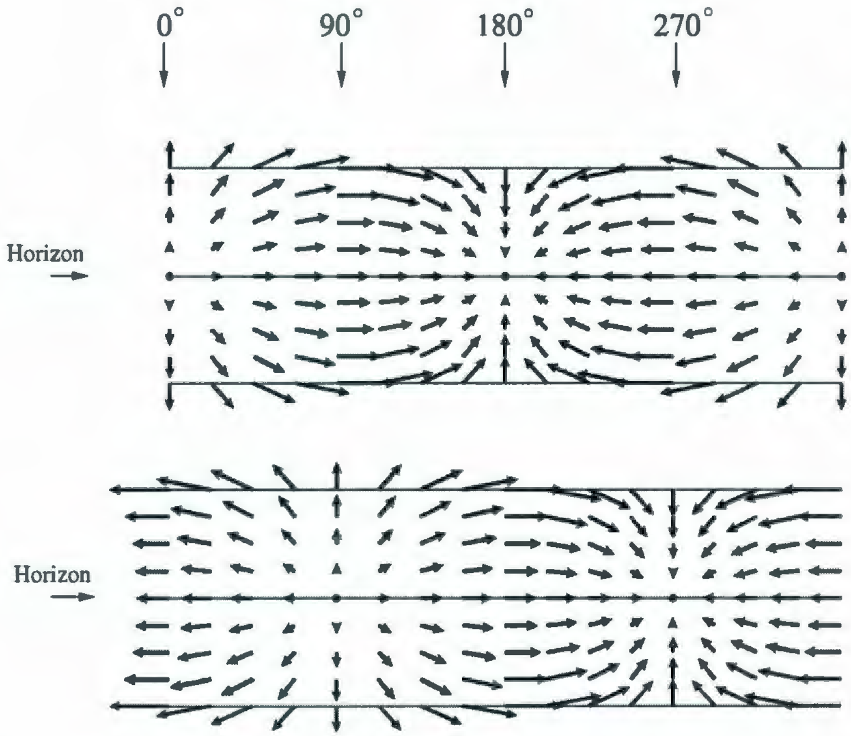



Figure 1.4: Ideal flow field for pure translation in a panoramic image [27]. Correspondence vectors are shown as arrows, and represent the change in feature positions within the image. The top image shows pure translation in the direction towards 0° and away from 180°. The bottom image shows translation in the direction towards 90° and away from 270°. The centers of expansion and contraction represent the direction of backward and forward motion respectively, and are ideally separated by 180°. The horizontal center line is the image horizon, and is only present if translation happens under planar movement. Features on the image horizon do not undergo any vertical shifting.

and how much to translation is much harder. Due to this difficulty, most correspondence methods have the additional assumption that all images have identical compass orientation prior to calculating homing direction. If both the snapshot and current images are taken from the same orientation in a planar environment it is possible to compute the home direction analytically from a single correct correspondence vector [38]. If the orientation is not the same, one can utilize some form of compass, or search for the change in orientation which would minimize the difference between the two images [41, 29].

One existing method of visual homing is that of Vardy and Möller [38]. Their method assumes all images have a constant orientation, and uses raw image windows as a correspondence detection method. Due to the simplicity of the windowing technique, many more matches can be attempted which compensate for lower matching accuracy. Once a set of correspondences is found, each of them is transformed into a unit homing vector via a trigonometric formula which is derived from the ideal flow field. These vectors are then summed to create a final homing vector which once normalized represents the estimated direction to the goal. While this method produces accurate results, it is constrained by the need for a stable image horizon in order for the formula to remain mathematically correct.

Various type of features have been utilized for determining correspondences, ranging in sophistication from raw image windows [38] to descriptors based on the Fourier-Mellin transform [30]. Other feature types which have been used are local features (such as corners) [40], distinctive landmarks [14], and high contrast features [4, 16, 19]. Recently, Scale Invariant Feature Transforms (SIFT) features have gained great popularity in many areas of computer vision and robotics due to the stability of their descriptor vectors with respect to changes in scaling, rotation, and illumination [21]. SIFT features have also been used to perform localization and visual homing [3, 29, 13, 34].

Pons et al [29] use SIFT landmarks in order to recover image orientation before implementing the vector based homing strategy of Vardy and Möller [38]. Their method uses a voting/matching scheme to minimize the horizontal component of the SIFT correspondence vectors. Any non-zero rotational component will extend the correspondence vectors to the left or right, raising the average horizontal length of the vectors. Two images are considered to have the same orientation when this length is minimized.

Briggs et al [3] deviate from the standard two-dimensional application of SIFT feature detection by utilizing one-dimensional representations of the environment in order to reduce processing time and memory. These one-dimensional images are formed by averaging the center scanlines from the two-dimensional panoramic images. Using the snapshot and current view images as the axes of a graph, images are matched using SIFT keypoints and the resulting correspondence curve is plotted. The direction of motion required to return to the goal is then extracted from this matching curve.

The method we will present in this thesis is similar to correspondence methods in that it relies upon finding correspondences between features. However, our interpretation of the resulting correspondences is markedly different. Consider the flow field for pure translation of an agent equipped with an omnidirectional camera. The field has a characteristic structure with foci of expansion and contraction separated by 180° (see Figure 1.4). If objects are distributed uniformly in the environment, half of them will appear to have expanded, while the remaining half will appear to contract. Typical correspondence methods consider how the features have shifted but not whether they have expanded or contracted. The problem is that in the presence of rotation it becomes much more difficult to determine the home direction from feature shifts. Hence, the two-stage process referred to above. However, whether a feature has changed in scale is independent of any change in orientation between the two views. We utilize the change in scale of corresponding SIFT features to determine the center of the *region of contraction* which corresponds to the home direction.

We will now give a detailed explanation the SIFT feature detection process in order to fully explain how our method utilizes them in order to perform visual homing.

# Chapter 2

# Scale Invariant Feature Transforms

The Scale Invariant Feature Transform (SIFT), developed by Lowe [21] is a robust image feature detection algorithm which is invariant to changes in image translation, rotation, and scale, as well as partially invariant to changes in illumination and 3D transformation. Features which the SIFT method detects are known as keypoints, and are described by a keypoint descriptor vector which contains image gradient information within a neighborhood of the keypoint. SIFT keypoints are detected and extracted via a four stage process.

The first stage of the process involves blurring the image by applying the Gaussian function $G$ with kernel size based on $\sigma$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.1}$$

The scale space $L(x, y, \sigma)$ of an image $I(x, y)$ is then the Gaussian function convolved $(*)$ with the image

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.2}$$

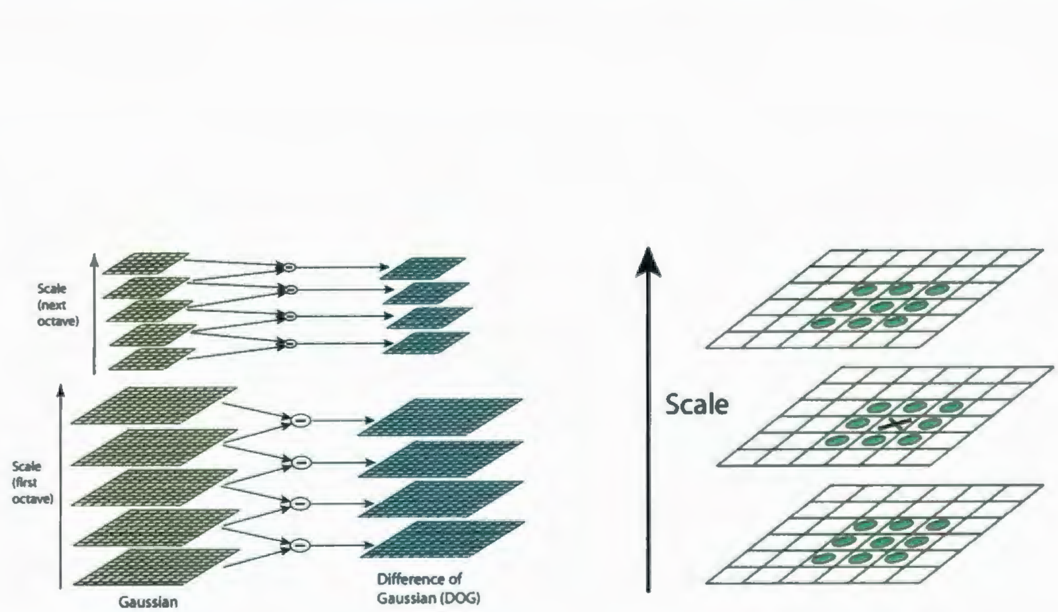



Figure 2.1: The original image is convolved with Gaussians to form layers of scale space (left). These layers are then subtracted to form the difference of Gaussian space. Local extrema are then detected in three dimensions within the DoG space (right) [21]

Lowe's algorithm detects candidate SIFT keypoints within the Difference of Gaussian space formed by taking the difference of two of these scale space images with values of $\sigma$ separated by a constant multiple k. That is,

$$
\begin{aligned}
D(x,y,\sigma) &= G(x,y,k\sigma) * I(x,y) - G(x,y,\sigma) * I(x,y) && (2.3)\\
&= L(x,y,k\sigma) - L(x,y,\sigma) && (2.4)
\end{aligned}
$$

Given an initial value for $\sigma$, we repeatedly apply $(G * I)$ with gradually increasing $\sigma$ in order to obtain the difference-of-Gaussian space images $D_1, D_2, ..., D_k$. Image half-sizing techniques can be utilized here for optimized performance, which are discussed at length in [21]. Local minima and maxima within the DoG space are then chosen as candidate keypoints. Each point in $D_n$ is compared to its 8 neighbours in $D_n$, as well as its 9 neighbors in $D_{n-1}$ and $D_{n+1}$, for a total of 28 neighbor comparison.

The second stage of SIFT localizes the keypoint within the image. Accurate interpolation of sub-pixel coordinates is done by fitting a 3D quadratic function to the local sample points [21]. Also in this step, keypoints are rejected for being in areas of low contrast or poor edge response.

The third stage of SIFT assigns an orientation to each keypoint within an image. By keeping the orientation assignment consistent with respect to the gradient within the image, keypoints can be stored with respect to their own orientation, maintaining rotation invariance among images. Since the scale $\sigma$ at which the keypoint was detected is stored, we use the image $L(x, y)$ with the closest value of $\sigma$ for orientation assignment, which maintains scale invariance. For each of these images, gradient magnitude and orientation are computed as

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.5)$$

$$\rho(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (2.6)$$

The final stage of SIFT then computes a descriptor vector based on an orientation histogram within a neighborhood of the keypoint. This neighbourhood is divided equally into $k_r$ regions, and the angles of the local gradients within these regions are summed into $k_b$ bins within the histogram. Figure 2.2 demonstrates an example keypoint descriptor with $k_r = 4$ and $k_b = 8$. The size of these neighborhoods are computed with respect to $\sigma$, the scale of the feature being detected. By doing this, we effectively scale the neighbourhood to the size of the keypoint, preserving scale invariance. We then store the vector with respect to $\rho(x, y)$ so that the descriptor vector is invariant to rotation within the image. Keypoints are matched by computing the sum of squared differences between their keypoint descriptor vectors. Once all four stages of the SIFT algorithm have been completed, we are left with a set of

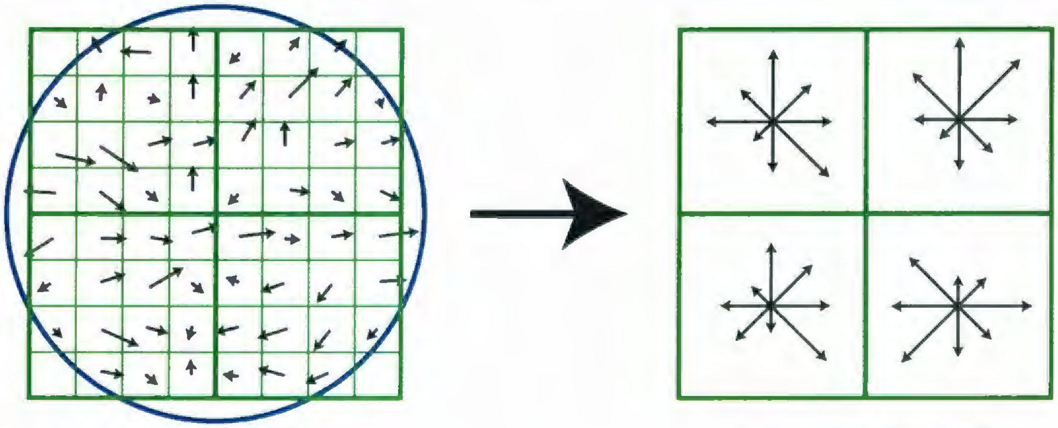



Figure 2.2: Local image gradients are combined into a histogram to form the SIFT keypoint descriptor vector [21]. The size of the neighbourhood is scaled with respect to $\sigma$ in order to maintain scale invariance.

keypoints of the form

$$\mathbf{f} = \{f_x, f_y, f_\sigma, f_\rho, \mathbf{f}_{kpd}\}, \tag{2.7}$$

where $\mathbf{f}_{kpd}$ is the keypoint descriptor vector with length $k_r \times k_b$.

## 2.1 Advantages

While Scale Invariant Feature Transforms have been used extensively for feature matching algorithms for many areas of robotics, for homing in scale space they are much more than just a feature matching technique. Not only do they provide very robust features for matching, they also give us the scale $\sigma$ at which they were detected. This scale space value is extremely important for homing in scale space, since it provides us with a measure of scale for the keypoint itself. By comparing the scales at which two matched keypoints are found, we can tell whether or not the feature has shrunk or grown. This method of comparing scales allows us to draw conclusions about the locations of the regions of expansion and contraction within the image.

The value of $\sigma$ stored within a keypoint is the scale at which the keypoint became a local minima or maxima in the Difference of Gaussian space. This is effectively the scale at which the feature had been 'blurred out of existence'. If we think of the value of $\sigma$ in these terms, then for two matched keypoints **a** and **b**, we can calculate the $\beta$ as:

$$\beta = a_\sigma - b_\sigma \tag{2.8}$$

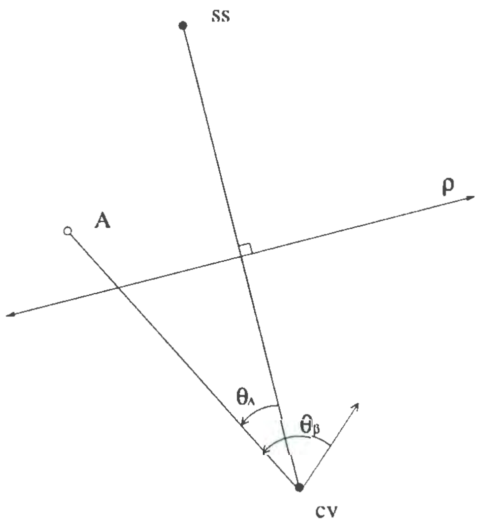
A value of $\beta > 0$ denotes a keypoint which is smaller in $b$ than in $a$. Conversely, a value of $\beta < 0$ denotes a keypoint which is smaller in $a$ than in $b$. While other methods have used SIFT has a feature matching algorithm, our method is the first to use the scale space information $\sigma$ for visual homing analysis.

