computer graphics & vision
Inffeldgasse 16/2, A-8010 Graz
www.icg.tu-graz.ac.at
TUG

# SURVEY OF APPEARANCE-BASED METHODS FOR OBJECT RECOGNITION

Peter M. Roth and Martin Winter

*Inst. for Computer Graphics and Vision*
*Graz University of Technology, Austria*

**Technical Report**
***ICG–TR–01/08***
**Graz, January 15, 2008**

contact: {pmroth,winter}@icg.tugraz.at

# Abstract

*In this survey we give a short introduction into appearance-based object recognition. In general, one distinguishes between two different strategies, namely local and global approaches. Local approaches search for salient regions characterized by e.g. corners, edges, or entropy. In a later stage, these regions are characterized by a proper descriptor. For object recognition purposes the thus obtained local representations of test images are compared to the representations of previously learned training images. In contrast to that, global approaches model the information of a whole image. In this report we give an overview of well known and widely used region of interest detectors and descriptors (i.e, local approaches) as well as of the most important subspace methods (i.e., global approaches). Note, that the discussion is reduced to methods, that use only the gray-value information of an image.*

**Keywords:** *Difference of Gaussian (DoG), Gradient Location-Orientation Histogram (GLOH), Harris corner detector, Hessian matrix detector, Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), Locally Binary Patterns (LBP), local descriptors, local detectors, Maximally Stable Extremal Regions (MSER), Non-negative Matrix Factorization (NMF), Principal Component Analysis (PCA), Scale Invariant Feature Transform (SIFT), shape context, spin images, steerable filters, subspace methods.*

# Annotation

This report is mainly based on the authors' PhD theses, i.e., Chapter 2 of [135] and Chapter 2 and Appendix A-C of [105].

# 1 Introduction

When computing a classifier for object recognition one faces two main philosophies: generative and discriminative models. Formally, the two categories can be described as follows: Given an input $x$ and and a label $y$ then a generative classifier learns a model of the joint probability $p(x, y)$ and classifies using $p(y|x)$, which is obtained by using Bayes' rule. In contrast, a discriminative classifier models the posterior $p(y|x)$ directly from the data or learns a map from input to labels: $y = f(x)$.

Generative models such as principal component analysis (PCA) [57], independent component analysis (ICA) [53] or non-negative matrix factorization (NMF) [73] try to find a suitable representation of the original data (by approximating the original data by keeping as much information as possible). In contrast, discriminant classifiers such as linear discriminant analysis (LDA) [26], support vector machines (SVM) [133], or boosting [33] where designed for classification tasks. Given the training data and the corresponding labels the goal is to find optimal decision boundaries. Thus, to classify an unknown sample using a discriminative model a label is assigned directly based on the estimated decision boundary. In contrast, for a generative model the likelihood of the sample is estimated and the sample is assigned the most likely class.

In this report we focus on generative methods, i.e., the goal is to represent the image data in a suitable way. Therefore, objects can be described by different cues. These include *model-based* approaches (e.g., [11, 12, 124]), shape-based approaches (e.g., ), and appearance-based models. Model-based approaches try to represent (approximate) the object as a collection of three dimensional, geometrical primitives (boxes, spheres, cones, cylinders, generalized cylinders, surface of revolution) whereas shape-based methods represent an object by its shape/contour. In contrast, for appearance-based models only the *appearance* is used, which is usually captured by different two-dimensional views of the object-of-interest. Based on the applied features these methods can be sub-divided into two main classes, i.e., *local* and *global* approaches.

A *local feature* is a property of an image (object) located on a single point or small region. It is a single piece of information describing a rather simple, but ideally distinctive property of the object's projection to the camera (image of the object). Examples for local features of an object are, e.g., the color, (mean) gradient or (mean) gray value of a pixel or small region. For object recognition tasks the local feature should be invariant to illumination changes, noise, scale changes and changes in viewing direction, but, in general, this cannot be reached due to the simpleness of the features itself. Thus, several features of a single point or *distinguished region* in various forms are combined and a more complex description of the image usually referred to as *descriptor* is obtained. A *distinguished* region is a connected part of an image showing a *significant* and *interesting* image property. It is usually determined by the application of an *region of interest detector* to the image.

In contrast, *global* features try to cover the information content of the whole image or patch, i.e., all pixels are regarded. This varies from simple statistical measures (e.g., mean values or histograms of features) to more sophisticated dimensionality reduction techniques, i.e., subspace methods, such as principle component analysis (PCA) [57], independent component analysis (ICA) [53], or non negative matrix factorization (NMF) [73]. The main idea of all of these methods is to project the original data onto a subspace, that represents the data optimally according to a predefined criterion: minimized variance (PCA), independency of the data (ICA), or non-negative, i.e., additive, components (NMF).

Since the whole data is represented global methods allow to reconstruct the original image and thus provide, in contrast to local approaches, robustness to some extend. Contrary, due to the local representation local methods can cope with partly occluded objects considerable considerably better.

Most of the methods discussed in this report are available in the Image Description ToolBox (IDTB)[1], that was developed at the Inst. for Computer Graphics and Vision in 2004–2007. The corresponding sections are marked with a star $\star$.

The report is organized as follows: First, in Section 2 we give an overview of local region of interest detectors. Next, in section 3 we summarize common and widely used local region of interest descriptors. In Section 4, we discuss subspace methods, which can be considered global object recognition approaches. Finally, in the Appendix we summarize the necessary basic mathematics such as elementary statistics and Singular Value Decomposition.

# 2 Region of Interest Detectors

As most of the *local appearance based* object recognition systems work on distinguished regions in the image, it is of great importance to find such regions in a highly repetitive manner. If a region detector returns only an exact position within the image we also refer to it as *interest point* detector (we can treat a point as a special case of a region). Ideal region detectors deliver additionally shape (scale) and orientation of a region of interest. The

[1] http://www.icg.tugraz.at/research/ComputerVision/IDTB_data, December 13, 2007

currently most popular distinguished region detectors can be roughly divided into three broad categories:

- corner based detectors,
- region based detectors, and
- other approaches.

*Corner based detectors* locate points of interest and regions which contain a lot of image structure (e.g., edges), but they are not suited for uniform regions and regions with smooth transitions. *Region based detectors* regard local blobs of uniform brightness as the most salient aspects of an image and are therefore more suited for the latter. *Other approaches* for example take into account the entropy of a region (Entropy Based Salient Regions) or try to imitate the human's way of visual attention (e.g., [54]).

In the following the most popular algorithms, which give sufficient performance results as was shown in , e.g., [31, 88–91, 110], are listed:

- Harris- or Hessian point based detectors (*Harris*, *Harris-Laplace*, *Hessian-Laplace*) [27, 43, 86],
- Difference of Gaussian Points (DoG) detector [81],
- Harris- or Hessian affine invariant region detectors (*Harris-Affine*) [87],
- Maximally Stable Extremal Regions (MSER) [82],
- Entropy Based Salient Region detector (EBSR) [60–63], and
- Intensity Based Regions and Edge Based Regions (IBR, EBR) [128–130].

## 2.1 Harris Corner-based Detectors$^\star$

The most popular region of interest detector is the corner based one of Harris and Stephens [43]. It is based on the second moment matrix

$$\boldsymbol{\mu} = \begin{bmatrix} I_x^2(\mathbf{p}) & I_x I_y(\mathbf{p}) \\ I_x I_y(\mathbf{p}) & I_y^2(\mathbf{p}) \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix} \tag{1}$$

and responds to corner-like features. $I_x$ and $I_y$ denote the first derivatives of the image intensity $I$ at position $\mathbf{p}$ in the x and y direction respectively. The corner response or *cornerness measure* $c$ is efficiently calculated by avoiding the eigenvalue decomposition of the second moment matrix by

$$c = Det(\boldsymbol{\mu}) - k \ \times \ Tr(\boldsymbol{\mu})^2 = (AC - B^2) - k \ \times \ (A + C)^2. \qquad (2)$$

This is followed by a non-maximum suppression step and a Harris-corner is identified by a high positive response of the *cornerness* function $c$. The Harris-point detector delivers a large number of interest-points with sufficient repeatability as shown , e.g., by Schmid et al. [110]. The main advantage of this detector is the speed of calculation. A disadvantage is the fact, that the detector determines only the spatial locations of the interest points. No *region of interest properties* such as scale or orientation are determined for the consecutive descriptor calculation. The detector shows only rotational invariance properties.

## 2.2 Hessian Matrix-based Detectors$^\star$

Hessian matrix detectors are based on a similar idea like Harris-detectors. They are in principle based on the Hessian-matrix defined in (3) and give strong responses on blobs and ridges because of the second derivatives used [91]:

$$\mathbf{M}_h = \begin{bmatrix} I_{xx}(\mathbf{p}) & I_{xy}(\mathbf{p}) \\ I_{xy}(\mathbf{p}) & I_{yy}(\mathbf{p}) \end{bmatrix}, \qquad (3)$$

where $I_{xx}$ and $I_{yy}$ are the second derivatives of the image intensity $I$ at position $\mathbf{p}$ in the x and y direction respectively and $I_{xy}$ is the mixed derivative in x and y direction of the image.

The selection criterion for Hessian-points is based on the determinant of the Hessian-matrix after non-maximum suppression. The Hessian-matrix based detectors detect blob-like structures similar to the Laplacian operator and shows also only rotational invariance properties.

## 2.3 Scale Adaptations of Harris and Hessian Detectors$^\star$

The idea of selecting a *characteristic scale* disburdens the above mentioned detectors from the lack in scale invariance. The properties of the scale space have been intensely studied by Lindeberg in [78]. Based on his work on *scale space blobs* the local extremum of the scale normalized Laplacian $S$ (see (4)) is used as a scale selection criterion by different methods (e.g., [86]). Consequently in the literature they are often referred as *Harris-Laplace* or *Hessian-Laplace* detectors. The standard deviation of Gaussian smoothing for scale space generation (often also termed *local scale*) is denoted by $s$:

$$S = s^2 \times |(I_{xx}(\mathbf{p}) + I_{yy}(\mathbf{p}))| \qquad (4)$$

The Harris- and Hessian-Laplace detectors show the same properties as their *plain* pendants, but, additionally, they have scale invariance properties.

## 2.4 Difference of Gaussian (DoG) Detector$^\star$

A similar idea is used by David Lowe in his *Difference of Gaussian* detector (DoG) [80,81]. Instead of the scale normalized Laplacian he uses an approximation of the Laplacian, namely the Difference of Gaussian function $D$, by calculating differences of Gaussian blurred images at several, adjacent local scales $s_n$ and $s_{n+1}$:

$$D(\mathbf{p}, s_n) = (G(\mathbf{p}, s_n) - G(\mathbf{p}, s_{n+1})) * I(\mathbf{p}) \tag{5}$$

$$G(\mathbf{p}, s_n) = G((x, y), s_n) = \frac{1}{2\pi s^2} e^{-(x^2+y^2)/2s^2} \tag{6}$$

In (5) $G$ is the variable-scaled Gaussian of scale $s$ (see also (6)), $I$ is the image intensity at $x, y$-position $\mathbf{p}$ and, $*$ denotes the convolution operation. The Difference of Gaussians can be calculated in a pyramid much faster then the Laplacian scale space and show comparable results. The principle for scale selection is nearly the same as for the Harris-Laplace detector. An accurate key point localization procedure, elimination of edge responses by a Hessian-matrix based analysis and orientation assignment with orientation histograms completes the carefully designed detector algorithm. The Difference of Gaussians (DoG) detector shows similar behavior like the Hessian-detector and therefore detects blob-like structures. The main advantage of the DoG detector is the obtained scale invariance property. Obviously this is penalized by the necessary effort in time.

## 2.5 Affine Adaptations of Harris and Hessian Detectors$^\star$

Recently, Mikolajczyk and Schmid [87] proposed an extension of the scale adapted Harris and Hessian detector to obtain invariance against affine transformed images. Scientific literature refers to them as *Harris-Affine* or *Hessian-Affine* detectors depending on the initialization points used. The affine adaptation is based on the shape estimation properties of the second moment matrix. The simultaneous optimization of all three affine parameters spatial point location, scale, and shape is too complex to be practically useful. Thus, an iterative approximation of these parameters is suggested.

Shape adaptation is based on the assumption, that the local neighborhood of each interest point $\mathbf{x}$ in an image is an affine transformed, isotropic patch around a normalized interest point $\mathbf{x}^*$. By estimating the affine parameters represented by the transformation matrix $\mathbf{U}$, it is possible to transform the local neighborhood of an interest point $\mathbf{x}$ back to a normalized, isotropic structure $\mathbf{x}^*$:

$$\mathbf{x}^* = \mathbf{U}\mathbf{x}\ . \tag{7}$$

The obtained affine invariant region of interest (*Harris-Affine* or *Hessian-Affine* region) is represented by the local, anisotropic structure normalized into the isotropic patch. Usually, the estimated shape is pictured by an ellipse, where the ratio of the main axes is proportional to the ratio between the eigenvalues of the transformation matrix.

As Baumberg has shown in [6] that the anisotropic local image structure can be estimated by the inverse matrix square root of the second moment matrix $\boldsymbol{\mu}$ calculated from the isotropic structure (see (1)), (7) changes to

$$\mathbf{x}^* = \boldsymbol{\mu}^{-\frac{1}{2}}\mathbf{x}. \tag{8}$$

Mikolajczyk and Schmid [87] consequently use the concatenation of iteratively optimized second moment matrices $\boldsymbol{\mu}^{(k)}$ in step $k$ of the algorithm, to successively refine the initially unknown transformation matrix $\mathbf{U}^{(0)}$ towards an optimal solution:

$$\mathbf{U}^{(k)} = \prod_k \boldsymbol{\mu}^{(-\frac{1}{2})(k)}\mathbf{U}^{(0)}\ . \tag{9}$$

In particular, their algorithm is initialized by a scale adapted Harris or Hessian detector to provide an approximate point localization $\mathbf{x}^{(0)}$ and initial scale $s^{(0)}$. The actual iteration loop (round $k$) consists of the following four main steps:

1. Normalization of the neighborhood around $\mathbf{x}^{(k-1)}$ in the image domain by the transformation matrix $\mathbf{U}^{(k-1)}$ and scale $s^{(k-1)}$.

2. Determination of the actual characteristic scale $s^{*(k)}$ in the normalized patch.

3. Update of the spatial point location $\mathbf{x}^{*(k)}$ and estimation of the actual second moment matrix $\boldsymbol{\mu}^{(k)}$ in the normalized patch window.

4. Calculation of the transformation matrix $\mathbf{U}$ according to (9).

The update of the scale in step 2 is necessary, because it is a well known problem, that in the case of affine transformations the scale changes are in general not the same in all directions. Thus, the scale detected in the image domain can be very different from that in the normalized image. As the affine normalization of a point neighborhood also slightly changes the local spatial maxima of the Harris measure, an update and back-transformation of the location $\mathbf{x}^*$ to the location in the original image domain $\mathbf{x}$ is also essential (step 3).

The termination criterion for the iteration loop is determined by reaching a perfect isotropic structure in the normalized patch. The measure for the amount of isotropy is estimated by the ratio $Q$ between the two eigenvalues $(\lambda_{max}, \lambda_{min})$ of the $\boldsymbol{\mu}$-matrix. It is exactly 1 for a perfect isotropic structure, but in practise, the authors allow for a small error $\epsilon$:

$$Q = \frac{\lambda_{max}}{\lambda_{min}} \leq (1 + \epsilon) \ . \tag{10}$$

Nevertheless, the main disadvantage of affine adaptation algorithms is the increase in runtime due to their iterative nature, but as shown in , e.g., [91] the performance of those shape-adapted algorithms is really excellent.

## 2.6 Maximally Stable Extremal Regions$^\star$

*Maximally Stable Extremal Regions* [82] is a watershed-like algorithm based on intensity value - connected component analysis of an appropriately thresholded image. The obtained regions are of arbitrary shape and they are defined by all the border pixels enclosing a region, where all the intensity values within the region are consistently lower or higher with respect to the surrounding.

The algorithmic principle can be easily understood in terms of thresholding. Consider all possible binary thresholdings of a gray-level image. All the pixels with an intensity below the threshold are set to 0 (black), while all the other pixels are set to 1 (white). If we imagine a movie showing all the binary images with increasing thresholds, we would initially see a totally white image. As the threshold gets higher, black pixels and regions corresponding to local intensity minima will appear and grow continuously. Sometimes certain regions do not change their shape even for set of different consecutive thresholds. These are the *Maximally Stable Extremal Regions* detected by the algorithm. In a later stage, the regions may merge and form larger clusters, which can also show stability for certain thresholds. Thus, it is possible that the obtained MSERs are sometimes nested. A second set of regions could be obtained by inverting the intensity of the source image and following the same process. The algorithm can be implemented very efficiently with respect to runtime. For more details about the implementation we refer to the original publication in [82].

