Coarse Visual Homing using Azimuthal Cues

Paul Fitzpatrick
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
paulfitz@ai.mit.edu

Abstract

In local homing, an agent returns to a previously visited location by moving to maximize the correspondence between what it sees currently and a remembered view from the target. Clearly this is straightforward if the agent is close enough to the target, and less easy over longer distances. For example, Franz et al [2] give a homing algorithm which works well when close to the target, but the accuracy of which falls off rapidly once the agent is far enough away for features seen from the target to be occluded. This paper presents a scheme that attempts to move an agent into the same basic geometric relationship with its environment as it remembers being in at the target, at which point Franz's homing algorithm stands a better chance of success.

1 Introduction

At its simplest, local homing tries to answer this question: given what an agent sees at its current location, and given a record of what it saw at some target location, in which direction should the agent move to return to that target? For example, honeybees exhibit an ability to return to an unmarked location based on the position of nearby landmarks, which it is suggested they achieve using this type of homing [1]. The same behavior has been implemented in robotic form using a number of different strategies. This paper will present a possible scheme for extending somewhat the currently limited area over which such strategies can be applied successfully.

Local homing, which works only if an agent is sufficiently close to its target to begin with, may not immediately seem useful. Perhaps the clearest way to see its utility is to consider its potential as a building block for homing over greater distances. Clearly if an agent starts far from its goal, comparing the current view with the desired view will rarely yield useful information. But by accumulating a mosaic of known locations, the task of homing from a starting location to some distant target breaks down into short-range homing to a series of intermediate locations [4]. See Franz et al [3] for an implemented system. Alternatively, local homing could also be employed when an agent is able to use dead-reckoning or other cues to return to the general area of its goal, but with some error for which it must compensate. Local homing can make this correction, and prevent errors from accumulating.

Given that local homing is useful, it does not immediately follow that extending the range it can succeed over is a worthwhile proposition. For example, in long-range homing as described above, limitations of the size of the "capture area" over which local homing succeeds can be compensated for by simply enforcing a shorter distance between neighboring homing targets. However, it seems reasonable to build as much robustness as possible into the local homing scheme, rather than trying to compensate for its brittleness at the next layer of abstraction. This paper suggests that such robustness can indeed be achieved.

At an abstract level, honeybees are living witnesses that extremely robust solutions to many of the problems that face robots are possible. A detailed understanding of how insects achieve this competence is potentially useful for robotic applications. Conversely, robotic implementations of homing can advance our understanding of the "rules of the game" that insects have learned to play so well. It is a reasonable hope that a better grasp of the general problems faced by autonomous agents may help to clarify parts of the honeybee's particular solutions. Towards this end, although the details in this paper are slanted towards robotic implementation, the overall design is strongly influenced by aspects of what is known about homing in bees.

The next two sections briefly review two different approaches to homing, as background. A new homing technique is then presented that can extend the range over which local homing is practical. The performance of the technique in a simulated environment is demonstrated through comparison with both honeybees and robotic implementations. Finally some consequences of the work are discussed.

1.1 Image-based homing

One theory of the mechanics behind the homing behavior of bees is that they memorize the retinal image they see at the goal (e.g. at a food source), and then on returning search for the position from which the view best matches

that image [1]. This type of homing is called *image-based homing*.

In image-based homing, there are two questions that need answers: how are qualities of matches between images evaluated, and what search strategy should the agent use to direct its motion. Robotic implementations often avoid directly grading how well images match, and instead compute in which direction the robot could move to most improve the match. The search strategy is then to follow this direction, recompute, move again, and so on [4].

How can the correct direction to move in be calculated? A common approach to homing is to somehow establish a correspondence between features of the current view and the home view. Cartwright and Collett presented a model of landmark navigation in bees along these lines. Their model bee matches dark and light areas of its current view against corresponding areas in the view apparent at the target, and moves appropriately to bring the areas into agreement. Robotic implementations along different lines include those of Hong [4], and Röfer [5]. In [6], Wittmann presented a more detailed model of landmark navigation than that given by Cartwright and Collett. In particular, he elaborated on this problem of establishing a correspondence between the current and home view. This is the difficult part of homing; once the correspondence is known, computing the homing vector is relatively straightforward. When close to the home, it is relatively easy to establish a correspondence, but once prominent features become sufficiently displaced, or occluded, or the environment changes somewhat, the correspondence problem becomes difficult. It certainly becomes extremely difficult to do from local considerations. Any pair of features wrongly put in correspondence can make the homing algorithm give inaccurate results.

A homing technique called *parameterized scene matching* advanced by Franz et al [2] avoids the problem of establishing explicit local correspondences by working over the entire view.

1.2 Parameterized scene matching

A competent image-based local homing algorithm has been implemented by Franz et al in [2]. This algorithm uses a transformation that takes the current view from the agent and "warps" it, based on the assumption that all landmarks are equally distant from the agent, to give an estimate of the view the agent would have if it moved some distance in a particular direction. Assuming this transformation is reasonably realistic for its particular situation, the agent need simply search for parameter values that yield a view closely matching the home view, and then direct the agent to move in the appropriate direction. This form of homing requires no object recognition, since it works over the entire view. Local correspondences are not established, but instead a complete hypothetical scene is compared with the home view.

This works well when all the objects visible in the home view are visible in the current view, and the agent is not too close to any of them. Once objects are hidden behind each other or appear visually to be merged, the algorithm starts to fail. This need not be seen as a fault of the algorithm, but rather a property of local homing, since local homing clearly must fail as the agent's view becomes less related to the home view.

The goal of the work presented here is to expand the range over which local homing is possible. A homing algorithm is given which can work in situations where there is less similarity between the current and home view, tolerating a greater degree of occlusion and alteration to the environment. The intention is to home into an area that has geometric similarities to that seen from the target, and then hand over to Franz's algorithm for returning more precisely to the home. The details of the algorithm are presented in the next section, followed by results showing its performance under various simulated scenarios.