# Chapter 3

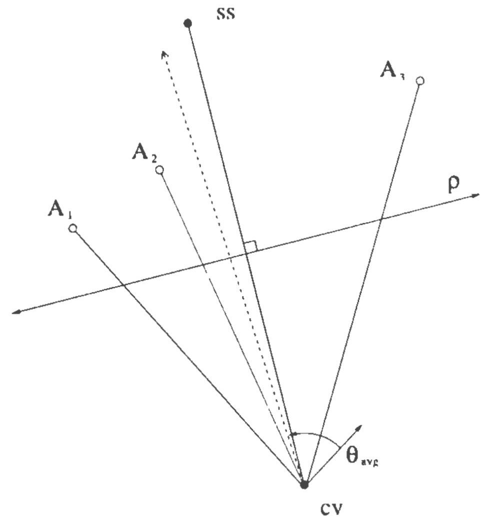
# Homing in Scale Space

Let **CV** represent the image taken at the location (**cv**) of current panoramic view from the robot's perspective, and **SS** be the image taken at the location (**ss**) of a stored panoramic snapshot taken from the goal location.

Consider the diagram shown in Figure 3.1 (a). If the robot has moved from position **ss** to position **cv**, the distance from the robot to feature A will have increased. This will be true of any feature on the same side of the perpendicular bisector $\rho$ of the line joining **ss** and **cv**. For any contracted feature A the feature's angle with respect to the line joining **ss** and **cv** $\theta_A$ will be governed by the constraint $|\theta_A| < \frac{\pi}{2}$, so a movement towards A will at least not take us any further from the goal. Since the poses of the robot at the two locations are arbitrary, only angle $\theta_\beta$ is known. For an ensemble of contracted features that are approximately uniformly distributed within the half-plane containing **ss**, the average of these angles $\theta_{avg}$ will point approximately towards **ss**. Therefore, heading in the direction of $\theta_{avg}$ as shown in figure 3.1 (b) will bring the robot closer to **ss**.




(a) Single contracted feature




(b) Multiple contracted features

Figure 3.1: Robot pose diagram showing the effect of contracted features during homing in scale space. Robot heading is represented by the vector at **cv**.

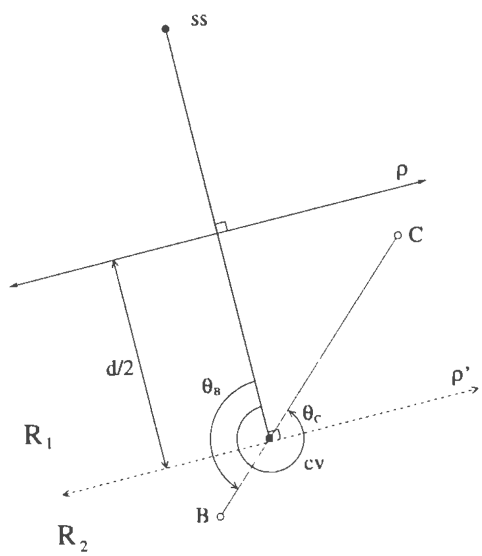



Figure 3.2: Robot pose diagram showing the effect of expanded features during homing in scale space.

For expanded features as shown in figure 3.2, the situation is somewhat different. Let $\rho$ be the perpendicular bisector of the line from **ss** to **cv**. Let $\rho'$ be the line through **cv** that is parallel to $\rho$. Consider two regions $R_1$ and $R_2$ as shown. If the robot moves *away* from a feature $B$ in $R_2$ then the distance to **ss** should decrease because $|\theta_B - \pi| < \frac{\pi}{2}$. A similar argument (to that proposed involving contracted features) suggests that the average bearing ($\pi$ subtracted) of a uniformly distributed set of features in $R_2$ would point directly away from **ss**. Therefore an opposite movement would lead us towards **ss**. Unfortunately, movements away from the expanded features in $R_1$ would carry the robot further from **ss**. Without knowing the position of **ss** we cannot determine whether an expanded feature is in $R_1$ or $R_2$. In many environments it will be likely that *cv* and *ss* observe features in $R_1$ from different sides, so $R_2$ will contain many more features than $R_1$. This will be true especially if the distance between **cv** and **ss**, $d$, is small. Therefore we will assume that movement away from the average angle of the expanded features will carry the robot towards **ss**.

Locating the centre of the region of expansion / contraction from **SS** to **CV** with respect to **CV** will allow us to determine which direction the robot must travel in order to reach the goal. Since **CV** is taken w.r.t. current robot orientation, not world orientation, we can then turn and move to approach the goal. We will use the change in scale information from SIFT feature correspondences to determine which features have contracted (as in figure 3.1) and which have expanded (as in figure 3.2). We then compute our estimated centers of expansion or contraction by taking the mean angle to each of the keypoints in each respective group. The result of this process is represented by angle $\theta_{avg}$ in figure 3.1 (b).

We use panoramic images of our environment to represent views from the robot's

(a) Scale decrease

(b) Scale increase

Figure 3.3: SIFT matched correspondences between **CV** (above) and **SS** (below). Correspondences in (a) show a scale decrease from **SS** to **CV**, thus having $\beta > 0$, indicating contraction. Conversely in (b) we see features which have $\beta < 0$, indicating expansion. Since these two regions should ideally be separated by $\pi$, they will be combined with a weighted average in order to more accurately compute the center of contraction.

perspective. These images are $w$ pixels wide by $h$ pixels high and represent a complete viewing angle of $2\pi$ in the horizontal direction, as well as $\alpha$ radians in the vertical direction. Each pixel represents a spacing of $\delta_x$ radians in azimuth, and $\delta_y$ radians in elevation, computable by:

$$\delta_x = \frac{2\pi}{w} \qquad \delta_y = \frac{\alpha}{h} \tag{3.1}$$

We therefore can convert our SIFT feature $\mathbf{f}$ with location $(f_x, f_y)$ within the images to angular coordinates $(f_{\theta x}, f_{\theta y})$ by

$$f_{\theta x} = f_x \delta_x \qquad f_{\theta y} = f_y \delta_y \tag{3.2}$$

in order to facilitate proper directional calculations.

Determining the center of the region of expansion or contraction requires detecting whether a feature has grown or shrunk with respect to its size in the snapshot image. If we revisit our SIFT feature vector, not only does it give us the location of a feature within an image, but also the scale $\sigma$ at which it was detected. Therefore, given a positive SIFT match between features $\mathbf{f_{ss}}$ and $\mathbf{f_{cv}}$ with scale values of $\sigma_{ss}$ and $\sigma_{cv}$ respectively, we can calculate:

$$\beta = \sigma_{ss} - \sigma_{cv}. \tag{3.3}$$

If $\beta > 0$ then the feature has shrunk from **SS** to **CV**, and conversely if $\beta < 0$ the feature has grown. We have many keypoint correspondences, so we must compute the center of these regions of expansion and contraction in order to find the home direction.

From the correspondences in Figure 3.3, we can see from the matches in (b) that the desks appear to be smaller in the snapshot, while the matches in (a) indicate that the filing cabinet seems to have grown. Since the cabinet represents the region of contraction in **CV**, this is the direction we wish to move. The central notion of our method lies in this fact: no additional interpretation of the flow field is required. Merely the sign of $\beta$ is enough to identify the change in feature size. The location of the corresponding keypoint within **SS** is not needed, since we are only concerned with the features in **CV** which have contracted. It remains for us to accurately locate the center of this region of contraction. Since the relationship between the angular orientation of **CV** and **SS** is not needed, our method achieves complete invariance to changes in relative orientation between the two images. Also, since this method does not rely on any notion of an image horizon, it is invariant to changes in relative 3D orientation and elevation. This claim will be satisfied if the following conditions hold: (1) the camera's field of view encompasses the true direction of translation, (2) a significant number of correct correspondences are found, (3) the corresponded features are approximately uniformly distributed throughout the environment, (4) $R_1$ does not contain a significant amount of features directly between **CV** and **SS**. This last constraint will almost always be satisfied since the images will have be taken of different sides of the same object.

Let us denote a matched feature pair $\mathbf{m} = (\mathbf{f}_{ss}, \mathbf{f}_{cv})$. To calculate the center of a particular region, we partition our set of correspondences $M = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\}$ into $M_{pos}$ and $M_{neg}$ based on the sign of $\beta$. To determine the center of these partitioned regions with respect to the robot's heading, we use the angular mean of the data, that is: given any set of angles $\theta_1, \theta_2, ..., \theta_n$:

$$\bar{\theta}(\theta_1, \theta_2, ..., \theta_n) = \arctan\left(\frac{\sum_{i=1}^{n} \sin(\theta_i)}{\sum_{i=1}^{n} \cos(\theta_i)}\right). \tag{3.4}$$

We will denote the angular mean of our partitions as $\bar{\theta}_{pos}$ and $\bar{\theta}_{neg}$ respectively. Note that the angular mean $\bar{\theta}_{neg}$ is the same as angle $\theta_\beta$ in figure 3.1. We argued in section 1 that the regions of expansion and contraction are separated by $\pi$ radians. We can use this fact to reduce the error in our calculation by allowing both the centers of expansion and contraction to contribute to the final result. Since both are ideally separated by a constant angle of $\pi$, $\bar{\theta}_{pos} = \bar{\theta}_{neg} + \pi$.

We wish to allow both regions to contribute in such a way that a certain amount of confidence can be given to either set of data. It is often the case that $|M_{pos}|$ is significantly greater than $|M_{negs}|$, or vice versa. In an effort to assign confidence to a partition, we will use its cardinality to perform a weighted average of the mean of the data. This will shift the final calculation in the direction of the region with the most correspondences. We can compute our final home angle $\theta_{homing}$ as follows:

$$\bar{s} = |M_{pos}| \sin(\bar{\theta}_{pos}) + |M_{neg}|(\sin(\bar{\theta}_{neg}) + \pi) \tag{3.5}$$

$$\bar{c} = |M_{pos}| \cos(\bar{\theta}_{pos}) + |M_{neg}|(\cos(\bar{\theta}_{neg}) + \pi) \tag{3.6}$$

and finally:

$$\theta_{homing} = atan2\left(\bar{s}, \bar{c}\right). \tag{3.7}$$

This value for $\theta_{homing}$ represents our final home vector with respect to the robot reference frame. Experimentally this weighted scheme has consistently shown to be more accurate than simply computing the unweighted average of the means of these regions. We summarize below our algorithm for determining the home direction:

1. Acquire and process the snapshot image **SS**

2. Acquire an image **CV** from current robot location.

3. Perform SIFT feature matching on **SS** and **CV** to obtain a set of $n$ matched feature pairs of the form $M = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\}$.

4. Partition $M$ into $M_{pos}$ and $M_{neg}$ where $pos, neg$ denote the sign of $\beta$ from equation 3.

5. Calculate the angular means $\bar{\theta}_{pos}$ and $\bar{\theta}_{neg}$ based on the values of $f_{0x}$ from $\mathbf{f}_{cv}$

6. Calculate the weighted angular mean of both $\bar{\theta}_{pos}$ and $\bar{\theta}_{neg} + \pi$ based on their cardinality as shown in equations 13-15.

7. Move the robot in the direction of the computed angle, $\theta_{homing}$.

# Chapter 4

# Experimental Methods

## 4.1 Image Databases

Several image databases were used for testing HiSS. The images were captured in an equally spaced grid by a panoramic camera affixed to the top of a robot (specified later). In figure 4.4 we can see samples of these panoramic images. In order to obtain the rectangular images required to perform both Homing in Scale Space, as well as the warping method [9], we used the unfolding algorithm described later in this section. Detailed information about each of the databases can be found in figure 4.1.

The A1OriginalH, CHall1H, and CHall2H databases were captured by Dr. Andrew Vardy at the University of Bielefeld using a Pioneer P3-DX robot. A1OriginalH was taken at the Robot Lab Computer Eng. Group at Bielefeld, while CHall1H and CHall2H are of the main hall of the university. Kitchen1H and Möller1H were captured by Sven Kreft and Sebastian Ruwisch, also using the Pioneer P3-DX robot [25]. These images were taken in a small kitchen setting and a living room setting

| Sample Image | Name | Pixels | Grid | Spacing |
|---|---|---|---|---|
|  | A1OriginalH | 561×81 | 10×17 | 30cm |
|  | CHall1H | 561×81 | 10×20 | 50cm |
|  | CHall2H | 561×81 | 8×20 | 50cm |
|  | Kitchen1H | 583×81 | 12×9 | 10cm |
|  | Möller1H | 583×81 | 22×11 | 10cm |
|  | ISLab | 346×50 | 9×8 | 61cm |

Figure 4.1: Detailed information for each of the six databases used.

Figure 4.2: The Pioneer P3-DX robot (left) and Pioneer P3-AT robot (right).

respectively. All of the objects in these databases remained stationary throughout the collection process. More details about these databases can be found in [39, 28]. See figure 4.2 for images of the robots used.

The ISLab database was captured by the author at the Intelligent Systems robotics laboratory at Memorial University using a Pioneer P3-AT robot. The setting for the database is a lab with an off white floor lit by fluorescent lighting. The area in which the robot operated was an open space of tiled floor measuring approximately 6 meters by 7 meters. Surrounding this area are various student workspaces consisting of desks, bookshelves, chairs, tables, and other robotic equipment (which can be seen in figure 4.3). Since it is an active laboratory, some of the images contain people who appear at different parts of the room in different images. This active setting provides for a more challenging environment for homing to take place, since some small features change locations between images. The floor of the lab is tiled by square tiles which measure 30.5×30.5cm. Images were captured at a grid equal to every second tile spacing. The image capture area is shown in figure 4.3.

## 4.2 Image Format

All images from the databases, as well as live robot trials are gray-scale panoramic images stored in portable gray map (PGM) format. Images are captured by a digital camera operating at 1024x768 pixel resolution which is pointed upward at a wide-angle hyperbolic mirror. Images taken by the camera are in a circular panoramic format as seen in figure 4.4. In order to perform visual homing these images will be transformed into rectangular coordinates by an image unfolding algorithm whose properties are

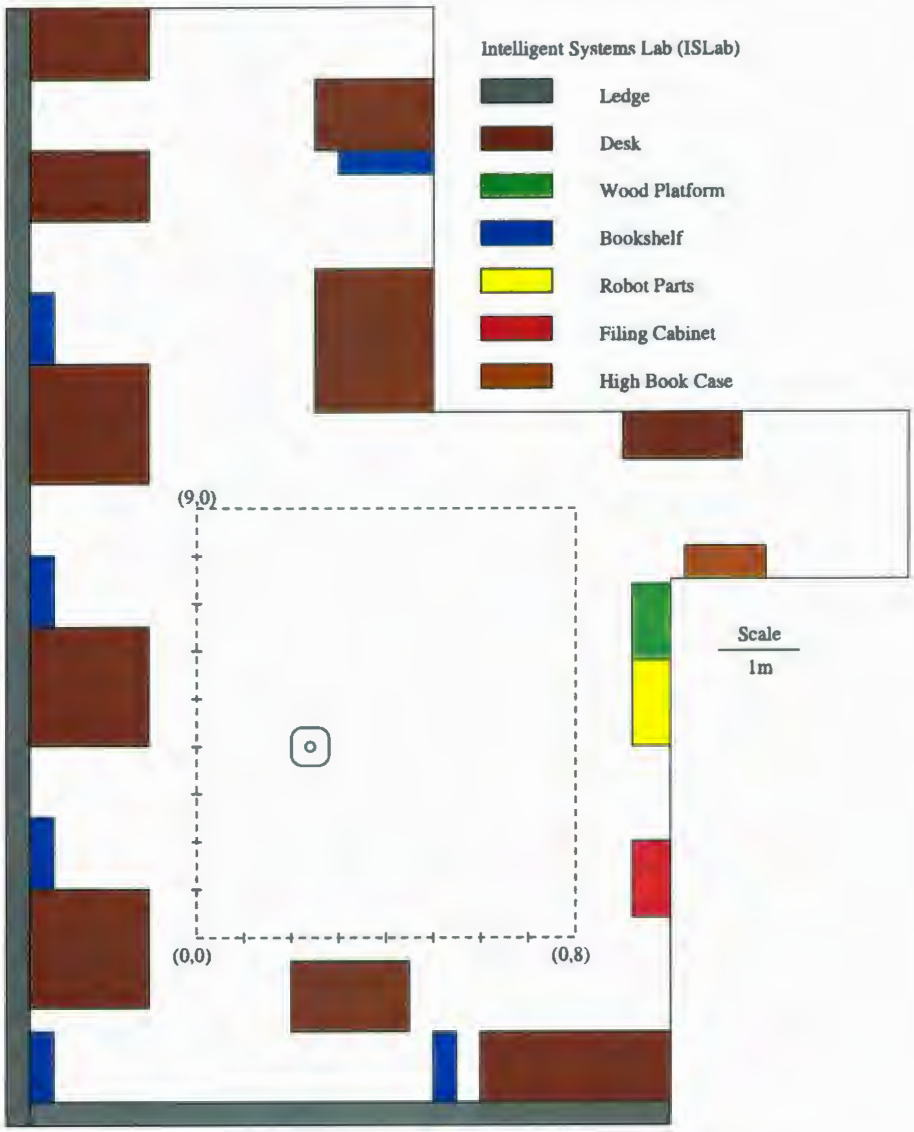



Figure 4.3: Diagram of the Intelligent Systems Lab at Memorial University of Newfoundland. The rounded square in the centre of the diagram represents the P3-AT robot.

