Image-Based Monte-Carlo Localisation with Omnidirectional Images

Emanuele Menegatti, Mauro Zoccarato, Enrico Pagello†, Hiroshi Ishiguro ‡

Intelligent Autonomous Systems Laboratory
Department of Information Engineering (DEI)
Faculty of Engineering, The University of Padua
Via Gradenigo 6/a, I-35131 Padova, Italy
†also with: Institute ISIB of CNR, Padua, Italy
‡Department od Adaptive Machine Systems
Osaka University, Suita, Osaka, 565-0871 Japan

Abstract

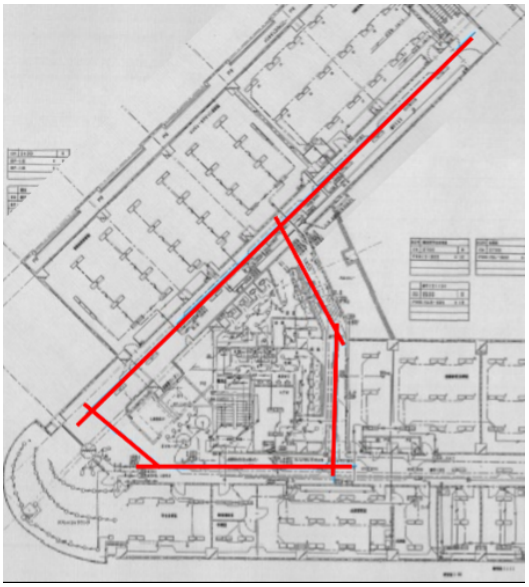
Monte Carlo localisation generally requires a metrical map of the environment to calculate a robots position from the posterior probability density of a set of weighted samples. Image-based localisation, which matches a robots current view of the environment with reference views, fails in environments with perceptual aliasing. The method we present in this paper is experimentally demonstrated to overcome these disadvantages in a large indoor environment by combining Monte Carlo and image-based localisation. It exploits the properties of the Fourier transform of omnidirectional images, while weighting the samples according to the similarity among images. We also introduce a novel strategy for solving the *"kidnapped robot problem*".

Key words: omnidirectional vision, image-based navigation, Fourier transform, Fourier signature, Monte-Carlo localisation

1 Introduction

In mobile robotics the localisation problem is fundamental. In several successful experiments, the robot is provided with a geometrical map of the environment, and it uses some kind of sensors to locate itself in this map [20; 23]. However, it is not always possible (or convenient) to build a geometrical map of the environment. In the **image-based localisation** approach

(a)



(b)

Fig. 1. (a) A picture of the robot used in the experiments. (b) The plan of the building where the omnidirectional image data set was built. This is a large environment in which the length of the longer corridor is 50 metres. The thick line represents the robot's path. Details of the map are unimportant; it is presented just to convey the complexity of the environment.

[1; 4; 11; 13; 19; 8], a map is not used and the agent uses a set of *views*" previously taken at different locations to locate its-self. These locations are called **reference locations**. The corresponding images are called **reference images**. When the robot moves, it can compare the current view with the reference images stored in its visual memory. In the image based localisation, the problem of finding the position of the robot in the environment is reduced to the problem of finding the best match for the current image among the reference images. Most of the cited works on image-based localisation just stop here. In this work, we enhance image based localisation with a statistical approach in a way similar to the work reported in [22] and in [9].

Classical image-based localisation does not work in environments with periodical structures (for instance, a corridor with a set of doors equally spaced or a set of junctions with the same appearance). In these environments, the appearance of the world is the same at different places, so the current view will match not only the corresponding reference image, but also all reference images similar to the current image. This is a case of **perceptual aliasing**, that is, the reading of the sensor is the same at different locations. So, the vision sensor is not able to discriminate these locations. A robot could use additional sensors to discriminate between two points, for example, a GPS sensor or other non-vision sensors. But what if these additional sensors are not available? The robot needs to manage situations, maybe transitory situations, in which it has evidence of being located at two distinct points at the

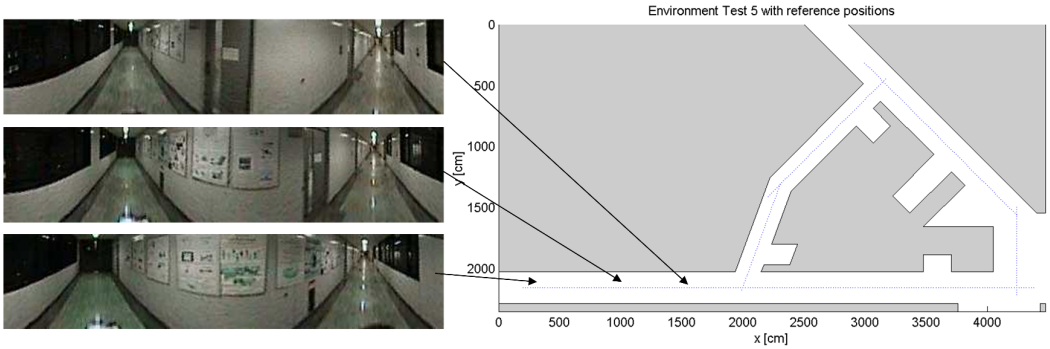


Fig. 2. Some panoramic images taken at different points in the environment. These images show that the environment appears very similar even at locations more than 5 metres away one from the other. The arrows show the actual position in which every image was grabbed.

