Assessing the Performance of SDM-based Robot Navigation with Different Image Processing Techniques

Mateus Mendes*†, A. Paulo Coimbra*, and Manuel M. Crisóstomo*

***Abstract*— Vision is one of the preferred sources of perceptions to implement intelligent robot navigation: it is biologically inspired, requires inexpensive hardware and a single image does contain huge amounts of information. However, problems such as illumination changes, scenario changes and electric noise in the image sensors pose challenges which are difficult to overcome. In previous work the authors developed algorithms to navigate a robot based on sequences of visual memories stored into a Sparse Distributed Memory—a kind of associative memory suitable to work with high-dimensional binary vectors, which also exhibits behaviours in many aspects similar to those of the human brain. This paper analyses the impact of using different image processing techniques to minimise the impact of noise in the images used: histogram equalisation, contrast normalisation and smoothing using a Gaussian filter. The results show that equalisation and smoothing have a positive effect on the performance of the system.**

***Index Terms*—Robot Navigation, View-based Navigation, Image Processing, SDM, Sparse Distributed Memory**

1 Introduction

The area of robot navigation has been subject to intense research in the last decades. Many different approaches have been tried to localise and navigate robots in a safe and robust way. Some of those approaches can only be used in structured environments, for they are based on the recognition of artificial landmarks, beacons or similar strategies that improve the accuracy of the system but require conditions specially arranged for the robot [1]. More generic strategies that work in unstructured environments include mapping and localisation using laser range finders, sonars or cameras.

Vision-based approaches are biologically inspired, since humans use mostly vision for localisation, and vision alone is responsible for about 80% of the sensory input of an average person [2]. To guide a robot based on visual information, the sensors required are inexpensive, but the processing power needed is huge. Every single image is usually described by hundreds or thousands of pixels, and every path that the robot learns is described by tens, hundreds or thousands of images. That makes the technique less appealing, because real-time operation may be compromised for large databases, or requires massive parallel processing. The use of cognitive information, in which the raw images are replaced by formal descriptions of the contents of the images, may solve part of the problem, but even so the initial problem of processing thousands or millions of pixels to extract the relevant information still remains.

There are two popular approaches for vision-based navigation: one that uses plain images [3], the other that uses omnidirectional images [4]. Omnidirectional images offer a 360° view, which is richer than a plain front or rear view. However, that richness comes at the cost of even additional processing power requirements. Some authors have also proposed techniques to speed up processing and/or reduce memory needs. Matsumoto [5] used images as small as 32×32 pixels. Ishiguro replaced the images by their Fourier transforms [6]. Winters compressed the images using Principal Component Analysis [7].

The images alone are a means for instantaneous localisation. View-based navigation is almost always based on the same idea: during a supervised learning stage the robot learns a sequence of views that, if followed with minimum drift, will lead it to a target location. By following the sequence of commands, possibly correcting the small drifts that may occur, the robot is later able to follow the learnt path autonomously [5]. To a great extent, this view-sequence based approach is similar to the way the human brain works [8, 9].

In previous work the authors presented a system to navigate a robot using images stored into a Sparse Distributed Memory (SDM) [10]. The SDM is a kind of associative memory based on the properties of high-dimensional boolean spaces, and thus suitable to work with large bi-

*ISR - Institute of Systems and Robotics, Dept. of Electrical and Computer Engineering, University of Coimbra, Portugal. E-mail: acoimbra@deec.uc.pt, mcris@isr.uc.pt. †ESTGOH, Polytechnic Institute of Coimbra, Portugal. E-mail: mmendes@estgoh.ipc.pt.

nary vectors such as raw images [8]. The present paper describes experiments using different image processing techniques, in order to make the system more robust to image noise and changes in dynamic environments. It is a revised and extended version of [11].

Section 2 explains navigation based on view sequences. Section 3 briefly describes the SDM. In Section 4 the experimental platform is presented. Section 5 describes the image processing techniques used. Section 6 shows and discusses the results obtained, and Section 7 draws some conclusions and opens perspectives of future work.

