Mobile Robot Localisation with Incremental PCA

Abstract

Mobile robots can employ different sensors to collect data about the environment. We propose to use a special sensor (a catadioptric camera) which provides panoramic views at each robot position. To model the environment for later navigation and localisation, we build a representation of the appearance by compressing the set of panoramic images by Principal Component Analysis (PCA). Since the batch application of the PCA is inappropriate in this case, we propose to apply an incremental approach. This leads to novel aspects regarding the adaptation of compressed partial representation. We provide empirical results which indicate the performance of the proposed method is comparable to the performance of the batch method in terms of compression, computational cost, and, most importantly, precision of localisation.

I. Introduction

Making a mobile robot move through the environment autonomously is a task of combining the readings of the input sensors into a model of the environment. This model is then a kind of a map used for a robot to find its current position (the problem of localisation) and direct further navigation.

Mobile robots are equipped with various sensors. Usually, these include low resolution range scanners in the form of sonar scanners, IR or laser beams. They measure the approximate distance from the sensor to the nearest obstacle in the sensor's direction. In order to use them for reliable models, we often have to calibrate them, and the building of the model requires probabilistic computation [1].

If we mount a camera on an autonomous mobile robot, we can make a sequence of images representing the appearance of the environment from the robot's point of view. So far, in the task of navigation and localisation the visual information has been used primarily as a basis for extracting some higher–level information. Usually, image features are extracted and then 3D information is estimated. Such methods concentrate only on a subset of information contained by the images, discarding the majority of other information. Furthermore, due to noisy input, the estimations can often be spurious, requiring the methods to detect outliers.

An alternative approach is to use whole images. In order to do that, we have to store a large quantity of images, and be able to compare them efficiently with each new image. We assume that the images taken at locations close together are more similar than those taken at locations farther apart. We can then place the robot at some unknown location, and compare the image taken instantly with those already in the model. The images that are the most similar to the new image indicate the current location of the robot. We thus perform the learning and localisation *visually*.

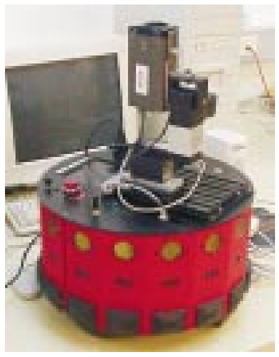
Fig. 1. MagellanPro mobile robot.

A method that would be suitable would have to lets us store a large set of images in a compressed form, and then retrieve them or recognise similar images. Appearance–based learning and recognition methods in computer vision have already been used in applications such as automated face recognition [13], scene recognition, pose estimation and object recognition [5], [11], [12].

Principal Component Analysis (PCA) [2], also known as Karhunen-Loéve Transform, is a method often used for image set compression, reconstruction and recognition. For a set of input images, PCA creates a low–dimensional space in which each image is represented by a point. If two images have a high level of correlation, then the corresponding points lie closer in the subspace. Correlation is a similarity measure that is often used for comparing images [6]. Using PCA, we transform the similarity measure into a distance between two points in a high dimensional space.

PCA thus complies well as a method for learning the environment and for localisation through recognition. However, PCA requires all input images to be known each time we compute the transform. Due to the spacial demands and computational complexity, the computation of the transform becomes forbidding as the number of input images increases.

Fortunately, it is possible to merge two results of PCA into one that is equivalent to the one obtained using all the input data from the two subsets at the same time [4]. It is also possible to update the result with a newly acquired image without having to recompute the entire transform [3], [7], [8].

So far, the incremental methods for computing PCA have been used to overcome the problems with a large set of input data. The data was used for building the set of eigenvectors, and once this was built, the input images were projected. The set of the input images had to be retained throughout the operation.

Having to store all training images cancels out the effect of image set compression. Considering the task at hand, the mobile robot doing the computation using an on–board computer, we might reach the storage limits. By maintaining a low–dimensional space with representations of each training image, we are already keeping an approximation of the training set. Therefore, keeping the original input images can be considered as redundant.

In this paper we study how to incrementally compute the PCA of a set of input images by storing only the image transforms in the high dimensional space. We analyse the effects of updating the representation of the image set, and apply the methods to the visual learning and localisation of a mobile robot.