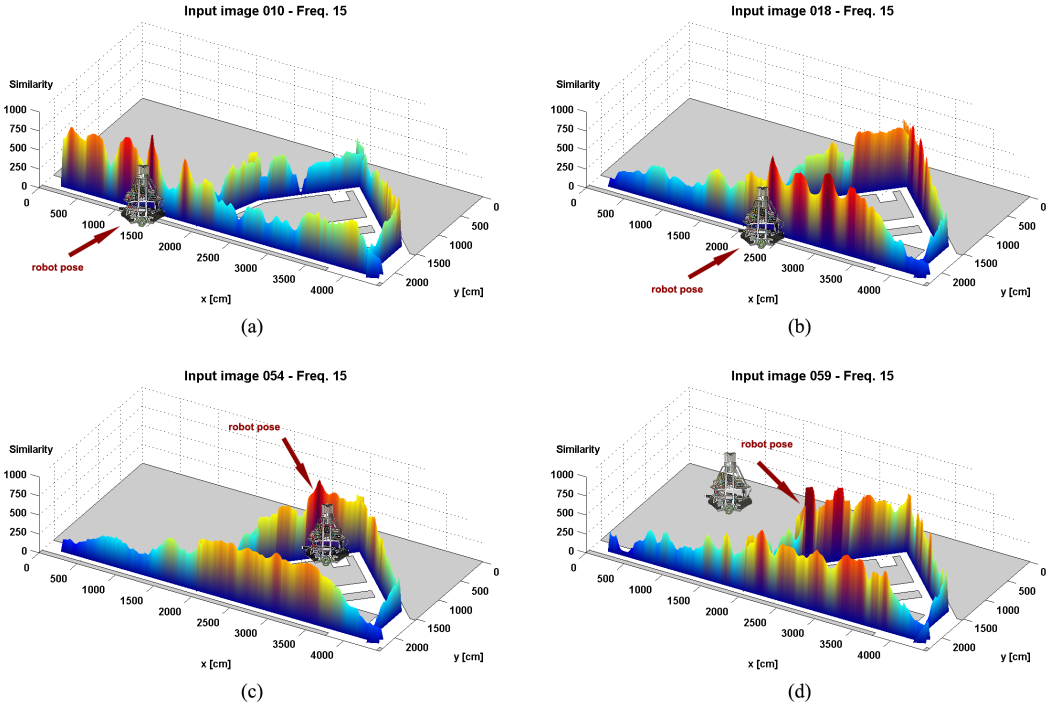


Fig. 3. Some examples of perceptual aliasing in our test environment. The height of the picks is proportional to the amount of similarity of the different images. (a) Input Image 10 matches not only the correct reference image at 1380 centimetres on the x-axis, but also reference images at 320, 620, and 1060 centimetres. The same happens in plots (b), (c), and (d).

same time. The solution we adopted here is to use a Monte-Carlo localisation process to manage the uncertainty about its position [6; 21]. This technique is able to manage a multi-modal probability density, thus, allowing the robot to estimate its position correctly when the current image matches more than one reference image.

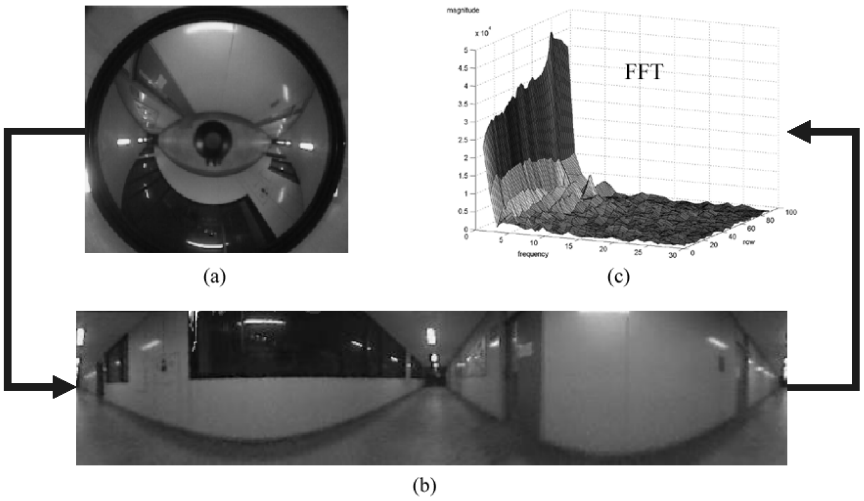


Fig. 4. The process of generation of the Fourier Signature. The omnidirectional image (top left) is unwarped in a panoramic cylinder (bottom) and the Fourier coefficient of every row of the panoramic cylinder are calculated (top right).

The environment in which we tested our system presents severe perceptual aliasing. We show in Fig. 2 some sample images from the reference image dataset to show that the environment looks the same also at locations more than five metres away from one another.

To have a general feeling of how severe the perceptual aliasing is in this environment, one might consider Fig. 3, in which the similarity [1] of the images grabbed by the robot in different positions is plotted against all reference images (in Fig. 3(a) the input image is number 10, in Fig. 3(b) it is number 18, in Fig. 3(c) it is number 54, in Fig. 3(d) it is number 59). These plots show that the similarity of the current input image is high (higher peaks in the plots) not only for the correct reference image, but also for other reference images in locations far away from the actual position of the robot.

2 Image Matching

In the image based localisation approach, the main problems are how to store in a memory-saving way the reference images and how to efficiently compare all of them with the current input image, considering that for a wide environment the number of reference images can be very large (e.g. in our 50×25 metres environment, we have about 500 reference images).

[1] The similarity between the current image and a reference image is calculated with Eq. 1.

