Using the Average Landmark Vector Method for Robot Homing

Alex GOLDHOORN [a,b] Arnau RAMISA [a,1] Ramón López DE MÁNTARAS [a]
Ricardo TOLEDO [c]

[a] *IIIA (Artificial Inteligence Research Institute) of the CSIC*
[b] *University of Groningen, The Netherlands*
[c] *CVC (Computer Vision Center), Spain*

Abstract. Several methods can be used for a robot to return to a previously visited position. In our approach we use the average landmark vector method to calculate a homing vector which should point the robot to the destination. This approach was tested in a simulated environment, where panoramic projections of features were used. To evaluate the robustness of the method, several parameters of the simulation were changed such as the length of the walls and the number of features, and also several disturbance factors were added to the simulation such as noise and occlusion. The simulated robot performed really well. Randomly removing 50% of the features resulted in a mean of 85% successful runs. Even adding more than 100% fake features did not have any significant result on the performance.

Keywords. Mobile Robot Homing, Average Landmark Vector, Invariant features

1. Introduction

The objective of this work consists in evaluating the suitability of a biologically inspired visual homing method, the ALV (Average Landmark Vector), for a mobile robot navigating in an indoor environment. For these first experiments, we have developed and used a simulated environment to assess the robustness of the algorithm to different levels of noise and changes in the scene.

Based on the work of Wehner [9], Lambrinos et al. studied in [4] the navigation strategies of an ant species which lives in the Sahara, the Cataglyphis. This ant cannot use pheromones (hormones which are dropped by the ants in order to 'define' a path, these hormones can be detected by other ants) to find their way back, because the pheromones evaporate in the desert. Three main strategies were found to be used by the Cataglyphis: path integration, visual piloting and systematic search. Path integration with the help of a biological compass is the main ant navigation technique, but either visual piloting or systematic search, depending on the availability of landmarks, are also used to finally find the nest.

[1]Correspondence to: Arnau Ramisa, Campus Universitat Autonoma de Barcelona, 08193 Bellaterra, Catalonia, Spain Tel.: +34 93 5809570; Fax: +34 93 5809661; E-mail: aramisa@iiia.csic.es.

The snapshot model has been used to explain the insect visual piloting techniques for about two decades [1]. This model assumes that a panoramic image of the target location is created and stored by the animal. When an insect wants to go back to the stored position it uses a matching mechanism to compare the current retinal image to the stored panorama. The authors of [4] suggest the Average Landmark Vector model as a better way to explain the comparison. This model assumes that the animal stores an average landmark vector instead of a snapshot. Landmarks can be (simple) features like edges. The direction to the destination is the difference of the ALV at the destination and the ALV at the current location. A significant advantage of the ALV is that it does not use any matching technique, thus being a much less computationally expensive method that can be used in real time. Posterior to this first simulated evaluation we plan to test this method on a real robot and, if the results are satisfactory, we will use it to complement the topological navigation system we are developing.

In [6] the authors successfully tested the ALV method on a small real robot under complete analog control, however this was in an 1 m $\times$ 1 m environment with walls of 30 cm heigh and black pieces of paper on the wall which served as landmark.

In [7], Smith et al. propose some improvements for the ALV to use it in large scale environments, and test it in a computer simulation. In their simulation an unidimensional sensor for the robot is used, and the landmarks are cylinders of different sizes spread across the environment. Their experiment is designed to simulate the biological model of an ant colony where ants have to reach some food in the environment and then return to the nest.

Another difference between our work and that of Smith et al. is that we designed our simulation to use automatically extract features from panoramic images. In a real robot, these features can be extracted from the images using, for example, a Harris corner detector or DoG extrema, and we do not need to manually set up special landmarks within the environment.

The rest of the paper is divided as follows: In Section 2 the ALV visual homing method is explained. In Section 3, the computer simulation environment is explained as well as the experiments designed to test the performance of the ALV method. Section 4 discusses the results, and finally, in Section 5 we present the main conclusions extracted from the results and we mention some future work.

2. Method

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

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

Where F is a n by 3 matrix containing the positions of the n features. F_i is the *i*th feature position vector and $\overrightarrow{x}$ is the current position of the robot. Eqn. 1 shows the ALV in a world centered system, therefore the current position of the robot ($\overrightarrow{x}$) is substracted.