In Section II, we briefly introduce the PCA. Then we present the incremental PCA. In the last part of this section we explain our contribution and the novel approach to the computation of the training set model. In Section III we present the experiments and their results. Section IV gives conclusions.

II. Method

A. Principal Component Analysis

PCA [2] is a method that takes as input a set of data vectors, then builds a low–dimensional space spanned by a set of orthogonal vectors, and represents the data vectors as points in this new space. In order to use images as the input data, we need to store each $w \times h$ image into a vector with $w \cdot h$ elements.

We denote the input data vectors as $\vec{x}_i$, $i = 1..N$, N being the number of input vectors. We compute the mean of input vectors as $\vec{m}_x = \frac{1}{N}\sum_{i=1}^{N} \vec{x}_i$.

The covariance matrix $\mathbf{C}$ of the input data is then computed as

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i - \vec{m}_x)(\vec{x}_i - \vec{m}_x)^\top \ . \tag{1}$$

We use the covariance matrix to set up an eigenproblem

$$\mathbf{C}\,\mathbf{U} = \mathbf{U}\,\mathbf{\Lambda} \ , \tag{2}$$

obtaining a matrix $\mathbf{U}$ with eigenvectors for columns, and a diagonal matrix of eigenvalues $\mathbf{\Lambda}$.

Eigenvectors form a basis for a high dimensional space into which we project the input images. If we take a data vector $\vec{y}$, we can project it as follows:

$$\vec{w} = \mathbf{U}^\top(\vec{y} - \vec{m}_x) \ . \tag{3}$$

Vector $\vec{w}$ holds the coefficients which then allow for the reconstruction of the image:

$$\vec{y}\,' = \mathbf{U}\vec{w} + \vec{m}_x = \sum_{j=1}^{N} w_j\vec{u}_j + \vec{m}_x \ , \tag{4}$$

where $\vec{u}_j$ is column j of matrix $\mathbf{U}$.

The reconstruction of the images is simply a linear combination of the principal components of the image set. However, if the eigenvectors can not fully represent the image, the reconstruction $\vec{y}\,'$ differs from the original input vector $\vec{y}$ by some residual vector $\vec{h}$:

$$\vec{h} = \vec{y}\,' - \vec{y} \ . \tag{5}$$

Vector $\vec{h}$ is perpendicular to all eigenvectors in $\mathbf{U}$.

The property of PCA is that the eigenvectors with higher eigenvalues store a higher level of information (eigenvalues indicate the level of variance in the direction of the corresponding eigenvector). Therefore, we can discard columns of $\mathbf{U}$ that correspond to the lowest eigenvalues, dropping only a low amount of information, while at the same time gaining on the level of compression.

For computation of PCA, we need to use all the input data at once. Hence, this method has also been named *batch method.*

B. Incremental PCA

The expression (2) features matrices that increase with the number of input images. The computation of the solution of the eigenproblem becomes computationally too expensive. Fortunately it is possible to update the eigenvectors $\vec{u}_j$, $j = 1..p$, eigenvalues $\vec{\lambda}$ and mean value $\vec{m}_x$ by an additional new data vector without having to recompute the entire set.

Here we summarise the method described in [3]. First, we update the mean:

$$\vec{m}\,'_x = \frac{1}{N+1}(N\vec{m}_x + \vec{y}) \ . \tag{6}$$

The covariance matrix can be updated as well:

$$\mathbf{C}\,' = \frac{N}{N+1}\mathbf{C} + \frac{N}{(N+1)^2}(\vec{y} - \vec{m}_x)(\vec{y} - \vec{m}_x)^\top . \tag{7}$$