The main advantage of this detector is the fact, that the obtained regions are robust against continuous (an thus even projective) transformations and even non-linear, but monotonic photometric changes. In the case a single interest point is needed, it is usual to calculate the center of gravity and take this as an anchor point , e.g., for obtaining reliable point correspondences. In contrast to the detectors mentioned before, the number of regions detected is rather small, but the repeatability outperforms the other detectors in most cases [91]. Furthermore, we mention that it is possible to define MSERs also on even multi-dimensional images, if the pixel values show an ordering.

## 2.7 Entropy Based Salient Region detector$^\star$

Kadir and Brady developed a detector based on the grey value entropy

$$H_F(s,x) = -\int p(f,s,\mathbf{x}) \times \log_2(p(f,s,\mathbf{x}))df \tag{11}$$

of a circular region in the image [61,62] in order to estimate the *visual saliency* of a region.

The probability density function for the entropy $p$ is estimated by the grey value histogram values ($f$, the features) of the patch for a given scale $s$ and location $\mathbf{x}$. The *characteristic scale* $S$ is select by the local maximum of the entropy function ($H_F$) by

$$S = \left\{ s | \frac{\delta}{\delta s} H_F(s,\mathbf{x}) = 0, \frac{\delta^2}{\delta s^2} H_F(s,\mathbf{x}) \prec 0 \right\}. \tag{12}$$

In order to avoid self similarity of obtained regions, the entropy function is weighted by a *self similarity factor* $W_F(s,\mathbf{x})$, which could be estimated by the absolute difference of the probability density function for neighboring scales:

$$W_F(s,\mathbf{x}) = s \int \left| \frac{\delta}{\delta s} p(f,s,\mathbf{x}) \right| df \ . \tag{13}$$

The final *saliency measure* $Y_F$ for the feature $f$ of the region $F$, at scale $S$ and location $\mathbf{x}$ is then given by Equation (14

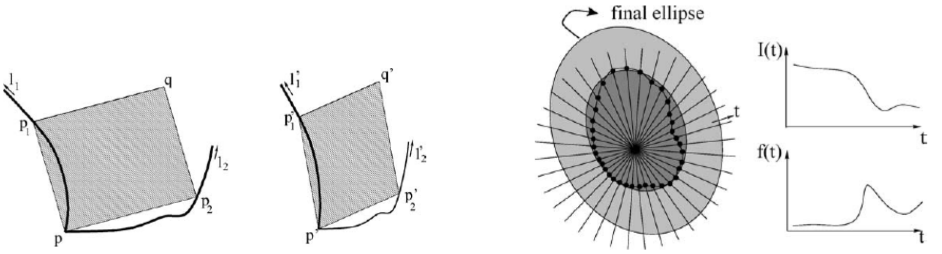
$$Y_F(S,\mathbf{x}) = H_F(S,\mathbf{x}) \times W_F(S,\mathbf{x}), \tag{14}$$

and all regions above a certain threshold are selected. The detector shows scale and rotational invariance properties. Recently, an affine invariant extension of this algorithm has been proposed [63]. It is based on an exhaustive search through all elliptical deformations of the patch under investigation.

It turns out that the main disadvantage of the algorithm is its long runtime - especially for the affine invariant implementation [91].

## 2.8 *Edge Based* and *Intensity Based* Regions

Tuytelaars et al. [128–130] proposed two completely different types of detectors. The first one, the so called *edge based regions detector* (EBR), exploits the behavior of edges around an interest point. Special photometric quantities $(I_1, I_2)$ are calculated and work as a stopping criterion following along the edges. In principle, the location of the interest point itself $(\mathbf{p})$ and the edge positions obtained by the stopping criterion $(\mathbf{p}_1, \mathbf{p}_2)$ define an affine frame (see Figure 1(a)). For further details on the implementation see [128] or [130]. The main disadvantage of this detector is the significant runtime. In particular it is faster than the EBSR detector but takes more time than all the other detectors mentioned so far.



(a) (b)

Figure 1: Principle of edge base regions (a) and intensity based regions (b) taken from [130].

The second one, the so called *intensity based region detector*, explores the image around an intensity extremal point in the image. In principle, a special function of image intensities $f = f(I, \mathbf{t})$ is evaluated along radially symmetric rays emanating from the intensity extreme detected on multiple scales. Similar to IBRs, a stopping criterion is defined, if this function goes through a local maximum. All the *stopping points* are linked together to form an arbitrary shape, which is in fact often replaced by an ellipse (see Figure 1(b)). The runtime performance of the detector is much better than for EBRs, but worse than the others mentioned above [91].

## 2.9 Summary of Common Properties

Table 1 summarizes the assigned category and invariance properties of the detectors described in this section. Furthermore we give a individual rating with respect to the detectors runtime, their repeatability and the number of detected points and regions (number of detections). Note, that those ratings are based on our own experiences with the original binaries provided by the authors (MSER, DoG, EBSR) and the vast collection of implementations provided by the Robotics Research Group at the University of Oxford[2]. Also the results from extensive evaluations studies in [31, 91] are taken into account.

| **detector** | **assigned category** | **invariance** | **runtime** | **repeat-ability** | **number of detections** |
|---|---|---|---|---|---|
| Harris | corner | none | very short | high | high |
| Hessian | region | none | very short | high | high |
| Harris-Lap.. | corner | scale | medium | high | medium |
| Hessian-Lap. | region | scale | medium | high | medium |
| DoG | region | scale | short | high | medium |
| Harris-Affine | corner | affine | medium | high | medium |
| Hessian-Affine | region | affine | medium | high | medium |
| MSER | region | projective | short | high | low |
| EBSR | other | scale | very long | low | low |
| EBR | corner | affine | very long | medium | medium |
| IBR | region | projective | long | medium | low |

Table 1: Summary of the detectors category, invariance properties and individual ratings due to runtime, repeatability and the number of obtained regions.

[2] http://www.robots.ox.ac.uk/∼vgg/research/affine/detectors.html, August 17, 2007

# 3 Region of Interest Descriptors

In this section we give a short overview about the most important state of the art region of interest descriptors. Feature descriptors describe the region or its local neighborhood already identified by the detectors by certain invariance properties. Invariance means, that the descriptors should be robust against various image variations such as affine distortions, scale changes, illumination changes or compression artifacts (e.g., JPEG). It is obvious, that the descriptors performance strongly depends on the power of the region detectors. Wrong detections of the region's location or shape will dramatically change the appearance of the descriptor. Nevertheless, robustness against such (rather small) location or shape detection errors is also an important property of efficient region descriptors.

One of the simplest descriptors is a vector of pixel intensities in the region of interest. In this case, cross-correlation of the vectors can be used to calculate a similarity measure for comparing regions. An important problem is the high dimensionality of this descriptor for matching and recognition tasks (dimensionality = number of points taken into account). The computational effort is very high and thus, like for most of the other descriptors, it is very important, to reduce the dimensionality of the descriptor by keeping their discriminative power.

Similar to the suggestion of Mikolajczyk in [90], all the above mentioned descriptors can roughly be divided into the following three main categories:

- distribution based descriptors,
- filter based descriptors and
- other methods.

The following descriptors will be discussed more detailed:

- SIFT [17, 80, 81],
- PCA-SIFT (gradient PCA) [65],
- gradient location-orientation histograms (GLOH), sometimes also called *extended SIFT* [90],
- Spin Images [72],
- shape context [9],
- Locally Binary Patterns [97],

- differential-invariants [68, 109],

- complex and steerable filters [6, 20, 32, 107], and

- moment-invariants [92, 129, 132].

## 3.1 Distribution-based descriptors

Distribution-based methods represent certain region properties by (sometimes multi-dimensional) histograms. Very often geometric properties (e.g., location, distance) of interest points in the region (corners, edges) and local orientation information (gradients) are used.

### 3.1.1 SIFT descriptor$^\star$

One of the most popular descriptors is the one developed by David Lowe [80, 81]. Lowe developed a carefully designed combination of detector and descriptor with excellent performance as shown in , e.g., [88]. The detector/descriptor combination is called *scale invariant feature transform* (SIFT) and consists of a scale invariant region detector - called *difference of Gaussian (DoG)* detector (Section 2.4) - and a proper descriptor often referred to as *SIFT-key*.

The DoG-point detector determines highly repetitive interest points at an estimated scale. To get a rotation invariant descriptor, the main orientation of the region is obtained by a 36 bin orientation histogram of gradient orientations within a Gaussian weighted circular window. Note, that the particular gradient magnitudes $m$ and local orientations $\phi$ for each pixel $I(x, y)$ in the image are calculated by simple pixel differences according to

$$\begin{aligned} m &= \sqrt{(I(x+1,y)-I(x-1,y))^2+(I(x,y+1)+I(x,y-1))^2} \\ \phi &= \tan^{-1}((I(x,y+1)+I(x,y-1))/(I(x+1,y)-I(x-1,y)) \ . \end{aligned} \tag{15}$$

The size of the respective window is well defined by the scale estimated from the DoG point detector. It is possible, that there is more than one main orientation present within the circular window. In this case, several descriptors on the same spatial location - but with different orientations - are created.

For the descriptor all the weighted gradients are normalized to the main orientation of the circular region. The circular region around the key-point is divided into 4 × 4 not overlapping patches and the histogram gradient orientations within these patches are calculated. Histogram smoothing is done in order to avoid sudden changes of orientation and the bin size is reduced to 8 bins in order to limit the descriptor's size. This results into a $4 \times 4 \times 8 = 128$ dimensional feature vector for each key-point. Figure 2 illustrates this procedure for a $2 \times 2$ window.

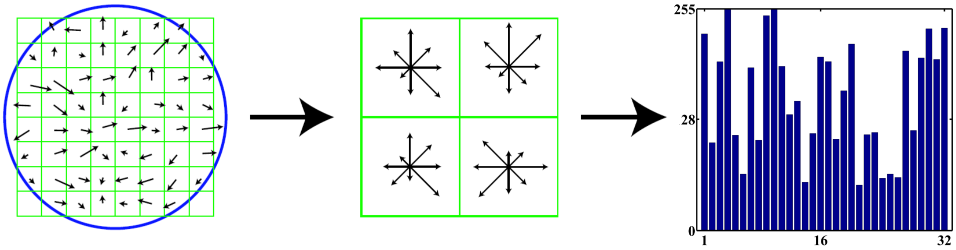


Figure 2: Illustration of the SIFT descriptor calculation partially taken from [81]. Note, that only a 32 dimensional histogram obtained from a $2 \times 2$ grid is depicted for a better facility of illustration.

Finally, the feature vector is normalized to unit length and thresholded in order to reduce the effects of linear and non-linear illumination changes.

Note that the scale invariant properties of the descriptor are based on the scale invariant detection behavior of the DoG-point detector. Rotational invariance is achieved by the main orientation assignment of the region of interest. The descriptor is not affine invariant itself. Nevertheless it is possible to calculate SIFT on other type of detectors, so that it can inherit scale or even affine invariance from them (e.g., Harris-Laplace, MSER or Harris-Affine detector).

### 3.1.2 PCA-SIFT or Gradient PCA

Ke and Sukthankar [65] modified the DoG/SIFT-key approach by reducing the dimensionality of the descriptor. Instead of gradient histograms on DoG-points, the authors applied Principal Component Analysis (PCA) (see Section 4.2) to the scale-normalized gradient patches obtained by the DoG detector. In principle they follow Lowe's approach for key-point detection They extract a $41 \times 41$ patch at the given scale centered on a key-point, but instead of a histogram they describe the patch of local gradient orientations with a PCA representation of the most significant eigenvectors (that is, the eigenvectors corresponding to the highest eigenvalues). In practice, it was shown, that the first 20 eigenvectors are sufficient for a proper representation of the patch. The necessary eigenspace can be computed off-line (e.g., Ke and

Sukthankar used a collection of 21.000 images). In contrast to SIFT-keys, the dimensionality of the descriptor can be reduced by a factor about 8, which is the main advantage of this approach. Evaluations of matching examples show that PCA-SIFT performs slightly worse than standard SIFT-keys [90].

### 3.1.3 Gradient Location-Orientation Histogram (GLOH)

Gradient location-orientation histograms are an extension of SIFT-keys to obtain higher robustness and distinctiveness. Instead of dividing the patch around the key-points into a 4 × 4 regular grid, Mikolajczyk and Schmid divided the patch into a radial and angular grid [90], in particular 3 radial and 8 angular sub-patches leading to 17 location patches (see Figure 3). The idea is similar to that used for shape context (see Section 3.1.5). Gradient orientations of those patches are quantized to 16 bin histograms, which in fact results in a 272 dimensional descriptor. This high dimensional descriptor is reduced by applying PCA and the 128 eigenvectors corresponding to the 128 largest eigenvalues are taken for description.

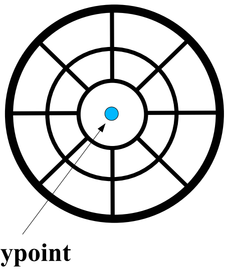


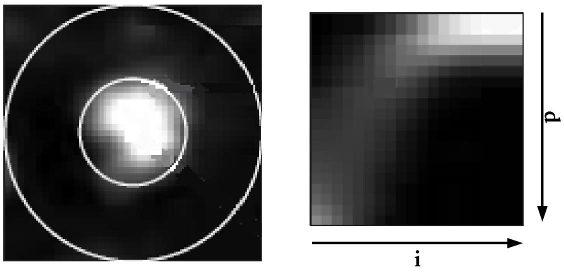
Figure 3: GLOH patch scheme

### 3.1.4 Spin Images$^\star$

Spin images have been introduced originally by Johnson and Hebert in a 3-D shape-based object recognition system for simultaneous recognition of multiple objects in cluttered scenes [56]. Lazebnik et al. [72] recently adapted this descriptors to 2D-images and used them for texture matching applications.

In particular they used an intensity domain spin image, which is a 2 dimensional histogram of intensity values $i$ and their distance from the center of the region $d$ - the spin image histogram descriptor (see Figure 4). Every row of the 2 dimensional descriptor represents the histogram of the grey values in an annulus distance $d$ from the center.

Finally a smoothing of the histogram is done and a normalization step achieves affine illumination invariance. Usually a quantization of the intensity



(a) (b)

Figure 4: Sample patch (a) and corresponding spin image (b) taken from [72].

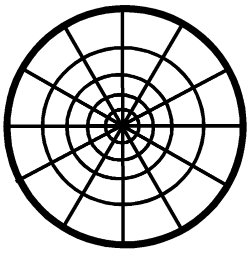


Figure 5: Histogram bins used for shape context.

histogram in 10 bins and 5 different radial slices is done thus resulting in a 50 dimensional descriptor [90]. The descriptor is invariant to in-plane rotations.

### 3.1.5 Shape Context

Shape context descriptors have been introduced by Belongie et al. [9] in 2002. They use the distribution of relative point positions and corresponding orientations collected in a histogram as descriptor. The primary points are internal or external contour points (edge points) of the investigated object or region. The contour points can be detected by any edge detector, e.g., Canny-edge detector [18], and are regularly sampled over the whole shape curve. A full shape representation can be obtained by taking into account all relative positions between two primary points and their pairwise joint orientations. It is obvious that the dimensionality of such a descriptor heavily increases with the size of the region. To reduce the dimensionality a coarse histogram of the relative shape sample points coordinates is computed - the *shape context*. The bins of the histogram are uniform in a $log - polar^2$ space (see Figure 5) which makes the descriptor more sensitive to the positions nearby the sample points.

Experiments have shown, that 5 bins for radius $log(r)$ and 12 bins for the angle Θ lead to good results with respect to the descriptor's dimensionality (60). Optional weighting the point contribution to the histogram with the gradient magnitude has shown to yield improved results [90].

### 3.1.6 Locally Binary Patterns

Locally binary patterns (LBP) are a very simple texture descriptor approach initially proposed by Ojala et al. [97]. They have been used in a lot of applications (e.g., [2, 44, 123, 139]) and are based on a very simple binary coding of thresholded intensity values.

In their simplest form they work on a 3 × 3 pixel neighborhood ($\mathbf{p}_1 ... \mathbf{p}_8$) and use the intensity value of the central point $I(\mathbf{p}_0)$ as reference for the threshold $T$ (see Figure 6(a)).

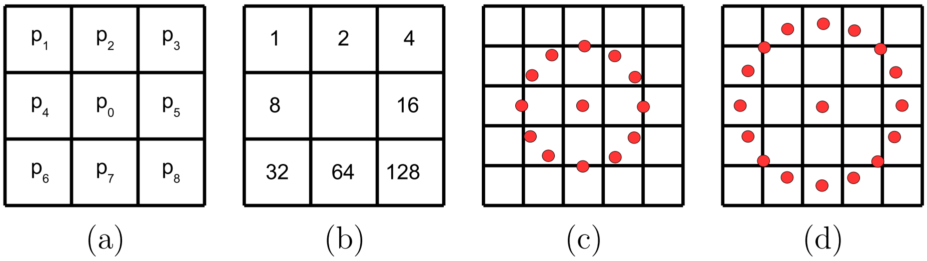


Figure 6: (a) Pixel neighborhood points and (b) their weights $W$ for the simplest version of locally binary patterns. Some examples for extended neighborhoods: (c) $r = 1.5, N = 12$ and (d) $r = 2.0, N = 16$

.

