Incremental Principal Component Analysis

Danijel Skočaj



University of Ljubljana

Faculty of Computer and Information Science

Udine; 5, 12 December 2014

Batch method for PCA

- All input images are processed simultaneously.

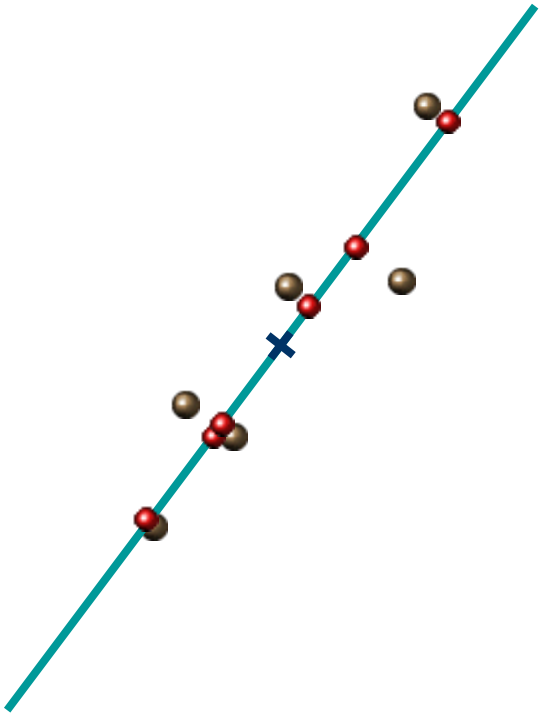
$$\boldsymbol{\mu} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j$$

$$\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$$

$$\mathbf{C} = \tfrac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top$$

$$\mathrm{SVD}(\mathbf{C}) \rightarrow \mathbf{U}, \boldsymbol{\Lambda}$$

$$\mathbf{A} = \mathbf{U}^\top\hat{\mathbf{X}}$$

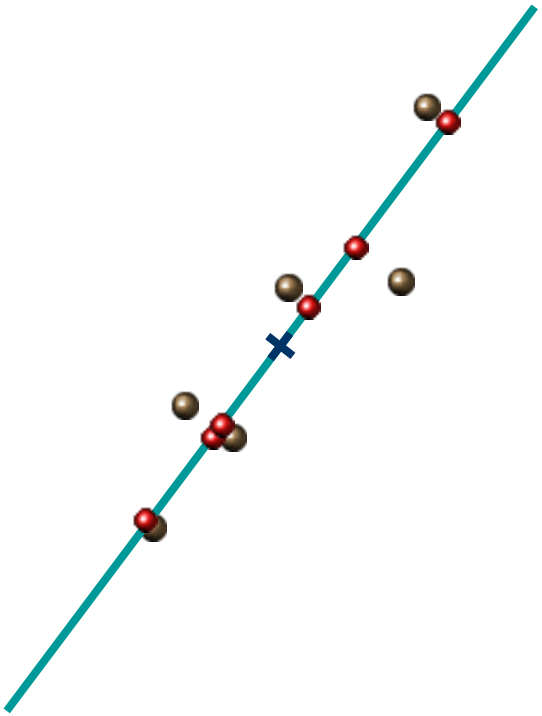


Batch method – pros and cons

- all training images processed simultaneously
+ simple
+ optimal (in the sense of MSRE)
+ easy to choose subsace dimension
– all training images have to be given in advance
– computationally unfeasible for large sets of training images
– model built once forever

Incremental approach

- Input images are being processed sequentaly.



Incremental approach – main idea

- At step *n+1* calculate a new eigenspace from the reconstructions of the first *n* training images and a new image using the standard batch method in a high-dimensional space.

reconstructions = coefficient vectors

- At step *n+1* calculate a new eigenspace from the coefficient vectors of the first *n* training images and a new image using the standard batch method in a low-dimensional space.