Image	Memory Required (in bit)	Memory Required
omnidirectional	$640 \times 480 \times 24$	7.3 Mbits
panoramic cylinder	$512 \times 80 \times 8$	328 Kbits
Fourier signature	$80 \times 15 \times 2 \times 8$	19 Kbits

Table 1
The different memory requirements illustrating the memory savings introduced by the Fourier signature.

In this paper we have fully developed a method we proposed in previous works [15; 17; 18]. The robot is equipped with an omnidirectional camera [10; 16]. At the setup stage, the robot takes a set of omnidirectional images at reference locations. At the running stage, while it moves in the environment, the robot compares the current omnidirectional image with the reference images. In order to store and match efficiently a large number of images, we transform each omnidirectional image into a compact representation. Fig. 4 sketches the steps of this transformation. From a colour omnidirectional image (Fig. 4(a)) we create a gray-scale **panoramic cylinder**, that is a new image obtained by unwarping the original omnidirectional image, Fig. 4(b). The panoramic cylinder is expanded row by row into its Fourier series. The robot memorises each view by storing the Fourier coefficients of the low frequency components, Fig. 4(c). We called the set of the stored coefficients the **Fourier signature** of the omnidirectional image. We showed in [15] that the first 15 Fourier components are a compact representation for the omnidirectional images that enable to effectively assess the similarity of the input images with respect to the reference images. The Fourier signature drastically reduces the amount of memory required to store a reference image, as reported in Table 1. For a typical omnidirectional image of 640×480 pixels, we build a 512×80 pixels panoramic image, which Fourier coefficients can be stored in a $15 \times 80 \times 2$ array (i.e. 15 Fourier coefficients, 80 rows, 2 for the phase and the magnitude of the Fourier Transform). So, compared to other techniques that stores the original omnidirectional colour image, the memory requirements decreases from 7.3 Mbits to 19 Kbits, as shown in [17].

With this approach matching the current view against the visual memory is also computationally inexpensive. In fact, we defined a simple dissimilarity function that takes as input the Fourier signatures of two images and gives a value proportional to the amount of "dissimilarity" of the two images. In other words, the less similar the two images are, the higher the value of the dissimilarity function. In Eq. 1 we defined the dissimilarity function as the $L1$ norm of the Fourier signatures of the two images O_i and O_j: here k indicates the frequency and y is the index of the row of the panoramic cylinder. For

more details on this procedure, please refer to [15].

$$Dis(O_i, O_j) = \sum_{y=0}^{l-1} \sum_{k=0}^{m-1} |F_{iy}(k) - F_{jy}(k)| \quad (1)$$

To asses the computational burden of our system, we performed the off-line experiments on a 1400 MHz AMD Athlon with 512 MB of RAM. On this machine, once grabbed the omnidirectional image, to unwarp it, to calculate the Fourier signature, and to calculate the similarity with all the reference images takes less than 30 ms.

2.1 Related Works

Let us highlight the advantages of our approach with respect to other ways of storing and comparing the images. First, using an omnidirectional camera reduces the number of images required to fully describe the environment. In fact, if a perspective camera is used, the view of the environment from a certain location changes with the direction of gaze. A solution can be to constrain the movements of the robot in order to have the perspective camera always pointing in the same direction [4], but this strongly limits the motion of the robot. An alternative can be to extract from the perspective images some features that reduce the amount of required memory while retaining a rich description of the image. A good example of this is reported in [22], where 936 images were stored in less than 4MB. Nonetheless, collecting such a large number of images is tedious and time consuming.

Using omnidirectional cameras, we have only one omnidirectional image for every reference location. The omnidirectional image contains the appearance of the environment in all possible gazing directions. Several authors, exploited omnidirectional images for image based localisation. The most common approach is to extract a set of eigenimages from the set of reference images and project the reference images and the current image into eigenspaces. Unfortunately, this approach does not lead to rotationally invariant images. Usually, the images are preprocessed in order to obtain the rotational invariance as in [1; 12; 8] or the heading of the sensor is constrained as in [13]. On the contrary, our technique, which uses the Fourier transform of the panoramic cylinder (i.e. the Fourier signature), is a natural representation for implementing rotational invariance, as detailed in [15].

3 Monte-Carlo Localisation

As we stated in the introduction, the image-based localisation approach is misleading in situation in which the appearance of the environment is the same at two different locations. In this work, we overcame this problem by exploiting a well-known probabilistic approach in order to estimate the correct position of the robot. The general method, known as **Bayesian filtering** (also known as Markov localisation in robotics) [2; 3; 6; 7; 21], recursively updates the probability density of the robot's position (the **belief**) using motion and perception information. In the Monte-Carlo method one represents the posterior probability density of the robot's pose with a set of discrete points in the configuration space of the robot. To better understand the rest of the paper, we need to briefly review the theory behind the Monte-Carlo Localisation approach (MCL). In the Bayesian Filtering problem one has to calculate the probability density of the robot's position $Bel(s_t) = p(s_t|O_t, a_t)$ over time. What is known is the **prior probability density** $Bel(s_0)$, which describes the initial robot uncertainty about its position; the **prediction model** $p(s_t|s_{t-1}, a_{t-1})$ that applies motion data a_{t-1} to actual state s_{t-1} obtaining a new state s_t; the **observation model** $p(O_t|s_t)$ that represents the probability of making observation O_t from state s_t.