The neighborhood pixels $p_i$ for $i = 1...8$ are then signed ($S$) according to

$$S(p_0, p_i) = \left\{ \begin{array}{lllc} 1, & [I(\mathbf{p}_i) - I(\mathbf{p}_0)] & >= & 0 \\ 0, & [I(\mathbf{p}_i) - I(\mathbf{p}_0)] & < & 0 \end{array} \right\} \tag{16}$$

and form a locally binary pattern descriptor value $LBP(\mathbf{p}_0)$ by summing up the signs $S$, which are weighted by a power of 2 (weight $W(\mathbf{p}_i)$) (see Figure 6(b)).Usually the LBP values of a region are furthermore combined in a *LBP-histogram* to form a distinctive region descriptor:

$$LBP(\mathbf{p}_0) = \sum_{i=1}^{8} W(\mathbf{p}_i) S(\mathbf{p}_0, \mathbf{p}_i) = \sum_{i=1}^{8} 2^{(i-1)} S(\mathbf{p}_0, \mathbf{p}_i) \; . \tag{17}$$

The definition of the basic LBP approach can be easily extended to include all circular neighborhoods with any number of pixels [98] by bi-linear interpolation of the pixel intensity. Figure 6(c) and Figure 6(d) show some examples for such an extended neighborhood ($r = 1.5/2.0$ and $N = 12/16$).

Locally Binary Patterns are invariant to monotonic gray value transformations but they are not inherently rotational invariant. Nevertheless this can be achieved by rotating the neighboring points clockwise so many times, that a maximal number of most significant *weight times sign products* ($W \times S$) is zero [98].

Partial scale invariance of the descriptors can be reached in combination with scale invariant detectors. Some preliminary unpublished work [120] in our group has shown promising results in an object recognition task.

## 3.2 Filter-based Descriptors

### 3.2.1 Differential-Invariants$^{\star}$

Properties of local derivatives (local jets) are well investigated (e.g., [68]) and can be combined to sets of differential operators in order to obtain rotational invariance. Such a set is called *differential invariant descriptor* and has been used in different applications (e.g., [109]). One of the big disadvantages of differential invariants is, that they are only rotational invariant. Thus, the detector has to provide sufficient information if invariance against affine distortions is required.

Equation (19) shows an example for such a set of differential invariants ($S_3$) calculated up to the third order. Note that the components are written using the *Einstein* or *Indicial* notation and $\epsilon$ is the antisymmetric epsilon tensor ($\epsilon_{12} = -\epsilon_{21} = 1$ and $\epsilon_{11} = -\epsilon_{22} = 0$). The indices $i, j, k$ are the corresponding derivatives of the image $L$ in the two possible image dimensions $(x, y)$. For example

$$L_i L_{ij} L_j = L_x L_{xx} L_x + L_x L_{xy} L_y + L_y L_{yx} L_x + L_y L_{yy} L_y \ . \tag{18}$$

where, e.g., $L_{xy} = (L_x)_y$ is the derivative in y-direction of the image derivative in x-direction ($L_x$). The stable calculation is often obtained by using Gaussian derivatives:

$$\mathbf{S}_3 = \begin{bmatrix} L \\ L_i L_i \\ L_i L_{ij} L_j \\ L_{ii} \\ L_{ij} L_{ij} \\ \epsilon_{ij} \left( L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l \right) \\ L_{iij} L_j L_k L_k - L_{ijk} L_i L_j L_l \\ -\epsilon_{ij} L_{jkl} L_i L_k L_l \\ L_{ijk} L_i L_j L_k \end{bmatrix} \tag{19}$$

### 3.2.2 Steerable and Complex Filters$^\star$

*Steerability* depicts the fact, that it is possible to develop a linear combination of some basis-filters, which yield the same result, as the oriented filter rotated to a certain angle. For example, Freeman and Adelson [32] developed such steerable filters of different types (derivatives, quadrature filters etc.). A set of steerable filters can be used to obtain a rotational invariant region descriptor.

*Complex filters* is an umbrella term used for all filter types with complex valued coefficients. In this context, all filters working in the frequency domain (e.g., *Fourier - transformation*) are also called complex filters.

A typical example for the usage of complex filters is the approach from Baumberg [6]. In particular, he used a variant of the Fourier-Mellin transformation to obtain rotational invariant filters. A set of complex valued coefficients ($u^X_{n,m}$, see (20)) is calculated and a normalization is done dividing the complex coefficients by a unit length complex number proportional to $u^X_{0,k}$:

$$u^X_{n,m} = \int \frac{d^n}{dr^n} \, G_\sigma(r) \; e^{im\theta} \; J^X(r,\theta) \; r \; dr \; d\theta \tag{20}$$

$$J^X(r,\theta) = I^X(r \;\; \cos\theta + x_0, r \;\; \sin\theta + y_0) \; . \tag{21}$$

The polar coordinates $(r,\theta)$ are defined with respect to the image patch center located at $(x_0, y_0)$ and $G_\sigma(r)$ is a Gaussian with standard deviation $\sigma$. $I^X$ is the intensity of the corresponding color component $X$.

Another prominent *complex filter* approach has been introduced by Schaffalitzky and Zisserman [107]. They apply a bank of linear filters derived from the family

$$K_{m,n}(x,y) = (x+iy)^m(x-iy)^n G_\sigma(x,y) \; , \tag{22}$$

where $G_\sigma(x,y)$ is a Gaussian with standard deviation $\sigma$. $K_{0,0}$ is the average intensity of the region and the *diagonal* filters holding the property $m - n < const$ are orthogonal. The *diagonal* filters are ortho-normalized and their absolute values are taken as invariant features of the image patch.

As an example for the use of complex, steerable filters we mention the approach presented by Carneiro and Jepson [20]. They use a complex representation $A(\rho,\phi)$ of steerable quadrature pair filters $(g,h)$ from [32] and tuned them to a specific orientation $(\theta)$ and scale $(\sigma)$:

$$\begin{aligned} g(\mathbf{x}, \sigma, \theta) &= G_2(\sigma, \theta) * I(\mathbf{x}) \\ h(\mathbf{x}, \sigma, \theta) &= H_2(\sigma, \theta) * I(\mathbf{x}) \\ A(\rho, \phi) = \rho(\mathbf{x}, \sigma, \theta) e^{i\phi(\mathbf{x}, \sigma, \theta)} &= g(\mathbf{x}, \sigma, \theta) + ih(\mathbf{x}, \sigma, \theta) \; . \end{aligned} \tag{23}$$

In particular, the feature vector $\mathbf{F}_{n,r,p}(\mathbf{x})$ of an interest point consist of a certain number of filter responses $n$ calculated at the interest point location $\mathbf{x}$, and on equally spaced circle points of radius $r$ around them ($p$ partitions). The direction of the first circle point is given by the main orientation of the center pixel.

## 3.3 Other Methods

### 3.3.1 Cross-Correlation

*Cross-correlation* is a very simple method based on statistical estimation of the similarities between image intensities or color components around an interest point. The real descriptor is only the linearized vector of pixel intensities or individual color components in a certain window around a detected interest point.

The matching for such simple region descriptors is done by calculating the *cross-correlation* between pairs of descriptors. The *similarity score* $s_{a,b}$ between the respective pixel intensities $I_a, I_a$ in the local window $a$ or $b$ around an interest point is given by

$$s_{a,b} = \frac{\sum_{i=1}^{N} \left[ (I_a(i) - \mu_a)(I_b(i) - \mu_b) \right]}{\sqrt{\sum_{i=1}^{N} \left( I_a(i) - \mu_a \right)^2} \sqrt{\sum_{i=1}^{N} \left( I_b(i) - \mu_b \right)^2}} \; . \tag{24}$$

The descriptor's dimensionality is the number of pixels $N$ in the region the descriptor is calculated from. Note, the size of the region of interest is usually determined by the detector itself. If this is not the case (e.g., for Harris-Points) an exhaustive search over a lots of varying interest point neighborhoods is necessary.

The biggest disadvantage of cross-correlation is its high computational effort, especially, if an exhaustive search is required. Furthermore it is obvious that a simple vector of image intensities shows no invariance to any image transformation. Invariance properties can only be achieved by normalization of the patches based on the invariance properties of the region detector itself.

### 3.3.2 Moment Invariants$^\star$

Generalized intensity and color moments have been introduced by Van Gool in 1996 [132] to use the intensity (see (25)) or multi-spectral (see (26)) nature of image data for image patch description:

$$M_{pq}^{u} = \int\int_{\Omega} x^p y^q \left[I(x,y)\right]^u dxdy \tag{25}$$

$$M_{pq}^{abc} = \int\int_{\Omega} x^p y^q \left[R(x,y)\right]^a \left[G(x,y)\right]^b \left[B(x,y)\right]^c dxdy \ . \tag{26}$$

The moments implicitly characterize the intensity ($I$), shape or color distribution ($R, G, B$ are the intensities of individual color components) for a region $\Omega$ and can be efficiently computed up to a certain order ($p+q$) and degree ($u$ respectively $a+b+c$). $x^p$ and $y^p$ are powers of the respective image coordinates in the patch. Combinations of such generalized moments are shown to be invariant to geometric and photometric changes (see ,e.g., [92]). Combined with powerful, affine invariant regions based on corners and edges (see, e.g., [129]) they form a very powerful detector-descriptor combination.

For completeness we mention that Mikolajczyk and Schmid [90] use *gradient moments* in their extensive evaluation study about various descriptors. The *gradient moments* are calculated by

$$M_{pq}^{u} = \int\int_{\Omega} x^p y^q \left[I_d(x,y)\right]^u dxdy \ , \tag{27}$$

where $I_d(x,y)$ is the image gradient in the direction of $d$ at the location $(x,y)$ in the image patch.

## 3.4 Summary of Common Properties

In Table 2 we summarize a few common properties of the descriptors mentioned in this section. Besides the assignment to one of our selected categories (dist = distribution based, filter = filter based approach) we consider the rotational invariance property, mention the descriptors dimensionality and give an individual rating with respect to the descriptors performance.

Among the most popular types of invariance against geometrical distortions (rotation, scale change, affine distortion) we considered only the rotational invariance in our summary, because invariance against geometrical distortions is the task of the precedent detector. It should provide a rotational, scale or affine normalized patch the descriptor is calculated from. Nevertheless, as the most common scale adaptation and affine normalization techniques (see Section 2.3 and 2.5) provide a normalized patch defined up to an arbitrary rotation, the descriptors invariance against rotation is however crucial.

| **descriptor** | **assigned category** | **rotational invariance** | **dimensionality** | **performance** |
|---|---|---|---|---|
| SIFT | distrib. | no | high (128) | good |
| PCA-SIFT | distrib. | no | low (20) | good [36] |
| GLOH | distrib. | no | high (128) | good |
| Spin images | distrib. | yes | medium (50) | medium |
| Shape context | distrib. | no[1)] | medium (60) | good [36] |
| LBP | distrib. | no[1)] | very high (256) | - [4)] |
| Differential Inv. | filter | yes | low (9) | bad [8] |
| Steerable Filters | filter | yes | low | medium [8] |
| Complex Filters | filter | yes | low[3)] (15) | bad |
| Cross correlation | other | no | very high[2)] (N) | medium [81][5)] |
| Color moments | other | yes | low (18) | - [4)] |
| Intensity moments | other | yes | low | - [4)] |
| Gradient moments | other | yes | low (20) | medium |

Table 2: Summary of the descriptors category, rotational invariance property, dimensionality of the descriptors and an individual performance rating based on the investigations in [88, 90]. Legend: [1)] in the proposed form, [2)] N is the number of samples in the patch, [3)] implementation similar to [107], [4)] no comparable results, [5)] unstable results.

The descriptors dimensionality is very important, because the dimensionality of the descriptor heavily influences the complexity of the matching process (runtime) and the memory requirements for storing the descriptors. We divide the descriptors into three main categories with respect to the dimensionality (low, medium, high) and furthermore denote the dimensionality of the *original* implementation by the authors in parentheses. Nevertheless we mention, that for most of the descriptors the dimensionality can be controlled by certain parameterizations (e.g., for PCA-SIFT it is possible to select an arbitrary number of significant dimensions with respect to the desired complexity).

The individual performance ratings are based on the evaluation work of Mikolajczyk and Schmid [88, 90]. In general, an appraisal of various descriptors is much more difficult than the personal review we did for the detector approaches. This is, because the descriptors can not be evaluated on their own, it is only possible to compare certain detector-descriptor combinations. Thus it is difficult to separate the individual influences and an excellent performing descriptor may show worse results in combination with an inappropriate, poor performing detector. The authors in [90] tackled that problem and did an extensive evaluation on different scene types and various detector-descriptor combinations. Thus, we refer to their results and rate the descriptors with our individual performance rankings (good, medium, bad). Please note that Mikolajczyk and Schmid did their evaluations on re-implementations of the original descriptors with occasionally differing dimensionality. We denote them in squared brackets behind our rating.

# 4 Subspace Methods

## 4.1 Introduction

In this section we discuss global appearance-based methods for object recognition. In fact, the discussion is reduced to *subspace methods*. The main idea for all of these methods is to project the original input images onto a suitable lower dimensional subspace, that represents the data best for a specific task. By selecting different optimization criteria for the projected data different methods can be derived.

## 4.2 Principal Component Analysis

Principal Component Analysis (PCA) [57] also known as Karhunen-Loève transformation (KLT) [3] [64,79] is a well known and widely used technique in statistics. It was first introduced by Pearson [100] and was independently rediscovered by Hotelling [48]. The main idea is to reduce the dimensionality of data while retaining as much information as possible. This is assured by a projection that maximizes the variance but minimizes the mean squared reconstruction error at the same time.

Due to its properties PCA can be considered a prototype for subspace methods. Thus, in the following we give the derivation of PCA, discuss the properties of the projection, and show how it can be applied for image classification. More detailed discussions are given by [24, 57, 83, 116].

[3] Most authors do not distinguish between PCA and KLT. In fact, it can be shown that for mean normalized data both methods are identical [36]. As for most applications the data is assumed to be mean normalized without loss of generality both terms may be used.

### 4.2.1 Derivation of PCA

Pearson [100] defined PCA as the linear projection than minimizes the squared distance between the original data points and their projections. Equivalently, Hotelling considered PCA as an orthogonal projection that maximizes the variance in the projected space. In addition, PCA can be viewed in a probabilistic way [106, 125] or can be formulated in context of neural networks [24, 96]. Hence, there are different ways to define PCA but, finally, all approaches yield the same linear projection.

In the following we give the most common derivation based on maximizing the variance in the projected space. Given $n$ samples $\mathbf{x}_j \in \mathbb{R}^m$ and let $\mathbf{u} \in \mathbb{R}^m$ with

$$||\mathbf{u}|| = \mathbf{u}^T\mathbf{u} = 1 \tag{28}$$

be an orthonormal projection direction. A sample $\mathbf{x}_j$ is projected onto $\mathbf{u}$ by

$$a_j = \mathbf{u}^T\mathbf{x}_j \ . \tag{29}$$

The sample variance in the projected space can be estimated by

$$S^2 = \frac{1}{n-1}\sum_{j=1}^{n}(a_j - \bar{a}) \ , \tag{30}$$

where $\bar{a}$ is the sample mean in the projected space. From

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j \tag{31}$$

we get

$$\bar{a} = \mathbf{u}^T\bar{\mathbf{x}} \ . \tag{32}$$

Thus, the sample variance in the projected space is given by

$$S^2 = \frac{1}{n-1}\sum_{j=1}^{n}(a_j - \bar{a}) = \frac{1}{n-1}\sum_{j=1}^{n}\left(\mathbf{u}^T\mathbf{x} - \mathbf{u}^T\bar{\mathbf{x}}\right) = \mathbf{u}^T\mathbf{C}\mathbf{u} \ , \tag{33}$$

where

$$\mathbf{C} \in \mathbb{R}^{m\times m} = \frac{1}{n-1}\sum_{j=1}^{n}(\mathbf{x}_j - \bar{\mathbf{x}})(\mathbf{x}_j - \bar{\mathbf{x}})^T \tag{34}$$

is the sample covariance matrix of $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$.