Incremental method for PCA

$$\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$$

$$\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$$

$$\mathbf{r} = \mathbf{x} - \mathbf{y}$$

$$\mathbf{U}' = \left[\mathbf{U}^{(n)} \quad \frac{\mathbf{r}}{\|\mathbf{r}\|} \right]$$

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$$

$$\mathrm{PCA}(\mathbf{A}') \rightarrow \boldsymbol{\mu}'' \; \mathbf{U}'', \boldsymbol{\lambda}''$$

$$\mathbf{U}'' = [\mathbf{u}''_1, \ldots, \mathbf{u}''_{k(n+1)}]$$

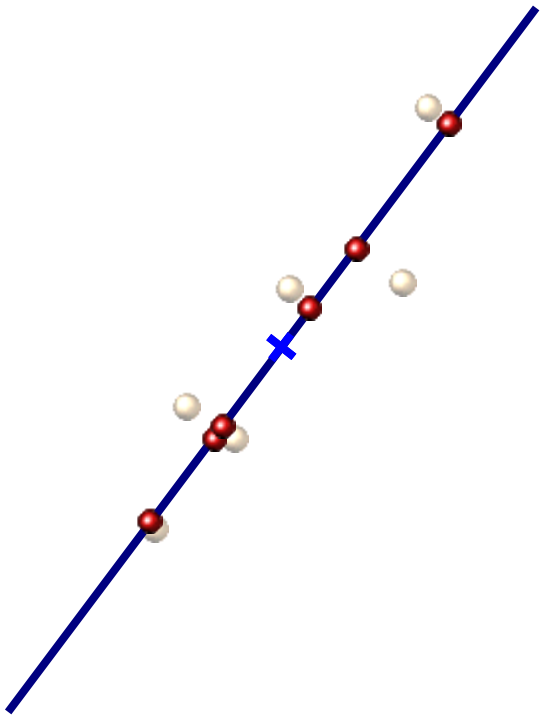
$$\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$$

$$\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$$

$$\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$$

$$\boldsymbol{\lambda}^{(n+1)} = [\lambda''_1, \ldots, \lambda''_{k(n+1)}]$$

$\boldsymbol{\mu}^{(n)}$ $\mathbf{U}^{(n)}$ $\mathbf{A}^{(n)}$ $\mathbf{x}$



$\boldsymbol{\mu}^{(n+1)}$ $\mathbf{U}^{(n+1)}$ $\mathbf{A}^{(n+1)}$ $\boldsymbol{\lambda}^{(n+1)}$

Algorithm for one update

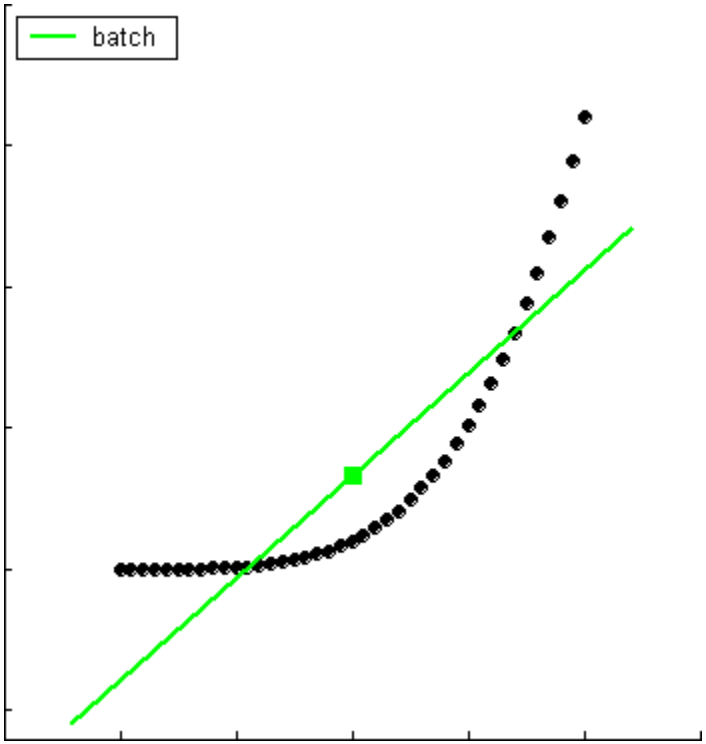
Input: current mean value $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$.