Using the Bayes Formula and Markov assumption about the state space the equation to calculate the belief is

$$Bel(s_t) = \eta p(O_t|s_t) \int p(s_t|s_{t-1}, a_{t-1}) Bel(s_{t-1}) ds_{t-1} \tag{2}$$

where η is a normalisation factor. In Monte-Carlo localisation, the posterior probability density of the robot's position (i.e. the belief) is approximated with a set of weighted samples $\{s_t^i, \omega_t^i\}_{i=1...N}$. The weight associated with every sample is proportional to the likelihood that the robot is occupying that position. The samples are updated recursively with a procedure called **sampling-importance-resampling** [3]. During resampling the samples are drawn with probability proportional to their weights. As a result, unlikely samples die out, while samples with high weights are replicated.

In our implementation, the motion and perception data used to update the samples come respectively from the odometry and from the omnidirectional camera. The weight w_i associated with every sample s_i is proportional to the similarity of the reference images closer to that sample with respect to the current input image, as shown in Eq. 3

$$w_i = \frac{1}{|C|} \sum_{g_j \in C} S_j \cdot (D - dist(s_i, g_j)) \quad , \tag{3}$$

where C is the set of the reference positions g_j at a distance less than D from the sample s_i, and S_j is the similarity value associated with the reference image at position g_j.

3.1 Related Works

The approach we used was inspired by the work of Wolf et al. [22] and turned out to be very similar to the concurrent work of Gross et al. [9], which we were unaware of. One might think these approaches are the same of the one presented by Dellaert et al. in [5]. Actually, they are different in that Dellaert et al., instead of using single reference images, constructed a complex global visual map of the environment by mosaicing all the images of the ceiling.

The main differences between our system and the work of Wolf et al. are *i)* they used a perspective camera while we used an omnidirectional camera (reducing the number of images to be stored); *ii)* they associated a *visibility region* with every reference image in the memory database to reduce the search space for image matching. The use of a visibility region implies the system needs a metrical map of the environment and increases the complexity of the algorithm. We do not need to use the visibility region because our omnidirectional camera supplies a 360° view of the environment; this also means that we rely only on the similarity computation to distinguish various localisation hypothesis with a gain in minor complexity of the algorithm.

The differences with Gross's et al. work are mainly in the way they calculate the image similarity. Basically, for every omnidirectional image they extract a feature vector from the colour panoramic cylinder. This feature vector is calculated dividing the colour panorama into 10 sections and taking the average RGB values of each of these sections. They defined a image similarity function that takes as arguments these features vectors and that implements a rotational invariance. The colour information available in our test environment is very poor, the predominant colours are gray and white, so a method based only on colours to match the images would not work. Capturing the brightness patterns in the environment as provided by the Fourier signature proved to be very effective.

Concerning, the Monte-Carlo localisation methods used in [22] and in [9], they are very similar to the one we used. The main difference is in the way the three works approach the kidnapped robot problem. In the next section, we present our original contribution to the kidnapped robot problem [17].

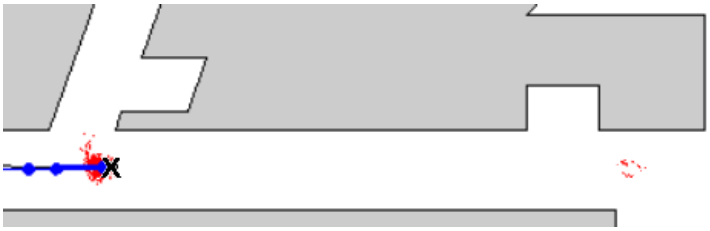


Fig. 5. An example of some samples being generated close to two different reference images matching the current input image. On the left the cluster corresponding to the correct reference image and on the right some samples corresponding to a reference image very similar to the current input image.

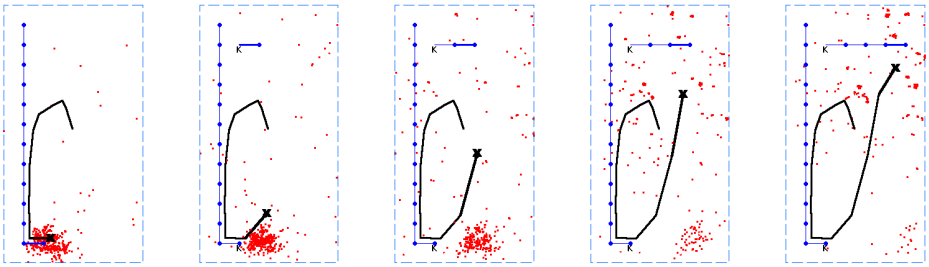
3.2 The proposed kidnapping strategy

In the kidnapped robot problem, the robot is lifted and moved to a different location, so it does not have any odometric feedback on this motion. This problem aims to test the system in situations in which the odometric information of the robot is totally wrong or the robot has to recover from an incorrect localisation.

In the Monte-Carlo localisation methods when the robot has a good localisation, the samples are generated only in a tight cloud close to the estimated position of the robot. This makes it difficult to solve the kidnapped robot problem. In fact, if the robot is moved to a new position without perceiving this motion (kidnapping), it will continue to generate the samples around the previous position, without generating any samples around the new unknown position. If the robot does not have any samples around the new position, it will never recover from the localisation error. The standard approach replaces a certain number of samples with others randomly drawn in the entire environment [7]. This approach relies on the hope that some samples of this subset will be generated by chance close to the position where the robot has been moved. In this case, these samples will have large weights in the next measurements and will survive and cluster the other samples close to them. The drawback is that it takes several iterations for the samples to cluster around the correct position and sometimes it might takes some iterations even to generate a sample close to the position where the robot was kidnapped. This was the approach used also by Wolf et al. in [22]. Gross et al. used an alternative approach, instead of generating a certain amount of samples at random position, they have a certain number of samples generated at fixed important positions in the environment (about the 3% of the total number of samples) [9].