We update the set of eigenvectors by adding a new vector and rotating them. Since the residual vector (5) is orthogonal to all eigenvectors, its normalised equivalent is suitable for the additional vector:

$$\vec{h}_n = \frac{\vec{h}}{||\vec{h}||_2} . \tag{8}$$

If the residual vector equals zero (i.e. when the eigenvectors fully represent vector $\vec{y}$) we set $\vec{h}_n = \vec{0}$. We obtain the new matrix of eigenvectors $\mathbf{U}'$ as

$$\mathbf{U}' = \left[\mathbf{U}\ \vec{h}_n\right] \mathbf{R} , \tag{9}$$

where $\mathbf{R}$ is a $(p+1) \times (p+1)$ rotation matrix. $\mathbf{R}$ is a result of the eigenproblem of the following form:

$$\mathbf{D}\ \mathbf{R} = \mathbf{R}\ \mathbf{\Lambda}' . \tag{10}$$

According to [3], to compose $\mathbf{D}$, we start with (7) and (9), and using the form of (2) we get

$$\mathbf{D} = \frac{N}{N+1} \left[\begin{array}{cc} \mathbf{\Lambda} & \vec{0} \\ \vec{0}^\top & 0 \end{array} \right] + \frac{N}{(N+1)^2} \left[\begin{array}{cc} \vec{w}\vec{w}^\top & \gamma\vec{w} \\ \gamma\vec{w}^\top & \gamma^2 \end{array} \right], \tag{11}$$

where $\gamma = \vec{h}_n(\vec{y} - \vec{m}_x)$.

There are other ways to construct $\mathbf{D}$. However, only the one described in [3] allows the change of mean. Other authors [7], [8] do not maintain the mean at all (i.e. the mean vector is always assumed to be zero). We decided to follow the method that allows for the updating of the mean, because (also according to [3]) not doing so contributes to worse classification and bigger residuals.

The new matrix $\mathbf{U}'$ contains $p+1$ eigenvectors. We can then decide to reduce the number of eigenvectors back to p. A criterion for doing so can be one of the standard criteria which include either keeping only a stipulated number of eigenvectors (e.g. a fraction of the number of the input images), discarding the eigenvector with the least eigenvalue unless it exceeds a stipulated fraction of the eigenspectrum energy, or keeping the eigenvectors with eigenvalues that exceed an absolute threshold.

C. Our contribution

We have focused on different ways of employing the representation of the visually learnt environment without having to keep the original images. In the process of learning, we would like to update our knowledge base with each new image, and also update all the representations of the previous observations to become compatible with the updated eigenvector model.

Assume we have at some point received n data vectors $\vec{x}_i$, $i = 1..n$. We have already used them to build a representation as a space spanned by p eigenvectors $\vec{u}_j$, $j = 1..p$. Apart from the mean, the eigenvectors and corresponding eigenvalues, we keep the representations of each $\vec{x}_i$ in the form of a coefficient vector $\vec{w}_{i(n)}$. As we update the coefficient vectors each time a new observation is taken into account, the value of the coefficient vector depends on the time the observation was taken. Hence the index (n).

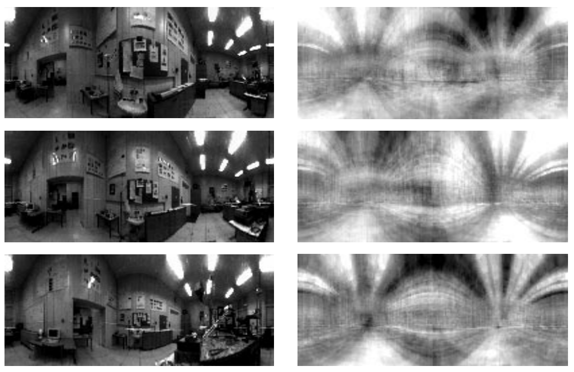

Fig. 2. a) A sample of images from the sequence taken with the mobile robot in a laboratory. The images have been transformed into the cylindrical form. b) First three principle components (eigenvectors).

When the new observation $\vec{x}_{n+1}$ arrives, we compute the new mean using (6), we construct (11) (using $\vec{x}_{n+1}$ in place of $\vec{y}$) and solve (10).