2 Geometric matching

Experiments on honeybees have shown that when homing to an unmarked location whose position is defined by prominent nearby landmarks, the direction to these landmarks appears to be the dominant cue used (see [1] and Section 3.1 for a discussion). The same series of experiments also showed that bees have a sense of absolute direction. For example, when the target location is near a single radially symmetric landmark, the bee will search in the correct direction relative to that landmark. The algorithm developed here is grounded in these results. It uses azimuthal or directional information as its dominant cue. It also assumes that directions are known relative to a global "compass", at least approximately. The exact features the algorithm relies on being extracted from the environment will be discussed in Section 2.2.

The idea behind geometric matching is very simple. It is perhaps useful to introduce it by contrasting it with Wittmann's approach to the homing problem [6]. Wittmann observed that given the current direction to a landmark, and the direction expected for it from the home view, there are an infinite number of possible homing vectors consistent with that. His approach to resolving this ambiguity is to combine information from a number of landmarks. First, a correspondence is established between all the landmarks in the current view and the home view. Then a homing vector is chosen from the range of possibilities for each alone according to some reasonable criterion. The average of these vectors is chosen as the overall homing vector.

Geometric matching starts from the observation that while an infinite number of homing vectors are possible

given the direction to a landmark in the current view and the home view, the directions of all these vectors lie within a definite bound. For example, if a single landmark is viewed due north from the home, and the same landmark is viewed due east from the current position, the homing vector will necessarily lie somewhere between due east and due south. By reasoning about these bounds it is possible to avoid committing to a correspondence between the current and home view, which will in general not be uniquely determined. Geometric matching explicitly evaluates the potential of each correspondence before choosing the homing vector.

Specifically: for each direction the agent can move in, the algorithm calculates an upper bound on the number of features of the current view that could possibly be made to correspond with features visible from the home by moving in that direction. This contrasts with homing schemes which explicitly commit to a specific correspondence between features, which has proven difficult to do robustly. Geometric matching avoids the problem by making a more conservative, but more reliable, statement about the environment. The algorithm cannot state in which direction the home is, but it *can* be quite certain about directions in which few or no features can be brought into correspondence. By moving away from such regions and towards regions that look more promising, the agent is likely to at least not be making its situation worse, and stands a good chance of making it better.

Consider the case where a single feature is known to be seen from the target and a single feature can be seen from the agent's current position. These features may correspond to the same aspect of the environment. If so, then the direction to the home must be somewhere between the current direction to the feature, and the direction from the feature to the home position (known from the home view). Moving in any direction[1] strictly within this range will eventually lead to the feature becoming visible in the same direction as from the goal (see Figure 1).

Given that a feature is visible at angle θ_j in the current view, and angle θ_k in the home view, each direction θ_i between these two is a possibility. The following trivial function captures this for convenience; it will made somewhat more elaborate later.

$$\mathcal{E}(i,j,k) = \begin{cases} 1 & \text{if } \theta_i \text{ lies between } \theta_j \text{ and } \theta_k + \pi \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In general, of course, different features of an object are visible when it is viewed from different directions. The approximation that the features viewed are the same

[1] For convenience, our implementation works over a discrete number of directions, where each direction is associated with the orientation of a single pixel in the agent's view.

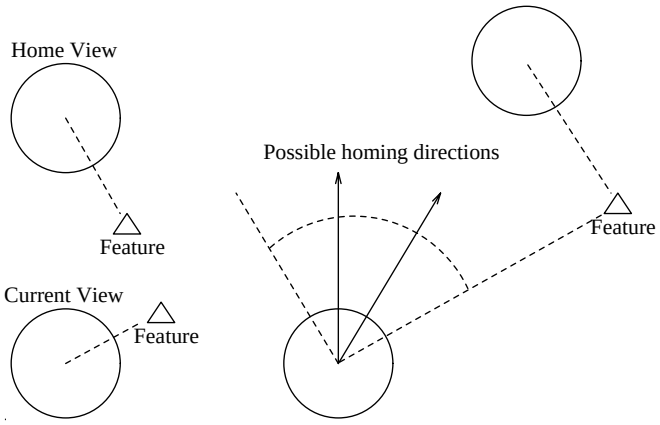

Figure 1: Reconciling the current and home view.

may be good enough to get the agent closer to the correct viewing position, and closer to the point at which the correct feature becomes visible. For large complex objects, this is less likely to be true, but in such cases there may be more features to use.

If there is one feature visible in the current view, and several in the home view, then that feature could be made to correspond with any in the home view. Hence the above process can be repeated for each pairing and the results merged to find all the directions that could result in the feature being seen in the same direction as a feature in the home view. Let ν_i represent this result.

For feature j in current view,

$$\nu_i = \max_{k \in F_0}(\mathcal{E}(i,j,k)) \quad (2)$$

If there are several features visible in both the current and home view, the set of directions that can potentially "explain" each feature in the current view individually can be determined as above. Then the results can be summed, to determine an upper bound on the number of features that could be brought into correspondence in each direction.

$$\nu_i = \sum_{j \in F_t} \max_{k \in F_0}(\mathcal{E}(i,j,k)) \quad (3)$$

Notice that the same feature in the home view could potentially be used to explain several features in the current view. This may seem sub-optimal, but in fact it captures the nature of occlusion in a simple way (to a first approximation).

This is an extremely simple metric which for a straightforward discrete implementation can be calculated for every direction in at worst $O(n^3)$ steps[2] (or $O(n^2)$ with a little optimization), and typically much less since features tend to be sparsely distributed — by their very nature as "special" aspects of the environment. The metric can be converted to a homing vector in many ways. The

[2] where n is the number of possible directions.

simplest is to take the weighted sum of unit vectors in each of the directions.

