Visual homing: a purely appearance-based approach

Tristan Mitchell Frédéric Labrosse

Department of Computer Science
University of Wales, Aberystwyth
Aberystwyth SY23 3DB, United Kingdom

e-mail: `ffl@aber.ac.uk`

Published in: *Proceedings of Towards Autonomous Robotic Systems*, pages 101–108 (2006), University of Essex, Colchester, United Kingdom, `http://www.taros.org.uk`.

Visual homing: a purely appearance-based approach

Tristan Mitchell **Frédéric Labrosse**
Department of Computer Science
University of Wales
Aberystwyth, Ceredigion
SY23 3DB
ffl@aber.ac.uk

Abstract

This paper presents an algorithm that uses visual input to perform homing for an autonomous mobile robot. An image captured at the target pose (position and orientation) is compared with the currently viewed image to determine the parameters of the next move of the robot toward the target (rotation and translation). The visual data is captured using an omnidirectional camera and images are compared using the Manhattan distance function to determine both the translation and rotation angle. Results, limitations and successes are presented and discussed.

1 Introduction

Homing, or the process of returning to a specified place, is an important task in mobile robotics and many methods have already been proposed. All methods explicitly extract features from the data gathered from the environment, such as analysing chromatic and geometric characteristics (Vassallo et al., 2000, Cassinis et al., 2002, Regini et al., 2002), image motion (Santos-Victor and Sandini, 1997, Crétual and Chaumette, 2001) or salient parts of images (Gaspar et al., 2000). The problem with feature-based approaches is that one needs to know in advance the kind of features the environment will have to be able to extract them. This makes these approaches not robust to environment changes and variability. Appearance-based methods have the advantage that no knowledge about the environment is needed and/or that no environmental changes are necessary to enable a robot to navigate in that environment.

There is now some evidence that insects perform homing only using retino-centric representations (images) of their environment and that they do not perform any feature extraction but rather use the raw images (Judd and Collett, 1998). The homing method proposed in this paper uses an appearance-based approach. Images are compared without performing any explicit feature extraction. This is similar to the work in (Rizzi et al., 1998) but our work presents some notable differences. First, it does not restrict camera motion to forward translation but also uses rotation. Second, we do not attempt to find a single displacement of the robot that would lead to the target. We rather progressively moves toward it, as has been done in the end in (Rizzi et al., 1998) because their algorithm fails to find the right displacement in one iteration. Finally, we use an omnidirectional camera as such a camera contains more information about the environment in a single frame, a property that helps visual navigation and related applications. We compare the performance of our method with the method in (Rizzi et al., 1998) in Section 4.

The use of the term "homing" here indicates the process by which an autonomous mobile robot drives toward a specified location within an environment. It does not necessarily imply that a "docking station" is required; the proposed algorithm performs equally well when homing toward an open space as docking with a physical object. However, it needs a visually salient feature of the environment that will allow the robot to turn toward the target (see Section 4 for a discussion). It is assumed that another process will take the robot in the vicinity of the target.

The algorithm involves comparing an image captured at the target pose (position plus orientation) with the currently perceived image. Homing is performed using only this visual data. Pixel colour values in the images are used to compute the next motion of the robot (a rotation and a translation). Homing is achieved by making a series of moves toward the target, stopping to readjust the course based on the new current image.

Section 2 describes the homing algorithm. Section 3 describes the processing performed to convert raw images into a robot motion. Section 4 presents results of the homing algorithm and discusses limitations of the method. Finally, Section 5 concludes and proposes future enhancements.