Figure 4.4: Panoramic images before unfolding into rectangular images. These images were taken from the A1OriginalH, CHall1H, and CHall2H databases.

described in [37]. This algorithm samples points of the circular panoramic image such that each pixel of the output rectangular image represents a constant angular shift from the robot's point of view. This algorithm does this by using parameters given by the manufacturer which are specific to a given hyperbolic mirror. This process can be seen in figure 4.5.

One known issue with this sampling process is that it produces poor resolution from points sampled near the center of the circular panoramic image. The sampling rate $\delta_\theta$ remains constant, but for smaller values of $\rho$ near the center of the image the sampling area is much smaller, and the same pixels may be sampled multiple times. This results in a rectangular image which is much clearer near the top of the image than at the bottom.

In order to perform visual homing trials which are rotation invariant, our input images will be rotated parallel to the image horizon by a random amount before each test is performed. Since each image is panoramic, they will be rotated randomly by $\theta \in [0, 360°)$. To simulate this angular rotation, we will shift each image (with width $w$ and height $h$) randomly to the right by $w_r \in [0, w-1]$ pixels. For some

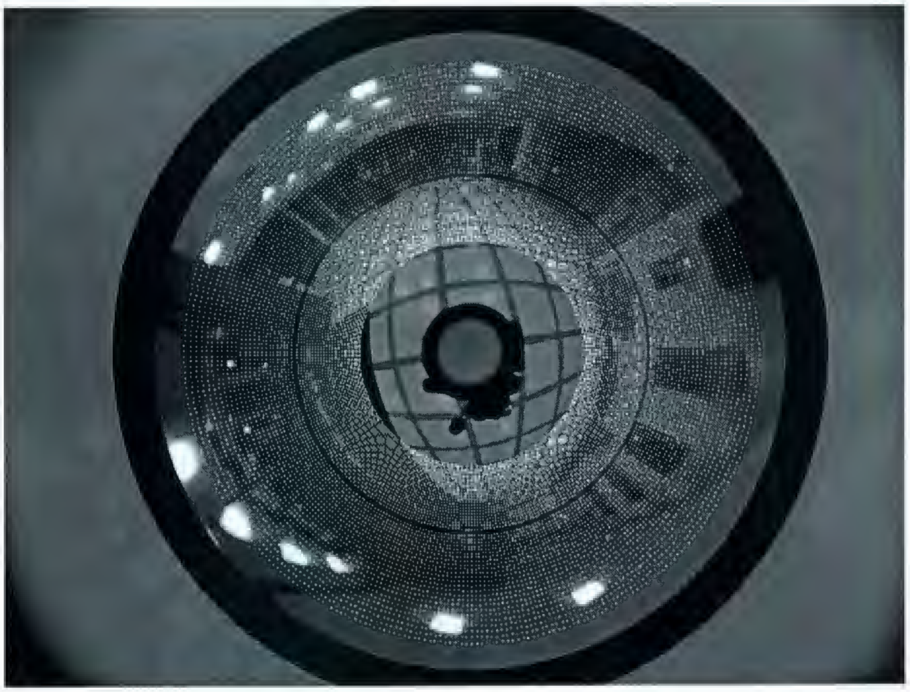


Figure 4.5: A sample image being converted from panoramic to rectangular view. The sample points selected by the unfolding algorithm in [37] are shown as white dots. The points are selected based on the parameters of the mirror which was used in the taking of the image.

Figure 4.6: Images from the A1OriginalH database taken at location (1,1). Top image shows the original image taken by the robot. Bottom image shows the image after a random amount of rotation, plus a random vertical shift. The remaining pixels after vertical shifting are filled in with black.

experiments we will also be simulating a random vertical shift within an image by randomly displacing it by some value $w_r \in [-h_r, h_r]$ where $h_r \in [0, h/2]$. Unlike horizontal shifting, vertical shifting will leave some portion of the image undefined, which we will fill in as black pixels. This vertical shifting will be done in order to test the robustness to shifts in the image horizon in both visual homing methods. This process provides only a crude emulation of elevation change, since it does not simulate the perspective change which would occur if the images had been taken at varying elevations. It is sufficient however to test robustness to shifts in the image horizon.

## 4.3 Image Database Trials

Implementation of the database trials for both homing in scale space as well as warping was done in C/C++. The main test framework for image reading and transformations, as well as data logging was programmed completely in C. David Lowe's SIFT implementation [21] was used for the detection and construction of SIFT keypoints for use in homing in scale space. For the warping method, we used Dr. Ralf Möller's warping method implementation written in C++.

Using each location in each database as a goal location, we applied both HiSS as well as the warping method from each other location in the database as a current location. In this way we were able to exhaust each possible homing scenario from each database. Results were then stored in separate files for later statistical analysis. All statistical analyzes were carried out using the R statistics software package [1].

Our method relies on a number of SIFT feature matches within an image in order to compute the center of a region of expansion and contraction, therefore we require as many keypoints as possible for this process. Fortunately, SIFT feature matching offers a number of parameters which can be changed in order to maximize keypoint production, while still maintaining accurate results [21]. The values changed from those of Lowe's original implementation are as follows:

1. The number of scales at which keypoints are extracted is increased from 3 to 6 to increase the number of overall keypoints, while maintaining feasible running time

2. The peak threshold for the magnitude of difference of Gaussian values is decreased from 0.08 to 0.01 in order to choose more keypoints from areas of low

contrast, since indoor environments often contain such areas

3. The ratio of scores from best to second best SIFT matching has been decreased from 0.6 to 0.8. As discussed in [21], this change results in a marginal decrease in match accuracy while dramatically increasing the number of matches.

Parameters for the warping method were selected to ensure fairness with respect to running time. We selected the following values for the parameters of the warping method search space: $RhoMax = 0.95$, $RhoSteps = 36$, $AlphaSteps = 36$, and $PsiSteps = 36$ [11]. These parameters represent the granularity of the space of the warping search, and correspond directly to the similarly named variables in equation 1.1, as well as the warping algorithm outline. On an Intel Core2 2.13GHz processor, this parameter selection resulted in an average execution time for the warping method which was 4.8% faster per snapshot than our scale space method. We consider this to be a fair metric for results comparison.

## 4.4 Live Trial Implementation

Live robot trials were conducted using the Pioneer P3-AT robot [23] at the Intelligent Systems Lab at Memorial University. The environment for the live trials was exactly the same as described for the collection of the ISLab database. To implement visual homing on the live robot we used the Advanced Robotics Interface for Applications (ARIA) [22] robotics sensing and control libraries. ARIA is an object oriented API which provides complete control of all functions of the robot without the need for low level hardware programming. Using the MobileSim [24] software package, we were able to perform simulations of homing in scale space in a virtual environment before implementing them on a live robot. This provided for a safe, controlled way to perform debugging and analysis before performing homing in a real lab.

Utilizing ARIA on the robot allowed us to construct a simple text based interface for robot navigation and visual homing trials. Utilizing ARIA's 'safe mode' capabilities, the robot's built-in ultrasonic sensors as well as laser range finders were used to stop the robot in case of imminent collisions with nearby objects in the environment. Odometry data was used during the experiments solely to turn the robot and move it forward by a certain distance after homing calculations were complete, no localization data was stored on the robot throughout the homing trials.

Five live robot trials were conducted, each of which had a fixed goal location in the environment. For each of these trials, five separate test locations were chosen in the environment to test visual homing to that particular trial's goal location. For each of these 25 tests, the following process was carried out:

1. Place the robot at the goal location for the trial.
2. Initiate capture and processing of the snapshot for the current goal location. For HiSS, all SIFT keypoints are computed and stored. For warping, the lookup table is constructed.
3. Manually shift the robot to the desired starting location for homing
4. Initiate the autonomous homing process.
5. Current view image is taken, compared to the snapshot image, and the homing angle and estimated distance to goal are computed.

6. Using a two-step controller, the robot first turns to match the direction of $\theta_{homing}$ and then travels the estimated distance to the goal.

7. The robot waits until a key is pressed by the user. This step was introduced in order to facilitate the manual recording of the position of the robot within the environment.

8. If the robot's new goal distance estimation is within the threshold, it ends the homing process. Otherwise, return to step 5.

9. The test was allowed to run for a maximum of 12 iterations of steps 5-8 before manually stopping the process.

# Chapter 5

# Results

In this chapter we will show and discuss the results collected from the simulated database homing trials as well as the live robot trials. We will describe the performance metrics used to compare methods, show some sample results, and finally compare the methods using statistical analysis.

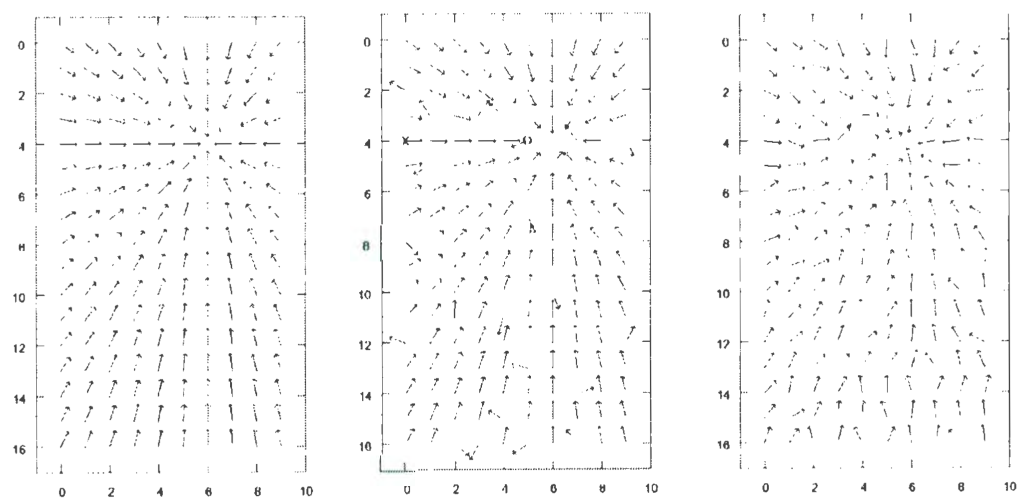
Given two images **SS** and **CV**, a visual homing algorithm computes $\theta_{homing}$, the direction needed to move in order to reach **ss** from **cv**. The robot will then move in the direction of $\theta_{homing}$ and determine whether or not it has arrived at the goal. In order to properly measure the accuracy of a given homing algorithm, we will require two performance metrics [38]. The first metric, known as the angular error, is the difference between $\theta_{homing}$ and the true homing direction $\theta_{ideal}$. The second metric is the return ratio, which measures the number of times the robot was able to successfully navigate to the goal location. Both of these methods are needed due to the fact that while the correlation between angular error and return ratio is strong, a higher angular error does not *always* result in a lower return ratio. It could be the case that one method has an angular error which is 10° higher than another, yet results in the a better return ratio, as illustrated in figure 5.1. It has been shown [11] that as long as the angular error remains under 90°, the robot will eventually converge to the goal location.

Each test was performed using both homing methods. Wherever 'H' or 'HiSS' is noted in a legend or table, it represents the results for the homing in scale space method. Wherever 'W' or 'Warp' is noted in a legend or table, it represents the results for the warping method. Since all tests were done with a certain level of vertical shifting, wherever 0px, 5px, 15px, or 24px is noted, it corresponds to the maximum random vertical shift for that particular trial. For example, '15H' or 'HiSS15' both refer to a trial performed by the homing in scale space method under 15 pixel maximum vertical shift.

## 5.1 Performance Metrics

The first metric for performance evaluation we use is the average angular error between correct home vectors and our computed home vectors. Our homing method takes images located at **cv** and **ss** and returns $\theta_{homing}$, the angle we compute to be the homing angle. Since images in the database were taken at known locations, we can compute the ideal home vector angle as follows: given the $(x, y)$ locations of current view **cv** and a goal snapshot **ss** on an evenly spaced capture grid, we can compute

$$\theta_{ideal}(\mathbf{ss}, \mathbf{cv}) = atan2(\mathbf{ss}_y - \mathbf{cv}_y, \mathbf{ss}_x - \mathbf{cv}_x) \tag{5.1}$$



(a) Ideal (b) Impulse (c) Gaussian

Figure 5.1: Under ideal conditions (a), $AAE_{(6,4)} = 0$ and $RR_{(6,4)} = 1$. In (b) we add high levels of random error to just a few of the vectors, resulting in $AAE_{(6,4)} = 0.165$ and $RR_{(6,4)} = 0.889$. Finally, in (c) we add a small amount of Gassian error to each homing vector resulting in $AAE_{(6,4)} = 0.255$ and $RR_{(6,4)} = 0.986$. Even though the angular error is higher in (c), it performs better than (b) using the return ratio metric.

thus, the angular error AE(**ss**,**cv**) can be found by:

$$AE(\mathbf{ss},\mathbf{cv}) = diff(\theta_{ideal} - \theta_{homing}) \tag{5.2}$$

where diff() is the true angular difference, implemented by the function

```
double diff(double a, double b)
   double diff = a - b;
   while (diff < -π) diff += 2π;
   while (diff >  π) diff -= 2π;
   return diff;
```

We can then obtain an overall average angular error as follows ($AE(\mathbf{ss},\mathbf{ss}) = 0$):

$$AAE(\mathbf{ss}) = \frac{1}{mn}\sum_{x=1}^{m}\sum_{y=1}^{n} AE(\mathbf{ss},\mathbf{cv}_{xy}). \tag{5.3}$$

Finally, to obtain a measure of performance for the entire image database we can use the total average angular error over a given database of images **db**, TAAE(**db**), which computes the overall average of AAE(**ss**) having computed the home direction to each possible location as the snapshot:

$$TAAE(\mathbf{db}) = \frac{1}{mn}\sum_{x=1}^{m}\sum_{y=1}^{n} AAE(\mathbf{ss}_{xy}). \tag{5.4}$$

While the average angular error gives us a measure of how accurate the homing vectors are in relation to the ideal homing vectors, it does not actually tell us whether or not the robot will reach the goal position. For this, we need a second performance metric: the return ratio. Before we discuss the return ratio, we first need to define the concept of 'successful' homing. Within an environment, we say a particular attempt at homing was a success if the agent was able to return to within a given distance threshold of the goal location from its current location. Given an image database, we will determine a binary quantity indicating return success from **cv** to **ss**, $RET(\mathbf{CV}, \mathbf{SS})$ with the following method:

1. Given images **CV** and **SS** from a grid of images with locations ($\mathbf{cv}_x$,$\mathbf{cv}_y$) and ($\mathbf{ss}_x$,$\mathbf{ss}_y$) respectively, calculate $\theta_{homing}$.