This simple algorithm works surprisingly well. Because it seeks explanations of what the agent sees currently in terms of what it saw at the home view, it is generally not badly affected if a landmark has been removed since the agent was at the home (see Section 3). If landmarks are added, then it is better to explain the home view in terms of the current view. A hybrid combining the two works particularly well.

If distance information is available for any or all of the features, this can be superimposed as a weighting on the likelihood of the explanations above, since for a given distance to a feature in the current view and a given distance to a feature in the home view, there is a direction that works best to reconcile the two. See Figure 2.

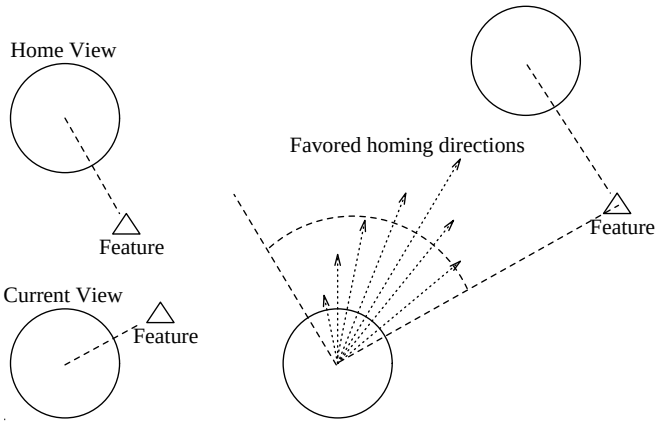

Figure 2: Using distance estimates

With distance information available, it is possible to calculate exactly the direction the agent should move in to reconcile a feature in the current view with one in the home view. Given that the agent will be simultaneously trying to reconcile other features, it is best to evaluate how good a match will result for each of the possible directions. This can be done by modifying the $\mathcal{E}(x)$ function used earlier as follows in the range where the original function returns unity:

$$\mathcal{E}(i,j,k) = \mathcal{M}\left(\frac{d_{j,t}}{d_{k,0}}\frac{\sin(\theta_j - \theta_i)}{\sin(\theta_k - \theta_i)}\right) \tag{4}$$

The above formula amounts to calculating how far from the feature the agent will be if it moves in the given direction long enough to bring the features into correspondence[3]. A normalization is performed against the desired distance, and then the quality of the explanation is calculated by the function $\mathcal{M}(x)$. This function is chosen to peak for $x = 1$ and fall off for x values to either side of 1. The algorithm appears insensitive to the exact nature of the metric beyond this qualitative constraint. Currently the function is chosen somewhat arbitrarily as:

$$\mathcal{M}(x) = \begin{cases} \frac{1}{x^2} & x > 1 \\ x^2 & x \leq 1 \end{cases} \tag{5}$$

The algorithm with distance cues tends to give faster convergence to the target than the simpler version, although it is not always as robust to occlusion. However, certain alterations to the environment that change its geometry significantly can only be dealt with using distance cues. If distance cues are extracted in the naive way that will be proposed in Section 2.2, using them places demands on how the agent moves. A reasonable approach is to use the simpler algorithm while its confidence is high[4], and then start to extract and factor in distance cues once its confidence becomes lower. Hence the agent could move in an unconstrained way until it is closer to the target. Another approach is to compute from distance cues at a rate inversely proportional to the confidence of the simpler algorithm. This could give faster convergence with lower constraints on the agents path, but has not yet been tried.

2.1 Accumulating geometric information

As the agent moves, the arrangement of the features it sees will change. Recall that the algorithm calculates an upper bound on the number of features that can be explained in a given direction. If it is assumed that the agent is distant enough from the target for the angle to the target not to be changing rapidly, then the upper bound in that direction will remain high, since there truly is a high number of features that can be explained in that direction. Hence the minimum of the upper bounds the agent calculates as it moves can be taken in each direction. This will tend to leave the homing direction unchanged, and will improve the tightness of the bounds in other directions. Of course, this is an approximation, but it turns out to work quite well. If the agent is close to the target then it has essentially completed its task anyway (to get into the right geometry to use parameterized scene matching). If the agent is further away, the approximation works well. As the agent gets closer to the target, the homing algorithm is more likely to have turned it to move directly towards the target, in which case the approximation will continue to work. Our current implementation of this, which for simplicity uses an averaging process with exponential decay that approximates the above, can lead to some overshoot and oscillation close to the target. This can be detected

[3]Since all angles are discrete, a look-up table can be used for $\sin(x)$.

[4]The algorithm is here considered "confident" in proportion to the difference between the highest and lowest bounds it calculates on the number of features that could be matched in different directions.

and used as a condition to switch to parameterized scene matching.

2.2 Cues for geometric matching

Unlike scene matching, which works over the entire view, geometric matching requires cues to be extracted from the environment. Edges are suitable cues for the simple direction-driven version of the algorithm. For inferring distances, an extremely rough approximation of optical flow was used. The algorithm was driven with cues extracted from regions of the view that are noted to be changing as the agent moves. These are termed *blurs* in this paper[5], and correspond to regions where optical flow information can be determined locally. No attempt is made to propagate this information to points where optical flow is not locally determined.

If there is an edge in view with no other sharp change in the image for some range to the left and right of it[6], then the distance to that edge can be approximated from the gross width of the segment of the image that changes around it in a fixed time interval and for a fixed displacement of the agent. If the displacement is at right angles to the direction to the edge, this effect is most pronounced. If the agent moves in a circle and merges differences in the view it sees as it goes, then the width of the "blur" will effectively be the same as if the agent had moved along the diameter of the circle perpendicular to the edge, since that gives the widest effect.