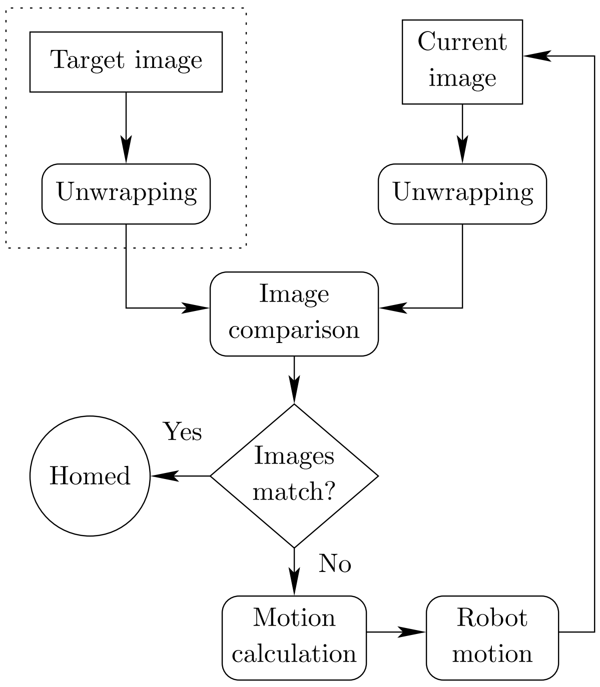


Figure 1: Flow chart of the homing algorithm

2 The homing algorithm

Figure 1 shows the steps taken in performing the homing. Our algorithm is inspired by the process used in (Cassinis et al., 2002).

A target image is specified at the beginning of the process. The section shown in the dotted line is the processing of the target image that is captured at the location to which homing should occur. This is performed only once for each target image and is performed separately to the homing process.

At its current location, the robot grabs a new image, the current image, and processes it in the same way as the target image at the beginning, Section 3.1. The processing of the images takes place on-line, immediately after capture.

The processed target and current images are compared. If the two images match, then homing occurred. Otherwise, a motion for the robot is computed based on the difference between the images, Section 3.2, the robot moves following instructions and the process is repeated.

3 Image processing

In this section, we describe the processing performed on the images, both target and current, needed for the homing algorithm, Section 2. We also describe how images are compared and how the difference between the images is converted into robot motion.

3.1 *Image unwrapping*

Image capture was performed using an omnidirectional camera and a real-time frame grabber. The image grabber was implemented as a server running on the robot that transmitted images to a client image receiver using sockets.

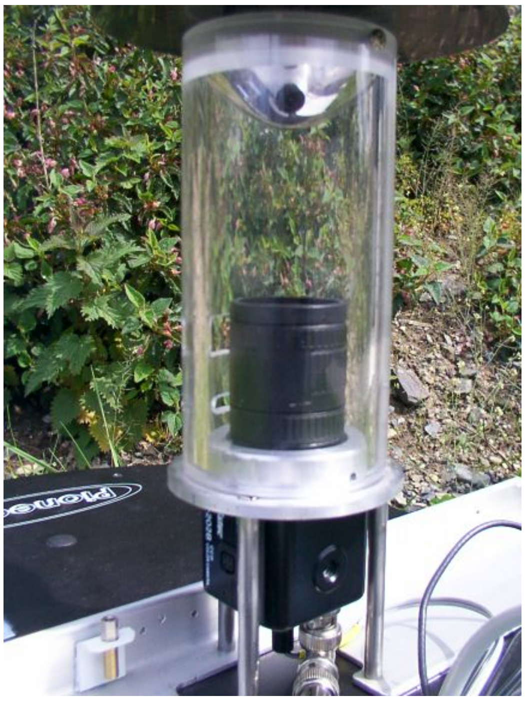
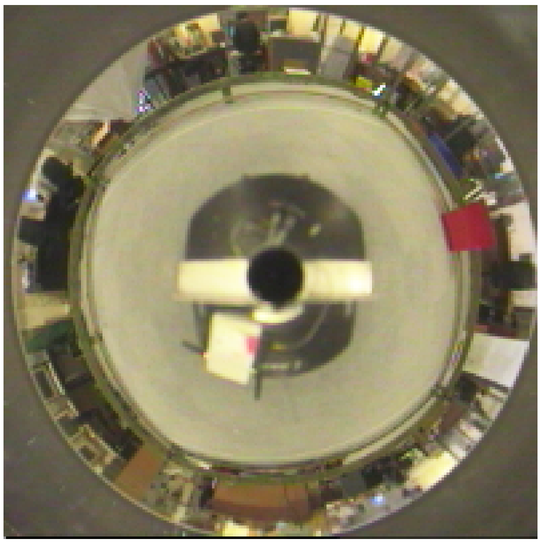


Figure 2: The omnidirectional camera

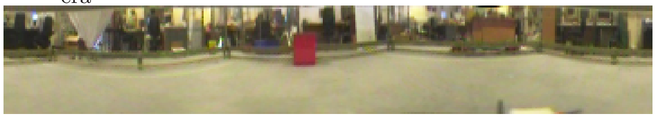
The omnidirectional camera is made of a “normal” camera pointed upward and looking at a hyperbolic mirror, thus providing an omnidirectional view of the robot’s surroundings. Figure 2 shows the camera, bottom of the perspex tube, and the mirror, top.