2. Calculate $\mathbf{cv}_{new} = (\mathbf{cv}_x + \cos(\theta_{homing}),\mathbf{cv}_y + \sin(\theta_{homing})$. This is the adjacent grid cell in the direction indicated by $\theta_{homing}$.

3. If $\mathbf{cv}_{new} = \mathbf{ss}$, the homing trial is considered successful. If $\mathbf{cv}_{new}$ is outside the boundary determined by the capture grid, or is the same as a **cv** which has already been visited this trial (i.e. a loop has been detected), homing is considered to have failed. Otherwise, return to step with $\mathbf{cv} = \mathbf{cv}_{new}$.

If we iterate this process using each of the locations within the database as the goal location, attempting to return to it from each of the other locations within the database, we can determine the total return ratio $TRR(\mathbf{db})$ as the percentage of successful homing attempts. If we define $RET(\mathbf{CV}, \mathbf{SS})$ as 1 for success, and 0 for failure, we can then calculate the return ratio and total return ratio as

$$RR(\mathbf{ss}) = \sum_{x=1}^{m}\sum_{y=1}^{n} RET(\mathbf{cv}_{xy}, \mathbf{ss}_{xy})/mn \quad (5.5)$$

$$TRR(\mathbf{db}) = \sum_{x=1}^{m}\sum_{y=1}^{n} RR(\mathbf{ss}_{xy})/mn. \quad (5.6)$$

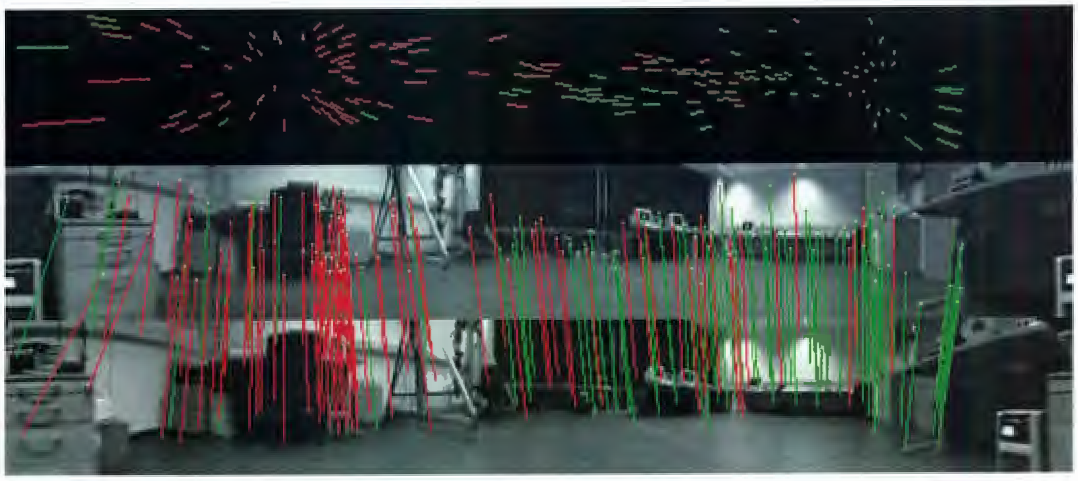


Figure 5.2: Sample correspondence vector image. The tail of the vector is denoted by a small white dot. We can see the flow field (top), **CV** (middle) and **SS** (bottom). Red and green vectors indicate contraction and expansion respectively.

## 5.2 Sample Results

Figures 5.2 and 5.3 are of typical correspondence vector sets generated between a current view and snapshot image. These two figures have been captured at the same orientation. The middle image represents the current view, bottom image represents the snapshot image, while the top image shows the overlaid vector field image. Red vectors represent the contracted features while green vectors represent expanded features.

Figure 5.3 shows the weighted averaging scheme implemented by homing in scale space. The yellow square (a) represents the computed center of the region of contraction $\bar{\theta}_{pos}$ while the blue square (b) represents the center of the region of expansion $\bar{\theta}_{neg}$. The gold square (e) shows the value of $\bar{\theta}_{pos} + \pi$, which ideally (see figure 1.4)

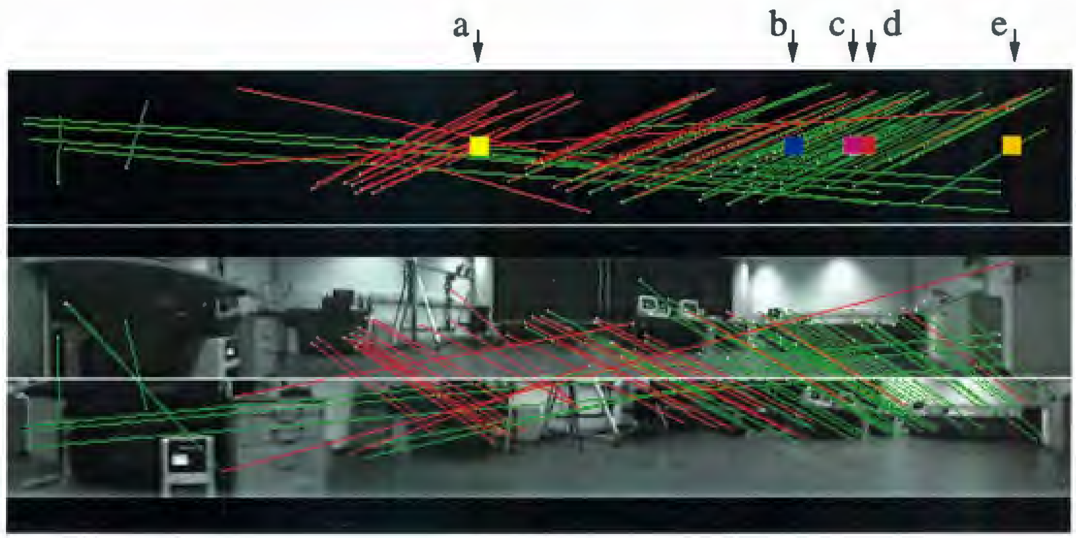



Figure 5.3: Illustration of the weighted mean process which takes place in homing in scale space. (a) $\bar{\theta}_{pos}$ (b) $\bar{\theta}_{neg}$ (c) $\theta_{ideal}$ (d) $\theta_{homing}$ (e) $\bar{\theta}_{pos} + \pi$

should be in the same location as $\bar{\theta}_{neg}$. Due to the higher confidence placed in the larger set of green vectors $M_{neg}$, the value for the computed homing angle $\theta_{homing}$ (d) is pulled closer to $\bar{\theta}_{neg}$, which (in this case) is closer to the true homing direction $\theta_{ideal}$.

In figure 5.4 we see the results of homing to location $(2, 3)$ in the A1OriginalH database from every other location in the database. Computed homing angles are represented by unit vectors in the diagram. These homing vector fields give a very good representation of the overall performance of homing to a particular location. The sign of a good homing trial is when a homing vector field looks very similar to the ideal homing vector field (i.e. all vectors point directly at the goal position).

Figures 5.6 and 5.7 present grayscale grids plotted for each $(x, y)$ location within each database. The gray scale value for a particular location within a database is

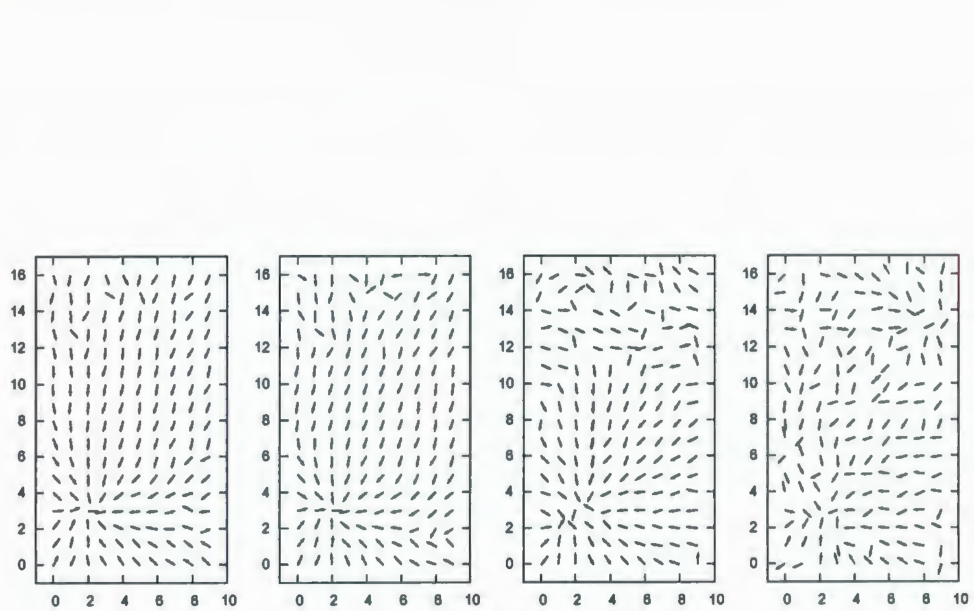


Figure 5.4: Homing vector images with goal position set to (2, 3). First two images show our HiSS method with horizontal shift only (first, AAE=12.3°) and combined vertical shift (second, AAE=18.1°). Last two images show warping method for horizontal shift only (third, AAE=39.2°) and combined vertical shift (fourth, AAE=59.4°). Note the dramatically reduced accuracy loss for the HiSS method over the warping method under simulated vertical shift.

Figure 5.5: Sample SIFT keypoints (located at base of arrow) including scale (length of arrow) and orientation (direction of arrow). Image taken from A1OriginalH database. Parameters were changed to detect fewer keypoints in this image for illustration purposes only.

scaled from black (0) to white (maximum of $TAAE(hiss)$ and $TAAE(warping)$ for the particular database). This view allows us to see which locations in a particular environment perform well (darker), or poorly (lighter). Keep in mind the aspect ratio for these figures is not 1:1, refer to the axes for coordinate information.

## 5.3 Angular Error - Return Ratio

We present the angular error and return ratio results for all database trials in figures 5.8 and 5.9 respectively.

In the case of the return ratio metric our data shows that for each test, HiSS has a higher percentage success rate than that of the warping method. We must now show that these percentages are statistically significant. To do this we will use the test of proportions [7], which tests to see if given proportions (in our cases, successes or failures) are considered to be statistically the same, or different. Most statistical tests output what is known as a P-value. This P-value is the probability of obtaining a result which is at least as extreme as the one observed, given the null hypothesis is true. Thus a sufficiently small P-value suggests that the null hypothesis (no statistically significant differences exist) should be rejected. Since we are interested in whether or not HiSS is more successful than warping, we will use the alternative hypothesis that HiSS > warping. To implement this test we use the R stats package prop.test() function. Upon running this test for each of the trials listed in figure 5.9, the P-value we obtained was $p < 2.2e - 16$. Since the alternative hypothesis is considered to be true for values of $p < 0.05$, this result confirms that HiSS does indeed outperform warping with respect to return ratio.

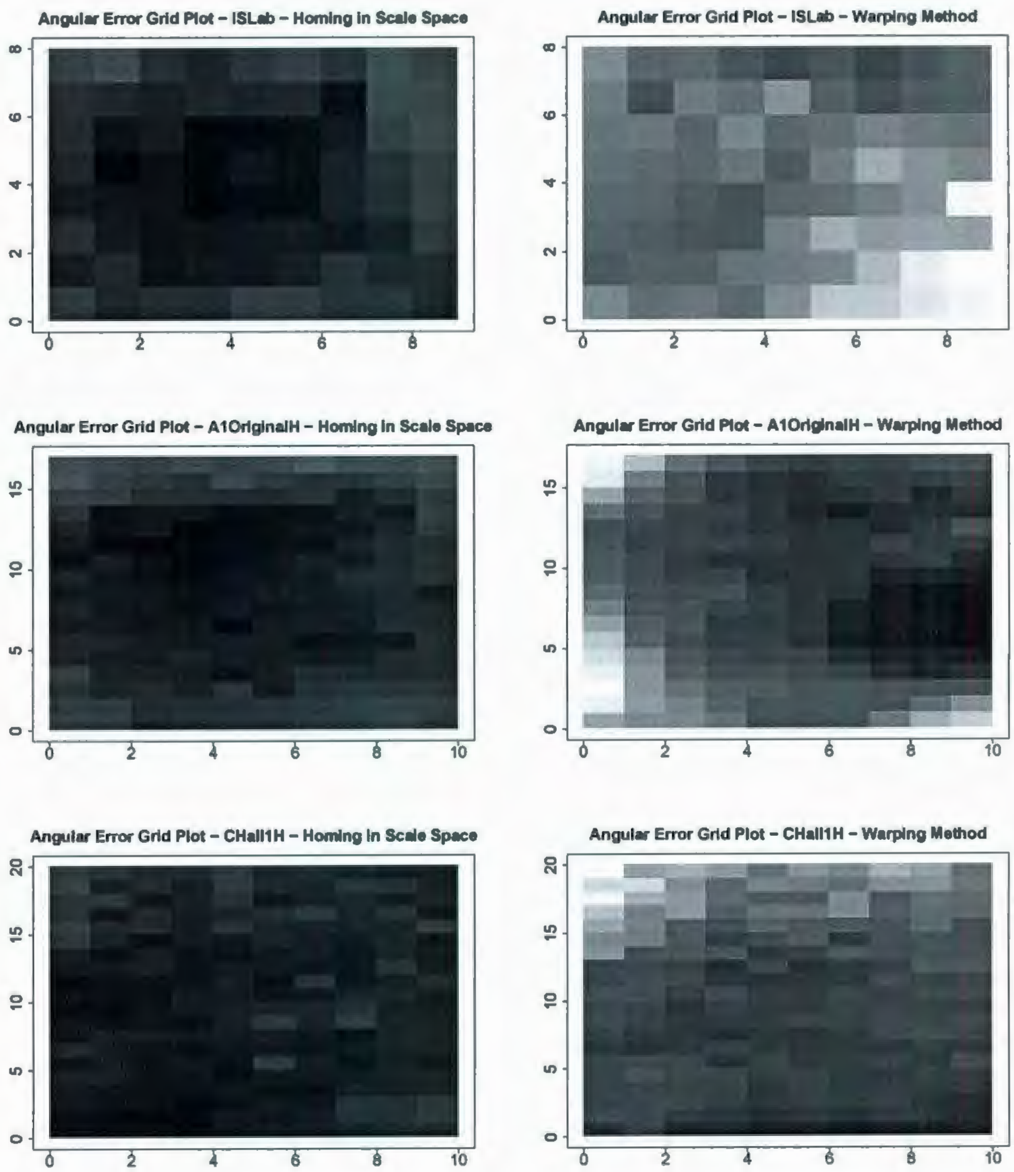



Figure 5.6: Grids showing TAAE results for ISLab, A1OriginalH, and CHall1H databases. (darker is better)