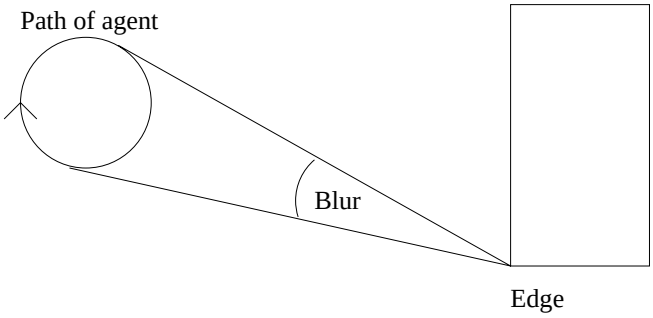

Figure 3: An idealized blur.

The agent does not need to move in a complete circle as described above. Sampling three or four points on the circle is an acceptable approximation for the purposes of geometric matching. Four points were used by our scheme. Note that, while moving like this would be acceptible when learning the home view, it does not lead to a particularly elegant path of the agent during homing itself. The path must zig-zag at the least if distance cues are needed for every object in its field of view.

If the distance cue is a blur, then the distance can be calculated from the width of the blur in radians $\Delta\theta$ as:

$$d = \frac{r}{\tan(\frac{\Delta\theta}{2})} \tag{6}$$

where r is the circle radius. For small angles, $\tan(\Delta\theta)$ can be approximated as $\Delta\theta$.

$$\mathcal{E}(i,j,k) = \frac{w_{k,0}}{w_{j,t}} \frac{\sin(j-i)}{\sin(k-i)} \tag{7}$$

Note that this entire approach is subject to error. Edges that are near each other can combine to appear as a single closer edge, for instance. However, it proved adequate for geometric matching, and is certainly simple. Much better approaches are possible.

3 Results

The results presented here are for the algorithm operating in a simulated environment. It has not yet been tested in a physically embodied implemention, so the results should be treated as preliminary. However the algorithm will be shown to be robust under simulated sensor noise, directional uncertainty, and environmental mutation.

The simulated agent has a 360° horizontal view of its environment, divided into 160 pixels, and with no vertical extent. This is an idealization of the robotic implementations using spherical or conical mirrors placed above a camera, or equivalent arrangements [2, 4, 5].

3.1 Comparison with the honeybee

First it will be shown that the algorithm behaves "sensibly" for various types of environmental mutation. To make this less subjective, the honeybee will be used as the benchmark against which the algorithm is judged. In particular, the experiments Cartwright and Collett performed with bees in [1] will be applied to the algorithm. See [6] for another homing algorithm compared with Cartwright and Collet's findings.

In the series of experiments described in [1], the bees were trained to an inconspicuous food source placed close to an arrangement of prominent landmarks (matte black against a white background). Then the source was removed, and the searching behavior of the bees recorded. Making the reasonable assumption that the bees would spend the most time searching in the location they believed most likely to be where the food source was, Cartwright and Collett could learn something of how the bees evaluated this likelihood. They did this by changing the arrangement of landmarks when the food source was removed, and examining the effect of this change on the bees' judgement.

The same will now be done for the geometric matching algorithm. It will be given a home view taken with one set of landmarks, and then asked to direct the agent

[5] The name is chosen to be more evocative than accurate.

[6] As will be discussed in Section 3, the simulated agent has a horizontal view with no vertical extent.

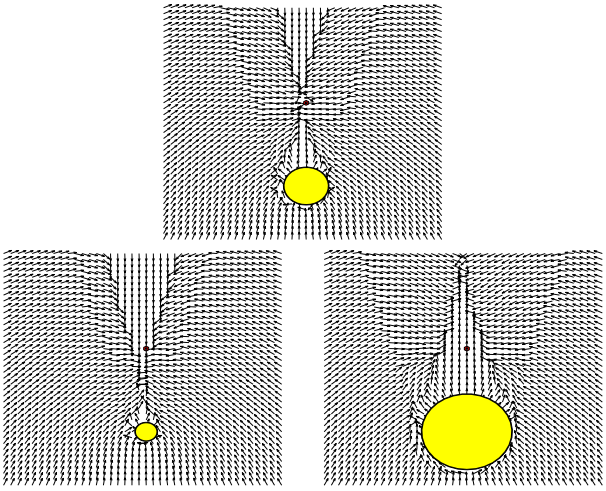

Figure 4: Effect of landmark size. Agent trained with top configuration, and tested with all three. Arrows indicate homing vectors generated by the algorithm.

in an environment in which those landmarks have been modified in some way.

The simplest experiment trained the bee to a food source at a certain distance from a single landmark, and tested where the bee searched if the landmark was halved or doubled in size. Its behavior was to search approximately at the point where the landmark appeared the same size to it as when it was trained. As Figure 4 shows, the geometric matching algorithm does the same. The bee and the algorithm also demonstrate that they are aware of direction in a global sense in this experiment — otherwise they could only search in a ring around the landmark.

Cartwright and Collett carried out a series of experiments with three landmarks to determine the relative importance of landmark size versus azimuthal direction for the bee. Changing the landmark size had no effect this time — the bee searched approximately where the landmarks were at the same angle as it had viewed them during training. The algorithm does the same (see Figure 5).

Moving the landmarks out on a ray from the home also has no effect on bees, confirming that azimuthal angle is the dominating cue in this situation. The algorithm also follows this pattern (see Figure 5).

Other distortions that bees (and the algorithm) handle similarly include addition of extra clutter between the landmarks, and distortion of the landmarks so that the bee has a choice between searching where the distances are correct or where the azimuthal directions are correct. This last is resolved in the same way by both agents — they search where the directions are correct (Figure 7), in preference to where the distances are correct.