An image captured by the omnidirectional camera appears as a wrapped omnidirectional image, as can be seen in Figure 3(a). To make the process of extracting the robot motion parameters easier, Section 3.3, these circular images were transformed into unwrapped panoramic images, Figure 3(b). The unwrapping is performed by scanning a line emanating from the centre of the omnidirectional image and rotating around the image by increments of 1°. Pixels of the panoramic image are taken along the line using the nearest pixel of the omnidirectional image[1], Figure 4. The omnidirectional image size was 200×200 pixels and the panoramic image size was of 360×60 pixels, one column per degree of angle and one row per usable distance away from the centre in the omnidirectional images.

[1]We have also used bi-linear interpolation, but this is more expensive to compute and did not change the performance of the system.



(a) An image captured using the omnidirectional camera



(b) The corresponding unwrapped image

Figure 3: Typical images of the test environment

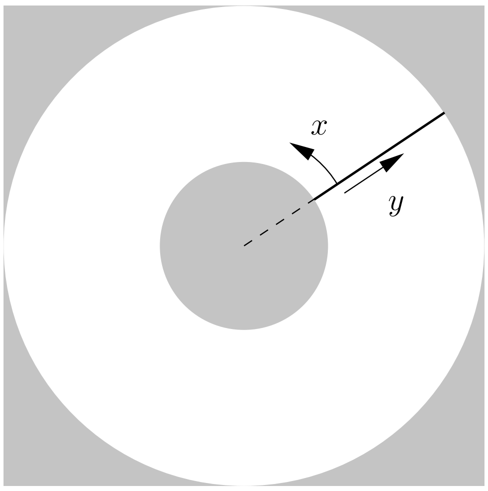


Figure 4: The unwrapping process. The grey areas correspond to unusable parts of the image (projection of the robot or outside of the mirror). x and y are the coordinates of pixels in the panoramic image.

This conversion does not produce a completely accurate representation of the environment, as the geometry of the mirror used in the omnidirectional camera was not taken into account in the above transformation and no calibration of the system has been done. In particular, the perspex tube used to link the camera and the mirror is not perfect and introduces many local distortions. Moreover, the process assumes that the optical axis of the camera projects at the centre of the image, which is very probably wrong. However, as long as the distortions do not change with time, calibration was not needed as we only use differences between images, Section 3.2.

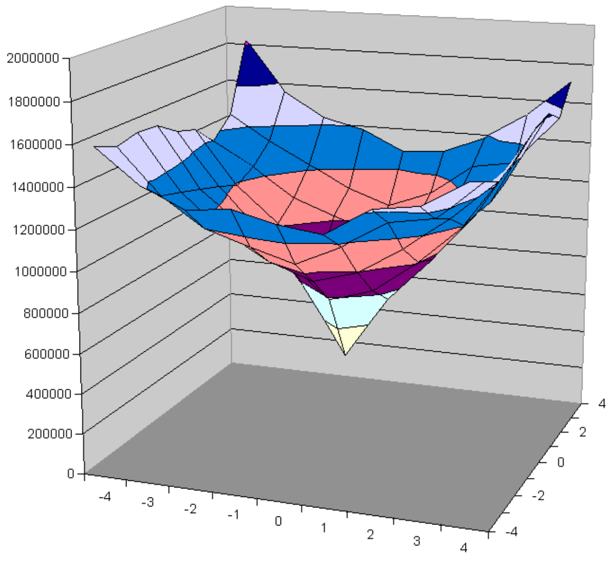


Figure 5: Manhattan distances between a target image and images taken from positions around the target position

3.2 Image comparison

The image comparison technique used relies on the fact that an image can be mapped as a single point in a multi-dimensional space, the *image space*. The image space has one dimension per pixel per colour coordinate. For example, a 360×60 pixels image using 3 colours per pixel can be plotted as a point in $64,800$ $(360 \times 60 \times 3)$ dimensional space. This allows us to measure the distance between images as a way of comparing them.