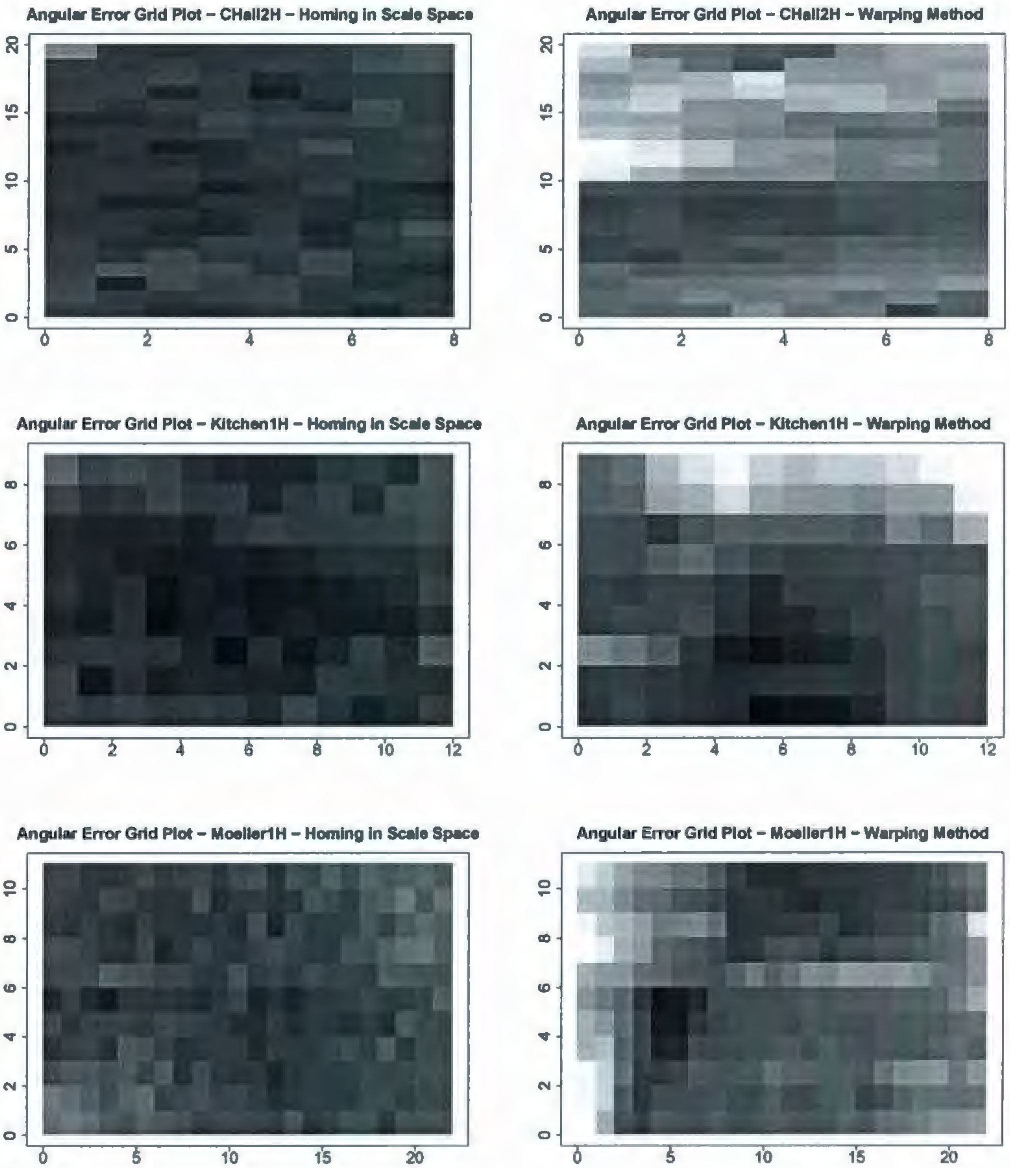
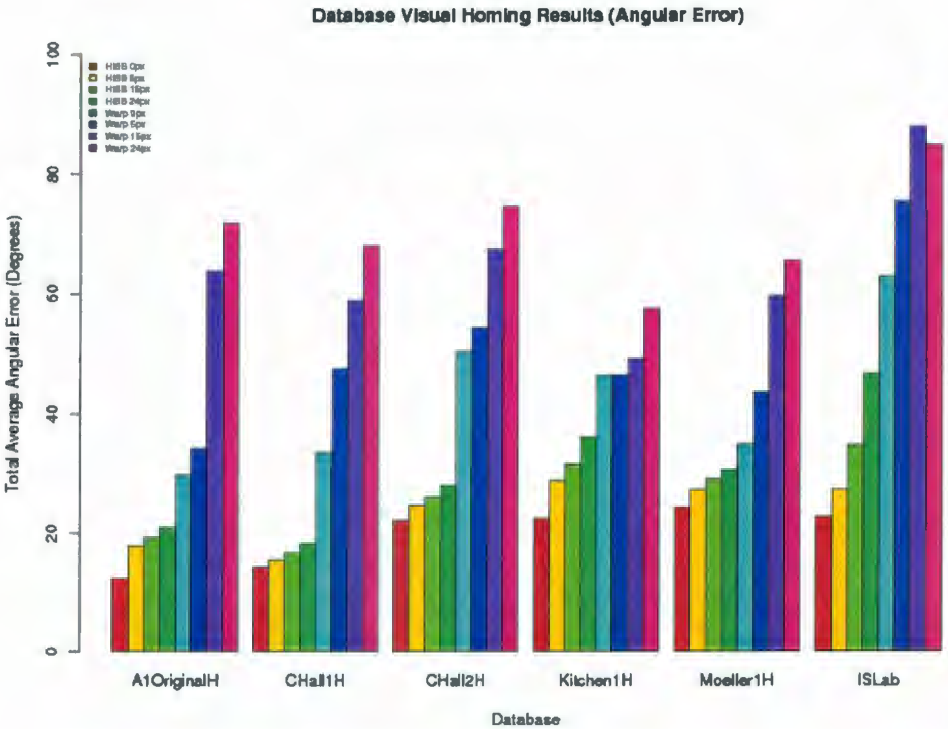
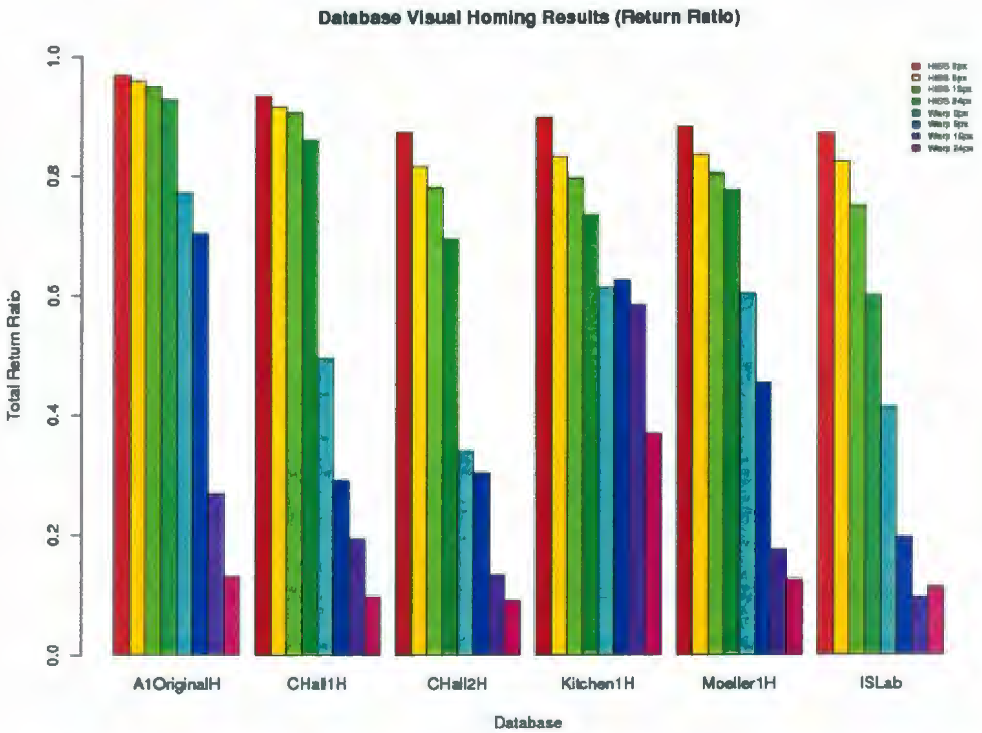



Figure 5.7: Grids showing TAAE results for CHall2H, Kitchen1H, and Möller1H databases. (darker is better)




| Database | 0H | 5H | 15H | 24H | 0W | 5W | 15W | 24W |
|---|---|---|---|---|---|---|---|---|
| A1originalH | 12.4° | 17.8° | 19.3° | 20.9° | 29.7° | 34.2° | 63.9° | 71.7° |
| Chall1H | 14.3° | 15.6° | 16.8° | 18.3° | 33.5° | 47.5° | 58.9° | 68.0° |
| Chall2H | 22.2° | 24.7° | 26.1° | 28.0° | 50.4° | 54.3° | 67.5° | 74.5° |
| Kitchen1H | 22.5° | 28.8° | 31.6° | 36.1° | 46.4° | 46.5° | 49.2° | 57.6° |
| Möller1H | 24.3° | 27.3° | 29.0° | 30.6° | 34.9° | 43.6° | 59.6° | 65.5° |
| RobISLab | 22.7° | 27.3° | 34.7° | 46.6° | 62.9° | 75.4° | 87.8° | 84.7° |

Figure 5.8: Database Results - Angular Error (lower is better)




| Database | 0H | 5H | 15H | 24H | 0W | 5W | 15W | 24W |
|---|---|---|---|---|---|---|---|---|
| A1originalH | 0.969 | 0.959 | 0.949 | 0.928 | 0.772 | 0.703 | 0.269 | 0.130 |
| Chall1H | 0.934 | 0.916 | 0.906 | 0.860 | 0.494 | 0.291 | 0.193 | 0.097 |
| Chall2H | 0.873 | 0.815 | 0.780 | 0.694 | 0.340 | 0.303 | 0.134 | 0.090 |
| Kitchen1H | 0.897 | 0.831 | 0.796 | 0.734 | 0.612 | 0.625 | 0.583 | 0.369 |
| Möller1H | 0.881 | 0.834 | 0.804 | 0.775 | 0.602 | 0.453 | 0.175 | 0.125 |
| RobISLab | 0.870 | 0.823 | 0.749 | 0.599 | 0.412 | 0.195 | 0.095 | 0.112 |

Figure 5.9: Database Results - Return Ratio (higher is better)

The TAAE for homing in scale space is overall lower than that for the warping method, but we must also show that there is indeed a statistically significant difference between the two sets of results. Certain statistical methods only work properly given an input data set which is normally distributed. In order to determine which tests to perform, we must first determine whether or not our data is normally distributed. For the angular error data, we will use the Shapiro-Wilk (W normality) test [32, 31]. For this test, the data we will be using is the set of differences between warping and HiSS for each trial, a sample of which is shown in figure 5.10. This data will be considered to be not normally distributed for output values for $p < 0.1$ [33]. Upon running the W test for each of the data sets individually, as well as all combined data sets as a whole, each test returned a result of $p < 2.2e^{-16}$, indicating that our data is not normally distributed. We can see this non-normal structure for the angular error data in the histograms shown in figure 5.10. For this reason, we will use the sign test in order to draw conclusions about our angular error results. In statistics, the sign test [12] is used to determine whether or not there is a significant difference between two variables $X$ and $Y$. That is, what is the chance that the observed result (in our case, HiSS outperforming warping) could have happened by chance? The sign test answers this question in the form of a P-value. We use the sign test with the alternate hypothesis that $AE(hiss) - AE(warp) < 0$, which represents a HiSS trial which is more accurate than warping. A P-value of $p < 0.05$ is sufficient to support this alternative hypothesis. [12, 18, 20]. The results from this test can be seen in figures 5.11 and 5.12. We can see from these results that each of the tests yields $p < 0.05$, showing that HiSS outperforms the warping method in every test we performed.

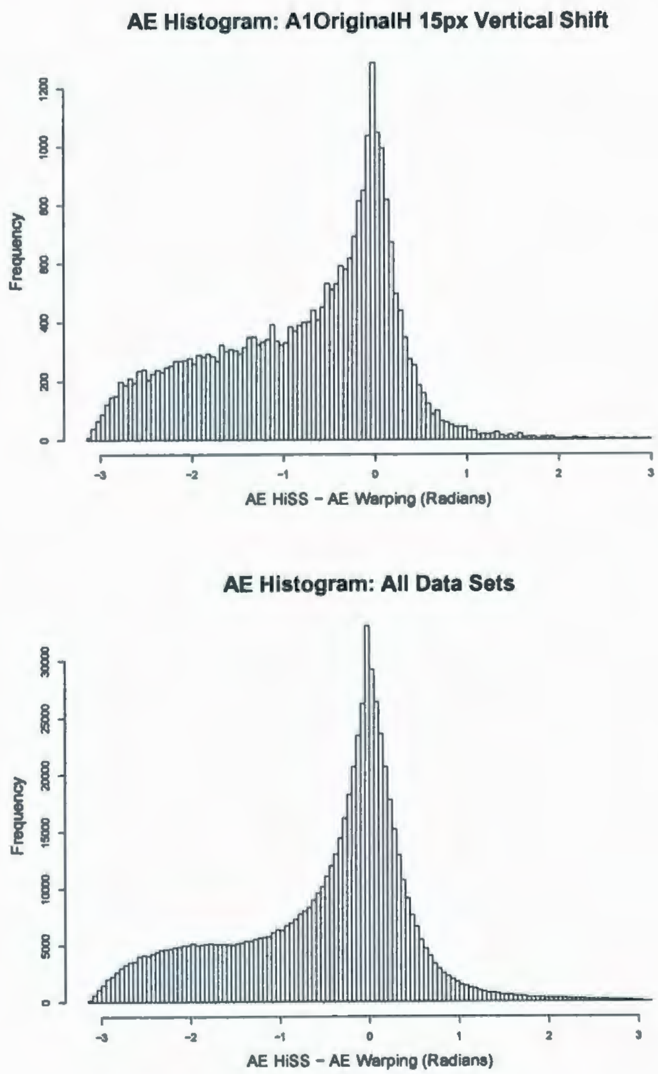



Figure 5.10: Angular error difference histogram. X-Axis value is angular error of homing in scale space minus that of the same trial for the warping method. The histogram shows a skew to the left side of 0, indicating an overall better performance for homing in scale space. Visual inspection along with the Shapiro-Wilk normality test concludes that the data set is not normally distributed.

Sign Test (With Alt. Hyp. HiSS-Warping < 0) - No Pixel Vertical Shift

| Database | Samples | Mean | Median | 95% CI | S-Value | P-Value |
|---|---|---|---|---|---|---|
| A1originalH | 28900 | -0.236 | -0.054 | $(-\pi, -0.051)$ | 11873 | 2.2e-16 |
| Chall1H | 40000 | -0.318 | -0.120 | $(-\pi, -0.116)$ | 14252 | 2.2e-16 |
| Chall2H | 25600 | -0.471 | -0.255 | $(-\pi, -0.246)$ | 8571 | 2.2e-16 |
| Kitchen1H | 11664 | -0.375 | -0.111 | $(-\pi, -0.102)$ | 4497 | 2.2e-16 |
| Möller1H | 58564 | -0.197 | -0.003 | $(-\pi, 0.0)$ | 28851 | 0.0057 |
| RobISLab | 5184 | -0.707 | -0.429 | $(-\pi, -0.399)$ | 1287 | 2.2e-16 |

Sign Test (With Alt. Hyp. HiSS-Warping < 0) - 5 Pixel Vertical Shift

| Database | Samples | Mean | Median | 95% CI | S-Value | P-Value |
|---|---|---|---|---|---|---|
| A1originalH | 28900 | -0.287 | -0.069 | $(-\pi, -0.066)$ | 11580 | 2.2e-16 |
| Chall1H | 40000 | -0.556 | -0.251 | $(-\pi, -0.244)$ | 11481 | 2.2e-16 |
| Chall2H | 25600 | -0.517 | -0.316 | $(-\pi, -0.305)$ | 7991 | 2.2e-16 |
| Kitchen1H | 11664 | -0.309 | -0.084 | $(-\pi, -0.074)$ | 4859 | 2.2e-16 |
| Möller1H | 58564 | -0.285 | -0.052 | $(-\pi, -0.049)$ | 26075 | 2.2e-16 |
| RobISLab | 5184 | -0.841 | -0.659 | $(-\pi, -0.621)$ | 1140 | 2.2e-16 |

Figure 5.11: Tables representing the results from the sign test applied to angular error data for HiSS-Warping for 0 and 5 pixel vertical shifts.

