Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Danijel Skočaj

Robustni pristopi k vizualnemu učenju in razpoznavanju na osnovi podprostorov

DOKTORSKA DISERTACIJA

Ljubljana, 2003

Mentor: prof. dr. Aleš Leonardis

University of Ljubljana

Faculty of computer and information science

Danijel Skočaj

Robust Subspace Approaches to Visual Learning and Recognition

doctoral dissertation

Ljubljana, 2003

Supervisor: prof. dr. Aleš Leonardis

Robustni pristopi k vizualnemu učenju in razpoznavanju na osnovi podprostorov

Danijel Skočaj

Mentor: Aleš Leonardis

Povzetek

Vizualno učenje mora biti robusten in kontinuiran proces. Vsi razpoložljivi vizualni podatki niso enako pomembni; v primeru prekrivanj in ostalih nezaželenih motenj v vidnem polju so lahko nekateri celo zavajajoči. Človeški vizualni sistem obravnava vizualne podatke selektivno in zgradi učinkovite predstavitve opazovanih predmetov in scen tudi v neidealnih pogojih. Te predstavitve lahko nato še posodablja z na novo pridobljenimi informacijami in jih tako prilagaja spremembam. V doktorski disertaciji proučimo ta načela in predlagamo več metod, ki uvedejo podobne principe tudi na področje strojnega vizualnega učenja in razpoznavanja.

Vizualno učenje je realizirano z modeliranjem osnovanim na izgledu predmetov in scen. Gradnja modelov temelji na metodi glavnih komponent (PCA), ki pa ima v svoji standardni izvedbi nekaj pomanjkljivosti, ki onemogočajo uveljavitev prej omenjenih načel. Za premostitev teh pomanjkljivosti je v disertaciji predlaganih več razširitev standardne metode glavnih komponent.

Standardna metoda PCA obdeluje vse učne slike hkrati in tako zahteva, da so vse slike podane vnaprej. Ker to ne ustreza zahtevam kontinuiranega učenja, predlagamo metodo za inkrementalno učenje, ki obdela slike zaporedno drugo za drugo in na vsakem koraku ustrezno posodablja zgrajeni model. Vsako sliko lahko takoj po posodobitvi modela zavrže, zaradi česar je ta metoda še posebej primerna za aplikacije, ki zahtevajo sprotno obdelavo podatkov.

Osnovna metoda za analizo glavnih komponent ravno tako ne upošteva, da imajo lahko različni deli slik kot tudi celotne posamezne slike različen vpliv na proces učenja, čeprav je to v praksi zelo pogosto. Zato v disertaciji predlagamo posplošeno metodo za analizo glavnih komponent, ki omogoča različno utežitev posameznih slikovnih elementov in slik ter jih tako selektivno obravnava. Uteženo metodo nato še nadgradimo v metodo za učenje iz delnih podatkov, ki zgradi model predmeta tudi, če del vhodnih podatkov manjka.

Slike, ki jih zajamemo z različnimi vizualnimi senzorji, pogosto niso idealne in vsebujejo različne moteče dodatke, kot so odbleski in delna zakrivanja. PCA je v svoji standardni izvedbi zelo občutljiva na tak ne-gaussov šum. Bilo je predlaganih že več metod za robustno razpoznavanje, zelo malo pa je bilo narejenega na področju robustnega učenja. V disertaciji je predstavljen nov pristop k robustni gradnji na osnovi podprostorov. Predlagani paketna in inkrementalna metoda zaznata nekonsistentnosti v vhodnih slikah in zgradita predstavitve samo iz konsistentnih podatkov. Zgrajeni modeli so tako bolj robustni ter učinkoviti, kar omogoča bolj zanesljivo vizualno učenje in razpoznavanje tudi, ko učni pogoji niso idealni.

V disertaciji so izpeljane vse zgoraj omenjene metode in predstavljeni ustrezni algoritmi. Vsi predlagani algoritmi so tudi eksperimentalno ovrednoteni na različnih slikovnih domenah. Iz rezultatov je razvidna uporabnost metod za vizualno učenje in razpoznavanje v različnih primerih.

Ključne besede: vizualno učenje, robustno učenje, robustno razpoznavanje, uteženo učenje, manjkajoči podatki, inkrementalno učenje, podprostorske metode, analiza glavnih komponent, na izgledu osnovano modeliranje, glediščno-osrediščen pristop, računalniški vid.

Robust Subspace Approaches to Visual Learning and Recognition

Danijel Skočaj

Supervisor: Aleš Leonardis

Abstract

In the real world, visual learning is supposed to be a robust and continuous process. All available visual data is not equally important; in the case of occlusions or other undesirable intrusions in the field of view some visual data can even be misleading. Human visual system treats visual data selectively and builds efficient representations of observed objects and scenes even in non-ideal conditions. Furthermore, these representations can afterwards be updated with newly acquired information, thus adapting to the changing world. In this dissertation we study these premises and propose several methods, which introduce similar principles in the machine visual learning and recognition as well.

We approach visual learning by the appearance-based modeling of objects and scenes. Models are built using principal component analysis (PCA), which has several shortcomings with respect to the premises mentioned above. In order to overcome these shortcomings, we propose several extensions of the standard PCA.

PCA-based learning is traditionally performed in a batch mode, thus requiring all training images to be given in advance. Since this is not admissible in the framework of continuous learning, we propose an incremental method, which processes images sequentially one by one and updates the representation at each step accordingly. Each image can be discarded immediately after the model is updated, which makes the method perfectly well suited for real on-line scenarios.

In addition, in the standard PCA approach all pixels of an image receive equal treatment. Also, all training images have equal influence on the estimation of principal subspace. In this dissertation, we present a generalized PCA approach, which estimates principal axes and principal components considering weighted pixels and images. We further extend this weighted approach into a method for learning from incomplete data, which builds the model of an object even when the part of input data is missing.

Images of objects and scenes are not always ideal and as such they may contain various deceptive additions like reflections or occlusions. PCA in its standard form is intrinsically non-robust to such non-gaussian noise. Several methods for robust recognition have already been proposed, however robust learning has been tackled very rarely. In the dissertation we introduce a novel approach to the robust subspace learning. The proposed batch and incremental methods detect inconsistencies in the training images and build the representations from consistent data only. As a result, the obtained models are more robust and efficient enabling more reliable visual learning and recognition even when the learning conditions are not ideal.

In the dissertation we derive all the methods mentioned above and present suitable algorithms. We also experimentally evaluate all the proposed algorithms on different image domains and determine the applicability of the methods in different scenarios.

Key words: visual learning, robust learning, robust recognition, weighted learning, missing pixels, incremental learning, subspace methods, principal component analysis, appearance-based modeling, view-centered approach, computer vision.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Learning, representation, recognition

Vision is the most important and informative human sense. It provides us a lot of information about our surroundings; about the encircling environment, nearby objects and subjects, their interplay, and the events that are happening. However, the visual data, which is acquired by our retina, is meaningless until some significant descriptions are inferred; thus until these particular parts of our surroundings are recognized to a certain extent. Recognition is hence an essential part of the human perception.

Recognition itself implies learning. The prefix 're-' in the expression indicates a repetition, meaning that we can only re-cognize something that we have already seen. Therefore, we first have to *learn* how an object looks and store its *representation* in our memory to be able to *recognize* it in the future. Learning, representation and recognition are thus three inseparable parts of visual perception.

Visual recognition seems to be an easy task for humans. We can recognize most (previously seen) objects almost immediately without any particular effort. On the other hand, visual learning and recognition is one of the most difficult tasks in computer vision. Why is that so? How does the human brain learn and store visual information? How is recognition performed? Can these principles be utilized in computer vision as well?

These questions have been raised in the fields of psychology, psychophysics, and neuroscience as well as in computer vision. In the eighties, computer vision scientists mainly followed the Marr's paradigm [43], also known as *inverse optics paradigm.* It advocates that objects (or scenes) are represented as 3-dimensional models. The main requirement of this paradigm is thus the reconstruction of 3-D models from 2-D images. Such representation is also referred to as *object-centered* representation. It models the object only and is viewpoint and illumination invariant. Once such a 3-D model is stored in the memory, the object can be recognized from an arbitrary viewpoint under a wide range of illumination conditions.

This approach to visual learning and recognition roughly includes two steps which have proven to be difficult: first, building a consistently registered 3-D object-centered model, and second, extracting features on an image of an unknown object and matching these features to the features in the objects representation. This principle is depicted in Fig. 1.1.

It turns out that these two steps, which may be unnecessary for recognition, can be circumvented. This can be achieved by modeling objects as a set of views captured during a systematic observation of each object. Recognition can then be performed by *directly* matching an unknown 2-D view with the stored 2-D views. This approach is depicted as the shortcut in Fig. 1.1. It is known as *viewer-centered* approach, since the representation depends on the position of the viewer, i.e., it depends on the views, from which an object was observed during the learning. Once such a *view-based* model is stored in the memory, the object can be recognized only from similar views and under similar illumination conditions. When the model is based on a set of images of object's appearances from a number of views, this approach is sometimes also referred to as *image-based* or *appearance-based* approach.

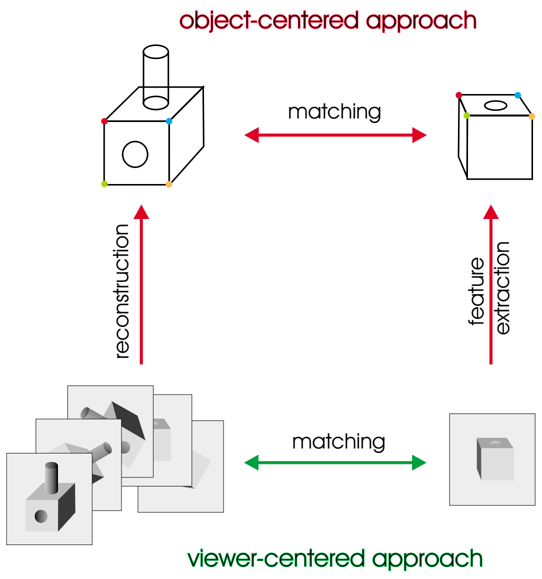


Figure 1.1: Object-centered and viewer-centered visual learning and recognition.

A big advantage of this approach is that it does not require any 3-D reconstruction. In this regard, it seems very unlikely that the human brain, which is the ultimate learning system, performs the complete 3-D reconstruction of the observed scene. Indeed, since the viewer-centered images are given as input to human visual system, it would not be surprising, if visual recognition was based on similar view-based mental representations [73]. In fact, in the nineties, several psychophysical, neurophysiological and behavioral studies provided a strong evidence that the human object recognition performance is strongly viewpoint dependent [73, 77, 15, 35, 72, 42]. The viewer-centered approach may seem very impractical, since each distinct view of an object necessitates a separate representation. However, due to the generalization capability, only a small number of viewer-centered representations, which provide enough information to enable generalization to the other unfamiliar views, are required.

Since 2-D images are given as input in computer vision systems as well, it would be convenient to apply the image-based approach to machine visual learning and recognition as well. Actually, a lot of research has been carried out in this direction. The fact that 3-D reconstruction is circumvented alleviates the process of learning significantly. What is essential for an effective image-based approach is an efficient representation, which is easy to build, does not require a huge amount of memory, enables generalization across views and is suitable for fast recognition (matching). Providing that an image-based method fulfills these requirements, it can be a good foundation for a powerful computer vision system for general object recognition.

1.2 Appearance-based approach

The appearance of an object is the combined effect of its shape, reflectance properties, pose in the scene, and illumination conditions [55]. It proves to be very difficult to differentiate all these factors from a set of images in order to obtain a view and illumination-invariant representation. In the appearance-based approach, the extraction of these physical properties is circumvented. However, in order to obtain a complete model of an object, one has to acquire all possible appearances of the object. The object has to be shown to an image sensor in several orientations and under various illuminations. Of course, there are infinitively many possible appearances of an object, if we consider all possible orientations and illumination conditions. However, only a number of images taken from distinct viewpoints under certain illumination conditions, coarsely encompassing all possible appearances, can model an object sufficiently well. Other images can then be obtained using generalization capabilities of the representation.

Nevertheless, the result of the systematic observation of the training object is a rather large set of images. Since all images represent the same object, they are usually highly correlated, therefore they contain a lot of redundant information. By reducing this redundancy, a set of images can be very efficiently compressed, without a considerable loss of significant information.

1.2.1 Principal component analysis

The most commonly used technique for compression of training images is based on *principal component analysis* (PCA) [34]. An image is considered to be a vector in high-dimensional space of all possible images. The basic idea of PCA is to efficiently map a high-dimensional input data to a low-dimensional subspace by reducing the redundancy and preserving as much information as possible. To achieve this goal, the directions with the largest variance of input data are found in the high-dimensional input space. The dimension of the space can be reduced by discarding the directions with small variance of the input data. By projecting the input data into this subspace, which has the principal directions for the basis vectors, we obtain an approximation with an error, which is minimal (in the least squares sense) among all linear transformations to a subspace of the same dimension.

Thus, learning is performed by estimating the *principal directions* considering all training images. Since these directions are usually obtained using the eigendecomposition, they are commonly referred to also as *eigenvectors*. An object is represented with the projections of the training images into the *principal subspace* (*eigenspace*) determined by the principal directions. It turns out that the correlation between two images can be approximated by the distance between their projections in the principal subspace. Thus, the recognition can be carried out by projecting an image of an unknown object into the principal subspace and finding the nearest projected training image [55].

Such representation fulfils all previously mentioned requirements for an efficient representation. First of all, the representation is easy to build. Well known statistical and algebraic methods for principal component analysis can be employed. It is very effective in terms of compression, since it requires an amount of memory as small as possible to represent input images to a certain degree of accuracy. Next, by interpolating between the projected points in the principal subspace, training images can be generalized to unfamiliar views as well. And lastly, the recognition can be performed very quickly, since it is reduced to the search for the closest point in a low-dimensional principal subspace.

Nevertheless, PCA is not the only subspace method that can be used to map high-dimensional images to a low-dimensional subspace. There exist several other techniques, each with its own properties and goal applications. In the next subsection we will briefly outline some of them.

1.2.2 Other subspace methods

PCA is an *unsupervised* method, which means that no additional information about the training images is necessary to build a representation. If, for instance, PCA is used for classification, no information on classes is used, thus the discriminant information might be lost. In this case, rather than maximizing the variance of all projections, one would prefer to maximize the distance between the projected class means, which increases the discriminant power of the transformation. This is the goal of *linear discriminant analysis* (LDA) [44]. Furthermore, next to maximizing the distance between the classes, *fisher's linear discriminant* [8] minimizes the distances within classes by minimizing within-class variance of the projections. It has been a popular tool in the field of pattern recognition, where it is frequently used to reduce the dimensionality of the input signal to alleviate the subsequent classification step.

For regression tasks *canonical correlation analysis* (CCA) [47, 46] is better suited. It relates two sets of observations by determining pairs of directions (canonical factors) that yield maximum correlation between the projections of these sets. Thus, it is suitable, for example, for estimation of orientation, where one set of observations consists of observed images, while the observations in the second set are object orientations from which the corresponding images were acquired.

Recently, another subspace technique became very popular — *independent component analysis* (ICA) [36, 6]. It is a powerful technique from signal processing known also as *blind source separation.* It finds a linear transformation so that the projections are as independent as possible. It can be seen as an extension of PCA, where the projections of the input data into the subspace are not only uncorrelated but also independent.

Another subspace technique is the *non-negative matrix factorization* (NMF) [39].

It finds factors with non-negative elements only. It tends to decompose the input images into parts (e.g., learn from a set of faces the parts a face consists of, i.e., eyes, nose, mouth, etc.).

All these methods are linear. They can be, however, extended to *nonlinear* feature extractors [20, 65, 48, 47]. This can be done by first mapping input vectors using a nonlinear mapping into a high-dimensional feature space and then performing a linear method on the obtained high-dimensional points. This procedure is equal to the employment of a non-linear method in the original space. To avoid computing a nonlinear mapping into a space of a very high (possibly infinite) dimension, the so called *kernel trick* can be applied [20]. It can be applied whenever it is possible to formulate the algorithm in such a way that it uses only dot products of the transformed input data. The dot products in feature space are then expressed in terms of kernel functions in input space, thus all operations can be performed in the original lower-dimensional input space.

All the methods mentioned above can be used for dimensionality reduction since they all find a transformation from a high-dimensional image space to a low-dimensional subspace. However, due to its superior reconstruction properties, PCA is by far the most popular and most frequently successfully applied method for the appearance-based learning and recognition. For that reason, in this dissertation we will focus on PCA and its extensions.

1.2.3 Applications and open problems

The appearance-based modeling of objects and scenes using subspace representations has become very popular in the vision community and has led to a variety of successful applications, e.g., object recognition [55], human face characterization [68] and recognition [76, 9], visual inspection [80], visual positioning and tracking of robot manipulators [54], illumination planning [51], mobile robot localization [37, 4], background modeling [57] and many more.

The main advantage of the appearance-based approach is that it does not require any knowledge about the shape and reflectance properties of objects. It uses raw image data directly without any extraction of geometric features. Since no significant preprocessing is required, the recognition is very efficient and can be usually accomplished in real-time.

However, this simplicity poses several limitations. The most notorious drawback of view-based approaches in general is a “combinatorial explosion” of the number of images required to model an object. Since every object is modeled with a number of images, which should roughly encompass all possible appearances, the number of required images can become prohibitive. The appearance of an object depends on a number of factors (i.e., view-points in 6 degrees of freedom, various illumination conditions, possible deformations). The number of required images grows exponentially with the number of factors that are considered.

However, in several practical applications the number of these factors can be reduced. If the recognition is performed in a controlled environment with restricted illumination conditions and constrained object orientations, there may be only one or two parameters that have to be considered, which can be reliably accomplished. The other approach to constrain the number of parameters that have to be modeled is to make subspace representations or recognition methods invariant to some factors (e.g., to illumination conditions [11]).

One solution is also to increase the capabilities for generalization across slight variations in appearance. However, although the representation should enable generalization across different appearances of an object, at the same time it has to discriminate between visually similar objects. Even more pronounced problem is classification of objects, thus class-level recognition. Since the recognition is based on appearance, which may vary significantly for different objects from the same perceptually-defined class, the appearance-based generalization from familiar class objects to unfamiliar ones is typically unfeasible. It seems that such problems can only be solved by using the combination of pure appearance-based approach with geometrically oriented methods, where the parts (regions, lines, corners, edges) of the object as well as structural relationships between them are detected and used for matching [73].

Since the appearance-based recognition is based on a simple template matching technique, the training and test images have to be aligned in the same manner. The basic appearance-based approach is rather sensitive to shift and scale. If the images can be segmented from the background, they can be re-scaled and pre-aligned to match the training images. Otherwise, a scale invariant recognition [10] can be used in conjunction with exhaustive search by moving a window of interest over the entire test scene.

The appearance-based method is also sensitive to occlusions and other non-gaussian noise. Several methods have been proposed to increase the robustness in the recognition stage, while the robust learning has been tackled very rarely. Since this is the main focus of the dissertation, we will discuss this and related problems more in detail in the next section.

1.3 Problem statement

In the real world, learning is usually a continuous, never-ending process. After we have acquired some knowledge about an object, we continuously update this knowledge every time when a new instance, or a new view of the object is encountered. The representation of the object in our memory becomes richer and the recognition more reliable. It is important to note that the representation of an object is not static; it changes through time adapting to the newly acquired information. A computer vision system for visual learning and recognition should be flexible as well. A learning method should be incremental, thus enabling the updating of the previously learned representations. An incremental approach would facilitate the representations to adapt to the newly encountered images. In this way, the new information would be incorporated into the existing representation and the recognition would become more reliable. In addition, in an incremental approach the training images are allowed to enter the learning process individually at different time instances. If the algorithm is not incremental, all the training images have to be given in advance, which is inadmissible in many realistic on-line scenarios.

The previous experience, prior knowledge, and the information obtained by other cognitive processes affect the level to which the newly acquired information is incorporated in the representation. One can also expect that more recent (or more reliable, or more informative, or more noticeable) experiences can have a stronger influence on the model than others. The psychophysical studies suggest that human perception is more tuned to some (e.g., more experienced) views than to others [60]. Therefore, a learning algorithm should enable a selective influence of individual training images to the process of learning. It should enable a selective treatment of individual pixels as well. In real world applications, it is often the case that not all data is available. The values of some pixels are missing or are totally unreliable. The *recognition* algorithm should be able to reliably recognize objects or scenes in spite of incomplete data. Furthermore, the *learning* algorithm should compensate the missing data as well and build the consistent representation, which would enable reliable recognition.

One has to be also aware that images of objects and scenes are not always ideal and as such they may contain noise, occlusions, reflections or other undesirable effects. A *recognition* method should be able to recognize an object under such conditions, providing that a consistent model of the object is available. However, if in the learning stage incorrect information is encoded into the representation, it may not be overcome in the recognition stage and the recognition may fail. The recognition can be performed reliably only if the representations are reliable. Therefore, an ultimate system for visual learning and recognition should be able to determine the objects of interest in the *learning* stage already, and include in the representation relevant information only.

Unfortunately, if we take a view-based approach to visual learning and recognition using PCA for building representations, the requirements mentioned above are not adequately met. The standard PCA approach is usually performed in a batch mode, i.e., all training images are processed simultaneously, which means that all of them have to be given in advance. The obtained representation is static and cannot be updated with new images. To make updating of the previously learned representation possible, one has to take an *incremental approach* to principal component analysis.

Next to that, in the standard PCA approach all pixels of an image receive an equal treatment. Also, all the training images have equal influence on the estimation of principal axes. To enable a selective influence of individual images and pixels, PCA can be generalized into a *weighted approach*, which considers individual pixels and images diversely, depending on the corresponding weights.

PCA in its standard form is also intrinsically non-robust to non-gaussian noise. The recognition method can be extended such that non-gaussian noise in test images is detected, and the recognition is performed by considering relevant parts of the image only, providing that a consistent representation is given. However, if the training images are taken under non-ideal conditions, the non-desirable effects should be detected in the learning stage already and not included into the representation. Thus, we need a method for *robust learning*, which is able to detect inconsistencies in the training images and build the representations from consistent data only.

In the following sections we will first review some existing extensions of the standard PCA approach, which cope with the problems mentioned above. Then we will outline our approach to the solution of these problems, which will be described in detail in this dissertation.

1.4 Related work

In this section we will review the most relevant work related to the problems, which will be discussed in the dissertation, namely incremental, weighted, and robust approaches to PCA. The most related algorithms will be further compared with our algorithms in the corresponding chapters.

1.4.1 Incremental PCA

Batch algorithms for PCA are usually based on eigenvalue decomposition or singular value decomposition of the covariance or the inner product matrix of the input data (these approaches will be described in detail in Chapter 2). Given many big images for input data, the covariance matrix as well as the inner product matrix become very large. Since the spatial and the time complexity of SVD of such large matrices are rather big, some problems cannot be solved using the batch method. This was the main motivation for some authors to propose incremental methods for PCA, which process images sequentially requiring singular value decomposition of low-dimensional matrices only [19]. Some other authors were motivated by the problems where not all training images were given in advance, thus requiring an algorithm for incrementally updating the principal subspace [32].

The first algorithm for incremental PCA in the computer vision community was proposed by Murakami and Kumar [50]. Then, Chandrasekeran et al. proposed an algorithm, which is based on SVD updating [19]. Very recently, Brand proposed a method for incremental singular value decomposition [14] as well. Also in the past several methods for SVD updating were proposed [16, 31]. All these methods keep the origin of the principal subspace in the origin of the image space, assuming that the mean of the input images is always zero. This is not true in general and this assumption may degrade the results of the classification [32].

By noting this problem, Hall et al. proposed a method for eigenspace updating, which sequentially shifts the origin of the eigenspace according to the new images, which are being added [32]. Furthermore, they extended their work by proposing a more general algorithm for merging and splitting eigenspaces of arbitrary dimensions [33] in contrast to all other methods, which are able to add only one new image at each update.

1.4.2 Weighted PCA

Basis vectors of the principal subspace, i.e., the principal directions in the input space, can also be estimated by minimizing the reconstruction error of all reconstructed input vectors. Also, one can tackle the principal component analysis using a probabilistic approach. PCA can be considered as a limiting case of the linear gaussian model, where the noise is infinitesimally small and equal in all directions [63]. From this observation Roweis derived an algorithm for calculating principal subspace [63], which is based on EM (expectation-maximization) algorithm [25]. Tipping and Bishop independently proposed a very similar Probabilistic PCA algorithm, based on an isotropic error model [75].

The EM algorithm for estimation of the principal subspace is composed from E-step and M-step, which are sequentially and iteratively executed. In E-step the new coefficients are computed using the estimated basis vectors, while in M-step the new basis vectors are estimated using the computed coefficients. This structure of EM algorithm allows us to introduce weights in order to perform weighted learning or learning from incomplete data.

Several methods for weighted learning with different derivations but very similar realizations have already been proposed. Wiberg [78] has proposed a method for subspace learning when data are missing based on the weighted least squares technique. This method was later extended by Shum et al. [66]. Gabriel and Zamir [30] proposed a method for subspace learning with any choice of weights, where each data point can have a different weight determined on the basis of reliability. A similar approach was also used in the work of Sidenbladh et al. [67] and De la Torre and Black [24].

Recently, Brand proposed a method for incremental singular value decomposition of data with missing values [14]. In the framework of eigenspace learning, a drawback of this algorithm is that it assumes that the mean is always zero. Liu and Chen proposed a method for incremental PCA with temporal weighting [41]. However, their method considers only a special case of temporal weights.

1.4.3 Robust PCA

A severe limitation of the basic approach to the subspace visual modeling is its nonrobustness to noise, occlusions, and cluttered background. If the background in the recognition stage differs from the background in the learning stage, or if an object in an image is occluded by some other object, or some other type of noise is present in the image, then the basic recognition method does not produce satisfying results. In theory, the breakdown point of the standard approach is 0%, which means that even a single erroneous data (pixel) can cause an arbitrary wrong result.

Different approaches have been proposed to improve the robustness of the recognition: modular eigenspaces [59], eigenwindows [56], search-window [52], adaptive masks [27], M-estimation [12, 22], and hierarchial approach [61]. The best results are produced by the method proposed by Leonardis and Bischof [40], which, instead of computing the coefficients by a projection of the data into the eigenspace, extracts them by a robust hypothesize-and-test paradigm using subsets of image pixels.

All methods mentioned above introduce the robustness in the *recognition stage*. They assume that the images in the learning stage were ideal and that the visual model is correct. Consequently, the outlying pixels in the recognition stage have a large reconstruction error. Therefore, considering the reconstruction error in a particular image pixel, the reliability of that pixel can be estimated. The *robust learning* is a much more difficult problem. Since in the learning stage the model of the object or the scene is being built, there is no previous knowledge, which could be used to estimate outliers.

Some authors have tackled also the problem of the robust visual learning. Xu and Yuille proposed an algorithm [79], which introduced robustness on the image level. During the learning stage, they discard images, which are inconsistent with the others. However, in practical applications this is not satisfactory. The robustness on the pixel level should be assured. Only single pixels should be discarded and not entire images. Gabriel and Zamir tried to solve this problem using a weighted singular value decomposition [30]. Recently, De la Torre and Black proposed a method for robust principal component analysis based on M-estimation [23, 24], which performs well on images with sufficient temporal correlation, but is very time consuming. Very recently, Aanæs et al. [3] proposed a method for robust factorization, which is tailored for different type of problems, however some of its principles are relevant for robust eigenspace learning as well.

1.5 Our approach

Our goal is to propose solutions to overcome the deficiencies of the standard PCA approach, which were stated in the Section 1.3. With this purpose, we have developed novel methods for incremental learning, weighted learning, learning from incomplete data and robust learning. In this section we will list these methods, and expose their similarities and dissimilarities with existing related work.

The *incremental algorithm*, which we will propose in this dissertation, estimates the identical principal subspace as the method proposed by Hall et al. [32]. However, the subspace is obtained in a different way. A significant advantage of our method is that it is able to treat different images differently, which enables to advance it into a weighted incremental method. Furthermore, our method maintains the low-dimensional representations of the previously learned images throughout the entire learning stage, meaning that each training image can be discarded immediately after the update.

Our basic batch algorithm for *weighted learning*, which we will derive by modifying EM algorithm [63], is closely related to some of the already proposed methods [30, 67], since in principle they all minimize the same error function – the weighted squared reconstruction error. Like in [24], we also present different batch algorithms, which are specialized for different types of weights (temporal, spatial). In addition, we adapt the EM algorithm for *learning from incomplete data* and further extend this algorithm with the regularization term, which adequately constrains the reconstructed values in missing pixels. Furthermore, we present a new approach to learning from incomplete data by iterative reconstruction of missing pixels. To enable continuous updating, we also extend our incremental method into a weighted version. Our *weighted incremental approach* considers temporal and spatial weights, thus it is more general than the method proposed in [41]. The incremental method is also adapted for learning from incomplete data, which in contrast to the method presented in [14], considers also the mean and updates its value at each step adequately. This method is in spirit related to a method for robust factorization [3]. What is common to both methods is that for each individual pixel a spatial weight is considered, which balances the influence of the value yielded by the current model and the influence of the pixel value of the input image.

The method for *robust learning*, which we will propose in this dissertation, is related to [23], however it is simpler and faster while still producing similar results. The proposed method iteratively detects outliers in all images, estimates the representation from inliers only and reconstructs outlying values. We will also propose a method for incremental robust learning, which sequentially determines consistencies in the input images and reconstructs inconsistent pixels using the previously acquired knowledge.

In the following chapters, we will theoretically derive all the methods mentioned above and present the algorithms suitable for implementation. We will experimentally evaluate all the proposed algorithms on different image domains and determine the applicability of the methods in different scenarios.

1.6 Organization of dissertation

In this chapter we gave the introduction to the dissertation and described the motivation for our work. Then we defined the problems, which are discussed in the dissertation, reviewed the related work and outlined our approach to the solutions of the problems.

In the next chapter we will discuss the basic PCA. We will derive PCA and present two algorithms for the calculation of the principal subspace. We will also present the main properties of PCA and show how they can be exploited for visual learning and recognition in the field of computer vision.

In Chapter 3 we will propose a method for incremental subspace learning. First we will present the algorithm and demonstrate its behavior on a simple 2-D example. Then the properties of the incremental learning will be derived by thorough experimental testing and theoretical explanations.

The weighted methods for subspace learning will be presented in Chapter 4. First we will present a batch algorithm for weighted PCA, where the weights can be set to arbitrary values. Then we will discuss the algorithm for learning from incomplete data, which can be considered as a special case of the algorithm for weighted learning. Both algorithms will then be incorporated in the incremental framework and evaluated in various experiments.

In Chapter 5 we will present robust algorithms for eigenspace learning, which are able to detect outliers in the training images and build the robust eigenspace representations. First, we will present a batch method for robust learning. Since the initial step of this method is still non-robust, a robust initialization of this algorithm based on a subsampling approach will then be discussed. Next, we will present the robust incremental algorithm for eigenspace learning. Finally, all presented algorithms will be evaluated in a number of experiments.

Chapters 3, 4, and 5 include a section where the experimental results, which serve for evaluation of the individual presented methods, are shown. In addition, in Chapter 6 we will present further experimental results, which will clarify some aspects of the proposed methods and will give a global view to all algorithms. Furthermore, we will compare the performance of the methods on various types of images and determine their applicability in different scenarios.

In Chapter 7 we will summarize our work, outline the contributions of the dissertation and give directions for future work.

In addition, we will give some useful supplements in Appendices. We will summarize the notation, which is used in the dissertation, overview some related topics from linear algebra and present more detailed comparison of several presented methods with related work. Finally, we will conclude the dissertation with the extended summary in Slovenian language.

Chapter 2

Basic PCA

2.1 Chapter overview

The appearance-based visual learning of objects and scenes is commonly realized using principal component analysis. PCA is a classic technique in statistical data analysis, feature extraction, and data compression. PCA was first formulated by Hotelling [34] in 1933 and since then it has been used in various applications in many areas.

In this chapter we will discuss the basic PCA to make the foundations for the extensions of this standard approach, which will be presented in the following chapters. Although it is primarily a statistical tool, we will approach PCA mainly from the algebraic perspective. First we will outline its properties considering a simple introductory example. Then we will derive PCA and present two algorithms for the calculation of the principal subspace. We will also present the main properties of PCA and show how they can be exploited in the field of computer vision for visual learning and recognition.

2.2 What is PCA?

Principal component analysis is a linear transformation from a high-dimensional input space to a low-dimensional feature space, which among all linear transformations guarantees the best possible representation of the high-dimensional input vectors in the low-dimensional feature space. It rotates the coordinate frame in a data-driven way, such that the variability of the input data can be efficiently described using only a small number of basis vectors.

This principle is illustrated in a simple 2-D example in Fig. 2.1. Consider eight 2-D points depicted as black dots. The goal of PCA is to find a new coordinate frame in which these eight points can be represented using only one basis vector as well as possible. This coordinate system is depicted with blue lines in Fig. 2.1 and its axes are referred to as *principal axes* (also *principal vectors* or *principal directions*). Due to the correlation between the elements of the input vectors, the first principal axis encompasses most of the variability of the input points. Thus, by projecting the input points onto the first principal axis we obtain 1-D *principal components* which are the best possible 1-D representations of the input points. The principal components (also referred to as *coefficients*) are depicted as cyan dots in Fig. 2.1.

The principal axes are obtained in such a way that they minimize the squared reconstruction error between the input points and their representations and they maximize the variance of the principal components. If we consider Fig. 2.1, the goal is to rotate the long blue line in a such direction, which yields the red lines as short as possible and the green lines as long as possible. It turns out that these two critera, namely the minimization of the reconstruction error and the maximization of the variance, are equivalent and can be uniquely satisfied using PCA.

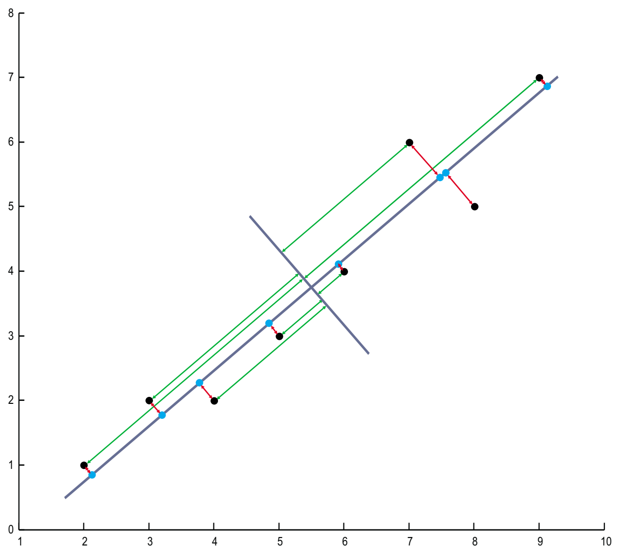


Figure 2.1: Principle of PCA.

PCA produces very good results, if the high-dimensional input vectors are correlated. This means that they contain redundant information. PCA removes the redundancy by decorrelating the input vectors; the new coordinates of the input vectors (principal components) are uncorrelated. As a consequence, the correlated high-dimensional input vectors can be efficiently represented as the uncorrelated low-dimensional vectors of principal components making PCA a very powerful tool for data compression.

2.3 Derivation and properties of PCA

After the initial informal outline of PCA, we will now derive and describe PCA more formally.

2.3.1 PCA by maximizing variance

First we will derive PCA by maximizing the variance in the direction of principal vectors.

Let us suppose that we have N M-dimensional vectors $\mathbf{x}_j$ aligned in the data matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$. Let $\mathbf{u}$ be a direction (a vector of length 1) in $\mathbb{R}^M$. The projection of the j-th vector $\mathbf{x}_j$ onto the vector $\mathbf{u}$ can be calculated in the following way:

$$a_j = \langle \mathbf{x}_j, \mathbf{u} \rangle = \mathbf{u}^\top \mathbf{x}_j = \sum_{i=1}^{M} u_i x_{ij} \quad . \tag{2.1}$$

We want to find a direction $\mathbf{u}$ that maximizes the variance of the projections of all input vectors $\mathbf{x}_j \ , \ j = 1 \ldots N$.