Output: new mean value $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$, new eigenvalues $\boldsymbol{\lambda}^{(n+1)}$.

1. Project new image $\mathbf{x}$ in current ES: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$.
2. Reconstruct new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
3. Compute residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
4. Append $\mathbf{r}$ as new basis vector: $\mathbf{U}' = \left[\begin{array}{cc} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{array} \right]$
5. Determine coefficients in new basis: $\mathbf{A}' = \left[\begin{array}{cc} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{array} \right]$.
6. Perform PCA on $\mathbf{A}'$. Obtain $\boldsymbol{\mu}'', \mathbf{U}'', \boldsymbol{\lambda}''$.
7. Project coeff. vec. to new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$.
8. Rotate subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$.
9. Update mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$.
10. New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$.

Incremental PCA - example

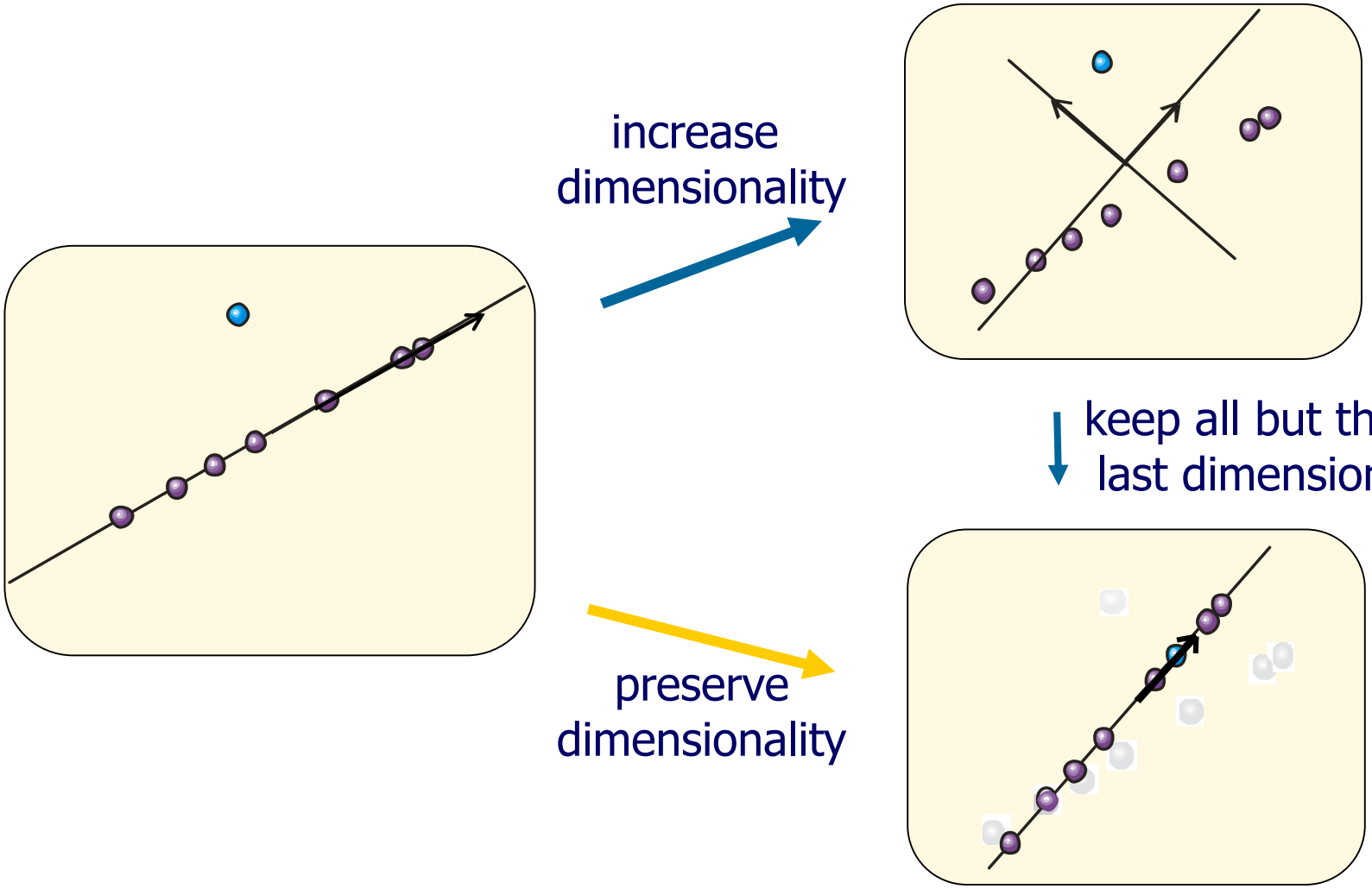
- Simple 2D example



Incremental approach – pros and cons

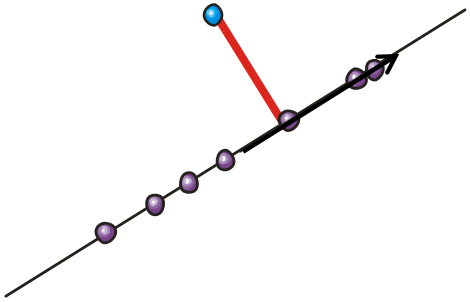