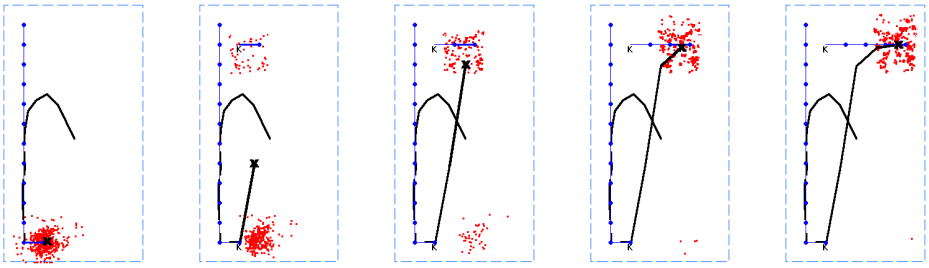
We propose a new solution: instead of randomly drawing the sample, the new samples should be generated only around the reference images that best match the current input image. Each time the samples are generated a number of

samples (10% of samples) are replaced with new samples drawn around the positions of the reference images most similar to the current input image. This assures that the newly generated samples are concentrated around possible locations only. This approach is made possible by the technique we use to match the input image with the reference images. Usually, all reference images matching the input image are close to each other, so the new samples concentrate in a region around the correct reference image. In case of kidnapping or perceptual aliasing, the reference images that are similar to the input image can be far away one from the other and so the new samples are generated at two different positions in the environment, like in Fig. 5. In other words, the samples are taking into account alternative possibilities for the actual location of the robot. In the case of kidnapping, the new location is stable and the samples will quickly cluster in the new location. In the case of perceptual aliasing, it is just a transient situation lasting for one or two steps that will not spoil the correct estimation of the robot's pose. In both cases, our system is able to quickly calculate the correct position.



(1) (2) (3) (4) (5)

Fig. 6. The kidnapped robot problem solved with the standard approach. Note the robot needs many steps in order to recover from the kidnapping. The small dots are the samples generated by the Monte-Carlo filter, the line with the dots is the actual path of the robot (the dots are the position at which the robot takes the input images), and the thick curve is the estimated path of the robot. The estimated position of the robot is calculated as the average position of the samples and is marked with a cross.



(1) (2) (3) (4) (5)

Fig. 7. The kidnaped robot problem with our newly proposed *kidnapped strategy*. Note the robot needs only 3-4 steps to recovers the correct localisation.

We compared our proposed technique of generating new samples with the standard technique used in [22]. Fig. 6 shows the solution to the kidnapped

robot problem with the standard approach of drawing 10% of new samples uniformly distributed in the environment at random positions. One can see that the system needs many steps to recover from the kidnapping and even after five steps the localisation error is still very high. In Fig. 6, the system is performing the same test using our technique, in just four steps the robot has fully recover from the localisation error. As we will see in Section 4, our system proved to give better performances in all the test we performed. The proposed *kidnapped strategy* has also been applied in the global localisation with a significative speed-up in the convergence of the estimated position to the real position (almost twice as fast).

We did not compared our approach to the Sensor Resetting Localisation (SRL) approach of Lenser et al. [14]. Their approach is well suited in situations were the kidnapping of the robot is frequent (i.e. the robot is frequently lifted and moved away, like in the Four-Legged League RoboCup Competition) and when there are landmarks with known positions. In this case, one can trust more the vision sensor than the localisation method. However, in typical indoor applications kidnapping is not so frequent and our system does not use fix landmarks, but relies only on the appearance of the environment. This appearance might temporary change due to occlusion from people walking by. In the case of temporary occlusion, we do not want all the sample to move away to the next most similar reference image, as it would happen with SRL. Every time there are more than one reference image similar to the current image, we want to take into account all new possible localisations, but with a bias toward the previous location. In other words, we want to trust less the vision sensor than the localisation algorithm.

A possible extension of our approach is to use the idea of "*hierarchical localisation*", introduced in [15]. The hierarchical localisation is the capability of the robot to calculate its position with a tight or broad accuracy. In [15], we showed that the hierarchical localisation can save computational time when a precise localisation is not needed. Here we suggest the hierarchical localisation can be used also to select if the Monte-Carlo filter has to take into account a smaller or a larger number of alternative localisation hypotheses. In our approach, the hierarchical localisation is obtained by controlling the number of Fourier components used to calculate the similarity between the images, therefore controlling the number of reference images that match the current input image. This idea is not discussed further, because it is out of the scope of the this paper.

Fig. 8. The odometric data used in the experiments. Note the odometric information used by the robot is really inaccurate.

4 Experiments

The system was tested along the corridors marked in the map of Fig. 1 (right). This is a large office building composed of a long corridor (about 50 metres) and four shorter corridors forming a loop (the loop is about 20 metres in diametre). The corridors are about 3 metres wide. The total path of the robot is about 100 metres long. Some panoramic snapshots are shown in Fig. 2 (left). As stated before, this environment is particularly challenging because of its high perceptual aliasing, that is all corridors look very similar. They have white walls, gray doors and little colour information is available. Along the path, the robot faces many false localisation hypotheses, but the correct localisation is never lost.