It follows that the mean of the projections is

$$\bar{a} = \frac{1}{N}\sum_{j=1}^{N} a_j = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M} u_i x_{ij} = \sum_{i=1}^{M} u_i \mu_i \tag{2.2}$$

and the variance is[1]

$$\begin{aligned}\sigma^2 = \frac{1}{N}\sum_{j=1}^{N}(a_j - \bar{a})^2 = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{M} u_i x_{ij} - \sum_{i=1}^{M} u_i \mu_i\right)^2 &= \\ = \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{M} u_i \hat{x}_{ij}\right)^2 = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M}\sum_{l=1}^{M} u_i \hat{x}_{ij} u_l \hat{x}_{lj} &= \\ = \sum_{i=1}^{M}\sum_{l=1}^{M} u_i u_l \frac{1}{N}\langle \hat{\mathbf{x}}_{i:}, \hat{\mathbf{x}}_{l:}\rangle = \sum_{i=1}^{M}\sum_{l=1}^{M} u_i c_{il} u_l &= \mathbf{u}^\top \mathbf{C}\mathbf{u} \ .\end{aligned} \tag{2.3}$$

[1]Subscript $\mathbf{x}_i$ denotes i-th *column* vector in the matrix $\mathbf{X}$, while $\mathbf{x}_{i:}$ denotes i-th *row* vector in the matrix $\mathbf{X}$. See Appendix A for details on the notation.

Here, μ_i is the mean of the i-th row in the data matrix $\mathbf{X}$ and $\hat{x}_{ij}$ is the value of x_{ij} with subtracted μ_i. If the vector $\boldsymbol{\mu}$ contains all row means, thus

$$\boldsymbol{\mu} = [\mu_1, \ldots, \mu_M]^\top = \frac{1}{N}\sum_{j=1}^{N} \mathbf{x}_j \ , \tag{2.4}$$

then[2]

$$\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N} \tag{2.5}$$

and $\mathbf{C}$ is the covariance matrix of $\mathbf{X}$, thus

$$\mathbf{C} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top \ . \tag{2.6}$$

Our goal is to maximize σ^2 under the constraint that $\|\mathbf{u}\| = 1$. Therefore, by using the technique of Lagrange multipliers, we have to maximize the function

$$F(\mathbf{u};\lambda) = \mathbf{u}^\top\mathbf{C}\mathbf{u} - \lambda(\mathbf{u}^\top\mathbf{u} - 1) = \sum_{i=1}^{M}\sum_{j=1}^{M} u_i c_{ij} u_j - \lambda\left(\sum_{i=1}^{M} u_i^2 - 1\right) . \tag{2.7}$$

A closed form solution of this maximization problem can be obtained in the following way:

$$\begin{aligned} \frac{\partial F}{\partial u_l} = \sum_{j=1}^{M} c_{lj}u_j + \sum_{i=1}^{M} u_i c_{il} - \lambda 2u_l &= 0 \ ; \ l = 1\ldots M \\ \sum_{i=1}^{M} c_{li}u_i &= \lambda u_l \ ; \ l = 1\ldots M \\ \mathbf{C}\mathbf{u} &= \lambda\mathbf{u} \ . \end{aligned} \tag{2.8}$$

Therefore, to find $\mathbf{u}$ and λ that maximize (2.7) we have to compute the eigenvectors and the eigenvalues of the covariance matrix $\mathbf{C}$. The largest eigenvalue equals the maximal variance, while the corresponding eigenvector determines the direction with the maximal variance.

By performing *eigenvalue decomposition* (EVD) or *singular value decomposition* (SVD) of the covariance matrix $\mathbf{C}$ (see Appendix B.2 for details) we can diagonolize $\mathbf{C}$

$$\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top \tag{2.9}$$

in such a way that the orthonormal matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N] \in \mathbb{R}^{M\times N}$ contains the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ in its columns and the diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{N\times N}$ contains

[2] $\mathbf{1}_{M\times N}$ denotes a matrix of the dimension $M \times N$, where every element equals 1.

the eigenvalues $\lambda_1, \ldots, \lambda_N$ on its diagonal. We will assume that the eigenvalues and the corresponding eigenvectors are arranged with respect to the descending order of the eigenvalues, thus $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$. Therefore, the most of the variability of the input random vectors is contained in the first eigenvectors. Hence, the eigenvectors are called *principal vectors* (also *principal axes* or *principal directions*).

This approach to calculation of principal vectors is very clear and widely used. However, if the size of the data vector M is very large, which is often the case in the field of computer vision, the covariance matrix $\mathbf{C} \in \mathbb{R}^{M \times M}$ (2.6) becomes very large and eigenvalue decomposition of $\mathbf{C}$ becomes unfeasible. If the number of input vectors is smaller than the size of these vectors ($N < M$), PCA can be sped up using the following method proposed by Murakami and Kumar [50].

Instead of the covariance (outer product) matrix $\mathbf{C} \in \mathbb{R}^{M \times M}$ the inner product matrix $\mathbf{C}' \in \mathbb{R}^{N \times N}$ (divided by the number of the input vectors) is calculated:

$$\mathbf{C}' = \frac{1}{N} \hat{\mathbf{X}}^\top \hat{\mathbf{X}} \ . \tag{2.10}$$

The eigenvalues and the eigenvectors of the covariance matrix $\mathbf{C}$ can then be determined from the eigenvalues λ'_i and eigenvectors $\mathbf{u}'_i$ of the matrix $\mathbf{C}'$ as:

$$\lambda_i = \lambda'_i \tag{2.11}$$

$$\mathbf{u}_i = \frac{\hat{\mathbf{X}}\mathbf{u}'_\mathbf{i}}{\sqrt{N\lambda'_i}} \ , \quad i = 1 \ldots N \ . \tag{2.12}$$

Note that $\mathbf{C}'$ is much smaller than $\mathbf{C}$ when $N \ll M$. Thus, the eigendecomposition of the $M \times M$ matrix $\mathbf{C}$ has been reduced to the much more feasible eigendecomposition of the $N \times N$ matrix $\mathbf{C}'$.

Considering the derivations above, the standard approach to PCA is outlined in Algorithm 1.

2.3.2 Properties of PCA

The orthonormal matrix $\mathbf{U}$ containing the principal vectors can serve as a linear transformation matrix for projection from the high-dimensional input space to the low-dimensional feature space and vice versa. The columns of $\mathbf{U}$ are the basis vectors of the new low-dimensional coordinate frame expressed with the high-dimensional coordinates. Thus an input vector can be projected into the principal subspace using the transformation matrix $\mathbf{U}^\top : \mathbb{R}^M \rightarrow \mathbb{R}^N$:

$$\mathbf{a} = \mathbf{U}^\top \hat{\mathbf{x}} \ . \tag{2.13}$$

Algorithm 1 : BPCA – batch PCA

```
Input: data matrix $\mathbf{X}$
Output: mean vector $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$.
 1: Estimate the mean vector: $\boldsymbol{\mu} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j$ .
 2: Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
 3: if $M \le N$ then
 4:     Estimate the covariance matrix: $\mathbf{C} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top$ .
 5:     Perform SVD on $\mathbf{C}$. Obtain the eigenvectors $\mathbf{U}$ and the eigenvalues $\boldsymbol{\lambda}$.
 6: else
 7:     Estimate the inner product matrix: $\mathbf{C}' = \frac{1}{N}\hat{\mathbf{X}}^\top\hat{\mathbf{X}}$ .
 8:     Perform SVD on $\mathbf{C}'$. Obtain the eigenvectors $\mathbf{U}'$ and the eigenvalues $\boldsymbol{\lambda}'$.
 9:     Determine the principal vectors $\mathbf{U}$: $\mathbf{u}_i = \frac{\hat{\mathbf{X}}\mathbf{u}'_i}{\sqrt{N\lambda'_i}}$ , $i = 1\dots N$ .
10:     Determine the eigenvalues $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$ .
11: end if
```

Thus, the coefficients a_j are computed as the projections of the input image onto each principal vector:

$$a_j = \langle \hat{\mathbf{x}}, \mathbf{u}_j \rangle = \sum_{i=1}^{M} u_{ij}\hat{x}_i \ , \ \ j = 1\dots N \ . \tag{2.14}$$

All the input vectors contained in the input matrix $\hat{\mathbf{X}}$ can thus be projected as $\mathbf{A} = \mathbf{U}^\top\hat{\mathbf{X}}$. Since $\mathbf{A}$ is an orthonormal transformation of the mean centered $\hat{\mathbf{X}}$, the principal components are also centered around zero:

$$\boldsymbol{\mu}_{\mathbf{A}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{a}_j = \frac{1}{N}\sum_{j=1}^{N}\mathbf{U}^\top\hat{\mathbf{x}} = \mathbf{U}^\top\frac{1}{N}\sum_{j=1}^{N}\hat{\mathbf{x}} = \mathbf{0} \ . \tag{2.15}$$

Now, let us calculate the correlation matrix of $\mathbf{A}$:

$$\begin{aligned}\mathbf{C}_{\mathbf{A}} = \frac{1}{N}\mathbf{A}\mathbf{A}^\top = \frac{1}{N}\mathbf{U}^\top\hat{\mathbf{X}}(\mathbf{U}^\top\hat{\mathbf{X}})^\top = \mathbf{U}^\top\frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top\mathbf{U} &= \\ = \mathbf{U}^\top\mathbf{C}\mathbf{U} = \mathbf{U}^\top\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top\mathbf{U} &= \boldsymbol{\Lambda} \ .\end{aligned} \tag{2.16}$$

Here, we replaced $\mathbf{C}$ with its diagonalized form (2.9) and considered the orthonormality of $\mathbf{U}$ (thus $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$). Therefore, the covariance matrix of the transformed data is the diagonal matrix $\boldsymbol{\Lambda}$, which contains the eigenvalues on its diagonal. This fact has two important implications. First, it proves that the transformed vectors are uncorrelated. Thus the redundancy caused by the correlation between the input vectors has been removed. Secondly, it shows that the variance in the direction of the i-th principal axis (the variance of the i-th principal components) is equal to the i-th eigenvalue λ_i, thus $\frac{1}{N}\sum_{j=1}^{N} a_{ij}^2 = \lambda_i$.

An important property of the diagonalization (2.9) is that it preserves the trace of the matrix which is being diagonalized [46]. Since the sum of the diagonal elements of the covariance matrix is the sum of variances of the input vectors, this implies that the total variance of the input data has been preserved and equals the sum of all eigenvalues:

$$\begin{aligned} \mathrm{VAR}(\mathbf{X}) = \sum_{i=1}^{M} \frac{1}{N}\hat{\mathbf{x}}_{i:}\hat{\mathbf{x}}_{i:}^{\top} = \sum_{i=1}^{M} c_{ii} &= \\ = \sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} \frac{1}{N}\mathbf{a}_{i:}\mathbf{a}_{i:}^{\top} &= \mathrm{VAR}(\mathbf{A}) \ . \end{aligned} \tag{2.17}$$

Now we will explain how can $\mathbf{U}$ serve as a transformation matrix for projection of the coefficient vector back into the input space. This operation is called *reconstruction*. The coefficient vector $\mathbf{a}$ is reconstructed using the transformation matrix $\mathbf{U} : \mathbb{R}^N \rightarrow \mathbb{R}^M$:

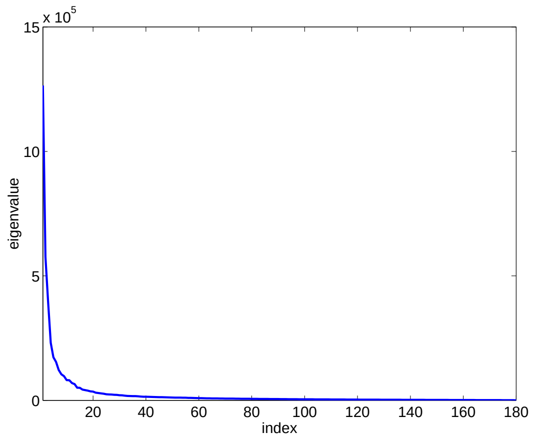
$$\hat{\mathbf{y}} = \mathbf{U}\mathbf{a} = \sum_{j=1}^{N} a_j \mathbf{u}_j \ . \tag{2.18}$$

Since N eigenvectors composing $\mathbf{U} \in \mathbb{R}^{M\times N}$ span the same subspace in $\mathbb{R}^M$ as all N input images composing $\mathbf{X} \in \mathbb{R}^{M\times N}$, each input image from $\mathbf{X}$ can be perfectly reconstructed without any reconstruction error. What is more interesting to us, is how well an input image is reconstructed from a subset of principal components only.

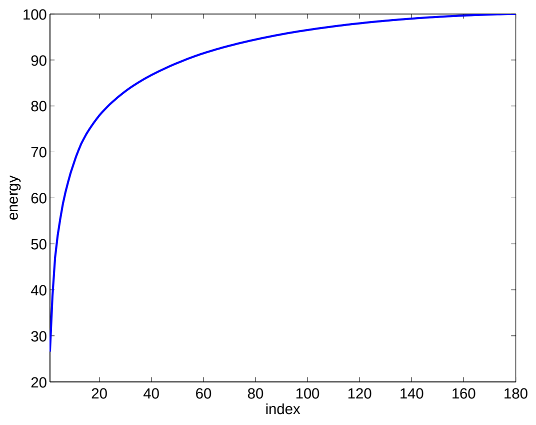
To realize this, we first consider how the variance is distributed across the principal axes. This distribution is called the *eigenspectrum* and it is practically a plot of eigenvalues sorted in the decreasing order. A typical eigenspectrum is depicted in Fig. 2.2(a). As one can observe, most of the variance is contained across the first few eigenvectors. This can also be measured with *energy*, which is defined as a fraction of the total variance. The energy contained in the first k eigenvectors can thus be calculated as

$$en_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \ . \tag{2.19}$$

The energy plot obtained from the eigenvalues depicted in Fig. 2.2(a) is shown in Fig. 2.2(b). Again, it is evident that most of the energy is contained in a first few eigenvectors already.



(a)



(b)

Figure 2.2: Typical (a) eigenspectrum, (b) energy.

From this we can conclude that we can obtain a good approximation of the input images by considering only a subset of eigenvectors associated with the largest eigenvalues. Therefore, from now on, we will consider only k, $k \ll N$, principal axes, thus $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_k] \in \mathbb{R}^{M \times k}$.

Now, an input vector is projected into the k-dimensional principal subspace using the transformation matrix $\mathbf{U}^\top : \mathbb{R}^M \to \mathbb{R}^k$:

$$\begin{aligned} \mathbf{a} &= \mathbf{U}^\top \hat{\mathbf{x}} = \mathbf{U}^\top (\mathbf{x} - \boldsymbol{\mu}) \\ a_j &= \langle \hat{\mathbf{x}}, \mathbf{u}_j \rangle = \sum_{i=1}^{M} u_{ij} \hat{x}_i = \sum_{i=1}^{M} u_{ij} (x_i - \mu_i) \ , \ \ j = 1 \dots k \end{aligned} \tag{2.20}$$

and reconstructed using the transformation matrix $\mathbf{U} : \mathbb{R}^k \to \mathbb{R}^M$:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{U}\mathbf{a} = \sum_{j=1}^{k} a_j \mathbf{u}_j \\ \mathbf{y} &= \hat{\mathbf{y}} + \boldsymbol{\mu} \ . \end{aligned} \tag{2.21}$$

Thus, an input image is approximated with a linear combination of the first k principal vectors.

The reconstruction error (residual vector) is equal to the difference between the input and the reconstructed vector:

$$\mathbf{e} = \hat{\mathbf{x}} - \hat{\mathbf{y}} = \sum_{j=1}^{N} a_j \mathbf{u}_j - \sum_{j=1}^{k} a_j \mathbf{u}_j = \sum_{j=k+1}^{N} a_j \mathbf{u}_j \ . \tag{2.22}$$

The most commonly used error measure is the squared reconstruction error, which is defined as a square of the length of the residuum. Considering the orthonormality of the eigenvectors $\mathbf{u}_j$ we obtain

$$e = \|\mathbf{e}\|^2 = \left\| \sum_{j=k+1}^{N} a_j \mathbf{u}_j \right\|^2 = \sum_{j=k+1}^{N} a_j^2 \ . \tag{2.23}$$

Thus, the squared reconstruction error is equal to the sum of squared discarded principal components. Since they are usually not known, the expected error can be approximated with expected values of the variance across the discarded eigenvectors, which are equal to the corresponding eigenvalues:

$$\mathcal{E}(e) = \sum_{j=k+1}^{N} \lambda_j \ . \tag{2.24}$$

The expected error is thus equal to the sum of the discarded eigenvalues. This consideration confirms the fact that by maximizing the variance in the first (non-discarded) eigenvectors, the squared reconstruction error is being simultaneously minimized. These two assertions are indeed two main properties of PCA.

Therefore, for a given dimension of a subspace k, PCA finds such principal vectors $\mathbf{u}_l,\ l = 1 \ldots k$ and coefficient vectors $\mathbf{a}_j \in \mathbb{R}^k, j = 1 \ldots N$ that minimize the total squared reconstruction error

$$e = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2 \ . \tag{2.25}$$

Thus, as an alternative to the maximization of the variance, the principal vectors and the principal components can be estimated by minimizing the squared reconstruction error (2.25). This is a nonlinear minimization problem and can be solved using one of the proposed algorithms, e.g., gradient descend algorithm [23] or neural networks [7, 26]. Alternatively, the minimization can be performed by iterating the two-step procedure where first the coefficients are estimated and then the principal vectors are computed. Such an algorithm [63], which was derived from the probabilistic point of view, will be described in the next subsection.

2.3.3 EM algorithm for PCA

PCA can be considered as a limiting case of a linear gaussian model, when the noise is infinitesimally small and equal in all directions. From this observation

Roweis [63] derived an algorithm for calculating principal axes, which is based on the EM (expectation-maximization) algorithm [25, 63, 75]. Here we will outline this probabilistic derivation of PCA and discuss the algorithm. A more comprehensive description can be found in [63].

A linear gaussian model assumes that an observed M-dimensional variable $\mathbf{x}$ was produced as a linear transformation of some k-dimensional latent variable $\mathbf{a}$ plus an additive gaussian noise. If we denote the transformation with $\mathbf{U} \in \mathbb{R}^{M\times k}$ and the noise with $\mathbf{v}$ (with the covariance matrix $\mathbf{R}$), the model can be written as

$$\mathbf{x} = \mathbf{U}\mathbf{a} + \mathbf{v} \qquad \mathbf{a} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \ . \tag{2.26}$$

Here, the latent variables $\mathbf{a}$ are assumed to be independent and identically distributed according to a unit variance spherical gaussian. Since the noise variables $\mathbf{v}$ are also independent and normally distributed, the model reduces to a single gaussian model for $\mathbf{x}$:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{U}\mathbf{U}^\top + \mathbf{R}) \ . \tag{2.27}$$

If the model parameters $\mathbf{U}$ and $\mathbf{R}$ are known, then for an observed variable $\mathbf{x}$ we can estimate the value of a hidden variable $\mathbf{a}$ using the posterior probability

$$\begin{aligned} P(\mathbf{a}|\mathbf{x}) &= \frac{P(\mathbf{x}|\mathbf{a})P(\mathbf{a})}{P(\mathbf{x})} = \frac{\mathcal{N}(\mathbf{U}\mathbf{a}, \mathbf{R})|_\mathbf{x} \mathcal{N}(\mathbf{0}, \mathbf{I})|_\mathbf{a}}{\mathcal{N}(\mathbf{0}, \mathbf{U}\mathbf{U}^\top + \mathbf{R})|_\mathbf{x}} = \\ &= \mathcal{N}(\boldsymbol{\beta}\mathbf{x}, \mathbf{I} - \boldsymbol{\beta}\mathbf{U})|_\mathbf{a} \ , \ \ \boldsymbol{\beta} = \mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top + \mathbf{R})^{-1} \ . \end{aligned} \tag{2.28}$$

If the model parameters are not known, they can be estimated by identifying the matrices $\mathbf{U}$ and $\mathbf{R}$ that make the model assign the highest likelihood to the observed data. This is usually achieved using the well known EM algorithm or one of its derivatives, which is tailored for the specific covariance structure of the noise (matrix $\mathbf{R}$).

In the case of PCA, the covariance of the noise becomes infinitesimally small and equal in all directions, thus

$$\mathbf{R} = \lim_{\epsilon \to 0} \epsilon \mathbf{I} \ . \tag{2.29}$$

Therefore, $\boldsymbol{\beta}$ from (2.28) becomes

$$\boldsymbol{\beta} = \lim_{\epsilon \to 0} \mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top + \epsilon\mathbf{I})^{-1} \tag{2.30}$$

and considering that $\mathbf{U}^\top(\mathbf{U}\mathbf{U}^\top)^{-1} = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top$ (2.28) reduces to

$$P(\mathbf{a}|\mathbf{x}) = \mathcal{N}((\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{x}, \mathbf{0})|_\mathbf{a} = \delta(\mathbf{a} - (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{x}) \ . \tag{2.31}$$

Since the noise is infinitesimally small, the covariance of the noise becomes zero and the posterior probability over hidden states collapses to a single point $\mathbf{a} = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{x}$.

After the covariance of the noise has been restricted a suitable EM algorithm can be applied. This algorithm consists of two steps, E and M, which are sequentially and iteratively executed:

- **E-step**: Estimate the unknown states $\mathbf{a}$ using the current model parameters $\mathbf{U}$.
- **M-step**: Compute the new model parameters $\mathbf{U}$, which maximize the expected joint likelihood of the estimated latent variables $\mathbf{a}$ and the observed variables $\mathbf{x}$.

In the context of PCA, the model parameters $\mathbf{U} \in \mathrm{I\!R}^{M\times k}$ are related to the first k principal vectors, the latent variables (or hidden states) $\mathbf{a}$ are associated with the vectors of the principal components $\mathbf{a}_j \in \mathrm{I\!R}^k, j = 1 \dots N$ and the observed variables $\mathbf{x}$ correspond to the mean-centered input vectors $\hat{\mathbf{x}}_j \in \mathrm{I\!R}^M, j = 1 \dots N$. Considering the derivation above, the two steps of the algorithm for PCA proposed by Roweis [63] look as follows:

- **E-step**: $\mathbf{A} = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\hat{\mathbf{X}}$
- **M-step**: $\mathbf{U} = \hat{\mathbf{X}}\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}$.

It has been proven that EM algorithms always converge to a local maximum of the likelihood [25] and that the only stable local extremum is the global maximum at which the true principal subspace is found [75, 74]. Therefore, the algorithm converges to the correct result. The convergence can be checked by looking at a difference in the successive estimates of the average lost variance per discarded dimension (σ^2). As shown in [75] (and discussed in the previous section), the maximum likelihood solution for σ^2 is

$$\sigma^2 = \frac{1}{N-k}\sum_{j=k+1}^{N} \lambda_j \ , \tag{2.32}$$

which corresponds to the average discarded eigenvalue. It can be conveniently calculated by

$$\sigma^2 = \frac{1}{N-k}\left(\mathrm{VAR}(\hat{\mathbf{X}}) - \sum_{j=1}^{k} \lambda_j\right) \ , \tag{2.33}$$

where $\mathrm{VAR}(\hat{\mathbf{X}})$ is the variance of $\hat{\mathbf{X}}$, which is defined as the sum of variances in rows of $\hat{\mathbf{X}}$. Since in the EM algorithm we do not explicitly calculate eigenvalues at each iteration, we can estimate σ^2 by

$$\sigma^2 = \frac{1}{N-k}\left(\mathrm{VAR}(\hat{\mathbf{X}}) - \mathrm{VAR}(\mathbf{A})\right) \ . \tag{2.34}$$

The initial solution for $\mathbf{U}$ can be determined using some other method or simply by setting the elements of $\mathbf{U}$ to random values.

At convergence, the columns of $\mathbf{U}$ *span the space* of the first k principal axes. However, they are not oriented in the directions of the principal axes nor they are orthogonal. To obtain the real principal axes, we first have to orthogonalize $\mathbf{U}$. Then, the input vectors are projected into $\mathbf{U}$ and PCA is performed on the obtained coefficients. Since the coefficient vectors $\mathbf{a}_j \in \mathbb{R}^k$ are low-dimensional, a standard SVD-based approach is used. The obtained eigenvectors are the true principal vectors expressed with the basis $\mathbf{U}$. Thus to obtain the principal vectors expressed in the input space coordinates, the vectors in $\mathbf{U}$ are rotated to match the obtained principal vectors in the subspace coordinates. The variance of principal components is not affected with the basis vectors, which are used for expressing principal axes, thus the obtained eigenvalues are already the true eigenvalues that we are looking for.

The EM algorithm for PCA is summarized in Algorithm 2. Neglecting the probabilistic interpretation, this algorithm can be regarded as an iterative procedure for solving a nonlinear optimization problem of minimizing the squared reconstruction error (2.25). The main advantage of this algorithm is that it does not involve the calculation of the covariance matrix of the input data. This can be very advantageous when for a large number of high-dimensional vectors a low-dimensional principal subspace has to be found. Furthermore, this algorithm can be adapted for weighted minimization and for dealing with missing data, as we will describe in the next chapter.

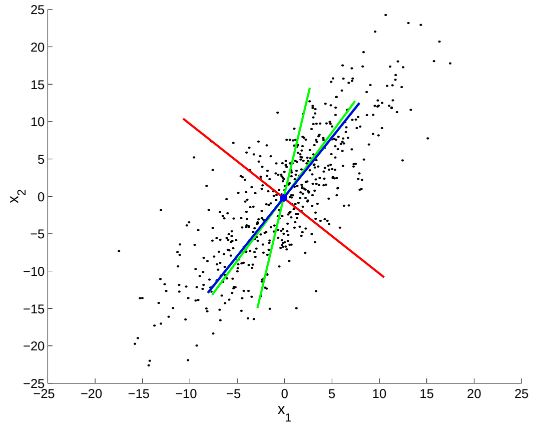
Finally, we present an example, which illustrates the behavior of the EM algorithm. Fig. 2.3(a) depicts a cloud of 2-D points drawn from a gaussian distribution. EM algorithm was used to estimate the first principal axis. The progress of the algorithm is indicated with the solid lines whose directions indicate the guess of the principal vector at each iteration, starting with the red line and ending with the blue line, which is indeed the correct first principal axis. Fig. 2.3(b) plots the value of σ^2, which is used as a stopping criterion. As one can observe, the convergence is reached very fast. A very similar convergence behavior was observed also in experiments presented in [63], as well as in our other experiments.

Algorithm 2 : EMPCA – EM algorithm for PCA

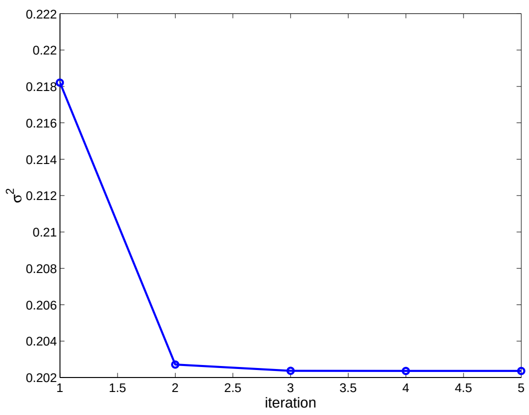
Input: data matrix $\mathbf{X}$, number of principal axes to be estimated k.

Output: mean vector $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$.

```
 1: Estimate the mean vector: $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$ .
 2: Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
 3: Set elements of $\mathbf{U} \in \mathbb{R}^{M\times k}$ to random values.
 4: repeat
 5:    E-step: $\mathbf{A} = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\hat{\mathbf{X}}$ .
 6:    M-step: $\mathbf{U} = \hat{\mathbf{X}}\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}$ .
 7: until the change of $\sigma^2 = \frac{1}{N-k}\left(\mathrm{VAR}(\hat{\mathbf{X}}) - \mathrm{VAR}(\mathbf{A})\right) \leq$ threshold
 8: Orthogonalize $\mathbf{U}$.
 9: Project input data on $\mathbf{U}$: $\mathbf{A} = \mathbf{U}^\top\hat{\mathbf{X}}$.
10: Perform PCA on $\mathbf{A}$. Obtain $\mathbf{U}'$ and $\boldsymbol{\lambda}'$.
11: Rotate $\mathbf{U}$ for $\mathbf{U}'$: $\mathbf{U} = \mathbf{U}\mathbf{U}'$.
12: Determine the eigenvalues $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$ .
```



(a)



(b)

Figure 2.3: Illustration of EM algorithm on a simple 2-D example: (a) progress and (b) convergence of the algorithm.

2.4 PCA for visual learning and recognition

The main idea of the appearance-based visual learning and recognition is that a 3-D object is represented as a set of 2-D images coarsely sampling all possible appearances of the object. Then an image of an object to be recognized is compared to all of the images of all training objects and the most similar training image is searched for. The unknown object is recognized to be the object depicted in the found image. Since a number of training images is very large and images contain a large number of pixels PCA seems to be a perfect tool for reduction of data dimensionality. In this section we will show, how PCA can be utilized for visual learning and recognition.

2.4.1 Theoretical issues

Let us assume that in the learning stage N training images of the same size were acquired. To simplify the explanation and without losing generality we assume that we are dealing with gray level intensity images. An image is thus represented as a matrix of gray values. All the image pixels are then aligned into one vector of the length M (by, for example, concatenating image rows). Thus, each training image is represented as a vector $\mathbf{x}_j = [x_{1j}, \ldots, x_{Mj}]^\top \in \mathbb{R}^M,\ j = 1 \ldots N$. Since images are typically rather big, $\mathbb{R}^M$ is usually a very high-dimensional input space, also called *image space*. All the input images (vectors) form the *data matrix* $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$.

The data matrix $\mathbf{X}$ is now in every respect regarded as it was in the previous sections. We perform PCA on $\mathbf{X}$ using one of the previously described algorithms and obtain the mean vector $\boldsymbol{\mu}$, the principal axes (eigenvectors) $\mathbf{U}$ and the eigenvalues $\boldsymbol{\lambda}$. Since the input vectors represent images, the mean vector and the eigenvectors can also be visualized as images and are usually referred to as the *mean image* and the *eigenimages*. As the variance is mainly contained in the first eigenimages, only k, $k \ll N$ eigenimages are retained, thus $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{M \times k}$. All the training images are then projected into the eigenspace and the coefficient vectors $\mathbf{a}_j = [a_{1j}, \ldots, a_{kj}]^\top \in \mathbb{R}^k\ \ j = 1 \ldots N$ are obtained. Therefore, each image $\mathbf{x}_j$ is approximated with the linear combination of the eigenimages

$$\mathbf{x}_j \approx \sum_{i=1}^{k} a_{ij}\mathbf{u}_i + \boldsymbol{\mu} \ . \tag{2.35}$$

Now, let us consider two images $\mathbf{x}_m$ and $\mathbf{x}_n$ and their projections in the eigenspace $\mathbf{a}_m$ and $\mathbf{a}_n$. The similarity between two images is usually determined using the correlation or by finding the *sum of squared distances* (SSD) between the gray values of the corresponding pixels in the images. By approximating the images with their projections, the SSD between the images can be expressed as

$$\begin{aligned}\|\mathbf{x}_m - \mathbf{x}_n\|^2 \approx \left\|\sum_{i=1}^{k} a_{im}\mathbf{u}_i - \sum_{i=1}^{k} a_{in}\mathbf{u}_i\right\|^2 &= \\ = \left\|\sum_{i=1}^{k}(a_{im} - a_{in})\mathbf{u}_i\right\|^2 &= \|\mathbf{a}_m - \mathbf{a}_n\|^2 \ , \end{aligned} \tag{2.36}$$

where the last simplification results from the orthonormality of the eigenvectors. This derivation has very important consequences [55]. It shows that the similarity between images can be measured by the distance of their projections in the eigenspace. SSD in the image space can be approximated with SSD in the eigenspace.

If the input images are normalized ($\|\mathbf{x}_j\| = 1$), then SSD is related to the *correlation* $\mathbf{x}_m^\top \mathbf{x}_n$ between images as

$$\|\mathbf{x}_m - \mathbf{x}_n\|^2 = (\mathbf{x}_m - \mathbf{x}_n)^\top (\mathbf{x}_m - \mathbf{x}_n) = 2 - 2\mathbf{x}_m^\top \mathbf{x}_n \ . \tag{2.37}$$

Therefore, minimizing SSD corresponds to maximizing the correlation between images. If the images are similar, their correlation is large and SSD in the image space as well as in the eigenspace is small.

In the recognition stage, an image of an unknown object is thus projected into the eigenspace and the closest projected training image in this subspace is searched for. Since the smallest distance between the projections in the eigenspace corresponds to the minimal SSD, the test image is the most similar to the training image, which corresponds to the closest projection. The problem of finding the most similar image in the image space is thus transformed to the search for the closest point in the low-dimensional eigenspace.

Since the appearance of an object changes slowly and smoothly by varying viewpoint and/or illumination direction, the eigenspace projections of the images form a continuous and smooth manifold. This manifold is defined in discrete points (coefficient vectors) only, however, by defining new points using spline interpolation between the existing points, we can increase their density. In this way we approximate coefficient vectors, which would be obtained by projecting additional training images into the eigenspace. By varying the values of the eigenspace coefficients we can simulate the variation of the parameters, which are being used for acquiring training images.

This implicates that the eigenspace representation is suitable for regression tasks as well. For example, in the learning stage we acquire a relatively small number of images from sparsely sampled views. Then, the projections of these images are interpolated in the eigenspace. After a new image of an unknown object is projected into the eigenspace and the closest point is found, not only that the correct object can be recognized, the orientation of the object can be determined as well, since the orientations related to the training images (and projections) are known. And due to the interpolated points, the resolution of the orientation, which can be estimated, is much higher than the sampling resolution of the views, from which the training images were acquired.

2.4.2 Usual procedure

A usual procedure for visual learning and recognition using PCA can be summarized as follows [55]. In the learning stage, a set of training images of all objects is obtained by varying the pose and the illumination conditions. Then an object is segmented from the background on each training image and resized to the common image size. In this way, the scale invariance is achieved. If we want to achieve the invariance on the illumination intensity as well, each image should be normalized (each pixel value should be divided by the norm of the image vector). Then, the eigenspace is constructed from all training images and their projections in the eigenspace are obtained. The projections of the images of each object lie on a manifold that is parameterized by the pose and the illumination. The manifold can be represented even more densely by interpolating the projected points. Thus, each object is represented with a manifold in the eigenspace.

In the recognition stage, a novel image of an object, which is to be recognized, is first segmented and normalized in exactly the same way as the training images were processed during the learning stage. Then it is projected into the eigenspace. The closest manifold reveals the identity of the object and the exact position of the closest point on the manifold determines the pose and the illumination direction.

2.4.3 Example

We will demonstrate the principle of appearance-based learning and recognition on a simple example. Let us assume that we want to model ten objects from the COIL20 database [53] shown in Fig. 2.4(a). Each object is modeled using 36 images, which were captured from the viewpoints evenly distributed around the object (Fig. 2.4(b)). In this simple example only one degree of freedom for varying object's pose and uniform illumination conditions were considered.

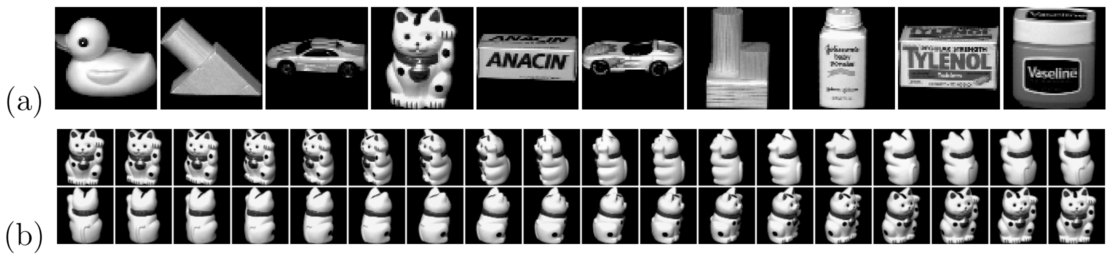


Figure 2.4: Training set: (a) one image of each object and (b) all images of one object.