Sign Test (With Alt. Hyp. HiSS-Warping $< 0$) - 15 Pixel Vertical Shift

| Database | Samples | Mean | Median | 95% CI | S-Value | P-Value |
|---|---|---|---|---|---|---|
| A1originalH | 28900 | -0.778 | -0.528 | $(-\pi, -0.513)$ | 6654 | 2.2e-16 |
| Chall1H | 40000 | -0.734 | -0.409 | $(-\pi, -0.399)$ | 9888 | 2.2e-16 |
| Chall2H | 25600 | -0.724 | -0.541 | $(-\pi, -0.525)$ | 6702 | 2.2e-16 |
| Kitchen1H | 11664 | -0.307 | -0.079 | $(-\pi, -0.067)$ | 5016 | 2.2e-16 |
| Möller1H | 58564 | -0.535 | -0.243 | $(-\pi, -0.234)$ | 20712 | 2.2e-16 |
| ISLab | 5184 | -0.927 | -0.915 | $(-\pi, -0.874)$ | 1133 | 2.2e-16 |

Sign Test (With Alt. Hyp. HiSS-Warping $< 0$) - 24 Pixel Vertical Shift

| Database | Samples | Mean | Median | 95% CI | S-Value | P-Value |
|---|---|---|---|---|---|---|
| A1originalH | 28900 | -0.885 | -0.718 | $(-\pi, -0.703)$ | 5903 | 2.2e-16 |
| Chall1H | 40000 | -0.867 | -0.638 | $(-\pi, -0.625)$ | 8217 | 2.2e-16 |
| Chall2H | 25600 | -0.812 | -0.697 | $(-\pi, -0.683)$ | 6057 | 2.2e-16 |
| Kitchen1H | 11664 | -0.375 | -0.133 | $(-\pi, -0.120)$ | 4796 | 2.2e-16 |
| Möller1H | 58564 | -0.609 | -0.386 | $(-\pi, -0.376)$ | 18772 | 2.2e-16 |
| ISLab | 5184 | -0.665 | -0.606 | $(-\pi, -0.573)$ | 1512 | 2.2e-16 |

Figure 5.12: Tables representing the results from the sign test applied to angular error data for HiSS-Warping for 15 and 24 pixel vertical shifts.

## 5.4 Homing Success / Correlations

As well as developing an algorithm for accurate visual homing, we wish to develop a method for predicting homing success. Given two input images for homing, is there any way to use homing in scale space to predict whether or not homing will succeed without actually moving the robot? To answer this, we want to find which types of data correlate strongly with homing success by comparing them with our angular error and return ratio data. To do this, we first look at what kinds of data are produced by our homing method.

The first type of data we will investigate is the percentage of keypoints from the $CV$ image which have found a match in the $SS$ image. Intuitively, the higher percentage of keypoints matched between $CV$ and $SS$, the more similar the images are, and the greater number of keypoints available for the HiSS algorithm to work with. Due to the nature of the algorithm, we would expect this to yield more accurate results than if less keypoints were matched.

The second idea for predicting homing success involves analyzing the centers of expansion and contraction. Ideally we know that translation within an environment causes the centers of expansion and contraction to be separated by 180°. We can then define $\gamma$ as

$$\gamma \;=\; diff(\bar{\theta}_{pos}, \bar{\theta}_{neg}) - \pi \tag{5.7}$$

which under ideal conditions would yield $\gamma = 0$. Values of $\gamma$ which stray from zero would then indicate that our calculated centers do not align opposite to each other, and we would expect inaccurate results to follow.

The last measure we will use is the standard deviation of the horizontal length

| Measure | AE | Dist | RR | $M_{\%}$ | $M_\gamma$ | $M_\sigma$ |
|---|---|---|---|---|---|---|
| AE | 1.000 | 0.381 | -0.232 | -0.388 | 0.162 | 0.359 |
| Dist | 0.381 | 1.000 | -0.248 | **-0.959** | 0.085 | **0.850** |
| RR | -0.232 | -0.248 | 1.000 | 0.242 | -0.144 | -0.207 |
| $M_{\%}$ | -0.388 | **-0.959** | 0.242 | 1.000 | -0.142 | -0.845 |
| $M_\gamma$ | 0.162 | 0.085 | -0.144 | -0.142 | 1.000 | 0.118 |
| $M_\sigma$ | 0.359 | **0.850** | -0.207 | -0.845 | 0.118 | 1.000 |

Figure 5.13: Data correlations for A1OriginalH with 0 pixel vertical shift

of the matched correspondence vectors between $SS$ and $CV$. Intuitively, the longer the horizontal length of these correspondences, the greater the shift the features have undergone in the image, and we would expect less accurate results. Given our set of correspondences $M = \{m_1, m_2, ..., m_n\}$ we calculate the standard deviation $M_\sigma$ of the differences between the $x$ coordinates of the snapshot and current view for each match $m_k \in M$. If we denote the $x$ coordinate of the snapshot of match $m_k$ by $x(\mathbf{ss}_k)$ and the current view by $x(\mathbf{cv}_k)$ then we have:

$$M_\sigma = \sqrt{\frac{1}{N}\left(\sum_{k=1}^{n} |x(\mathbf{ss}_k) - x(\mathbf{cv}_k)|\right)^2} \tag{5.8}$$

Since our data is not normally distributed, we will be using the Spearman method (also known as Spearman's $\rho$) [35] to calculate our correlation coefficients. The Spearman method is a non-parametric measure of rank correlation between data sets. It works in much the same way as the standard Pearson method of determining correlation, with the added step of converting the raw scores into a ranking system based on the total data set. The final output is the correlation between the ranks of the data sets, which has been shown to be more robust under non-normal data, as well as for showing non-linear correlations [17].

The table in figure 5.13 shows sample correlation data from the A1OriginalH database trial under no vertical shift. Each trial performed on the other databases yielded similar results. We can see that there are several high correlation coefficients present in the matrix, which are also high in each of the other database trials. The highest is a negative correlation between $M_{\%}$ and the true distance to the goal, which tells that as we get closer to the goal location, the percentage of keypoints matched gets higher. This observation goes along with our prediction earlier. Another strong correlation is between $M_{\sigma}$ and distance. As we get closer to the goal, the horizontal length of our correspondence vectors shrink.

In order to predict homing success, we would need to see high correlations to angular error or return ratio. Unfortunately, we did not see these high correlations in our data, with values only reaching 0.4 as opposed to the 0.95 we see in our distance correlation. However, due to these extremely high correlations with true measured distance to the goal, we will use the highest of these, $M_{\%}$, as a way to attempt to predict the distance to the goal in our live trials.

## 5.5 Distance Estimation

In order to properly return an agent to a goal, it is not only necessary that we have an accurate homing angle, but we must also have an accurate distance estimation in order to stop the agent at the goal location. If there is no notion of the distance between **cv** and **ss**, the robot will simply oscillate around the goal location with no means to come to a complete stop. As shown in the previous section, there is a very high correlation between the true distance to the goal and the percentage of keypoints matched $M_{\%}$. In figure 5.14 we can see that this high level of correlation is due to a seemingly exponential relationship between $M_{\%}$ and the true distance $d$. Using this observation, we will attempt to find a function fitting the form $d = ae^{bM_{\%}}$ using nonlinear regression. We performed nonlinear regression using the R stats package nls() nonlinear least squares function. Each of the images in figure 5.14 shows the results of using nonlinear regression on each database in order to find a distance estimation function. Each graph shows 4 functions (one for each of the 0, 5, 15, 24 pixel vertical shifts) overlaid on a plot of $M_{\%}$ vs. distance $d$. Note that due to the similarity of the resulting values of $a$ and $b$ the lines are difficult to distinguish. The table in figure 5.15 shows the computed values of $a$ and $b$ for the functions $d = ae^{bM_{\%}}$, as well as the standard errors for $a$ and $b$, and the residual standard error for the method.

We can see by figures 5.14, 5.15 and 5.16 that the distance estimation function fits nicely to the exponential curve. The function also remains remarkably similar despite large vertical shifting within the image (represented by the different lines), making this method for distance estimation feasible for environments without level movement surfaces. One downside to this approach however is that as the true distance from the goal increases, so does the uncertainty in the use of $M_{\%}$ as a prediction for $d$. At areas in the graph where the slope of the computed function has a larger magnitude, similar values of $M_{\%}$ can yield dramatically different distances. This would lead us to believe that this distance estimation method will be less accurate for long range

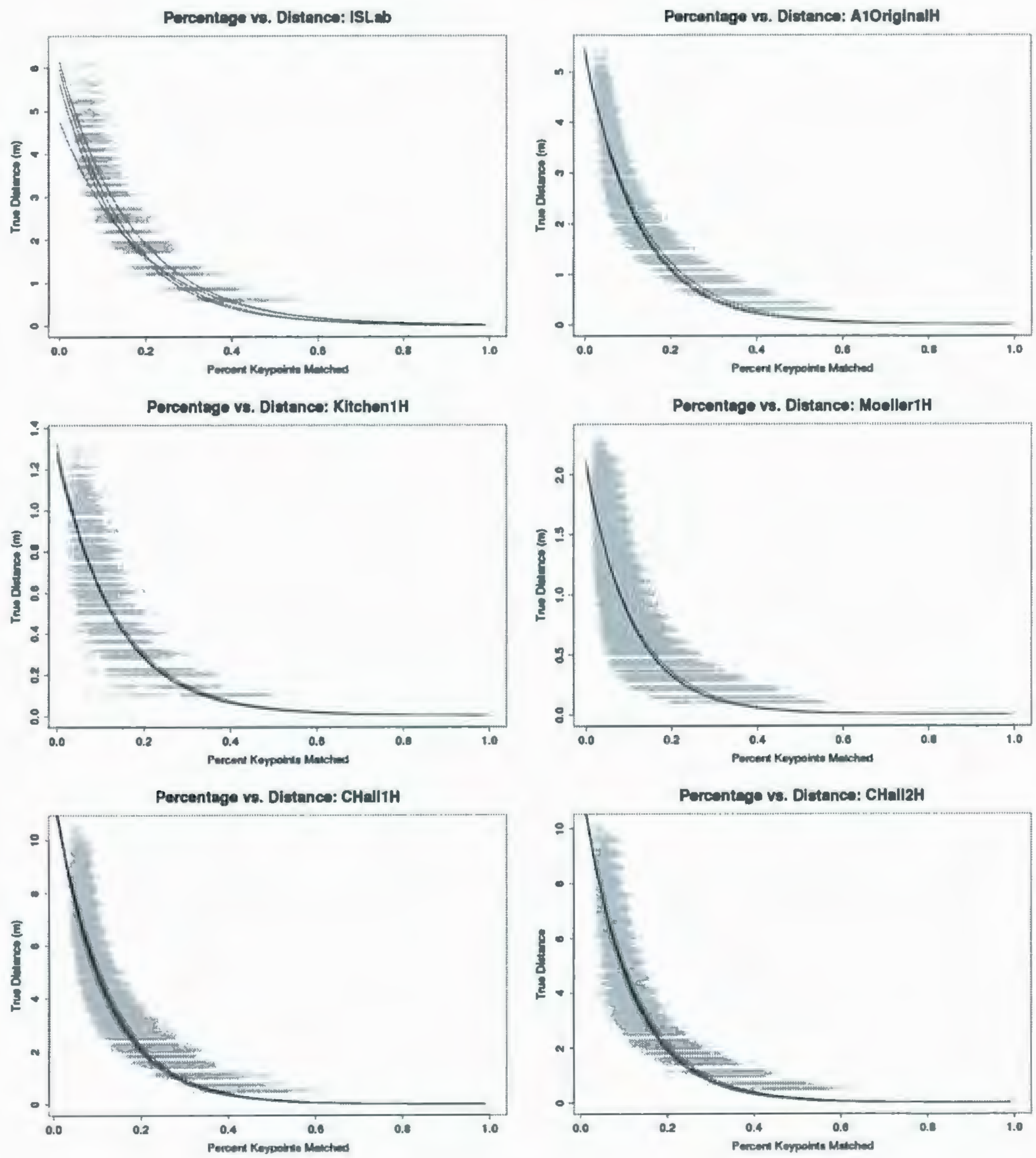



Figure 5.14: Percentage Matched vs. Distance graphs for each database

| Database | Trial | $a$ | $b$ | $a$ std. err | $b$ std. err | RSE |
|---|---|---|---|---|---|---|
| Islab | 0px Vert | 10.06780 | -5.85665 | 0.05436 | 0.04763 | 0.8335 |
| | 5px Vert | 9.78177 | -6.43592 | 0.06124 | 0.06236 | 0.9595 |
| | 15px Vert | 9.26169 | -6.53532 | 0.06932 | 0.07979 | 1.144 |
| | 24px Vert | 7.8007 | -5.4061 | 0.0731 | 0.1028 | 1.494 |
| A1OriginalH | 0px Vert | 17.68985 | -7.27702 | 0.03602 | 0.02122 | 1.24 |
| | 5px Vert | 17.89868 | -7.73030 | 0.03793 | 0.02307 | 1.269 |
| | 15px Vert | 18.23209 | -8.03045 | 0.04010 | 0.02428 | 1.286 |
| | 24px Vert | 18.11404 | -8.17131 | 0.04154 | 0.02585 | 1.344 |
| CHall1H | 0px Vert | 23.85965 | -8.45522 | 0.05915 | 0.02441 | 1.668 |
| | 5px Vert | 23.85637 | -8.75030 | 0.06172 | 0.02625 | 1.734 |
| | 15px Vert | 23.82518 | -8.82529 | 0.06291 | 0.02698 | 1.772 |
| | 24px Vert | 23.32249 | -8.90887 | 0.06561 | 0.02945 | 1.907 |

Figure 5.15: Table of results for functions plotted in figure 5.14. $a$ and $b$ correspond to the values output by performing non linear regression on function $d = ae^{bM_{\%}}$. Standard errors for $a$ and $b$, as well as the residual standard error (RSE) are also included.

| Database | Trial | $a$ | $b$ | $a$ std. err | $b$ std. err | RSE |
|---|---|---|---|---|---|---|
| CHall2H | 0px Vert | 23.37360 | -8.50571 | 0.08667 | 0.03625 | 1.88 |
| | 5px Vert | 23.71199 | -8.87797 | 0.09259 | 0.03905 | 1.941 |
| | 15px Vert | 24.10059 | -9.11256 | 0.10206 | 0.04246 | 2.037 |
| | 24px Vert | 24.07061 | -9.26291 | 0.10868 | 0.04574 | 2.144 |
| Kitchen1H | 0px Vert | 12.71268 | -7.15042 | 0.07095 | 0.05730 | 1.447 |
| | 5px Vert | 12.91063 | -7.58406 | 0.07962 | 0.06509 | 1.53 |
| | 15px Vert | 13.26301 | -7.77610 | 0.08449 | 0.06660 | 1.538 |
| | 24px Vert | 12.50825 | -7.31220 | 0.09264 | 0.07631 | 1.76 |
| Möller1H | 0px Vert | 20.28980 | -8.72208 | 0.05359 | 0.03502 | 2.685 |
| | 5px Vert | 20.66887 | -9.24524 | 0.05959 | 0.03914 | 2.787 |
| | 15px Vert | 21.09868 | -9.46355 | 0.06302 | 0.04028 | 2.808 |
| | 24px Vert | 21.01630 | -9.41256 | 0.06539 | 0.04160 | 2.886 |

Figure 5.16: Table of results for functions plotted in figure 5.14. $a$ and $b$ correspond to the values output by performing non linear regression on function $d = ae^{bM_{\%}}$. Standard errors for $a$ and $b$, as well as the residual standard error (RSE) are also included.

homing, but become more accurate as we approach the goal.