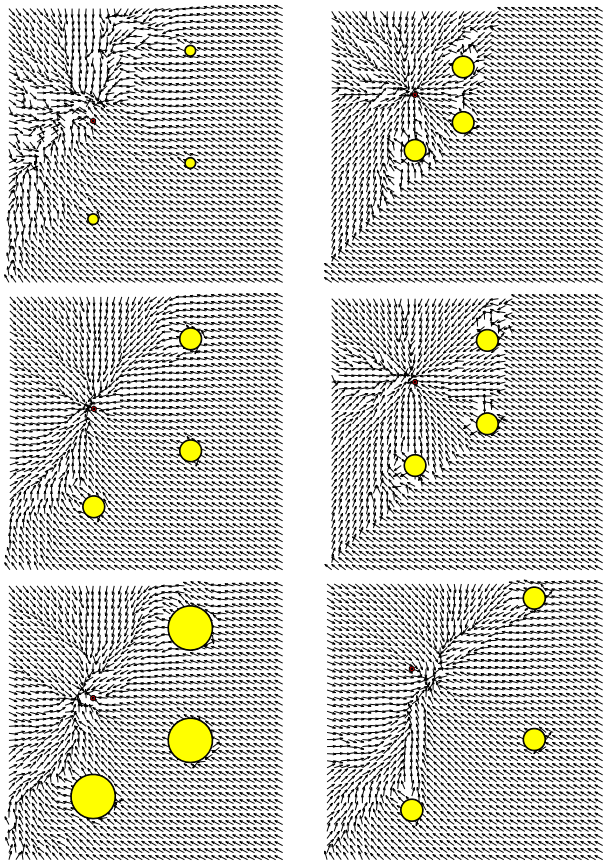

Figure 5: Effect of changes in landmark size and distance when target is defined by azimuthal cues. Training array in middle in both cases.

The bee and the algorithm can cope with removal of landmarks (Figure 6). Removal of the top landmark is actually the first and only case here where distance cues are necessary for the algorithm to work correctly.

Rotation of the landmarks can be coped with to an extent by both algorithms. A 90° rotation causes both bee and algorithm to fail. The algorithm fails more dramatically than the bee (Figure 7).

Cartwright and Collett tried some cases with greater numbers of landmarks, not with the bee but with a simulation they made of it. The corresponding results for the geometric algorithm are shown in Figure 8. The algorithm works reasonably well for a cluster of nine objects, and also with the standard configuration where distant distracting landmarks added.

The simulation was also run on a square arrangement adjacent to a rectangle, with the agent trained to the square alone. The real bee shows much more intelligent behavior in this case than either Carwright and Collett's model or our own. The behavior of the algorithm in this case is shown in Figure 9.

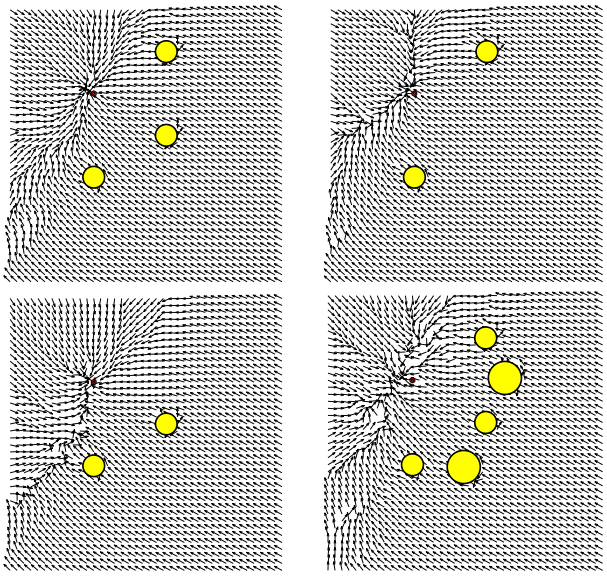

Figure 6: Effect of removing or adding landmarks.

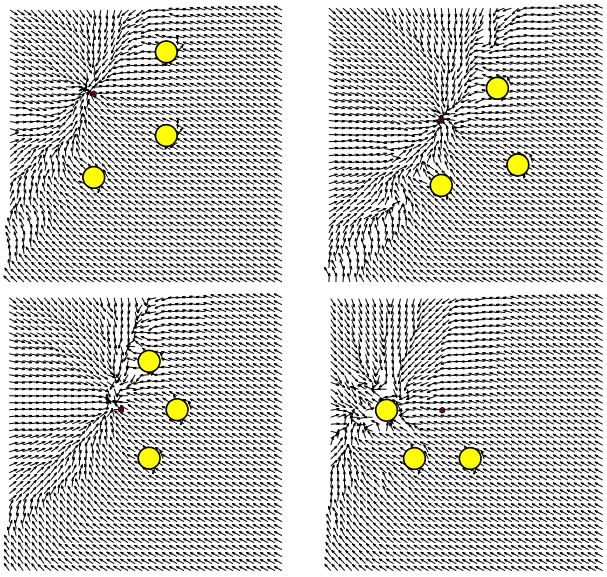

Figure 7: Effect of rotating the configuration of landmarks. Top right shows effect of forcing a choice between distance and azimuthal angle.

A final trial[7] was performed where the algorithm was trained with the standard array of three landmarks used by Cartwright and Collett, and then exposed to a configuration with a duplicate of the array incrementally added beside it, one landmark at a time (see Figure 10). For one extra landmark, homing is unaffected. For two, in this geometry homing is disturbed somewhat. The location homed to is close enough to the actual home for scene matching to correct for the offset. For three extra landmarks, there are significant homing errors.

[7] This is not one of the Cartwright and Collett experiments.