Fig. 2.5(a) depicts the mean image and the first five eigenimages, which were obtained by applying PCA on the training images. Each training image is then approximated using a linear combination of eigenimages. Fig. 2.5(b) shows the approximations of one image using varying number of principal components.

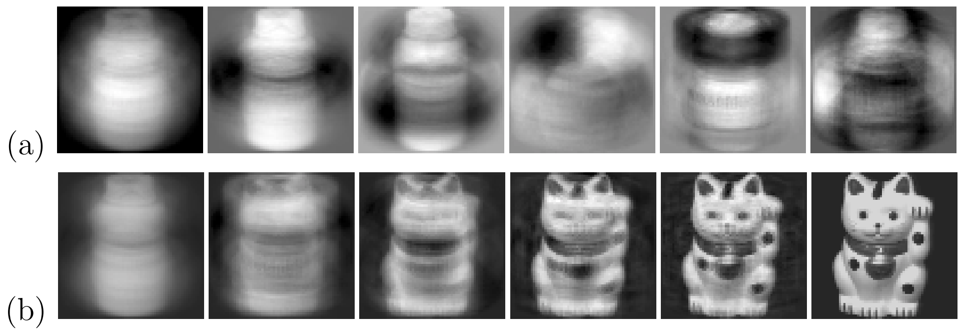


Figure 2.5: (a) Mean image and first five eigenimages. (b) Reconstructed images using 1, 8, 15, 40, 100, 360 principal components.

As we can see a relatively small number of principal components is sufficient for a relatively good reconstruction of an image. In this case 15 principal components are enough to visually distinguish the object as well as for automatic recognition using eigenspace approach. The reason for this is that most of the visual variability of the images is captured using the first eigenvectors, as can be seen from the eigenspectrum and the plot of the energy shown in Fig. 2.6. These plots are usually used for determining what number of eigenvectors to use. This number can be found as the index of the eigenvalue, where the eigenspectrum levels off and becomes near constant (*scree test*) [18]. Alternatively, this number can be found as the index of the eigenvalue, where the energy exceeds some predefined threshold.

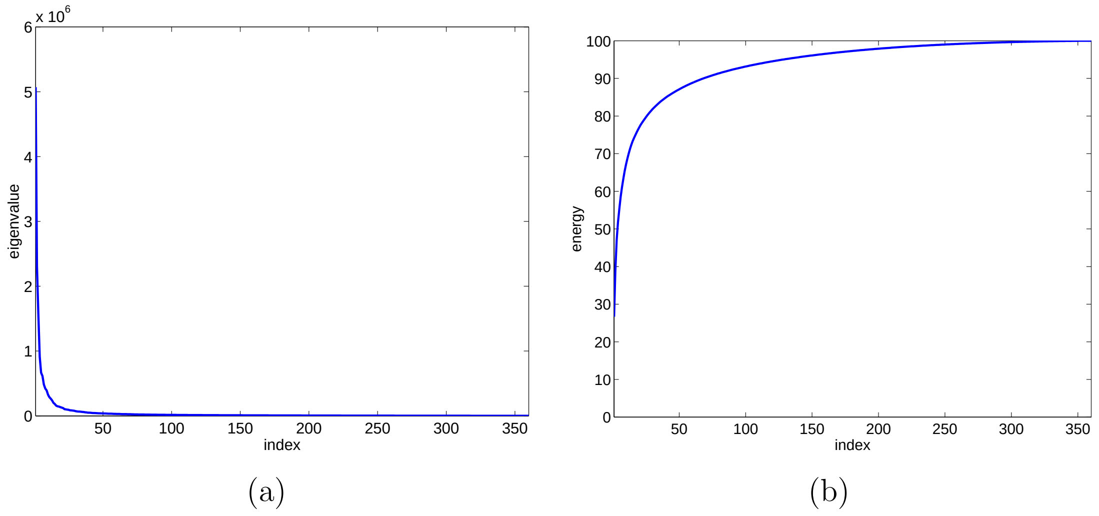


Figure 2.6: (a) Eigenspectrum. (b) Energy.

Fig. 2.7(a) depicts the projections of the training images of three objects in the three-dimensional eigenspace. As one can observe, the points representing different objects (each object is represented with the points of the same color) are nicely separated in three dimensions already. This is even more noticeable in Fig. 2.7(b) which depicts the manifolds for the three objects, which were obtained by interpolating the projected points.

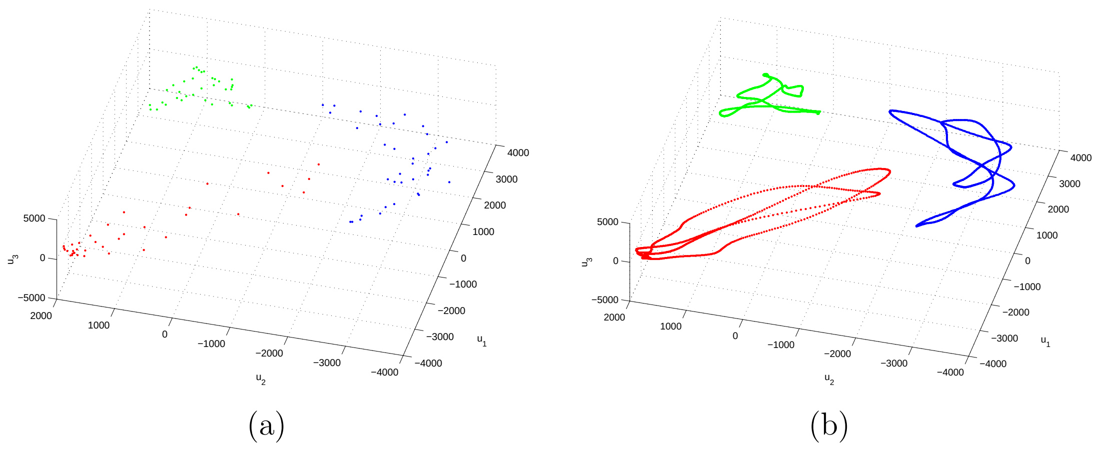


Figure 2.7: (a) Projections of training images of three objects shown using three most significant principal axes. (b) Interpolated points; manifolds.

In the recognition stage, a novel image is projected into the eigenspace and its identity and orientation are identified by determining the closest point in the eigenspace.

2.4.4 Discussion

Until now we were mainly discussing the recognition of objects. However, the same principles can be used for the recognition of scenes and localization in an environment as well. The environment is modeled using a number of images, which are usually taken with an omnidirectional sensor. An example of the panoramic image is shown in Fig. 2.8. Then the model is built using PCA. Later, scene recognition and the localization of the sensor are performed in the same manner as object recognition.

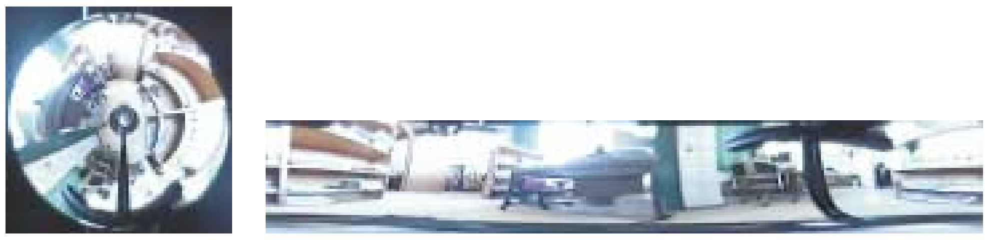


Figure 2.8: Panoramic image: (a) original, (b) unwarped.

Furthermore, not only the intensity images, but also the images of various modalities can be utilized using this method, ranging from color images [22], to range images [69, 17, 2], infrared images [29], binary images [45], gradient images [11, 1], and many others (some of them are depicted in Fig. 2.9). Since an image is treated as a high-dimensional vector it can contain any values measured using some optical (or even non-optical) device or obtained by image processing. The only requirement is that the images are (possibly highly) correlated.

In this section only the basic PCA approach was discussed. It suffers from several deficiencies, which were not discussed (sensitivity to shift, scale, rotation, cluttered background, occlusions, noise, illumination changes). A lot of methods were proposed to overcome some of these disadvantages. This is also the focus of the dissertation; to increase the robustness of the PCA-based visual learning and recognition.

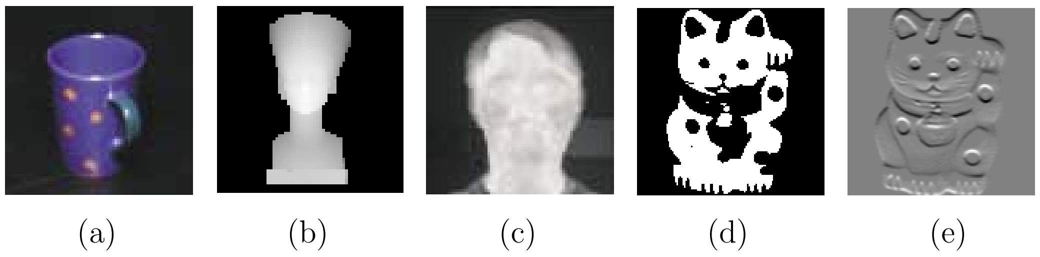


Figure 2.9: Different modalities: (a) color image, (b) range image, (c) infrared image, (d) binary image, (e) gradient image.

2.5 Chapter summary

In this chapter we introduced the basic PCA. We presented this method from different viewpoints and gave two algorithms for estimation of principal subspaces. We presented the main properties of PCA and showed how to utilize them for the efficient appearance-based modeling and recognition of objects and scenes.

The main purpose of this chapter was to give to the reader the adequate knowledge about fundamental principles of PCA. In the following chapters these basic principles will be extended with the aim to increase applicability of PCA for the appearance-based visual learning and recognition in the real world conditions.

Chapter 3

Incremental PCA

3.1 Chapter overview

As described in the previous chapter, most methods for subspace learning operate in a batch mode, i.e., all training images are processed simultaneously in one step. Incremental methods, on the other hand, take the training images sequentially and update the current principal subspace step by step.

There are two deficiencies of batch methods that can be overcome using incremental approach. The first deficiency is related to the computational and storage issues. Images are usually not very small and if we operate with a large number of images, the spatial and the time complexity of the estimation of the principal subspace becomes unfeasible.

The second deficiency of the batch methods is that they require all of the training images to be given in advance. This is inadmissible in an on-line scenario, where the images to be processed are obtained sequentially. In such scenarios it is also often required that the images are not kept in the memory and are discarded after being processed.

In this chapter we propose a method for incremental learning that overcome these two problems. It takes the training images sequentially and computes the new eigenspace from the subspace obtained in the previous step and the current input image.

First, we will present the algorithm and demonstrate its behavior on a simple 2-D example. Since the proposed method is highly related to the method proposed by Hall et al. [32], we will make a comparison between both methods. Then the properties of the incremental learning will be derived by thorough experimental testing and theoretical explanations.

3.2 Algorithm

Let us suppose that we have already built an eigenspace from the first n images. In the step $n+1$ we could calculate a new eigenspace from the *reconstructions* of the first n input images and a new image using the standard batch method. The computational complexity of such an algorithm would be prohibitive, since at each step we would have to perform the batch PCA on a set of high-dimensional data. However, identical results can be obtained by using low-dimensional *coefficient vectors* of the first n input images instead of their high-dimensional reconstructions, since coefficient vectors and reconstructed images encompass the same visual variability, i.e., they are the same points represented in different coordinate frames. Since the dimension of the eigenspace is small, this algorithm is computationally very efficient.

In practice, this algorithm is realized in the following way (see Fig. 3.1 for illustration). First, a novel image $\mathbf{x}$ is projected into the current eigenspace $\mathbf{U}^{(n)}$ and the obtained coefficient vector $\mathbf{a}$ is reconstructed $(\mathbf{y})$[1]. The difference between the input image and its reconstruction is orthogonal to the current eigenspace, therefore if the current eigenspace is enlarged with the residual vector $\mathbf{r}$, a new basis $\mathbf{U}'$ is obtained. In this basis all the points from the current eigenspace and the new image can be represented without any loss $(\mathbf{A}')$. Now, the batch PCA is performed on these points in the low-dimensional space and the principal axes $\mathbf{U}''$ are obtained. All the low-dimensional points are then projected into the new eigenspace $(\mathbf{A}^{(n+1)})$. Finally, the current basis vectors $\mathbf{U}'$, expressed in the image space coordinates, are rotated to match the new basis vectors expressed in the subspace coordinates $(\mathbf{U}^{(n+1)})$, and the mean vector is updated $(\boldsymbol{\mu}^{(n+1)})$. More formally, the necessary steps for updating an eigenspace are given in Algorithm 3.

This algorithm increases the dimension of the subspace for one. After the update, we can discard the least significant principal vector to preserve the dimension of the subspace [4, 33].

The initial values of the mean image, the eigenvectors, and the coefficients can be obtained by applying the batch PCA on a small set of images. Alternatively, one can simply set the first training image as the initial eigenspace by assigning

[1]Superscript denotes the step which the data is related to. $\mathbf{U}^{(n)}$ denotes the values of $\mathbf{U}$ at the step n.

$\boldsymbol{\mu}^{(1)} = \mathbf{x}_1$, $\mathbf{U}^{(1)} = \mathbf{0}_{M\times 1}$, and $\mathbf{A}^{(1)} = 0$. In this way, the algorithm is completely incremental, requiring only one image to be available at each time instant.

Algorithm 3 : IPCA – incremental PCA

```
Input: current mean vector $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients
    $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$.
Output: new mean vector $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$,
    new eigenvalues $\boldsymbol{\lambda}^{(n+1)}$.
 1: Project a new image $\mathbf{x}$ into the current eigenspace: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$ .
 2: Reconstruct the new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 3: Compute the residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
 4: Append $\mathbf{r}$ as a new basis vector: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{bmatrix}$ .
 5: Determine the coefficients in the new basis: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ .
 6: Perform PCA on $\mathbf{A}'$. Obtain the mean value $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$, and the
    eigenvalues $\boldsymbol{\lambda}''$.
 7: Project the coefficient vectors to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$ .
 8: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$ .
 9: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ .
10: New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ .
```

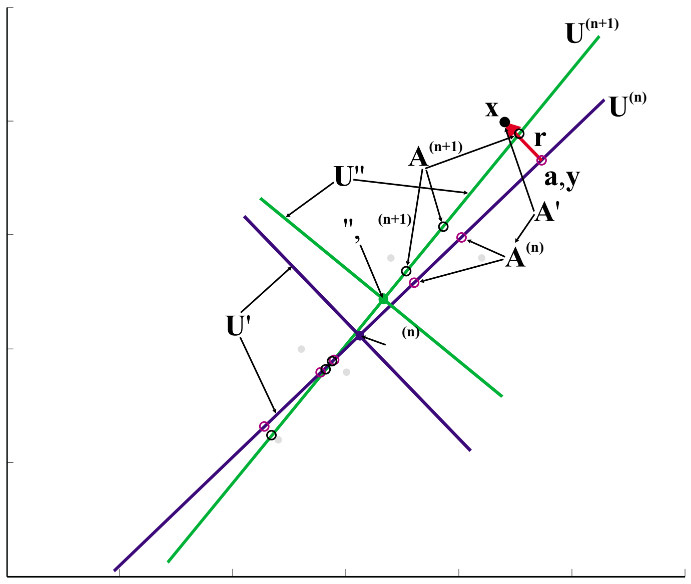


Figure 3.1: Illustration of one update.

This algorithm is perfectly well suited for continuous on-line scenarios, where the training images enter into the system sequentially and eigenspace is updated at each step accordingly. Since it maintains the low-dimensional representations of the previously learned images throughout the entire learning stage, each training image can be discarded immediately after the update.

3.2.1 Simple 2-D example

We will demonstrate the behavior of the proposed algorithm on a simple 2-D example. The 2-D input space contains 41 points shown as black dots in Fig. 3.2. The goal is to estimate 1-D principal subspace, i.e., the first principal axis. The eigenspace is being built incrementally. At each step one point (from the left to the right) is added to the representation and the eigenspace is updated accordingly. Fig. 3.2 illustrates how the eigenspace evolves during this process. The principal axis, obtained at every sixth step, is depicted. The points, which were appended to the model at these steps, are marked with crosses. One can observe, how the origin of the eigenspace (depicted as a square) and the orientation of the principal axis change through time, adapting to the new points, which come into the process. In the end, the estimated eigenspace, which encompasses all training points, is almost equal to the eigenspace obtained using the batch method.

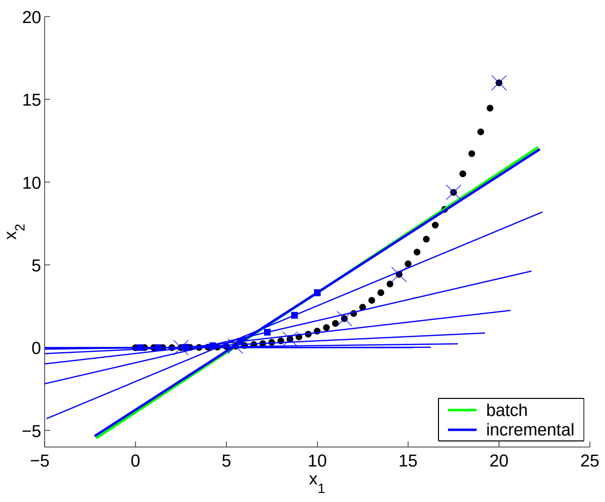


Figure 3.2: Illustration of incremental learning.

3.2.2 Comparison with related work

The proposed algorithm is similar to the algorithm introduced by Hall et al. [32]. Furthermore, both algorithms estimate an identical subspace. However, the subspace is obtained in different ways. The core of both algorithms is SVD of a covariance matrix, which is obtained differently. Nevertheless, the estimated covariance matrix is equal in both cases as we show in Appendix C.1. However, the diversity of the calculation of the covariance matrix yields to some differences in the properties of the algorithms.

In contrast to our method, which between the learning steps passes coefficient vectors of all training images, the Hall's method passes eigenvalues only. This can be advantageous, since less data is being passed from step to step and calculation of the covariance matrix is faster. However, this can also be disadvantageous, because the coefficients are not estimated and maintained during the learning process, thus less information is available.

Our algorithm calculates the coefficients at that time instant, when the particular image is added to the model, and then maintains their values throughout the process of incremental learning. This is slightly slower, however it has two advantages. The first advantage is, that each image can be discarded immediately after it has been used for updating the subspace. This is very appropriate (and possibly required) for on-line scenarios. And finally, since more information is preserved, our method can be advanced into a method for weighted learning of eigenspaces, which is impossible for Hall's method. This makes our method more suitable for application in a robust framework as will be shown in the following chapters.

3.3 Experiments and discussion

Principal component analysis in its standard batch form is optimal in the sense of the squared reconstruction error. Thus, its incremental version necessarily degrades the results. But, how severe is this degradation? Are the results still usable? What additional factors influence the results? To clarify these issues and to evaluate the proposed algorithm we will answer the following questions:

- How much does the incremental method degrade the results in comparison with the batch method?

- How does discarding of training images influence the results?

- How does the order of the training images influence the results?
- How (when) to increase the dimension of the subspace?
- How does the amount of energy contained in the subspace influence the results?
- What is the relation between the number of training images, the dimension of the subspace and the reconstruction error?

The answers will be corroborated by the results of extensive experiments. All the experiments were performed using all the images from the COIL20 database downsampled to the size 32×32 pixels. Half of the images (every second image) were used as a training set, while the other half of the images composed the test set. Thus, the training set consisted of 720 images; 36 images of each of 20 objects. All the objects are depicted in Fig. 3.3.

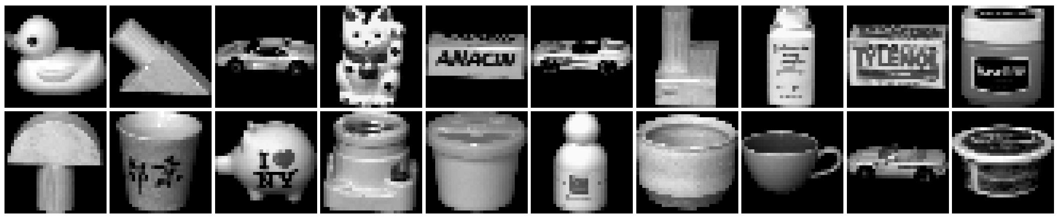


Figure 3.3: Objects from COIL20 database.

The results will be presented as mean squared reconstruction errors. Since the results on the test set were very similar to the results using the training images, we will present only the results of the latter.

3.3.1 Discarding images

The model, i.e., the representation of objects, which is obtained at the end of the learning process, consists of the estimated subspace (principal axes) and of the coefficients (principal components) of the training images. The proposed method estimates both – the subspace and the coefficients. The coefficients are estimated at that time instant, when the particular image is added to the model, and only the low-dimensional representation of the image is stored and then maintained throughout the process of incremental learning, allowing the high-dimensional images to be discarded immediately after the update. However, during the re-estimation of the coefficients at each step of incremental learning, some information is lost. Therefore, one can expect that at the end of the learning stage these coefficients are not as good as the coefficients, which are obtained by projecting the original training images into the final principal subspace. Certainly, for the latter case, the images have to be kept in the memory until the end of the learning stage. The question is, how does the discarding of images influence the results? And how much does the incremental method degrade the results in comparison with the batch method at all?

To answer these questions we performed the following experiment. We built eigenspaces of various dimensions from all training images. The images were coming into the learning process in a sorted sequential order, i.e., first all images of the first object, then all images of the second object, and so on.

Fig. 3.4(a), depicts the mean squared reconstruction errors (MSRE) of the images reconstructed from the coefficients obtained by projecting the training images into the eigenspaces, which were built using the batch method (in plots indicated as *batch*) and the proposed incremental method (*incX*). The results are very similar; MSRE obtained using the incremental method is only 3.1% worse on the average. The curve *incA* represents reconstruction errors of images obtained from the coefficients, which were calculated at that time instant, when the particular image was added to the model and then maintained throughout the process of incremental learning. Using this approach, an image can be discarded immediately after the model is updated. As one can observe, the squared reconstruction errors are still quite similar. In this case, the degradation of the results is 8.6% on the average.

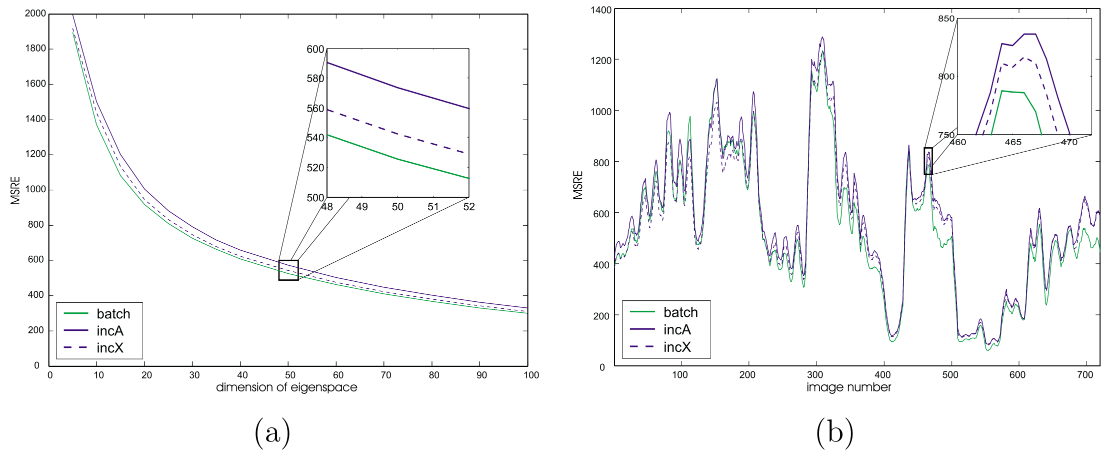


Figure 3.4: Comparison of batch and incremental method with and without discarding training images. MSRE produced by batch and incremental approach: (a) for various dimensions of eigenspace, (b) for dimension 50.

Fig. 3.4(b) shows the MSRE of all 720 images for the dimension of the eigenspace 50. One can observe, that the curves representing the incremental approach follow the curve produced by the batch method very closely without large deviations over the whole sequence of images.

All the results clearly indicate that the incremental method is almost as effective as the batch. This assertion especially holds true when the training images are not discarded. This means that the principal axes are estimated very accurately. However, also when the training images are discarded during the learning stage the squared reconstruction error is always below 10%, which means that the coefficients are still estimated well enough for most applications.

3.3.2 Order of training images

In the second experiment we changed the training sequence by giving the training images to the learning process in a random order. Thus, the eigenspace in the early stage of the learning process already encompassed images of several objects. Therefore, it was a good approximation of the final eigenspace. The incoming training images in the later stages were just refining the current eigenspace.

Consequently, the results have improved. They are shown in Fig. 3.5. MSRE produced by *incArnd* and *incXrnd* approaches, are only 3.1% and 1.3% worse than the results of the batch method, respectively. For comparison, the results obtained using the ordered sequence of the training images are also depicted (*incAseq* and *incXseq*).

What really matters is the order of the training images in the early stages of the learning process. To obtain optimal results, these images should be heterogeneous, encompassing different objects and views. This assures that the evolving eigenspace is rich and comprehensive enough in the beginning of the learning process already and that it is not specialized for representing a specific object only. In this way, the eigenspace can be adapted to the images of all objects more effectively.

3.3.3 Increasing subspace dimension

The proposed algorithm increases the subspace dimension by one. After the update we can discard the least significant basis vector to keep the subspace dimension small. The important issue is how (when) to increase the dimension of the subspace.

Here we present the results of three different strategies for increasing the subspace

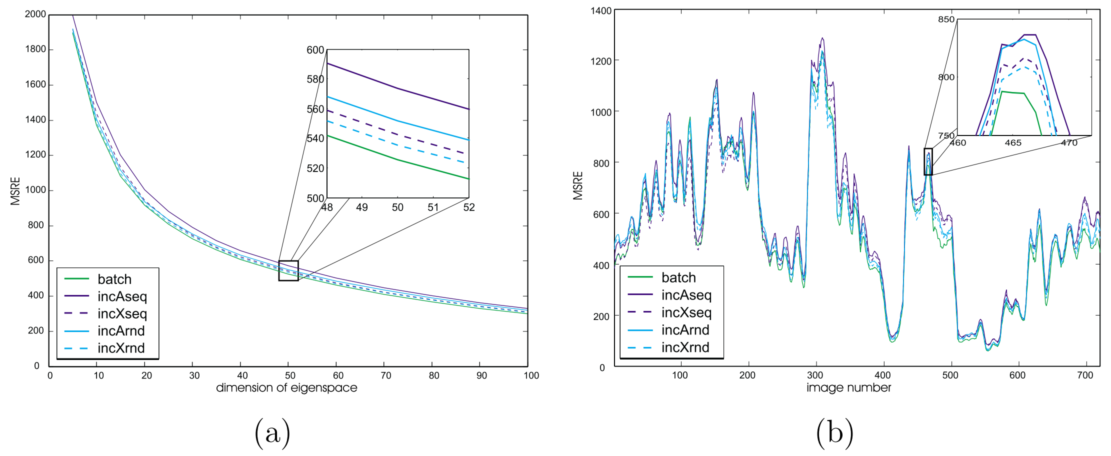


Figure 3.5: Comparison of batch and incremental method using ordered and random sequence of training images. MSRE (a) for various dimensions of the eigenspace, (b) for dimension 50.

dimension; each of them produced a subspace of the dimension 20 at the end of the learning process:

- *inc20ni20*: First we built the 20-dimensional subspace from the first 20 images using the batch method. The rest of the images were added using the incremental algorithm without further increasing the dimension of the subspace.

- *inc1ai20*: We started the learning process with the first image and added the next 19 images using the incremental method keeping all basis vectors. The rest of the images were then incrementally added without further increasing the dimension of the subspace.

- *inc1ci20*: We started the learning process with the first image. Then we were adding the rest of the images using the incremental method. After each update we decided to keep the least significant basis vector or not based on the error, which would be produced by discarding it [4], with the goal to obtain the dimension 20 at the end of the learning process.

All the results are given for the case when the training images are discarded ($inc{*}A$) and preserved ($inc{*}X$). They are also compared with the optimal results obtained with the batch method on all training images keeping the first 20 principal vectors (*batch20*). The results are presented in Table 3.1. They are given as absolute MSRE and as the relative increase (in %) of MSRE with regard to the batch method (in brackets).

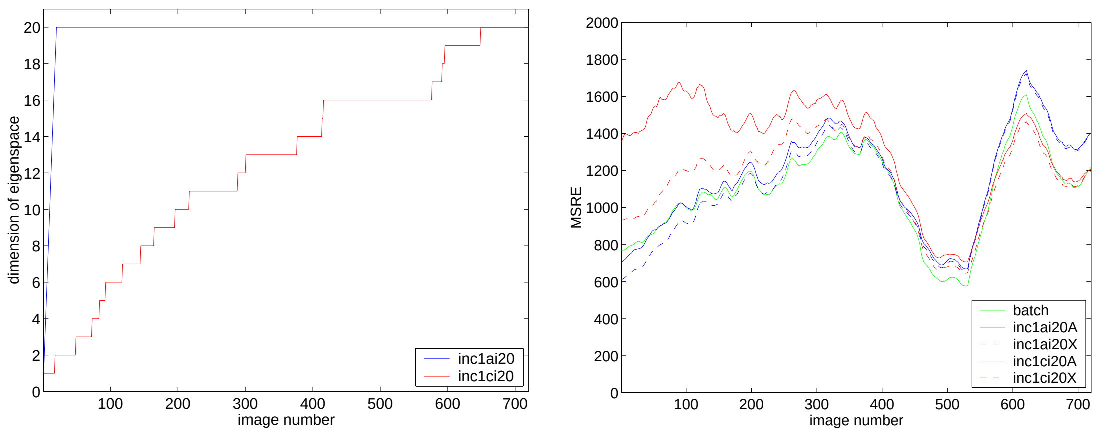
Table 3.1: MSRE for different learning strategies.

batch20	1079	
inc20ni20A	1151	(6.67)
inc20ni20X	1110	(2.87)
inc1ai20A	1151	(6.67)
inc1ai20X	1110	(2.87)
inc1ci20A	1323	(22.61)
inc1ci20X	1152	(6.76)

We can observe that *inc20ni20* and *inc1ai20* strategies yield identical results. This is expected, since *inc1ai20* incrementally builds the 20-D eigenspace from the first 20 images without discarding any basis vector thus without any loss of information. Therefore the obtained initial 20-D subspace is exactly the same as obtained by the batch method in *inc20ni20*.

We can also observe that *inc1ci20* produced significantly inferior results despite the fact that the subspace dimension at the end of the learning process was the same in all cases. The reason for this is that the dimension of the evolving subspace was smaller during the process of learning. This is evident from Fig. 3.6(a), which depicts the growth of the dimension of the evolving subspace for *inc1ai20* and *inc1ci20*. For that reason, the reconstructions of the representations of the images from the beginning of the training sequence are significantly worse as can be observed on Fig. 3.6(b). The results are improved if we do not discard the training images (*inc1ci20X*), which indicates that the subspace is estimated reasonably well and that the main cause for the reconstruction errors are inferior final approximations of the coefficients of the initial training images.

Therefore, if we know what the dimension of the final subspace is supposed to be (e.g., we know what compression ratio we want to achieve), we should keep increasing the subspace until this dimension is achieved and then keep preserving this dimension until the end of the learning process. However, if the final dimension is not known in advance, some criterion for increasing/preserving the subspace dimension should be used. To achieve better results, this criterion should be dynamic. It should be less conservative at the beginning of the learning process to enable better reconstruction of the initial images. Afterwards, this criterion should be more conservative to keep the dimension of the evolving subspace small.



(a) (b)

Figure 3.6: (a) Growth of subspace dimension. (b) MSRE for different learning strategies.

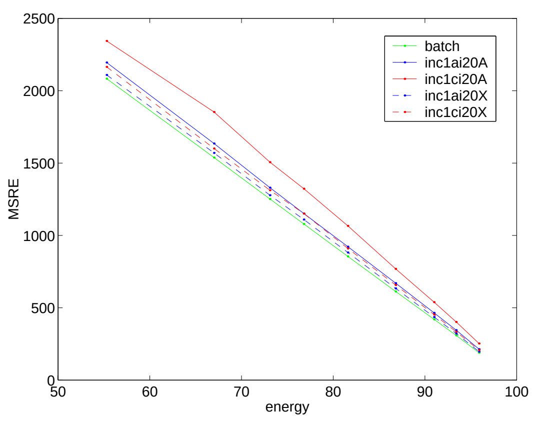
3.3.4 Energy of subspace

The next question we want to answer is how does the amount of the energy contained in the subspace influence the results. To find out we run the algorithms nine times producing eigenspaces of nine different dimensions containing different amounts of the energy. The results are given in Table 3.2 and in Fig. 3.7.

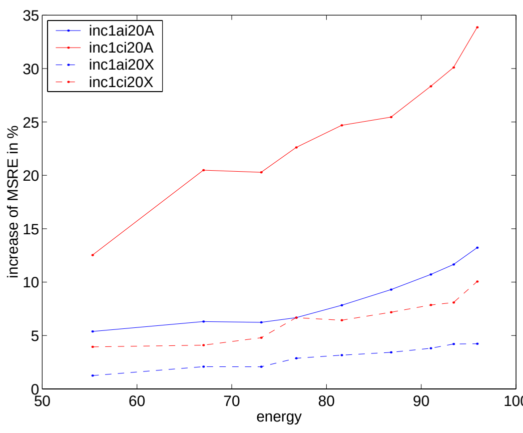
Table 3.2: MSRE for various sizes of eigenspaces.

			inc1ai20				*inc1ci20*			
K	*en*	*batch*	*A*		*X*		*A*		*X*	
5	55.3	2083	2195	(5.4)	2109	(1.2)	2344	(12.5)	2165	(3.9)
10	67.0	1538	1635	(6.3)	1570	(2.1)	1853	(20.5)	1601	(4.1)
15	73.1	1252	1330	(6.2)	1278	(2.1)	1506	(20.3)	1312	(4.8)
20	76.8	1079	1151	(6.7)	1110	(2.9)	1323	(22.6)	1151	(6.7)
30	81.6	855	922	(7.8)	882	(3.2)	1066	(24.7)	910	(6.4)
50	86.8	613	670	(9.3)	634	(3.4)	769	(25.4)	657	(7.2)
80	91.0	420	465	(10.7)	436	(3.8)	539	(28.3)	453	(7.9)
110	93.4	309	345	(11.7)	322	(4.2)	402	(30.1)	334	(8.1)
164	95.9	189	214	(13.2)	197	(4.2)	253	(33.9)	208	(10.1)

Fig. 3.7(a) depicts MSRE for five learning strategies with respect to the energy, which was contained in the estimated subspaces. As expected, MSRE of the batch



(a)



(b)

Figure 3.7: (a) Dependence of MSRE on energy. (b) Relative increase of MSRE with regard to the batch method.

method decreases perfectly linearly. Similar trends can also be observed for the incremental method. One can also observe that the absolute difference between the batch and the incremental method is bigger when the dimension of the subspace is small, since the errors themselves are bigger as well.

Fig. 3.7(b) is more interesting. It shows a relative increase of MSRE of the incremental strategies with regard to the batch method. One can observe that the percentage of the increase goes up slightly when the dimension of the subspace grows. This trend is similar for all four strategies of incremental learning.