The distance function chosen for the comparison was the Manhattan distance:

$$d(\mathcal{C}, \mathcal{T}) = \sum_i \sum_j \sum_k |\mathcal{C}_{(i,j)k} - \mathcal{T}_{(i,j)k}|, \qquad (1)$$

where $d(\mathcal{C}, \mathcal{T})$ is the distance between the current image and the target image, and $(i, j)k$ denotes the k^{th} colour component for pixel (i, j) (we have used R, G and B from the RGB colour space and a* and b* from the CIE L*a*b* colour space, Section 3.4).

To test the behaviour of the Manhattan distance function, a grid was marked out on the floor of the environment used for homing. Each grid square is approximately 50cm by 50cm. A target image was captured at grid position $(0, 0)$ (in Figure 5, this is in the centre of the grid, in Figure 7 this is at the centre of the near side) and images were also captured at each grid position. Each image was then compared against the target image and the Manhattan distance plotted. Figure 5 shows the result. As can be seen, there is a clear mini-

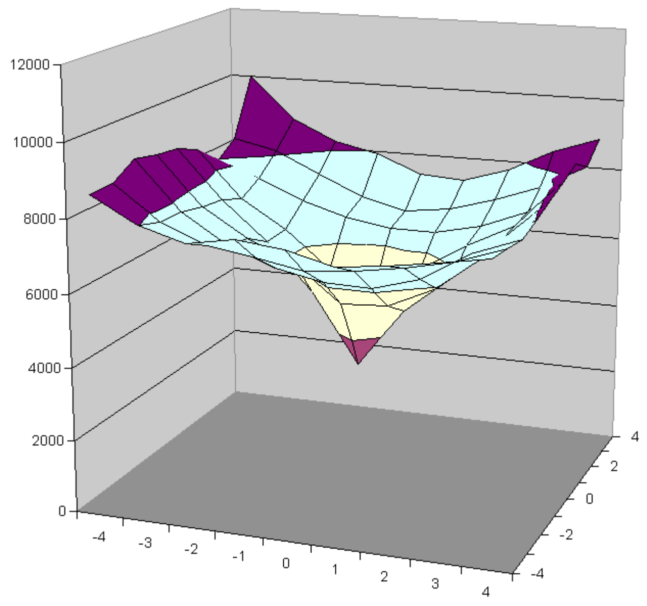


Figure 6: Euclidean distances between a target image and images taken from positions around the target position

mum in the function at the target position as well as a well defined "bowl" around the minimum. This indicates that the Manhattan distance in image space is indeed a good measure of how close the robot is from its target, at least in the range of spatial distances concerned in the homing process.

We have also tried the Euclidean distance to measure differences between images in the image space. Figure 6 shows that qualitatively, the function is very similar. However, perhaps surprisingly, the graph is flatter when away from the target position, which, we expect, does not help the convergence. Because of this and because computing Euclidean distances is more expensive than computing Manhattan distances, we decided to use the latter.

3.3 Mapping distances in the image space to robot motion

Image comparison is used to calculate a rotation angle for the robot as well as a displacement after the rotation. To achieve this, the current image is shifted by one pixel at a time to the right and the Manhattan distance is calculated for each shift (Section 3.2). As the unwrapped images are 360 pixels wide, a one-pixel shift corresponds to a rotation of $1°$. The best angle of rotation for the robot is given by the angle with the lowest Manhattan distance value after comparison with the target image. In early experiments, this angle was used to rotate the robot toward the target.

Experiments quickly showed that using this angle tended to make the robot home with a wrong orientation as it was heading straight at the target, ending wrongly aligned with it, instead of having a "round" path that would lead it in front of the target. Instead of using the best un-rotation angle, we used some proportion of it, in other words under-steering, 50% in all the experiments reported. This has some implications, discussed in Section 4.

Following the rotation, the robot moves forward[2] by an amount dependent on the distance between the unrotated current image and the target image. The length of the step is proportional to the difference between the distance and a set point empirically determined. As long as the distance between current image and target image decreases, the robot moves forward. If the distance increases, then the robot reverses as this corresponds to the robot over-shooting the target.