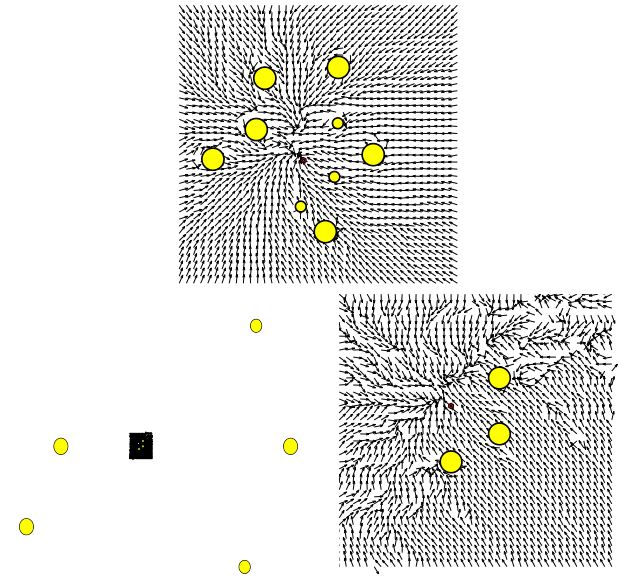

Figure 8: Dealing with large numbers of landmarks. Bottom right is an expansion of the center of the configuration shown to its left.

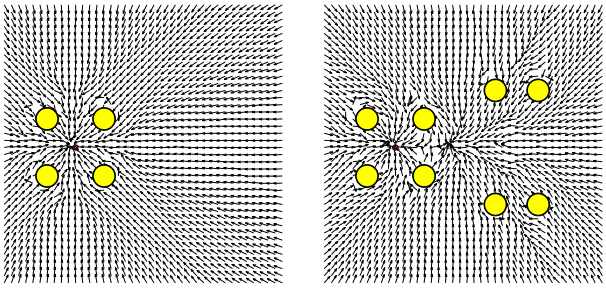

Figure 9: Discrimination experiment.

3.2 Comparison with scene matching

This section compares the properties of geometric matching with those of parameterized scene matching. This is an unfair comparison for a number of reasons. First, parameterized scene matching does not require the compass information that geometric matching takes as given. Secondly, parameterized scene matching has been implemented physically while geometric matching has currently only been tested in an idealized simulated environment. To make the comparison slightly fairer, parameterized scene matching was implemented in our simulated environment and given the same compass information as

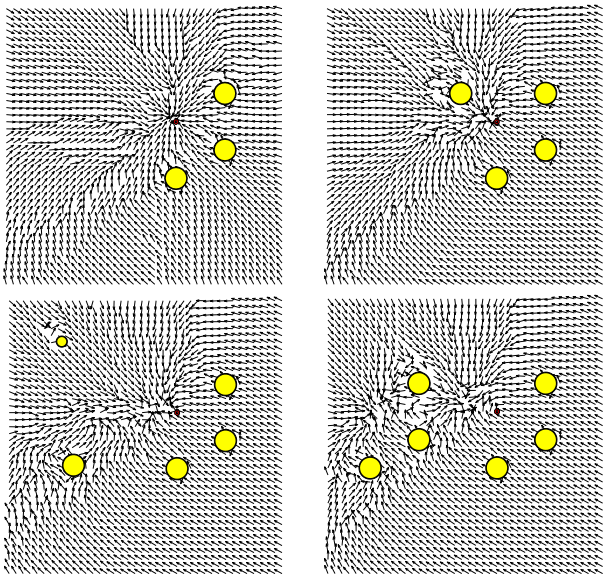

Figure 10: Effect of duplicating landmarks.

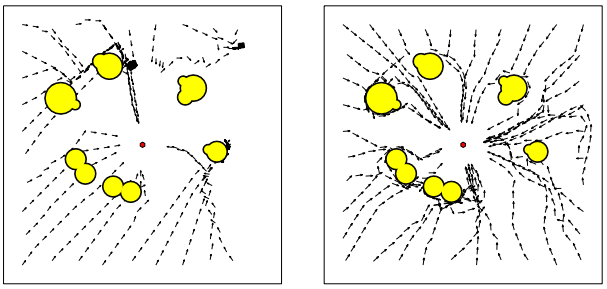

Figure 11: Performance of scene matching when far from home. Scene matching on left, geometric on the right.

geometric matching. Since compass direction is simply one of the parameters that parameterized scene matching searches over, the algorithm can readily make use of this information.

Figure 11 show the results of a trial testing the algorithms in an environment that (*very* approximately) models that used by Franz et al to test their robot. Note that the homing algorithm is being tested on areas outside the ring of landmarks, where it was not designed to work. Within the ring, it works robustly. Also, geometric matching has the advantage of some naive obstacle avoidance behavior.

In Figure 12 the two algorithms are placed in a very simple environment with just two landmarks, chosen to illustrate the value of using geometric matching before scene matching. When the two objects are visible, both algorithms succeed. When one is occluded, scene matching has no way to use its knowledge that there should

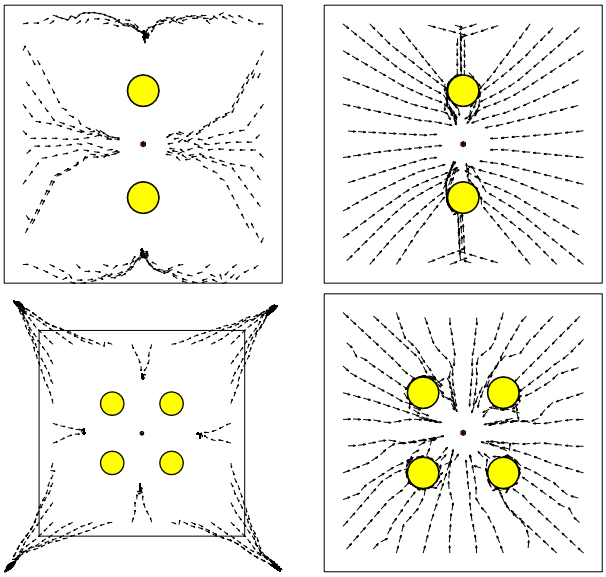

Figure 12: Geometries where scene matching is difficult. Results of scene matching on left, geometric matching on right.

be two objects in view, and so homes to a suboptimum point. Geometric matching successfully returns to the correct geometric location. Obstacle avoidance is invoked since the target lies at the opposite side of an object. Note that in this case, with only two objects, if one of the objects had actually been removed, scene matching would work sensibly whereas geometric matching would end up cycling.

