Long-Range Visual Homing

Andrew Vardy
Department of Computer Science, Faculty of Engineering & Applied Science
Memorial University of Newfoundland
St. John's, Canada
`http://www.cs.mun.ca/~av`

***Abstract*— A biologically-inspired approach to robot route following is presented. Snapshot images of a robot's environment are captured while learning a route. Later, when retracing the route, the robot uses visual homing to move between positions where snapshot images had been captured. This general approach was inspired by experiments on route following in wood ants. The impact of odometric error and another key parameter is studied in relation to the number of snapshots captured by the learning algorithm. Tests in a photo-realistic simulated environment reveal that route following can succeed even on relatively sparse paths. A major change in illumination reduces, but does eliminate, the robot's ability to retrace a route.**

***Index Terms*— visual homing, route following, robot navigation, insect navigation.**

I. Introduction

It has been postulated that insects memorize *snapshot images* at key locations in their environments [1], [2], [3], [4]. These snapshot images may later allow a goal to be pinpointed [3], [4], or a route to be retraced [2]. It has been shown that when wood ants retrace a route, they move so as to minimize the difference between the current image and a snapshot image [2]. This paper investigates the application of this same concept to allow a robot to learn and follow a route through an environment. Experiments in a photo-realistic simulation demonstrate the feasibility of this idea for robot navigation.

In general, the ability to follow a route can be achieved by *recognition-triggered responses* [5], which associate local navigation methods with particular sensory stimuli. A robot following a route recognizes its current location as being associated with a particular local navigation method such as corridor following [6], wall following, or visual homing. The local navigation method is then followed until the goal is attained. This goal may itself be intermediate, in which case it serves as the start point for the next leg of the route. This style of navigation was first proposed for robots by Kuipers and Byun [7].

We describe route learning and following methods which use visual homing for local navigation. Visual homing is the ability to return to a goal position by comparing the image currently viewed with a snapshot image taken from the goal. Cartwright & Collett showed that insects such as honeybees have this ability [4]. Their experiments showed that honeybees approaching a food source move so as to visually re-position landmarks to match their remembered positions as seen from the goal. The bearing and apparent size of landmarks were found to be more significant cues than the distance of those landmarks. This is telling, as distance information could have been inferred by a bee while approaching the goal—perhaps then integrated into a volumetric world model. Instead, bees use the simpler strategy of moving so as to minimize the difference between images. The algorithm Cartwright & Collett proposed was based on pairing image features (regions), and moving so as to correct for differences in the bearing and size of those features. Cartwright & Collett's algorithm, as well as several interesting variants were tested for robot visual homing by Lambrinos et al [8]. A number of different methods for visual homing have been proposed (see reviews in [9], [10], [5]), many of which were directly inspired by Cartwright & Collett.

Most of the published approaches to visual homing have focused on moving a robot to a single goal position. However, the approach described here employs visual homing to retrace a route composed of intermediate goal positions. A small number of other researchers have investigated this problem. Hong et al. tested one of the first visual homing algorithms by applying it to travel along a route where the way-points of the route were uniformly spaced [11]. This work did not address the problem of learning more efficient sparse routes, which is addressed here. Franz et al. used a visual homing algorithm [12] to learn and travel between nodes in a graph representation of the environment [13]. Nehmzow & Owen conducted a series of experiments on route learning using a self-organized network of place identifiers [14]. While their robot did not employ visual homing, it used a sonar-based homing approach to achieve the same purpose. More recently, Argyros et al. proposed a route following method based on feature-based visual homing [15]. This work differs from the method presented here in that the features Argyros et al. employ are tracked between frames. Tracking may be lost if the homing algorithm receives new data with insufficient frequency.

The method used here is as follows. In learning a route, the direction back to the last captured *snapshot image* is computed by visual homing and compared to the direction computed by odometry. If there is a sufficient angular difference between the two, a new snapshot image is captured. The impact of the angular difference threshold and the degree of odometric error on the number of snapshots captured is investigated here. The route following method uses visual homing to home, in sequence, to the positions where snapshot images had been captured.

In the next section we describe the visual homing method used to travel between snapshot positions. Following this, the learning and route following algorithms are presented. The next section describes the simulated test environment. The results section follows which details the findings on both route learning and following. We conclude with a short discussion and closing remarks.