In order to update the coefficients $\vec{w}_{i(n)}$, we first have to reconstruct each image:

$$\vec{x}_{i(n)} = \mathbf{U}\vec{w}_{i(n)} + \vec{m}_x \tag{12}$$

and project it to the new space of eigenvectors:

$$\vec{w}_{i(n+1)} = (\mathbf{U}')^\top(\vec{x}_{i(n)} - \vec{m}_x') . \tag{13}$$

Note that if $\vec{w}_{i(n)}$ has p elements, then $\vec{w}_{i(n+1)}$ has $p+1$ elements, which complies with the increase of the number of the eigenvectors. All the data denoted earlier by $\vec{x}_{i(n)}$ including the new data vector $\vec{x}_{n+1}$ are fully represented by the eigenvectors $\vec{u}'_j$. Therefore, there are no differences between $\vec{x}_{i(n)}$ and $\mathbf{U}'\vec{w}_{i(n+1)}$.

Once all the observations are updated according to the newly–arrived observation, we can decide whether to keep the new dimension or discard it. We can make this decision based on one of the criteria already listed in the previous subsection. However, since the observations are kept only as projections of the input data, the removal of the eigenvectors affects these observations as well.

To follow a life cycle of an observation, let us assume at some point a data vector $\vec{x}_k$ has been added to the eigenvector representation. At first, this vector is fully recoverable from the presentation with the eigenvectors. If we decide, however, to reduce the number of eigenvectors, we can only recover the approximation $\vec{x}_{k(k+1)}$ which differs from $\vec{x}_k$ by some non-zero residual. The next time we decide to reduce the number of eigenvectors after having inserted a few additional observations, we will be left with $\vec{x}_{k(l)}$, $l > k$, which is an approximation of $\vec{x}_{k(k+1)}$, etc.

Therefore, we can think of a few additional criterion for reducing the number of eigenvectors from $p+1$ back to p. We can decide on whether the overall reconstruction error would exceed an absolute threshold when reducing to p. The basic idea behind this method is to find out how much error we make if we update the eigenvectors and then discard one dimension. The method can be summarised as follows:

1. Compute $\vec{x}_{i(n)}$, $i = 1 \ldots n$. Let $\vec{x}_{n+1(n)} = \vec{x}_{n+1}$.
2. Update the eigenvectors for $\vec{x}_{n+1}$ using the methods described above. We obtain $p+1$ new eigenvectors.
3. Discard $\vec{u}'_j$ with the smallest eigenvalue.
4. Compute $\vec{x}_{i(n+1)}$, $i = 1 \ldots n+1$.
5. Compute $E = \sum_{i=1}^{n+1} ||\vec{x}_{i(n+1)} - \vec{x}_{i(n)}||_2$.

E is an error measure we can then use for comparing to a threshold. If it exceeds the threshold, we establish that discarding the new dimension would cause too big a degradation in the approximation of the previous observations. If this is the case, then we should keep all $p+1$ eigenvectors. Otherwise, we can retain only p eigenvectors. This method gives more control over the way we handle the observations.

III. Experimental results

With the experiments we tested how well the incremental approach of building the eigenvector representations compares to the batch method, and what the effects of keeping and updating the observations only in the form of the coefficients are. We also performed some real localisation tests.

We captured most of the images using a camera mounted on our MagellanPro autonomous mobile robot (Figure 1). On the top of the camera we placed a hyperbolic mirror which enabled an ordinary camera to capture panoramic images. The images show the surroundings of the robot as it is reflected in the mirror. The mirror itself covers only a circular portion of the image. Even though PCA can take the images in such a form, we transform the hyperbolic images into cylindrical images. This transformation is carried out by mapping image pixels in polar coordinates into Cartesian coordinates (Figure 2a). We thus obtained a cylindrical image.

As the robot moves, it internally keeps track of its position in space and orientation. The coordinates of the so–called odometry were reliable enough to assign them to each image. We could also align each image in a way that they appeared to have been all taken by a robot facing in the same direction. We performed this by shifting the columns of the cylindrical images to simulate the rotation. With this internal compass we avoided the problems with arbitrary orientation.