2 Navigation using view sequences

The approach followed to navigate the robot is based on using visual memories stored into a SDM, as described in [10]. It requires a supervised learning stage, in which the robot is manually guided. While being guided, the robot memorises a sequence of views automatically. It stores a sequence of views for each path. Images that are very similar to previously stored images are discarded, because they would, with high probability, not add any relevant information to the known information.

While running autonomously, the robot performs automatic image-based localisation and obstacle detection. Localisation is estimated based on the similarity of two views: one stored during the supervised learning stage and another grabbed in real-time. To minimise possible drifts to the left or to the right, the robot tries to find matching areas between those two images and calculates the horizontal distance between them in order to infer *how far* it is from the correct path. The technique is described in more detail in [10].

3 Sparse Distributed Memory

The Sparse Distributed Memory is an associative memory model proposed by Kanerva in the 1980s [8]. It is suitable to work with high-dimensional binary vectors. Kanerva shows that the SDM exhibits the properties of large boolean spaces, which are, to a great extent, similar to that of the human cerebellum. The SDM naturally implements behaviours such as tolerance to noise, operation with incomplete data, parallel processing and *knowing that one knows*.

In the proposed approach, an image is regarded as a high-dimensional vector, and the SDM is used simultaneously as a sophisticated storage and retrieval mechanism and a pattern recognition tool. For space constraints the original SDM model is not described in the present paper. A description can be found, for example, in [12]. In the present work, a model that we call "auto-associative arithmetic" is used, as exemplified in Figure 1. It is a very simplified version, optimised to process large arrays of integers in real time, at the cost of loosing some characteristics of the original model. The main modules of the SDM are an array of addresses and an array of data vectors. It is even acceptable an auto-associative version, in which the same array is used simultaneously as addresses and data, as long as datum ζ is only stored at location ζ. The auto-associative memory needs about one half of the storage space.

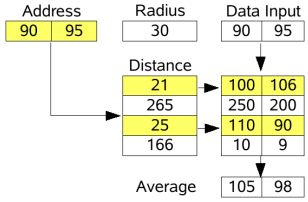


Figure 1: Simplified auto-associative arithmetic SDM, used in real-time for robot navigation.

Every input address will activate all the memory addresses that are within a predefined activation radius. Different methods can be used for computing the distance—the present implementation uses the sum of the absolute differences. In Figure 1 the example address vector $< 90, 95 >$ will activate address $< 100, 106 >$ (distance 21) and address $< 110, 90 >$ (distance 25).

Reading from the memory is done by averaging the integer values columnwise. Learning is achieved updating each byte value using the equation:

$$h_t^k = h_{t-1}^k + \alpha \cdot (x^k - h_{t-1}^k), \quad \alpha \in \mathbb{R} \wedge 0 \leq \alpha \leq 1 \quad (1)$$

In the equation, h_t^k is the k^{th} number of the memory location, at time t, x^k is the corresponding number in the input vector x and α is the learning rate. In the present implementation α was set to 1, enforcing one-shot learning.

The memory locations are managed using the Randomised Reallocation (RR) algorithm [13]. Using the RR, the system starts with an empty memory and allocates new locations when there is a new datum which cannot be stored into enough existing locations. The new locations are placed *randomly* in the neighbourhood of the new datum address.

4 Experimental platform

The robot used is a Surveyor[1] SRV-1 (Figure 2). Among other features, it has a built in digital video camera and a 802.15.4 radio communication module. It is controlled in real time from a laptop. The overall software architecture is as shown in Figure 3. It contains three basic modules: i) the SDM, where the information is stored; ii) the Focus (following Kanerva's terminology), where the navigation

[1] http://www.surveyor.com.

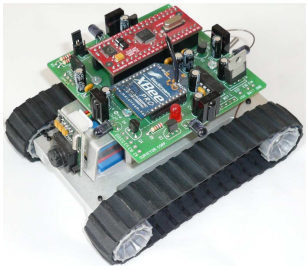


Figure 2: Robot used.