II. Visual Homing

The visual homing method employed here is Möller and Vardy's Matched-Filter Descent in Image Distances (MF-DID) [16]. This method builds upon the work of Zeil et al. who found that near the snapshot position the image distance function varies smoothly and monotonically with spatial distance [10]. Zeil et al. tested gradient descent on image distances as a means of homing. However, in order to estimate the local gradient of the image distance function, it was necessary for the robot to make exploratory movements to sample this function at different points in space. These exploratory movements are costly in terms of both time and energy. Möller and Vardy found that these movements could be eliminated by warping the current image *as if* the robot had made an exploratory movement. An approximation to the image distance function can then be sampled. Möller and Vardy found this new method superior in performance to the original gradient descent method on a database of images[1].

III. Learning

For the robot to learn the required route, it must first be manually driven along it. The learning algorithm records snapshot images and associated odometric information from particular points along the route. The simplest method would be to record snapshots at the maximum capture rate. However, this strategy is undesirable because of the large amount of memory required. The odometric information stored by the algorithm is an approximate vector back to the position of the last snapshot—known as an *odometry motion vector*. If the robot's ability to dead reckon were extremely accurate, these odometry motion vectors would be sufficient to allow the robot to retrace its route. However, any odometric sensor incurs cumulative error which renders odometry insufficient as the sole cue for navigation.

The learning algorithm proceeds as follows. We assume that the robot has already captured a snapshot image and that its odometric estimate of the snapshot position is (x_{ss}, y_{ss}). At each step along the route, the robot uses visual homing to compute the angle β_{vh} to the last snapshot position. The odometric position gives a second estimate of the angle to the last snapshot,

$$\beta_o = atan2(y'_o - y_{ss}, x'_o - x_{ss}) + \pi$$

[1]However, the fixed step size imposed by the image database may have been detrimental to the performance of the original method.

where (x'_o, y'_o) is the current position from odometry. The difference between these two angles $\Delta\beta$ is computed and mapped to the range $[0, \pi)$. If $\Delta\beta$ exceeds a threshold angle ϕ, a new snapshot is captured.

When a snapshot is captured the *previous* image is stored as the snapshot image (it is added to a growing list of snapshot images). The previous image is stored because at the current position $\Delta\beta > \phi$. This condition is a heuristic, which indicates that homing back to the last snapshot position may not be successful. Whenever a snapshot is captured, an *odometry motion vector* is also stored. This vector points from the previous position of the robot to the position of the last snapshot vector: $[x_{ss} - x_o, y_{ss} - y_o]$. The robot's previous position from odometry is (x_o, y_o).

The algorithm just described learns the *reverse route* while travelling forwards along the route. At each step, we estimate the ability of the visual homing algorithm to take the robot back to the last snapshot position. The region surrounding a snapshot position within which homing to that position would be successful is known as the *catchment area* [4]. Catchment areas for different snapshots vary in size and shape. The above algorithm gives us some assurance that the route segment between the current position and the last snapshot position lies in the catchment area of the last snapshot. However, we cannot conclude that the last snapshot lies in the catchment area of the current image.

The *forward route* can easily be learned by treating the current image as the snapshot and determining whether the last snapshot position falls within the current image's catchment area. This algorithm would be essentially the same as that presented above, with the exception that the roles of the images are interchanged. For the rest of this paper we will deal only with the reverse route, and will therefore drop the term 'reverse'.

IV. Route Following

We assume at the start of route following that the robot is placed at the beginning of the route. In principle, the stored odometry motion vectors would be sufficient to follow the route. However, in the presence of noise this strategy becomes untenable. We apply a two-stage process for route following. In the first stage, the robot travels along the current odometry motion vector. That is, it simply turns in the direction of this vector and travels by a distance given by the vector's magnitude, scaled by a factor k. The scaling factor k is set to 0.8 in the experiments below. Without this factor it is possible for the robot to overshoot the goal. Overshooting remains possible even with $k < 1$, but its likelihood is reduced.

In the second stage, visual homing is used to guide the agent to the goal. Visual homing thus acts to correct for errors in the odometry motion vector, and for further errors that may be caused by moving along this vector (not simulated here). After each step of visual homing the agent tries to determine if it is sufficiently close to the goal (see below). If so, the next stored snapshot is selected as the goal and the two-stage process begins again.

