# Mobile Robot Localization Using an Incremental Eigenspace Model

Matej Artač, Matjaž Jogan, Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25, 1001 Ljubljana, Slovenia
{*matej.artac, matjaz.jogan, ales.leonardis*}*@fri.uni-lj.si*

*Abstract*— **When using appearance-based recognition for self-localization of mobile robots, the images obtained during the exploration of the environment need to be efficiently stored in the memory. PCA offers means for representing the images in a low-dimensional subspace, which allows for efficient matching and recognition. For active exploration it is necessary to use an incremental method for the computation of the subspace. While such methods have been considered before, only the on-line construction of eigenvectors has been addressed. Representations of the images in the subspace were computed only after the final subspace had been built, requiring that all the images were kept in the memory. In this paper we propose to use an incremental PCA algorithm with the updating of partial image representations in a way that allows the robot to discard the acquired images immediately after the update. Such a model is open-ended, meaning that we can easily update it with new images. We show that the performance of the proposed method is comparable to the performance of the batch method in terms of compression, computational cost and the precision of localization. We also show that by applying the repetitive learning, the subspace converges to that constructed with the batch method.**

*Keywords*— **Robot localization, on-line visual learning, PCA updating, view-based robot localization, repetitive learning.**

## I. Introduction

In this paper we approach the problem of mobile robot localization as a task of recognizing a panoramic view from a set of panoramic views acquired in the learning phase (Figure 1). The main motivation for applying appearance-based recognition to the problem of localization is the analogy between recognizing an object in the scene and recognizing the environment. In contrast to object recognition, the target to be recognized in the case of localization is not only a part of the image (on a cluttered background), but rather the complete image.

If we use panoramic images as representations of locations, the views taken from nearby positions and oriented in the same way will be strongly correlated as it is the case when looking at an object from two nearby viewpoints. This allows us not only to design an efficient strategy based on correlation, but also to build a compact representation that eliminates redundancy.

For building a compact model from a set of images, the approach which constructs a *space of eigenvectors* proved itself as a viable one [1], [2]. Therefore, several researchers also in the area of robotics and computer vision developed working localization algorithms and tested them in real-world environments [3], [4], [5], [6]. This approach is basically a two-stage procedure: in the *learning stage*, several panoramic snapshots are acquired which together form a good depiction of the environment. Then, an orthonormal basis is computed (Figure 2), which allows us to represent every image of the training set with only a few parameters. Furthermore, intermediate images can be approximated by a spline interpolation of parameters. In the *localization stage*, the robot is acquiring momentary images, which we then project into the low-dimensional subspace. We search for the nearest point in the subspace, and associate it to the robot's presumed location.

The approach applied in the standard way, however, has its drawbacks. One of them is the fact that the localization stage is strictly separated from the learning stage. In the learning stage we capture all images first, and only then can we construct the model. The model built in this way can not be modified unless we keep the original images. To update the model with new images, we have to construct a new one from the scratch. Therefore, standard approaches are not optimal for performing simultaneous learning (environment exploration) and localization (SLAM [7]). Furthermore, the original images take a lot of storage.

To overcome these problems, we can instead apply an incremental method for building the subspace. Incremental computation of eigenvectors has been considered before [8], [9], [10], [11], [12]. However, for a method to be completely on-line, we have to simultaneously update both the eigenvectors and the low-

The authors acknowledge the support from the Ministry of Education, Science and Sport of Republic of Slovenia (Research Program 506).

Fig. 1. Three panoramic images from the sequence taken inside a laboratory. The images have been transformed into the cylindrical form.