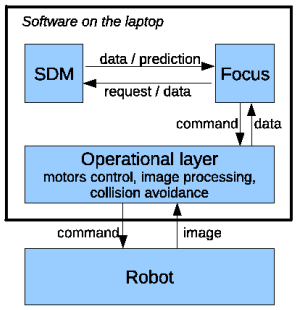


Figure 3: Architecture of the implemented software.

algorithms are run; and iii) an operational layer, responsible for interfacing the hardware and some tasks such as motor control, collision avoidance and image processing.

For vision-based navigation, the vectors stored in the SDM consist of arrays of bytes, as summarised in Equation 2:

$$x_i =< im_i, seq\_id, i, timestamp, motion > \qquad (2)$$

In vector x_i, im_i is the image i, in PGM (Portable Grey Map) format and 80×64 resolution. In PGM images, every pixel is represented by an 8-bit integer: 0 is black, 255 is white. *seq_id* is an auto-incremented, 4-byte integer, unique for each sequence. It is used to identify which sequence the vector belongs to. i is an auto-incremented, 4-byte integer, unique for every vector in the sequence, used to quickly identify every image in the sequence. *timestamp* is a 4-byte integer, storing Unix timestamp. It is not being used so far for navigation purposes. *motion* is a single character, identifying the type of movement the robot performed after the image was grabbed. The image alone uses 5120 bytes. The overhead information comprises 13 additional bytes. Hence, the input vector contains 5133 bytes.

5 Image processing techniques

Three different image processing techniques have been tried: contrast normalisation, equalisation and smoothing.

5.1 Image problems

Even using the best or most expensive cameras, images always suffer from huge amounts of noise and benefit from some image processing. The typical problems which may be minimised by fast algorithms running in real-time are caused by electric noise in the camera sensors, illumination changes and scenario changes.

Digital images are always grabbed by sensors which contain arrays of sensitive cells which in turn capture the value of each single pixel. Small numeric variations due to electrical noise, temperature, camera heading or other factors in a single sensor cell are normal, considering the sensitivity and size of each cell. As the final image is composed of thousands or millions of pixels, the sum of the differences of pixel values between two consecutive images of exactly the same scene can be huge. Applying a Gaussian filter to the image is usually very effective minimising this type of noise.

Illumination changes are another type of noise that is very common and hard to solve. Images captured under sun light are easily affected, for the sun rays change direction and intensity along the day. But even indoors illumination levels change very often, due to causes such as broken, degrading or changed light bulbs, change of reflectiveness of the materials and similar problems. Techniques of contrast stretching and histogram equalisation are popular ones to minimise the problem of illumination differences.

5.2 Contrast enhancement

The quality of images as grabbed directly from the camera depends a lot on ambient illumination. Dim light produces dark images with little contrast, while good illumination provides images with better contrast. The practical consequence of those dynamic environmental conditions is that a robot which has been in one place and captured images of it, will not recognise the same place if it goes there again and the illumination is significantly different. Under dim light the pixel values will tend towards 0, the black pixel. Under strong light the values will move towards the other end of the interval: 255, the white pixel. The difference between two such images may eventually lead the system to consider them as two different pictures, even if they correspond to the same view. The problem may be minimised using techniques to adjust the distribution of the pixel values, enhancing the contrast of the images.

Figure 4 shows an image captured under dim light, and the corresponding frequency distribution of the pixel values and cumulative histograms. As shown, the image is very dark—it is rich in brightness intensities below 150 and contains almost no values above that level. The grey levels with higher frequencies are in general below 50.

The cumulative histogram is a curve which rises very fast until 100 and a horizontal line above 150. Besides the visual discomfort of working with such images (which is only relevant during the supervised learning stage), dealing with them in the robot memory can also be difficult, as stated above. Computing the distance between two images of the same place, one with the histogram tending to the left and the other with the histogram tending to the right, can possibly result in a huge difference, meaning that the images may not be considered the same by the robot.