The next step is to calculate the direction to the destination (or home) this is called the homing vector. It can be calculated by subtracting the ALV at the destination location (i.e. home: $\overrightarrow{d}$) from the ALV at the current location ($\overrightarrow{x}$):

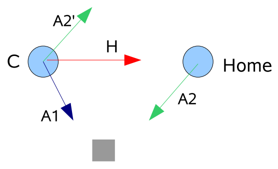

Figure 1. The calculation of the homing vector. Both ALV's (A_1 and A_2) point to the average feature position, which is drawn as a gray block. The homing vector (H) is calculated by subtracting the ALV at the destination location (A_2) from the ALV at the current location (A_1). This subtraction is shown, by the addition of the inverse vector, A_2', to A_1. The robots are aligned in this example.

$$homing(F, \overrightarrow{x}, \overrightarrow{d}) = ALV(F, \overrightarrow{x}) - ALV(F, \overrightarrow{d}) \qquad (2)$$

In this work we will use automatically extracted features from panorama images. One of the most relevant problems for using this method in the real world is that the panoramas have to be aligned before the homing vector can be calculated, this is not taken into account in the simulation. However we have done tests with random rotation of the panoramas to perceive the effect of non-alligned panoramas.

Figure 1 shows an example of the calculation of the homing vector. To simplify the image we only show the average feature position (the gray square). The ALV is calculated from two positions: c_1 and c_2 where the panoramas were taken. The ALV's from both panoramas are shown: A_1 and A_2. The homing vector is calculated (with c_1 as current position and c_2 as home; figure 1) by subtracting the ALV at the destination position (A_2) from the ALV at the current position (A_1). This can be done by adding the opposite vector of A_2 (shown as A_2') to A_1. This results in the homing vector, H_1, which points to the destination location. It can be easily proven that the destination will be reached when there is no noise present [4, 3].

In the previously discussed example the depth was known and therefore the destination was found in one step. However in our examples the depth is not known, because we only use one panorama image, therefore the process of calculating the homing vector has to be repeated several times.

A constraint of the homing method discussed here is that it assumes an isotropic feature distribution [2]. This means that the frequency and distance of the features are independent of the location where the panoramas were taken. A last constraint is that the features are static, that is they do not change over time.

3. Simulation and experiments

The simulation environment contains one or more walls which make up a room. These walls are defined by feature points (the dots in figure 2). In the simulation the robot only moves in a plane (xz-plane), so the robot will not change its height (y-direction). The camera is set 1 m from the ground (at $y = 1.0$ m).

Four slightly different worlds have been created where the centre of the room is set to (0, 0) on the xz-plane. Figure 2 shows the first world (world 1) and it shows the letters for each wall. The figure also shows the start and end location of the robot. The other worlds are similar to the first world, but not all the walls are present. Table 1 shows in which world which walls are present. The walls have a height of 2 m and a roughness of

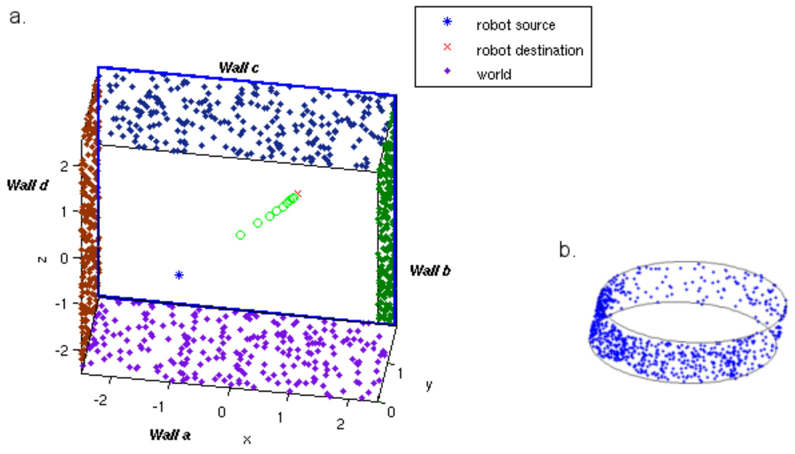

Figure 2. a. World 1 of the simulation. In this case the world consists of four walls (world 1). The dots represent the features by which the robot recognizes the wall. The robot origin (source) and destination are also shown and the path which the robot took (green circles). b. A panorama projection image at the destination location.

Table 1. The different worlds used in experiment 1 and per world which walls were present.