A. Arrival Detection

For the moment, assume that the visual homing algorithm successfully drives the robot to the near vicinity of the current snapshot position. At this point the robot must *detect* its arrival at the current snapshot position and therefore activate the next snapshot. If the robot fails to detect arrival, it will move towards the goal, eventually arriving in its near vicinity, but then overshooting it. It will then approach again from the other side—and overshoot again. This oscillation will continue indefinitely if no arrival detection mechanism is present. If the arrival detection mechanism is late in detecting arrival then the cost will be increased travel time and energy. Premature arrival detection may cause the robot to switch to the next snapshot prior to arriving in that snapshot position's catchment area. This can cause the robot to lose the route entirely.

The arrival detection method used here is quite straightforward. This method compares the current image to the snapshot and declares arrival when the difference between them is sufficiently small. Image difference is measured by computing the sum of squared error (SSE) between the two images.

$$SSE = \sum_i \sum_j [SS(i,j) - CV(i,j)]^2$$

where SS is the snapshot image and CV the currently viewed image. Arrival is declared when this value falls below the threshold, τ_{sse}.

Depending on the setting of τ_{sse} this method may not detect that the robot has arrived in the vicinity of its current goal. To prevent indefinite oscillation the robot will timeout after 10 homing steps and switch to the next snapshot.

V. Test Environment

The test environment is a simulated world visualized using the POV-Ray ray-tracing program [17]. This freeware program allows photo-realistic images to be rendered from within a simulated environment. Features of this program used here include shadows, anti-aliasing, and panoramic projection. POV-Ray was used for online rendering of the images seen by the homing robot. Therefore, computationally demanding features such as radiosity were not employed.

The simulated environment has the appearance of an art gallery. Five images commonly used as examples in the image processing literature[2] are used as 'paintings'. Twenty paintings were distributed throughout the environment as shown in figure 1. The walls, floor, and ceiling were covered with repetitive textures (e.g. grass, bricks). Illumination was provided by white lights with positions as indicated in figure 1. Panoramic images are rendered by projecting all visible objects onto an upright cylinder located at the position of the viewing robot. Images from within this environment are shown in figure 2.

[2]The images used were the 'baboon', 'earth from space', 'Lenna', 'Walter Cronkite', and 'peppers' images from the USC-SIPI image database [18].

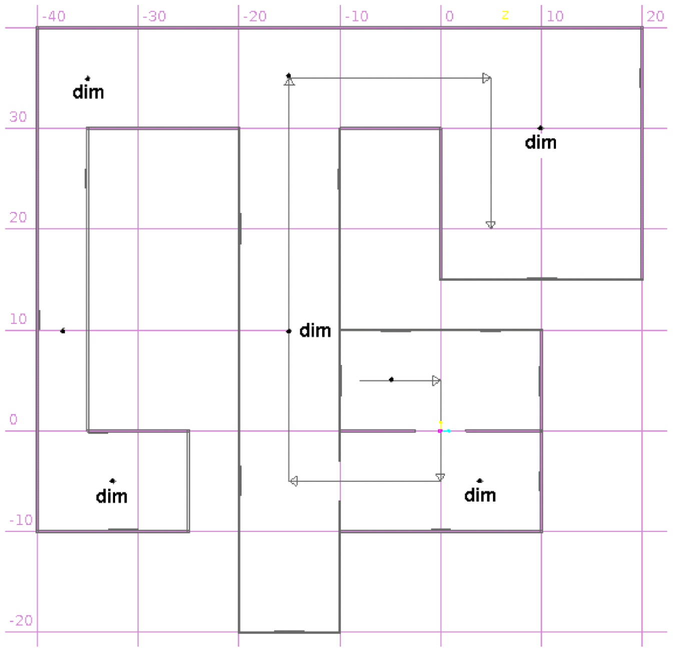


Fig. 1. Overhead view of the test environment. Arrows indicate the learned route. Dots indicate the position of lights, with those labelled 'dim' dimmed for the experiment in section VI-C.