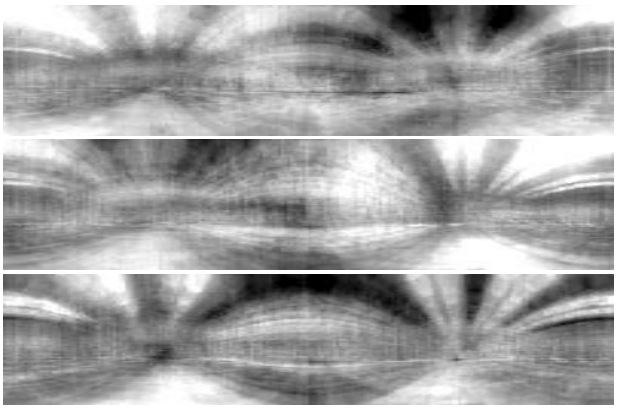

Fig. 2. First three eigenvectors (eigenimages) created by PCA from the laboratory sequence (Figure 1).

dimensional representations of images. In this way, we can discard the original images immediately after the updating of the subspace. One has to be aware, however, that the low-dimensional representations of the images are only approximations of the originals.

In this paper we propose to use an incremental PCA algorithm with the updating of partial image representations without keeping the original images. We discuss how to update the representation with a new observation in order to keep the overall reconstruction error bounded. We analyze how the incremental building influences the accuracy of the representation and compare it to the batch method. We show that 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 localization. We also examine the possibility of improving the model with repetitive learning.

The paper is organized as follows. In Section 2 we briefly review the standard procedure for building the space of eigenvectors. In Section 3 we first describe the incremental method for computing the space of eigenvectors. We then introduce the method for the on-line updating of the image representations and discuss how to efficiently update the model. In Section 4 we give a comparison to the batch method, examine the improvement obtained by repetitive learning, and test the localization on a large database. We conclude with a summary and an outline of the work in progress.

## II. PCA

In this section we briefly outline the standard procedure of building the space of eigenvectors from a set of training images. We represent images from the training set as normalized image vectors $\mathbf{x}_i \in \mathbb{R}^{m\times 1};\ i = 1\ldots n$, where $m$ is the number of pixels in the image and $n$ is the number of images.

The most straightforward way for computing the eigensystem is by solving the SVD of the covariance matrix $C \in \mathbb{R}^{m\times m}$ computed as

$$C = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top , \qquad (1)$$

where $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ is the mean image vector.

The eigenvectors $\mathbf{u}_i$, $i = 1\ldots n$ corresponding to non-zero eigenvalues of the covariance matrix span a subspace of a maximum of $\min(m, n)$ dimensions. If we sort the eigenvectors according to the corresponding eigenvalues, we get an ordering which reflects the principal directions of variance in the learning set of images. We can therefore choose a subset of only $k$ eigenvectors with the largest eigenvalues to be included in the model. Each image can thus be optimally approximated in the least-squares sense up to the degree of error by taking into account the $k$ most informative eigenvectors only. Namely, every model image $\mathbf{x}_i$ projects into some point $\mathbf{a}_i$ in the $k$-dimensional subspace, spanned by the selected eigenvectors [1].

## III. Incremental PCA

To overcome the inability of the batch method to progressively acquire environment representation, and the high storage demands, several algorithms were developed that allow for an incremental building of the subspace of eigenvectors [8], [9], [10], [11], [12]. It was shown in [12] that it is possible to merge two sets, ($U$, $\boldsymbol{\lambda}$, $\bar{\mathbf{x}}$) and ($V$, $\boldsymbol{\mu}$, $\bar{\mathbf{y}}$), where ($U$, $\boldsymbol{\lambda}$) and ($V$, $\boldsymbol{\mu}$) are (*eigenvectors*, *eigenvalues*) matrix pairs, and $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are mean vectors, into a single set ($W$, $\boldsymbol{\omega}$, $\bar{\mathbf{z}}$) that forms the new representation of all the input data, where $W$ is a novel set of eigenvectors, $\boldsymbol{\omega}$ the novel set of eigenvalues and $\bar{\mathbf{z}}$ the new average vector.

A special case of such merging is updating the eigenvectors, eigenvalues, and mean value by a single new

image. We assume we have already built a set of eigenvectors $\mathbf{u}_j$, $j = 1 \dots p$, after having used the images $\mathbf{x}_i$, $i = 1 \dots n$ as an input. The corresponding eigenvalues are $\boldsymbol{\lambda}$ and the mean image is $\bar{\mathbf{x}}$. We would like to update this information to take into account a new image $\mathbf{x}_{n+1}$.