3.3.5 Number of training images

A very interesting question is also what is the relation between the number of training images, the dimension of the subspace and the reconstruction error. First we will explore how does the number of training images influence the results, if the encompassed energy is constant.

For that purpose we performed the following experiment. First we built the principal subspace from the first 36 images (all the training images of the first object), then from the first 72 images (two objects), and so on, until we have built the principal subspace from all 720 images (20 objects). At each run we kept a number of principal vectors, which encompassed 85% of the energy. We performed all these runs in the batch mode and in the incremental mode using the strategy analogous to *inc1ai20*. At the end of each run, both algorithms produced the eigenspace of the same size.

The results are shown in Fig. 3.8. The plot in Fig. 3.8(a) depicts an average MSRE in all already learned objects (at each step one object more has been encompassed in the representation). One can observe that the relationship between the batch and the incremental method is very similar throughout the entire process of learning. This means that on average the number of images (objects) does not significantly influence the degradation of the results in comparison with the batch method.

However, in Fig. 3.8(b) we can observe that this assertion does not hold true for individual objects. It shows MSRE throughout time for five individual objects. We can see that using the incremental method the first objects are reconstructed considerably better than rest of the objects. This is due to the strategy of the incremental learning, which in the early stages operates with the subspace of the same dimension as at the end. This means that the initial images were stored encompassing more than 85% of their energy. Thus, the initial eigenspace was very specialized for the initial objects enabling their excellent reconstruction at the end as well. This effect is also clear from Fig. 3.8(c), which shows a relative increase of MSRE with regard to the batch method for all objects at the end of the learning process when all the objects have been encompassed into the representation.

Figs. 3.8(d) depicts a relative increase of MSRE for five objects after each run. It confirms the findings from Fig. 3.8(b) that the relative increase of the error for the individual objects does not grow when new objects are added to the representation.

We can conclude that the number of training images does not significantly influence the degradation of the results of the incremental learning, at least in the performed experiments where the number of training images was not extremely large. However, the objects from the beginning of the learning process are reconstructed better, due to the strategy of the incremental learning. We could suppress this effect by modifying the criterion for increasing/preserving the dimension of the evolving eigenspace or by applying the weighted version of the incremental learning.

3.3.6 Compression ratio

The last question that we will answer is how does the number of the training images (objects) influence the growth of the subspace dimension and reconstruction error. How long should we keep increasing the subspace dimension? What compression ratio can we achieve?

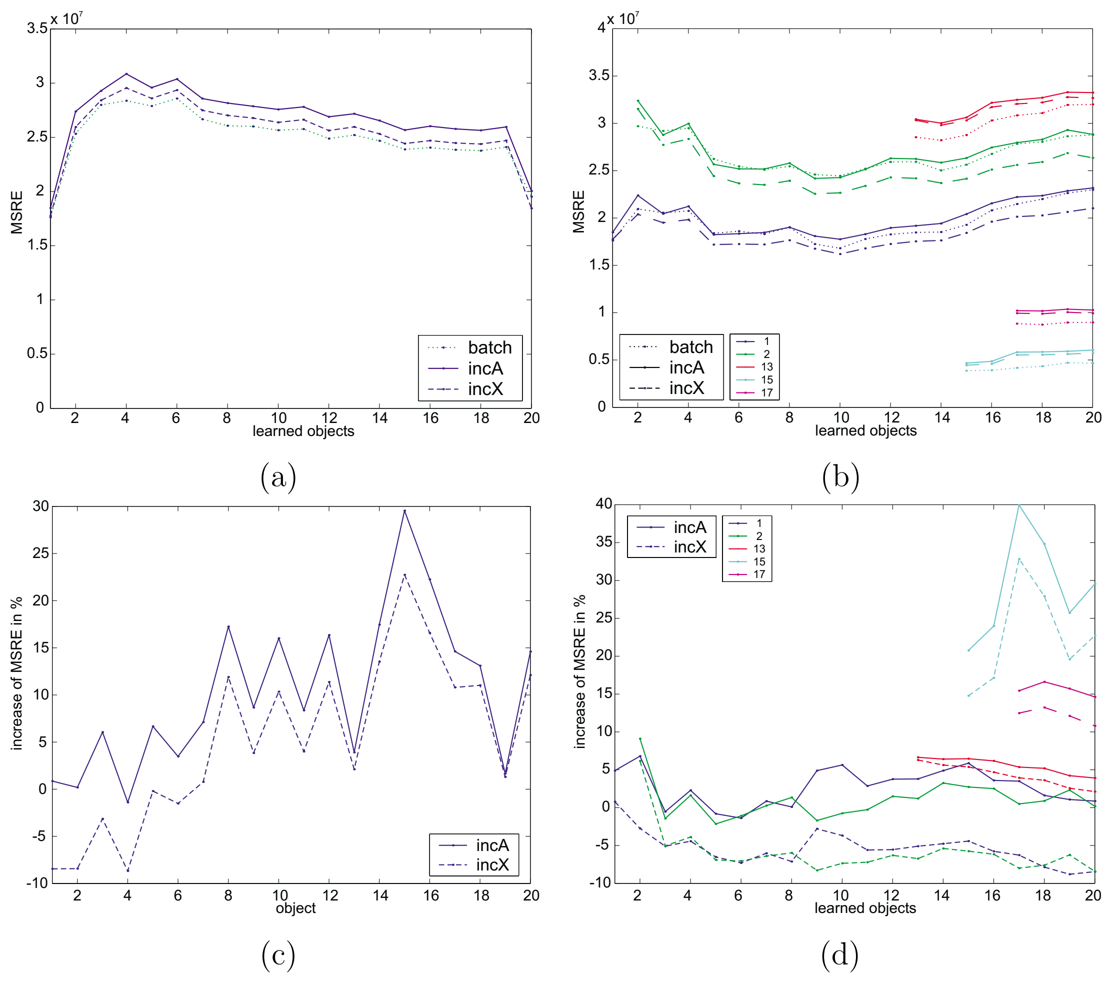
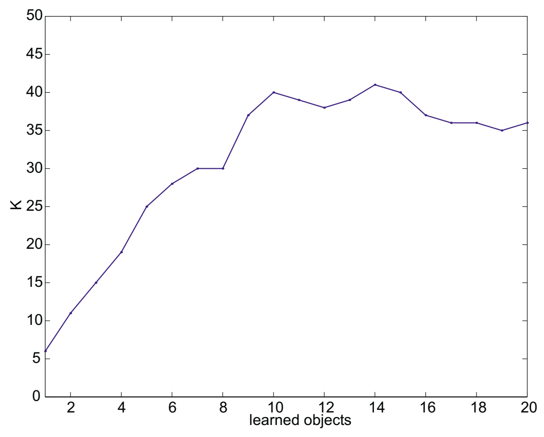
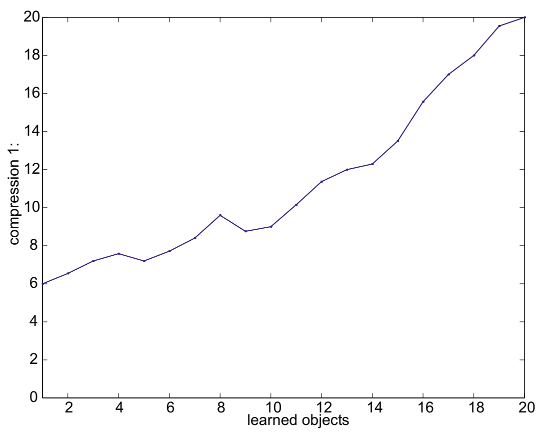


Figure 3.8: (a) Average MSRE in already learned images. (b) MSRE for five objects after various phases of the learning process. (c) Relative increase of MSRE for all objects at the end of the learning process. (d) Relative increase of MSRE for all five objects after various phases of the learning process.

Fig. 3.9(a) shows the number of principal vectors which are required to preserve 85% of the energy of various numbers of objects. It is very interesting to observe that the required dimension K does not increase any more after the tenth object has been added to the representation. Consequently, the compression ratio grows from 1:9 at ten objects (360 images) to 1:20 at 20 objects (720 images) as depicted in Fig. 3.9(b). This is partially due to the fact that some of the objects from the second half of the learning sequence are similar to the objects from the first half of the sequence (e.g., the two toy cars) and they do not require new dimensions for suitable representation.



(a)



(b)

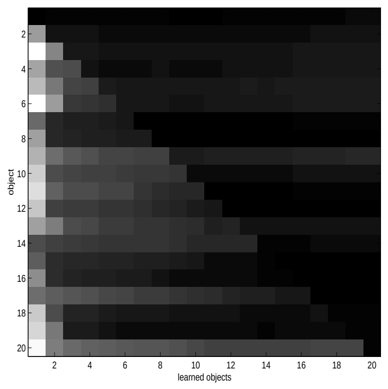
Figure 3.9: (a) Dimension of the principal subspace, which preserves 85% of the energy. (b) Compression ratio.

Moreover, it seems that the visual variability of the first half of the images also encompasses the variability of the second half. One could expect that after the first half of the images has been encompassed into the representation, the obtained eigenspace is big enough and very general to represent any image. However, this is not the case. We will demonstrate this with the following experiment.

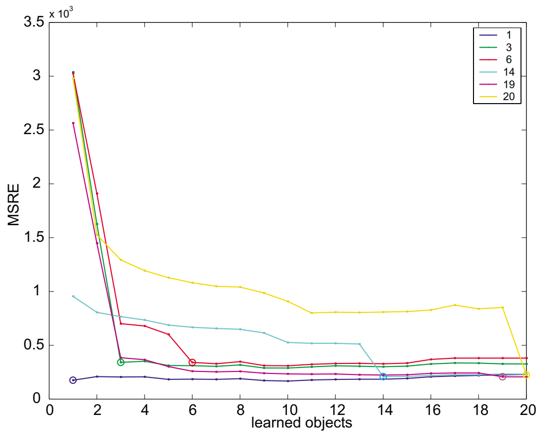
After each run of the algorithm (after a new object was added to the representation), we projected all the training images into the eigenspace. All MSRE are depicted in Fig. 3.10(a). For each step (the number of already learned objects — abscissa) and for every object (ordinate) the grey level of the corresponding matrix element represents the MSRE of the reconstructions of the particular object after the particular number of objects have been learned. One can observe that the values in the upper triangle are significantly smaller than the values in the lower triangle. This means that MSRE drops after an object is added to the representation. This is the case in the left half of the matrix (the first objects) as well as in the right half of the matrix (the last objects). This means that although the dimension of the subspace does not keep increasing when the objects from the second half of the training sequence are being added, the novel images rotate the evolving subspace in the direction, which improves their reconstruction.

This is illustrated in Fig. 3.10(b) as well. It depicts MSRE for six objects after each run of the algorithm. For five objects MSRE significantly drops after the object is added to the representation (marked with a circle). The only exception is the object no. 19, whose error has very similar behavior as the error of the object no. 5. This is not surprising, since both objects are very similar toy cars, thus representing very similar visual variability.

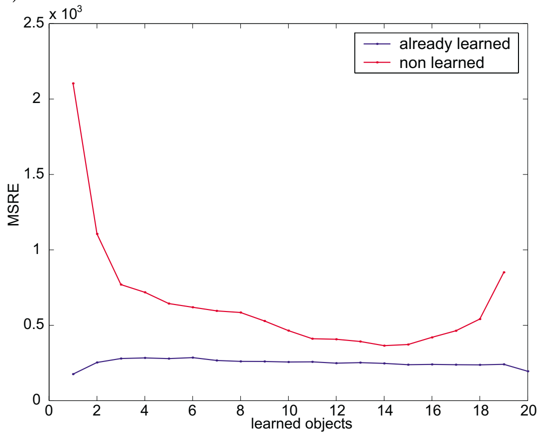
The summary of these results is depicted in Fig. 3.10(c). It shows the average MSRE of all already learned images and the average MSRE of all not yet learned images. We can observe that the MSRE of the already learned images is always very similar. The MSRE of the rest of the images is at the beginning very large, since the eigenspace is specialized for a few objects which were presented to the learning process. As the number of the learned objects increases, the eigenspaces becomes richer, which enables better reconstruction of all images. However the MSRE of the non-learned objects never reaches the MSRE of the already learned objects.



(a)



(b)



(c)

Figure 3.10: (a) MSRE for all objects after various phases of learning process. (b) MSRE for six objects. (c) Average MSRE in all already learned objects and in the rest of the objects.

Based on the presented results, the final conclusion would be that the principal subspace keeps growing until some limit is reached. After that only the rotations of the principal subspace are applied to keep adapting to the new images. Still, these rotations considerably improve the reconstruction results of the added images.

3.4 Chapter summary

In this chapter we presented a novel algorithm for incremental learning of eigenspaces. It takes the training images sequentially and computes the new eigenspace from the subspace obtained in the previous step and the current input image. We compared the algorithm with the most similar related work and exposed its advantages and disadvantages. We also performed a number of experiments to evaluate the proposed method and to clarify various issues about the incremental learning.

There are three main advantages of the proposed method. Firstly, it is simple, concise, easy to understand and easy to implement. Secondly, it estimates the representations of the training images and then maintains them throughout the entire process of learning, thus the training images can be discarded. And finally, it can be advanced into the algorithm for weighted incremental learning. We will discuss this topic in the next chapter.

Chapter 4

Weighted PCA

4.1 Chapter overview

Previous experience, prior knowledge, and the information obtained by higher-level cognitive processes affect the level to which the training images should be incorporated in the representation. Some pixels are often more informative or more reliable than others. The significance of the training images can also vary. Therefore, a learning algorithm should enable a selective influence of training images as well as pixels in individual images to the process of learning. However, in the standard PCA approach, all pixels of an image receive equal treatment. Also, all the training images have equal influence on the estimation of the principal axes.

In this chapter we present a generalized PCA approach, which estimates principal axes and principal components considering weighted pixels and images. First we will present a batch algorithm for weighted PCA, where the weights can be set to arbitrary values. Then a special case of this algorithm will be discussed; the algorithm for handling missing pixels, which considers weights of binary values only. Both algorithms will be then incorporated in the incremental framework and extensively evaluated in various experiments.

4.2 Batch algorithm for weighted PCA

The standard PCA can be extended into the weighted version by introducing weights into Eq. (2.25). The weights are assembled in the matrix $\mathbf{W} \in \mathbb{R}^{M\times N}$, where w_{ij} is the weight of the i-th pixel in the j-th image. The goal is to minimize the *weighted* squared reconstruction error

$$\mathcal{E} = \sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij} \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2 \quad . \tag{4.1}$$

Here, the values of the matrix $\hat{\mathbf{X}}$ are obtained by subtracting the *weighted* mean vector $\boldsymbol{\mu}$ from the training images $\mathbf{x}_j$. The elements of the weighted mean vector are calculated as

$$\mu_i = \frac{\sum_{j=1}^{N} w_{ij} x_{ij}}{\sum_{j=1}^{N} w_{ij}} \quad , \quad i = 1 \dots M \quad . \tag{4.2}$$

In practice, it is useful to deal with two types of weights: *temporal* weights ${}^{\mathsf{t}}\mathbf{w} \in \mathbb{R}^{1\times N}$, which put different weights on individual images and *spatial* weights ${}^{\mathsf{s}}\mathbf{w} \in \mathbb{R}^{M}$, which put different weights on individual pixels within an image[1]. Since different types of weights yield different algorithms for estimating the weighted principal subspace, we will discuss both types of weights separately.

4.2.1 Temporal weights

Temporal weights determine how important for estimation of principal subspace the individual images are. If the temporal weight of an image is larger than the weights of other images, the reconstruction error of this image should be smaller than the reconstruction errors of the other images. Similarly, the contribution of its principal components to the estimation of the variances should be larger in comparison with other principal components.

Since all the pixels in an image are equally treated, the weight matrix $\mathbf{W}$ contains the row vector ${}^{\mathsf{t}}\mathbf{w}$ in each row, thus $\mathbf{W} = \mathbf{1}_{M\times 1}{}^{\mathsf{t}}\mathbf{w}$ and the minimization problem (4.1) reduces to:

$$\mathcal{E} = \sum_{i=1}^{M}\sum_{j=1}^{N} {}^{t}w_j \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2 \quad , \tag{4.3}$$

where the *weighted mean* vector, which is used to obtain $\hat{\mathbf{X}}$, is calculated as

$$\boldsymbol{\mu} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} \sum_{j=1}^{N} {}^{t}w_j \mathbf{x}_j \quad . \tag{4.4}$$

[1]The left superscript is used to distinguish between temporal (${}^{\mathsf{t}}\mathbf{w}$) and spatial (${}^{\mathsf{s}}\mathbf{w}$) weights. ${}^{\mathsf{t}}\mathbf{w}$ is a row vector, while ${}^{\mathsf{s}}\mathbf{w}$ is a column vector.

This minimization problem can be solved using a modified EM algorithm or by algorithm which maximizes the weighted variance. The latter can be derived as follows. Following (2.2) and (2.3) the weighted mean of the projections of input vector onto a direction $\mathbf{u}$ can be expressed as

$$\bar{a} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} \sum_{j=1}^{N} {}^{t}w_j a_j = \sum_{i=1}^{M} u_i \mu_i \tag{4.5}$$

and the weighted variance as

$$\sigma^2 = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} \sum_{j=1}^{N} {}^{t}w_j (a_j - \bar{a})^2 = \sum_{i=1}^{M} \sum_{l=1}^{M} u_i c_{il} u_l = \mathbf{u}^\top \mathbf{C} \mathbf{u} \ , \tag{4.6}$$

where $\boldsymbol{\mu}$ is the *weighted mean* (4.4) and $\mathbf{C}$ is the *weighted covariance matrix*, whose elements are calculated as

$$c_{il} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} \sum_{j=1}^{N} {}^{t}w_j \hat{x}_{ij} \hat{x}_{lj} \ . \tag{4.7}$$

If we define the matrix ${}^{\mathbf{t}}\hat{\mathbf{X}}$ by re-scaling the input vectors centered around the weighted mean:

$${}^{\mathbf{t}}\hat{\mathbf{x}}_j = \sqrt{{}^{t}w_j} \hat{\mathbf{x}}_j, \ \ j = 1 \ldots N \ , \tag{4.8}$$

the weighted covariance matrix can be conveniently written as

$$\mathbf{C} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} {}^{\mathbf{t}}\hat{\mathbf{X}} {}^{\mathbf{t}}\hat{\mathbf{X}}^\top \ . \tag{4.9}$$

Considering different formulation of the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\mathbf{C}$, the minimization of (4.6) is equivalent to the minimization of (2.3). Therefore, it can be solved by using a modified standard PCA algorithm as presented in Algorithm 4.

Using this algorithm, the estimated principal subspace does not depend on all training images equally. For instance, if a training image has weight 2, while all other images have weight 1, the result of the proposed algorithm is the same as the result of the standard algorithm, which has two copies of the particular image in the training set.

Algorithm 4 : TWPCA – temporally weighted PCA

```
Input: data matrix $\mathbf{X}$, temporal weights $^{\mathbf{t}}\mathbf{w}$
Output: weighted mean vector $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$.
 1: Estimate the weighted mean vector: $\boldsymbol{\mu} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} \sum_{j=1}^{N} {}^{t}w_j \mathbf{x}_j$ .
 2: Scale the input data centered around the weighted mean:
    $^{\mathbf{t}}\hat{\mathbf{x}}_j = \sqrt{^{t}w_j}(\mathbf{x}_j - \boldsymbol{\mu}),\ \ j = 1 \dots N$ .
 3: if $M \leq N$ then
 4:     Estimate the weighted covariance matrix: $\mathbf{C} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} {}^{\mathbf{t}}\hat{\mathbf{X}} {}^{\mathbf{t}}\hat{\mathbf{X}}^{\top}$ .
 5:     Perform SVD on $\mathbf{C}$. Obtain the eigenvectors $\mathbf{U}$ and the eigenvalues $\boldsymbol{\lambda}$.
 6: else
 7:     Estimate the weighted inner product matrix: $\mathbf{C}' = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j} {}^{\mathbf{t}}\hat{\mathbf{X}}^{\top} {}^{\mathbf{t}}\hat{\mathbf{X}}$ .
 8:     Perform SVD on $\mathbf{C}'$. Obtain the eigenvectors $\mathbf{U}'$ and the eigenvalues $\boldsymbol{\lambda}'$.
 9:     Determine the principal vectors $\mathbf{U}$: $\mathbf{u}_i = \frac{^{\mathbf{t}}\hat{\mathbf{X}}\mathbf{u}'_i}{\sqrt{\sum_{j=1}^{N} {}^{t}w_j}\sqrt{\lambda'_i}}$ , $i = 1 \dots N$ .
10:     Determine the eigenvalues $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$ .
11: end if
```

4.2.2 General weights

Before we derive the algorithm for estimating principal subspace using arbitrary weights, we will discuss the weighed estimation of subspace coefficients considering spatial weights. Spatial weights control the influence of individual pixels within an image. Therefore, if a part of an image is unreliable or not important for the estimation of principal components, its influence can be diminished by decreasing the weights of the corresponding pixels.

The coefficients are traditionally estimated by the standard projection (2.20) of an image $\mathbf{x}$ onto the principal vectors. From these coefficients, the reconstructed image $\mathbf{y}$ can be obtained using (2.21). The coefficient estimation can also be formulated as a problem of finding a coefficient vector $\mathbf{a}$ that minimizes the squared reconstruction error between the original image $\mathbf{x}$ and its reconstruction $\mathbf{y}$, thus minimizing

$$e = \|\hat{\mathbf{x}} - \mathbf{U}\mathbf{a}\|^2 = \sum_{i=1}^{M} \left(\hat{x}_i - \sum_{j=1}^{k} u_{ij} a_j \right)^2 \quad . \tag{4.10}$$

The solution to this minimization problem (the vector $\mathbf{a}$) can be found by solving an over-constrained system of linear equations

$$\hat{x}_i = \sum_{j=1}^{k} a_j u_{ij} \quad , \quad i = 1 \dots M \tag{4.11}$$

in the least squares sense, where $\hat{x}_i$ and u_{ij} are known and a_j unknown variables [40, 70].

Now, we are able to introduce the spatial weights ${}^{\mathbf{s}}\mathbf{w}$ into the estimation of coefficients. The goal is to find a coefficient vector $\mathbf{a}$ that minimizes the *weighted* squared reconstruction error:

$$e = \sum_{i=1}^{M} {}^{s}w_i \left(\hat{x}_i - \sum_{j=1}^{N} u_{ij} a_j \right)^2 \quad . \tag{4.12}$$

This can be achieved by putting different weights in the equations in (4.11). The coefficients can be calculated by solving the following system of linear equations:

$$\sqrt{{}^{s}w_i}\hat{x}_i = \sqrt{{}^{s}w_i} \sum_{j=1}^{k} a_j u_{ij} \quad , \quad i = 1 \dots M \ . \tag{4.13}$$

In the minimization process, the equations arising from the pixels with large weights will have larger influence on the estimation of the coefficients and the reconstruction error in such pixels will be smaller.

Since solving an over-constrained system of linear equations is equivalent to the pseudoinverse (see Appendix B.1), we can formulate this minimization problem also in a more concise way[2]:

$$\mathbf{a} = ((.\sqrt{{}^{\mathbf{s}}\mathbf{w}}\mathbf{1}_{1\times k}) \circ \mathbf{U})^{\dagger} (.\sqrt{{}^{\mathbf{s}}\mathbf{w}} \circ \hat{\mathbf{x}}) \ . \tag{4.14}$$

In the similar way we can also approach to the weighted estimation of principal subspace considering arbitrary weights, thus to minimizing (4.1). The weighted squared reconstruction error (4.1) can be minimized using a modified EM algorithm. By noting that $\mathbf{a}_j = (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \hat{\mathbf{x}}_j = \mathbf{U}^\dagger \hat{\mathbf{x}}_j$ we can replace E-step of the EM algorithm by solving the system of linear equations (4.11) for each $\mathbf{a}_j$, $j = 1 \dots N$. A very similar observation holds also for M-step. Therefore, if we consider weights as well, we can estimate the principal subspace by iteratively solving the following systems of linear equations:

[2] $.\sqrt{\mathbf{A}}$ is an operator that calculates the square root of each element of the matrix $\mathbf{A}$. $\mathbf{A} \circ \mathbf{B}$ denotes the Hadamard (point wise) product between two matrices of equal dimension.

- **E-step**: Estimate $\mathbf{A}$ in the following way: For each image j, $j = 1 \dots N$, solve the following system of linear equations in the least squares sense:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}} \sum_{p=1}^{k} u_{ip} a_{pj} \ , \ \ i = 1 \dots M \ . \tag{4.15}$$

- **M-step**: Estimate $\mathbf{U}$ in the following way: For each pixel i, $i = 1 \dots M$, solve the following system of linear equations in the least squares sense:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}} \sum_{p=1}^{k} u_{ip} a_{pj} \ , \ \ j = 1 \dots N \ . \tag{4.16}$$

At convergence, the columns of $\mathbf{U}$ span the space of the first k principal axes, but they are not oriented in the directions of the principal components and $\mathbf{U}$ is not orthogonal. However, when we are dealing with weighted learning, we usually use a weighted method for estimation of principal components (4.14) in the recognition stage as well. Since this method is also based on solving the system of linear equations, $\mathbf{U}$ does not have to be orthogonal at all. Nevertheless, we can still orthogonalize $\mathbf{U}$ to enable calculation of the coefficients using the standard projection (if the weights of all pixels are equal). However, one has to be aware that even after the orthogonalization the directions of $\mathbf{u}_j$, $j = 1 \dots k$, are not the principal directions. By using a similar approach to determination of principal directions as in the original EM algorithm, we can obtain only an approximation of the true principal directions, since this approach is based on the standard non-weighted PCA on the coefficient vectors. Thus, the total variance encompassed with k basis vectors is as large as possible, however the distribution of this variance over the individual basis vectors is not optimal. As a consequence, a $k-1$ dimensional subspace, obtained by discarding the least significant basis vector $\mathbf{u}_k$, cannot be considered as principal any more.

By expressing (4.15) and (4.16) in a more concise way, the modified EM algorithm for weighted PCA is shown in Algorithm 5.

4.2.3 Simple 2-D example

We will demonstrate the behavior of the proposed algorithm on our simple 2-D example with 41 2-D points shown in Fig. 4.1(a). The goal is to estimate 1-D principal subspace.

Algorithm 5 : WPCA – weighted PCA

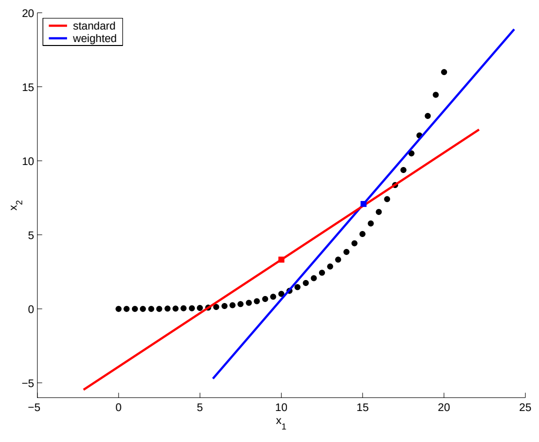
Input: data matrix $\mathbf{X}$, weight matrix $\mathbf{W}$, number of principal axes to be estimated k.

Output: weighted mean vector $\boldsymbol{\mu}$, $\mathbf{U}$ spanning principal subspace.

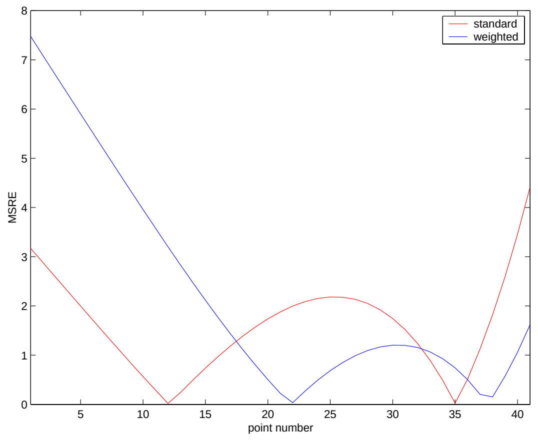
```
 1: Estimate the weighted mean vector: $\mu_i = \frac{\sum_{j=1}^{N} w_{ij} x_{ij}}{\sum_{j=1}^{N} w_{ij}}$ , $i = 1 \dots M$ .
 2: Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
 3: Set the elements of $\mathbf{U} \in \mathrm{I\!R}^{M\times k}$ to random values.
 4: repeat
 5:    E-step: $\mathbf{a}_j = ((.\sqrt{\mathbf{w}_j}\mathbf{1}_{1\times k}) \circ \mathbf{U})^\dagger(.\sqrt{\mathbf{w}_j} \circ \hat{\mathbf{x}}_j),\ j = 1 \dots N$ .
 6:    M-step: $\mathbf{u}_{i:} = (.\sqrt{\mathbf{w}_{i:}} \circ \hat{\mathbf{x}}_{i:})((\mathbf{1}_{k\times 1}.\sqrt{\mathbf{w}_{i:}}) \circ \mathbf{A})^\dagger,\ i = 1 \dots M$ .
 7: until convergence.
 8: Orthogonalize $\mathbf{U}$.
 9: Project input data on $\mathbf{U}$: $\mathbf{a}_j = ((.\sqrt{\mathbf{w}_j}\mathbf{1}_{1\times k}) \circ \mathbf{U})^\dagger(.\sqrt{\mathbf{w}_j} \circ \hat{\mathbf{x}}),\ j = 1 \dots N$ .
10: Perform PCA on $\mathbf{A}$. Obtain $\mathbf{U}'$ .
11: Rotate $\mathbf{U}$ for $\mathbf{U}'$: $\mathbf{U} = \mathbf{U}\mathbf{U}'$.
```

First, we put different weights on the training points. We set temporal weights to ${}^{t}w_j = j^2$ meaning that the points at the end of the training sequence should have a stronger influence on the estimation of the principal subspace. Consequently, the reconstructions of these points should be better. Fig. 4.1(a) depicts the principal axis estimated using the standard and the weighted PCA. The weighted mean vector is closer to the end of the point sequence, since the weights of these points have larger values. The principal axis is oriented in such a direction that enables superior reconstruction of these points. Consequently, the mean squared reconstruction error in the points at the end of the point sequence is much smaller when the weighted PCA is used, (see Fig. 4.1(b)). This results in a smaller weighted reconstruction error (1.78 for the standard and 0.97 for the weighted method).

Next, we put different weights on the coordinates. For each image, we set the spatial weight vector ${}^{\mathrm{s}}\mathbf{w} = [1.0, 0.1]^\top$, meaning that the first coordinate should have ten times bigger influence to the results than the second coordinate. The results are shown in Fig. 4.2(a). For both coordinates, the input value and the reconstructed values using the standard and the weighted method are presented. One can observe that the first coordinate is almost perfectly reconstructed with the weighted method in contrast to the standard method, which treats both coordinates equally. This is also evident from Fig. 4.2(b), which depicts the reconstruction errors in both



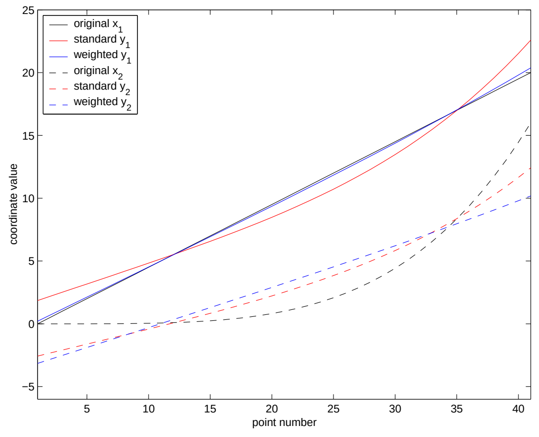
(a)



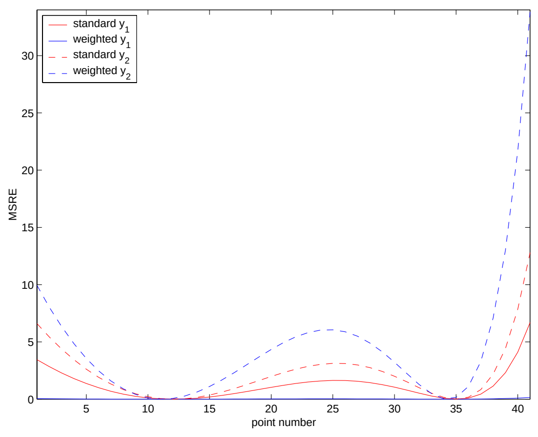
(b)

Figure 4.1: Weighted learning considering temporal weights. (a) Obtained principal axis. (b) Reconstruction errors.

coordinates of the individual points. MSRE in the first coordinate is very close to zero in all points when the weighted method is used. Consequently, this method produces significantly smaller weighted reconstruction error (57.93 for the standard and 19.98 for the weighted method).



(a)



(b)

Figure 4.2: Weighted learning considering spatial weights. (a) Reconstructed coordinate values. (b) Reconstruction errors in both coordinates.

4.3 Batch PCA on incomplete data

In the real world applications, it is often the case that not all data is available. The values of some pixels are missing or are totally unreliable. Such pixels are referred to as *missing pixels*. The estimation of the principal subspace in the case of incomplete data can be regarded as a special case of weighted PCA where the weights of missing pixels are set to zero. In this section we will consider this special case of the weighted learning more thoroughly.

4.3.1 Modified EM algorithm

First let us denote the sets of indices of non-missing (known) and missing pixels in the j-th image with $\mathcal{I}_j^\bullet$ and $\mathcal{I}_j^\circ$, respectively, and the sets of indices of non-missing and missing pixels in the i-th row of the data matrix $\mathbf{X}$ with $\mathcal{I}_{i:}^\bullet$ and $\mathcal{I}_{i:}^\circ$, respectively.

Now, the goal is to minimize the reconstruction error of known pixels, thus in E and M steps of the EM algorithm we have to set up the equations arising from the non-missing pixels only:

- **E-step**: Estimate $\mathbf{A}$ in the following way: For each image j, $j = 1 \dots N$, solve the following system of linear equations in the least squares sense:

$$\hat{x}_{ij} = \sum_{p=1}^{k} u_{ip} a_{pj} \ , \ \ i \in \mathcal{I}_j^\bullet \ . \tag{4.17}$$

- **M-step**: Estimate $\mathbf{U}$ in the following way: For each pixel i, $i = 1 \dots M$, solve the following system of linear equations in the least squares sense:

$$\hat{x}_{ij} = \sum_{p=1}^{k} u_{ip} a_{pj} \ , \ \ j \in \mathcal{I}_{i:}^\bullet \ . \tag{4.18}$$

Here, the mean image is obtained by estimating the mean over the known pixels:

$$\mu_i = \frac{1}{|\mathcal{I}_{i:}^\bullet|} \sum_{j \in \mathcal{I}_{i:}^\bullet} x_{ij} \ , \ \ i = 1 \dots M \ . \tag{4.19}$$

When dealing with images containing a considerable amount of missing pixels, such a formulation results in an ill-posed problem. The principal vectors are optimized to ensure the optimal reconstruction error in known pixels. Since the reconstruction error in missing pixels is not considered in the minimization process, the reconstructed missing pixels can have arbitrary values. The generalization ability of this algorithm is relatively weak, thus although such an algorithm can produce very small reconstruction errors in known pixels, it may at the same time produce very strange reconstructed values in missing pixels.

To alleviate this problem we impose additional application dependent constraints to the minimization process. When the images are ordered, as in the case of image sequences, we can extend the algorithm to include also a smoothness prior to enforce that the values of reconstructed missing pixels are changing smoothly over time. Thus, in the M-step we minimize the second derivative of the reconstructed missing pixels. The new M-step looks as follows:

- **M-step**: Estimate $\mathbf{U}$ in the following way: For each pixel i, $i = 1 \dots M$, solve the following system of linear equations in the least squares sense:

$$\begin{aligned} \hat{x}_{ij} &= \sum_{p=1}^{k} u_{ip} a_{pj} \ , \ \ j \in \mathcal{I}_{i:}^{\bullet} \\ 0 &= \alpha \sum_{p=1}^{k} u_{ip}(a_{p,j-1} - 2a_{pj} + a_{p,j+1}) \ , \ \ j \in \mathcal{I}_{i:}^{\circ} \ , \end{aligned} \tag{4.20}$$

  where α is the parameter which weights the influence of the smoothness constraint.