A robot's position within this environment is given by its Cartesian coordinates (x, y) and orientation θ. Movement commands consist of a rotation $\Delta\theta$ followed by a translation d. The simulated odometry system incorporates error according to the odometry motion model presented in chapter 5 of [19]. This model is defined by four parameters: $\alpha_1, \alpha_2, \alpha_3$, and α_4. The movement perceived by the odometry system $(\Delta\hat{\theta}, \hat{d})$ is perturbed from the actual movement by the addition of Gaussian noise,

$$\Delta\hat{\theta} = \Delta\theta + \text{sample}(\alpha_1|\Delta\theta| + \alpha_2 d) \quad (1)$$

$$\hat{d} = d + \text{sample}(\alpha_3 d + \alpha_4|\Delta\theta|) \quad (2)$$

where sample(σ) returns a random sample from a Gaussian distribution with mean zero and standard deviation σ. The new odometric estimate of position (x'_o, y'_o, θ'_o) is calculated from the old estimate as follows,

$$x'_o = x_o + \hat{d}\cos(\theta_o + \Delta\hat{\theta}) \quad (3)$$

$$y'_o = y_o + \hat{d}\sin(\theta_o + \Delta\hat{\theta}) \quad (4)$$

$$\theta'_o = \theta \quad (5)$$

We assume that the robot possesses a compass. Thus, in equation 5, the odometric estimate of orientation is set to the true orientation. This assumption is useful here because it simplifies the task for visual homing. With an accurate compass, an omnidirectional image can be counter-rotated to match the orientation held at the reference position. In this case, only the translation component of motion needs to be estimated by visual homing. A robot can easily be equipped with a magnetic compass, although these tend to be inaccurate