Here we briefly summarize the method described in [11]. First, we update the mean:

$$\bar{\mathbf{x}}' = \frac{1}{n+1}(n\bar{\mathbf{x}} + \mathbf{x}_{n+1}) \,. \tag{2}$$

The covariance matrix can be updated as well:

$$C' = \frac{n}{n+1}C + \frac{n}{(n+1)^2}(\mathbf{x}_{n+1} - \bar{\mathbf{x}})(\mathbf{x}_{n+1} - \bar{\mathbf{x}})^\top. \tag{3}$$

We now have to update the set of eigenvectors in order to reflect the additional image $\mathbf{x}_{n+1}$. Since the residual vector $\mathbf{h}_{n+1} = (U\mathbf{a}_{n+1} + \bar{\mathbf{x}}) - \mathbf{x}_{n+1}$ is orthogonal to each eigenvector in $U$, its normalized equivalent is a suitable candidate for the additional vector in the basis:

$$\hat{\mathbf{h}}_{n+1} = \begin{cases} \frac{\mathbf{h}_{n+1}}{||\mathbf{h}_{n+1}||_2} & \text{if } ||\mathbf{h}_{n+1}||_2 \neq 0 \\ \mathbf{0} & \text{otherwise} \end{cases} \,. \tag{4}$$

We obtain the new $m \times (k+1)$ matrix of eigenvectors $U'$ by appending $\hat{\mathbf{h}}$ to the eigenvectors $U$ and rotating them:

$$U' = \left[ U \; \hat{\mathbf{h}}_{n+1} \right] R \,, \tag{5}$$

where $R$ is a $(k+1) \times (k+1)$ rotation matrix. $R$ is the solution to the eigenproblem of the following form:

$$D \, R = R \, \Lambda' \,. \tag{6}$$

$D$ is a $(k+1) \times (k+1)$ matrix consisting of known components of $\Lambda$ and $\mathbf{x}_{n+1}$, and $\Lambda'$ is a diagonal matrix of the new eigenvalues. According to [11], we can construct $D$ as

$$D = \frac{n}{n+1}\begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{n}{(n+1)^2}\begin{bmatrix} \mathbf{a}\mathbf{a}^\top & \gamma\mathbf{a} \\ \gamma\mathbf{a}^\top & \gamma^2 \end{bmatrix}, \tag{7}$$

where $\gamma = \hat{\mathbf{h}}_{n+1}(\mathbf{x}_{n+1} - \bar{\mathbf{x}})$ and $\mathbf{a} = U^\top(\mathbf{x}_{n+1} - \bar{\mathbf{x}})$.

There are other ways for constructing $D$. However, only the method described in [11] allows for the updating of mean. Other authors [13], [8], assume the mean vector is at the origin.

### A. Updating image representations

If we want to perform a true on-line incremental algorithm that does not require keeping the original images in the memory until the model has been constructed, we have to handle the image representations accordingly. Note that in the process of learning, after we update the subset of eigenvectors, we also have to update all of the representations of all the images according to the new basis set. This is due to the fact that these approximations are the only form in which we store the images throughout the process. Our contribution focuses on how to update and preserve these approximations.

During the process of learning at a discrete time $n$, we have learnt $n$ images $\mathbf{x}_i$, $i = 1 \dots n$, which has produced a space of $k$ eigenvectors $\mathbf{u}_j$, $j = 1 \dots k$. The images are presented with coefficient vectors $\mathbf{a}_{i(n)}$, $i = 1 \dots n$.

When a new observation $\mathbf{x}_{n+1}$ arrives, we compute the new mean $\bar{\mathbf{x}}'$ using (2), we construct the intermediate matrix $D$ (7) and solve the eigenproblem (6). Eq. (5) then produces an updated subspace base $U'$, but with no image representations.