Thus, the overall algorithm minimizes the following error function:

$$\mathcal{E} = \sum_{j=1}^{N} \sum_{i \in \mathcal{I}_j^{\bullet}} \left(\hat{x}_{ij} - \sum_{p=1}^{k} u_{ip} a_{pj} \right)^2 + \alpha \sum_{j=1}^{N} \sum_{i \in \mathcal{I}_j^{\circ}} \left(\sum_{p=1}^{k} u_{ip} a''_{pj} \right)^2 \ . \tag{4.21}$$

To summarize the algorithm in a more concise way we will introduce a new notation. Let us partition a training image (a *column* in the data matrix) $\hat{\mathbf{x}}_j$ into $\hat{\mathbf{x}}_j^{\bullet}$ and $\hat{\mathbf{x}}_j^{\circ}$ vectors of $M^{\bullet}$ known and M° unknown values in $\hat{\mathbf{x}}_j$, respectively, and assign the corresponding *rows* of $\mathbf{U}$ to $\mathbf{U}^{\bullet}$ and $\mathbf{U}^{\circ}$. Similarly, we can partition a *row* in the data matrix $\hat{\mathbf{x}}_{i:}$ into $\hat{\mathbf{x}}_{i:}^{\bullet}$ and $\hat{\mathbf{x}}_{i:}^{\circ}$ row vectors of the $N^{\bullet}$ known and N° unknown values in $\hat{\mathbf{x}}_{i:}$, respectively, and assign the corresponding *columns* of $\mathbf{A}$ to $\mathbf{A}^{\bullet}$ and $\mathbf{A}^{\circ}$. Using this notation, the modified EM algorithm for the estimation of the principal subspace from incomplete data is given in Algorithm 6.

In the recognition stage, an input image $\mathbf{x}$ is projected into the principal subspace $\mathbf{U}$ in an analogical way. The coefficients a_j are obtained by solving an overconstrained system of linear equations, which arise from the non-missing pixels only:

$$\hat{x}_i = \sum_{j=1}^{k} a_j u_{ij} \ , \ \ i \in \mathcal{I}^{\bullet} \ , \tag{4.22}$$

Algorithm 6 : MPPCAtmpSm – PCA on incomplete data by temporal smoothing

```
Input: data matrix $\mathbf{X}$, weight matrix $\mathbf{W}$ with binary values, number of principal
    axes to be estimated $k$.
Output: mean vector $\boldsymbol{\mu}$, $\mathbf{U}$ spanning principal subspace.
 1: Estimate the weighted mean vector: $\mu_i = \frac{1}{N^\bullet}\sum_{j=1}^{N^\bullet} \mathbf{x}_{ij}^\bullet$ , $i = 1 \dots M$ .
 2: Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
 3: Set elements of $\mathbf{U} \in \mathbb{R}^{M\times k}$ to random values.
 4: repeat
 5:    E-step: $\mathbf{a}_j = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}_j^\bullet$, $j = 1 \dots N$ .
 6:    M-step: $\mathbf{A}' = [\mathbf{a}_2 \dots \mathbf{a}_N, \mathbf{a}_N] - 2\mathbf{A} + [\mathbf{a}_1, \mathbf{a}_1 \dots \mathbf{a}_{N-1}]$
              $\mathbf{u}_{i:} = [\hat{\mathbf{x}}_{i:}^\bullet \; \mathbf{0}_{1\times N^\circ}][\mathbf{A}^\bullet \; \alpha\mathbf{A}'^\circ]^\dagger$ , $i = 1 \dots M$ .
 7: until convergence.
 8: Orthogonalize $\mathbf{U}$.
 9: Project input data on $\mathbf{U}$: $\mathbf{a}_j = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}_j^\bullet$, $j = 1 \dots N$ .
10: Perform PCA on $\mathbf{A}$. Obtain $\mathbf{U}'$ .
11: Rotate $\mathbf{U}$ for $\mathbf{U}'$: $\mathbf{U} = \mathbf{U}\mathbf{U}'$.
```

or, expressing with the pseudoinverse:

$$\mathbf{a} = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^\bullet \; . \tag{4.23}$$

4.3.2 Iterative reconstruction of missing pixels

Alternatively, the principal axes can in M-step also be obtained by applying the standard PCA on all pixels providing that missing pixels are filled-in. The question is how to optimally fill-in the values of the missing pixels. Since not all the pixels of an image are known, some coordinates of the corresponding point in the image space are undefined. Thus, the position of the point is constrained to the subspace defined by the missing pixels. Given the principal subspace $\mathbf{U}$, which models the input data, the optimal location is a point in the missing pixels subspace which is closest to the principal subspace. This point is obtained by replacing the missing pixels with the reconstructed values, which are calculated by (2.21) using the coefficients $\mathbf{a}_j$ estimated in E-step of the current iteration and the principal axes $\mathbf{u}_j$ obtained in the previous iteration.

This principle is illustrated in a simplified 3-D example in Fig. 4.3. The green line $\mathbf{u}$ represents the 1-D principal subspace in the 3-D input space. The input image $\mathbf{x}$ has defined the first two coordinates only, thus x_3 is considered as missing pixel. Therefore, the proper location of $\mathbf{x}$ can be located anywhere on the black line. The goal is to optimally determine this location by finding a point in the missing pixel subspace (black line), which is closest to the principal subspace (green line). To achieve this, $\mathbf{x}$ is first projected into the subspace considering only the first two coordinates by using (4.22). The projection and the reconstructed point are depicted as the red point in Fig. 4.3. The third coordinate of this reconstructed point is the best guess for the proper value of the missing pixel. Thus this value is used as the value of the unknown coordinate of the input image $\mathbf{x}$ and the (most probably) proper location of $\mathbf{x}$ is determined (the blue point in Fig. 4.3).

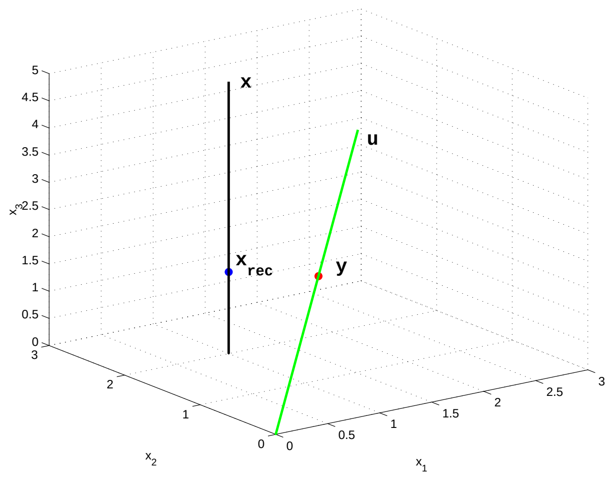


Figure 4.3: The principle of filling missing pixels.

Therefore, the new M-step looks as follows:

- **M-step**: Estimate $\mathbf{U}$ by applying the standard PCA on $\mathbf{X}$ with the reconstructed missing pixels:

$$x_{ij} = y_{ij} \ , \ j = 1 \ldots N \ , \ i \in \mathcal{I}_j^{\circ} \text{ where } \mathbf{Y} = \mathbf{U}\mathbf{A} + \boldsymbol{\mu}\mathbf{1}_{1\times N} \ . \tag{4.24}$$

One advantage of this approach is that it does not assume a smoothness prior, therefore it is appropriate for visual learning from unordered image sequence as well. Furthermore, this approach produces *real* principal axes. The columns of $\mathbf{U}$ do not just span the principal subspace, but they are mutually orthogonal and oriented in the directions of the principal axes as well (assuming that the missing values are estimated correctly). This means that the principal subspace of the dimension

$k-1$ can be obtained from the estimated k-dimensional principal subspace simply by discarding the least significant principal axis $\mathbf{u}_k$.

The convergence of the algorithm can be sped up by a more efficient initialization of the principal vectors $\mathbf{U}$. Instead of simply setting its elements to random values, one can estimate the initial values of $\mathbf{U}$ from an estimate of $\mathbf{A}$ obtained by performing SVD on the inner product matrix $\mathbf{C}'$, which is estimated from the non-missing pixels only:

$$c'_{ij} = \frac{M}{N|\mathcal{P}|} \sum_{p\in\mathcal{P}} \hat{x}_{pi}\hat{x}_{pj} \ , \ \mathcal{P} = \{p \mid p \in \mathcal{I}_i^\bullet \wedge p \in \mathcal{I}_j^\bullet\} \ , \ i,j = 1\ldots N. \tag{4.25}$$

The entire procedure is outlined in Algorithm 7.

Algorithm 7 : MPPCAitRec – PCA on incomplete data by iterative reconstruction

```
Input: data matrix $\mathbf{X}$, weight matrix $\mathbf{W}$ with binary values, number of principal
    axes to be estimated $k$.
Output: mean vector $\boldsymbol{\mu}$, principal vectors $\mathbf{U}$.
1: Estimate the weighted mean vector: $\mu_i = \frac{1}{N^\bullet}\sum_{j=1}^{N^\bullet} \mathbf{x}_{ij}^\bullet$ , $i = 1\ldots M$ .
2: Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
3: Estimate the inner product matrix $\mathbf{C}'$ from the known data:
   $c'_{ij} = \frac{M}{N|\mathcal{P}|}\sum_{p\in\mathcal{P}} \hat{x}_{pi}\hat{x}_{pj}$ , $\mathcal{P} = \{p \mid p \in \mathcal{I}_i^\bullet \wedge p \in \mathcal{I}_j^\bullet\}$ , $i = 1\ldots N$ , $j = 1\ldots N$.
4: Perform SVD on $\mathbf{C}'$ yielding eigenvectors as an estimate for $\mathbf{A}$.
5: Estimate an initial $\mathbf{U}$: $\mathbf{u}_{i:} = \hat{\mathbf{x}}_{i:}^\bullet \mathbf{A}^{\bullet\dagger}$ , $i = 1\ldots M$ .
6: repeat
7:    E-step: Estimate $\mathbf{A}$: $\mathbf{a}_j = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^\bullet$, $j = 1\ldots N$ .
8:    M-step: Estimate $\mathbf{U}$ by applying the standard PCA on $\mathbf{X}$ with reconstructed
      missing pixels: $x_{ij} = y_{ij}$ , $j = 1\ldots N$ , $i \in \mathcal{I}_j^\circ$ where $\mathbf{Y} = \mathbf{U}\mathbf{A} + \boldsymbol{\mu}\mathbf{1}_{1\times N}$ .
9: until convergence.
```

4.3.3 Simple example

We will illustrate the algorithm for subspace learning by iterative reconstruction of missing pixels on the following example. Let us consider a set of vectors which is formed of 10 shifted harmonic (sinus) functions. Next, we randomly remove 20% of the elements, as depicted in the top row of Fig. 4.4(a). Now, the goal is to find the optimal principal axes representing these vectors by applying the algorithm described in the previous subsection.

Fig. 4.4 illustrates the performance of the proposed method. First, the initial values of the principal axes are obtained from the covariance matrix estimated from the non-missing values of the input vectors. Then, based on the E-step, we calculate the coefficients and the reconstructed signals. The missing pixels are then replaced with their reconstructed values from the previous iteration. The initialization, the first, the third, and the tenth iteration of the algorithm are shown in subsequent rows of Fig. 4.4. Note how the functions representing estimated principal axes are getting smoother after every iteration. Finally, we obtain almost perfect reconstruction of the input vectors.

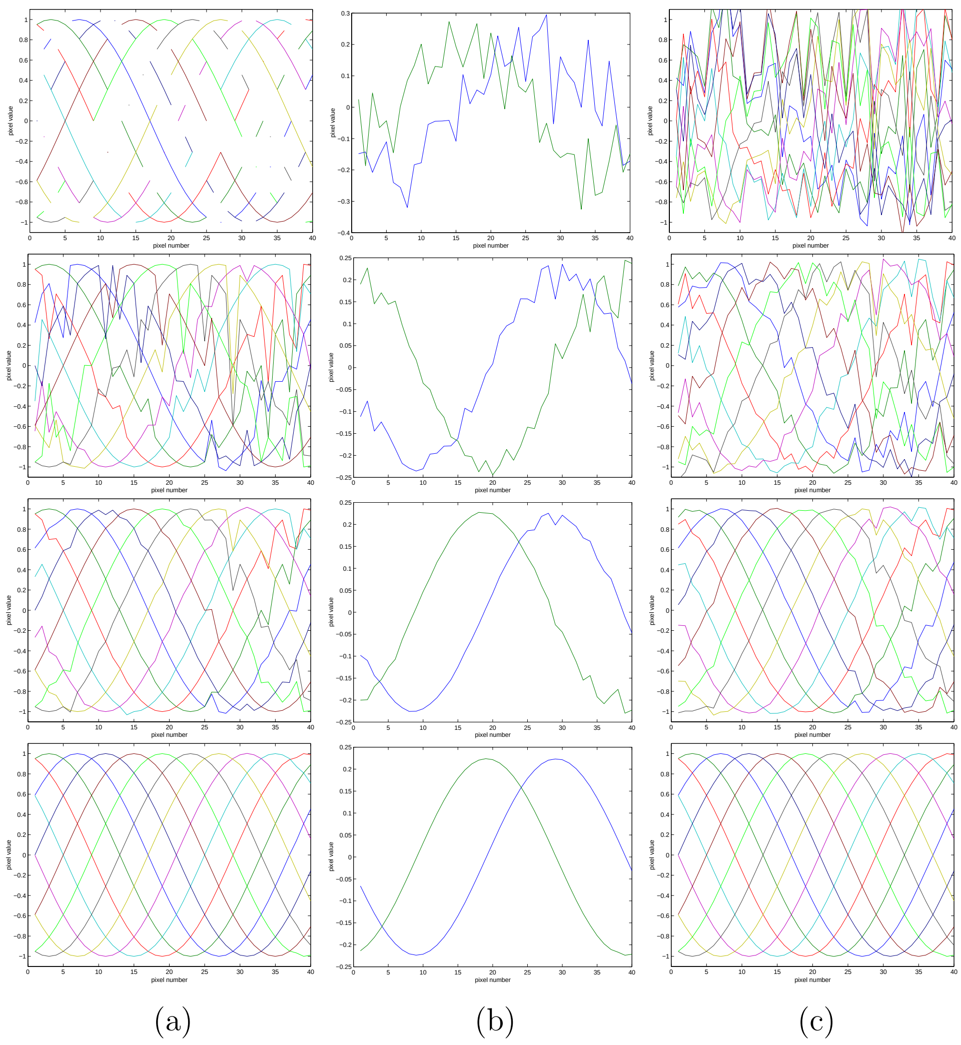


Figure 4.4: Simple example with missing pixels: (a) input data at each iteration, (b) estimated principal axes, (c) reconstructed vectors.

4.4 Incremental weighted PCA

In Section 3.2 we presented the algorithm for incremental learning, while in Section 4.2 we presented the algorithm for weighted learning. In this section we will combine both methods into a unified, weighted and incremental approach.

4.4.1 Algorithm

It is quite straightforward to incorporate the temporal weights into the incremental algorithm. The core of this algorithm is still the standard batch PCA on low-dimensional data (step 6 of the Algorithm 3). We can replace this standard PCA with the weighted algorithm, which considers temporal weights. This is feasible, because our incremental algorithm maintains low-dimensional coefficients of all input images throughout the process of the incremental learning (in contrast to other incremental approaches [32, 41]). Therefore, the representation of each image can be arbitrarily weighted at each update.

Incorporating spatial weights into the process of incremental learning is more complex. After the current eigenspace is updated with a new input image, this image is discarded. Therefore, in the later stages we cannot associate weights to individual pixels. This can be done only during the update. For that reason we define spatial weights in a different manner.

Let us assume, that the weights range from 0 to 1. If a weight is set to 1, it means that the corresponding pixel is fully reliable and should be used as is. If a weight is set to 0, it means that the value of the corresponding pixel is completely useless and it is not related to the correct value. We can recover an approximate value of this pixel by considering the knowledge acquired from the previous images. By setting the weight between 0 and 1, we can balance between the influence of the value yielded by the current model and the influence of the pixel value of the input image.

We can achieve this by adding a preprocessing step to the update algorithm. First we calculate the coefficients of the new image $\mathbf{x}$ by using the weighted method (4.13) considering larger influence of the pixels with large weights. By reconstructing the coefficients we obtain the reconstructed image $\mathbf{y}$ which contains pixel values yielded by the current model. By blending images $\mathbf{x}$ and $\mathbf{y}$ considering spatial weights by using the following equation

$$x_i^{new} = {}^{s}w_i x_i + (1 - {}^{s}w_i) y_i \ , \ \ i = 1 \ldots M \ , \tag{4.26}$$

we obtain the image which is then used for updating the current eigenspace. In this way, a selective influence of pixels is enabled also in the incremental framework.

The principle of blending is illustrated in Fig. 4.5. In this simple 2-D example the black dot represents a new point, which is to be used to update the current eigenspace, depicted as the blue line. The red dot represents a point reconstructed from the coefficient, which was estimated using the weighted method considering spatial weights. This reconstructed point is then blended with the input point into the point, which is afterwards used for updating the eigenspace (the green dot). The results of this process are presented for five different weight vectors. One can observe how the position of the blended point varies according to the weight vector.

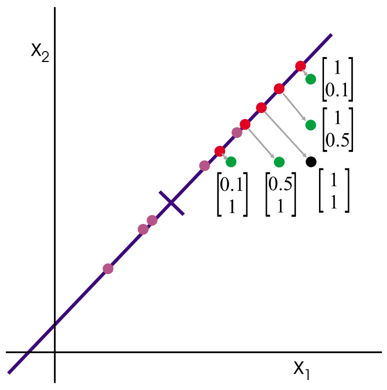

x_2
x_1
$\begin{bmatrix}1\\0.1\end{bmatrix}$
$\begin{bmatrix}1\\0.5\end{bmatrix}$
$\begin{bmatrix}0.1\\1\end{bmatrix}$
$\begin{bmatrix}0.5\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\end{bmatrix}$

Figure 4.5: Illustration of blending of input point (black) and reconstructed points (red) into blended points (green) for different spatial weights.

By augmenting these additional steps, the overall algorithm for weighted incremental learning is given in Algorithm 8.

A potential drawback of incremental methods is a propagation of errors, since images are added sequentially. In the case of the weighted incremental algorithm this could be problematic if in the early stages of the learning process many images contain significant number of pixels with small weights. In this case the algorithm fails to correctly recover the values of these pixels because the model is too flexible and it cannot predict the correct values. When the model encompasses a sufficient number of views it becomes more stable and this is no longer a problem.

Algorithm 8 : WIPCA – weighted incremental PCA

```
Input: current mean vector $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients
    $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$, spatial weights for new image ${}^{\mathbf{s}}\mathbf{w}$, temporal weights
    for all images ${}^{\mathbf{t}}\mathbf{w}^{(n)}$.
Output: new mean vector $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$,
    new eigenvalues $\boldsymbol{\lambda}^{(n+1)}$.
 1: Calculate the coefficients considering spatial weights:
    $\mathbf{a} = ((.\sqrt{{}^{\mathbf{s}}\mathbf{w}}\mathbf{1}_{1\times k}) \circ \mathbf{U}^{(n)})^{\dagger}(.\sqrt{{}^{\mathbf{s}}\mathbf{w}} \circ (\mathbf{x} - \boldsymbol{\mu}^{(n)}))$ .
 2: Reconstruct the image yielded by the current eigenspace: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 3: Blend the input and the reconstructed image considering the spatial weights ${}^{\mathbf{s}}\mathbf{w}$:
    $\mathbf{x} = {}^{\mathbf{s}}\mathbf{w} \circ \mathbf{x} + (\mathbf{1}_{M\times 1} - {}^{\mathbf{s}}\mathbf{w}) \circ \mathbf{y}$ .
 4: Project the obtained image $\mathbf{x}$ into the current eigenspace: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$ .
 5: Reconstruct the new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 6: Compute the residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
 7: Append $\mathbf{r}$ as a new basis vector: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{bmatrix}$ .
 8: Determine the coefficients in the new basis: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ .
 9: Perform TWPCA on $\mathbf{A}'$ considering the temporal weights ${}^{\mathbf{t}}\mathbf{w}^{(n)}$. Obtain the
    mean value $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$, and the eigenvalues $\boldsymbol{\lambda}''$.
10: Project the coefficient vectors to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$ .
11: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$ .
12: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ .
13: New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ .
```

4.4.2 Simple 2-D example

We will illustrate the behavior of the proposed algorithm on the 2-D example, which was already used for the demonstration of the non-weighted incremental PCA in Section 3.2.1 and the batch method for weighted PCA in Section 4.2.3. The goal is to estimate the 1-D principal subspace from 41 input 2-D points, which sequentially enter into the learning process. The eigenspace is being built incrementally considering temporal weights, which give larger influence to the recent points (the weights are set to ${}^{t}w_j = j^2$).

Fig. 4.6 depicts the evolution of the eigenspace. By comparing this figure with Fig. 3.2 it is evident how the points at the end of the point sequence have larger influence to the estimation of the principal axis. At the end of the sequence, the estimated eigenspace is very similar to the eigenspace obtained using the batch weighted method.

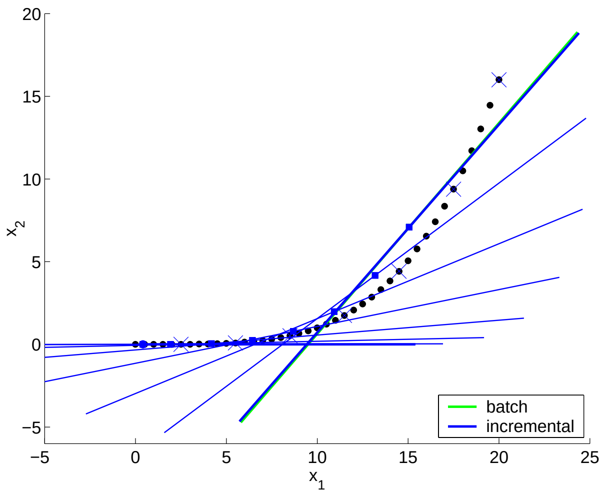


Figure 4.6: Illustration of weighted incremental learning.

The second example in Section 4.2.3, which considers spatial weights, cannot be handled using the proposed method for incremental weighted learning. In that 2-D example the weight of the second coordinate was 0.1 in all training points, meaning that the value of this coordinate is very unreliable and that in the blending step, the value estimated from the current model should have ten times larger influence. However, since all the training points have equal weights, this model can never be reliably estimated, therefore the values of the second coordinate predicted by the model are not accurate either. This example demonstrates that it is very important that at least in the early stages of the learning process, enough reliable information should be available.

4.4.3 Comparison with related work

In the literature, the most related work is the algorithm for the eigenspace updating with temporal weighting proposed by Liu and Chen [41]. We show in Appendix C.2 that their algorithm can be considered as a special case of ours when the temporal weights in our method form a geometric sequence. On the other hand, our method can handle arbitrary temporal weights. Moreover, our weighted method can handle temporal and spatial weights as well. And finally, our method estimates the coefficients and maintains their values throughout the learning process, while the method proposed by Liu and Chen does not.

4.5 Incremental PCA on incomplete data

The algorithm for incremental learning from the data that contains missing pixels is a special case of the algorithm for weighted incremental learning. In this case, only the spatial weights are considered, which can have just binary values distinguishing known and missing pixels.

4.5.1 Algorithm

The blending step in the algorithm for weighted incremental learning reduces into the imputation of missing pixels. Before the current eigenspace is updated with the new image, the missing pixels have to be optimally filled in.

The main idea how to fill in the missing pixels is similar as in the case of iterative batch algorithm described in Section 4.3.2. Using the current principal subspace $\mathbf{U}^{(n)}$, which models the input data seen so far, and a set of known pixels in the new image $\mathbf{x}$, we want to estimate the values of missing pixels in $\mathbf{x}$. Thus, we want to locate the point representing the image $\mathbf{x}$ in the subspace defined with the values of the known pixels. The optimal location is a point in the missing pixels subspace which is closest to the current principal subspace (see Fig. 4.3). This point is obtained by imputing missing pixels with the reconstructed values, which are calculated from the coefficients estimated from the known pixels only. Since these coefficients reflect the novel information in the new image contained in the known pixels, we hope that the prediction in the missing pixels will be fine as well. Such an improved image is the best approximation of the correct image that we can obtain from the information contained in the known pixels and in the current eigenspace.

Thus, the new image $\mathbf{x}$ is first projected into the current principal subspace $\mathbf{U}^{(n)}$ by solving the system of linear equations arising from non-missing pixels (4.22). The obtained coefficient vector $\mathbf{a}$ is then reconstructed and the values in the reconstructed image are used for imputation of missing pixels. The improved image is then used for updating the current eigenspace. These steps are more formally presented in Algorithm 9.

Algorithm 9 : MPIPCA – incremental PCA on incomplete data

```
Input: current mean vector $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients
    $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$, binary spatial weights for new image ${}^{\mathbf{s}}\mathbf{w}$.
Output: new mean vector $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$,
    new eigenvalues $\boldsymbol{\lambda}^{(n+1)}$.
 1: Impute the missing pixels in the new image $\mathbf{x}$: $\hat{\mathbf{x}}^{\circ} = \mathbf{U}^{(n)\circ}\mathbf{U}^{(n)\bullet\dagger}\hat{\mathbf{x}}^{\bullet}$ .
 2: Project the obtained image $\mathbf{x}$ into the current eigenspace: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$ .
 3: Reconstruct the new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 4: Compute the residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
 5: Append $\mathbf{r}$ as a new basis vector: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{bmatrix}$ .
 6: Determine the coefficients in the new basis: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ .
 7: Perform PCA on $\mathbf{A}'$. Obtain the mean value $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$, and the
    eigenvalues $\boldsymbol{\lambda}''$.
 8: Project the coefficient vectors to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$ .
 9: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$ .
10: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ .
11: New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ .
```

4.5.2 Simple 2-D example

We will illustrate the behavior of the proposed algorithm on our 2-D example. In this case the second coordinate of five points is unknown. Therefore these points could lie anywhere in the dashed lines depicted in Fig. 4.7.

The proposed method sequentially estimates the unknown coordinate values and defines the position of the points on the dashed lines (shown as circles). The eigenspace is being updated using these improved points. Since these points are a good approximation of the correct ones, the evolution of the eigenspace is very similar to the evolution of the eigenspace, which was built from the ideal data (see Fig. 3.2) and the final principal axis is very close to the optimal one.

4.5.3 Comparison with related work

Recently, Brand presented an algorithm for incremental singular value decomposition of data containing missing pixels [14]. Thus, his algorithm uses SVD for learning, while our algorithm uses PCA. This means that in the framework of eigenspace

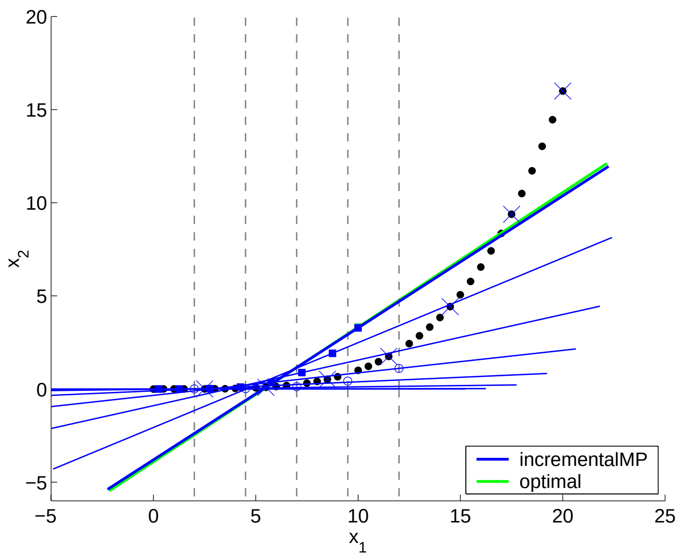


Figure 4.7: Illustration of incremental learning from incomplete data.

learning, Brand's algorithm assumes that the mean is always 0. In contrast, our algorithm considers also the mean and updates its value at each step adequately.

What is common to both algorithms is the way how they treat missing pixels. In both cases, before the current subspace is updated with a new image, missing pixels are imputed using known pixel values and the current subspace. As we show in Appendix C.3, the imputation rule in our algorithm

$$\hat{\mathbf{x}}^{\circ} = \mathbf{U}^{\circ}\mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^{\bullet} \tag{4.27}$$

is equivalent to the imputation rule in the Brand's [14] method.

As shown in [14], the imputation of missing pixels using (4.27) minimizes the distance of the vector representing a new image to the current subspace and maximizes the concentration of the variance in the top singular values. Consequently, such imputation rule minimizes the rank of the updated SVD guaranteeing parsimonious model of the data.

4.6 Experimental results

In this chapter we presented several algorithms for weighted subspace learning. We performed a number of experiments in order to evaluate the proposed methods. In this section we will present some of the results.

4.6.1 Batch weighted method

First we will demonstrate the performance of the *batch* method for *weighted learning* using *temporal weights*. Imagine that we want to build a subspace representation of the 'duck' object from the COIL20 database. If we assume that it is more likely to see the duck from the front than from the back it is reasonable to model the front views more accurately [60]. Therefore, we put larger weights on the images of the front (and side) views of the object and built the eigenspace using the batch weighted method. Fig. 4.8 shows the images reconstructed with the standard method and with the weighted method. One can observe that the first three images are reconstructed worse with the weighted method, since they represent the rear views of the duck where the weights were small. In contrast, the front (and side) views of the duck are reconstructed considerably better than with the standard approach. Fig. 4.9 confirms this finding. It shows the reconstruction errors in the images for both methods. To increase clarity, the reconstruction error was smoothed over the consecutive images reducing local deviations. One can observe that in the second half of the image sequence the reconstruction error produced by the proposed weighted method is smaller. Consequently, the total weighted reconstruction error is smaller as well, as presented in Table 4.1. This is exactly what we wanted to achieve.

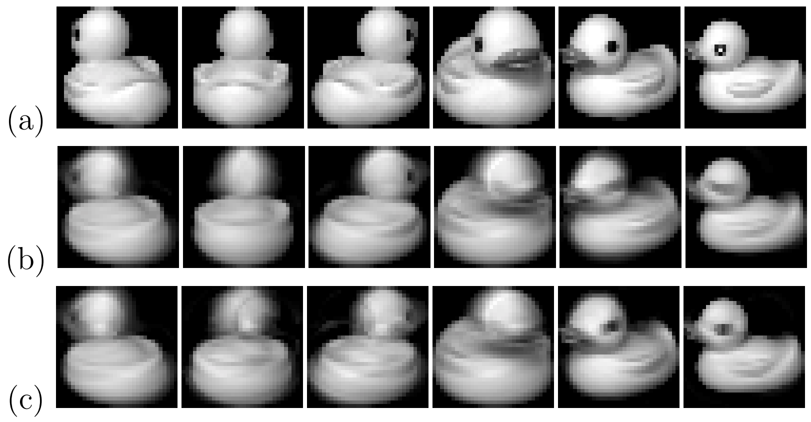


Figure 4.8: (a) Six images from training set. (b) Reconstruction using standard PCA. (c) Reconstruction using weighted PCA.

Table 4.1: Weighted reconstruction errors of standard and weighted method.

standard PCA	weighted PCA
660	605

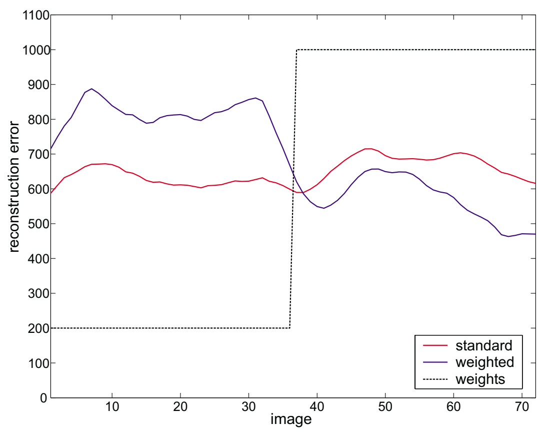


Figure 4.9: Reconstruction errors of standard and weighted method.

The performance of the weighted method for estimating coefficients using *spatial weights* is presented in Fig. 4.10. In this experiment we modeled three objects from the COIL20 database. The training set consisted of 24 images of each of the three objects – altogether 72 images of size 32 × 32 pixels. Some of the images are shown in Fig. 4.10(a). Later, we wanted to recognize the object depicted in Fig. 4.10(b). The left half of this test image represents the 'duck' object, while the right half represents the 'cat' object. If the standard recognition was used, the method could not choose between the 'duck' and the 'cat' and the reconstruction was indistinguishable (Fig. 4.10(c)). When the weighted method was used and one half of the image was considered more important (the left part in Fig. 4.10(d) and the right part in Fig. 4.10(e)), the method reconstructed the 'duck' and the 'cat', respectively, very well. Here, we assumed that the information about the weights was signaled by some higher-level cognitive process.

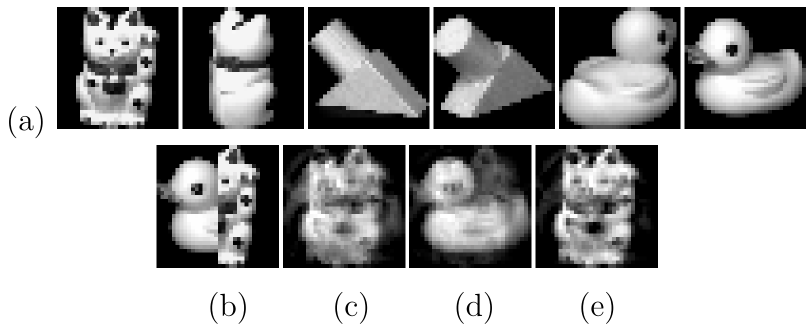


Figure 4.10: (a) Six images from training set, (b) test image, (c) standard, and (d,e) weighted reconstructions.

4.6.2 Incremental weighted method

We evaluated the *incremental and temporally weighted* method on the set of training images, which was already used for the evaluation of the non-weighted incremental method. The training set consisted of 720 images of all twenty objects from the COIL20 database. The objects are depicted in Fig. 3.3.

On each image we put a weight, which was proportional to the second power of the image index, giving more influence to the objects and the images at the end of the image sequence. The results are depicted in Fig. 4.11. The reconstruction errors of the incremental weighted method (*WincA*, *WincX*) do not differ significantly from the results of the batch weighted method (*Wbatch*). And certainly, the results of the batch and the incremental weighted methods are better than the results of the standard methods for images with larger weights. This is also reflected in better weighted squared reconstruction errors as presented in Table 4.2.

Table 4.2: Weighted reconstruction errors of batch and incremental methods.

batch	*incA*	*incX*	*Wbatch*	*WincA*	*WincX*
617	658	648	554	583	565

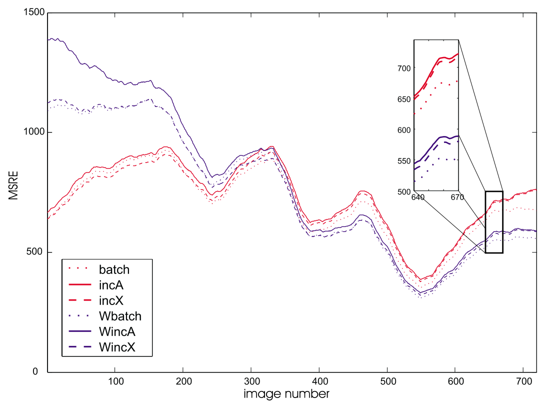


Figure 4.11: Reconstruction errors of batch and incremental standard and weighted methods.