In this test, the reference images were taken every 20 cm on a line approximately in the center of the corridor. Every reference image was labelled with the ground-truth position of the robot. In a second run the robot acquired the input images at different positions along the corridors and the actual positions of the robot were recorded to calculate the localisation error at run time. The tests were performed on the system off-line. The inputs are the sequences of the current images presented one by one every time the robot takes a new step, and the recorded odometry shown in Fig. 8. The aim of the experiments is to demonstrate the system is able to reliably localise the robot and to recover from localisation errors. We carried out several experiments on global localisation, position tracking and robot kidnapping, introducing different amount of noise to the odometers' data. The system is able to successfully localise the robot in any situation.

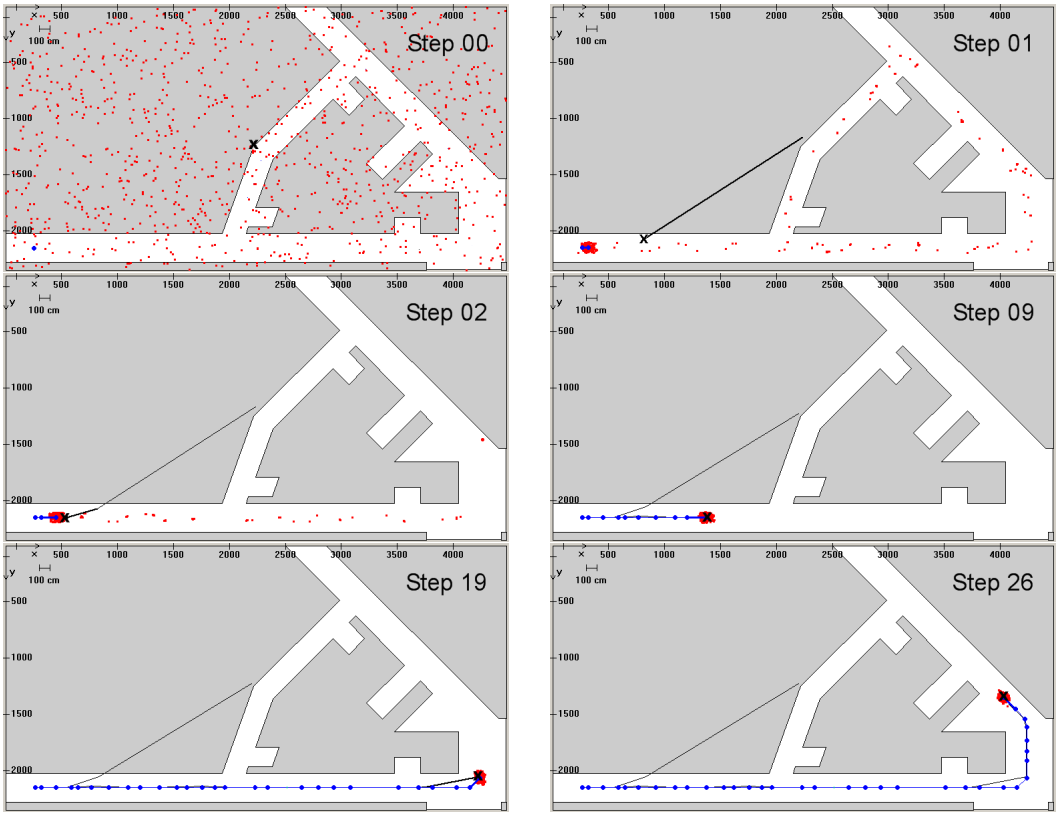


Fig. 9. An example of Monte-Carlo localisation. These are some snapshot of our system while it is performing a global localisation (Steps 00, 01, and 02) and then position tracking (Steps 09, 19, and 26).

Global Localisation and Position Tracking

The screenshots of Fig. 9 show an experiment of global localisation and position tracking. The small red dots are the samples generated by the Monte-Carlo filter, the bigger blue dots are the ground-truth positions of the input images taken by the robot, the thinner blue line is the ground-truth path of the robot, and the thicker black curve is the estimated path of the robot. The estimated position of the robot is calculated as the average position of the samples and is marked with a black cross. Our system is able to localise the robot, without any prior information on the robot's position, after processing about 5-6 images. The correct localisation is achieved even if we use a very low number of samples. Experimentally, we observed that the minimal number of samples to obtain a reliable localisation is about the same as the number of the reference images. In few iterations, the localisation error sets below the distance of the reference images (i.e. 20 cm). In Fig. 9 (Step 00) the robot does not have any information on its position, so it generates samples uniformly distributed in the environment. Note the robot generates samples also in forbidden positions (that is outside of free space and inside wall or obstacles). This is because the robot does not have any information on the geometrical map of the environment. The only information the robot has is

the relative position of the images one with respect to the others. This proves our image-based Monte-Carlo approach works without the need of a geometrical map of the environment. The rough map of the environment presented in this paper is needed only for displaying purposes. In Fig. 9 (Steps 01), the robot grabs the first omnidirectional image and after the resampling stage almost all samples are already concentrated around the correct position. All samples inside the obstacles disappeared. Only few samples are left along the corridors in locations that have a certain amount of similarity with the current input image. In Step 02, after grabbing the second image, samples are even more concentrated and survive only in the first corridor. In Fig. 9 (Steps 09, 19, and 26), the position tracking experiment is presented. It is not shown in the screenshots, but the system is able to keep track of the robot's position when the robot takes long steps (about 200 cm) in dead-reckoning mode (i.e. it travels for a while without tacking any picture) with large odometric errors. In this condition, as soon as a new image is grabbed, the system is able to correct the errors caused by odometry.