In order to remap the coefficients $\mathbf{a}_{i(n)}$ into this new subspace, we first have to reconstruct each image using the old eigenvectors $U$:

$$\mathbf{x}_{i(n)} = U\mathbf{a}_{i(n)} + \bar{\mathbf{x}} \,, \;\; i = 1 \dots n \,, \tag{8}$$

and project it to the new subspace of eigenvectors:

$$\mathbf{a}_{i(n+1)} = (U')^\top(\mathbf{x}_{i(n)} - \bar{\mathbf{x}}') \,, \;\; i = 1 \dots n+1 \,, \tag{9}$$

where $\mathbf{x}_{n+1(n)} = \mathbf{x}_{n+1}$.

We can combine (5) and (8) into (9) and compute the new coefficients directly:

$$\mathbf{a}_{i(n+1)} = (R')^\top \begin{bmatrix} \mathbf{a}_{i(n)} \\ 0 \end{bmatrix} + \boldsymbol{\eta} \,, \tag{10}$$

where $\boldsymbol{\eta}$ is a vector computed only once for all coefficients as

$$\boldsymbol{\eta} = \left[ U \; \hat{\mathbf{h}}_{n+1} \right]^\top (\bar{\mathbf{x}} - \bar{\mathbf{x}}') \,. \tag{11}$$

The transformations described above yield a model that represents the input images with the same accuracy as the previous one, therefore we can now discard the old subspace and the coefficients that represented the images in it. $\mathbf{x}_{n+1}$ is represented accurately as well, so we can safely discard it. The representation of all $n+1$ images is possible because the subspace is spanned by $k+1$ eigenvectors, which is an *increase* of the model's dimensionality.

Since we would often like to keep the size of the representation low, we can decide to *keep* the dimensionality $k$. We do this by retaining the first $k$ eigenvectors of $U'$ (the eigenvectors are sorted by a decreasing order of the eigenvalues). This, however, results in a reduction of the accuracy of the representations. In order to balance the storage requirements with the level of accuracy, we need a criterion which reflects the cost of preservation of the dimensionality. Authors [11] have used values, such us a fraction of the smallest eigenvalue in the sum of all eigenvalues. However, since in our method each image influences the development of the model separately when it is added, we propose two additional criteria.

We decide to preserve the dimensionality if either

- the new observation can be represented with the current (old) set of eigenvectors satisfactorily, or
- the overall increase of error does not exceed an absolute threshold.

For the first criterion, we compute the size of the residual vector $\mathbf{h}_{n+1}$ when we project $\mathbf{x}_{n+1}$ into the subspace spanned by $U$. If the size of the residual exceeds an absolute threshold, we conclude the increase of the subspace is justified.

The second criterion is based on the sum of the reconstruction errors of the image representations. If this sum exceeds a threshold, this indicates that discarding the new dimension would significantly degrade the approximation of the images. To compute this, we would have to subtract each image $\mathbf{x}_{i(n)}$ from the approximation $\mathbf{x}_{i(n+1)}$ obtained by preserving the dimensionality, and sum the results for all $k$. But since the difference is only in the last eigenimage,

$$\mathbf{x}_{i(n)} = \sum_{l=1}^{k} \mathbf{u}_l a^l_{i(n)} = \sum_{l=1}^{k+1} \mathbf{u}'_l a^l_{i(n+1)} = \mathbf{x}_{i(n+1)} + \mathbf{u}'_{k+1} a^{k+1}_{i(n+1)} \tag{12}$$

it therefore holds

$$\Delta\mathbf{x}_i = \mathbf{x}_{i(n)} - \mathbf{x}_{i(n+1)} = \mathbf{u}'_{k+1} a^{k+1}_{i(n+1)} \ , \tag{13}$$

where $a^{k+1}_{i(n+1)}$ is the component $k+1$ of vector $\mathbf{a}_{i(n+1)}$. Since $|\mathbf{u}'_{k+1}| = 1$, then $|\Delta\mathbf{x}_i| = |a^{k+1}_{i(n+1)}|$. The sum $\sum_{i=1}^{n+1} |a^{k+1}_{i(n+1)}|$ yields the required criterion value.

The computation of this criterion value can be decreased even further. By definition, each eigenvalue $\lambda_j$ represents the variance of the $n$ coefficients in the direction of eigenvector $\mathbf{u}_j$:

$$\lambda_j = \frac{1}{n} \sum_{l=1}^{n} (a^j_l)^2 \ . \tag{14}$$

Hence, in our case we can use the value $(n+1)\lambda_{k+1(n+1)}$ as our criterion value.