When the distance in image space between the current image and the target image becomes lower than some set threshold, the robot decides that it has docked with the target.

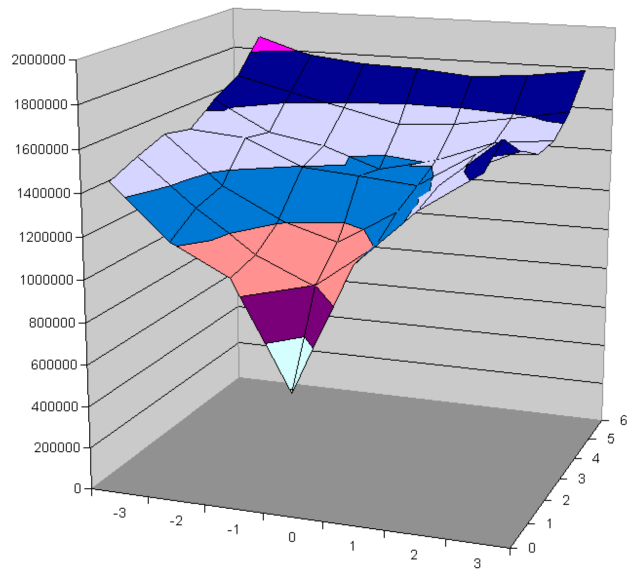
3.4 Colour space considerations

An important consideration is the colour space used for the distance computation. As can be seen from the image on Figure 3, the environment in which the experiments were performed is a large (4m by 4m) area free from obstacles delimited by a low fence in the middle of a cluttered area. This translates into images made of a large homogeneously grey area (bottom of the unwrapped images) and an almost random multi-coloured area (top of the unwrapped image). A "docking station" was materialised by a red box. Note that the red box is not explicitly extracted by the algorithm, it was only used to introduce some features in the random background of the room.

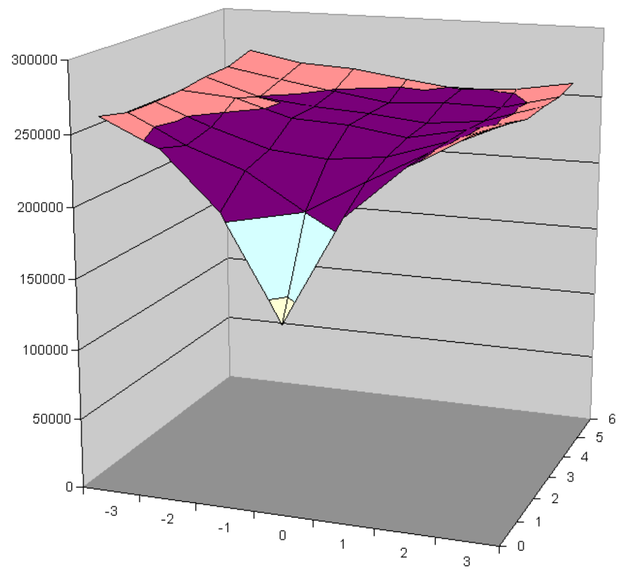
To improve the differences between colours, we decided to use the CIE $L^*a^*b^*$ colour space because of its perceptual linearity (Fairchild, 1998). Colours in this colour space are represented using 3 coordinates. The coordinate L^* represents the luminance, the coordinate a^* represents colour between green and red and the coordinate b^* represents colour between blue and yellow. The chrominance information is thus only conveyed by the two coordinates a^* and b^*. To make the system more robust to illumination changes, we do not use the coordinate L^*.

We performed experiments as in Section 3.2 to compare the distance function in the RGB and CIE $L^*a^*b^*$ colour spaces. Figure 7 show the graphs of the two functions with the same images. This result seems disappointing as the Manhattan function seems to behave the same way in the two colour spaces.

[2]This actually happens at the same time as the robot is rotating. However, the rotation is faster than the translation, which thus mostly happens after the rotation.