Next, we present the results of the proposed *incremental method for learning from incomplete data* to improve the results of the *visually-based localization* of a mobile robot [4]. In the learning stage, the representation of the environment (our lab in this case) is built from panoramic images taken from several locations. We can simulate the in-plane robot rotation by shifting cylindrical panoramic images and generating spinning images [38]. Three such views of two locations are depicted in Figs. 4.12(a,b). We thus obtain all necessary views of the environment, which are used for building the representation using PCA. Later, in the localization stage, a novel image is taken and projected into the eigenspace. The location of the robot is determined by searching for the closest projected training image.

However, due to the construction of the panoramic sensor, not only the environment is captured in the image, but also the holder of the panoramic mirror (the dark vertical bar in Figs. 4.12(a,b)) and the surface of the robot (lower part of the images). If the robot is oriented differently in the localization stage, the holder appears in a different position in the image, which makes the test image less similar to the correct training image and the localization can fail.

The proposed method offers a solution to this problem. Since we know, that the holder is not a part of the environment, we can mask it out during the learning, and learn only the parts, which belong to the environment. We can achieve this by using the incremental method for learning from incomplete data considering undesirable parts as missing pixels (see Figs. 4.12(c)).

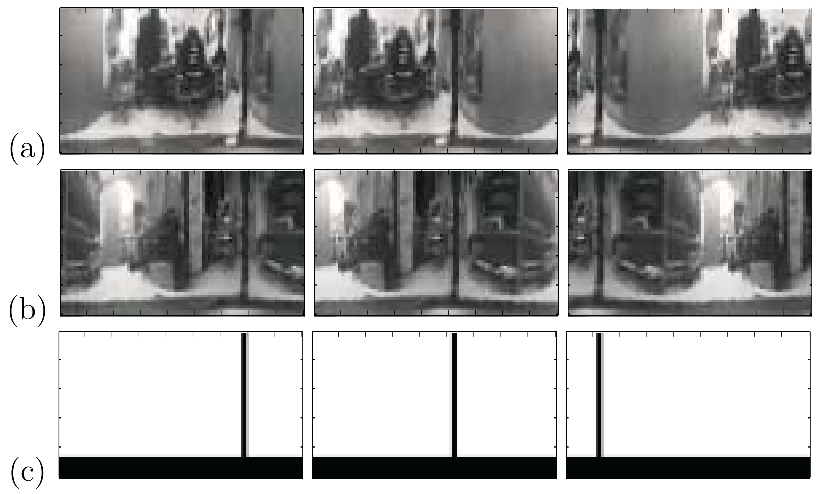


Figure 4.12: (a,b) Spinning images from two locations. (c) Weights.

In the learning stage, the robot was moving from one part of the lab to the other. In the localization stage, the robot returned to the starting position following approximately the same path in the opposite direction. The results are presented in Fig. 4.13. The gray levels represent coefficient errors; i.e., the distances between the projections of the test images (given in the x axis) and the projections of the training image (y axis). Since the path of the robot was approximately the same as in the learning stage, we expect that the coefficient error would be minimal on the diagonal of the error matrix. Since the standard approach incorporated in the representation also the vertical holder, which was in a different image position in the localization stage, the results of the standard method are not very good (the diagonal of the error matrix in Fig. 4.13(a) is very indistinct). In contrast, the proposed method did not incorporate the holder into the representation of the environment. Consequently, the values around the diagonal in Fig. 4.13(b) are significantly smaller, which makes the localization much more accurate and reliable.

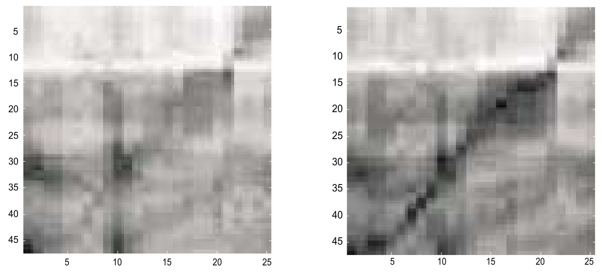


Figure 4.13: Coefficient errors using (a) standard and (b) proposed method.

4.6.3 Learning from incomplete data

In this subsection we will present the results of the proposed batch methods for *learning from incomplete data*. The goal was to learn a 9-D subspace representation of the 'duck' object from 72 images from the COIL20 database (see Fig. 4.14(a)). From each image we "erased" a square, which covered 20% of the image (Fig. 4.14(b)) and considered erased pixels as missing pixels.

First we applied the standard PCA algorithm on the training images with missing pixels filled-in using the simple mean substitution (in figures referred to as *MS*). The missing pixels were replaced with the mean values calculated from non-missing pixels only. Then we applied the proposed algorithms: Algorithm 7, which iteratively reconstructs missing pixels (*itRec*), the modified EM algorithm which calculates the principal subspace from non-missing pixels only (*EM*), and Algorithm 6, which assumes the smoothness prior. The latter algorithm was tested in the circumstances when the smoothness prior hold (the training images were aligned in the sequential order – *EMts*) and when this assumption was disadvantageous (the training images were aligned in a random order – *rndEMts*).

Figs. 4.14(c–g) depict some training images with the missing pixels filled-in with the reconstructed values, which were obtained using the methods described above. One can observe that both proposed algorithms (*itRec* and *EMts*) reconstructed missing squares significantly better than the standard PCA on the training images with mean-substituted missing pixels (*MP*). The modified EM algorithm without smoothness prior (*EM*) mainly performed well too, however in some pixels it failed to correctly reconstruct the pixel values to a large extent. As expected, when the training images were randomly ordered, the incorrectly assumed smoothness prior significantly degraded the results (*rndEMts*).

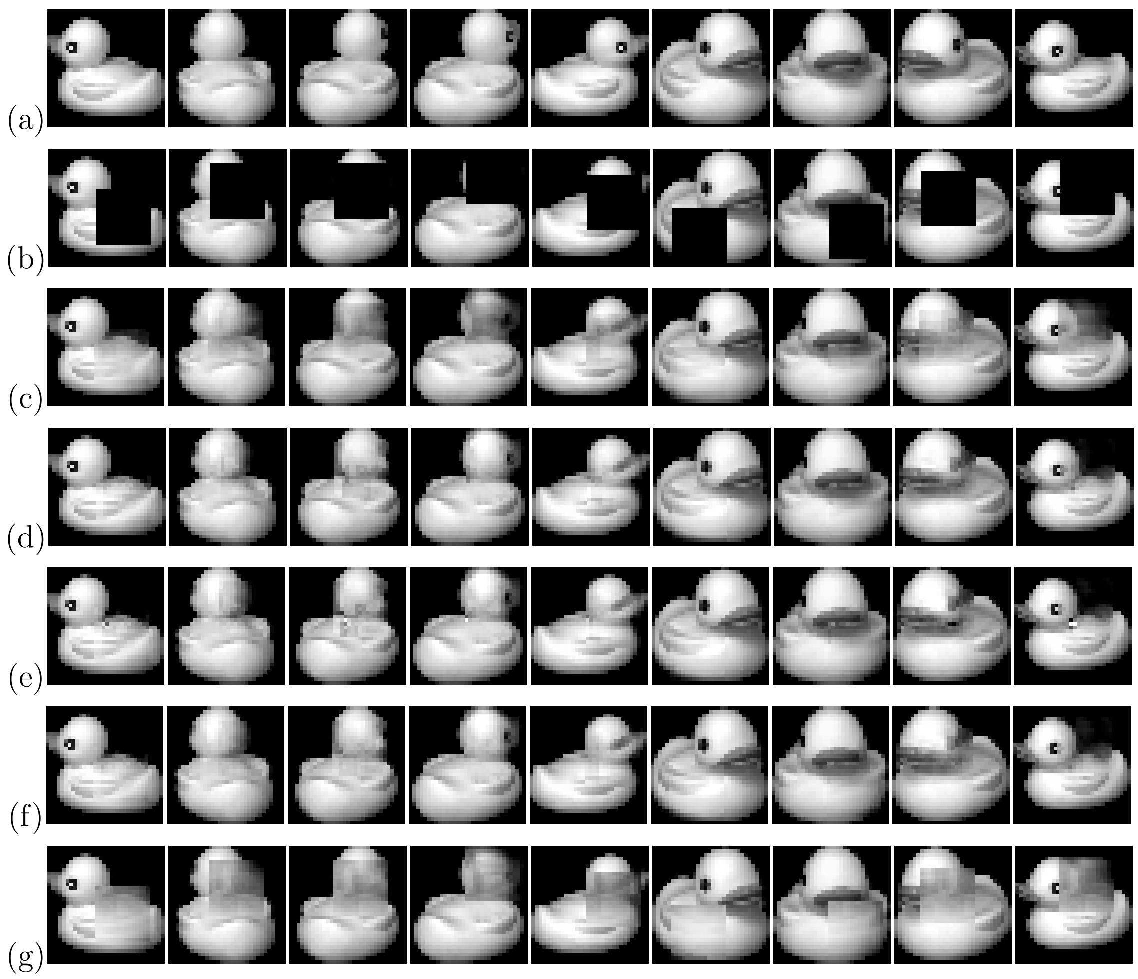


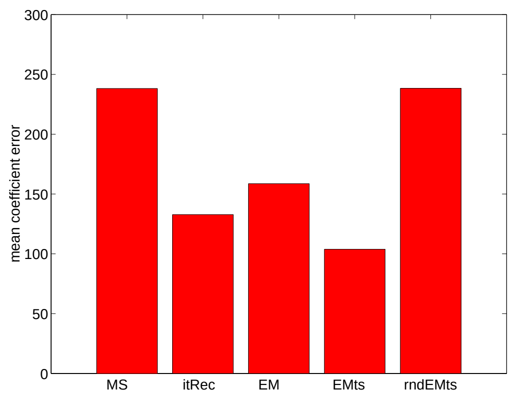
Figure 4.14: (a) Nine complete images, (b) training images with missing squares, reconstructed missing pixels using (c) *MS*, (d) *itRec*, (e) *EM*, (f) *EMts* and (g) *rndEMts* approaches.

The same conclusions can be drawn also from Fig. 4.15, which presents more quantitative results. For comparison, we projected original images without missing pixels into the obtained eigenspaces. Ideally, we would obtain exactly the same coefficients as by projecting the training images considering the non-missing pixels only. Fig. 4.15(a) compares the obtained mean coefficient error, i.e., the mean distance between the corresponding coefficient vectors, while Fig. 4.15(b) plots the values of all 72 distances in increasing order. Fig. 4.15(c) plots the recognition rate, i.e. how many times the projected complete images were the closest to the corresponding projected non-complete training images. Fig. 4.15(d) indicates the similarity between the estimated principal axes and optimal ones, which were obtained by performing PCA on the complete training images without missing pixels. For each of the first nine principal axes, the dot product between the ideal and the estimated principal axis is depicted. All these figures clearly indicate that the proposed methods for learning from incomplete data perform well and considerably improve the results of the simple mean-substitution approach.

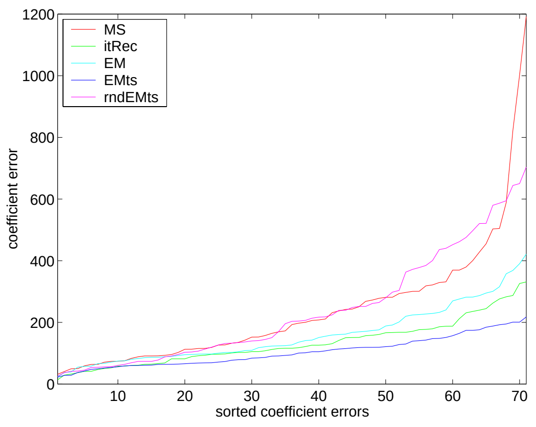
We also tested the performance of the algorithms with regard to the amount of missing pixels present in training images. We erased from 10% to 90% of randomly selected spatially incoherent pixels in each image. All the algorithms mentioned above were applied at each level of missing pixels.

One such trial at 50% of missing pixels is shown in Fig. 4.16. One complete and one training image are shown as well as the same training image with reconstructed missing pixels. Again, it is evident that *itRec*, *EM*, and *EMts* approaches reconstruct missing pixels significantly better than *MS* and *rndEMts*.

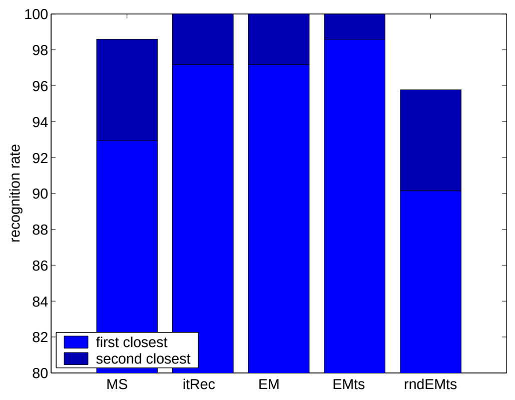
Fig. 4.17 summarizes the results obtained at all levels of missing pixels. For each level of missing pixels it plots the mean coefficient error, the mean squared reconstruction error in missing pixels, the recognition rate and the mean dot product between the corresponding estimated and optimal principal axes. All the plots show that *itRec*, *EM*, and *EMts* perform very good when the amount of missing pixels is 50% or less. Note, that in this case the missing pixels are spatially incoherent, which alleviates the problem to some extent. It can be observed that the unconstrained EM-based approach *EM* breaks down at 70% of missing pixels. At this level, some of the missing pixels are reconstructed very incorrectly, since no constraints are given. In contrary, if the reconstructed values are constrained with the smoothness prior (*EMts*), the best results are achieved. Of course, if this smoothness prior does not hold true, this algorithm does not perform well (*rndEMts*). In such cases, the best



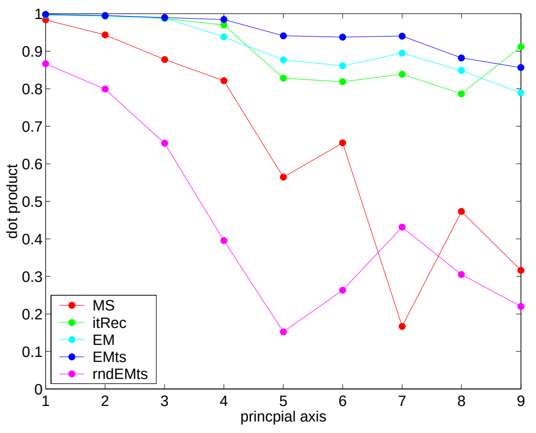
(a)



(b)

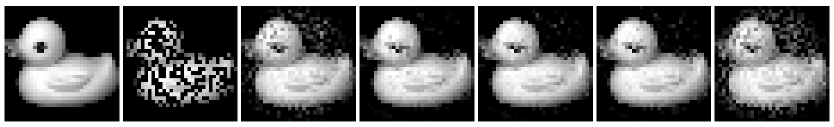


(c)



(d)

Figure 4.15: Performance of learning algorithms: (a) mean coefficient error, (b) sorted coefficient errors, (c) recognition rate, (d) dot products between corresponding estimated and optimal principal axes.



(a) (b) (c) (d) (e) (f) (g)

Figure 4.16: (a) One complete image, (b) training image containing 50% of missing pixels, reconstructed missing pixels using (c) *MS*, (d) *itRec*, (e) *EM*, (f) *EMts* and (g) *rndEMts* approaches.

choice is iterative reconstruction of missing pixels (*itRec*).

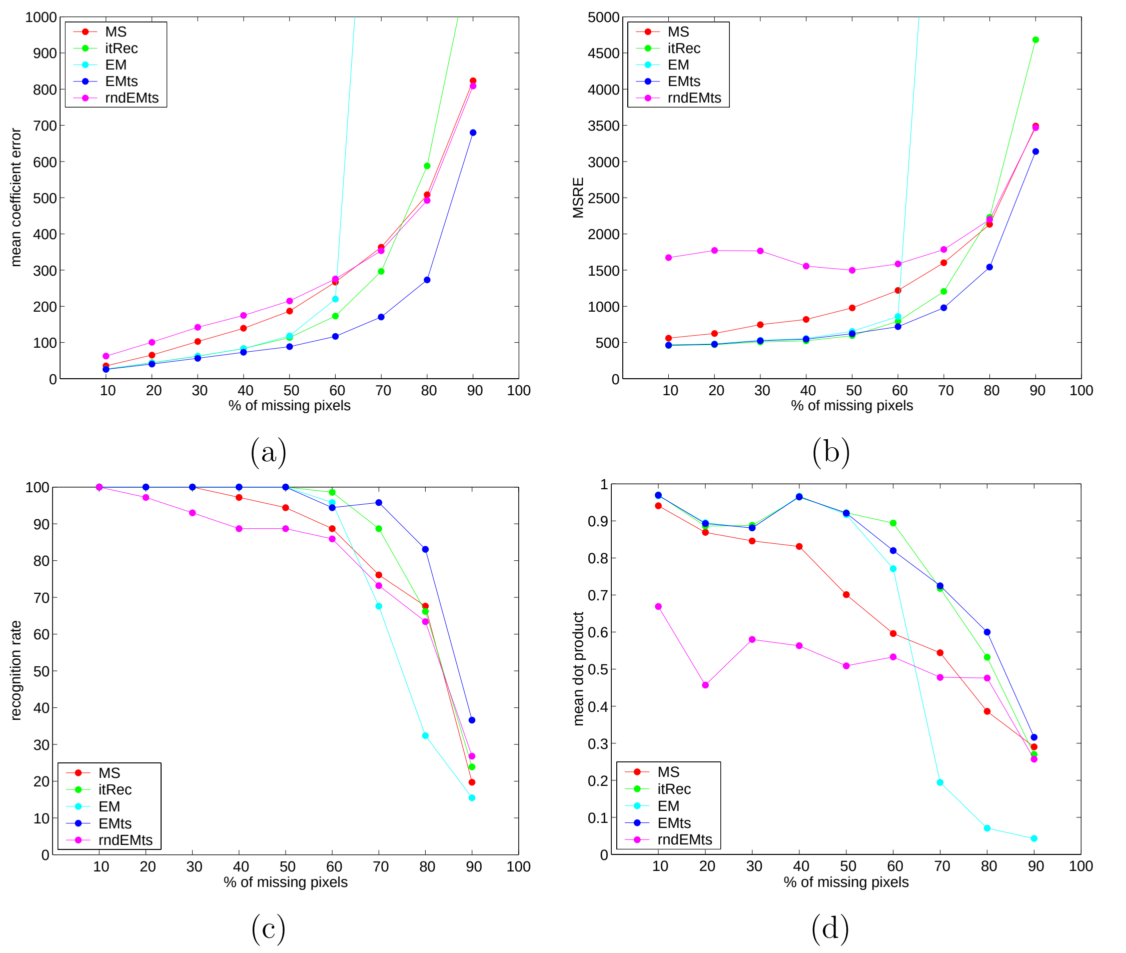


Figure 4.17: Performance of learning algorithms with regard to amount of missing pixels present in training images: (a) mean coefficient error, (b) mean squared reconstruction error in missing pixels, (c) recognition rate, (d) mean dot product between corresponding estimated and optimal principal axes.

4.6.4 Recognition from incomplete images

We applied the method for *recognition from incomplete data* on a rather specific domain of range images. Due to imperfections of range sensors, acquired images usually contain missing pixels; the pixels, which belong to the object, however their depth could not be obtained [71, 69, 2]. This is evident in Fig. 4.18 where an intensity image and the corresponding range image containing missing pixels are shown.

In the following experiments we will assume that the subspace model has already been correctly built (possibly from complete training images). For this particular problem of recognition of range images, we slightly extended the method for recog-

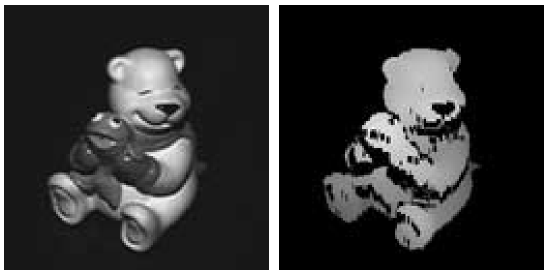


Figure 4.18: (a) Intensity image. (b) Range image containing missing pixels.

nition from incomplete data. The subspace coefficients were calculated from the known pixels only; missing pixels were excluded from the calculation. Since the depth was not defined in the background pixels either, they were excluded from the computation as well. However, the background pixels define the contour of the object, which proves to be very useful for recognition. Because of this, we performed the second step to refine the coefficients a_i. We reconstructed the coefficients and obtained the reconstructed image $\mathbf{y}$. Then we selected all pixels from $\mathbf{x}$ where the reconstruction error $\|y_i - x_i\|^2$ was consistent with the distribution of the reconstruction errors in non-missing pixels. Afterwards we marked the newly selected pixels as non-missing and recalculated the coefficient vector $\mathbf{a}$. These two steps were iterated until the convergence was reached, which happened after only a few iterations. In this way, many background pixels were gradually included into the computation, which significantly improved the results of the recognition.

We performed the experiments on two sets of images created from six 3-D models. The models (depicted in Fig. 4.19) were created from real objects by NRC-CNRC Institute for Information Technology[3].

The range images in the first set were generated by rotating the model around one axis in $1°$ steps (one degree of freedom). From each model 360 range images of the size 64×64 pixels were generated. Twelve range images generated from the first model are shown in Fig. 4.20.

The range images in the second set were generated by rotating a model around two axes (two degrees of freedom). From each model 545 range images of the size 64×64 pixels were created. The range images were generated from the views which were equally spaced in the object's pose space (see Fig. 4.21). Twelve range images

[3]www.vit.iit.nrc.ca/3D/Pages_HTML/3D_Models.html

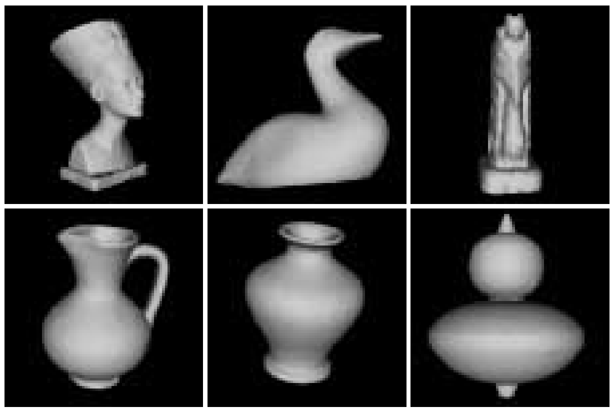


Figure 4.19: 3-D models used to create range images.

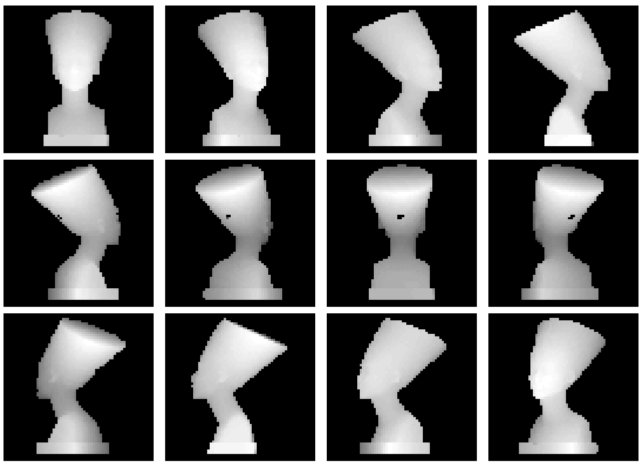


Figure 4.20: Twelve range images from 1 DOF example.

generated from the first model are shown in Fig. 4.22.

The background pixels and the missing pixels are usually set to 0. We rather set this level to the mean value of all non-background and non-missing pixels in all images. Since the difference between missing pixels and pixels in the object is smaller, the convergence of the reconstruction algorithm is much faster. Due to the same reason, the recognition results of the standard eigenspace method have also been improved.

We simulated missing pixels by setting a number of pixel values to the level of missing pixels. These pixels were chosen in two ways: *spatially incoherent* – by randomly choosing pixels, and *spatially coherent* – by choosing the pixels concentrated in two image areas. Both types of simulated missing pixels are depicted in Fig. 4.23(a).

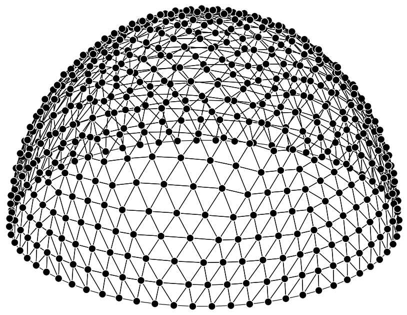


Figure 4.21: Viewpoints for generating range images in 2 DOF experiment.

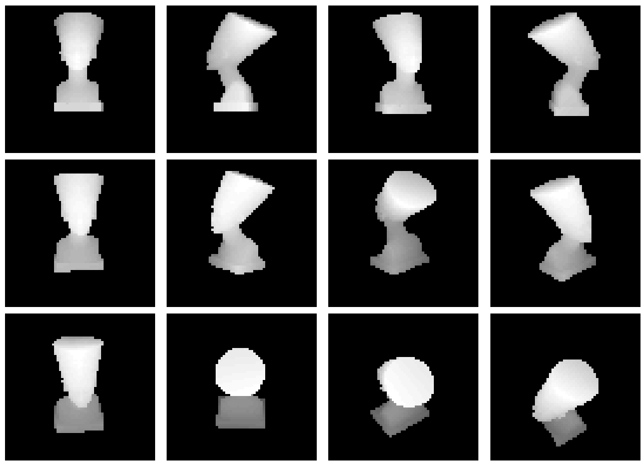


Figure 4.22: Twelve range images from 2 DOF experiment.

We first present the results of 1 DOF experiment. A universal all-encompassing eigenspace was built from 360 range images (60 images of each object). In addition, for each object an object eigenspace was created from 72 range images. All the images that were used were uniformly distributed in the object's pose space. In all tests we used eigenspaces of the dimension 15.

First we tested the recognition of the objects. A hundred range images were randomly chosen among all generated range images of all objects and different amounts of missing pixels were added (0–80%). Figs. 4.23(b,c) show the reconstruction of one such trial. The summary of the recognition rate is plotted in Figs. 4.24(a,c). It is evident that the proposed method outperforms the standard one. The recognition rate of the proposed method is close to 100% even at 80% of missing pixels.

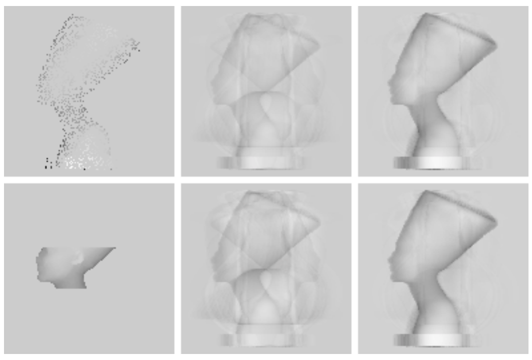
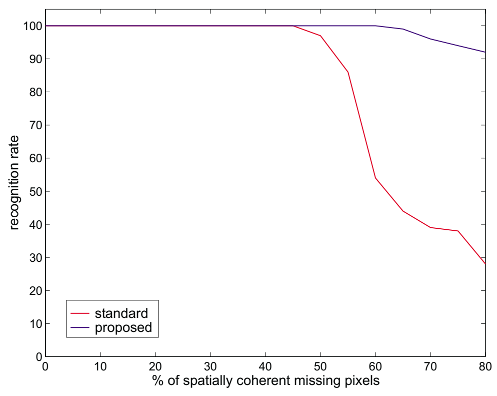


Figure 4.23: (a) Input images with spatially incoherent (top row) and spatially coherent (bottom row) missing pixels, images reconstructed with (b) standard method, and (c) proposed method.

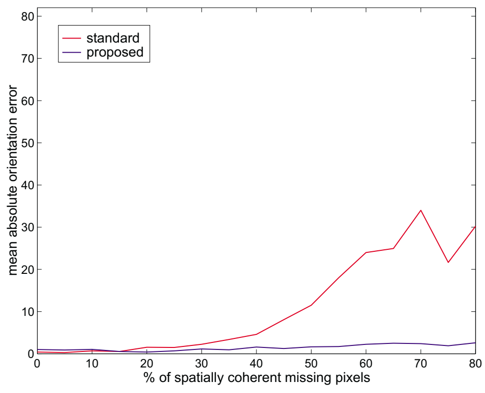
We obtained similar results also in the case of determining the orientation of the objects. Figs. 4.24(b,d) plot the results of determining the orientation of the first object in Fig. 4.19. A hundred range images were randomly chosen among all generated range images of the object. The orientation was determined by finding the closest image in the training set (the closest point in the eigenspace). The error was defined as the difference between the orientation of the detected training image and the orientation of the training image that should be detected. For each level of missing pixels the average absolute error was calculated. Using the proposed method, this error is very close to zero even at 80% of missing pixels, while the standard approach is not useful when a range image contains more than 40% of missing pixels.

Finally, we present the results of 2 DOF experiment. A universal eigenspace was built from 654 range images (109 images for each object). Each object eigenspace was created from 109 range images, which were uniformly distributed in the object's pose space. We performed only tests for spatially incoherent missing pixels. Here, the measure for the orientation error was the angle between the detected orientation and the orientation of the object in the test image. As depicted in Fig. 4.24(e,f), the recognition rate and the orientation error are almost constant for all amounts of missing pixels from 0–80%.

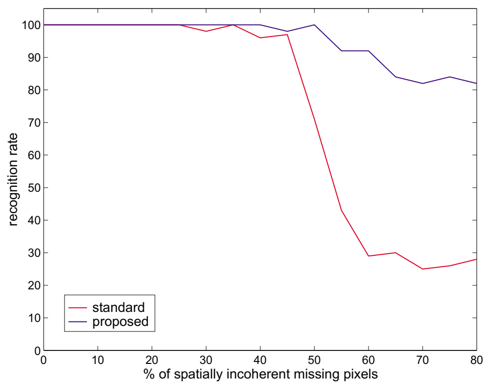
All the results clearly show that providing that a good model is given, the recognition can be very efficient even when only a small portion of pixels is known.



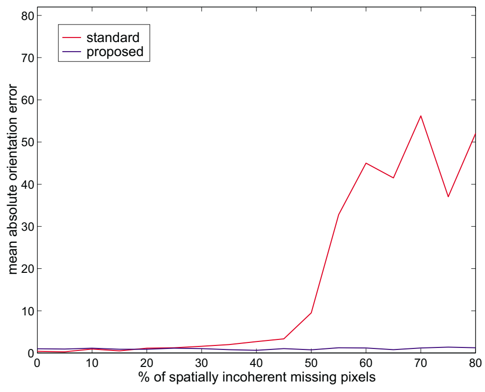
(a)



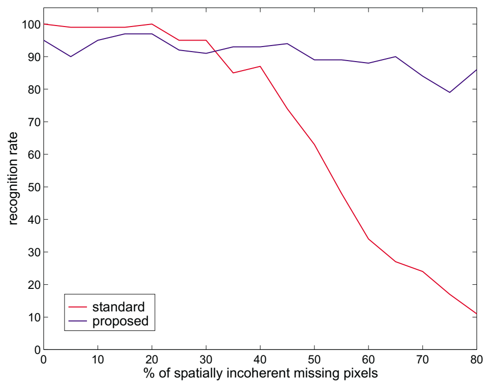
(b)



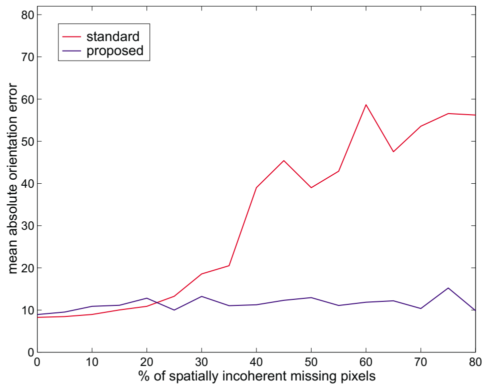
(c)



(d)



(e)



(f)

Figure 4.24: Comparison of standard method and method for recognition from incomplete images. (a–d) 1 DOF experiment, (e,f) 2 DOF experiment. (a,b) spatially coherent missing pixels, (c–f) spatially incoherent missing pixels. (a,c,e) recognition rate, (b,d,f) mean absolute orientation error depending on the percentage of missing pixels.

4.7 Chapter summary

In this chapter we presented several algorithms for weighted learning of eigenspaces and weighted recognition. By modifying the standard algorithm for PCA we derived the batch algorithm, which considers temporal weights. By modifying EM algorithm we derived the algorithm, which considers spatial weights as well. We also adapted the EM algorithm to cope with missing data. Then the incremental algorithm was adapted for weighted learning. We presented the algorithm for incremental weighted learning considering temporal and spatial weights, which was afterwards used for incremental learning from incomplete data. In the next chapter we will show, how these algorithms can be utilized for the robust subspace learning.

Chapter 5

Robust PCA

5.1 Chapter overview

In the previous chapter we presented several algorithms for weighted learning and learning from incomplete data. These algorithms assume that the weights are known in advance. However, in the real world environment, this is often not the case. Images may contain various outliers (occlusions, reflections, etc.) whose exact position in an image is not known. Since the standard PCA is intrinsically sensitive to non-gaussian noise such intrusions may considerably degrade the results of the visual learning and recognition.

To alleviate this problem several methods for *robust recognition* have been proposed. They introduce the robustness in the recognition stage and they assume that the images in the learning stage were ideal and that the visual model is correct. However, if the training images are taken under non-ideal conditions, the obtained representations encompass various non-desirable effects, which cannot be overcome at the recognition stage. This clearly indicates that we need a method for *robust learning* in order to obtain parametric representations insensitive to these effects. More specifically, we need a procedure, which is able to detect inconsistencies in the input data, eliminate them, and then calculate the representation from the consistent data only.

In this chapter we will present robust algorithms for eigenspace learning, which are able to detect outliers in training images. These outliers are then treated as missing pixels and the principal subspace is estimated from inliers only, using one of the algorithms for learning from incomplete data presented in the previous chapter. First, we will present two versions of a batch method for robust learning. Since the initial step of this method is still non-robust, a robust initialization of this algorithm based on a subsampling approach is then discussed. Then we will present the robust incremental algorithm. All presented algorithms will be finally evaluated in a number of experiments.

5.2 Robust batch method

First we will present a one-stage batch method for robust learning of eigenspace representations. It tries to detect outliers in the training images and estimates the eigenspace from inliers only. Next, the algorithm is further improved by employing a two-stage procedure, where first the outlying images are detected, which enables more reliable detection of the outlying pixels in the second stage.

5.2.1 One-stage method

The basic idea of our robust PCA algorithm is to determine the outliers and treat them as missing pixels. The principal subspace is then obtained with one of the algorithms described in the previous chapter. Based on the estimated principal axes and coefficients we can again determine the outliers and repeat the process. The crucial question is how to determine the outliers.

If we have some additional knowledge about the object or scene, which is being modeled, or an algorithm for outlier detection tailored for that particular problem, we can take advantage of it and use such algorithm for the detection of outliers. However, if we have no additional knowledge we have to rely solely on the information contained in the training images. We can detect outliers by checking the consistency over the entire image sequence. The pixels, whose reconstruction error significantly deviates from the distribution of reconstruction errors of all pixels, are treated as outliers. In other words, the outliers are pixels with large reconstruction error.

A question is how to set the threshold for determining outliers. We have investigated two approaches.

The first approach gives a global threshold. Depending on the number of principal components we choose to represent the images with, we expect that an average error per image is equal to the discarded variance, which can be computed as $\sigma_{img}^2 = \sum_{i=k+1}^{N} \lambda_i$. When we divide this number by the number of pixels, we get an average expected error per pixel $\sigma^2_{pix} = \frac{1}{M}\sum_{i=k+1}^{N}\lambda_i$. We treat all those pixels as outliers whose reconstruction error is larger than σ^2_{pix} multiplied by a factor, which depends on the expected quantity of outliers in the images. This approach is suitable for the images with similar visual variability across the whole image.