- training images processed sequentally
+ training images can be given at different time instances
+ very large training sets can be processed
+ model can be updated
+ simple
– non-optimal (in the sense of MSRE)
– non-trivial way of choosing subspace dimension

Icreasing/preserving dimensionality

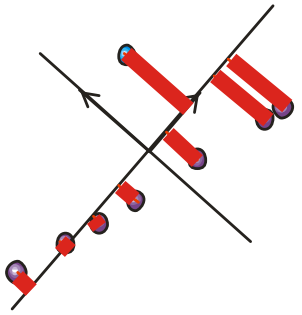


Criteria for discarding eigenvectors

- Keep a **stipulated number** of eigenvectors.
- Increase dimensionality if **the last image** added is significantly different from the ones already in the representation (*residual*).



- Increase dimensionality if the accuracy of representation of **all images** degrades too much; *threshold value:* $(n+1)\lambda_{k+1(n+1)}$



Experiments and discussion

- How much does the incremental method degrade the results in comparison with the batch method?
- How does discarding of training images influence the results?
- How does the order of training images influence the results?
- How does the criterion for increasing/ preserving the dimension of the eigenspace influence the results?

Experiments

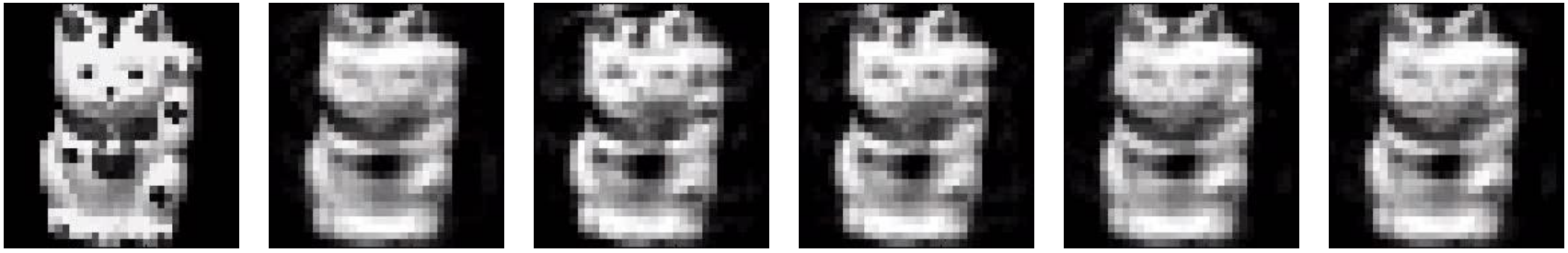
- Training images: 720 images of 20 objects from COIL20 database.