5.2.1 Contrast stretching

Contrast stretching (also called contrast normalisation) is a method that consists in applying a linear function to the image, in order to improve its contrast. The intensities of an image with poor contrast, such as the test image shown in Figure 4(a), do not extend through the full range of available pixel values. In the case of the test image they all actually fall in the range [7, 178]. Applying a linear transform to the image it is possible to stretch the intensity levels so that they occupy all the range [0, 255], resulting in an image with improved contrast. Mathematically, the technique is simply a function that maps one interval into another interval:

$$f : [a, b] \longrightarrow [c, d] \tag{3}$$

In 3, a and b are the minimum and maximum intensity levels found in the image, and c and d are the minimum and maximum intensity levels that can be used. The value of each pixel P_{in} is then reassigned into P_{out}, according to Equation 4:

$$P_{out} = c + (P_{in} - a)\left(\frac{d-c}{b-a}\right) \tag{4}$$

Figure 5 shows the result of applying the contrast-stretching transform to the image. The result is not impressive. Both ends of the the histogram were slightly stretched, so that it occupies all the range [0, 255]. There is at least one pixel with intensity 0, and there is also at least one pixel with intensity 255, but the general distribution of the pixel intensities is very similar to what can be observed in the original image.

5.2.2 Histogram equalisation

Histogram equalisation is a technique to manipulate images based on the analysis of the corresponding histograms. The method consists in taking the original cumulative histogram of the original image, normalise it towards 255 (or the maximum intensity value that can be found in the image), and then use it as a mapping function to the original image, thus achieving a uniform histogram after the transform. The uniform histogram obtained will correspond to a brightness distribution where all the values should be equally probable. Due to the discrete nature of the digital images, the result is usually just an approximation. Nonetheless, the technique results in a significant improvement of the original images, as shown in Figure 6.

The procedure is formalised as follows. First, let's consider an image x, of L different grey levels and n pixels. Let n_i be the number of occurrences of grey level i in the image. The probability of occurrence of a pixel of level i in the image is given by equation:

$$p_x(i) = p(x = i) = \frac{n_i}{n}, i \in 0, ..., L \tag{5}$$

Probability $p_x(i)$ is, indeed, the same as the grey level's frequency value in the histogram, once it is normalised into the interval [0, 1]. The cumulative probability distribution function of p also coincides with the image's normalised cumulative histogram:

$$c(i) = \sum_{j=0}^{i} p_x(j), i \in 0, ..., L \tag{6}$$

Value $c(i)$ is a transformation that for each pixel value x will produce a corresponding value y, so that the cumulative probability function of y will be linearised across the value range. That is, if it is used the transformation of Equation 7 to map every pixel value in the image from x to y, the output image will exhibit a linear histogram in the co-domain [0, 1].

$$y_i = c(i) \cdot x_i, i \in 0, ..., L \tag{7}$$

To get the image back into the original [0, L] interval, it is necessary to apply a linear transform to stretch the co-domain:

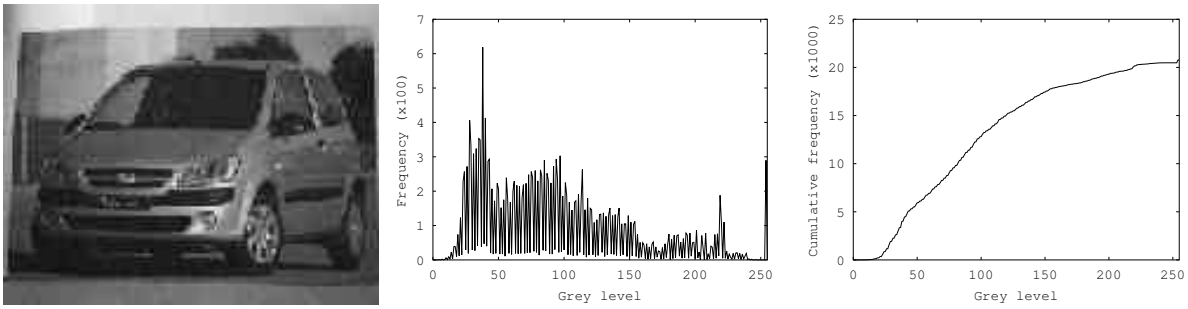
$$y'_i = y_i \cdot L \tag{8}$$

For real time processing, it is possible to merge equations 6, 7 and 8 together. That way it is possible to skip the prior normalisation of the cumulative histogram into the interval [0, 1] and perform all the operations with a single loop to process all the image (providing the cumulative histogram has already been computed) [14], as shown in Algorithm 1.