The second approach estimates a local threshold. De la Torre and Black [23] have proposed a method for computing a local threshold for each pixel based on the Median Absolute Deviation (MAD) of a pixel. They compute for each pixel p: $\sigma_p = \beta(1.4826\text{med}_R(|\mathbf{r}^p - \text{med}_R(|\mathbf{r}^p|)|), \sigma_{\min})$ where med_R is median taken over a region R, around pixel p, $\sigma_{\min}$ is the MAD over the whole image, and β is a constant factor. The initial error $\mathbf{r}^p$ is calculated using the standard PCA on the whole image set. This method has the advantage that it takes into account the variability within an image.

Based on this, our robust PCA approach is outlined in Algorithm 10.

Algorithm 10 : RPCA – robust PCA

```
Input: data matrix $\mathbf{X}$, number of principal axes to be estimated $k$.
Output: mean vector $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$, coefficients $\mathbf{A}$.
1: repeat
2:    Perform the standard PCA on $\mathbf{X}$ and obtain $\boldsymbol{\mu}'$, $\mathbf{U}' \in \mathbb{R}^{M\times k'}$ and $\mathbf{A}' \in \mathbb{R}^{k'\times N}$.
3:    Reconstruct the training images using $\boldsymbol{\mu}'$, $\mathbf{U}'$ and $\mathbf{A}'$ and calculate the reconstruction error.
4:    Detect outliers using either global or local threshold.
5:    Treat outliers as missing pixels and perform PCA using an algorithm for learning from incomplete data (MPPCA) to obtain $\boldsymbol{\mu}$, $\mathbf{U}$, $\boldsymbol{\lambda}$, and $\mathbf{A}$ from inliers only.
6:    Reconstruct the training images using $\boldsymbol{\mu}$, $\mathbf{U}$ and $\mathbf{A}$ and replace missing pixels in $\mathbf{X}$ with reconstructed values.
7: until the change in the outlier set is small.
```

Usually, only a few iterations (even only a single one) of this algorithm are sufficient for convergence.

In the initial step of this algorithm a standard PCA is performed on the entire set of training images. The obtained initial subspace $\mathbf{U}'$ is then used for the detection of outliers. Since the outlying pixels are not consistent with other pixels they usually appear as a high-frequency noise in a sequence of training pixels. Thus, they are modeled predominately with the eigenvectors, which correspond to small eigenvalues, while the consistent pixels (signal) are mostly modeled with the first eigenvectors. For that reason, only the first k' eigenvectors ($k' < k$) are used for the detection of outliers.

The value of k' and the threshold for outlier detection depend on the content and the type of training images and on the amount and the degree of the deviation of outliers. Most of all, they also depend on the application's goal and on the user's perception what should be considered as an outlier. Therefore, these values are application dependent and cannot be automatically determined in general.

A drawback of this algorithm is the first step, which still relies on the standard non-robust PCA. If the training images contain many outliers, they can distort the principal subspace in such a way that the detection of outliers becomes very unreliable. One solution to this problem is to divide each iteration of the algorithm into two stages.

5.2.2 Two-stage method

If the training set consists of a certain number of images, which contain a large portion of outliers, these images can degrade the obtained initial eigenspace considerably. Therefore, we propose to detect such *outlying images* and to calculate the initial eigenspace from other images, which are assumed to be predominately outlier-free. In this way, the obtained initial eigenspace is less distorted by outliers. Consequently, the detection of outliers in all images is more reliable.

Now, in each iteration a two-stage procedure is executed, where first the outlying images and then the outlying pixels are detected, as outlined in Algorithm 11.

Since $\mathbf{X}''$ mostly consists of training images, which contain not many outliers, the principal subspace $\mathbf{U}''$ is more appropriate for the detection of outliers than the initial principal subspace $\mathbf{U}'$. There is, however, an additional requirement: $\mathbf{X}''$ has to roughly encompass the same visual variability as $\mathbf{X}$. In other words, if the training images are sorted in an ordered sequence, $\mathbf{X}''$ has to enclose images, which are approximately evenly distributed over the entire sequence of training images $\mathbf{X}$. For that reason, not only the mean reconstruction error of an individual image, but also its position and the relationship to the other training images have to be considered, when the outlying images are detected and discarded in the step 4 of the proposed two-stage algorithm.

Algorithm 11 : RPCA2 – two-stage robust PCA

Input: data matrix $\mathbf{X}$, number of principal axes to be estimated k.

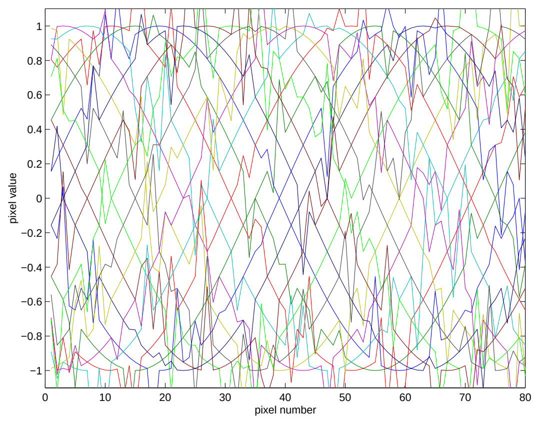
Output: mean vector $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$, coefficients $\mathbf{A}$.

```
1: repeat
2:     Perform the standard PCA on $\mathbf{X}$ and obtain $\boldsymbol{\mu}'$, $\mathbf{U}' \in \mathbb{R}^{M\times k'}$ and $\mathbf{A}' \in \mathbb{R}^{k'\times N}$.
3:     Reconstruct the training images using $\boldsymbol{\mu}'$, $\mathbf{U}'$ and $\mathbf{A}'$ and calculate the reconstruction error.
4:     Detect *outlying images*. Let $\mathbf{X}''$ be a set of non-outlying images.
5:     Perform the standard PCA on $\mathbf{X}''$ and obtain $\boldsymbol{\mu}''$, $\mathbf{U}'' \in \mathbb{R}^{M\times k'}$.
6:     Project all training images in $\mathbf{X}$ using $\boldsymbol{\mu}''$ and $\mathbf{U}''$, reconstruct the obtained coefficients $\mathbf{A}''$ and calculate the reconstruction error.
7:     Detect *outlying pixels* using either global or local threshold.
8:     Treat outliers as missing pixels and perform PCA using an algorithm for learning from incomplete data (MPPCA) to obtain $\boldsymbol{\mu}$, $\mathbf{U}$, $\boldsymbol{\lambda}$, and $\mathbf{A}$ from inliers only.
9:     Reconstruct the training images using $\boldsymbol{\mu}$, $\mathbf{U}$ and $\mathbf{A}$ and replace missing pixels in $\mathbf{X}$ with reconstructed values.
10: until the change in the outlier set is small.
```

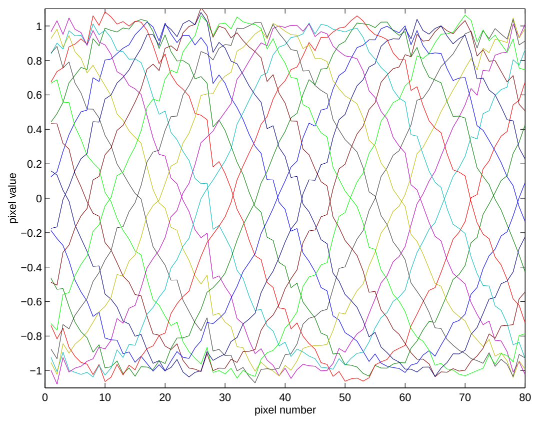
5.2.3 Simple example

Fig. 5.1 illustrates the performance of the one-stage robust algorithm. The training set consists of 40 1-D vectors, formed from shifted sinus functions (only half of them are shown). 20% of the elements are contaminated with random noise (Fig. 5.1(a)). The goal is to find the optimal principal axes for the representation of the original outlier-free vectors.

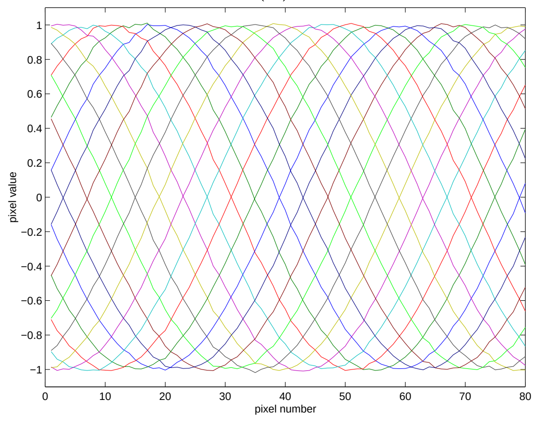
Figs. 5.1(b-d) depict the reconstruction after the second step (line 3 of the proposed algorithm) in the first iteration, and the reconstruction at the end of the first, and at the end of the second iteration. One can observe, how initially very noisy signals become more and more regular, converging to the ground truth. Therefore, the algorithm successfully detected outliers in the training signals and correctly reconstructed their values.



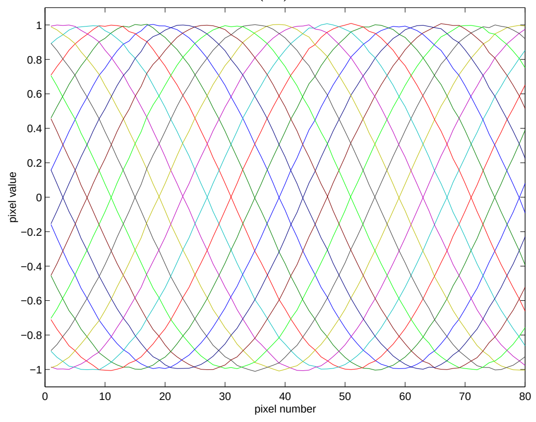
(a)



(b)



(c)



(d)

Figure 5.1: Simple example with noisy signals: (a) input data, (b) reconstruction after using standard PCA, (c) reconstruction after first iteration, (d) reconstruction after second iteration of robust algorithm.

5.3 Subsampling-based robust initialization

To further increase the effectiveness of the robust learning algorithm, the initial eigenspace should be estimated in a more robust way. Subsampling-based approaches (e.g., RANSAC [28]) are known to be more robust, although they are often computationally very demanding [23]. They are based on a random sampling of the input data and on determining subsets, which do not contain many outliers. These subsets are then used to calculate a solution.

The subsampling approach has been very successfully applied in the method for *robust recognition* [40]. In this section we will discuss, how can a similar approach be used for *robust learning* as well.

5.3.1 Subsampling-based robust recognition

Leonardis and Bischof [40] proposed a well known and widely used method for subsampling-based robust recognition. In the recognition stage, the task is to recognize a test image using the representation (principal subspace), which is assumed to be obtained from non-corrupted training images. The recognition is performed by robust estimation of eigenspace coefficients.

Their method is based on the observation that the coefficients can be estimated from a small subset of pixels by solving an overdetermined system of linear equations (4.22). Therefore, if the selected subset does not contain outliers and it is sufficiently representative, the obtained coefficients are approximately equal to the coefficients, which would be obtained from the outlier-free image.

Since the positions of outliers in the image are not known, a subsampling approach is used to exclude the outliers from the computation of coefficients. By randomly subsampling the image, a various subsets of pixels are formed. From these subsets different hypotheses about the true value of the coefficients are estimated. Finally, the best hypothesis, which gives the estimated values of the coefficients, is selected.

More specifically, they randomly choose a subset of pixels $\hat{\mathbf{x}}^\bullet$ and apply the following robust procedure. First, they calculate the coefficients by solving an overdetermined system of linear equations (4.22) arising from $\hat{\mathbf{x}}^\bullet$. Then, based on the error distribution of the subset of pixels, they reduce the number of selected pixels by a predefined factor α (they exclude those points with the largest reconstruction error) and solve (4.22) again with the reduced set of pixels. They repeat this procedure until the size of $\hat{\mathbf{x}}^\bullet$ is reduced to a predefined number of pixels.

After that, they include in the computation of coefficients a_j all compatible pixels. They reconstruct the image $\hat{\mathbf{y}}$ and select all pixels from $\hat{\mathbf{x}}$ where the reconstruction error $|\hat{y}_i - \hat{x}_i|$ is consistent with the distribution of the reconstruction errors of the pixels, which remained in $\hat{\mathbf{x}}^\bullet$ after α-trimming. Then they recalculate the coefficient vector $\mathbf{a}$ using all compatible points to form a hypothesis.

However, one cannot expect that every initial randomly chosen set of pixels will produce a good hypothesis, despite the robust procedure. Thus, to further increase the robustness of the hypothesis generation step, i.e., increase the probability of detecting a correct hypothesis if there is one, they initiate, as in [5, 28], a number of trials. Then, in the selection step, they choose the best hypothesis, i.e., the hypothesis with the smallest reconstruction error of compatible points.

The algorithm for robust recognition, which will be used in the procedure for incremental robust learning as well, is summarized in Algorithm 12.

Algorithm 12 : robRec – robust recognition

Input: test image $\mathbf{x}$, principal axes $\mathbf{U}$, mean image $\boldsymbol{\mu}$.

Output: coefficient vector $\mathbf{a}$.

```
 1: repeat
 2:    Randomly choose a subset of pixels $\hat{\mathbf{x}}^\bullet$ from $\mathbf{x} - \boldsymbol{\mu}$.
 3:    repeat
 4:       Calculate the coefficient vector from the subset of pixels: $\mathbf{a} = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^\bullet$.
 5:       Reconstruct the selected pixels: $\hat{\mathbf{y}}^\bullet = \mathbf{U}^\bullet\mathbf{a}$.
 6:       Exclude the pixels with the largest reconstruction error from $\hat{\mathbf{x}}^\bullet$.
 7:    until $\hat{\mathbf{x}}^\bullet$ is reduced to a predefined number of pixels.
 8:    repeat
 9:       Reconstruct all pixels $\hat{\mathbf{y}} = \mathbf{U}\mathbf{a}$.
10:       Add the compatible pixels to $\hat{\mathbf{x}}^\bullet$.
11:       Re-calculate the coefficient vector: $\mathbf{a} = \mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^\bullet$.
12:    until the change in $\hat{\mathbf{x}}^\bullet$ is small.
13: until the number of hypotheses is sufficient.
14: Select the best hypothesis for $\mathbf{a}$.
```

Fig. 5.2 shows the execution steps of the proposed algorithm in the case of a good hypothesis. Figs. 5.2(b-g) depict the pixels (in grey) that were used for calculating the coefficient vector from the occluded range image shown in Fig. 5.2(a). As one can observe, in α-trimming phase all the pixels in the occluded region were eliminated, and later were not included in the set of compatible points. Therefore, the coefficient vector $\mathbf{a}$ was calculated only from "good" pixels which resulted in a perfect reconstruction (Fig. 5.2(h)).

5.3.2 Main idea for subsampling-based learning

Due to demonstrated performance of the method for robust recognition, we wanted to introduce the subsampling approach into the process of *learning* as well. However, it turned out that such an extension is not straightforward.

There are at least four issues that make subsampling-based learning a much more difficult task. Firstly, the representation of the object or scene is not given

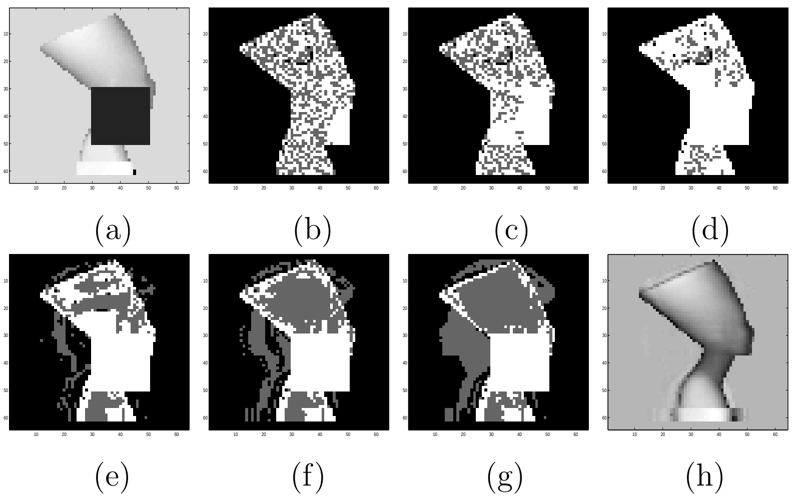


Figure 5.2: Robust estimation of coefficients: (a) Occluded image, (b-d) three α-trimming iterations, (e-g) adding compatible points, (h) reconstructed image.

and has to be learned. Secondly, not only one, but a lot of images (an entire training set) have to be subsampled. Thirdly, the subsampling cannot be performed in an arbitrary way. And finally, the representation cannot be directly learned from small and sparse samples of pixels.

To overcome these difficulties, we do not calculate the principal axes directly. Instead, we rather use a subsampling scheme to detect outliers and then employ the modified EM algorithm to robustly estimate the principal axes from inliers.

The main idea is to randomly choose a subset of pixels, perform the standard PCA on them, and reconstruct pixel values from the estimated coefficients. Then we repeat this procedure on several different subsets of pixels and obtain several reconstructions of each pixel. Since each of these reconstructed values is potentially the true pixel value, we call it *a hypothesis*. At the end, the best hypothesis is selected for each pixel and used for the detection of outliers. The procedure is outlined in Algorithm 13.

This procedure replaces the first step in the first iteration of the robust algorithm presented in the previous section. It increases the robustness of the initialization of the algorithm. In the subsequent iterations the standard PCA can be used instead.

The crucial point of this procedure is how to choose a subset of pixels and how to select the best hypothesis. If these two steps are performed well, this algorithm can improve the robustness of the basic robust algorithm.

Algorithm 13 : robIni – subsampling-based robust initialization of RPCA

Input: data matrix $\mathbf{X}$.

Output: detected outliers.

```
1: repeat
2:     Choose a subset of pixels $\mathbf{X}^\bullet$ from the data matrix $\mathbf{X}$.
3:     Perform the standard PCA on the selected subset. Obtain $\boldsymbol{\mu}^\bullet$ and $\mathbf{U}^\bullet$.
4:     Reconstruct the selected pixels $\mathbf{Y}^\bullet = \mathbf{U}^\bullet\mathbf{U}^{\bullet\top}(\mathbf{X}^\bullet - \boldsymbol{\mu}^\bullet\mathbf{1}_{1\times N^\bullet})$ forming a hypothesis for each selected pixel.
5: until the number of generated hypotheses is sufficient.
6: Select the best hypothesis for each pixel.
7: Detect outliers by comparing input and estimated pixel values.
```

5.3.3 Random generation of hypotheses

Looking from the statistical point of view, PCA is performed on a set of variables (pixels in the context of computer vision) measured on a set of statistical units (images). If we do not want to cope with missing data, all variables should be measured on all statistical units. Thus, the subsampling cannot be performed in an arbitrary way by randomly selecting pixels across the entire input data matrix.

We rather generate the hypotheses in the following way. First, we randomly select a subset of pixel indices and then we randomly select a subset of images. A sample is formed from all pixels with selected indices on all selected images. These pixels appear on intersections of the selected rows (pixel indices) and columns (images) in the data matrix shown in Fig. 5.3. From each sample an eigenspace is built and the pixels are reconstructed to provide one hypothesis for each selected pixel.

The goal of the hypotheses generation step is to form for each pixel at least one hypothesis, which is obtained from a "good" subset of pixels. A "good" sample should contain as few outliers as possible, assuring small reconstruction errors in selected inliers and yielding the correct reconstructed values in selected outliers as well. In such a way, a hypothesis, which is very close to the correct value, can also be generated for an outlier.

5.3.4 Semi-random generation of hypotheses

We can increase the probability that some sample contains predominantly inliers by generating a large number of hypotheses. Moreover, we can also increase this

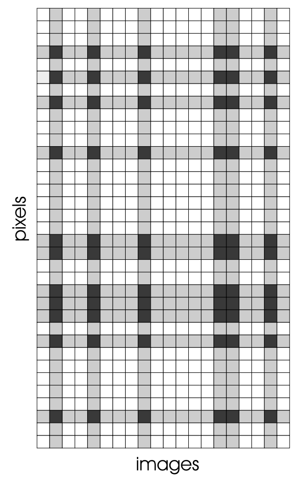


Figure 5.3: Randomly selected pixels in a data matrix.

probability by choosing more reliable pixels, i.e., the pixels, which are very consistent with the information contained in the rest of the images. Thus, we generate the hypotheses in the following way. First, we perform the standard PCA on the entire set of input images. Then we calculate the mean reconstruction errors across images (columns in the data matrix) and across pixel vectors (rows). We choose the images and the pixel vectors with the smallest reconstruction errors. By intersecting the chosen images and pixel vectors, we obtain the set of the most reliable pixels. Later, we add a subset of randomly chosen pixels to form a sample. Thus, each hypothesis is obtained from a set consisted of the most reliable pixels and the randomly selected pixels. We call this procedure semi-random generation of hypotheses.

One can argue that the first step is still the non-robust standard PCA. However, in this case we are looking only for a small subset of very reliable pixels with the smallest reconstruction errors, which are probably not affected by the outliers too much. However, if needed, a more robust technique or measure can be used (e.g., MAD (median absolute deviation) [23, 62]).

5.3.5 Selection of hypotheses

In the hypotheses generation step, a cost value is assigned to each generated hypothesis. In the end, in the selection step, a hypothesis with the smallest cost value is selected for each pixel in the data matrix. A very important issue is the evaluation of the hypothesis, i.e., how to obtain the cost value, which reflects the difference between the hypothesis and the correct value of the pixel.

One would expect that the mean reconstruction error of the selected pixels composing a “good” sample is rather small. Therefore, it could be a good measure for evaluation of the hypotheses. However, it turns out that not all selected pixels equally influence the estimation of a hypothesis. Pixels in the same pixel vector (a row in the data matrix) affect more the reconstructed values of other pixels in the pixel vector than the reconstructed values in other parts of the data matrix. For that reason, we calculate the cost value for a hypothesis as the mean reconstruction error of the selected pixels belonging to the same pixel vector (row) as the pixel, which the hypothesis is being calculated for. This approach has produced better results than all other approaches that we have considered.

After the best hypothesis for each pixel is selected, we obtain the reconstructed image, where the values in inliers as well as in outliers should tend to the correct values. Therefore, the reconstruction error in outliers is large, which makes their detection easier. The detected outliers are then regarded as missing pixels; the principal subspace is estimated from inliers only, using the proposed batch method for robust learning.

5.4 Robust incremental method

The batch method for robust learning is iterative. At each iteration an approximation of the final eigenspace is estimated, which makes possible to detect outliers and refine the estimated subspace throughout the subsequent iterations. However, at each iteration all the training images are needed for estimating the eigenspace, therefore all of them have to be given in advance.

Incremental algorithm, on the other hand, processes images sequentially one by one, thus only one image has to be available at each time instant. However, at each step a representation of the already learned images is available, which makes possible to detect outliers in the current image. By assuring that the outliers are sequentially detected and that the images are added in a robust way we arrive at a robust incremental learning algorithm.

5.4.1 Algorithm

In the robust framework we are aware that images may contain outliers. We treat as outliers all pixels, which are not consistent with the information contained in other images. Since at each step we have a current model of the object or scene seen so far, we can detect outliers in the new image and replace them with the values, which are yielded by the current model.

This is achieved by projecting the new image into the current eigenspace in a robust manner. Instead of a simple projection, a robust procedure summarized in Algorithm 12 (robRec) is used. Coefficients are obtained mainly from inliers, thus their reconstructions tend to the correct values in outliers as well. Consequently, the reconstruction error in outliers is large, which makes their detection easier.

After the outliers are detected, their values are replaced with the reconstructed values. Since these values are yielded by predominantly inliers and the current model, they are good approximations of the correct values. Such an improved outlier-free image is then used for updating the eigenspace. Since the outliers are detected during the learning process using a robust procedure, the obtained eigenspace is robust as well.

The overall algorithm for robust incremental learning is presented in Algorithm 14.

A potential danger of error propagation in the method for robust incremental learning is even more pronounced as in the case of weighted incremental learning. Not only the recovery of the unreliable pixels, but also their detection can cause incorrect results, which are then propagated throughout the learning process. If the outliers are incorporated into the representation, especially in the early stages of the learning process, then the inliers in subsequent images are recognized as outliers and are not modeled. Similarly, if the initial eigenspace encompasses only a limited number of appearances of an object or a scene, then all the pixels in the subsequent images, which significantly differ from the appearances of the first images, will be considered as outliers and no novel information will be added to the model.

Therefore, the initial eigenspace, which is obtained by using the batch method, should be reliable and stable. It should roughly model heterogeneous appearances of an object or a scene and it should be obtained from a set of pixels containing as few outliers as possible. Thus, the reliable images from various parts of the image sequence should be used for initialization. When the model encompasses a sufficient number of appearances it becomes more stable and this is no longer a problem. Certainly, the initialization also has to be performed in a robust manner, using a robust batch algorithm.

Algorithm 14 : RIPCA – robust incremental PCA

```
Input: current mean vector $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients
    $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$.
Output: new mean vector $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$,
    new eigenvalues $\boldsymbol{\lambda}^{(n+1)}$.
 1: Estimate the coefficient vector $\mathbf{a}$ from the new image $\mathbf{x}$ using the robust method
    robRec.
 2: Reconstruct the obtained coefficient vectors: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 3: Detect outliers $\mathbf{x}^\circ$ in $\mathbf{x}$ (pixels, where $|y_i - x_i| > \mathit{threshold}$).
 4: Replace the outliers in $\mathbf{x}$ with the reconstructed values: $\mathbf{x}^\circ = \mathbf{y}^\circ$.
 5: Project the obtained image $\mathbf{x}$ into the current eigenspace: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$ .
 6: Reconstruct the new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
 7: Compute the residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
 8: Append $\mathbf{r}$ as a new basis vector: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{bmatrix}$ .
 9: Determine the coefficients in the new basis: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ .
10: Perform PCA on $\mathbf{A}'$. Obtain the mean value $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{U}''$, and the
    eigenvalues $\boldsymbol{\lambda}''$.
11: Project the coefficient vectors to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^\top(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$ .
12: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$ .
13: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ .
14: New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ .
```

5.4.2 Simple 2-D example

To demonstrate the behavior of the robust incremental algorithm, we significantly changed the values of the second coordinate of five points in our 2-D example. Fig. 5.4 shows that when the non-robust incremental method is used, these outlying points pull the origin in a wrong direction and incorrectly orient the estimated principal axis. On the other hand, the robust method sequentially detects the outlying coordinate values, replaces these values with their reconstructions (shown as circles) and updates the eigenspace accordingly. In the end, the principal axis obtained using this approach is very close to the optimal one.

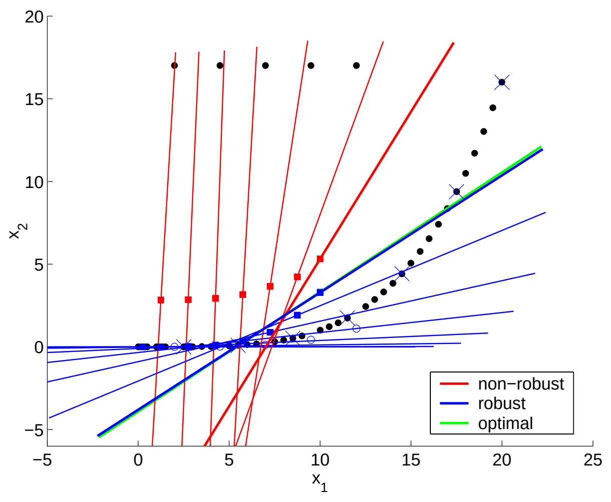


Figure 5.4: Illustration of robust incremental learning.

5.5 Experimental results

In this chapter we presented several algorithms for robust learning of eigenspaces. In this section we will show the results of the experiments, which were performed to evaluate the proposed methods.

5.5.1 Batch method

First, we will evaluate the results obtained by applying the batch algorithm to a set of panoramic images and compare the results of our approach to the ones obtained by a recently proposed method [23] on their set of input images.

Robust PCA on Panoramic Images

First, we tested the performance of the one-stage batch method on the images with known ground truth shown in Fig. 5.5. We captured 30 panoramic images of the size 100×100 in the faculty hall. Then, we synthetically applied gradual illumination changes and nonlinear illumination changes (a shadow—the vertical “cloud”) to this set of images. In addition, we added, as an outlier area, a square on a randomly chosen position in every image. The goal was to learn the panoramic representation capturing the illumination variations (linear and nonlinear) but discarding the outliers. Since these images are temporally well correlated, we included the smoothness prior in M-step of the EM algorithm for the calculation of the principal axes from incomplete data.

The results are depicted in Fig. 5.5. In the images reconstructed from the first principal component obtained with the standard PCA, one can clearly observe that the linear illumination changes are modeled, but not the nonlinear. If the model consists of the first 8 principal axes produced by the standard PCA, then all illumination changes are captured in the reconstructions, however, the model also contains the outliers (squares). On the other hand, using our robust algorithm, one can observe from the reconstructions based on the first 8 principal axes, that all illumination changes are captured in the model, while the outliers are not, which is exactly what we want to achieve.

Since the positions of the outliers are known, we can calculate the mean absolute reconstruction error in the outliers (squares) and in the inliers (background). The reconstruction error in the inliers should be as small as possible, while the reconstruction error in the outliers should be large, enabling the efficient detection of outliers. Table 5.1 compares the reconstruction errors with the errors obtained using optimal principal axes, which were estimated from the data without outliers (ground truth). It is evident that the robust PCA outperforms the standard one since the errors obtained with the proposed algorithm are much closer to the optimal ones.

Table 5.1: Comparison of the reconstruction errors obtained using the standard and the robust PCA.

		num.	rec. error in	
data	method	of PA	inliers	outliers
ground truth	standard PCA	8	1	2805
with outliers	standard PCA	1	146	2601
with outliers	standard PCA	8	21	540
with outliers	robust PCA	8	6	2608

We also applied the proposed method to a sequence of panoramic images, which were captured at several locations in the laboratory. At each location a sequence of 180 images of the size 100×100 was captured, of which 60 were used in the learning stage. During the acquisition, we varied the illumination conditions and people were free to walk around in the laboratory. To enable an efficient and robust appearance-based localization of a mobile platform, a representation has to be built which parametrically models the laboratory under different illumination conditions

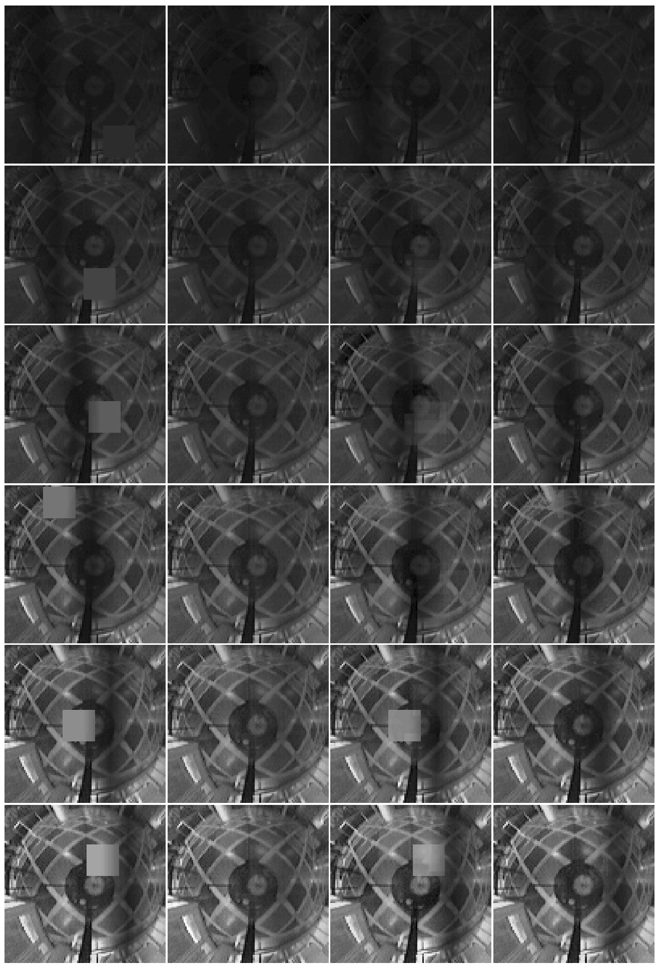


Figure 5.5: Comparison of our method with standard PCA. (a) Input images (every fifth image from training set). (b) Reconstruction based on first principle axis (PA) using standard PCA. (c) Reconstruction based on first 8 PA using standard PCA. (d) Reconstruction based on first 8 PA using *robust PCA*.

but at the same time excludes randomly moving subjects (objects) in the training images.

The results for one sequence are shown in Fig. 5.6. Since people appear in most of the training images, the standard PCA incorporates them in the representation. Consequently, they appear in the reconstructed images as undesirable "ghost people". In contrast, the images, which were reconstructed based on the output of the robust PCA, do not contain these effects since the outliers (people) are eliminated from the representation during the process. Therefore, these images represent solely the appearance of the location under different illuminations. Robust representations obtained at different locations can be combined in an overall appearance-based representation of the laboratory, suitable for mobile robot localization and navigation, or for performing surveillance tasks.

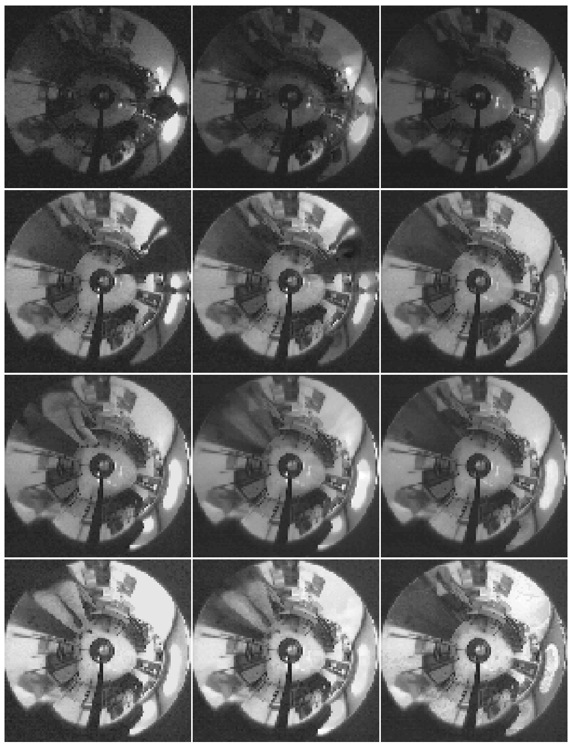


Figure 5.6: (a) Four panoramic images from sequence. Their reconstructions based on (b) standard PCA, and (c) robust PCA.

Comparison with the Previous Work

We performed an experiment where we applied our method on the same image sequence[1] of 256 images of size 120×160 pixels as De la Torre and Black, who recently proposed a method for robust learning of appearances based on PCA [23]. Some of the results obtained by our algorithm are presented in Fig. 5.7. By visually comparing Fig. 5.7 with Fig. 8 in [23], we can conclude that both algorithms produce

[1]Images were obtained from http://www.salleurl.edu/~ftorre/papers/rpca2.html.

very similar results. However, as reported in [23], their algorithm takes three hours on a 900 MHz Pentium III to produce these results, while our algorithm finishes the task in 19 minutes on a 550 MHz Pentium III.