- Comparison of squared reconstruction error.

Experimental results

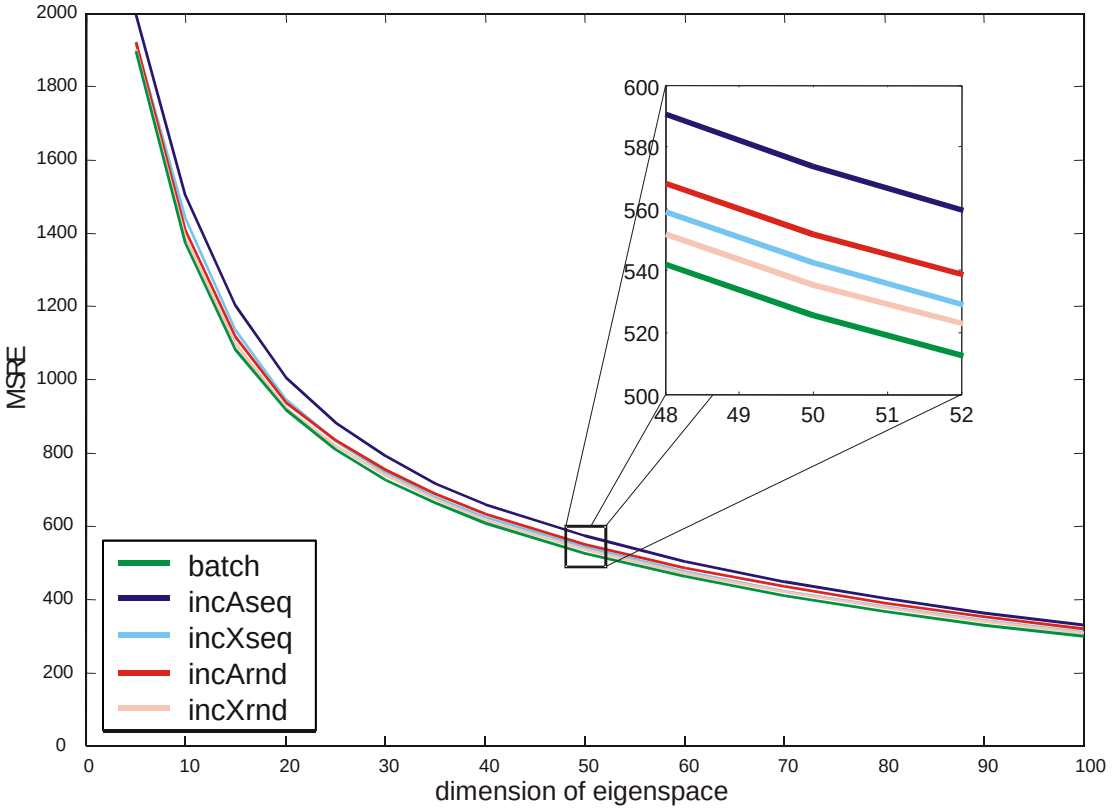
- Approaches tested:
  - batch: batch method.
  - incA: incremental method, discarding images.
  - incX: incremental method, preserving images.
  - seq: sequential order of training images.
  - rnd: random order of training images.
- Very similar results.



training im. batch incAseq incXseq incArnd incXrnd

Experimental results

MSRE :



average degradation of results :