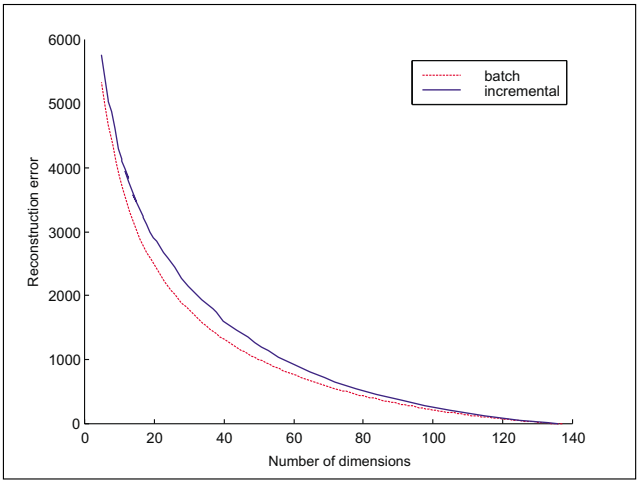


Fig. 3. Comparison of the level of representation for batch method and incremental method.

## IV. Experimental results

We performed two sets of experiments. First we tested the performance of the proposed method. We compared the results with those of the batch method. We investigated the effects of the model updating on the representations of the images in the model and analyzed the effect of repetitive learning.

The second set of experiments shows the performance of the method in the real-world application, where we acquire images at known locations and localize those taken at unknown locations.

### *A. Performance of the incremental method*

We captured the input images for this set of experiments with a Magellan Pro mobile robot equipped with an omnidirectional sensor. The set contained 150 images. Each image was oriented in the same direction.

A useful measure for the performance of the PCA is the compression ratio with respect to the reconstruction error. Fig. 3 shows the comparison between the overall reconstruction error for the batch method and the incremental method. In the experiment we used 135 input images. Using the batch method, we did a single run of batch PCA on a set of input images to compute all eigenvectors. Then we discarded the least significant eigenvectors one by one. At each iteration, we computed the overall reconstruction error.

When computing the same measure for the incremental method, we forced the method to produce a certain number of resulting eigenvectors. Initially, we set the threshold to a low value such that the output model had the same number of eigenvectors as the number of input images. As a higher threshold value means less eigenvectors, we gradually raised the

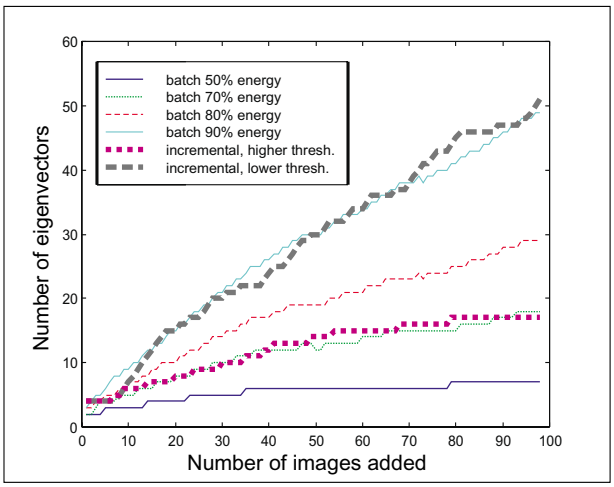


Fig. 4. Comparison of batch method and incremental method by the number of eigenvectors. The curves for the batch method represent the number of eigenvectors containing a certain percentage of the overall eigenvalue energy.