Hence, to maximize the variance in the projected space, we can consider the following optimization problem:

$$\begin{aligned} &\max \mathbf{u}^T \mathbf{C} \mathbf{u} \\ &\text{s.t. } \ \mathbf{u}^T \mathbf{u} = 1 \ . \end{aligned} \tag{35}$$

To compute the optimal solution for (35) we apply a Lagrange multiplier:

$$f(\mathbf{u}, \lambda) = \mathbf{u}^T \mathbf{C} \mathbf{u} + \lambda \left(1 - \mathbf{u}^T \mathbf{u}\right) \ . \tag{36}$$

By partial deviation of (36) with respect to $\mathbf{u}$

$$\frac{\partial f(\mathbf{u}, \lambda)}{\partial \mathbf{u}} = 2\mathbf{C}\mathbf{u} - 2\lambda \mathbf{u} = 0 \tag{37}$$

we get

$$\mathbf{C}\mathbf{u} = \lambda \mathbf{u} \ . \tag{38}$$

Hence, the maximum for the Lagrange multiplier is obtained if $\lambda$ is an eigenvalue and $\mathbf{u}$ is an eigenvector of $\mathbf{C}$. A complete basis[4] $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}]$ can be obtained by computing a full eigenvalue decomposition (EVD)

$$\mathbf{C} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T \ . \tag{39}$$

Alternatively, from (28) and (33) it can be seen that there is a strong relationship between the optimization problem (35) and the Rayleigh quotient

$$R(\mathbf{u}) = \frac{\mathbf{u}^T \mathbf{C} \mathbf{u}}{\mathbf{u}^T \mathbf{u}} \ , \tag{40}$$

which is maximized if $\mathbf{u}$ is an eigenvector of $\mathbf{C}$. Moreover, if $\mathbf{u}$ is an eigenvector and $\lambda$ is an eigenvalue of $\mathbf{C}$ we get

$$R(\mathbf{u}) = \frac{\mathbf{u}^T \mathbf{C} \mathbf{u}}{\mathbf{u}^T \mathbf{u}} = \frac{\mathbf{u}^T \lambda \mathbf{u}}{\mathbf{u}^T \mathbf{u}} = \lambda \ . \tag{41}$$

Each eigenvector $\mathbf{u}$ of $\mathbf{C}$ is projected onto its corresponding eigenvalue $\lambda$. Hence, the variance described by the projection direction $\mathbf{u}$ is given by the eigenvalue $\lambda$.

Other derivation based on the maximum variance criterion are given in, e.g., [13, 57]. Contrary, equivalent derivations obtained by looking at the

[4] One dimension is lost due to the mean normalization. Hence, the complete basis consists only of $n-1$ basis vectors.

mean squared error criterion are given, e.g., by Diamantaras and Kung [24] or by Duda et al. [26]. While Diamantaras and Kung discuss the derivation from a statistical view by estimating the expected reconstruction error, Duda et al. give a derivation, that is similar to that in the original work of Pearson [100], who was concerned to find lines and planes, that best fit to a given set of data points. For a probabilistic view/derivation of PCA see [106,125].

### 4.2.2 Batch Computation of PCA$^\star$

For batch methods, in general, it is assumed that all training data is given in advance. Thus, we have a fixed set of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$ organized in a matrix $\mathbf{X} = [\mathbf{x}_1, \dots \mathbf{x}_n] \in \mathbb{R}^{m \times n}$. To estimate the PCA projection we need to solve the eigenproblem for the (sample) covariance matrix $\mathbf{C}$ of $\mathbf{X}$. Therefore, we first have to estimate the sample mean

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j \tag{42}$$

and the mean normalized data $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n\}$, where

$$\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}} \ . \tag{43}$$

Then, the sample covariance matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ is calculated by

$$\mathbf{C} = \frac{1}{n-1} \hat{\mathbf{X}} \hat{\mathbf{X}}^T \ . \tag{44}$$

Solving the eigenproblem for $\mathbf{C}$ yields the eigenvectors $\mathbf{u}_j$ and the corresponding eigenvalues $\lambda_j$ sorted in decreasing order. The whole projection basis (subspace) is given by

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{n-1}] \in \mathbb{R}^{m \times n-1} \ . \tag{45}$$

One degree of freedom is lost due to the mean normalization. Hence, the dimension of $\mathbf{U}$ is reduced by one. As most information is captured within the first eigenvectors corresponding to the greatest eigenvalues usually only $k < n - 1$ eigenvectors are used for the projection. The algorithm is summarized more formally in Algorithm 1.

### 4.2.3 Efficient Batch PCA$^\star$

The dimension of the covariance matrix directly depends on $m$, the number of rows of $\mathbf{A}$, which may be quite large for practical applications (e.g., when the data vectors represent images). Thus, the method described above

**Algorithm 1** Batch PCA

```
Input: data matrix $\mathbf{X}$
Output: sample mean vector $\bar{\mathbf{x}}$, basis of eigenvectors $\mathbf{U}$, eigenvalues $\lambda_j$

Compute sample mean vector:
    $\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j$
Normalize input images:
    $\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}}$
    $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n]$
Compute covariance matrix:
    $\mathbf{C} = \frac{1}{n-1}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$
Compute eigenvectors $\mathbf{u}_j$ and eigenvalues $\lambda_j$ of $\mathbf{C}$:
    $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}]$
```

is not feasible for solving the eigenproblem for large matrices due to memory requirement and computational costs. But due to the properties of the covariance matrix there exist more efficient methods to estimate the PCA projection matrix (e.g., [94]). It is well known (see Appendix C.6), that for any matrix $\mathbf{A}$ the matrix products $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ share the same non-zero eigenvalues. Let $\mathbf{u}$ and $\lambda$ be an eigenvector and an eigenvalue of $\mathbf{A}^T\mathbf{A}$. Thus, we have

$$\mathbf{A}^T\mathbf{A}\mathbf{u} = \lambda\mathbf{u} \ . \tag{46}$$

By left multiplying both sides of (46) with $\mathbf{A}$ we get

$$\mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{u}) = \lambda(\mathbf{A}\mathbf{u}) \ . \tag{47}$$

Hence, $\lambda$ is also an eigenvalue of $\mathbf{A}\mathbf{A}^T$; the corresponding eigenvector is given by $\mathbf{A}\mathbf{u}$. To further ensure that the eigenbasis has unique length the thus obtained eigenvectors have to be scaled by the square root of the corresponding eigenvalue.

Let

$$\breve{\mathbf{G}} = \frac{1}{n-1}\hat{\mathbf{X}}^T\hat{\mathbf{X}} \tag{48}$$

or re-written

$$\breve{\mathbf{G}} = \frac{1}{\sqrt{n-1}}\hat{\mathbf{X}}^T\frac{1}{\sqrt{n-1}}\hat{\mathbf{X}} \tag{49}$$

be the scaled Gram matrix[5] of $\hat{\mathbf{X}}$. Solving the eigenproblem for $\breve{\mathbf{G}}$ yields the eigenvalues $\breve{\lambda}_j$ and the eigenvectors $\breve{\mathbf{u}}_j$. Hence, from (47) and (49) we get that the eigenvalues $\lambda_j$ and the eigenvectors $\mathbf{u}_j$ of the covariance matrix $\mathbf{C}$ are given by

$$\begin{aligned}\lambda_j &= \breve{\lambda}_j\\ \mathbf{u}_j &= \frac{1}{\sqrt{n-1}\sqrt{\lambda_j}}\ \hat{\mathbf{X}}\breve{\mathbf{u}}_j\ .\end{aligned} \tag{50}$$

If $\hat{\mathbf{X}}$ has (much) more rows than columns, i.e., $n < m$, which is often the case for practical applications, $\breve{\mathbf{G}} \in \mathbb{R}^{n\times n}$ is a much smaller matrix than $\mathbf{C} \in \mathbb{R}^{m\times m}$. Thus, the estimation of the eigenvectors is computationally much cheaper and we get a more efficient method. The thus obtained algorithm is summarized more formally in Algorithm 2.

**Algorithm 2** Efficient Batch PCA

```
Input: data matrix $\mathbf{X}$
Output: sample mean vector $\bar{\mathbf{x}}$, basis of eigenvectors $\mathbf{U}$, eigenvalues $\lambda_j$

1: Compute sample mean vector:
       $\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j$
2: Normalize input images:
       $\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}}$
       $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n]$
3: Compute Gram matrix:
       $\breve{\mathbf{G}} = \frac{1}{n-1}\hat{\mathbf{X}}^T\hat{\mathbf{X}}$
4: Compute eigenvectors $\breve{\mathbf{u}}_j$ and eigenvalues $\breve{\lambda}_j$ of $\breve{\mathbf{G}}$

5: Determine eigenvalues of $\mathbf{C}$:
       $\lambda_j = \breve{\lambda}_j$
6: Determine eigenvectors of $\mathbf{C}$:
       $\mathbf{u}_j = \frac{1}{\sqrt{n-1}\sqrt{\lambda_j}}\hat{\mathbf{X}}\breve{\mathbf{u}}_j$
       $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}]$
```

[5] For the definition of the Gram matrix, its properties, and its relation to the covariance matrix see Appendix Appendix A and Appendix C.6, respectively.

### 4.2.4 PCA by Singular Value Decomposition$^\star$

For symmetric and positive semi-definite matrices Singular Value Decomposition (SVD) and Eigenvalue Decomposition (EVD) become equivalent (see Appendix C.6). As the covariance matrix is a positive and semi-definite matrix SVD may be applied to compute the EVD. But still the matrix multiplications $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ or $\hat{\mathbf{X}}^T\hat{\mathbf{X}}$ have to be performed, respectively. To even avoid these matrix multiplications we can apply SVD directly on the mean normalized data matrix $\hat{\mathbf{X}}$ to compute the eigenvectors $\mathbf{u}_i \in \mathbb{R}^m$ of the sample covariance matrix $\mathbf{C}$.

Consider the SVD[6] of the mean normalized sample matrix $\hat{\mathbf{X}} \in \mathbb{R}^{m\times n}$:

$$\hat{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{51}$$

Then, the SVD of $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ is given by

$$\hat{\mathbf{X}}\hat{\mathbf{X}}^T = \mathbf{U}\mathbf{\Sigma}\underbrace{\mathbf{V}^T\mathbf{V}}_{\mathbf{I}_{m\times m}}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T\ . \tag{52}$$

From (51) and (52) it is clear that the left singular vectors of $\hat{\mathbf{X}}$ and $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ are identical. In addition, the left singular vectors of $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ are also the eigenvectors of $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ and the squared singular values $\sigma_j^2$ of $\hat{\mathbf{X}}$ are the eigenvalues $\lambda_j$ of $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$. Hence, we can apply SVD on $\hat{\mathbf{X}}$ to estimate the eigenvalues and the eigenvectors of $\hat{\mathbf{X}}\hat{\mathbf{X}}^T$. The algorithm using this SVD approach to compute the PCA projection matrix is summarized more formally in Algorithm 3.

For our application we use this implementation of PCA for two reasons: (a) the computation of SVD is numerically often more stable than the computation of the EVD and (b) since there exist several incremental extensions of SVD this approach can simply be adapted for on-line learning.

### 4.2.5 Projection and Reconstruction

If the matrix $\mathbf{U} \in \mathbb{R}^{m\times n-1}$ was calculated with any of the methods discussed above it can be used to project data onto the lower dimensional subspace. Thus, given an input vector $\mathbf{x} \in \mathbb{R}^m$ the projection $\mathbf{a} \in \mathbb{R}^{n-1}$ is obtained by

$$\mathbf{a} = \mathbf{U}^T\hat{\mathbf{x}}\ , \tag{53}$$

where $\hat{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean normalized input data. Hence, the $j$-th element of the projected data $\mathbf{a} = [a_1, \ldots, a_{n-1}]$ is obtained by computing the inner product of the mean normalized input vector $\hat{\mathbf{x}}$ and the $j$-th basis vector $\mathbf{u}_j$:

[6] A detailed discussion on SVD is given in Appendix Appendix C.

**Algorithm 3** SVD PCA

```
Input: data matrix $\mathbf{X}$
Output: sample mean vector $\bar{\mathbf{x}}$, basis of eigenvectors $\mathbf{U}$, eigenvalues $\lambda_j$

Compute sample mean vector:
    $\bar{\mathbf{x}} = \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j$
Normalize input images:
    $\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}}$
    $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n]$
Compute left singular vectors $\breve{\mathbf{u}}_j$ and singular values $\sigma_j$
Compute eigenvalues $\lambda_j$ of $\mathbf{C}$:
    $\lambda_j = \frac{\sigma_j^2}{n-1}$
Compute eigenvectors $\mathbf{u}_j$ $\mathbf{C}$:
    $\mathbf{u}_j = \breve{\mathbf{u}}_j$
    $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}]$
```

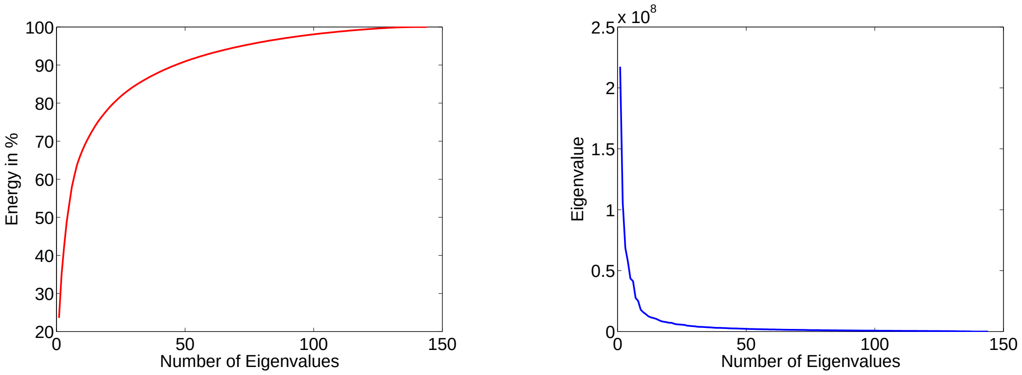
$$a_j = \mathbf{u}_j^T \hat{\mathbf{x}} = \left\langle \mathbf{u}_j^T, \hat{\mathbf{x}} \right\rangle = \sum_{i=1}^{m} u_{i,j} x_i \ . \tag{54}$$

Once the projection $\mathbf{a}$ was estimated the original data can be reconstructed by

$$\mathbf{x} = \mathbf{U}\mathbf{a} + \bar{\mathbf{x}} = \sum_{j=1}^{n-1} a_j \mathbf{u}_j + \bar{\mathbf{x}} \ . \tag{55}$$

As finally shown by (41) the variance of the $j$-th principal axis $\mathbf{u}_j$ is equal to the $j$-th eigenvalue $\lambda_j$. Thus, most information is covered within the eigenvectors according to the largest eigenvalues. To illustrate this, Figure 7 shows a typical example of the accumulated energy (a) and the decreasing size of the eigenvalues (b). The energy can be considered the fraction of information, that is captured by approximating a representation by a smaller number of vectors. Since this information is equivalent to the sum of the corresponding eigenvalues the thus defined accumulated energy describes the accuracy of the reconstruction.

Hence, it is clear, that usually only $k$, $k < n$, eigenvectors are needed to represent a data vector $\mathbf{x}$ to a sufficient degree of accuracy:



(a) (b)

Figure 7: Most reconstructive information is captured by the larges eigenvalues: (a) accumulated energy (variance) and (b) decreasing eigenvalues.

$$\tilde{\mathbf{x}} = \mathbf{U}_k\mathbf{a} + \bar{\mathbf{x}} = \sum_{j=1}^{k} a_j\mathbf{u}_j + \bar{\mathbf{x}} \; . \tag{56}$$

A measurement for the quality of the reconstruction is the squared reconstruction error:

$$\begin{aligned}\epsilon &= ||\mathbf{x} - \tilde{\mathbf{x}}||^2 = \left|\left|\sum_{j=1}^{n-1} a_j\mathbf{u}_j + \bar{\mathbf{x}} - \left(\sum_{j=1}^{k} a_j\mathbf{u}_j + \bar{\mathbf{x}}\right)\right|\right|^2 = \\ &= \left|\left|\sum_{j=k+1}^{n-1} a_j\mathbf{u}_j\right|\right|^2 = \sum_{j=k+1}^{n-1} a_j^2 \; .\end{aligned} \tag{57}$$

Hence, the squared reconstruction error is equal to the squared coefficients of the discarded eigenvectors. Since these are usually not known the expected error can be described by the expected value of the discarded variance, which is given by

$$\mathrm{E}\left[\epsilon\right] = \sum_{j=k+1}^{n-1} \lambda_j \; . \tag{58}$$

### 4.2.6 PCA for Image Classification

PCA was introduced to Computer Vision by Kirby and Sirovich [66] and became popular since Turk and Pentland [127] applied it for face recognition. Therefor, images are considered to be high dimensional vectors and a given image $\mathbf{I}$ of size $h \times w$ is arranged as a vector $\mathbf{x} \in \mathbb{R}^m$, where $m = hw$. More formally this was discussed by Murase and Nayar [95] in the field of object recognition and pose estimation. From [95] it is clear that high dimensional image data can be projected onto a subspace such that the data lies on a lower dimensional manifold, which further reduces the computational complexity for a classification task. Other approaches use PCA as a pre-processing step to reduce the dimensionality first (e.g., [7, 53]) or use PCA to extract features and to perform a different learning algorithm to compute a classifier (e.g., [55]).