Kidnapped Robot

In Fig. 10, we compare again the standard strategy of drawing samples randomly distributed in the environment (left column) with our new kidnapped robot strategy (right column). The test is performed on a much larger environment than the one in Fig. 6 and Fig. 7. The result is that the proposed kidnapped strategy both allows a quick recovery from the localisation error due to the kidnapping and also improves the global localisation (less observations are necessary to cluster all the samples around the actual robot's position). In Fig. 11 a statistical analysis of the performance of the two methods is plotted. We repeated the experiment of Fig. 10 twenty times and we calculated the average error in the case of the standard strategy (red dashed line) and our new strategy (blue solid line). The localisation error decreases toward lower values much faster with our strategy both in the global localisation phase and in the recovering phase after kidnapping. Also in the position tracking phase the localisation error obtained using our method is smaller, because with the standard strategy some of the samples are randomly spread out in the environment and so they bias the estimation of the actual position of the robot, generating a sensible error (see Fig. 11 right, where the plot is zoomed on the residual error).

The Monte-Carlo Localisation software is the most computationally intensive routine in our system. It requires about 170 ms per cycle using 1000 samples. Even if the software was not optimised for computational speed, it could offer real-time performance.

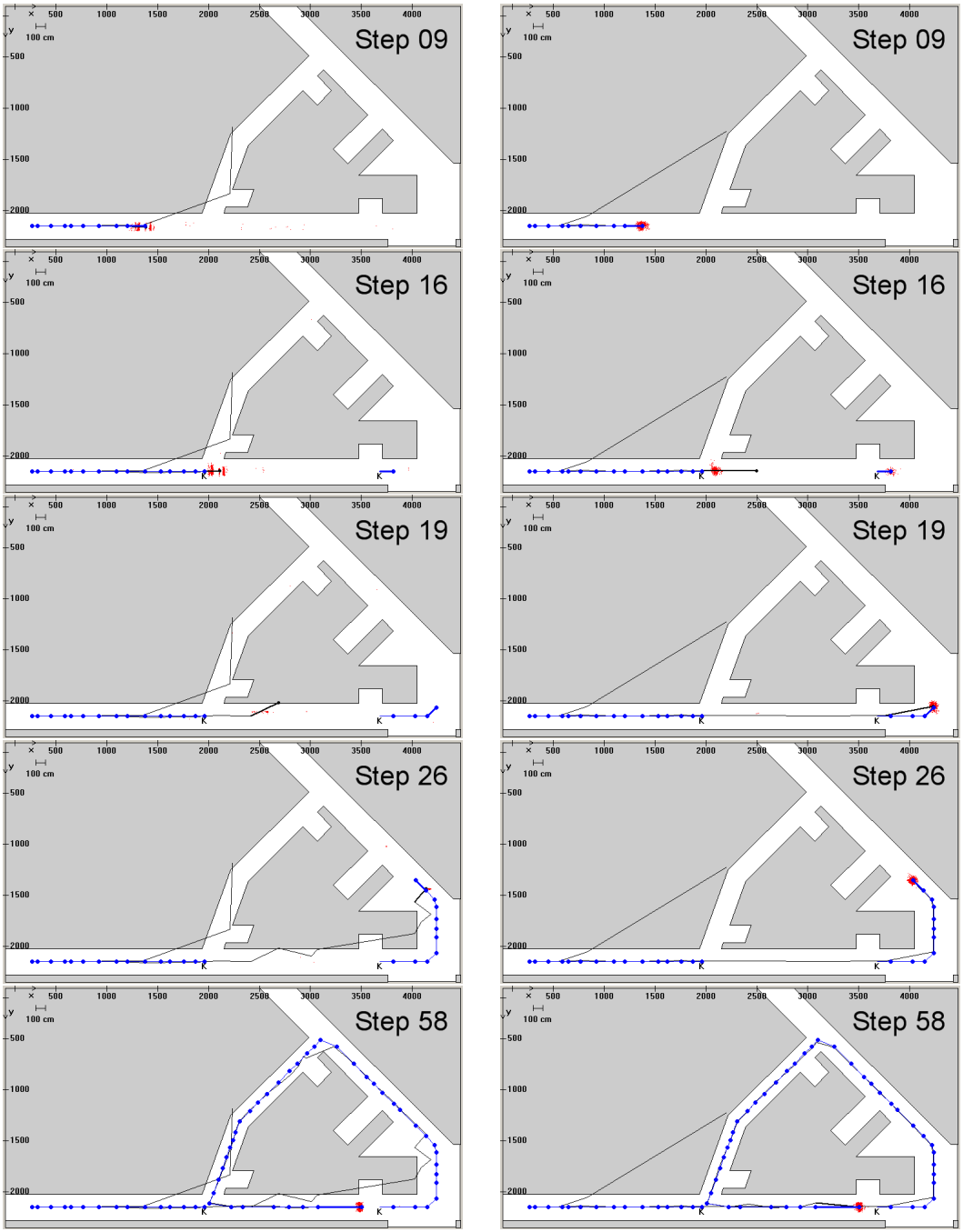


Fig. 10. Here you can see some snapshots in our test environment. The standard kidnapped robot strategy (left series of images) is compared with our new strategy (right series). The blue line is the actual path of the robot and the black curve is the estimated path of the robot. The snapshots show that with our strategy it is possible to speed-up global localisation and also to quickly relocalise after the kidnapping.