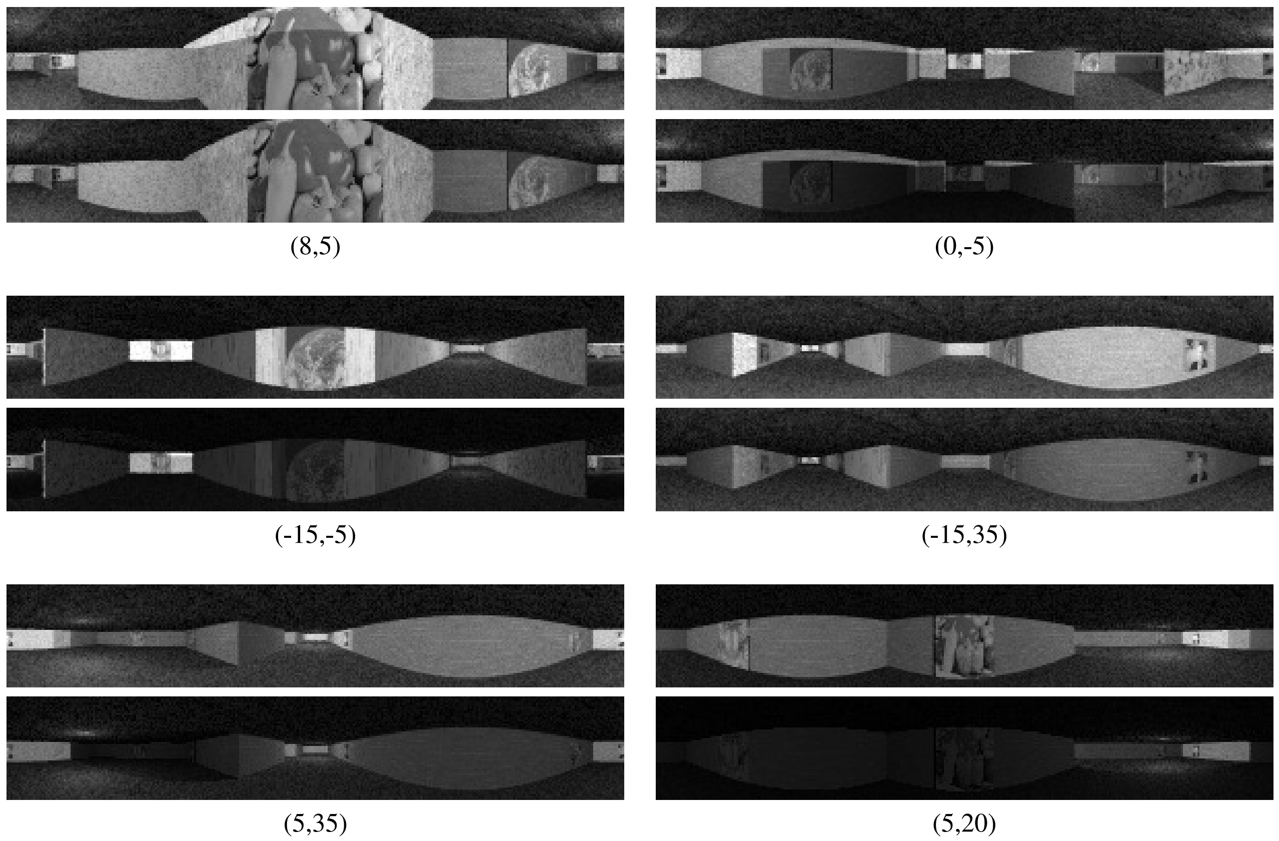


Fig. 2. Panoramic images from the test environment. The top two images in each block of four were generated in the original test environment. The bottom two were generated in the modified environment described in section VI-C. Images were rendered at the corners of the route shown in figure 1.

indoors. Alternatively, a search for the lowest image distance in orientation space can serve as a visual compass [10].

VI. Results

A. Learning

The number of snapshots captured while learning a route depends upon the threshold angle ϕ and upon the degree of odometric error. This dependence was investigated by learning the route described above using various values of ϕ and various degrees of odometric error. The values of ϕ tested were $75°$, $60°$, $45°$, $30°$, and $15°$. The four odometry parameters, α_1, α_2, α_3, and α_4, were all set to the same value, chosen from $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

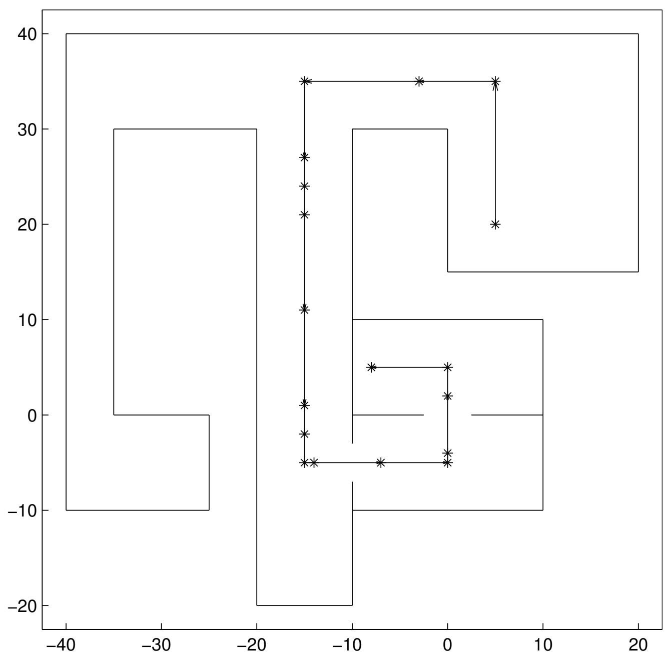
Figure 3 shows examples of learning the route with $\phi = 45°$ and two different levels of odometric error. Figure 4 plots the number of captured snapshots versus odometric error. It is clear that the number of captured snapshots is an increasing function of both ϕ and odometric error. Thus, to obtain a sparse route we must use a relatively large value of ϕ (such as $45°$) and maintain low odometric error.

Figure 5 shows home vector fields surrounding a subset of snapshot positions for the route learned with $\phi = 45°$. These home vector fields are generally of good quality, in that an agent placed in the near vicinity of each snapshot position could home successfully. In particular, it appears that our route following method will be robust to small deviations from the route.

However, some of these vector fields have an asymmetric appearance. This is particularly evident for vector fields captured within the central corridor. Recall that these home vectors are obtained by estimating the gradient of the image distance function. If this function has a symmetric cup-like appearance then the negative of this function's gradient will always point towards the global minimum. However, if this function has a more elongated shape, the gradient will be deflected away from the global minimum. Within the central corridor the image distance function is indeed elongated along the axis of the corridor. This is because movements along the corridor axis produce relatively small changes in the image. However, movements orthogonal to the corridor axis cause a large shift in the apparent position of the corridor walls. Thus, many ceiling or floor pixels become wall pixels, and vice versa, resulting in a large change in SSE. A means of correcting this effect has been suggested by Möller [20].

B. Route Following

In this section we test the ability of a homing robot to follow a learned route. The robot is positioned directly upon the first node of the route. The snapshot image associated with the second node is then activated and route following commences.