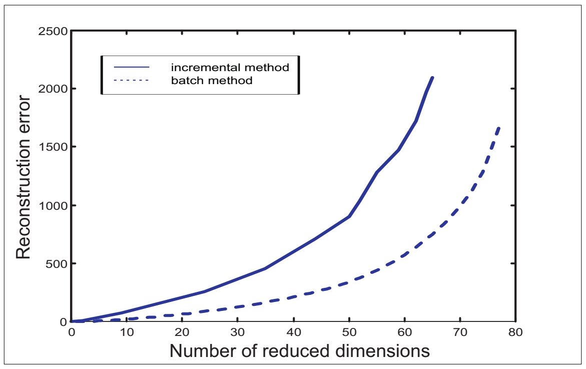

Fig. 3. Comparison of the reconstruction error for batch method and incremental method.

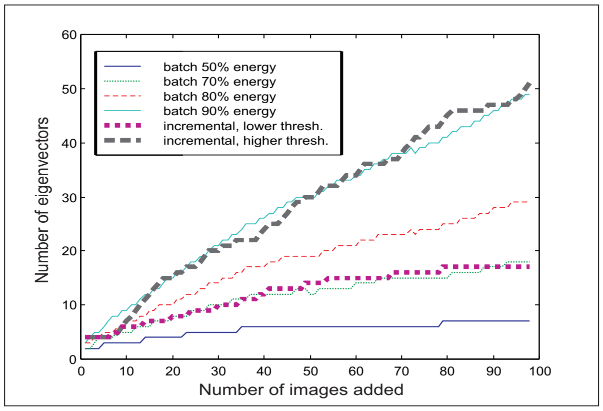

Fig. 4. Comparison of batch method and incremental method by the number of eigenvectors. The curves made by the batch method represent the number of eigenvectors containing a certain percentage of the overall eigenvalue energy. The incremental methods produce a number of eigenvectors we retain. The values depend on the number of images we add.

A useful measure of performance of the PCA is the overall reconstruction error. We can obtain it by summing the norms of the residual vectors of each observation. Figure 3 shows the comparison between the overall reconstruction error for batch method and incremental method. The reconstruction error depends on the number of eigenvectors we discard. For the batch method, after having computed PCA, we discarded the least significant eigenvectors one by one. In each iteration, we computed the overall reconstruction error. For the incremental method, for each point on the curve a whole run was carried out, only varying the threshold between the runs.

The results indicate the reconstruction error grows roughly twice as fast for the incremental method as it does for the batch method. Hence, if we want to obtain the same accuracy, we have to adjust the threshold parameter so that the number of discarded eigenvectors is halved. However, since we need to keep no additional data vectors, we still consider the performance of the incremental method to be good.

Another way of comparing the incremental method with the batch method is through observing the dis-

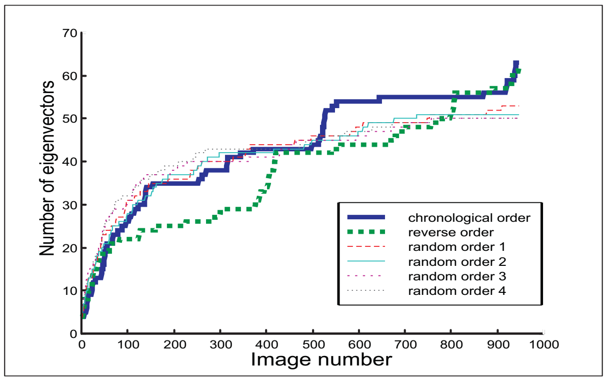

Fig. 5. The growth of the number of eigenvectors as the new training images are being added. Chronological, reverse and four different random data orders were used. 950 images were added altogether.

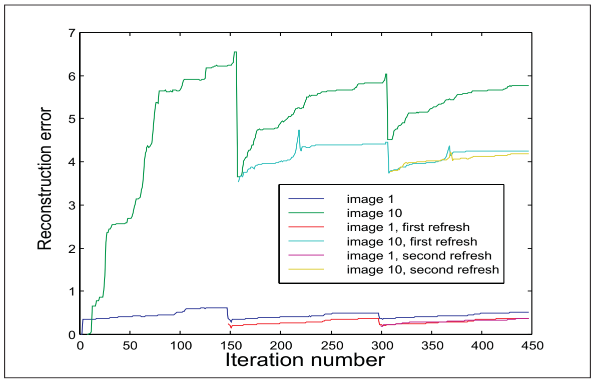

Fig. 6. Reconstruction error for individual images when using the same data sequence multiple times. The sequence of 150 images was used three times. The chart shows reconstruction error for images 1 and 10.

tribution of eigenspectrum energy. If we use a batch method, we often decide to keep a certain number of eigenvectors such that the accumulation of the corresponding eigenvalues contains a stipulated percentage of the entire eigenvalue energy. By running the incremental method, we can observe how the number of preserved eigenvectors changes according to criteria as we add new data vectors.