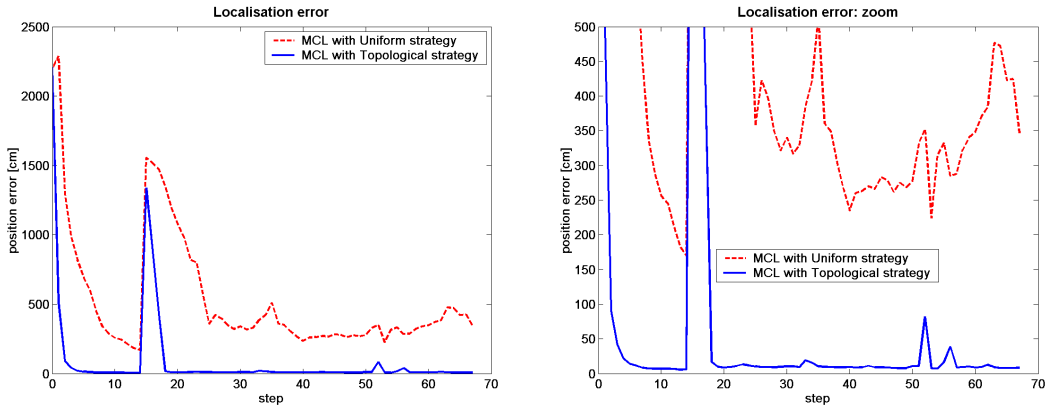


Fig. 11. Average localisation error during the experiment of Fig. 10. The strategy we proposed outperforms the uniform strategy in robustness and speed in recovering the correct position. On the right, a zoom of the plot on the left to show the amount of residual error for the two approaches.

5 Conclusions

In this paper, we presented a robust image-based localisation system that can operate in every type of environment. We presented our solution to the problem of lowering the computational and memory requirements posed by image-based localisation. This solution uses the Fourier transform of omnidirectional images grabbed by the robot. We discussed the advantages of this solution with respect to the solutions devised by other authors. To overcome the limitation of the image-based localisation approach, namely the lack of robustness in the case of perceptual aliasing, we used a Monte-Carlo localisation technique. We showed that this system is able to track the position of the robot while moving and it is able to estimate the position of the robot without any prior knowledge on the real position. Moreover, we showed that our image-based Monte-Carlo approach does not require a geometrical map of the environment, but just the position of reference images with respect to each other (that is a kind of topological map). A new approach in solving the kidnapped robot problem was proposed: we generated a certain number of samples (here 10%) close to all reference images that match the current input image. This approach was shown to outperform the standard approach of generating samples randomly distributed in the environment.

References

[1] H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localisation using omnidirectional images. In A. K. Jain, S. Venkatesh, and B. C. Lovell, editors, *Proc. of the 14th International Conference on Pattern Recognition*, volume vol. I, pages 1799–1803, 1998.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

[3] J. Carpenter, P. Clifford, and P. Fearnhead. An improved particle filter for non-linear problems. In *IEEE Proc. Radar, Sonar and Navigation*, volume 146, 1999.

[4] R. Cassinis, D. Duina, S. Inelli, and A. Rizzi. Unsupervised matching of visual landmarks for robotic homing using Fourier-Mellin transform. *Robotics and Autonomous Systems*, 40(2-3), August 2002.

[5] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, June 1999.

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo Localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of National Conference on Artificial intelligence (AAAI'99)*, pages 343–349, July 1999.

[8] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transaction on Robotics and Automation*, Vol 16(number 6), December 2000.

[9] H.-M. Gross, A. Koenig, C. Schroeter, and H.-J. Boehme. Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2003)*, October 2003, Las Vegas USA.

[10] H. Ishiguro. Development of low-cost compact omnidirectional vision sensors. In R. Benosman and S. Kang, editors, *Panoramic Vision*, chapter 3, pages 23–38. Springer, 2001.

[11] H. Ishiguro and S. Tsuji. Image-based memory of environment. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-96)*, pages 634–639, 1996.

[12] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Proc. of the 15th Int.Conference on Pattern Recognition (ICPR00)*, volume 4, pages 136–139. IEEE Computer Society, September 2000.

[13] B. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appareance-based robot localization. *Image and Vision Computing*, vol. 19(6):pp. 381–391, April 2001.

[14] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Francisco, USA*, pages 1225–1232, 2000.

[15] E. Menegatti, T. Maeda, and H. Ishiguro. Image-based memory for robot navigation using properties of the omnidirectional images. *Robotics and Autonomous Systems, Elsevier*, page (to appear), 2004.

[16] E. Menegatti, F. Nori, E. Pagello, C. Pellizzari, and D. Spagnoli. Designing an omnidirectional vision system for a goalkeeper robot. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V.*, pages 78–87. Springer, 2002.

[17] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Hierarchical image-based localisation for mobile robots with Monte-Carlo localisation. In *Proc. of European Conference on Mobile Robots (ECMR'03)*, pages 13–20, September 2003.

[18] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based localisation Monte-Carlo localisation without a map. In Springer, editor, *AI*IA 2003: Advances in Artificial Intelligence : 8th Congress of the Italian Association for Artificial Intelligence, Proceedings*, pages 423–435, September 2003.

[19] T. Pajdla and V. Hlaváč. Zero phase representation of panoramic images for image based localization. In F. Solina and A. Leonardis, editors, *8-th International Conference on Computer Analysis of Images and Patterns*, number 1689 in Lecture Notes in Computer Science, pages 550–557, Tržaška 25, Ljubljana, Slovenia, September 1999. Springer Verlag.

[20] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. In *International Journal of Robotics Research*, volume Vol. 19, pages 972–999, November 2000.

[21] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for mobile robots. *Artificial Intelligence Journal*, 128(1-2), 2001.

[22] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[23] Y. Yagi, Y. Nishizawa, and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor copis. *IEEE Transaction on Robotics and automation*, VOL. 11(NO. 5):pp. 634–648, October 1995.