(a) $\alpha_i = 0$, snapshots = 18

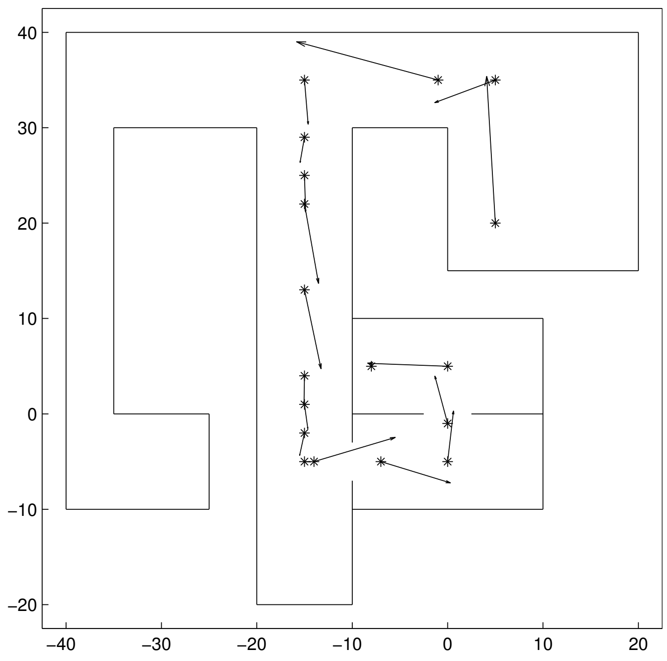

(b) $\alpha_i = 0.4$, snapshots = 18

Fig. 3. Routes learned for $\phi = 45°$ and two different degrees of odometric error. Snapshot positions are indicated by '*' characters. The odometry motion vectors stored at each snapshot are shown.

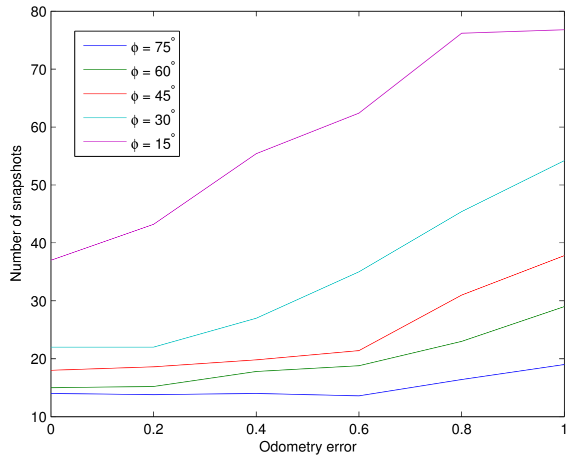


Fig. 4. Number of snapshots learned for the route for varying values of ϕ, and for varying levels of odometry error. Each data point for non-zero odometric error was obtained by averaging the number of snapshots captured across five learning runs.

Homing attempts were made on routes learned with no odometry error. As found in the previous section, increasing the odometric error has the same essential effect on learned routes as decreasing ϕ. Thus the effect of odometric error on learned routes was not tested for route following. Note that current odometry information is not used during path following.

To select the arrival detection threshold, τ_{sse}, candidate values were chosen from the set $\{20, 40, \ldots 200\}$. Route following was successful on the route learned with $\phi = 45°$ for $\tau_{sse} \leq 160$. Route following was judged to be successful if the agent was able to return to within 2.5 units of each snapshot position. Within these successful cases we generally find that higher values of τ_{sse} lead to shorter routes because arrivals are detected earlier. However, if τ_{sse} is set too high then premature arrival detection can occur. This is a serious problem as it can cause route following to fail. Setting τ_{sse} too low merely extends the length of the route followed. Thus, we choose a conservative value of $\tau_{sse} = 60$.

Figure 6(a) shows an example of route following for the route learned with $\phi = 45°$. An oscillatory motion is noticeable in some parts of the route trace. This is due to imperfections in the home vector fields as discussed in section VI-A. The oscillatory motion that results has actually been reduced somewhat by averaging the current home vector with the two previously computed home vectors.

Route following was tested for routes learned with ϕ set to each of the following values: $75°, 60°, 45°, 30°, 15°$. It was found that the route could be successfully followed in all cases.