A similar situation occurs in the lower half of Figure 12. Here there are four landmarks arranged in a rectangle. If the agent is approaching from outside, landmarks may be either occluded or appear as a single larger object. Geometric matching survives this, but scene matching does not. In this particular case, seen from a diagonal the landmarks merge into a single landmark, and the agent moves outwards to equalize the size of that landmark with one of the actual landmarks seen from the goal. Seen from another angle, the landmarks merge into a pair of landmarks, again driving the agent to a suboptimal solution.

3.3 Noise susceptibility

This section examines how robust geometric matching is to various forms of noise. In particular the consequences of vision noise, quantization effects, and compass errors will be examined. Cartwright and Collett's standard array was arbitrarily chosen as the test case with which to demonstrate these effects.

Figure 13 shows how the algorithm behaves under imposed noise in its vision input. The noise level is intro-

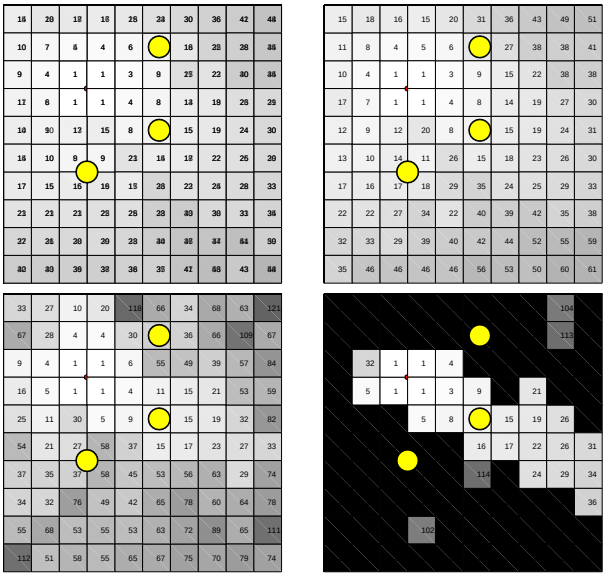

Figure 13: Effect of vision noise (no noise in top left, 10% noise top right, 20% bottom left, 25% bottom right).

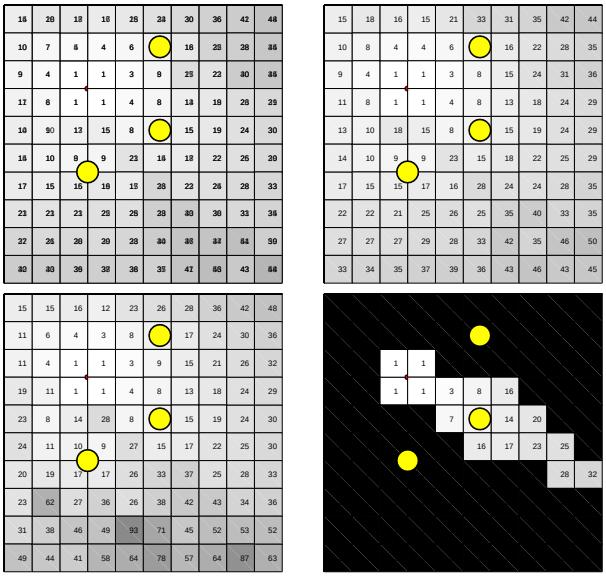

Figure 14: Effect of resolution change (160 pixels top left, 80 pixels top right, 40 pixels bottom left, 20 pixels bottom right).

duced as a probability that *for each direction* the agent will imagine a feature to be present there that is not, or will fail to see a feature that is in fact present. The figure gives the number of steps[8] the agent takes to get to the goal from an array of starting points with and without imposed noise. At 25% noise the algorithm fails quite dramatically. Here there are at most 6 true features visible to the agent, and at this noise level an average of 40 imaginary features are being hallucinated into existence.

The effect of decreasing the resolution of the views is shown in Figure 14. The algorithm fails once the resolution falls below about 30 pixels.

Next the effect of feeding the algorithm incorrect compass readings is examined. The results for the standard array are shown in Figure 15. Note that the agent will no longer home to exactly the right place, but the results show that it will nonetheless pass close to the home at some point. This is quite dependent on the geometry of the environment, but in general the algorithm is robust to 30-45° angular error.

It is worth examining the behavior of the algorithm under various sources of noise simultaneously. The left grid in Figure 16 is the test case with no noise. The right grid has 10% noise in detecting features, 20° error in the compass direction, and one object missing.

Finally, the effect of removing a portion of the agent's rear field of view is examined (see Figure 17). Note that the results here are somewhat worse than they appear. The agent tends to home correctly initially, reach the goal, and then simply fly on away from the landmarks.

[8] The dimensions of the figure are 50 steps by 50 steps.