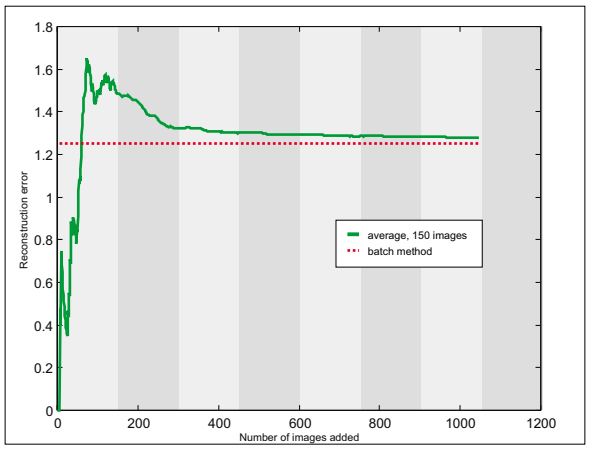


Fig. 5. Average reconstruction error as we repeatedly update the model with the same sequence of 150 images. The boundary denoted as 'batch method' represents the reconstruction error of the batch method where the number of eigenvectors is the same as the number of eigenvectors produced by the incremental method. The shaded regions depict the repeating sequences.

threshold, running the method each time. After each run we computed the reconstruction error.

The results indicate that the reconstruction error is slightly higher for the incremental method than for the batch method. In our case, the error was higher by 10% on the average.

Another way of comparing the incremental method with the batch method is by observing the distribution of the energy of eigenspectrum (which reflects the reconstruction error). If we use the 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 eigenspectrum energy. By running the incremental method, we can observe how the number of preserved eigenvectors changes according to the criteria as we add new data vectors.

Fig. 4 shows two superimposed charts, one for the batch method and the other one for the incremental method. The same images have been used in both experiments. In the case of the incremental method, two runs were carried out, differing only in the value of the threshold. We can see how the results fit: in the case of a lower threshold, the growth of the number of eigenvectors was very similar to the growth of the same value for the batch method, where 90% of the energy was retained. In the second run of the incremental method we raised the threshold, and the curve became comparable to the one for batch method with 70% of eigenvalue energy retained.

### *B. Repetitive learning*

In practice, when we explore the environment and learn the model, we can come across the same observation multiple times. Therefore, we experimented with the effect of revising the observations. We took a sequence of 150 images and used it multiple times in a row as the input of our method. We thus simulated the robot repeatedly exploring on the same path. We observed the average reconstruction error of the observations.

Fig. 5 shows the result of the experiment. Each shaded area of the graph represents one run of the sequence. As the images are introduced for the first time (image numbers 1 through 150), we can see how the average reconstruction error makes several rises and falls.

When we update the model with the same images for the second time (image numbers 151 through 300), keeping only the representations of the new images, we can see that the reconstruction error decreases. In the following sequences, the reconstruction error decreases steadily. We can see that it converges towards the reconstruction error obtained by the batch method at the same level of compression. These results suggest that the model is improving, if it gets updated with the same set of images.

### *C. Localization*

A true test for the algorithm is solving a task of localization. Fig. 6 shows a setup and results of such an experiment. Inside a laboratory, we used a sparse, equally spaced grid of 62 locations separated by 60 cm for a set of training images. We used another collection of 100 test images taken at locations surrounded by at least four neighboring locations previously used for training. We ran the updating algorithm which pro-

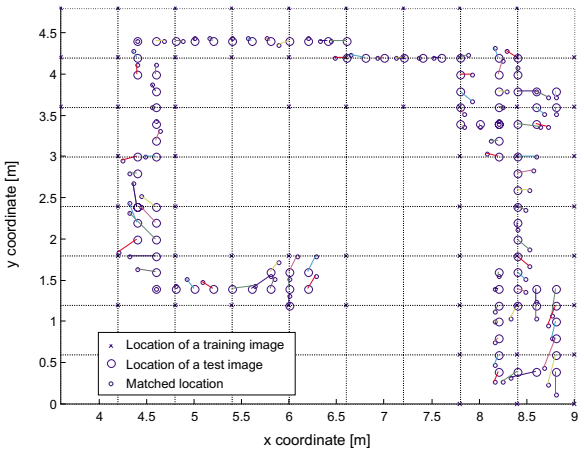


Fig. 6. Localization with incremental PCA. The the circles and x marks on the map denote the locations where images have been taken. The lines connect each test image location with the training image location found by the algorithm.

duced a subspace spanned by 14 eigenvectors, which is 23% of the number of training images. We then interpolated the coefficients in the subspace in order to get an approximation of the environment sampled 4 cm apart in each direction. Test images were then projected into this subspace, and they were matched with the closest interpolated coefficient vector.