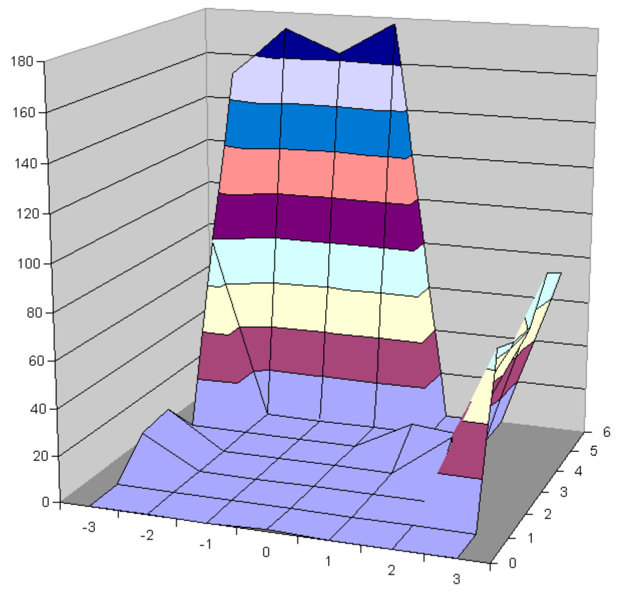


(a) RGB

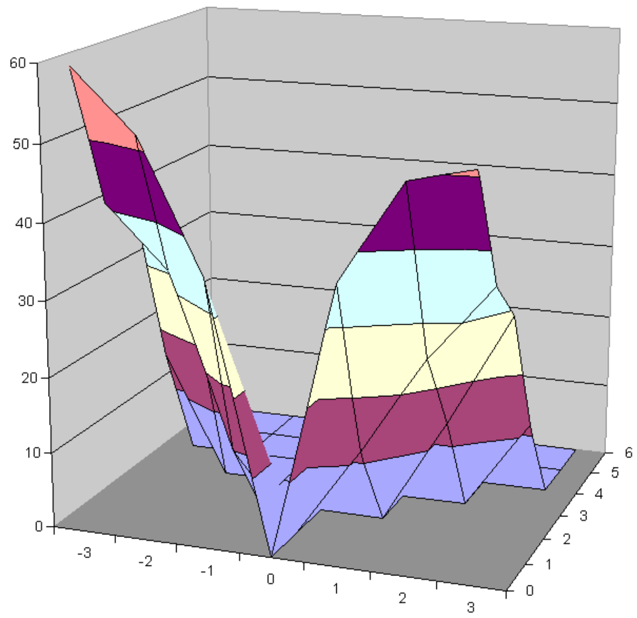


(b) CIE L*a*b*

Figure 7: Manhattan distances in two colour spaces



(a) RGB



(b) CIE L*a*b*

Figure 8: Rotation angles in two colour spaces

However, investigating the effect of the colour space on the rotation angle extracted from the images to rotate the robot, Section 3.3, was more conclusive. Using the same images as for Figure 7, we computed the angle by which the robot needs to be rotated to face the target. The absolute value of the angle is plotted on Figure 8. The graphs show that, apart from noise when far away from the target, the robot does not rotate when computing the distances in the RGB space. However, distances computed in the CIE L*a*b* colour space more accurately separate colours and provide the right rotation angles (no rotation when far away from the target and increasing angle when off the front of the target). We however believe this is an artifact of the presence of the red box near the target position. We discuss this further in Section 5.

We thus decided to use the CIE L*a*b* colour space as the distance function produced better rotation angles.

4 Results

We tested our system in our research laboratory, a typical image of which is shown on Figure 3. To assess the performance of the algorithm, we ran several trials from different initial positions with different (random) initial orientations. The robot was tracked using the Vicon mo-