Another issue of note is the fact that this function varies with image dimensions. Homing within an environment using images with a height of 50 pixels will yield a different distance estimation function than an image with a height of 100 pixels. Experimentally, we have found that as resolution increases, more keypoints are found, and a higher value for $M_{\%}$ results.

## 5.6 ISLab Trials

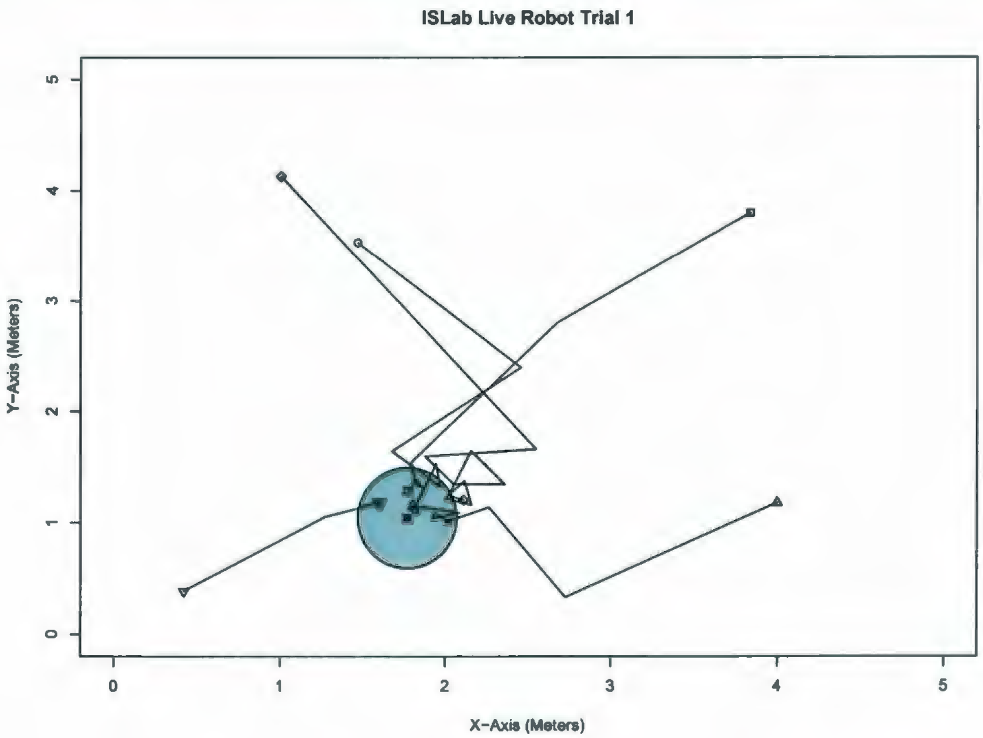
To test this algorithm online on our Pioneer robot, we used the same environment as used in the ISLab database. Five different goal locations were chosen, with 5 starting locations for each goal location spaced evenly throughout the environment. The robot takes an image at its current location, compares it to the goal image, computes the estimated distance and homing angle, and moves that amount in that direction. This process repeats until the robot believes it is within 30cm of the goal (success) or for a maximum of 12 iterations (failure). The distance estimation function used for the homing in scale space method was found by using the ISLab image database as a training set. A real-time distance estimation function calculator is discussed in the future work section.

It was our original intention to compare homing in scale space to the warping method by using both visual homing methods to conduct online trials. Upon attempting visual homing in the ISLab using the warping method, it was found to be too inaccurate to carry out the trials. Of several dozen initial tests, the robot would almost inevitably veer off of the allotted limits for navigation. We suspect this is due

Figure 5.17: Images taken from the ISLab database. Top image shows location (6,3) while the bottom image shows location (6,4). The movement in these images should be only translation to the right from the top image to the bottom image. Due to inconsistencies in the elevation at which the images were taken, we can see objects (such as the garbage can in the center) have shifted vertically as well. This vertical shifting resulted in extremely poor performance for the warping method during live trials.

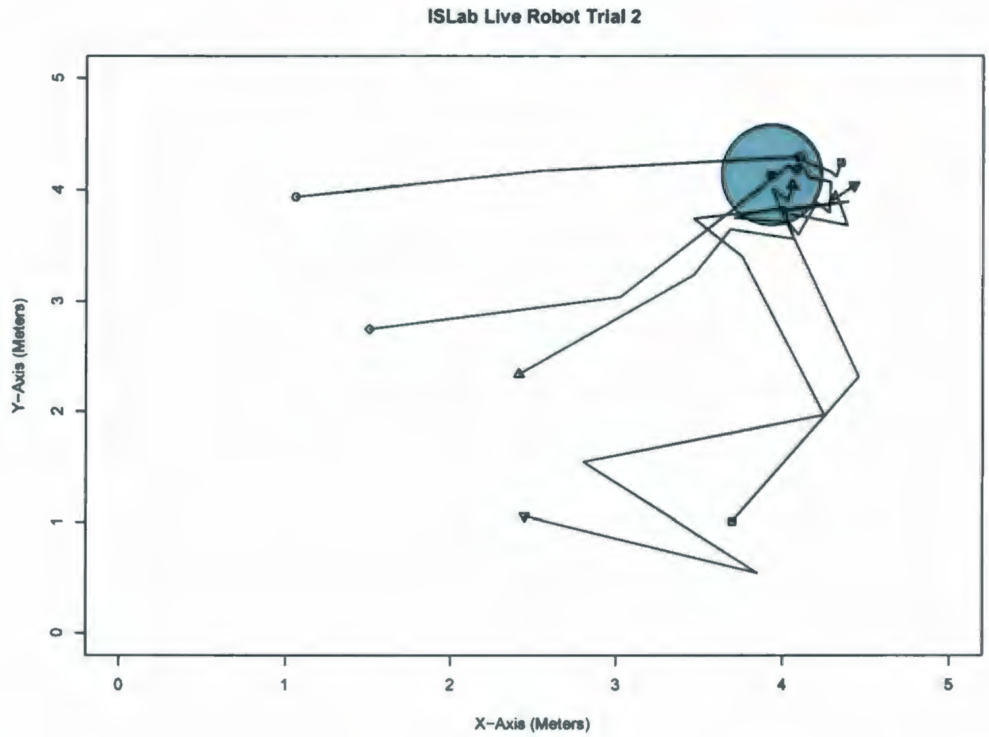
to the nature of the images captured by the robot. Due to the robot's wheels being imperfectly shaped, the height of the camera affixed to the top of the robot varies between 1-2cm throughout the course of a full revolution of the robot's wheels. Since the warping method relies heavily on the stability of the horizon within an image, we believe that this variance caused enough shift of the image horizon to cause the warping method to perform poorly. Due to this, results for live trials using the warping method were not included. An illustration of this can be seen in figure 5.17.

We will define two types of success with respect to visual homing for our live robot trials. Type A success means the robot came to stop within both an estimated distance of 30cm and an actual distance of 30cm. Type B success means that at some point the robot came within a true distance of 30cm of the goal, however it did not stop due to error in its distance estimation. If the robot passed within 30cm of the goal at any intermediary step during a trial, but estimated it was not within the threshold, we will record it as having been an undetected arrival (UA). Therefore, type B success is equivalent to any trial which recorded an undetected arrival without achieving type A success. Figures 5.18 through 5.22 show results for each of the robot trial paths, along with a table of the associated estimated distance, actual distance, and distance estimate error for the final stage of the homing trial.



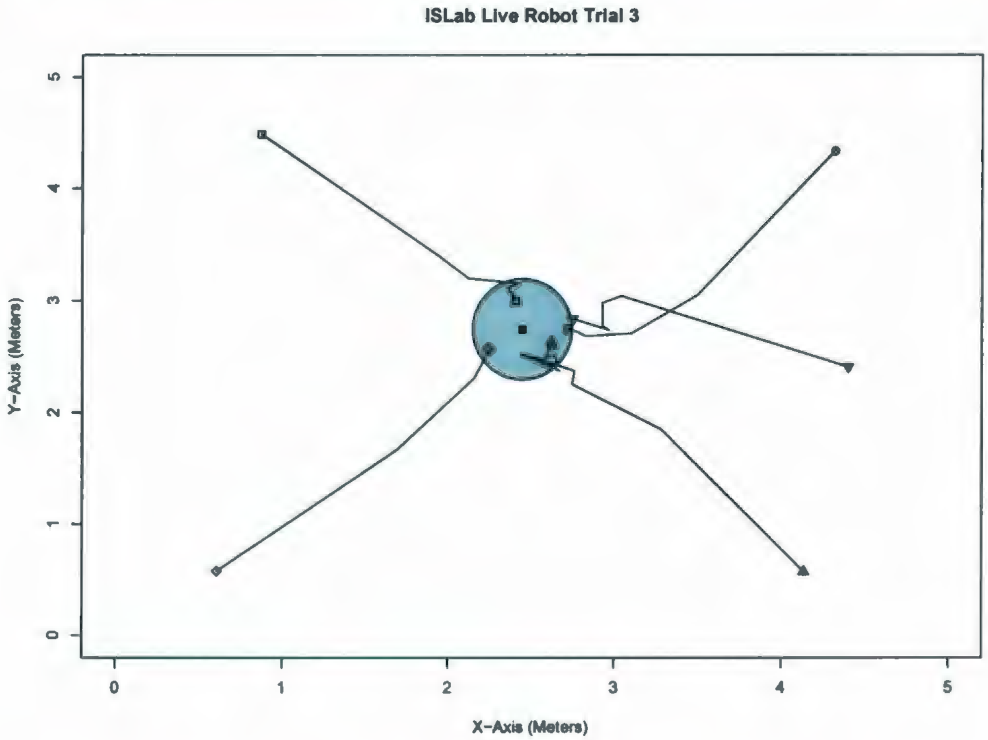

| Trial | | Est Dist | Act Dist | Error | Success | UA | Steps |
|---|---|---|---|---|---|---|---|
| 1 | ○ | 0.25 | 0.09 | 0.16 | A | NO | 5 |
| | □ | 0.3 | 0.24 | 0.06 | A | YES | 6 |
| | ◇ | 0.43 | 0.38 | 0.05 | B | YES | 12 |
| | △ | 0.26 | 0.12 | 0.14 | A | YES | 7 |
| | ▽ | 0.28 | 0.22 | 0.06 | A | NO | 2 |

Figure 5.18: ISLab Live Homing Trial 1



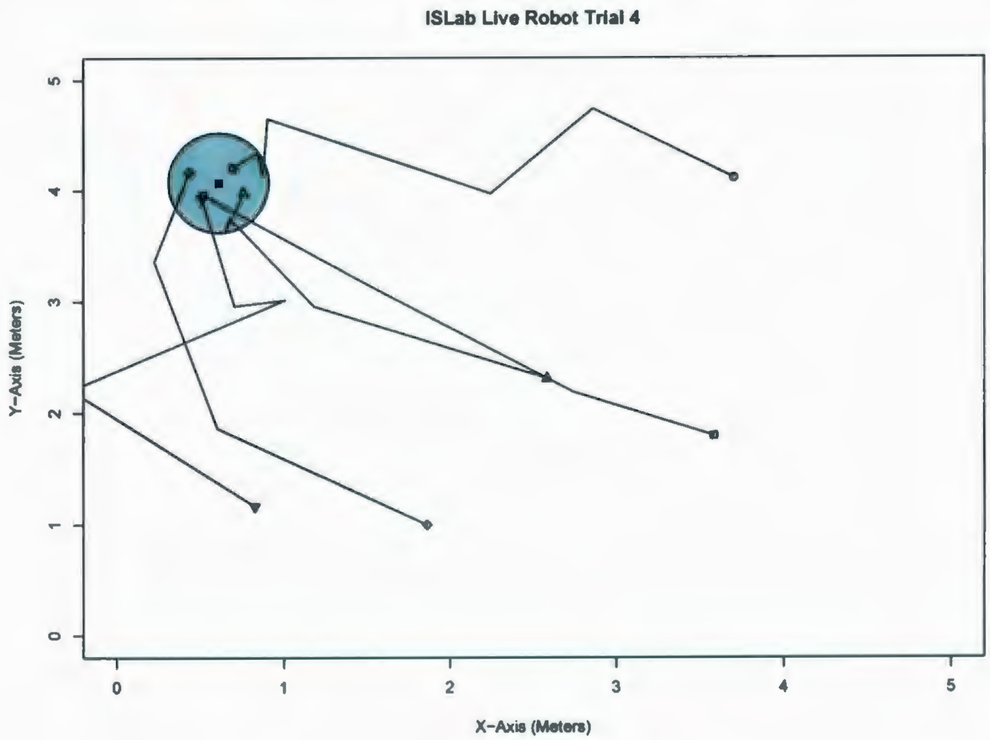

| Trial | | Est Dist | Act Dist | Error | Success | UA | Steps |
|---|---|---|---|---|---|---|---|
| 2 | ○ | 0.29 | 0.24 | 0.06 | A | NO | 2 |
| | □ | 0.57 | 0.43 | 0.15 | A | YES | 12 |
| | ◇ | 0.23 | 0.16 | 0.07 | A | YES | 3 |
| | △ | 0.29 | 0.16 | 0.13 | A | YES | 9 |
| | ▽ | 0.62 | 0.50 | 0.12 | NO | NO | 12 |

Figure 5.19: ISLab Live Homing Trial 2



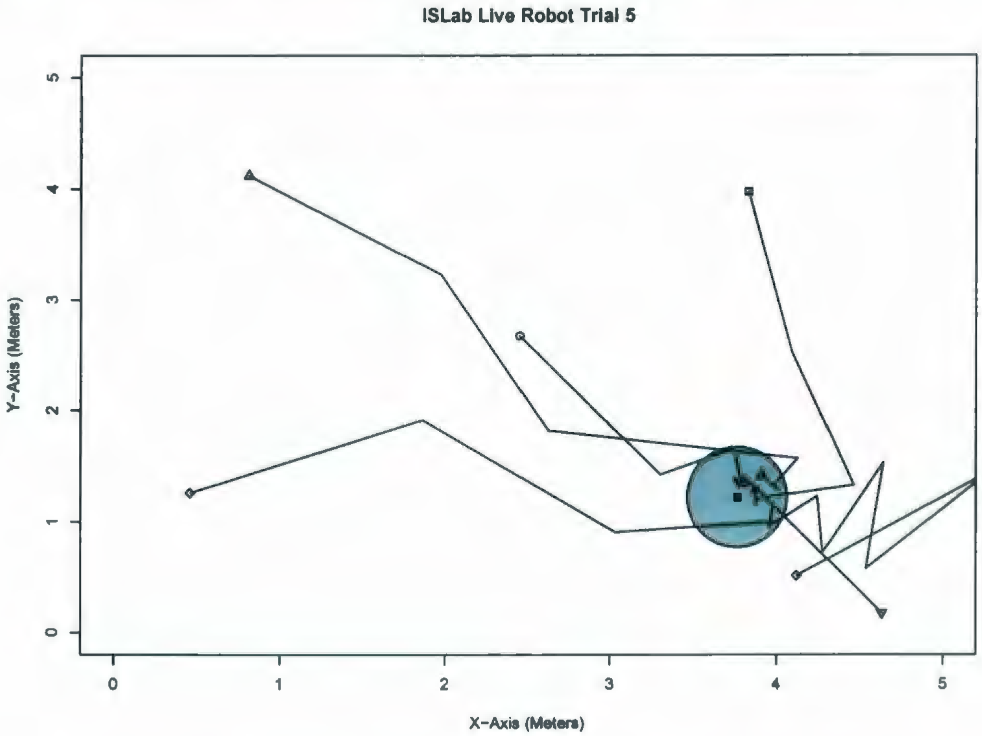

| Trial | | Est Dist | Act Dist | Error | Success | UA | Steps |
|---|---|---|---|---|---|---|---|
| 3 | ○ | 0.29 | 0.27 | 0.02 | A | NO | 5 |
| | □ | 0.28 | 0.26 | 0.02 | A | NO | 6 |
| | ◇ | 0.27 | 0.27 | 0.00 | A | NO | 4 |
| | △ | 0.29 | 0.22 | 0.07 | A | YES | 10 |
| | ▽ | 0.29 | 0.32 | 0.03 | A | NO | 5 |