Algorithm 1: Histogram equalisation

```
begin
    alpha ⟵ L/numPixels;
    for each pixel do
        y ⟵ C(x) * alpha;
    end
end
```

In Algorithm 1, $numPixels$ is the total number of pixels in the image, and $C(x)$ the cumulative frequency of grey level x.

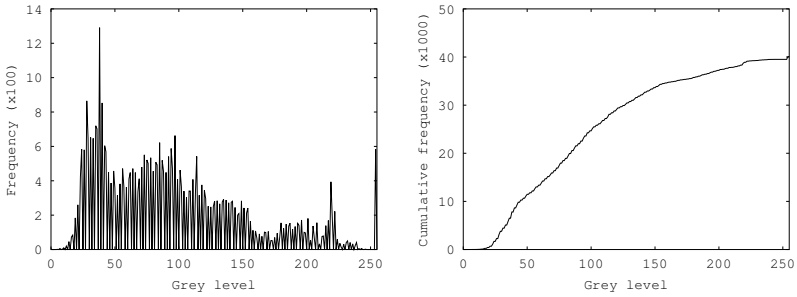


(a) Original image, as captured under dim light.

(b) Histogram.

(c) Cumulative histogram.

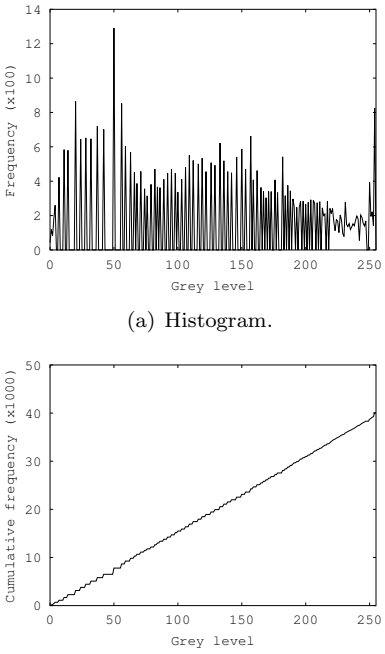
Figure 4: Image captured under dim light, and corresponding frequency and cumulative histograms.



(a) Histogram.

(b) Cumulative histogram.

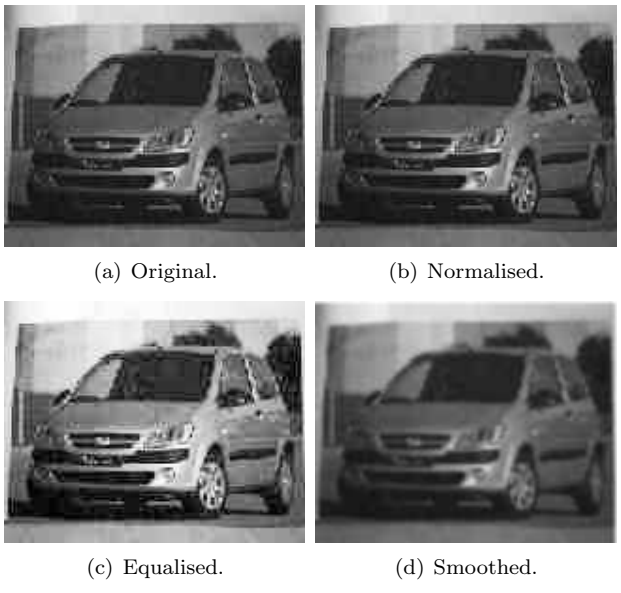
Figure 5: Frequency and cumulative histograms of the image after contrast stretching.



(a) Histogram.

(b) Cumulative histogram.

Figure 6: Frequency and cumulative histograms of the image after histogram equalisation.



(a) Original.

(b) Normalised.

(c) Equalised.

(d) Smoothed.

Figure 7: Comparison of the original, normalised, equalised and smoothed images.