C. Modified Environment

The ability of the method to cope with a large change in illumination was tested. The lights labelled 'dim' in figure 1

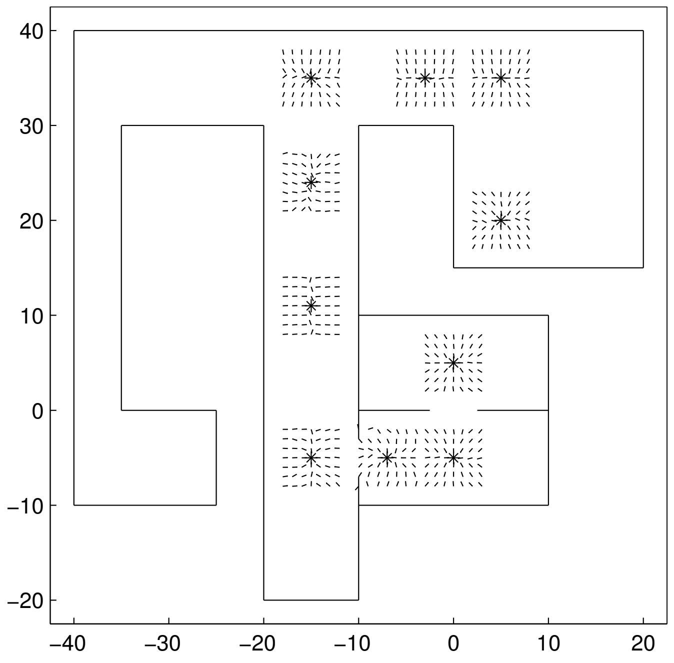
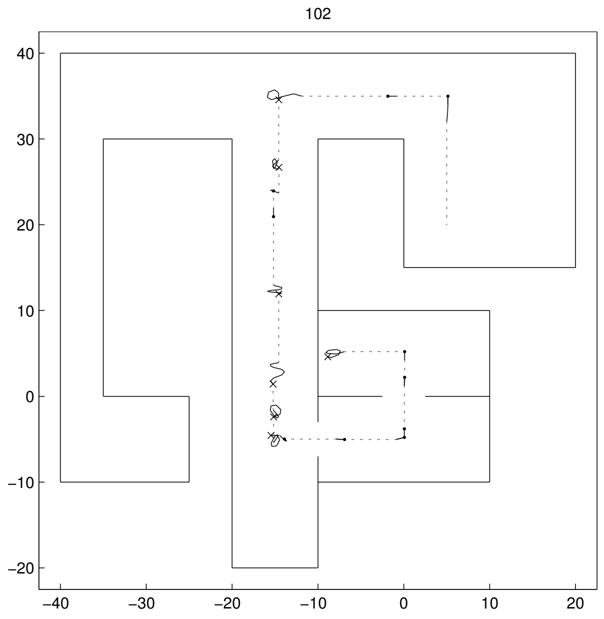


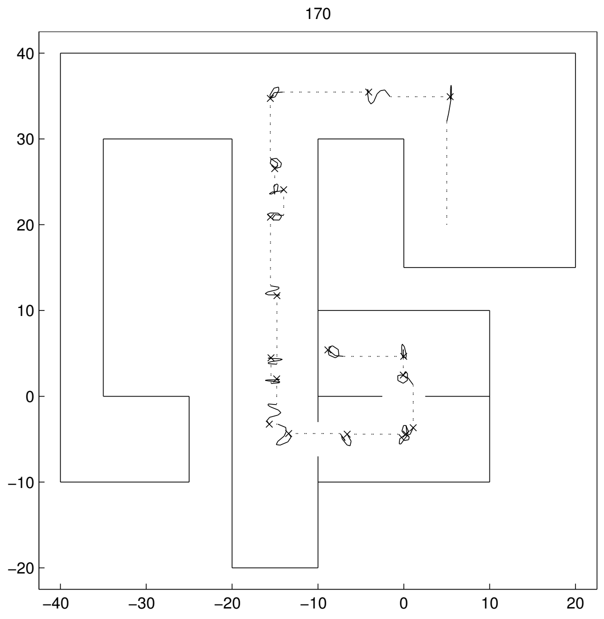
Fig. 5. Home vector fields for a subset of the snapshot positions learned with $\phi = 45°$. This subset was chosen so as to avoid overlap between adjacent home vector fields.

were dimmed by applying a fading function such that the light intensity falls off linearly with distance. As can be seen in figure 2, the change in local lighting conditions is considerable for some positions.