Figure 5.20: ISLab Live Homing Trial 3




| Trial | | Est Dist | Act Dist | Error | Success | UA | Steps |
|---|---|---|---|---|---|---|---|
| 4 | ○ | 0.27 | 0.16 | 0.12 | A | YES | 6 |
| | □ | 0.27 | 0.15 | 0.12 | A | NO | 3 |
| | ◇ | 0.24 | 0.20 | 0.05 | A | NO | 3 |
| | △ | 0.25 | 0.18 | 0.07 | A | NO | 4 |
| | ▽ | 0.27 | 0.18 | 0.09 | A | NO | 4 |

Figure 5.21: ISLab Live Homing Trial 4




| Trial | | Est Dist | Act Dist | Error | Success | UA | Steps |
|---|---|---|---|---|---|---|---|
| 5 | ○ | 0.28 | 0.14 | 0.14 | A | YES | 4 |
| | □ | 0.26 | 0.13 | 0.13 | A | YES | 5 |
| | ◇ | N/A | 0.79 | N/A | B | YES | 12 |
| | △ | 0.25 | 0.25 | 0.00 | A | YES | 6 |
| | ▽ | 0.29 | 0.14 | 0.15 | A | YES | 2 |

Figure 5.22: ISLab Live Homing Trial 5

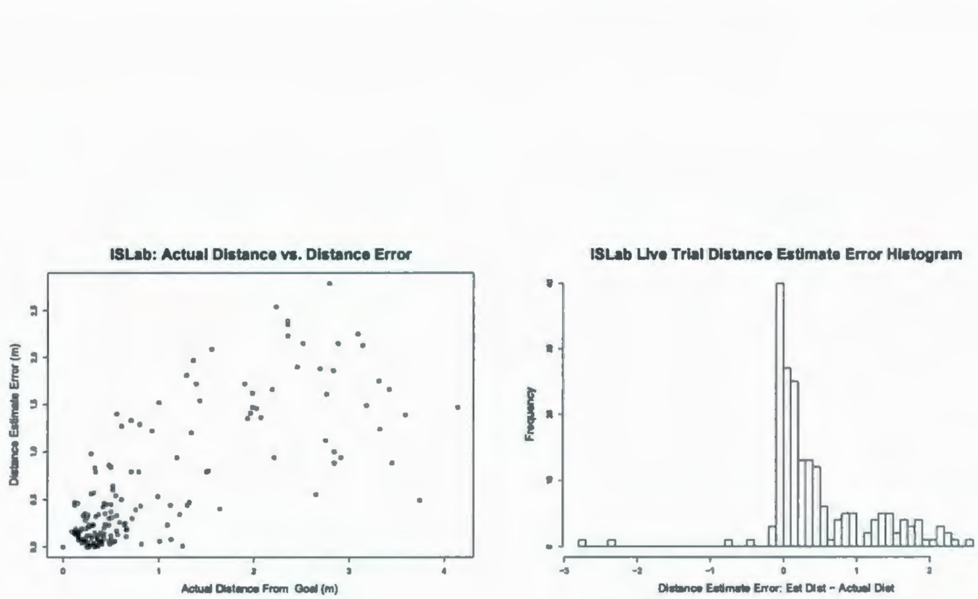



Figure 5.23: Graph (left) of actual distance from goal $d_a$ vs. distance error $d_{err} = |d_e - d_a|$, along with the associated distance estimate error histogram (right).

For the 25 homing trials conducted, 21 of them resulted in type A success, 3 resulted in type B success, and only one resulted in failure. 13 of the trials resulted in the recording of an undetected arrival, which means that the method is actually getting closer to the goal than its distance estimation function would lead us to believe.

We can see by the graph in figure 5.23 that as the actual distance to goal ($d_a$) increases, so does the distance error ($d_{err}$). Using the Spearman method of correlation between these two values yields a correlation coefficient of 0.784, which strongly reinforces this relationship. The second graph is a histogram of $d_e - d_a$, showing a possible reason for the high number of UAs in the live trials. The distance estimation function nearly always returns a value which is higher than that of the actual distance to the goal, resulting in the robot thinking it is further away from where it is, with a mean of 0.462m and a median of 0.195cm. A possible reason for this is the fact that the distance estimation function was computed from the ISLab database, in which images were spaced 61cm apart. Since distances during the actual homing trials may be much lower than this, using only these larger distances as the basis for estimating the parameters of the exponential function may be the cause of the error.

## 5.7 Discussion

In our efforts to show that homing in scale space is an accurate, robust visual homing method, we performed several tests in which the results conclusively showed superior performance to the warping method. By choosing SIFT as the underlying feature detection method, we were able to produce accurate feature matches, even under 3D orientation changes. Using these SIFT feature matches, we were able to determine the foci of expansion and contraction in a given pairing of panoramic snapshot and current view images. The resulting homing angle acquired by using the weighted means method proved to be an accurate measure for visual homing, producing both low angular error and high return ratios. By randomly shifting our images horizontally and vertically we have shown that our method achieves complete rotation invariance for visual homing, as well as robustness to simulated vertical shifting.

We chose the one-dimensional warping method as a comparison since it was the leading visual homing method which does not require an image pre-alignment step. In each database trial we conducted HiSS was shown to produce lower average angular error as well as a higher return ratio than the warping method, especially for the trials in which we simulated vertical shifting. The statistical methods we used confirmed that these results were indeed statistically significant. Our live robot trials in the Intelligent Systems laboratory have showed us that not only does the method work well for image databases, but the method is capable of achieving high rates of homing success on a live robot, with type A success rate of 84%. If we combine this with our type B successes, we see that homing in scale space was able to bring the robot to within 30cm of the goal in 24 of the 25 of the trials. All of this was achieved in an environment under which the warping method failed to produce results accurate enough to be meaningfully recorded.

# Chapter 6

# Conclusion

In creating the homing in scale space method, we wanted to determine whether scale change information from a feature detector such as SIFT (but not necessarily SIFT) is useful for computing the home direction. We found not only is this scale change data useful, but produces more accurate results than existing methods. Another related question is how to achieve invariance to changes in orientation and vertical shifting. Due to the nature of SIFT keypoints, which allow features to be matched despite changes in 3D orientation, we have been able to overcome these constraints which have hindered other homing methods.

We also wanted to determine whether or not we could implement our method on a real robot, where efficiency and computing time are of great importance. By utilizing low-resolution images and by implementing HiSS efficiently in C, we were able compute a homing vector to under half a second on a 700MHz processor. This implementation has shown that a real-time robotic visual homing system based on HiSS can be constructed, and is able to achieve the accuracy and speed necessary to be considered useful for real time autonomous robot navigation.

We therefore conclude that due to its accuracy, speed, and robustness, homing in scale space should be considered as a viable method for applications of visual homing.

## 6.1 Future Work

Several ideas relating to homing in scale space were conceived throughout the course of this research, but due to time constraints were not able to be fully explored.

The first of these ideas was the scale difference threshold. Recall the value of $\beta = \sigma_{ss} - \sigma_{cv}$ which determined whether or not a keypoint was classified as belonging to the region of contraction or expansion. In cases of images which were taken at locations which were very close together, we would see many very small values for $\beta$. If we consider factors such as noise or improper camera focus, we see that these keypoints may be misclassified. The scale difference threshold would be some value $T_\beta$ for which keypoints with values of $|\beta| < T_\beta$ are discarded. Initial exploration into this idea yielded better results for some databases, while yielding worse results for others. A possible reason for this could be due to the spacing between images in a given database. Possible future work may include scaling $T_\beta$ based on the estimated distance between images to produce overall better homing results.

Distance estimation was another area where improvements could be made. The distance estimation formula which was used in our live robot trials was computing using nonlinear regression based on data from data collected from the ISLab database. We propose that a distance estimation function for a particular environment could be calculated by odometry data from a live robot, eliminating the need for an existing database. To do this, the robot would first need to acquire an image of a goal location within the environment. As the robot then moved around to various locations, true distance from odometry could be compared to the value of $M_{\%}$ used to calculate the distance estimation function. If some incremental version of nonlinear regression was then used, the estimation function could not only be calculated live, but could grow increasingly accurate as the robot continued to navigate. An additional strategy such as Kalman filtering could be used to then track our confidence in the distance estimate from odometry. This continuous movement within an environment would also cover a much wider range of distances than the discretely spaced database model, possibly yielding more accurate results.

A new database of images is needed for testing visual homing under arbitrary 3D transformations. Such a database would need to be captured not only along a grid of different locations, but at varying elevations and 3D orientations as well. Our attempts at simulating vertical shifting in images is clearly not ideal due to the missing region of the image, but also due to the lack of change in 3D perspective which would be present in a real elevation shift. If this database was captured in outdoor as well as indoor environments, it would allow for a much better overall comparison of visual homing algorithms.

The ultimate end goal of homing in scale space would be to apply the method to unmanned aerial vehicles (UAVs). The lack of constraints on the presence of an image horizon should allow this method to be applied to situations where more arbitrary transformations are performed. Since aerial vehicles can navigate in six degrees of freedom, our new homing vector would be described using the angle of rotation $\theta_{homing}$ as well as angle of elevation $\varphi_{homing}$. $\theta_{homing}$ would still be calculated by the angular mean of $f_{\theta x}$, while $\varphi_{homing}$ would be calculated by the angular mean of $f_{\theta y}$ for a given set of keypoints. With these angles, along with an estimated distance from the goal, visual homing in UAVs would be possible.

## 6.2 Summary

In summary, we have presented a novel method for local visual homing using SIFT scale space information which is robust to changes in relative 3D orientation. The major contribution of this method is its ability to perform accurate visual homing under fewer constraints than previous methods. We showed that the method was not only invariant to robot orientation, but also showed far less accuracy loss than the warping method under simulated changes in elevation.

# Bibliography


[1] The R project for statistical computing, 2008.

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

[3] A. Briggs, Y. Li, D. Scharstein, and M. Wilder. Robot navigation using 1d panoramic images. In *International Conference on Robotics and Automation*, pages 2679–2685. 2006.

[4] B. Cartwright and T. Collett. Landmark learning in bees. *Journal of Comparative Physiology A*, 151:521–543, 1983.

[5] B. Cartwright and T. Collett. Landmark learning in bees. *Biological Cyber-netics*, 57:85–93, 1987.

[6] T. Collett and M. Collett. Memory use in insect visual navigation. *Nature Reviews Neuroscience*, 3:542–552, 2002.

[7] J. Devore. *Probability and Statistics for Engineering and the Sciences.* Brooks/Cole, 1991.

[8] S. N. Ferdaus, A. Vardy, G. Mann, and R. Gosine. Comparing global measures of image similarity for use in topological localization of mobile robots. 2008.

[9] M. Franz, B. Schölkopf, P. Georg, H. Mallot, and H. Bülthoff. Learning view graphs for robot navigation. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 138–147. ACM Press, 1997.

[10] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5:111–125, 1998.

[11] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79:191–202, 1998.

[12] J. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference.* Marcel Dekker Inc., New York, 1992.

[13] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool. Omnidirectional vision based topological navigation. *International Journal of Computer Vision*, 74(3):219–236, 2007.

[14] J. Hong, X. Tan, B. Pinette, R. Weiss, and E. Riseman. Image-based homing. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacremento, CA*, pages 620–625, 1991.

[15] W. Hübner and H. Mallot. Metric embedding of view-graphs: A vision and odometry-based approach to cognitive mapping. *Autonomous Robots*, 23:183196, 2007.

[16] S. V. K. Weber and M. Srinivassan. Insect-inspired robotic homing. *Adaptive Behaviour*, pages 65–97, 1999.

[17] M. Kendall. *Rank correlation methods.* Griffin, 1962.

[18] L. Kitchens. *Basic Statistics and Data Analysis.* Duxbury, 2003.

[19] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems, Special Issue: Biomimetic Robots*, 30:39–64, 2000.

[20] E. L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks.* Holden and Day, San Francisco, 1975.

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

[22] MobileRobots. Advanced robotics interface for applications (aria) robotic sensing and control libraries, 2008.

[23] MobileRobots. The high-performance all-terrain robot (p3-at), 2008.

[24] MobileRobots. Mobilesim, the mobilerobots simulator, 2008.

[25] MobileRobots. The pioneer p3-dx robot, 2008.

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

[27] R. Möller and A. Vardy. Local visual homing by matched-filter descent in image distances. *Biological Cybernetics*, 95:413–430, 2006.

[28] R. Möller, A. Vardy, S. Kreft, and S. Ruwisch. Visual homing in environments with anisotropic landmark distribution. *Autonomous Robots*, 23:231–245, 2007.

[29] J. S. Pons, W. Hübner, J. Dahmen, and H. Mallot. Vision-based robot homing in dynamic environments. In K. Schilling, editor, *13th IASTED International Conference on Robotics and Applications*, pages 293–298. 2007.

[30] A. Rizzi, D. Duina, S. Inelli, and R. Cassinis. Unsupervised matching of visual landmarks for robotic homing using Fourier-Mellin transform. In *Intelligent Autonomous Systems 6*, pages 455–462. 2000.

[31] P. Royston. Algorithm AS 181: The W test for normality. In *Applied Statistics*, pages 176–180. 1982.

[32] P. Royston. An extension of shapiro and wilk's w test for normality to large samples. In *Applied Statistics*, pages 115–124. 1982.

[33] P. Royston. A remark on algorithm AS 181: The W test for normality. In *Applied Statistics*, pages 547–551. 1995.

[34] D. L. S. Se and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *Int. J. Robot. Res.*, 21:39–64, 2001.

[35] C. Spearman. The proof and measurement of association between two things. *Amer. J. Psychol.*, 15:72–101, 1904.

[36] A. Vardy. Long-range visual homing. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics.* IEEE Xplore, 2006.

[37] A. Vardy. A simple visual compass with learned pixel weights. In *Proc. of Canadian Conference on Electrical and Computer Engineering.* IEEE Xplore, 2008.

[38] A. Vardy and R. Möller. Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1/2):47–90, 2005.

[39] A. Vardy and R. Möller. Panoramic image database, 2008.

[40] A. Vardy and F. Opacher. Low-level visual homing. *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life*, pages 875–884, 2003.

[41] J. Zeil, M. Hofmann, and J. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.