World	Description	Wall a	Wall b	Wall c	Wall d
1	All walls	x	x	x	x
2	2 sided, I	-	-	x	x
3	2 sided, II	x	-	x	-
4	1 sided	-	-	x	-

1 mm (i.e. the deviation from the 'straight' line). The locations of the features on each wall have been randomized (uniformly) before each test-run of the virtual robot.

The default amount of features for each world is 1000. These features are uniformly distributed in all the present walls (for example, in a world with four walls, each wall will have 250 features).

The virtual robot acquires information about the environment using a simulated omnidirectional camera, which captures a cylindrical projection of the features of the world at its current location. The images of the feature points in the panoramic image are used to calculate the ALV as in Eq. 1. The next step is to compute the homing vector (Eq. 2).

Which features are visible to the robot depends on their closeness and their y-position (height). The image plane has a limited vertical field of view, but it has a 360° view in the xz-plane. To test the performance of the ALV, we added Gaussian noise to the positions of the features before projecting them. We also added the possibility of features to disappear or new features to appear during a test-run.

3.1. Experiments

The influence of the variables listed in table 2 are investigated by the experiments which will be discussed in this section. The table shows the default values as bold. The noise parameter is the standard deviation of the Gaussian noise which is added to the position

Table 2. The tested parameter values. The values in bold are the default values. The values in italic are the values which have been tested in a second experiment.

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

of the not yet projected features. Random features are added or removed to simulate occlusions. If no features are projected in the current panorama, because they are not visible to the robot for example, the robot tries to use the previous homing vector as it should point to the destination. If the previous vector is not available (i.e. has a zero length), a random vector is used. If no vector can be computed more than five times in a row, the test-run counts as a failure.

To prevent the simulation from running endlessly, a limit on the number of steps (iterations) was added. We experimentally found that 2000 was an appropriate limit. As a last constraint, the robot is allowed to travel at a maximum distance of ten times the ideal distance. We expect the robot to perform worse when the amount of disturbance is higher.

4. Analysis of the results

The experiments discussed in this work were done in a simulated environment in worlds with four, two or only one wall present. For each parameter, each of its values listed in table 2 were tested twenty times for all four worlds thereby using default values for the other parameters. The default value of each parameter is shown in table 2 as bold.

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

Since almost none of the results were normally distributed, we use the Wilcoxon Rank Sum test (also called the Mann-Whitney U test; [10]). In the discussion of the results, $\alpha = 0.05$ will be used. A more extended discussion of the results can be found in [3].

4.1. Noise in the feature position

Adding Gaussian noise to the positions of the features before each projection showed that the robot was able to reach its goal in almost all test-runs within our set limitations, when the noise had a standard deviation of 0.001 or less. Except for 40% of the runs in world 4 in which the test failed. All test-runs failed for a standard deviation of 0.5 and more.

As expected a higher noise rate increased the number of iterations which were used and also the difference with the ideal distance. It is difficult to compare the amount of noise to a situation in the real world. In the simulation we used Gaussian noise, because noise in nature generally is normal. We have found the level at which noise is tolerated by the model.

4.2. Occlusions

Occlusions were simulated by removing randomly chosen features before every projection. Removing until 500 of the 1000 features per world did not have any significant effect on the performance. In world 4 however, the result dropped to a success rate of 50%. When 95% of the features were removed, the success rate for worlds 1 to 4 was respectively : 60%, 35%, 40% and 15%. For 99.0% removed features 15% of runs in world 2 succeeded, 10% in world 3 and none in worlds 1 and 4. For 99.5% removed features none of the runs succeeded. The ALV method assumes an isotropic feature distribution [2]. This assumption can explain why the results in world 4 were worse than in the other worlds. The robot used more iterations when more features were removed, which was not significant for all number of removed features but was to be expected since the ALV every time has a different error. Also the difference with the ideal distance increased with the amount of removed features.

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

4.3. Number of features

Having more (reliable) features present in the world increases the performance of the robot (higher success rate, less iterations and a smaller difference with the ideal distance). The success rate was 100% for 100 and more features for world 1, for 500 and more features for worlds 2 and 3 and 1000 and more features for world 4. The success rate for 20 features per world varies from 80% for world 1 to 50% for world 4. Having more features available increases the amount of information, and therefore it should improve the performance of the robot when it uses the features to navigate.