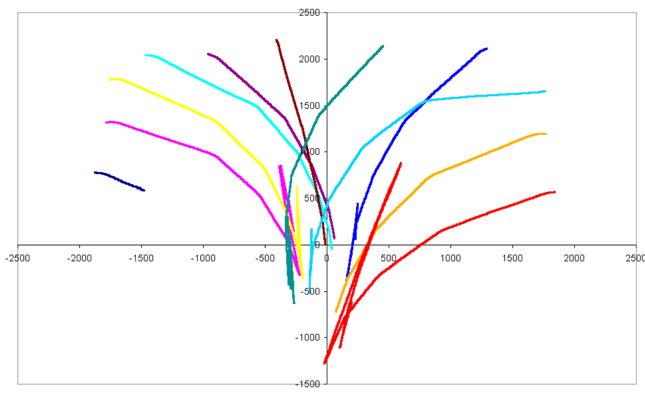


Figure 9: Paths taken by the robot as tracked using the Vicon system

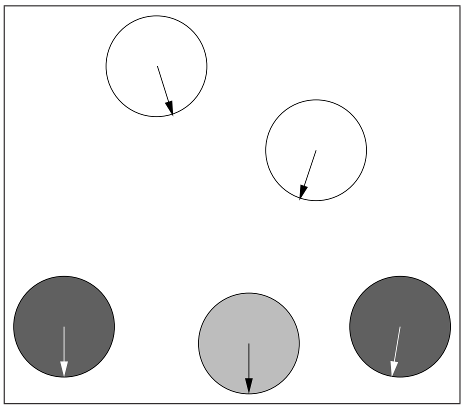


Figure 10: Different cases. The grey circle corresponds to the target, the white circles correspond to places from where homing will be successful and the black circles correspond to places from where the homing will not be as successful.

tion tracking system. The grabbed paths are shown on Figure 9 as lines, only representing the position of the robot (the orientation can roughly be inferred by the change in position).

As can be seen, all but one paths end within ±50cm around the target. The one path that does not was actually successful but not tracked because the reflective marker used by the Vicon system was momentarily hidden by somebody moving in the room.

In one case, the right-most path on Figure 9, the robot did over-shoot its target by about 1m and did oscillate only making very slow progress[3]. In this case however, the robot started from a point sideways to the target, configuration for which the un-rotation of the images provides a not too good orientation for the robot. This shows a fundamental limitation of the algorithm: the robot cannot be too far off on either side of the target for it to successfully home onto it, Figure 10. As long as the robot is sufficiently in front of the target (white circles on Figure 10), then the alignment provided by

[3]We thus stopped it before it had reached convergence.

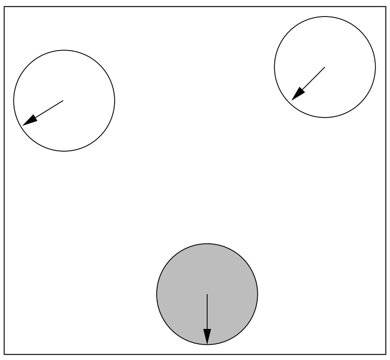


Figure 11: Good and bad position/orientation relative to the target (grey circle). The white circle on the right corresponds to a right position/orientation while the circle on the left corresponds to a wrong position/orientation.

the un-rotation of the current image to match the target image will be such that, with under-steering, the robot will successfully home onto the target. This is because the red box materialising the "docking station" creates a strong feature in the images, especially when working in the CIE L*a*b* colour space. We remind the reader that the features are however not explicitly extracted. This is why the angle as displayed on Figure 8(b) are better. In the RGB colour space, Figure 8(a), the red box does not stand out as strongly when compared to the other parts of the images and the best un-rotation of the images then gives an angle that will align the robot with the target orientation, preventing the robot from going toward the target[4].

Another influencing factor is the under-steering, Section 3.3. Two problems occur. The first is again explained on Figure 10. When starting from the position corresponding to the black circles, under-steering prevents the robot from reaching its target, as can clearly be seen on Figure 9. Figure 11 is used to explain the second problem. It shows two positions/orientations relative to the target that are good or bad as a starting point for the algorithm, because of the under-steering. Under-steering will make sure that the good starting configuration will succeed while making the other starting configuration even worse. Note that some over-steering would actually make the wrong starting position better (and would also solve the oscillations exhibited on the right-most path of Figure 9). However, there is no obvious way of deciding when to over-steer and when to under-steer if we do not assume anything about the current pose of the robot with respect to its target. We decided on under-steering as usually, in homing situations, the robot is more or less correctly aligned at the beginning of the process.