Figure 4 shows two superimposed charts, one for the batch method and the other one for the incremental method. The same data has been used for both methods. In the case of the incremental method, two runs were carried out, changing only the threshold parameter. We can see that the results compare well: in the case of a lower threshold parameter value used, the growth of the number of eigenvectors was very similar to the growth of the same value for batch method where 90% of the energy was retained. In the second run of the incremental method we raised the threshold, and the curve becomes comparable to the one for batch method with 70% of eigenvalue energy retained.

We then experimented how the order in which we add the images affects the growth of the number of the eigenvectors.

We first used the same set of images and the same thresholds as in the previous experiment. The chronological (from first to last) order and reverse order were used. The results are shown on the Figure 5. The curves show different dynamics, but finally they reach the same value. The figure also shows the behaviour of the incremental methods by adding a random order of updating the eigenvectors. This order is not very life–like, but it shows some interesting results, nonetheless. We can see that in the end, with the random order, we end up with less eigenvectors than with the sequential order. This type of order thus forces the space of eigenvectors to become more descriptive in the earlier stage due to the diversity of the data.

Keeping data only as projections into the space of eigenvectors, we make changes to their presentation each time we modify the eigenvectors. During the course of experiments, we can observe the difference between the original data and the current representation. These differences are expressed in the form of the overall reconstruction error. We can obtain it by summing the norms of the residual vectors of each observation.

In practice, when we explore the environment and learn the model, we can come across the same observation multiple times. Therefore, we experimented the effect of revising the observations. We took a sequence of 150 images and used it three times in a row as the input of our method. Again, we observed the reconstruction error of individual observations.

Figure 6 shows the results for two images in the sequence. We can see that, after we start with the same sequence (image 151 in the sequence is the same as image 1, image 152 equals image 2 etc.), the representation of the one already in the model improves. At the end of the second sequence we notice the reconstruction errors are lower than they were at the end of the first sequence. We take similar observations during the third repeated sequence.

Hence, we can safely replace the old observations with the newly acquired equivalents. With a higher level of repetition, we can anticipate a better model of the environment.

A true test of the algorithm is when solving a task of localisation. Figure 7 shows a result of such experiment. The circles on the figure denote the locations in real–world coordinates where each image was taken (Figure 2a). We used a sparse, equally spaced grid of 62 locations for a set of training images. The squares in the grid had 60×60 cm. We then made another collection of 100 images taken at locations surrounded by at least four neighbouring locations previously used for training. We ran the incremental PCA algorithm which produced a space with 16

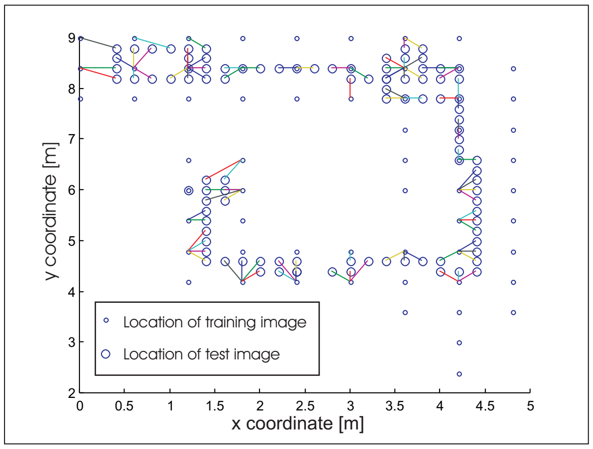

Fig. 7. Localisation with incremental PCA. The figure shows the locations where images have been taken. The lines connect each test image location with the training image location found by the algorithm.

eigenvectors (Figure 2b), which is 26% of the number of training images. Test images were then projected into this space, and they were matched with the closest training coefficient vector.