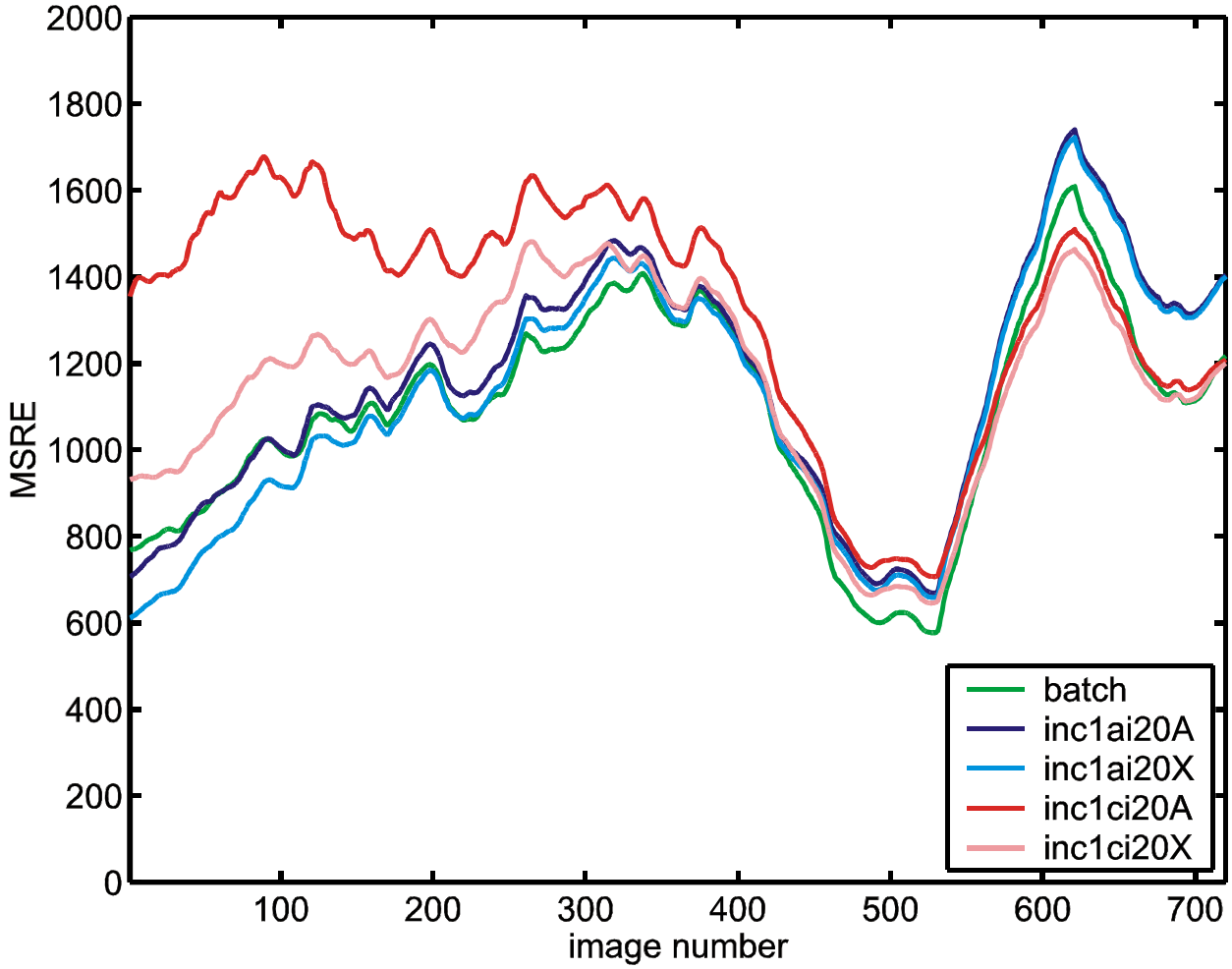
batch	incAseq	incXseq	incArnd	incXrnd
0%	8.6%	3.1%	3.1%	1.3%