The snapshot images and odometry motion vectors learned for the unmodified environments were used for homing within the modified environment. On the initial test it was found that route following failed for all learned routes. However, an inspection of the route traces revealed that the route was still being followed, but that the distance from the robot to the true route was deviating by a larger amount. Therefore, the threshold for successful route following was increased to 5 units. With this increased threshold, route following was found to be successful for all learned routes. However, for the routes learned with $\phi = 15°$ and $\phi = 75°$ the robot's trace either intersected one of the walls, or came very close to doing so. Therefore, we can only say that route following was truly successful for three of the five learned routes. Performance on the $\phi = 45°$ route is shown in figure 6(b). Note that this particular route took 170 moves to travel along, whereas the same route took only 102 moves in the unmodified environment. Also, no arrivals were detected in the modified environment, whereas 10 out of 17 were detected in the unmodified environment. The lack of successful arrival detection accounts for the increase in oscillatory behaviour which is evident in figure 6(b).



(a) Original environment; Total moves: 102



(b) Modified environment; Total moves: 170

Fig. 6. Routes followed in the original and modified environments (routes learned with $\phi = 45°$). Dashed lines indicate movement along odometry motion vectors. Solid lines indicate visual homing. Transition points between snapshot positions are marked by either a dot (arrival detected) or an 'x' (timeout).

VII. Conclusions

The proposed route learning and following methods were found to perform quite successfully within the unmodified environment for all tested values of ϕ. It was especially encouraging that the following method should succeed even on relatively sparse paths (for $\phi = 75°$ the number of snapshots captured was 14). As expected, the number of snapshots captured while learning a route was found to be an increasing function of ϕ and odometric error. Tests within the modified environment showed that performance was clearly dependent upon some degree of constancy between the conditions of the environment in which the route was learned, and the conditions of the environment in which the route is followed. Nevertheless, route following was found to succeed for three of the five learned paths when the threshold for successful route following was increased.

The route following methods of wood ants and the visual homing methods of honeybees provided inspiration for the methods described here. Yet, at this stage in the research, we cannot make claims as to the relevance of these results to biology. One possible direction of future research would be to replicate the experimental conditions presented in [2] using a mobile robot instead of an ant as the test subject. Comparisons between the performance of the algorithm presented here and related algorithms might allow us to infer possible correspondences with the algorithm used by ants.

Other possible directions of research might leave the problem of the relationship to biology aside and focus on developing these methods for technical applications. One finding of this work has been the difficulty of the arrival detection problem. This is the main focus of current research. Future research efforts will be directed at improving visual homing in the presence of occlusions, studying visual compass methods, and elaborating the proposed method for navigation within a graph of snapshots. It remains necessary to perform a detailed comparison between the method presented here and visual mapping-based approaches such as [21]. The method described here forms only a route-based map of the environment (in the sense described by [5]). Such a map lacks many of the advantageous properties of metric maps, but the question remains whether such a map would be sufficient for a robot to operate autonomously in an unstructured environment.

VIII. Acknowledgements

Thanks for helpful suggestions from Ralf Möller and two anonymous reviewers.

References

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

[2] S.P.D. Judd and T.S. Collett. Multiple stored views and landmark guidance in ants. *Nature*, 392:710–714, 1998.

[3] R. Wehner, B. Michel, and P. Antonsen. Visual navigation in insects: Coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199:129–140, 1996.

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

[5] M.O. Franz and H.A. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems, Special Issue: Biomimetic Robots*, 30:133–153, 2000.

[6] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000.

[7] B.J. Kuipers and Y.-T Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[8] 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.

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

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

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

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

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

[14] U. Nehmzow and C. Owen. Robot navigation in the real world: Experiments with Manchester's fortytwo in unmodified, large environments. *Robotics and Autonomous Systems*, 33:233–242, 2000.

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

[16] R. Möller and A. Vardy. Local visual homing by matched-filter descent in image distances. *Biological Cybernetics (Accepted 14 June, 2006)*, 2006.

[17] POV-Ray. The persistence of vision raytracer, http://www.povray.org/, 2006.

[18] University of Southern California. Image database, http://sipi.usc.edu/database/, 2006.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[20] R. Möller. Newton-based matched-filter descent in image distances. *Biological Cybernetics (Submitted)*, 2006.

[21] R. Sim and J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. In *Proceedings of the IEEE/RSJ Conference on Robots and Systems*, 2006.