The algorithm gave excellent performance on our test images. As we can see from the Figure 7, it indicated one of the four neighbouring locations for each test image, scoring a 100% recognition rate.

IV. Conclusion

We investigated the incremental method of Principal Component Analysis and applied it to building a view–based representation for mobile robot. We exploited its capabilities of storing the input data without having to use much preprocessing which would reduce or alter the information contained by the data.

We proposed a method which requires less data storage. During the training, it uses each input image in its full form only once. Once we updated the model and the previous observations using the new observation, we only kept its reduced equivalent.

We came to a conclusion that our method compares well with the batch method in terms of the information storage capabilities. Because all data is not present during the computation, our method requires a larger number of eigenvectors for the same level of approximation. From the computation point of view, when we deal with a large set of data, the proposed method becomes increasingly more efficient than the batch method. Considering we also need less storage, the gain is larger than the cost.

Additional experiments indicated the method performs well when applied to some task. Specifically, the localisation through recognition of the images gave promising results even when the level of compression was high. We are encouraged to use the methods to further investigate the possibilities of navigation using visual information. The method can be exploited for other tasks involving appearance recognition as well.

Further research includes using the proposed method for robust recognition and localisation where occlusion has been introduced to the test images. PCA has already been used for robust recognition and pose determination [9] and for robust localisation [5].

We will also consider the possibilities of incrementally building multiple low–dimensional model representations in place of one higher–dimensional model. We expect, as a result, each type of environment (e.g. corridor, laboratory) will have its own model which would better represent each type of environment than a more general model [10].

References

[1] Martin C. Martin, Hans P. Moravec, *Robot Evidence Grids*, Carniege Mellon University, 1996

[2] H. Murase and S. K. Nayar, Visual learning and recognition of 3-d objects from appearance, *Int. J. Comput. Vision*, 14:5-24, 1995

[3] P. Hall, D. Marshall and R. Martin, *Incremental Eigenalysis for Classification*, British Machine Vision Conference, vol. 1, September 1998, pp. 286-295

[4] P. Hall, D. Marshall and R. Martin, *Merging and Spliting Eigenspace Models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 9, September 2000, pp. 1042-1048

[5] M. Jogan, A. Leonardis, Robust localization using panoramic view-based recognition, *15th International Conference on Pattern Recognition, Barcelona, Spain, September 3 - 7, 2000*, Proceedings, Vol. 4, IEEE Computer Society, 2000, pp. 136-139

[6] E. Trucco, A. Veri, *Introductory Techniques for 3-D Computer Vision*, Prentice-Hall., 1998.

[7] A. Levy, M. Lindenbaum, Sequential Karhunen-Loeve Basis Extraction and its Application to Images, *IEEE Transactions on Image Processing*, vol. 9, no. 8, August 2000, pp. 1371-1374

[8] S. Chandrasekaran *et. al.*, An Eigenspace Update Algorithm for Image Analysis, *Graphical Models and Image Processing*, vol. 59, no. 5, September, pp. 321-332, 1997

[9] D. Skočaj, A. Leonardis, Robust recognition and pose determination of 3-D objects using range images in eigenspace approach, *Third International Conference on 3-D Digital Imaging and Modeling*, proceedings, 28 May - 1 June 2001, IEEE Computer Society, pp. 171-178

[10] A. Leonardis, J. Maver, H. Bischof, Multiple eigenspaces by MDL, *15th International Conference on Pattern Recognition, Barcelona, Spain, September 3 - 7, 2000*, Proceedings, Vol. 1, IEEE Computer Society, 2000, pp. 233-237

[11] H. Murase, S. K. Nayar, Visual learning and recognition using 3-D object from appearance, /it International Journal of Computer Vision, 14:5-24, 1995

[12] S. K. Nayar, H. Murase, S. A. Nene, Parametric appearance representation, In S.K. Nayar and T. Poggio, editors, *Early Visual Learning*, pp. 131-160, Oxford University Press, 1996

[13] M. Turk, A. Pentland, Eigenfaces for recognition, *Journal of Cognitive Neuroscience*, 3(1):71-86, 1991