Experiments – discarding criterion

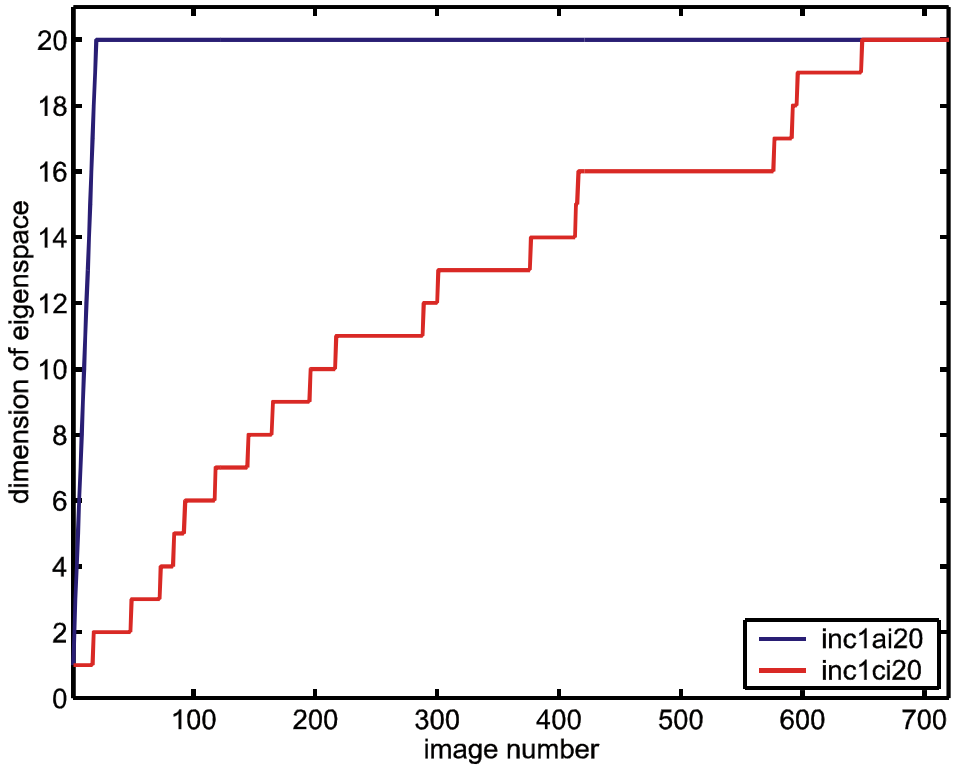
- Degradation of the results for K=20:

batch	0%	Batch on all images. Keep 20 dim.
inc20ni20A	6.67%	Batch on first 20 images then incremental on the rest withouth increasing dimension.
inc20ni20X	2.87%	
inc1ai20A	6.67%	Start with 1 image, add next 19 withouth discarding, then add the rest withouth increasing dim.
inc1ai20X	2.87%	
inc1ci20A	22.61%	Start with 1 image and add the rest according to discarding criterion to achieve final dimension 20.
inc1ci20X	6.76%	

Experiments – discarding criterion



Experiments – discarding criterion

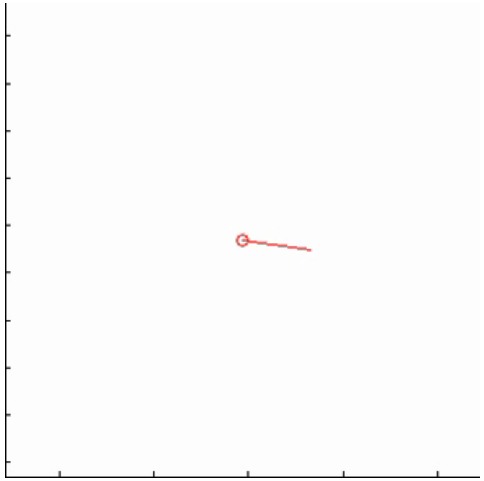


- Discarding criterion should be:
  - less conservative at the beggining
  - more conservative afterwards.