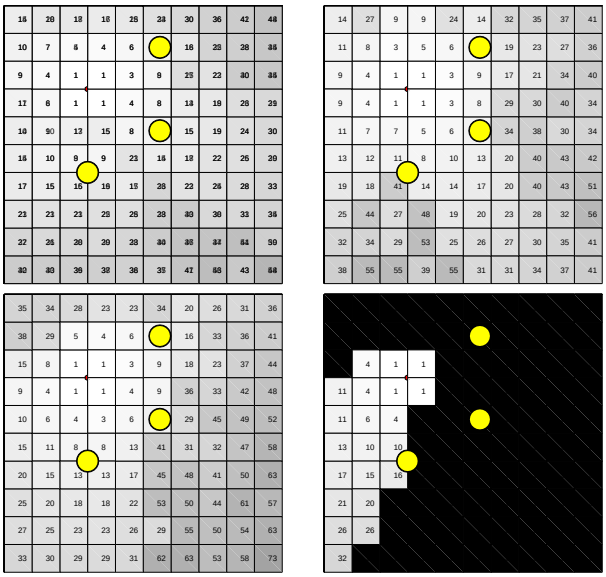

Figure 15: Effect of incorrect compass readings (0° offset in top left, 30° in top right, 60° bottom left, 90° bottom right).

4 Discussion

This paper has advanced evidence that local homing can be made quite robust. Directional cues, which honeybees apparently use in preference to other properties such as distance and landmark size, do indeed seem to be well

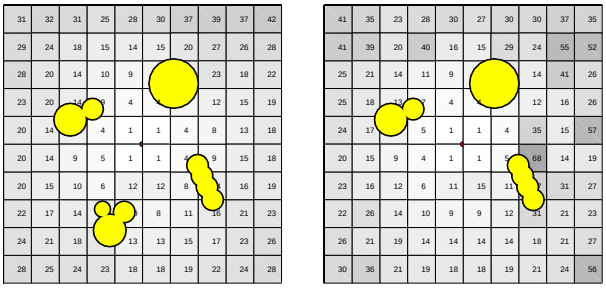

Figure 16: Effect of simultaneous sources of error

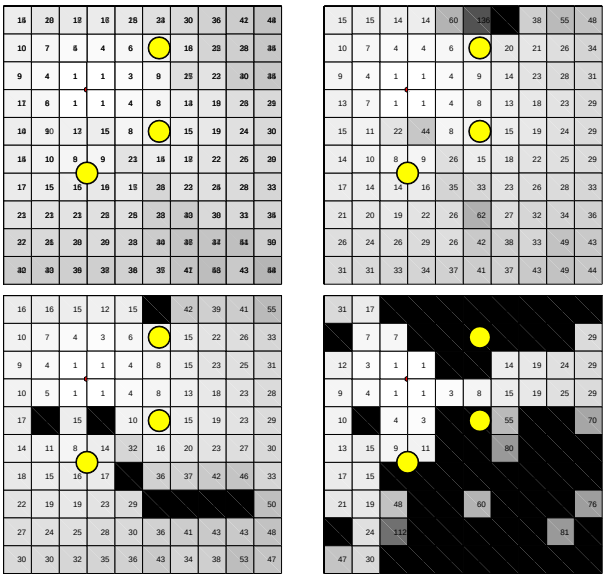

Figure 17: Effect of blocking the agent's rear view (0° top left, 30° top right, 90° bottom left, 180° bottom right).

suited as primary cues for solving aspects of the homing problem competently.

In Cartwright and Collett's early study on landmark learning in bees [1], there is a discussion of how bees resolve conflicts between matching directions to landmarks and matching their apparent size. The algorithm described here makes no explicit representation of the apparent size of an object, but still behaves as the bee does. When there is a single landmark, the algorithm homes to put the landmark in the right direction. More accurately, it tries to put the two features at the edges of the landmark in the right directions. Hence it will clearly home to the point where the apparent size of the landmark is correct. This suggests that matching directional cues will also match landmark size when possible, and conflicts between directional and size cues need not be resolved separately from conflicts between directional cues alone.

The algorithm relies on distance cues to resolve some ambiguities. This places constraints on how the agent must move if it is to obtain these cues. In general, robotic implementations of homing algorithms drive the robot in a very simple way, by merely following the directions of the algorithm blindly. In contrast, bees follow a complex, erratic seeming path when searching. Two-way interaction between an agent's search strategy and a homing algorithm could potentially improve both. This seems an interesting area for further research.

The algorithm also depends on having a rough knowledge of the direction the home image was taken in. In the absence of a compass, dead-reckoning is the most obvious alternative. For a single homing task, the tolerance for error is probably sufficient for this to work, since rotational components of motion are particularly easy to estimate and compensate for with a 360° view available. For a sequence of homing tasks, where for example the agent is working its way towards a distant goal, dead-reckoning will be subject to failure due to accumulating errors. However, if the direction a home view was taken in is known relative to the direction a neighbor's home view was taken in, then the situation is more hopeful. Franz's parameterized scene matching does not require directional information, and so can be used to correct for dead-reckoning errors at each step so they don't accumulate.

The algorithm at its heart assigns weights to explanations of features in the current view by features in the home view. If extra information about those features becomes available, such as the texture or color of the surfaces they appear on, then that information can easily be combined into the weighting system to produce a more informed and presumably more competent algorithm.

References

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

[2] M. O. Franz, B. Scholköpf, and H. H. Bülthoff. Homing by parameterized scene matching. In P. Husbands and I Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life*. MIT Press, 1997.

[3] M. O. Franz, B. Scholköpf, P. Georg, H. A. Mallot, and H. H. Bülthoff. Learning view-graphs for robot navigation. In W. L. Johnson, editor, *Proceedings of the 1st International Conference on Autonomous Agents*, pages 138–147. ACM Press, 1997.

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

[5] T. Röfer. Controlling a robot with image-based homing. In B Krieg-Bruckner, G Roth, and H Schwegler, editors, *ZKW Bericht Nr. 3*. Report series of the Center for Cognitive Sciences at the University of Bremen, 1995.

[6] T. Wittmann. Modeling landmark navigation. In B Krieg-Bruckner, G Roth, and H Schwegler, editors, *ZKW Bericht Nr. 3*, 1995.