In the following, we consider PCA directly a method for image classification. Given a set of $n$ templates $\mathbf{x}_j \in \mathbb{R}^m$, $j = 1, \ldots, n$, representing the object. Then, an unknown test sample $\mathbf{y}$ can be classified by simply template matching. Therefor, the correlation between the test sample $\mathbf{y}$ and the templates $\mathbf{x}_j$ is analyzed:

$$\rho = \frac{\mathbf{x}_j^T \mathbf{y}}{||\mathbf{x}_j||\,||\mathbf{y}||} > \theta \ . \tag{59}$$

If the correlation is above some threshold $\theta$ for at least one template $\mathbf{x}_j$ a match is found. Assuming normalized images $||\mathbf{x}_j|| = ||\mathbf{y}|| = 1$ we get $||\mathbf{x}_j - \mathbf{y}||^2 = 2 - 2\mathbf{x}_j^T \mathbf{y}$. Hence, we can apply a simpler criterion based on the sum-of-squared differences:

$$||\mathbf{x}_j - \mathbf{y}||^2 < \theta \ . \tag{60}$$

Clearly, this is not feasible if $n$ and $m$ are quite large due to the expected computational costs and the memory requirements. Thus, it would be desirable to have a lower dimensional representation of the data. In fact, PCA provides such a lower dimensional approximation.

Assuming that a subspace $\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$ and the sample mean $\bar{\mathbf{x}}$ were estimated from the training samples $\mathbf{x}_j$. Let $\hat{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{x}}$ and $\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}}$ be the mean normalized unknown test sample and the mean normalized $j$-th template and $\mathbf{a} = \mathbf{U}_k^T \hat{\mathbf{x}}_j$ and $\mathbf{b} = \mathbf{U}_k^T \hat{\mathbf{y}}$ be the corresponding projections onto the subspace $\mathbf{U}_k$. From (56) we get that $\hat{\mathbf{y}}$ and $\hat{\mathbf{x}}_j$ can be approximated by a linear combination of the basis $\mathbf{U}_k$. Since $\mathbf{U}_k$ is an orthonormal basis we get

$$||\hat{\mathbf{x}}_j - \hat{\mathbf{y}}||^2 \approx \left|\left| \sum_{j=1}^{k} a_j \mathbf{u}_j - \sum_{j=1}^{k} b_j \mathbf{u}_j \right|\right|^2 = \left|\left| \sum_{j=1}^{k} (a_j - b_j)\, \mathbf{u}_j \right|\right|^2 = ||\mathbf{a} - \mathbf{b}||^2 \ . \tag{61}$$

Thus, it is clear that we can compute the matching on the projected PCA subspace instead on the original image space:

$$||\mathbf{a} - \mathbf{b}||^2 < \theta \ . \tag{62}$$

Once we have obtained the projection $\mathbf{b} = [b_1, \ldots, b_k]$ we can reconstruct the image using (56) and determine the reconstruction error

$$\epsilon_k = ||\mathbf{y} - \tilde{\mathbf{y}}||^2 \ . \tag{63}$$

Alternatively, we can perform the classification by thresholding this error [75]. To illustrate this consider the following example. An object, a soft toy, was learned form 125 different views. Examples of these views and the first five eigenimages of the resulting representation are shown in Figure 8.

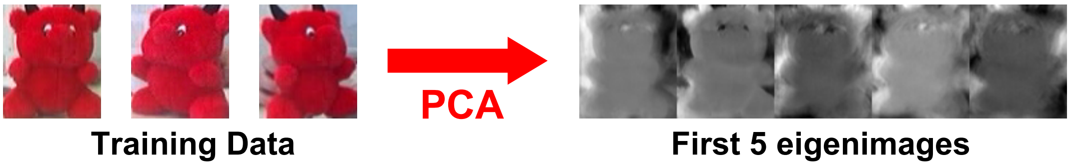


Figure 8: PCA Learning: A lower-dimensional representation is obtained from input images showing different views of the object.

The results for the recognition task (using only 10 eigenimages) are shown in Figure 9 and Table 3, respectively. From Figure 9 it can be seen that the reconstruction for the learned object is satisfactory while it completely fails for the face. More formally, Table 3 shows that the mean squared and the mean pixel reconstruction error differ by a factor of approximately 100 and 10, respectively. In addition, also considering the distance in the subspace shows the lower-dimensional representation for the learned object is much closer to the trained subspace.

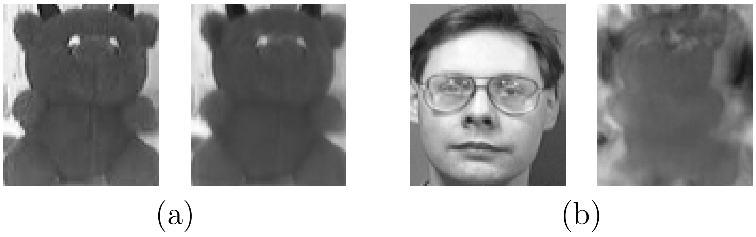

(a) (b)

Figure 9: Test images and its reconstruction: (a) an object representing the learned object class (soft toy); (b) an object not representing the learned object class (face).

| | mean squared error | mean pixel error | min. distance |
|---|---|---|---|
| devil | 3.77e+005 | 4.99 | 32.65 |
| face | 2.12e+007 | 47.11 | 1629.11 |

Table 3: Mean squared error, mean pixel error and minimum distance to subspace for the learned object class (devil) and an unknown object (face).

#### 4.2.7 Robust PCA$^\star$

In general, when analyzing real-world data one is confronted with unreliable data, e.g., errors in measurement or missing data. Hence, robust methods [42, 51] are needed. The same applies for visual learning (e.g., parts of the object of interest are occluded or pixels are unreliable due to camera noise). When addressing robustness in field of learning one has to distinguish between robustness in the evaluation stage and robustness in the learning stage.

For the first case it is assumed that the samples in the learning stage are undisturbed. Hence, in the evaluation stage unreliable data can be reconstructed from the previously learned model (e.g., [14, 75, 104]). In contrast robust learning, i.e., learning from unreliable data, is a more difficult problem, since there is no previous knowledge, that can be used to estimate outliers. Several methods have been proposed to robustly extract the principal axes in the presence of outliers [126, 136]. Other approaches use robust M-estimator [126] or are based on the EM formulation of PCA [106, 117, 125].

In the following, we will summarize the sub-sampling-based approach of Leonardis and Bischof [75]. It was originally indented for the recognition stage only, but as shown in Section 4.2.8 if a sufficient starting model is available it can also be applied for incremental robust learning.

The main idea is that the coefficients $\mathbf{a} = [a_1, \ldots, a_k]$ for a mean normalized sample $\hat{\mathbf{x}}$ can be approximately estimated from a smaller subset of pixels by solving an overdetermined linear system. If the subset does not contain outliers this estimate is an accurate approximation for the true values. To ensure robustness the coefficients are estimated in an iterative process. First, a subset of pixels $\hat{\mathbf{x}}^\star$ is randomly selected and the coefficients $\mathbf{a}^\star$ are computed by solving an overdetermined system. Next, the images is reconstructed and those pixels with the greatest reconstitution error are discarded. This steps are iterated until a pre-defined number of remaining pixels is reached.

Since not all random initializations would produce good results several different hypotheses are generated as described above. Finally, the best one is chosen. The selection is done based on the reconstruction error of the compatible points or more formally using the MDL principle [75].

To illustrate the benefits of robust object recognition consider the following example. First, a PCA representation was learned from undisturbed images and then the reconstruction error[7] for both, the standard and the robust approach, was analyzed when the portion of noise (occlusion) is increased. As can be seen in Figure 10 the reconstruction error is continuously growing for the standard approach while the performance of the robust method stays the same even for 25% occlusion.

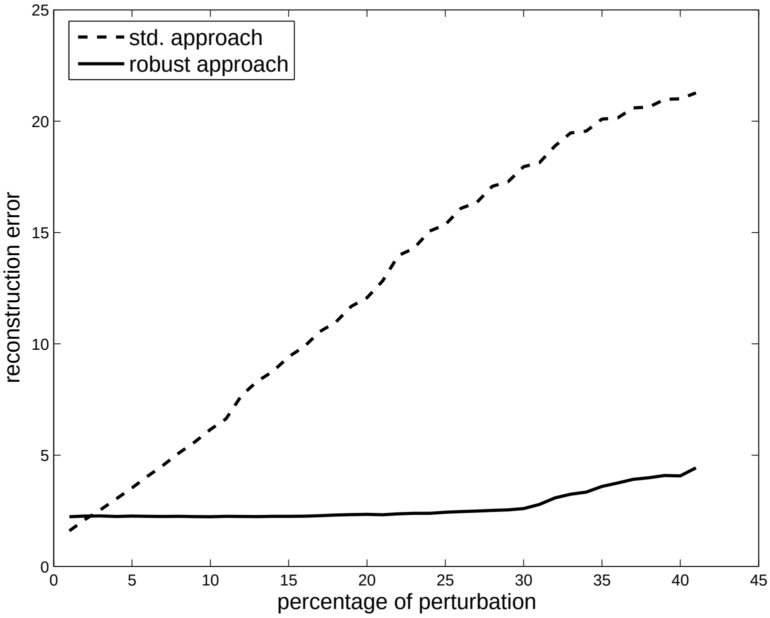


Figure 10: Robust PCA in evaluation stage: smaller reconstruction error when applying the robust method for corrupted data.

### 4.2.8 Incremental PCA$^\star$

Not all input data may be given in advance (e.g., receiving data from a video stream) and huge data sets can not be processed by the standard batch methods due to memory constraints. To solve these problems different incremental PCA approaches have been proposed that are based on incremental SVD-updating (e.g., [16, 21, 40, 94]). Recently even robust and incremental [77, 118] approaches have been proposed. In contrast to batch methods for incremental learning the current representation is updated whenever a

[7] The reconstruction error was computed from the undisturbed original images. Alternatively, the distance between the learned eigenspace and the projected shape may be used as measurement (which will yield similar curves).

new data vector is available. In the following we discuss the incremental PCA method, that was proposed in [118].

Assuming that an eigenspace was already built from $n$ images, at step $n+1$ the current eigenspace can be updated in the following way: First, the new image $\mathbf{x}$ is projected in the current eigenspace $\mathbf{U}^{(n)}$ and the image is reconstructed: $\tilde{\mathbf{x}}$. The residual vector $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$ is orthogonal to the current basis $\mathbf{U}^{(n)}$. Thus, a new basis $\mathbf{U}'$ is obtained by enlarging $\mathbf{U}^{(n)}$ with $\mathbf{r}$. $\mathbf{U}'$ represents the current images as well as the new sample. Next, batch PCA is performed on the corresponding low-dimensional space $\mathbf{A}'$ and the eigenvectors $\mathbf{U}''$, the eigenvalues $\boldsymbol{\lambda}''$ and the mean $\boldsymbol{\mu}''$ are obtained. To update the subspace the coefficients are projected in the new basis $\mathbf{A}^{(n+1)} = \mathbf{U}''^{T}\left(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}\right)$ and the subspace is rotated: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$. Finally, the mean $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ and the eigenvalues $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ are updated. In each step the dimension of the subspace is increased by one. To preserve the dimension of the subspace the least significant principal vector may be discarded [41]. The method is summarized more detailed in Algorithm 4.

To obtain an initial model, the batch method may be applied to a smaller set of training images. Alternatively, to have a fully incremental algorithm, the eigenspace may be initialized using the first training image $\mathbf{x}$: $\boldsymbol{\mu}^{(1)} = \mathbf{x}$, $\mathbf{U}^{(1)} = \mathbf{0}$ and $\mathbf{A}^{(1)} = \mathbf{0}$.

To illustrate the benefits of the incremental approach we trained and evaluated models using different PCA algorithms (batch, “iterated batch”[8] and incremental) on a representative data set containing 85 images of size 320 × 240 pixels. Figure 11(a) shows, as expected, that the reconstruction error is decreasing if the number of training images is increased. But, in addition, Figure 11(a) reveals that the reconstruction error is similar for both, the incremental and the ”iterated batch” method. Thus, there is only a non-essential loss of accuracy when using the incremental approach. But as can be seen in Figure 11(b) there are huge differences in the computational costs[9] for the different methods. In fact, compared to the “iterated batch” the computational costs of the incremental method are only approximately 1/40! The results for the whole training set containing 85 images are summarized in Table 4.

This method can easily be extended in a robust manner, i.e., corrupted input images may be used for incrementally updating the current model. To achieve this, outliers in the current image are detected and replaced by more confident values: First, an image is projected to the current eigenspace using

[8] A full batch PCA is applied on all currently available images whenever a new image arises.

[9] The learning times were obtained by evaluating the training in MATLAB on a 3GHz machine.

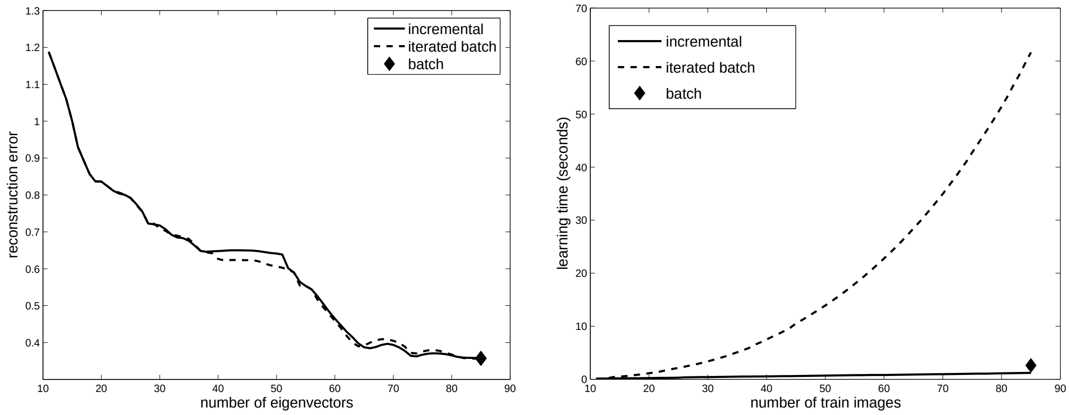
**Algorithm 4** Incremental PCA

```
Input: mean vector $\boldsymbol{\mu}^{(n)}$, eigenvectors $\mathbf{U}^{(n)}$, coefficients $\mathbf{A}^{(n)}$, input image $\mathbf{x}$
Output: mean vector $\boldsymbol{\mu}^{(n+1)}$, eigenvectors $\mathbf{U}^{(n+1)}$, coefficients $\mathbf{A}^{(n+1)}$,
    eigenvalues $\boldsymbol{\lambda}^{(n+1)}$

Project image $\mathbf{x}$ to current eigenspace:
    $\mathbf{a} = \mathbf{U}^{(n)T}\left(\mathbf{x} - \boldsymbol{\mu}^{(n)}\right)$
Reconstruct image:
    $\tilde{\mathbf{x}} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$
Compute residual vector:
    $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$
Append $\mathbf{r}$ as new basis vector to $\mathbf{U}$:
    $\mathbf{U}' = \begin{pmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{||\mathbf{r}||} \end{pmatrix}$
Determine the coefficients in the new basis:
    $\mathbf{A}' = \begin{pmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & ||\mathbf{r}|| \end{pmatrix}$
Perform PCA on $\mathbf{A}'$ and obtain the mean $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$ and
    the eigenvalues $\boldsymbol{\lambda}''$.
Project coefficients to new basis:
    $\mathbf{A}^{(n+1)} = \mathbf{U}''^{T}\left(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}\right)$
Rotate subspace:
    $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$
Update mean:
    $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$
Update eigenvalues:
    $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$
```

the robust approach (see Section 4.2.7) and the image is reconstructed. Second, outliers are detected by pixel-wise thresholding (based on the expected reconstruction error) the original image and its robust reconstruction. Finally, the outlying pixel values are replaced by the robustly reconstructed values.

The benefits of using this robust extension of the incremental approach is illustrated in Figure 12. For this purpose we used learned a shape model for a data set containing binary shapes of a bathing toy (i.e., an octopus). First, we trained an initial model using only 15 clean shapes to get a consistent starting model. Later on, the training is continued from corrupted data, where we have randomly added black and white bars occluding 25% of the image. By adding these shapes the non-robust model gets more and



(a)

(b)

Figure 11: Incremental PCA approach: (a) incremental PCA performs similar as incremental batch PCA, (b) incremental PCA is computationally much cheaper than incremental batch PCA.

| method | computation time |
|---|---|
| incremental PCA | 4.72s |
| batch PCA | 6.55s |
| iterated batch PCA | 205.38s |

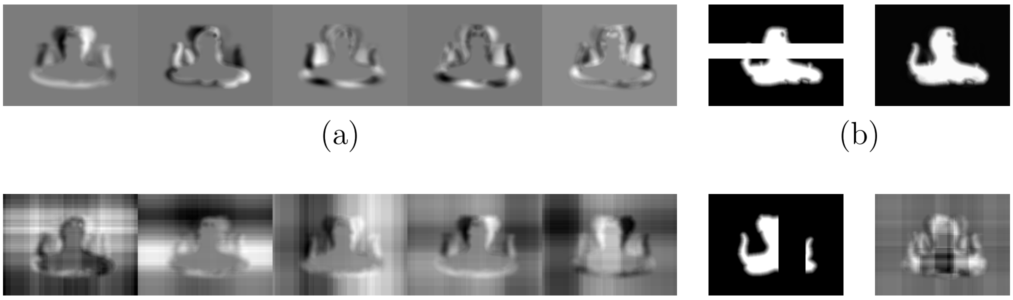
Table 4: Performance evaluation.

more corrupted (see (Figure 12(c) for the first 5 eigenimages) while a stable model is estimated by using the robust approach (see Figure 12(a) for the first 5 eigenimages). Examples of (robust) reconstructions are shown in Figure 12(b) (robust eigenspace) and Figure 12(d) (non-robust eigenspace), respectively.