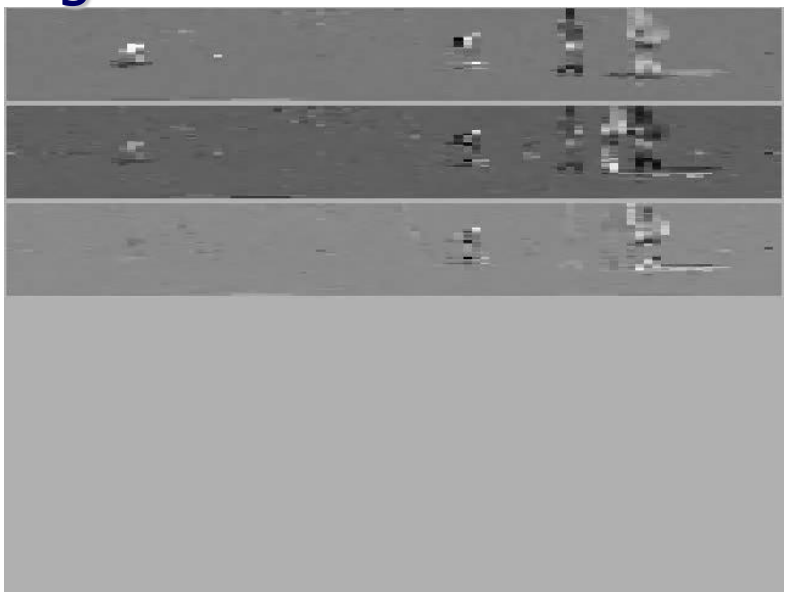
Learning a map of the environment



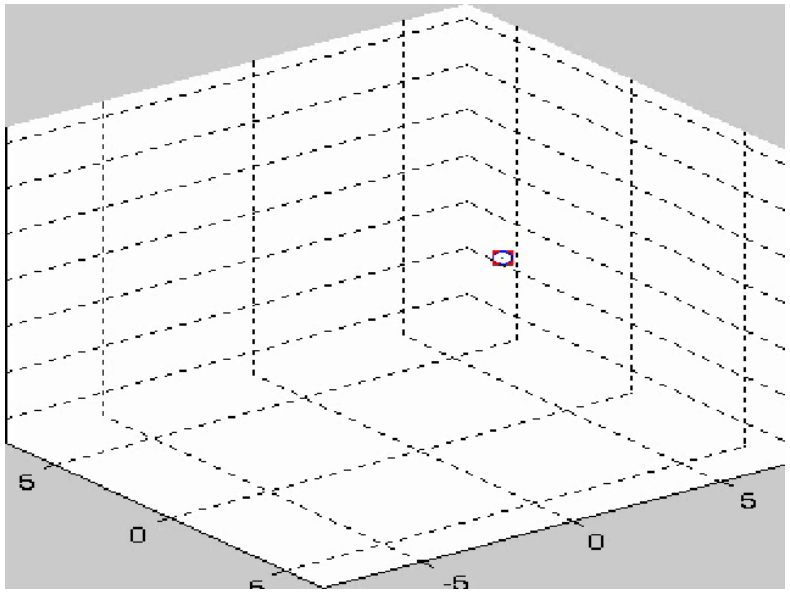
GPS coordinates



First six eigenvectors



First three coefficients



Conclusions

- Incremental method does not significantly degrade the results of the batch method.
- If the training images are discarded after the update, the results are degraded, but still good.
- Better results are achieved, if heterogenuous images are present in the beggining of the training sequence.
- Discarding criterion can significantly influence the results.

Literature

- D. Skočaj and A. Leonardis, *Incremental and robust learning of subspace representations*, Image and Vision Computing, 2008
  - http://vicos.fri.uni-lj.si/data/publications/skocajIMAVIS08.pdf

- Danijel Skočaj, *Robust subspace approaches to visual learning and recognition*, PhD thesis, University of Ljubljana, 2003
  - http://vicos.fri.uni-lj.si/data/publications/skocajPhD03.pdf
- C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006