4.4. Wall length

When we look at the size of the room than we see that almost all runs were successful in square rooms. However in rectangular rooms in which the wall in the x direction was much longer than the wall in the z direction, the robot was less successful. All 20 runs succeeded for all worlds for a wall length of 5 and 10 m. For a wall length of 30 m in the x direction about 60% of the runs succeeded.

The robot performed worse in rooms in which one wall was much larger than the other, this can be explained by the way the features are projected. Feature pairs which are at the same distance on the wall, will be projected closer to each other on the panorama when they are further away than when they are closer to the robot. In [8] the same problem is mentioned: the vergence angle α is small near the antipodal direction and therefore the uncertainty of the depth is high. The result of a small vergence angle is that the projections of the features on both panoramas are very close to each other, therefore the difference between the vectors pointing to the projected features on both panoramas is very small. In our experiments there were about 1000 features equally divided over each present wall To calculate the homing vector, the AL vectors are subtracted, but in the case of the features on the walls in the x-direction, this results in very short vectors. We have measured the length of the homing vectors to verify this. The mean length of the homing vector in a room of 5 m $\times$ 5 m was 0.025 m (and a standard deviation of 0.007 m). However when one wall of the room was 50 m and the other 5 m, it resulted in very small homing vectors of 10^{-6} m. For this reason a lot of runs failed because the maximum number of iterations (2000) had been exceeded.

The number of iterations increases with increasing wall length, but the difference with the ideal distance is not significant in all worlds.

4.5. Rotation of the panorama projection

As expected the method did not work when the panorama was rotated randomly, therefore the panoramas (or at least the ALV's) should be aligned before calculating the homing vector. This can be accomplished by using a compass for example.

4.6. Results per world

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

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

5. Conclusions and Future work

In this work we evaluated the robustness of a homing method which uses the average feature vector which is calculated from data of a panorama. A big advantage of the ALV is its simplicity: the only requirement is that it needs features extracted from the environment. In the simulation the method was always successful in worlds with 1000 features and more.

Observations in the real world however, are almost never the same, therefore we have tested how robust the method is to noise and occlusions. Removing up to 50% of the features present in the world resulted in a mean success rate of 85%. Adding fake features had unexpectedly no significant negative effect on the results. A maximum tolerated noise level has been found but tests have to be done with sensors in the real world to find out the degree of noise tolerance in the real world.

As expected the robot performed best in world 1 which has four walls and therefore satisfies to the isotropic distribution which is one assumption of the method. In the simulation the panoramas were always aligned, but in the real world an alignment method should be used, for example a compass could be used.

In the future simulated tests can be done in a world in which the features are not uniformly distributed over the walls, and where the worlds contain some objects. We will continue doing experiments with this homing method on a real robot. For this experiment we will use a panorama image and extract SIFT features [5].

Acknowledgements

The authors want to thank David Aldavert for the base code for the simulation environment. This work has been partially supported by the FI grant from the Generalitat de Catalunya and the European Social Fund, the MID-CBR project grant TIN2006-15140-C03-01 and FEDER funds and the Marco Polo Fund of the University of Groningen.

References

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

[2] M. O. Franz, B. Schölkopf, H. A. Mallot, , and H. H. Bülthoff. Where did i take the snapshot? scene-based homing by image matching. *Biological Cybernetics*, (79):191–202, 1998.

[3] A. Goldhoorn, A. Ramisa, R. L. de Mántaras, and R. Toledo. Robot homing simulations using the average landmark vector method. Technical Report RR-IIIA-2007-03, IIIA-CSIC, Bellaterra, 2007.

[4] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30(1-2):39–64, 2000.

[5] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[6] R. Möller. Visual homing in analog hardware. *International Journal of Neural Systems*, 1999.

[7] L. Smith, A. Philippides, and P. Husbands. Navigation in large-scale environments using an augmented model of visual homing. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, and D. Parisi, editors, *SAB*, volume 4095 of *Lecture Notes in Computer Science*, pages 251–262, 2006.

[8] A. S. K. Wan, A. M. K. Siu, R. W. H. Lau, and C.-W. Ngo. A robust method for recovering geometric proxy from multiple panoramic images. In *ICIP*, pages 1369–1372, 2004.

[9] R. Wehner. Spatial organization of foraging behavior in individually searching desert ants, Cataglyphis(Sahara Desert) and Ocymyrmex(Namib Desert). *Experientia. Supplementum*, (54):15–42, 1987.

[10] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.