Table I shows the results of the localization for both incremental and batch method with 14 eigenvectors. The error is computed as a distance in cm between the closest training position and the matched one. We can see the results are below the sampling distance of 60 cm, and they are comparable for the two methods.

TABLE I

Comparison of localization results

| | Incremental | Batch |
|---|---|---|
| Average error | 14 cm | 13 cm |
| Peak error | 42 cm | 51 cm |
| Cases under 25 cm | 86% | 91% |

## V. Conclusion

In this paper we discussed the problem of mobile robot self-localization using appearance-based recognition of panoramic images. We use PCA in order to represent images in a low-dimensional subspace. As the robot should be capable of incrementally learning and updating its model of the world, we adopt the incremental computation of PCA. Previous research addressed the on-line construction of eigenvectors only, while representations of the images were computed only after the final eigenspace was built. This requires keeping all the images until the final step of the computation.

We therefore proposed to use an incremental PCA algorithm with the updating of partial image representations in a way that allows discarding the acquired images immediately after the update. The results of our tests indicate that the proposed method is comparable to the batch method in terms of compression, computational cost, and the precision of localization. We also tested how our algorithm performs when images are learnt repetitively, e.g., when the robot comes across the same positions multiple times. We show that in this case the properties of the incrementally built subspace converge to that constructed with the batch method.

Currently, we are exploring various advantages of the on-line learning to the mobile robot localization, e.g. the capability of actively selecting the next learning position or the strategies for on-line building of multiple subspaces.

## References

[1] S. K. Nayar, S. A. Nene, and H. Murase, "Subspace methods for robot vision," *IEEE Trans. on Robotics and Automation*, vol. 12, pp. 750–758, October 1996.

[2] A. Leonardis and H. Bischof, "Robust recognition using eigenimages," *CVIU*, vol. 78, pp. 99–118, April 2000.

[3] M. Jogan and A. Leonardis, "Robust Localization using Panoramic View-Based Recognition," in *15th ICPR*, vol. 4, pp. 136–139, IEEE Computer Society, September 2000.

[4] J. Gaspar, N. Winters, and J. Santos-Victor, "Vision-based navigation and environmental representations with an omnidirectional camera," *IEEE Trans. on Robotics and Automation*, vol. 16, pp. 890–898, December 2000.

[5] H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura, "Memory-based self-localisation using omnidirectional images," in *14th ICPR*, pp. 297–299, August 1998.

[6] N. Vlassis, R. Bunschoten, and B. Kröse, "Learning task–relevant features from robot data," in *IEEE Int. Conf. on Robotics and Automation* , Seoul, Korea, pp. 499–504, 2001.

[7] A. J. Davison and N. Kita, "Sequential localisation and map-building for real-time computer vision and robotics," *RAS*, vol. 36, no. 4, pp. 171–183, 2001.

[8] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang, "An eigenspace update algorithm for image analysis," *Graphical Models and Image Processing*, vol. 59, pp. 321–332, Sept. 1997.

[9] J. Winkeler, B. S. Manjunath, and S. Chandrasekaran, "Subset selection for active object recognition," in *CVPR*, vol. 2, pp. 511–516, June 1999.

[10] M. Gu and S. C. Eisenstat, "A stable and fast algorithm for updating the singular value decomposition," Tech. Rep. YALEU/DCS/RR-966, New Haven, CT, 1993.

[11] P. Hall, D. Marshall, and R. Martin, "Incremental eigenalysis for classification," in *British Machine Vision Conference*, vol. 1, pp. 286–295, Sept. 1998.

[12] P. Hall, D. Marshall, and R. Martin, "Merging and spliting eigenspace models," *PAMI*, vol. 22, no. 9, pp. 1042–1048, 2000.

[13] A. Levy and M. Lindenbaum, "Sequential Karhunen-Loeve basis extraction and its application to images," *IEEE Trans. on Image Processing*, vol. 9, pp. 1371–1374, 2000.