## 4.3 Non-negative Matrix Factorization$^\star$

Non-negative matrix factorization (NMF) was originally proposed to model physical and chemical processes [99,115]. Later it was introduced by Lee and Seung [73] in Computer Vision for object representation. In contrast to PCA, non-negative matrix factorization does not allow negative entries whether in the basis nor in the encoding. As a result we obtain additive basis vectors mostly representing local structures. Thus, if the underlying data can be described by distinctive local information the representation may be sparse.

Formally, NMF can be described as follows. Given a non-negative matrix (i.e., a matrix containing vectorized images) $\mathbf{V} \in \mathbb{R}^{m \times n}$ the goal of NMF is



(a) (b)

(c) (d)

Figure 12: Robust incremental vs. plain incremental approach: (a) eigenspace obtained by robust incremental learning from noisy data, (b) test image ("over-segmented") and its reconstruction from robust eigenspace, (c) eigenspace obtained by incremental learning from noisy data, (d) test image ("under-segmented") and its reconstruction from non-robust eigenspace.

to find non-negative factors $\mathbf{W} \in \mathbb{R}^{n \times r}$ and $\mathbf{H} \in \mathbb{R}^{r \times m}$, that approximate the original data:

$$\mathbf{V} \approx \mathbf{WH} \,. \tag{64}$$

As two factors have to be estimated there does not exist a closed-form solution. Thus, both matrices, $\mathbf{W}$ and $\mathbf{H}$, have to be estimated in an iterative way. Therefore, we consider the optimization problem

$$\begin{aligned} \min ||\mathbf{V} - \mathbf{WH}||^2 \\ \text{s.t. } \mathbf{W}, \mathbf{H} > 0 \,, \end{aligned} \tag{65}$$

where $||.||^2$ denotes the squared Euclidean Distance. The optimization problem (65) can be iteratively solved by the following update rules:

$$\begin{aligned} \mathbf{H}_{a,j} &\leftarrow \mathbf{H}_{a,j} \frac{\left[\mathbf{W}^T\mathbf{V}\right]_{a,j}}{\left[\mathbf{W}^T\mathbf{WH}\right]_{a,j}} \\ \mathbf{W}_{i,a} &\leftarrow \mathbf{W}_{i,a} \frac{\left[\mathbf{VH}^T\right]_{i,a}}{\left[\mathbf{WHH}^T\right]_{i,a}} \,, \end{aligned} \tag{66}$$

where [.] denote that the multiplications and divisions are performed element by element. A more detailed derivation and description of the algorithm, also applying different objective functions, is given in [74, 134][10].

[10] MATLAB implementations for these different methods can be found at http://journalclub.mit.edu/jclub/publication?com_id=2;publication_id=21 (March 28, 2007).

To speed up the convergence and to ensure a global minimum (the optimization problem is not convex neither for $\mathbf{W}$ nor for $\mathbf{H}$) there were several extensions proposed, e.g., [45, 50], that additionally consider sparsity constraints and re-formulate the problem to a convex optimization problem.

## 4.4 Independent Component Analysis$^\star$

Independent Component Analysis (ICA) was originally introduced by Hérault, Jutten, and Ans [1, 46, 47] in the field of neurophysiology. But it became widely known and popular method not until it was introduced in signal processing for blind source separation, i.e., separation of mixed audio signals [22, 59]. This problem problem is often described by the task of identifying a single speaker in a group of speakers ("cocktail-party problem").

Thus, the goal is to express a set of $n$ random variables $x_1, \ldots, x_n$ as a linear combination of $n$ statistically independent random variables $s_j$:

$$x_j = a_{j,1}s_1 + \cdots + a_{j,n}s_n, \text{ for all } j \tag{67}$$

or written in matrix notation

$$\mathbf{x} = \mathbf{A}\mathbf{s} \, , \tag{68}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$, $\mathbf{s} = [s_1, \ldots, s_n]^T$, and $\mathbf{A}$ is the matrix containing the coefficients $a_{i,j}$. The goal of ICA is to estimate the original components $s_i$ or, equivalently, the coefficients $a_{i,j}$. By definition, the random variables $s_i$ are mutually independent and the mixing matrix is invertible. Based on this constraints the ICA problem can be formulated [19] as

$$\mathbf{u} = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{A}\mathbf{s} \, . \tag{69}$$

To compute these projections/mixing matrices several objective function were developed, that take into account the independence, e.g., based on the kurtosis, negentropy or mutual information. But for practical computations only computationally efficient methods such as InfoMax[11] [8] and Fast ICA[12] [52] should be applied. For a more detailed discussion on theory and application of ICA see [53].

To apply ICA for object recognition (face recognition) Bartlett et al. [5,25] proposed two architectures. For *architecture I* the images are considered to

[11] A MATLAB implementation can be found at: ftp://ftp.cnl.salk.edu/pub/tony/sep96.public (March 29th, 2007)

[12] A MATLAB and C++ implementation can be found at: http://www.cis.hut.fi/projects/ica/fastica/ (March 29th, 2007)

be a linear mixture of statistically independent basis images. In contrast, for *architecture II* the goal is to find statistically independent coefficients representing the input data. Therefore, the input data has to be transposed. For both architectures PCA is applied as a pre-processing step. In addition, whitening (see, e.g., [34, 52]) can be used to further reduce the computational complexity.

## 4.5 Canonical Correlation Analysis$^\star$

Canonical Correlation Analysis (CCA) was first introduced by Hotelling [49]. The goal is to find pairs of directions, that yield the maximum correlation between two random variables.

Formally, given two mean normalized random variables $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}_x$ and $\tilde{\mathbf{y}} = \mathbf{y} - \bar{\mathbf{x}}_y$ CCA is defined as the problem of finding a set of two basis vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ such that the correlation between the projections $x = \mathbf{W}_x^T \mathbf{x}$ and $y = \mathbf{W}_y^T \mathbf{y}$ is maximized. These are obtained by maximizing the correlation coefficient

$$\rho = \frac{\mathrm{E}\left[xy\right]}{\sqrt{\mathrm{E}\left[x^2\right]\mathrm{E}\left[y^2\right]}} = \frac{\mathrm{E}\left[\mathbf{w}_x^T x y^T \mathbf{w}_y\right]}{\sqrt{\mathrm{E}\left[\mathbf{w}_x^T x x^t \mathbf{w}_x\right]\mathrm{E}\left[\mathbf{w}_y^T y y^T \mathbf{w}_y\right]}} \; . \tag{70}$$

For practical computation we can re-write (70) by directly using the within-class covariance matrices $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ and the between-class covariance $\mathbf{C}_{xy}$:

$$\rho = \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y} \; . \tag{71}$$

The maximum $\rho$ with respect to $\mathbf{w}_x$ and $\mathbf{w}_y$ is the maximum canonical correlation. The projections onto $\mathbf{w}_x$ and $\mathbf{w}_y$, i.e., $x$ and $y$ are called *canonical factors*. To efficiently estimate the optimal solution and thus to obtained the canonical factors Borga [15] re-formulated the original optimization problem such that the canonical factors can be estimated from the Rayleigh quotient. A more efficient and numerical more stable solution was proposed by Melzer et al. [84], who applied SVD decomposition on the input data. Detailed discussions on derivation and calculation of CCA can be found in [15, 83].

## 4.6 Linear Discriminant Analysis$^\star$

If the training set is labeled, this additional information can also be used for subspace learning. To enable a more efficient classification the goal of Linear Fisher Discriminant Analysis (LDA) [30] is to maximize the distance between cluster centers. More formally, let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be $n$ samples, where each sample belongs to one of $c$ classes $\{X_1, \ldots, X_c\}$. LDA computes a classification function $g(x) = \mathbf{W}^T x$, where $\mathbf{W}$ is selected as the linear projection that minimizes within-class scatter

$$\mathbf{S}_B = \sum_{j=1}^{c} n_j \left(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}\right)\left(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}\right)^T \tag{72}$$

whereas it maximizes the between-class scatter

$$\mathbf{S}_W = \sum_{j=1}^{c} \sum_{x_k \in X_j} \left(\mathbf{x}_k - \bar{\mathbf{x}}_j\right)\left(\mathbf{x}_k - \bar{\mathbf{x}}_j\right)^T \ , \tag{73}$$

where $\bar{\mathbf{x}}$ is the mean over all samples, $\bar{\mathbf{x}}_j$ is the mean over class $j$, and $n_j$ is the number of samples in class $j$. In fact, this projection is obtained by maximizing the Fisher-criterion

$$\mathbf{W}_{opt} = \arg\max_{\mathbf{W}} \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right|} \ . \tag{74}$$

The optimal solution for this optimization problem is given by the solution of the generalized eigenproblem

$$\mathbf{S}_B \mathbf{w} = \boldsymbol{\lambda} \mathbf{S}_W \mathbf{w} \tag{75}$$

or directly by computing the eigenvectors for $\mathbf{S}_W^{-1}\mathbf{S}_B$.

The rank of $\mathbf{S}_W^{-1}\mathbf{S}_B$ is most $c-1$. Thus, for many practical application this matrix is singular and the eigenproblem can not be solved. This is referred as the small sample size problem. To overcome this problem several solutions were proposed, e.g, [7, 69, 140]. Moreover, there are several extensions of the standard approach such as robust classification [29], incremental LDA [131], or 2D representations (see Section 4.7.2).

## 4.7 Extensions of Subspace Methods

### 4.7.1 Kernel Subspace Methods

All of the methods discussed in the previous sections are linear. But they can easily be extended in a non-linear way. Therefor the input vectors are first projected onto a high-dimensional space by a non-linear mapping and then a linear methods is applied in this high dimensional space. For efficient computation the kernel trick (see, e.g., [23]) can be applied. Given a kernel-function

$$\mathrm{k}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y}) \tag{76}$$

and an algorithm, that can be formulated by using dot-products only in the high dimensional space. Then, we can replace the dot-products in the high dimensional space by kernel functions in the original lower-dimensional input space, i.e., all operations can directly be performed in the input space. Thus, for several linear subspace methods a kernel extension was proposed. These include Kernel PCA [112], Kernel LDA [113], Kernel ICA [4], Kernel CCA [84], and Kernel NMF [138]. More detailed discussions on kernel methods can be found in [23, 111, 114].

### 4.7.2 2D Subspace Methods$^\star$

For subspace methods usually the input images are vectorized and saved as one-dimensional vector in a matrix. As images, i.e., gray-value images, can naturally be seen as a matrix, it is straight forward to extend existing methods from a vector-based representation to a matrix representation. Hence, recently several authors proposed 2D methods for PCA (e.g., [70, 137]) and LDA (e.g., [70, 76]), but the same ideas can also be applied for other subspace methods. As advantages the authors argue that due to the matrix representation local structure information can be modeled in a more suitable way. Moreover, especially if the number of samples is quite large for PCA the computation of the covariance matrix is cheaper since the row-rank is much smaller. For LDA for the same reason the small sample size problem does not appear. But as a disadvantage, the encoding size, i.e., the representation in the subspace, is enlarged.

### 4.7.3 Binary Subspace Methods$^\star$

Subspace methods can be used to model shape models, e.g., by Active Shape Models [35], that are represented by binary data. It is clear that binary data can not be represented well by a model, that was developed for gray-value images. For instance, for PCA a Gaussian distribution of the data is assumed. Thus, inspired by the ideas of probabilistic PCA [125] binary PCA approaches were proposed [108, 141]. The main idea is to replace the Gaussian distribution in the probabilistic approach by a Bernulli distribution, that represents binary data more conveniently. Hence, a better model for binary images is obtained. But, as a disadvantage, an iterative process has to be run for both, the training and the evaluation.

# Appendix A Statistical Definitions

When working on (even simple) statistics one faces different definitions for the same parameters and data describing matrices. The reason for this confusion is that many authors do not distinguish between theoretic parameters defined by the distribution and parameters, that were estimated from the data (see Section Appendix B). Thus, in the following we give the exact definition of the most important statistical parameters.

### Mean

(a) Let $X$ be a continuous random variable and $f_X$ be a probability density function, then the *mean* or *expected value* of $X$ is given by

$$\mu = \mathrm{E}[X] = \int_{-\infty}^{\infty} x f_X(x) dx \ . \tag{77}$$

(b) Let $X$ be a discrete random variable and $\mathcal{X}$ be a finite set, then the *expected value* $\mu = \mathrm{E}[X]$ is given by

$$\mu = \mathrm{E}[X] = \sum_{x_j \in \mathcal{X}} x_j P(X = x_j) \ . \tag{78}$$

### Sample Mean

Given $n$ observations $x_1, \ldots, x_n$, then the *sample mean* as an estimate for the true mean, the expected value, is estimated by

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j \ . \tag{79}$$

### Variance

Let $X$ be a random variable, then the *variance* or the *second central moment* of $X$ is defined by

$$\sigma^2 = \mathrm{Var}[X] = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] = \mathrm{E}\left[X^2\right] - \mathrm{E}\left[X\right]^2 = \mathrm{E}\left[X^2\right] - \mu^2 \ . \tag{80}$$

### Sample Variance

Given $n$ observations $x_1, \ldots, x_n$ and the sample mean $\bar{x}$, then the *sample variance* as an estimate for the true variance is estimated by

$$s^2 = \frac{1}{n-1} \sum_{j=1}^{n} \left(x_j - \bar{x}\right)^2 \ . \tag{81}$$

If the true mean $\mu$ is known this estimate can be replaced by

$$s^2 = \frac{1}{n} \sum_{j=1}^{n} \left(x_j - \mu\right)^2 \ . \tag{82}$$

For larger $n$, in particular if $n \to \infty$ the difference vanishes. But for smaller $n$ it is important to choose the correct estimate for the variance (see Appendix Appendix B)!

### Covariance Matrix

Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$ and let $\boldsymbol{\mu} = \mathrm{E}\left(\mathbf{X}\right)$ be the expected value of $\mathbf{X}$. Let

$$\hat{\mathbf{x}}_j = \mathbf{x}_j - \boldsymbol{\mu} \tag{83}$$

be the mean normalized (centered) observations and $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\}$. Then, the *covariance matrix* $\mathbf{\Sigma}$ of $\mathbf{X}$ is defined by

$$\mathbf{\Sigma} = \mathrm{E}\left[\hat{\mathbf{X}}\hat{\mathbf{X}}^T\right] \ . \tag{84}$$

Following (79) it can be computed by

$$\mathbf{\Sigma} = \frac{1}{n}\hat{\mathbf{X}}\hat{\mathbf{X}}^T \ . \tag{85}$$

### Sample Covariance Matrix

Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$ and the sample mean vector $\bar{\mathbf{x}}$ estimated by (79). Let

$$\hat{\mathbf{x}}_j = \mathbf{x}_j - \bar{\mathbf{x}} \tag{86}$$

be the mean normalized (centered) observations and $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n\}$. Then, the *sample covariance matrix* of $\mathbf{X}$ is defined by

$$\mathbf{C} = \frac{1}{n-1}\hat{\mathbf{X}}\hat{\mathbf{X}}^T \ . \tag{87}$$

### Scatter Matrix

Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$ and $\hat{\mathbf{x}}_j$ estimated either by (83) or (86). Then, the *scatter matrix* is given by

$$\mathbf{S} = \hat{\mathbf{X}}\hat{\mathbf{X}}^T \ . \tag{88}$$

In fact, the scatter matrix can be considered the (sample) covariance matrix without the pre-multiplied norming factor $\frac{1}{n}$ or $\frac{1}{n-1}$. Thus, for technical reasons often the scatter matrix is addressed instead of the covariance matrix. Considering the eigenproblem, the same eigenvectors are obtained and the eigenvalues differ only by the pre-multiplied factor.

### Autocorrelation Matrix

Given a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$. Then, the *autocorrelation matrix* is given by

$$\hat{\mathbf{S}} = \frac{1}{n}\mathbf{X}\mathbf{X}^T \ . \tag{89}$$

If the columns of the matrix $\mathbf{X}$ are mean normalized the autocorrelation matrix of $\mathbf{X}$ becomes the covariance matrix of $\mathbf{X}$.

### Gram Matrix

Given matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ of $n$ observations $\mathbf{x}_j \in \mathbb{R}^m$. Then the *Gram matrix* is the matrix of all possible inner products of $\mathbf{X}$, i.e.,

$$\mathbf{g}_{i,j} = \mathbf{x}_i^T \mathbf{x}_j \ , \tag{90}$$

or written in matrix notation:

$$\mathbf{G} = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{n \times n} \ . \tag{91}$$

The Gram matrix can not be considered a statistical description of data. But due to its structure, in particular if the data is centered, important characteristics of the covariance matrix can be derived from the Gram matrix.