5.3 Smoothing

Gaussian filters are often used in image processing as a way to reduce noise levels. The final image is blurred, and the contours are smoother than they were in the original image. The process consists of running through the image a filter in which each pixel is replaced by a weighted average of the neighbouring pixels, where the nearest neighbours contribute more than the farthest neighbours, according to the Gaussian function. For an image, a two dimensions Gaussian must be applied, as shown in Equation 9.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (9)$$

In the formula, σ is the standard deviation of the Gaussian distribution. A large σ will give more importance to farther pixels, while a small σ gives more importance to the nearest pixels. In practise, pixels outside the radius 3σ are usually ignored, for their contribution to the final value is negligible.

In the present work, the Gaussian filter was applied using OpenCV library[2]. The standard deviation was computed automatically, but only the contributions of the immediate neighbours were used.

Figure 7 shows the original image, a contrast stretched image, an equalised image and a smoothed image. Obviously, the same image can be subject to different processing methods. For example, it makes sense to smooth an image and then equalise. It makes no sense to equalise and contrast stretch, since both methods intend to achieve the same goal, and contrast stretch will have no effect on an equalised image.

6 Experiments and results

To assess the performance of the system, the concept of "Momentary Localisation Error" (MLE) was defined. A MLE is counted when the robot, in the autonomous run mode, retrieves image im_{j-i} after retrieving image im_j, for $i, j > 0$. When that happens, it means that one of the predictions was wrong, because the robot is not expected to get back in the sequence. Most MLEs are not fatal: a wrong prediction may still lead to a correct robot move; a wrong move may not lead the robot to get lost; and the effect of a wrong move may be neutralised right away by the drift correction algorithm. Nonetheless, most MLEs may cause delays to the robot, if they make it perform a wrong move.

6.1 Experiments

The experiments performed consisted in teaching the robot a path described by about 100 images. Later, the robot was made to follow the same path again, under different illumination levels and applying different image processing techniques.

Table 1 summarises the results of one experiment, in which the robot was following a path described by 120 images in a test bed. The path was taught with ambient illumination of about 300 Lux. The first column of the table states the approximate illumination level during the autonomous runs. Columns 2–4 identify which processing methods were applied. Column 5 shows the number of MLEs that have been counted for each experiment. It should be noted that the step of the robot during the autonomous run is about 1/4 of the learning step size, hence 12 MLEs actually represent prediction errors in about 2.5% of the predictions. Column 6 is the average of the best similarity measures, measured between the robot's *current* view and the pool of images in the SDM. Column 7 is the corresponding standard deviation. Columns 8 is the average distance to the second best prediction. Comparing columns 6 and 7 it is possible to have an idea of how successful the system is in separating the best prediction from the second best. Column 9 is the standard deviation of the 8^{th} column. Column 10 is the average distance between the *current* image and the pool of images in the SDM.

6.2 Discussion

As the table shows, contrast normalisation usually produces very bad results. It pulls the images apart, because the average distance increases, but the number of MLEs also increases. When the illumination level is reduced, using contrast stretched images the robot does not complete the path. Equalisation has a very large impact on the average distances. That happens because the pixel values are stretched over all the domain for histogram equalisation. But in this case the processing has a positive impact on the number of localisation errors. And that effect is even more pronounced when the illumination is reduced. Using equalised images the robot is always able to complete the path. Smoothing the images using the Gaussian filter greatly reduces the average distances. That is an expected result, since the pixel values are approximated to the values of the neighbouring pixels. The performance of the system is excellent with smoothing alone, except when the illumination is very faint. In that case equalisation is required for the robot to successfully complete the path.

In summary, the results show that noise reduction through the use of a Gaussian filter and histogram equalisation largely contribute to robust navigation. They reduce the number of momentary localisation errors, thus improving the speed and accuracy of the process, and help the robot finish the path even under dim light.

[2]http://opencv.willowgarage.com/wiki/ (last checked 2012.02.27).

Table 1: Results with path described by 120 images. Suppressed lines on lower illumination levels mean the robot did not finish the path.