As stated in Section 1, the proposed method

[4]The visual compass described in (Labrosse, 2004) uses that same property to extract the heading of the robot. In that case, having the correct alignment is a desirable feature.

bears some resemblance with the method proposed in (Rizzi et al., 1998) but has some important differences. We compare here the performance of our method with the published results (Rizzi et al., 1998, Fig. 7). All our trials arrived within about ±50cm of the target, similar to those of (Rizzi et al., 1998). However, we have tried from starting positions sideways to the target, unlike (Rizzi et al., 1998) who only report results starting from positions in front of the target, admittedly further away compared to our trials. We haven't tried such far away starting positions but we do not expect that they would pose any problem, as long as the images are visually not too different from the target. Our results are thus comparable to their's with a much simpler matching algorithm and without constraints such as constant height and no rotation.

5 Conclusion, discussion and future work

The visual homing method that has been presented implements an algorithm to allow a mobile robot equipped with an omnidirectional camera to navigate toward a target position within its environment. The method is purely appearance-based; in particular, it does not extract features from the images but only measures the difference between images. Tests have shown that while this method has been successful in driving the robot toward the target, precise positioning was not always achieved and thus the method may not be suitable for accurate docking procedures. Moreover, although we do get good results from a wide range of starting positions/orientations relative to the target, we have shown that there are some limitations in terms of initial configuration. Moreover, although no explicit feature extraction is performed, the algorithm does need a visually standing out feature in the environment to materialise the "docking" station. We have characterised what constitute good starting positions/orientation to ensure success of the algorithm.

The use of the Manhattan distance function in the image space to calculate a distance and rotation angle for the robot was successful. Graphs presented show that when in the CIE L*a*b* colour space, useful output was obtained.

It is to be noted that all the experiments were taking place in a research lab with people walking about during the experiments. Moreover, we also tried, although we did not record any formal results, to modify the environment by introducing objects in the free area in which the robot was moving, after the target image was grabbed. The robot still successfully homed.

One of the remaining problems that need to be addressed is when to decide that the robot has docked. At the moment, Section 3.3, the robot stops when the distance in image space between the current image and the target image is below a set threshold. This threshold however depends on the environment and the lighting conditions. The latter can be partially solved by not using the luminance information (the L* component in the CIE L*a*b* colour space), which we did. The former could be solved by looking at the derivative of the distance function. However, as can be seen on Figure 5, the minimum of the function is very sharp, making the computation of the derivative unreliable. Further investigations are needed.

The method provides a good basis from which to continue research on this type of visual navigation. In particular, we want to investigate better ways of computing the turning angle of the robot at each step of the algorithm. This could for example use the gradient of the distance function, Figure 5, around the current position. Another approach could be to control the speed (rotation and translation) of the robot instead of relative moves.

We also want to apply this work to navigation in an environment represented using a topological map by homing in on a succession of targets selected from the map (Neal and Labrosse, 2004). In such a situation, most problems we noted in this work become irrelevant as "homing" does not need to be accurate when navigating from place to place.

References

Cassinis, R., Duina, D., Inelli, S., and Rizzi, A. (2002). Unsupervised matching of visual landmarks for robotic homing using fourier-mellin transform. *Robotics and Autonomous Systems*, 40:131–138.

Crétual, A. and Chaumette, F. (2001). Visual servoing based on image motion. *International Journal of Robotics Research*, 20(11):857–877.

Fairchild, M. D. (1998). *Color Appearance Models*. Addison-Wesley.

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

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

Labrosse, F. (2004). Visual compass. In *Proceedings of Towards Autonomous Robotic Systems*, pages 85–92, University of Essex, Colchester, UK.

Neal, M. and Labrosse, F. (2004). Rotation-invariant appearance based maps for robot navigation using an artificial immune network algorithm. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 863–870, Portland, Oregon, USA.

Regini, L., Tascini, G., and Zingaretti, P. (2002). Appearance-based robot navigation. In *Proceedings of the Workshop su agenti robotici, Associazione Italiana per l'Intelligenze Artificiale, VIII Convegno*, Siena, Italy.

Rizzi, A., Bianco, G., and Cassinis, R. (1998). A bee-inspired visual homing using color images. *Robotics and Autonomous Systems*, 25:159–164.

Santos-Victor, J. and Sandini, G. (1997). Visual behaviors for docking. *Computer Vision and Image Understanding*, 67(3):223–238.

Vassallo, R. F., Schneebeli, H. J., and Santos-Victor, J. (2000). Visual servoing and appearance for navigation. *Robotics and Autonomous Systems*, 31(1–2).