# Appendix B Statistical Parameter Estimation

For many theoretic discussion it is assumed that the parameters of a distribution are known, which is usually not the case in practice. Thus, these parameters have to be estimated from a set of given samples. In the following we will discuss how to obtain good estimates for the expected value

$\mu = \mathrm{E}\left[x\right]$, i.e., the sample mean, and the variance $\sigma^2 = \mathrm{E}\left[x^2\right] - \mathrm{E}\left[x\right]^2$, i.e., the sample variance. The quality of a estimate is characterized by two criteria, the *unbiasedness* and the *consistency*.

**Definition 1:** *An estimator $\Theta$ is an unbiased estimator for the parameter $\theta$ if*

$$\mathrm{E}\left[\Theta\right] = \theta \ . \tag{92}$$

**Definition 2:** *A sequence of estimators $\Theta_n$ for the parameter $\theta$ is said to be consistent if*

$$\lim_{n\to\infty} \mathrm{P}\left[\left|\Theta_n - \theta\right| \geq \varepsilon\right] = 0, \ \ \forall \varepsilon > 0 \ . \tag{93}$$

*This condition ensures that the sequence of estimators $\Theta_n$ converges to the true value $\theta$ if the number of samples is increased.*

**Theorem 1:** *Let $x_j$ be independent identical distributed random variables with $\mathrm{E}\left[x_j\right] = \mu$ and $\mathrm{Var}\left[x_j\right] = \sigma^2$. Then,*

*(a) the sample mean*

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j \tag{94}$$

*is an unbiased and consistent estimate for the expected value $\mu$*

*(b) and the sample variance*

$$s^2 = \frac{1}{n-1} \sum_{j=1}^{n} \left(x_j - \bar{x}\right)^2 \tag{95}$$

*is an unbiased estimate for the variance $\sigma^2$.*

*Proof.*

(a) To prove the unbiasedness of the sample mean we need to show that the expected value of the sample mean $\bar{x}$ is the expected value $\mu$. Therefore, we consider the linearity of the expected value:

$$\mathrm{E}\left[\bar{x}\right] = \mathrm{E}\left[\frac{1}{n} \sum_{j=1}^{n} x_j\right] = \frac{1}{n} \sum_{j=1}^{n} \mathrm{E}\left[x_j\right] = \frac{1}{n} \sum_{j=1}^{n} \mu = \mu \ . \tag{96}$$

To prove the consistency we first estimate the variance of the sample mean:

$$\operatorname{Var}[\bar{x}] = \operatorname{Var}\left[\frac{1}{n}\sum_{j=1}^{n} x_j\right] = \frac{1}{n}\sum_{j=1}^{n}\operatorname{Var}[x_j] = \frac{\sigma^2}{n}\ . \tag{97}$$

From Chebyshev's inequality we get

$$\mathrm{P}\left[|\bar{x} - \mathrm{E}[\bar{x}]| \geq \varepsilon\right] \leq \frac{1}{\varepsilon^2}\operatorname{Var}[\bar{x}] = \frac{\sigma^2}{n\varepsilon^2} \to 0,\ \text{ for } n \to \infty\ . \tag{98}$$

Hence, we have proved that $\bar{x}$ is a consistent and unbiased estimate for $\mu$.

(b) The expected value for $s^2$ is given by

$$\mathrm{E}\left[s^2\right] = \mathrm{E}\left[\frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})^2\right]\ . \tag{99}$$

Due to the linearity of the expected value we can re-write (99) to

$$\begin{aligned}
&\frac{1}{n-1}\sum_{j=1}^{n}\mathrm{E}\left[(x_j-\bar{x})^2\right] = \\
&= \frac{1}{n-1}\sum_{j=1}^{n}\left(\mathrm{E}\left[x_j^2\right] - 2\,\mathrm{E}[x_j\bar{x}] + \mathrm{E}\left[\bar{x}^2\right]\right) = \\
&= \frac{1}{n-1}\sum_{j=1}^{n}\left(\mu^2+\sigma^2 - \frac{2}{n}\mathrm{E}\left[x_j\sum_{k=1}^{n}x_k\right] + \mu^2 + \frac{\sigma^2}{n}\right) = \\
&= \frac{1}{n-1}\sum_{j=1}^{n}\left(2\mu^2 + \left(1+\frac{1}{n}\right)\sigma^2 - \frac{2}{n}\mathrm{E}\left[x_j^2\right] - \frac{2}{n}\sum_{j\neq k}\mathrm{E}[x_jx_k]\right) = \\
&= \frac{1}{n-1}\sum_{j=1}^{n}\left(2\mu^2 + \left(1+\frac{1}{n}\right)\sigma^2 - \frac{2}{n}\left(\mu^2+\sigma^2\right) - 2\frac{n-1}{n}\mu^2\right) = \\
&= \frac{1}{n-1}\sum_{j=1}^{n}\left(\frac{n-1}{n}\sigma^2 - \frac{2}{n}\mu^2 + \frac{2}{n}\mu^2\right) = \sigma^2\ .
\end{aligned} \tag{100}$$

Hence, we get that $\mathrm{E}[s^2] = \sigma^2$ and that $s^2$ is an unbiased estimate for $\sigma^2$.

□

From Theorem 1 (2) it is clear that for practical applications, where the expected value is estimated from the data, (81) should be used to estimate the sample variance. In fact, the the sum of squared differences should be pre-multiplied by $\frac{1}{n-1}$ rather than by $\frac{1}{n}$. The same also applies in case of the covariance matrix! For a more detailed discussion on parameter estimation see, e.g., [34, 119].

# Appendix C Singular Value Decomposition

## C.1 Introduction

In this section, we discuss the main properties of SVD and its application in Computer Vision. Here, SVD is often used to solve the eigenproblem for the covariance matrix or, in robust statistics, to solve overdetermined systems of equations. As the underlying data represents (gray-value) images the discussion is reduced to real-valued matrices.

Singular Value Decomposition (SVD) is a powerful and thus widely used method in Linear Algebra. Starting in the second half of the 19th century several authors (Beltrami [10], Jordan [58] and Sylvester [122]) co-discovered SVD for real, square and non-singular matrices. Later Autonne [3] and Eckart and Young [28] extended it to complex matrices and to rectangular matrices, respectively. But it took until the 1970s when Golub and Kahan [37] and Golub and Reinsch [38] developed algorithms, that are suitable for practical use. In fact, most of the modern SVD-algorithms are based on the method of Golub and Reinsch[13]. More general discussions (overview of methods and application in engineering) are given in [67, 121], for numerical details see [39, 102].

## C.2 Eigenvalue Decomposition

Given a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, a scalar $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ if there exists a vector $\mathbf{v} \in \mathbb{R}$, $\mathbf{v} \neq \mathbf{0}$ such that

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}, \tag{101}$$

where $\mathbf{x}$ is referred to as eigenvector. More precisely, $\mathbf{x}$ is a right eigenvector if $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ and a left eigenvector if $\mathbf{x}^T\mathbf{A} = \lambda\mathbf{x}^T$. Usually, by eigenvector a

[13] The method is based on a Householder transform followed by a $QR$ decomposition. Most newer SVD implementation just use a more efficient variant for the $QR$ decomposition.

right eigenvector is meant. The total set of eigenvalues $\lambda(\mathbf{A}) = \{\lambda_1, \ldots, \lambda_n\}$ is called the spectrum of $\mathbf{A}$.

To compute the eigenvalues $\lambda_i$ and the eigenvectors $\mathbf{x}_i$ (101) is re-written as

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0} \; , \tag{102}$$

which is a homogeneous linear system. From Cramer's Theorem (see, e.g., [71]) we get that a homogeneous system has a non-trivial solution $\mathbf{x} \neq \mathbf{0}$ if the determinant of the coefficients is zero:

$$det(\mathbf{A} - \lambda\mathbf{I}) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} - \lambda \end{vmatrix} = 0 \; . \tag{103}$$

By developing (103) we obtain a polynomial

$$p(\lambda) = \alpha_0 + \alpha_1\lambda + \ldots + \alpha_{n-1}\lambda^{n-1} + \alpha_n\lambda^n = 0 \tag{104}$$

of $n$-th order in $\lambda$, which is referred to as the *characteristic polynomial* of $\mathbf{A}$. Hence, the eigenvalues $\lambda_i$ of $\mathbf{A}$ can be estimated by computing the roots of the characteristic polynomial $p(\lambda)$. For larger $n$ direct computation of the determinant is not appropriate. Thus, a numerical approximation is used to solve (103). Once the eigenvalues $\lambda_i$ are determined the corresponding eigenvectors $\mathbf{x}_i$ can be computed by solving (102), which can simply be done by a Gauss elimination.

When developing (103) we get the following special coefficients (see [85] for a proof):

$$\begin{array}{rcl} \alpha_0 & = & \det(\mathbf{A}), \\ \alpha_{n-1} & = & (-1)^{n-1}(a_{11} + \ldots + a_{nn}), \text{ and} \\ \alpha_n & = & (-1)^n. \end{array} \tag{105}$$

From (105) we get the trace of a matrix $\mathbf{A}$

$$\mathrm{tr}(\mathbf{A}) = \sum_{j=1}^{n} a_{jj} \tag{106}$$

, which can also be determined by

$$\mathrm{tr}(\mathbf{A}) = \sum_{j=1}^{n} \lambda_j \; . \tag{107}$$

Moreover, the determinant of a matrix $\mathbf{A}$ is determined by

$$\det(\mathbf{A}) = \prod_{j=1}^{n} \lambda_j \; . \tag{108}$$

Considering the solution of the characteristic polynomial we can define the multiplicity of an eigenvalue. If the eigenvalue $\lambda$ is root of order $m_a$ of the characteristic polynomial $m_a(\lambda)$ is called the *algebraic multiplicity* of $\lambda$. On the other hand $m_g(\lambda)$, which is defined as the number of linearly independent eigenvectors corresponding to $\lambda$, is referred as to the *geometric multiplicity* of $\lambda$. In general, $m_g(\lambda) \leq m_a(\lambda)$. If $m_g(\lambda) < m_a(\lambda)$ the eigenvalue $\lambda$ is said to be a defective eigenvalues. A matrix having at least one *defective eigenvalues* is referred as to defective matrix. This is important as only non-defective matrices are diagonalizable, which is important for many numerical computations.

**Theorem 2:** *Given a diagonalizable matrix $\boldsymbol{A} \in I\!\!R^{m\times m}$ there exist a non-singular matrix $\boldsymbol{T}$ such that*

$$\boldsymbol{A} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^{-1} \; , \tag{109}$$

*where $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$. The decomposition (109) is also referred to as eigenvalue decomposition (EVD).*

*Proof.* For a proof see, e.g., [39]. □

## C.3 Singular Value Decomposition

**Theorem 3:** *If $\boldsymbol{A}$ is a real $m{\times}n$-matrix, then there exist two orthogonal matrices*

$$\boldsymbol{U} \in I\!\!R^{m\times m} = (u_1, \ldots, u_m) \qquad \textit{and} \qquad \boldsymbol{V} \in I\!\!R^{n\times n} = (v_1, \ldots, v_n)$$

*such that*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \; , \tag{110}$$

*where*

$$\boldsymbol{\Sigma} \in I\!\!R^{m\times n} = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$

*and*

$$\boldsymbol{S} \in I\!\!R^{m\times m} = \mathrm{diag}(\sigma_1, \ldots, \sigma_p), \quad p = \min\{m, n\}$$

*with*

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0 \ .$$

*The decomposition (110) is referred as to Singular Value Decomposition (SVD).*

*Proof.* A proof can be found in [39]. □

**Remark 1:** The diagonal elements $\sigma_i$ of $\mathbf{\Sigma}$ are referred to as singular values. The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the $i$th left singular vector and the $i$th right singular vector, respectively. The left singular vectors are the orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$ and the right singular vectors are the orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^T$. Thus, the singular values $\sigma_i$ of $\mathbf{A}$ are the positive square roots of $\mathbf{A}^T\mathbf{A}$. A proof/derivation can be found in [85].

**Remark 2:** Theorem 3 also holds for complex matrices: If $\mathbf{A} \in \mathbb{C}^{m\times n}$, then there exist two unitary matrices[14] $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ and a pseudo-diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ such that $\mathbf{A} = \mathbf{U\Sigma V}^H$.

**Remark 3:** By comparing the columns of $\mathbf{AV} = \mathbf{U\Sigma}$ and $\mathbf{A}^T\mathbf{U} = \mathbf{V\Sigma}^T$ it is easy to verify that

$$\left.\begin{array}{rcl} \mathbf{A}v_j &=& \sigma_i u_j \\ \mathbf{A}^T u_j &=& \sigma_i v_j \end{array}\right\} j = 1, \ldots, p \ . \tag{111}$$

## C.4 Applications of SVD

Since there exist computational efficient algorithms SVD is widely applied for many different tasks. These include

- computing the Moore-Penrose Inverse,
- numerically estimating the rank of a matrix,
- solving a linear system in the least square sense,
- approximating a matrix by a matrix of lower rank,
- solving the eigenproblem for real symmetric matrices, and
- orthogonalizing matrices.

[14] A matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is unitary if $\mathbf{A}^H\mathbf{A} = \mathbf{A}\mathbf{A}^H = \mathbf{I}_n$. $\mathbf{A}^H$ denotes the Hermitian matrix of $\mathbf{A}$.

In fact, when looking at the source code of many MATLAB routines, e.g., to compute the pseudo inverse (*pinv*), to orthogonalize a matrix (*orth*) or to compute the rank of a matrix (*rank*) it can be seen that all of these methods implicitly call the SVD sub-routine.

## C.5 Pseudoinverse

A *pseudoinverse* can be seen as a generalization of an inverse of a matrix for singular and rectangular matrices. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. A matrix $\mathbf{A}^{+} \in \mathbb{R}^{nxm}$ is a pseudoinverse of $\mathbf{A}$ if

$$\begin{aligned} \mathbf{A}\mathbf{A}^{+}\mathbf{A} &= \mathbf{A} \\ \mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} &= \mathbf{A}^{+} \,. \end{aligned} \tag{112}$$

From this definition many possible pseudo inverses may be estimated. In fact, the general defined pseudoinverse is not unique. The class of pseudo inverses defined by (112) is known as *matrix 1-inverse*. A particular type of pseudoinverses is the *Moore-Penrose Inverse* [93, 101]. The Moore-Penrose Inverse $\mathbf{A}^{\dagger}$ of a matrix $\mathbf{A}$ is the unique solution, that fulfills the four Penrose equations [39, 85]:

$$\begin{aligned} \mathbf{A}\mathbf{A}^{\dagger}\mathbf{A} &= \mathbf{A} \\ \left(\mathbf{A}\mathbf{A}^{\dagger}\right)^{T} &= \mathbf{A}\mathbf{A}^{\dagger} \\ \mathbf{A}^{\dagger}\mathbf{A}\mathbf{A}^{\dagger} &= \mathbf{A}^{\dagger} \\ \left(\mathbf{A}^{\dagger}\mathbf{A}\right)^{T} &= \mathbf{A}^{\dagger}\mathbf{A} \,. \end{aligned} \tag{113}$$

From (112) and (113) it is clear that every Moore-Penrose Inverse is a pseudoinverse. But this does not hold conversely [103]. Thus, we get:

**Theorem 4:** *Every matrix has a unique Moore Penrose Inverse* $\boldsymbol{A}^{\dagger}$*.*

*Proof.* The proof directly follows from the definition (113). □

**Remark 4:** Many authors do not distinguish between (112) and (113). Thus, usually the term “pseudoinverse” is used even though actually the unique “Moore-Penrose Inverse” is meant.

**Theorem 5:** *If the inverse of* $\boldsymbol{A}^{T}\boldsymbol{A}$ *exists then the Moore-Penrose Inverse is given by*

$$\boldsymbol{A}^{\dagger} = \left(\boldsymbol{A}^{T}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{T} \,.$$

*Proof.* Consider the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{c} \,.$$

To obtain a square matrix we pre-multiply both sides with $\mathbf{A}^T$. Hence, we get

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{c}$$

and we can solve the system for $\mathbf{x}$:

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{c} \ .$$

□

In general, similar to the inverse of a matrix the Moore-Penrose Inverse can be estimated from a matrix decomposition, i.e., from the SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

$$\mathbf{A}^\dagger = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^T \ . \tag{114}$$

## C.6 SVD to Solve EVD

From Remark 1 it is clear that there is a strong relationship between eigenvalues and eigenvectors on the one hand and singular values and singular vectors on the other hand. In fact, EVD and SVD are equivalent for real symmetric matrices[15].

**Theorem 6:** *For a real symmetric matrix* $\boldsymbol{A}$ *SVD is equivalent to EVD.*

*Proof.* Let

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{115}$$

be the SVD of $\mathbf{A}$. Since $\mathbf{A}$ is a symmetric matrix, $\mathbf{U} = \mathbf{V}$, and we get

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{u}_j^T \ . \tag{116}$$

By right multiplying the left and the right term of (116) by $\mathbf{u}_j$ we get that

$$\mathbf{A}\mathbf{u}_j = \sigma_j \mathbf{u}_j \ . \tag{117}$$

Hence, $\sigma_j$ and $\mathbf{u}_j$ are an eigenvalue and the corresponding eigenvector of $\mathbf{A}$.