Ill.	Processing				Prediction distances average and standard deviation					
Lux	Norm.	Smooth	Eq.	MLE	Best	STD	2.º best	STD	All	STD
300				12	31 666.21	8 800.13	37 165.49	10 390.48	77 590.05	8 504.28
			x	7	70 529.58	9 854.34	84 783.35	15 445.20	173 977.35	6 136.90
		x		8	25 263.00	7 995.92	31 248.93	9 510.58	73 569.17	7 674.41
		x	x	5	53 159.66	10 604.60	67 000.91	16 218.73	161 532.96	7 252.45
	x			46	50 234.40	9 257.81	59 974.55	11 513.63	122 594.43	8 759.42
	x	x		46	39 647.00	9 530.61	49 935.3	12 098.61	115 224.81	8 927.40
200				12	33 668.63	8 770.54	38 923.05	9 030.89	75 506.68	9 253.50
			x	8	72 356.83	12 726.24	86 921.15	15 093.46	167 967.36	6 615.11
		x		8	27 361.70	6 881.68	32 958.81	7 816.18	72 568.28	7 958.29
		x	x	9	55 608.00	15 919.44	68 433.52	17 793.36	157 815.93	8 018.35
100				24	45 513.50	13 110.90	49 117.13	13 634.93	77 992.48	13 913.57
			x	16	98 074.94	18 997.85	107 352.45	17 429.33	177 365.60	9 670.86
		x	x	14	89 827.27	17 764.19	99 179.93	15 511.93	168 941.61	7 271.88

7 Conclusions

Intelligent robot navigation based on visual memories is a long sought goal. However, there are many problems to be solved, such as the presence of noise in the images and illumination changes. The approach followed in the present work relies on memories stored into a SDM. The performance of the system is improved by processing the images using a Gaussian filter and histogram equalisation. Contrast normalisation usually produces bad results. Future research will be done in order to identify relevant objects or features in the images. The vectors stored into the SDM will contain feature information, leading the way to work at a cognitive level. That should improve the performance of the system and possibly decrease memory storage and/or processing time needs. Another open line of research is to improve speed using massive parallel processing, taking advantage of the fact that SDMs architecture is appropriate for that.

References

[1] Christopher Rasmussen and Gregory D. Hager. Robot navigation using image sequences. In *Proc. 13th National Conf. on AI*, pages 938–943. AAAI Press, 1996.

[2] Steven Johnson. *Mind wide open*. Scribner, New York, 2004.

[3] Yoshio Matsumoto, Kazunori Ikeda, Masayuki Inaba, and Hirochika Inoue. Exploration and map acquisition for view-based navigation in corridor environment. In *Proceedings of the International Conference on Field and Service Robotics*, pages 341–346, 1999.

[4] Yoshio Matsumoto, Masayuki Inaba, and Hirochika Inoue. View-based navigation using an omniview sequence in a corridor environment. In *Machine Vision and Applications*, 2003.

[5] Yoshio Matsumoto, Masayuki Inaba, and Hirochika Inoue. View-based approach to robot navigation. In *Proc. of IEEE/RSJ IROS 2000*, 2000.

[6] Hiroshi Ishiguro and Saburo Tsuji. Image-based memory of environment. In *in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 634–639, 1996.

[7] Niall Winters and José Santos-Victor. Mobile robot navigation using omni-directional vision. In *In Proc. 3rd Irish Machine Vision and Image Processing Conference (IMVIP'99)*, pages 151–166, 1999.

[8] Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, Cambridge, 1988.

[9] Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. Times Books, New York, 2004.

[10] Mateus Mendes, Manuel M. Crisóstomo, and A. Paulo Coimbra. Robot navigation using a sparse distributed memory. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, Pasadena, California, USA, May 2008.

[11] Mateus Mendes, A. Paulo Coimbra, and Manuel M. Crisóstomo. Improving sdm-based robot navigation using image processing techniques. In *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2012, WCE 2012*, pages 666–671, London, U.K., 4-6 July 2012.

[12] Mateus Mendes, Manuel M. Crisóstomo, and A. Paulo Coimbra. Assessing a sparse distributed memory using different encoding methods. In *Proc.*

of World Congress on Engineering, WCE 2009, pages 37–42, London, UK, July 2009.

[13] Bohdana Ratitch and Doina Precup. Sparse distributed memories for on-line value-based reinforcement learning. In *ECML*, 2004.

[14] James Matthews. Histogram equalization. *Generation 5*, November 2004. www.generation5.org.