[15] A matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$ is said to be symmetric if $\mathbf{A}^T = \mathbf{A}$.

Since $\mathbf{U}$ is an orthogonal matrix, $\mathbf{U}^{-1} = \mathbf{U}^T$, and we can re-write (116) as

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{-1} \, , \tag{118}$$

which is equivalent to (109).

□

**Remark 5:** From Theorem 6 we get that real symmetric matrices have the following properties:

(a) All eigenvalues $\lambda_j$ are real.

(b) The eigenvectors $\mathbf{u}_j$ of distinct eigenvalues $\lambda_j$ are orthonormal.

(c) The inverse $\mathbf{A}^{-1}$ is a real and symmetric matrix having the same eigenvectors as $\mathbf{A}$ while the eigenvalues are reciprocal.

**Remark 6:** Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ then $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$

(a) are symmetric positive semi-definite matrices,

(b) have the same rank, and

(c) share the same non-zero eigenvalues.

*Proof.* Consider the SVD

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \tag{119}$$

Then,

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\boldsymbol{\Sigma}\underbrace{\mathbf{V}^T\mathbf{V}}_{\mathbf{I}_{m\times m}}\boldsymbol{\Sigma}^T\mathbf{U} = \mathbf{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\mathbf{U}^T \tag{120}$$

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\boldsymbol{\Sigma}^T\underbrace{\mathbf{U}^T\mathbf{U}}_{\mathbf{I}_{n\times n}}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\mathbf{V}^T \, . \tag{121}$$

Thus, we get that the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$. Similarly, the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$. Additionally, the non-zero singular values for both matrices are given by $\sigma_j^2$. Thus, we proved (b) and (c). The symmetry for both matrices directly follows from the matrix multiplications. In addition, since all eigenvalues $\lambda_j = \sigma_j^2 \geq 0$ the matrix $\mathbf{A}$ is positive semi-definite and we finally proved (a).

□

# References

[1] Bernard Ans, Jeanny Hérault, and Christian Jutten. Adaptive neural architectures: detection of primitives. In *Proc. of COGNITIVA*, pages 593–597, 1985.

[2] Ilkka Autio. Using natural class hierarchies in multi-class visual classification. *Pattern Recognition*, 39(7):1290–1299, 2006.

[3] Léon Autonne. Sur les matrices hypohermitiennes et les unitairs. *Comptes Rendus de l'Academie des Sciences*, 156:858–860, 1913.

[4] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[5] Marian Stewart Bartlett, Javier R. Movellan, and Terrence J. Sejnowski. Face recognition by independent component analysis. *IEEE Trans. on Neural Networks*, 13(6):1450–64, 2002.

[6] Adam Baumberg. Reliable feature matching across widely separated views. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 774–781, June 2000.

[7] Peter N. Belhumeur, Joao Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[8] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[9] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[10] Eugenio Beltrami. Sulle funzioni bilineari. In *Giornale di Matematiche ad Uso delgi Studenti Della Universita*, volume 11, pages 98–106, 1873.

[11] Paul Besl and Ramesh Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[12] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. European Conf. on Computer Vision*, pages 329–342, 1996.

[15] Magnus Borga. *Learning Multidimensional Signal Processing*. PhD thesis, Linköping University, Department of Electrical Engineering, 1998.

[16] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proc. European Conf. on Computer Vision*, volume I, pages 707–720, 2002.

[17] Matthew Brown and David Lowe. Invariant features from interest point groups. In *Proc. British Machine Vision Conf.*, pages 656–665, 2002.

[18] John F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[19] Jean-François Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters*, 4(4):112–114, 1997.

[20] Gustavo Carneiro and Allan D. Jepson. Phase-based local features. In *Proc. European Conf. on Computer Vision*, volume 1, pages 282–296, 2002.

[21] Shivkumar Chandrasekaran, B. S. Manjunath, Yuan-Fang Wang, Jay Winkeler, and Henry Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, 1997.

[22] Pierre Comon. Independent component analysis - a new concept? *Signal Processing*, 36:287–314, 1994.

[23] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[24] Kostas I. Diamantaras and Sun-Yuan Kung. *Principal Component Neural Networks: Theory and Applications*. John Wiley & Sons, 1996.

[25] Bruce A. Draper, Kyungim Baek, Marian Stewart Bartlett, and J. Ross Beveridge. Recognizing faces with pca and ica. *Computer Vision and Image Understanding*, 93(1):115–137, 2003.

[26] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons, 2000.

[27] Yves Dufournaud, Cordelia Schmid, and Radu Horaud. Matching images with different resolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 612–618, 2000.

[28] Carl Eckart and Gale Young. A principal axis transformation for non-hermitian matrices. *Bulletin of the American Mathematical Society*, 45:118–121, 1939.

[29] Sanja Fidler, Danijel Skočaj, and Aleš Leonardis. Combining reconstructive and discriminative subspace methods for robust classification and regression by subsampling. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(3):337–350, 2006.

[30] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[31] Friedrich Fraundorfer and Horst Bischof. Evaluation of local detectors on non-planar scenes. In *Proc. Workshop of the Austrian Association for Pattern Recognition*, 2004.

[32] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[33] Yoav Freund and Robert E. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[34] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1999.

[35] Michael Fussenegger, Peter M. Roth, Horst Bischof, and Axel Pinz. On-line, incremental learning of a robust active shape model. In *Proc. DAGM Symposium*, pages 122–131, 2006.

[36] Jan J. Gerbrands. On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 14(1-6):375–381, 1981.

[37] Gene H. Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2:205–224, 1965.

[38] Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solution. *Numerische Mathematik*, 14:403–420, 1970.

[39] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. John Hopkins University Press, 1996.

[40] Peter Hall, David Marshall, and Ralph Martin. Incremental eigenanalysis for classification. In *Proc. British Machine Vision Conf.*, volume I, pages 286–295, 1998.

[41] Peter Hall, David Marshall, and Ralph Martin. Merging and splitting eigenspace models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049, 2000.

[42] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, Werner A., and Stahel. *Robust Statistics*. John Wiley & Sons, 1986.

[43] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. of Alvey Vision Conf.*, pages 147–151, 1988.

[44] Marko Heikkilä and Matti Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(4):657–662, 2006.

[45] Matthias Heiler and Christoph Schnörr. Learning non-negative sparse image codes by convex programming. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 1667–1674, 2005.

[46] Jeanny Hérault, Bernard Ans, , and Christian Jutten. Circuits neuronaux à synapses modifiables: Décodage de messages composites par apprentissage non supervisé. *Comptes Rendus de l'Acadmie des Sciences*, 299(III-13):525–528, 1984.

[47] Jeanny Hérault, Christian Jutten, and Bernard Ans. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In *Actes du Xéme colloque GRETSI*, pages 1017–1022, 1985.

[48] Harold Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[49] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[50] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[51] Peter J. Huber. *Robust Statistics*. John Wiley & Sons, 2004.

[52] Aapo Hyvärinen. The fixed-point algorithm and maximum liklihood estimation for independent component analysis. *Neural Processing Letters*, 10(1):1–5, 1999.

[53] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

[54] Laurent Itti and Christof Koch. Computational modeling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001.

[55] Omar Javed, Saad Ali, and Mubarak Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 696–701, 2005.

[56] Andrew E. Johnson and Martial Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-d scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[57] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[58] Camille Jordan. Sur la réduction des formes bilinéaires. *Comptes Rendus de l'Académie des Sciences*, 78:614–617, 1874.

[59] Christian Jutten and Jeanny Herault. Blind separation of sources, part I. *Signal Processing*, 24(1):1–10, 1991.

[60] Timor Kadir, Djamal Boukerroui, and Michael Brady. An analysis of the scale saliency algorithm. Technical report, Robotics Research Laboratory, Department of Engineering Science, University of Oxford, 2003.

[61] Timor Kadir and Michael Brady. Saliency, scale and image description. *Intern. Journal of Computer Vision*, 45(2):83–105, 2001.

[62] Timor Kadir and Michael Brady. Scale saliency : A novel approach to salient feature and scale selection. In *Intern. Conf. on Visual Information Engineering*, pages 25–28, 2003.

[63] Timor Kadir, Andrew Zisserman, and Michael Brady. An affine invariant salient region detector. In *Proc. European Conf. on Computer Vision*, pages 228–241, 2004.

[64] Kari Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung. *Annales Academiae Scientiarum Fennicae, Series A1: Mathematica-Physica*, 37:3–79, 1947.

[65] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. Technical Report IRP-TR-03-15, School of Computer Science, Carnegie Mellon University and IntelResearch Pittsburgh, 2003.

[66] Michael Kirby and Lawrence Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[67] Virginia C. Klema and Alan J. Laub. The singular value decomposition: Its computation and some applications. *IEEE Trans. on Automatic Control*, 25(2):164–176, 1980.

[68] Jan Koenderink and A.J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[69] Hui Kong, Eam Khwang Teoh, Jian Gang Wang, and Ronda Venkateswarlu. Two dimensional fisher discriminant analysis: Forget about small sample size problem. In *Proc. IEEE Intern. Conf. on Acoustics, Speech, and Signal Processing*, volume II, pages 761–764, 2005.

[70] Hui Kong, Lei Wang, Eam Khwang Teoh, Xuchun Li, Jian-Gang Wang, and Ronda Venkateswarlu. Generalized 2d principal component analysis for face image representation and recognition. In *Proc. IEEE Joint Intern. Conf. on Neural Networks*, volume I, pages 108–113, 2005.

[71] Erwin Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 1993.

[72] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using affine-invariant regions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 319–324, 2003.

[73] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[74] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

[75] Aleš Leonardis and Horst Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000.

[76] Ming Li and Baozong Yuan. 2d-lda: A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters*, 26(5):527–532, 2005.

[77] Yongmin Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.

[78] Tony Lindeberg. Feature detection with automatic scale selection. *Intern. Journal of Computer Vision*, 30(2):77–116, 1998. Technical report ISRN KTH NA/P–96/18–SE.

[79] Michel Loève. Fonctions aléatoires du second ordre. *Processus stochastiques et mouvements Browniens*, 1948.

[80] David Lowe. Object recognition from local scale-invariant features. In *Proc. IEEE Intern. Conf. on Computer Vision*, pages 1150–1157, 1999.

[81] David Lowe. Distinctive image features from scale-invariant keypoints. *Intern. Journal of Computer Vision*, 60:91–110, 2004.

[82] Jiri Matas, Ondrei Chum, U. Martin, and Tomas Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. British Machine Vision Conf.*, volume I, pages 384–393, 2002.

[83] Thomas Melzer. *Generalized Canonical Correlation Analysis for Object Recognition*. PhD thesis, Vienna University of Technology, 2002.

[84] Thomas Melzer, Michael Reiter, and Horst Bischof. Appearance models based on kernel canonical correlation analysis. *Pattern Recognition*, 36(9):1961–1971, 2003.

[85] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001.

[86] Krystian Mikolajczk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume I, pages 525–531, 2001.

[87] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proc. European Conf. on Computer Vision*, volume I, pages 128–142, 2002.

[88] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 257–263, 2003.

[89] Krystian Mikolajczyk and Cordelia Schmid. Comparison of affine-invariant local detectors and descriptors. In *Proc. European Signal Processing Conf.*, 2004.

[90] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[91] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, TimorKadir, and Luc Van Gool. A comparison of affine region detectors. *Intern. Journal of Computer Vision*, 65(1-2):43–72, 2005.

[92] F. Mindru, T. Moons, and Luc Van Gool. Recognizing color patterns irrespective of viewpoint and illumination. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 368–373, 1999.

[93] Eliakim H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:394–395, 1920.

[94] H. Murakami and Vijaya Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 4(5):511–515, 1982.

[95] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *Intern. Journal of Computer Vision*, 14(1):5–24, 1995.

[96] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[97] Timo Ojala, Matti Pietikainen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

[98] Timo Ojala, Matti Pietikinen, and Topi Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[99] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[100] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Sciences*, 6(2):559–572, 1901.

[101] Roger Penrose. A generalized inverse for matrices. *Proc. of the Cambridge Philosophical Society*, 51:406–413, 1955.

[102] William H. Press, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993. On-line available under http://www.nrbook.com/a/bookcpdf.php.

[103] Calyampudi R. Rao and Sujit K. Mitra. *Generalized inverse of matrices and its applications*. John Wiley & Sons, 1971.

[104] Rajesh Rao. Dynamic appearance-based recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 540–546, 1997.

[105] Peter M. Roth. *On-line Conservative Learning*. PhD thesis, Graz University of Technology, Faculty of Computer Science, 2008.

[106] Sam Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, pages 626–632, 1997.

[107] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or how do i organize my holiday snaps? In *Proc. European Conf. on Computer Vision*, volume I, pages 414–431, 2002.

[108] Andrew I. Schein, Lawrence K. Saul, and Lyle H. Ungar. A generalized linear model for principal component analysis of binary data. In *Intern. Workshop on Artificial Intelligence and Statistics*, 2003.

[109] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[110] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *Intern. Journal of Computer Vision*, 37(2):151–172, 2000.

[111] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

[112] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[113] Gunnar Rätsch Sebastian Mika, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller. Fisher discriminant analysis with kernels. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 41–48, 1999.

[114] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[115] J. Shen and G. W. Israël. A receptor model using a specific non-negative transformation technique for ambient aerosol. *Atmospheric Environment*, 23(10):2289–2298, 1989.

[116] Danijel Skočaj. *Robust subspace approaches to visual learning and recognition*. PhD thesis, University of Ljubljana, Faculty of Computer and Information Science, 2003.

[117] Danijel Skočaj, Horst Bischof, and Aleš Leonardis. A robust PCA algorithm for building representations from panoramic images. In *Proc. European Conf. on Computer Vision*, volume IV, pages 761–775, 2002.

[118] Danijel Skočaj and Aleš Leonardis. Weighted and robust incremental method for subspace learning. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 1494–1501, 2003.

[119] Tsu-Teh Soong. *Fundamentals of Probability and Statistics for Engineers*. John Wiley & Sons, 2004.

[120] Sabine Sternig. Object recognition with locally binary patterns. Bachelor's Thesis at Graz University of Technology, 2005.

[121] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566, 2003.

[122] James J. Sylvester. On the reduction of a bilinear quantic of the nth order to the form of a sum of n products by a double orthogonal substitution. *Messenger of Mathematics*, 19:42–46, 1889.

[123] Valtteri Takala, Timo Ahonen, and Matti Pietikainen. Block-based methods for image retrieval using local binary patterns. In *Proc. Scandinavian Conf. on Image Analysis*, pages 882–891, 2005.

[124] Michael J. Tarr, Pepper Williams, William G. Hayward, and Isabel Gauthier. Three-dimensional object recognition is viewpoint dependent. *Nature Neuroscience*, 1(4):275–277, 1998.

[125] Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 61:611–622, 1999.

[126] Fernando de la Torre and Michael J. Black. Robust principal component analysis for computer vision. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume I, pages 362–369, 2001.

[127] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[128] Tinne Tuytelaars and Luc Van Gool. Content-based image retrieval based on local affinely invariant regions. In *Intern. Conf. on Visual Information and Information Systems*, pages 493–500, 1999.

[129] Tinne Tuytelaars and Luc Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *Proc. British Machine Vision Conf.*, pages 412–422, 2000.

[130] Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *Intern. Journal of Computer Vision*, 1(59):61–85, 2004.

[131] Martina Uray, Danijel Skočaj, Peter M. Roth, Horst Bischof, and Aleš Leonardis. Incremental LDA learning by combining reconstructive and discriminative approaches. In *Proc. British Machine Vision Conf.*, volume I, pages 272–281, 2007.

[132] Luc Van Gool, T. Moons, and D. Ungureanu. Affine/ photometric invariants for planar intensity patterns. In *Proc. European Conf. on Computer Vision*, volume 1, pages 642–651, 1996.

[133] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[134] Stefan Wild, James Curry, and Anne Dougherty. Motivating nonnegative matrix factorizations. In *Proc. SIAM Applied Linear Algebra Conf.*, 2003.

[135] Martin Winter. *Spatial Relations of Features and Descriptors for Appearance Based Object Recognition*. PhD thesis, Graz University of Technology, Faculty of Computer Science, 2007.

[136] Lei Xu and Alan L. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. on Neural Networks*, 6(1):131–143, 1995.

[137] Jian Yang, David Zhang, Alejandro F. Frangi, and Jing yu Yang. Two-dimensional pca: A new approach to appearance-based face representation and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(1):131–137, 2004.

[138] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Non-negative matrix factorization on kernels. In *Proc. Pacific Rim Intern. Conf. on Artificial Intelligence*, pages 404–412, 2006.

[139] Hongming Zhang, Wen Gao, Xilin Chen, and Debin Zhao. Object detection using spatial histogram features. *Image and Vision Computing*, 24(4):327–341, April 2006.

[140] Xiao-Sheng Zhuang and Dao-Qing Dai. Inverse fisher discriminate criteria for small samples size problem and its application to face recognition. *Pattern Recognition*, 38(11):2192–2194, 2005.

[141] Zoran Zivkovic and Jakob Verbeek. Transformation invariant component analysis for binary images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 254–259, 2006.