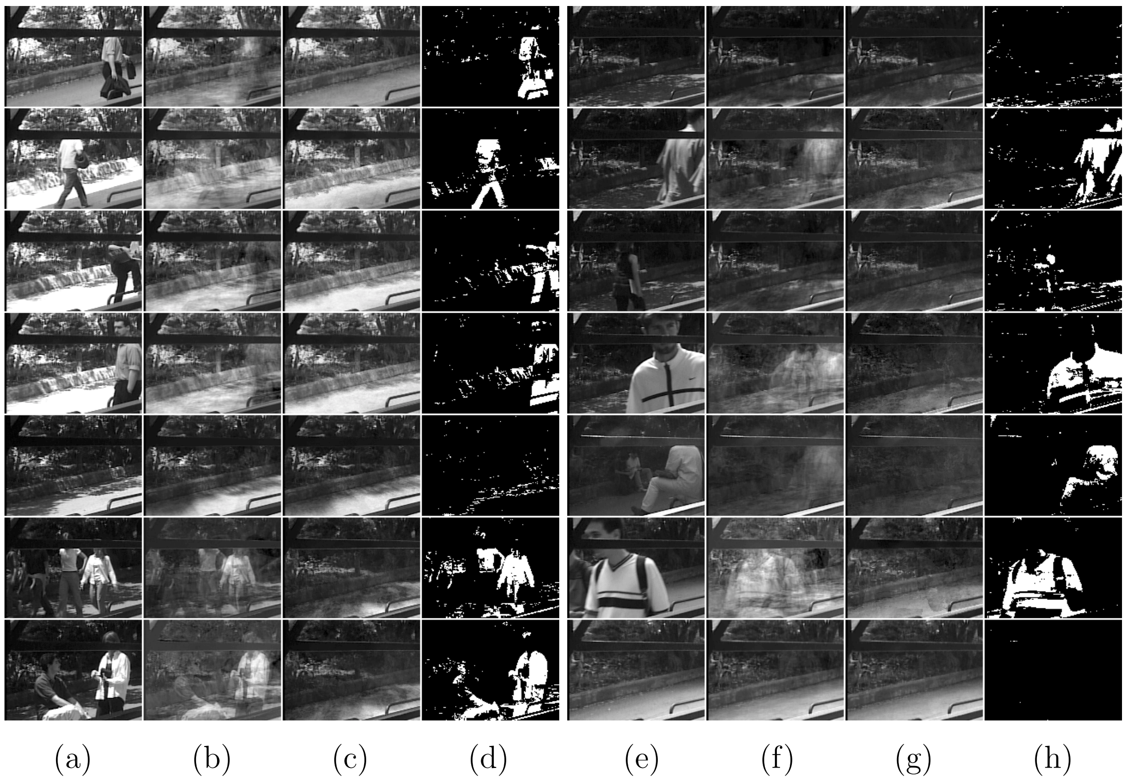


Figure 5.7: (a),(e) Original data from [23], (b),(f) standard PCA reconstruction, (c),(g) our robust PCA reconstruction, (d),(h) outliers obtained by our method.

5.5.2 Subsampling-based initialization

We evaluated the performance of the method for subsampling-based initialization on the images with known ground truth, similar to the images which were used for evaluation of the robust method. We enlarged the outlying squares in order to degrade the results of the robust method, which uses the standard PCA for initialization. Furthermore, we added a square on a randomly chosen position in upper half of every second image only (see Fig. 5.9(a)) to investigate how different strategies of subsampling and selection of hypothesis influence the results. The goal of the method was to enable the efficient detection of outliers in the initialization step of the robust learning algorithm.

We compared the results obtained by the standard PCA (in plots indicated as *std*), with the results of the proposed method, where the random sampling (*rnd*) and the semi-random sampling (*srnd*) were used for the generation of hypotheses. In addition, we generated the results assuming that the determination of the most reliable images and pixels in the hypotheses generation step was optimal (*srndOG*). Finally, we obtained also the optimal results (*opt*) using the best possible selection criterion (*srndOGS*) (since we knew the ground truth, we could select a hypothesis, which was the closest to the correct value).

Table 5.2 presents the mean absolute reconstruction errors (MARE) in inliers and in outliers for all approaches. MARE produced by the proposed approach is smaller in inliers and slightly bigger in outliers in comparison with the standard method. This increases the ratio between these two values, making the outlier detection much easier and more reliable.

MARE	*std*	*rnd*	*srnd*	*srndOG*	*srndOGS*
in inliers	6.9	5.7	3.7	3.6	2.5
in outliers	28.4	29.5	29.0	30.3	31.2
inIL/inOL	4.11	5.16	7.93	8.34	12.62

Table 5.2: MARE obtained using different learning approaches.

Very similar conclusions can be drawn also from the plots in Fig. 5.8. The proposed method narrows the histogram of reconstruction errors in inliers (see Fig. 5.8(a)). As a consequence we obtain less false positives during the process of outlier detection (Fig. 5.8(b)). This is also evident from the example of five images depicted in Figs. 5.9(b,c). Outliers which were detected correctly, are shown in white, while false detections are gray. The most descriptive approach for the evaluation of different methods is depicted in Fig. 5.8(c). It plots a portion of false positives (FP), which are obtained when the thresholds for detecting outliers are set to the levels that produce certain portions of false negatives (FN). As we can see, the proposed approach produces less false positives than the standard approach for all levels of FN. The same results are given also using more familiar precision/recall curves in Fig. 5.8(d).

The results show that the proposed method outperforms the non-robust standard PCA. They also show that the semi-random generation of hypothesis significantly improves the results of the random generation. We can also see that by improving the criterion for determining the most reliable images and pixels, the results can

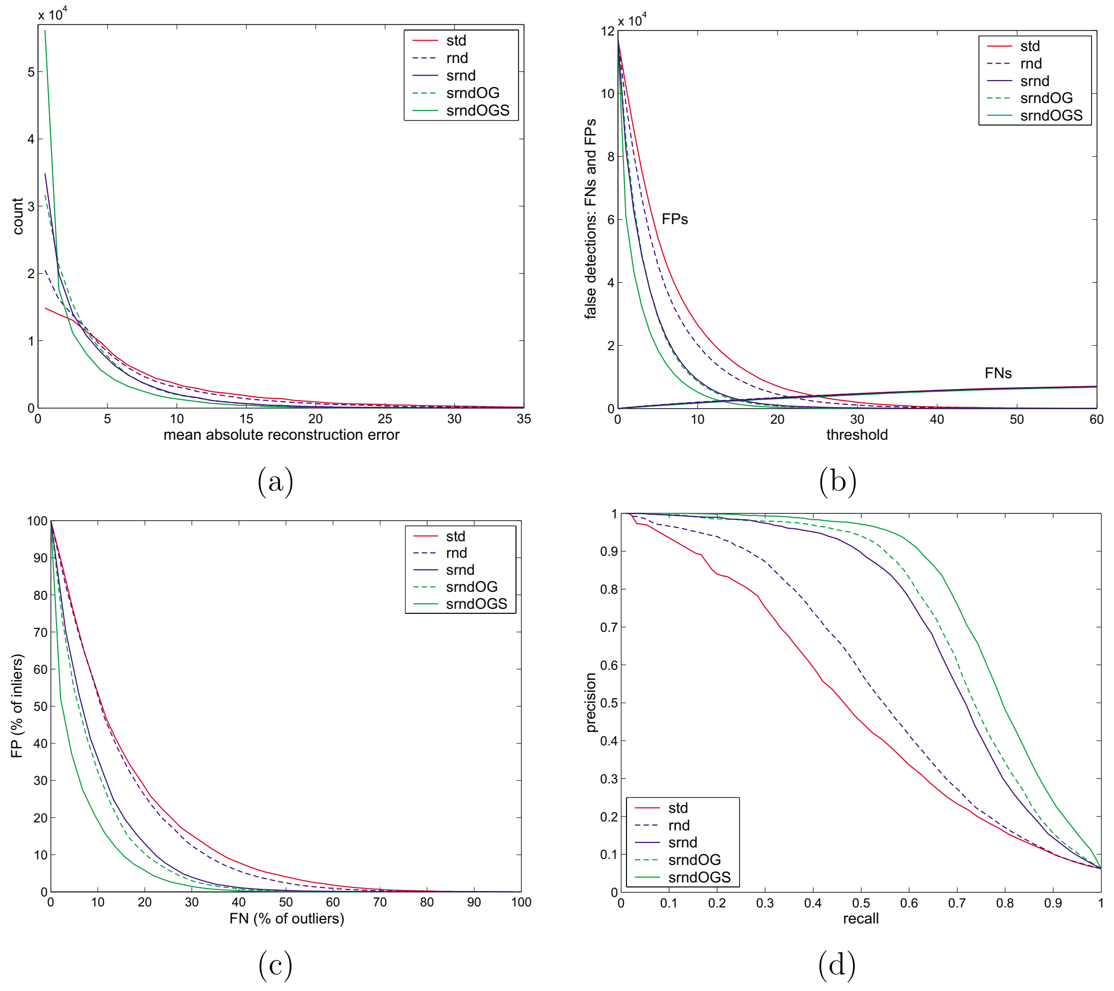


Figure 5.8: (a) Histogram of reconstruction errors in inliers. (b) Number of FP and FN for various thresholds. (c) Number of FP for various numbers of FN. (d) Precision/recall curves.

be further improved. And finally, we can also observe that the optimal results are even better. Therefore, by using a better selection criterion, the results could be improved even further.

Since the goal of the proposed algorithm is to robustly initialize the algorithm for robust eigenspace learning, we also compared the results produced by this algorithm. Fig. 5.10 plots FP/FN curves obtained by this algorithm using different initialization approaches. One can observe that the robust learning algorithm significantly improved the results of the initialization stage (please, note here the different scale than in Fig. 5.8(c)). Moreover, one can also observe, that the results of the robust algorithm initialized by the proposed subsampling-based approach are significantly better than the results of the robust method initialized by the standard approach. This was in fact the main goal of the proposed algorithm.

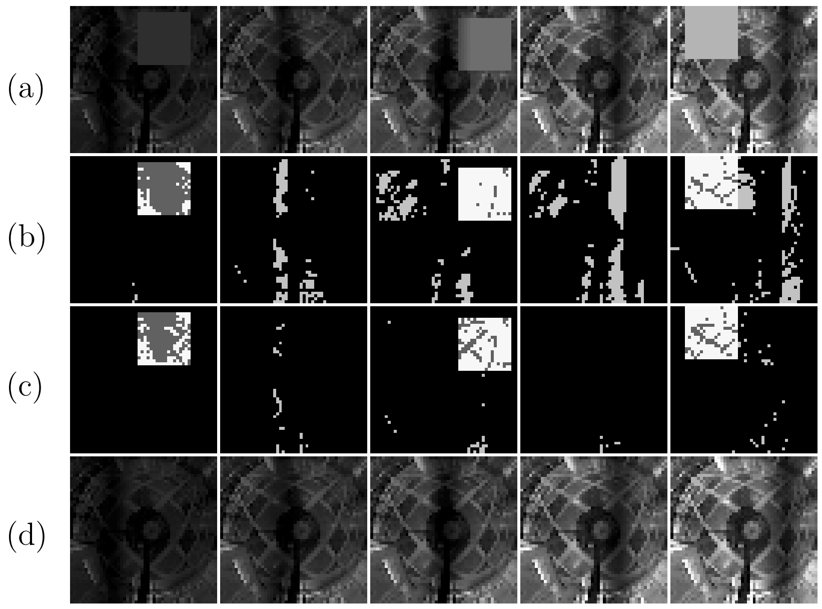


Figure 5.9: (a) Training images. Outliers obtained using (b) *std* and (c) *srnd* methods. (d) Reconstructed training images after applying robust learning algorithm initialized using *srnd* approach.

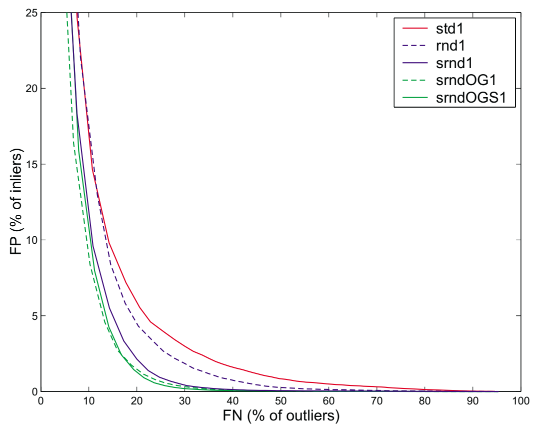


Figure 5.10: FP/FN curves obtained using robust learning algorithm.

However, one has to note that this particular image set was very appropriate for the subsampling-based initialization, since one half of evenly spread images was outlier-free as well as the bottom half of each image. Therefore, it was relatively easy to correctly determine the reliable images and pixels. However, if the outliers are evenly spread over the whole training images (which is, in fact, not very common in the real world applications), the improvements obtained by using the proposed subsampling-based approach often do not justify the application of this method due to its high computational requirements. Then again, we can reduce this requirement by processing input data matrix in smaller parts. The outliers can be detected for every part independently and later merged together. In this way, spatially and temporally local properties of images can have a stronger influence on the detection of outliers, which can even improve the results.

5.5.3 Incremental method

We tested the performance of the proposed robust incremental method on the images with known ground truth as well. In this case we added a square on a randomly chosen position in 80% of the images (see Fig. 5.11(a)). Thus, each fifth image did not contain outliers. The goal was, again, to learn the representation capturing the illumination variations (linear and nonlinear) but discarding the outliers.

We tested several approaches to exhibit some properties of the proposed method. The results are given using two measures. The first measure is the mean squared reconstruction error of the reconstructed outlier-free (ground truth) images (Table 5.3). Besides MSRE, a precision/recall curve is given for each method in Fig. 5.12. A P/R curve indicates the utility of a method for outlier detection. If the learned representation does not include outliers, this ability is high and the precision and recall values are close to 1. In addition, some reconstructed training images are visualized in Fig. 5.11(b–e).

First we applied the standard batch method on ground truth images, i.e., training images without outliers (*batchOnGT*), which produced the optimal results. Then we applied the standard batch method (*batchStd*) on the training images containing outliers, which generated poor results, since the standard method is sensitive to occlusions. Therefore, we applied the robust batch method (*batchRob*), which produced better results. However, since a lot of relatively large occlusions were present in the training images, the results were still not satisfactory.

Then, we tested the proposed robust incremental method. First we applied this method under the assumption, that the occlusions were known (*robIncKnownOL*). Therefore, the algorithm did not have to detect outliers. The results are excellent; they are very close to the optimal ones. This means that the algorithm for updating the eigenspace works fine even if some data in the input images are missing and that the efficiency of the robust incremental algorithm mainly depends on the ability to detect outliers.

It turns out that this ability significantly depends on the initial stage of the learning process. If the seed (the initial eigenspace, which is used for the initialization of the incremental algorithm) is not reliable and is affected by occlusions (*robIncPoorSeed*), the results are not very good. If the seed is too small and is built from the training images, which are not dispersed over the whole image sequence (*robIncNonDispSeed*), the results are very poor. To demonstrate this, we built the seed using a few images from the first half of the image sequence. Consequently, the first half of the images was reconstructed well, however the images from the end of the sequence were reconstructed badly. Since not even the rough appearance of these images was encompassed in the initial eigenspace, all the changes in these images were considered to be outliers and were not added into the representation. For this reason, the vertical cloud was not modeled correctly as can be observed in Fig. 5.11(d).

Finally, we built the seed from the images with the smallest reconstruction errors (images without outliers), which were evenly dispersed over the whole image sequence (*robIncGoodSeed*). This approach produced excellent results, which are rather close to the optimal ones. This indicates that when the eigenspace, which is being updated, is stable enough, i.e., roughly encompassing different views of objects or scenes, the outliers in the training images are successfully detected and correctly reconstructed.

The latter approach is very related to the two-stage robust batch method. In fact, this method also produces very similar results, which are significantly better than the results of the one-stage batch method. From this we can deduce that the initial representation, which is learned in the beginning of the learning process, is crucial for efficient estimation of outliers and robust learning.

batch			*robInc*			
OnGT	*Std*	*Rob*	*KnownOL*	*PoorSeed*	*NonDispSeed*	*GoodSeed*
1.7	61.1	29.8	2.0	21.2	166.0	2.9

Table 5.3: MSRE obtained using different learning methods and seeds.

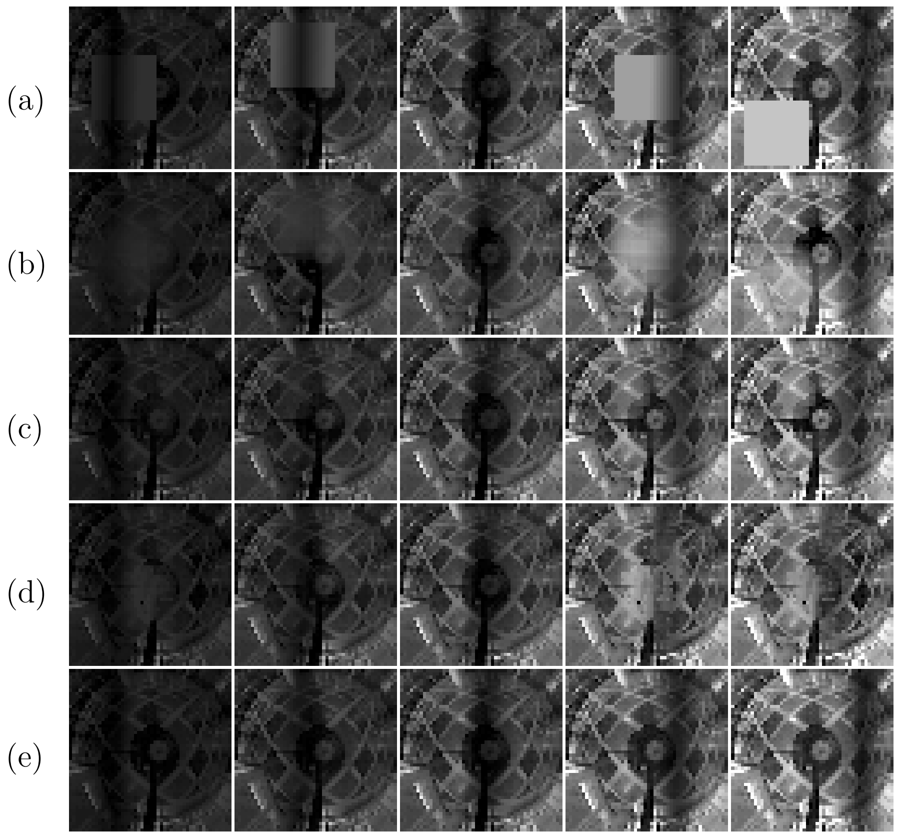


Figure 5.11: (a) Training images, reconstructions using (b) *batchStd*, (c) *batchRob*, (d) *robIncNonDispSeed*, and (e) *robIncGoodSeed* approaches.

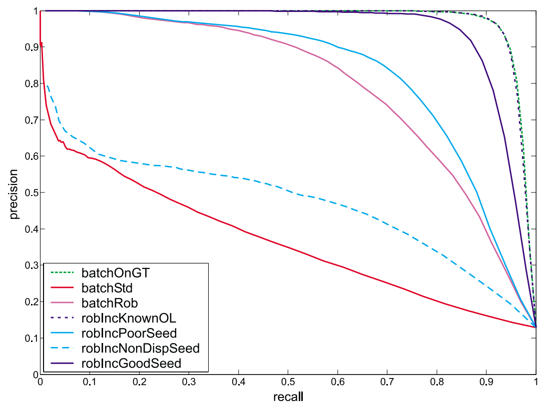


Figure 5.12: Precision/recall curves obtained using different learning methods and seeds.

5.6 Chapter summary

In this chapter we presented robust algorithms for eigenspace learning. The algorithms iteratively detect outliers in the training images. These outliers are then treated as missing pixels and the principal subspace is estimated from inliers only, using one of the algorithms for learning from incomplete data presented in the previous chapter.

First, we presented two versions of the batch method for robust learning. The first one is the one-stage method, which detects outliers using the representation built form all training images. The second version of this method first detects outlying images and then detects outlying pixels using the representation built from consistent images only. Since the initial step of this method is still non-robust, the robust initialization of this algorithm based on a subsampling approach was also proposed.

Finally we presented the method for robust incremental learning. The method sequentially updates the representation using the previously acquired knowledge for determining consistencies and discarding inconsistencies in the input images. The experimental results show that the performance of the robust learning algorithm depends on the early stages of the learning process. If the initial representation is built satisfactorily, the inconsistencies in other training images are successfully detected and the representation is robustly built, encompassing all consistent information in the training images and discarding outliers.

Like in the previous chapters, several experiments were performed to evaluate the proposed methods. In the next chapter, we will present the results of additional experiments, which will give more global comparison of the algorithms presented in all chapters of this dissertation.

Chapter 6

Performance evaluation

6.1 Chapter overview

Each of the previous three chapters includes a section where the experimental results, which serve for evaluation of the presented methods, are shown. Namely, in Section 3.3 the experiments using incremental PCA approach are discussed, in Section 4.6 the results of the weighted approaches are shown and in Section 5.5 the robust methods are evaluated.

In this chapter we will present additional experimental results, which will clarify some aspects of the proposed methods and give a global view to all algorithms. Furthermore, we will compare the performance of the methods on various types of images and determine their usefulness in different scenarios. First, we will illustrate the performance of the incremental algorithms for building the representations of faces. Then we will evaluate most of the proposed algorithms on three image domains with different characteristics. From these experiments we will draw some conclusions about the applicability of the proposed algorithms.

6.2 Performance of incremental algorithms

First we present the results of the experiments where the incremental PCA was used for building *representations of faces*. For the testbed we used the ORL face database [64] consisting of 400 images (10 images of each of 40 subjects), rescaled to the size of 32×32 pixels. The images entered into the learning process sequentially one by one (one image of each of 40 subjects first, then the second image of each subject and so on to the tenth image). Six training images are shown in Fig. 6.2(a).

The goal was to represent all 400 images with just 25 images (eigenfaces) using incremental learning.

First, we present the performance of the non-robust incremental method in comparison with the standard batch method. Table 6.1 shows the mean squared reconstruction errors (MSRE) of the images reconstructed from the coefficients obtained by projecting the training images into the eigenspaces, which were built using the batch method (*batch*) and the proposed incremental method (*incX*). The results are very similar; MSRE obtained using the incremental method is only 1% worse. When the coefficients, which were obtained and maintained during the learning process, are reconstructed, the mean squared reconstruction error (*incA*) is still very similar. In this case, the degradation of the results is 2%. Figs. 6.2(a-c) show some training images and their reconstructions using the batch and the incremental (*incA*) method. We can hardly notice any difference between both reconstructed images. The results show that the proposed incremental method is almost as efficient as the batch method (which is optimal in the sense of MSRE) and are completely in accordance with the results from Section 3.3.

In the second experiment we put different temporal weights on the training images. Since people's faces are changing through time, we would prefer that the latest images are better represented than the old ones. Such representation would be more appropriate for face recognition, since new images of faces, which will have to be recognized, will be more similar to the last training images than to the first ones. Therefore, we put larger weights on the images at the end of the image sequence. The results are depicted in Fig. 6.1. For every group of 40 images the mean squared reconstruction error is presented for non-weighted (*incA*, *incX*) and weighted (*WincA*, *WincX*) incremental method. We can observe that the MSRE in the last four groups of images is smaller when the weighted method is used. This means that these images are better represented and their reconstructions are more detailed as can be observed on the last two images in Fig. 6.1(d). Although the overall mean squared reconstruction error of all training images is bigger, the *weighted* reconstruction error is smaller, as presented in Table 6.1. This is exactly what we wanted to achieve.

Then we erased a quarter of each of the last 360 images (Fig. 6.2(e)) to test the performance of the incremental algorithm which sequentially estimates the values of missing pixels. We left the first 40 images complete in order to make possible to learn at least the initial representations from the complete data. First we calculated

	batch	*incA*	*incX*	*WincA*	*WincX*
MSRE	582	594	587	636	602
WSRE		598	597	570	562

Table 6.1: Results of batch PCA, IPCA and WIPCA.

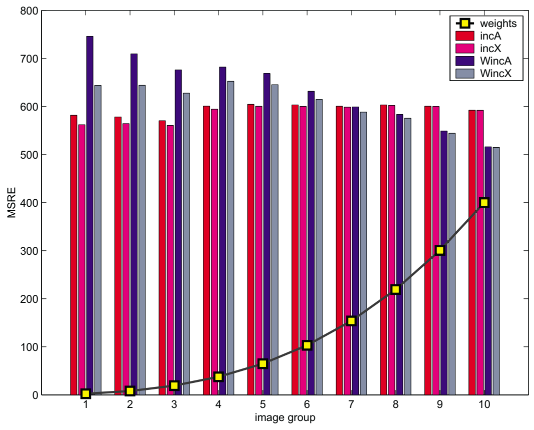


Figure 6.1: Results of weighted incremental PCA.

the mean image over all non-missing pixels, imputed the missing quarter on each image with the mean values and performed the standard incremental PCA. Then we applied the incremental algorithm for learning from incomplete data, which sequentially imputed missing pixels using the previously acquired knowledge. This method reconstructed the missing quarters significantly better, as can be observed in Figs. 6.2(f,g).

Table 6.2 shows the overall mean squared reconstruction error between the complete original images and the reconstructed images obtained from the coefficients, which were estimated during the incremental learning. The error obtained using the method which sequentially estimates missing pixels (*robust*) is significantly smaller than when the simple mean-substitution was used (*standard*).

Finally, we occluded each of the last 360 images with a randomly positioned square of a random intensity (Fig. 6.2(h)). The standard non-robust incremental method included also the squares into the representation, so they appear in the reconstructed images shown in Fig. 6.2(i) as well. On the other hand, the proposed robust incremental method managed to sequentially detect squares as outliers and reconstruct their values before the update. Therefore, the squares were not included into the representation and the reconstructed images (Fig. 6.2(j)) look much more similar to the optimal reconstructions shown in Fig. 6.2(b). Therefore, the robust method significantly outperformed the standard one, as can also be concluded from the mean squared reconstruction errors presented in Table 6.2.

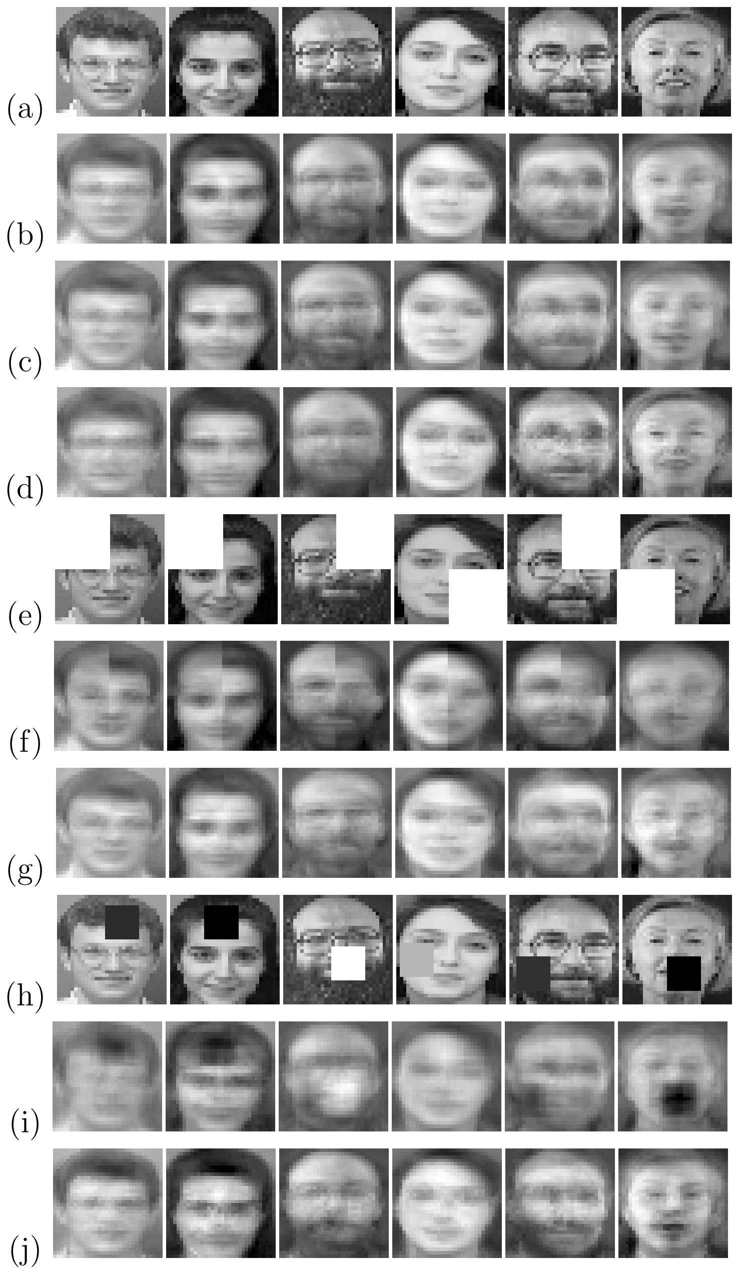


Figure 6.2: (a) Training images. Reconstructions using (b) batch PCA, (c) incremental PCA, (d) weighted IPCA. (e) Training images with missing pixels. Reconstructions using (f) mean substitution and standard IPCA, (g) IPCA with reconstruction of missing pixels. (h) Occluded training images. Reconstructions using (i) standard IPCA, (j) robust IPCA.

	standard	*robust*	*optimal*
missing pixels	760	644	594
occlusions	915	710	594

Table 6.2: Results of standard and robust IPCA.

6.3 Evaluation of all algorithms

In this section we will evaluate the performance of most of the proposed algorithms on three different image domains (objects, faces, and background). We will investigate how the type and the order of input images influence the results.

6.3.1 Image domains

We tested how the algorithms are suitable for modeling objects (in tables and plots referred to as *"obj"*) by building the representations of all twenty objects from COIL20 database. The training set consisted of 36 images of each object; 720 images of the size 32×32 pixels altogether. When the learning was performed in the incremental manner, the training images were input into the algorithms in two different ways. First, the input order was sequential (*seq*); all the images of the first object were first processed, then all the images of the second object and so on. The beginning of the image sequence is shown in Fig. 6.3(a). Then, the input order was alternate (*alt*); the first image of each object was first processed, then the second image of each object and so on (see Fig. 6.3(b)).

We performed similar experiments on the images of faces as well (*face*). For the testbed we used the images from ORL face database, which contains 400 images. Since we wanted to perform all experiments on the same (possibly high) number of images, we made an assumption that faces are symmetric and doubled the number of available images by flipping all the images over the vertical bisector. Then, we took 18 images of each of 40 subjects to form a training set of 720 images resized to the size of 32×32 pixels. The beginning of the training sequences in the sequential and the alternate order are shown in Fig. 6.3(c) and (d), respectively.

Finally, we tested how the algorithms are suited for background modeling (*bckg*). The sequence of 720 images of the size 32×32 pixels with gradually increasing global brightness and non-linear vertical shadow moving across the images was used. The training images were generated in the sequential and alternate order in this case as well. Some of them are shown in Figs. 6.3(e,f).

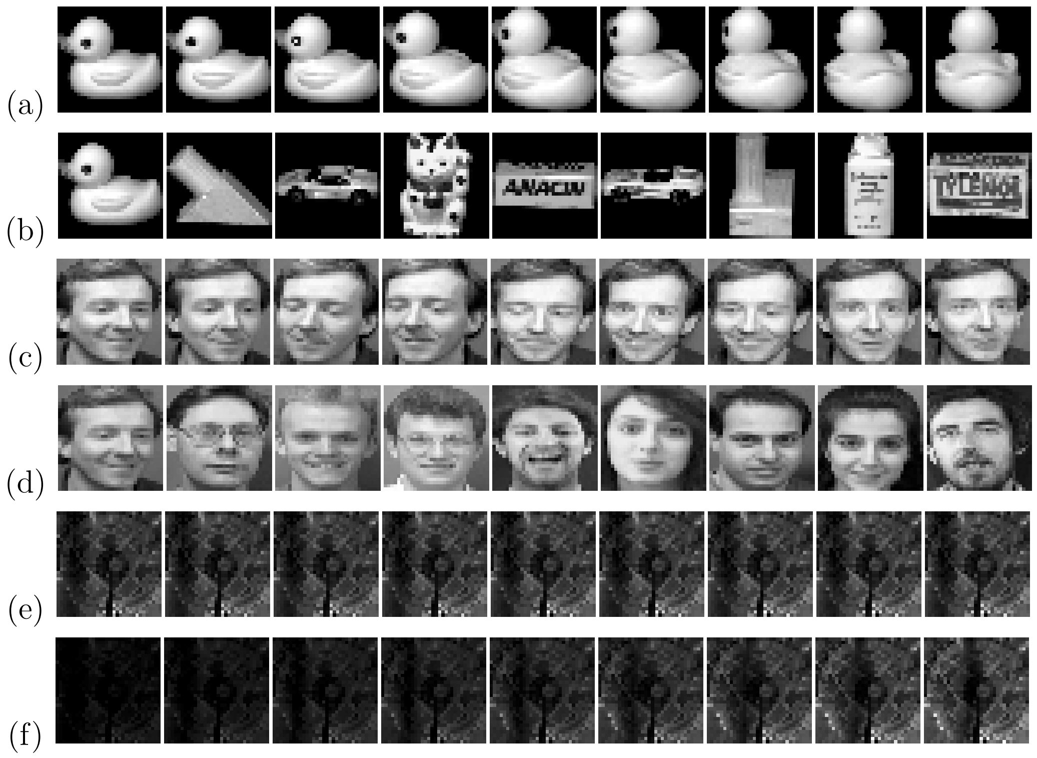


Figure 6.3: First nine images from training sequences: (a,b) objects from COIL database, (c,d) faces from ORL database, (e,f) synthetically illuminated background; (a,c,e) sequential order, (b,d,f) alternate order.

We have chosen these three image domains (objects, faces, and background) because of their characteristic properties. The objects are very different in shape and texture. Even different views of an individual object may vary significantly as well. Therefore, the value of each pixel may considerably vary through the image sequence. Thus, the training images are not very temporally correlated. On the other hand, faces of different subjects do not differ so radically, nor the images of an individual subject captured in slightly different poses. Thus, these images are more temporally correlated. Even greater temporal coherence can be observed in the sequence of the images for the background modeling, where the static background is completely temporally correlated and only the illumination changes through time.

The second interesting property of these three types of images is how gradual the changes in the appearance through the image sequence are, i.e., how similar adjacent images are. In the case of images for the background modeling, the illumination changes gradually, thus the adjacent images are very similar. Also in the case of objects, the changes in appearance of an object are fairly gradual, since the adjacent images were captured from nearby views. These changes are the least gradual in the training sequence for modeling faces, because the individual images of a subject were not captured in a controlled progression of poses. The properties of all three types of images are summarized in Table 6.3.

	objects	faces	background
temporal correlation	low	high	very high
graduality of changes	high	low	very high

Table 6.3: The main characteristics of three types of images.

To test the weighted and the robust algorithms we then added a square of a random intensity on a randomly chosen position on each image, except on the images which appear between the first 40 images in the sequential or in the alternate order. Thus, approximately 10% of images remained non-occluded. Some occluded training images from all three image domains are shown in Fig 6.4.

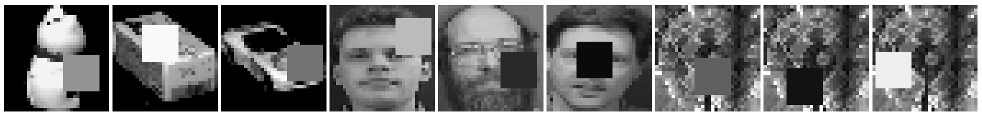


Figure 6.4: Some occluded training images.

6.3.2 Evaluated algorithms

We performed several experiments on all six different input image sequences: the objects in the sequential (*obj_seq*) and the alternate (*obj_alt*) order, the faces in the sequential (*face_seq*) and the alternate (*face_alt*) order and the background images in the sequential (*bckg_seq*) and the alternate (*bckg_alt*) order.

First, the original images without outliers were processed using the standard batch algorithm (*GT_batch*) and the proposed incremental algorithm by discarding (*GT_incA*) and preserving (*GT_incX*) the training images.

In all subsequent experiments, the occluded training images were processed. The goal was to build representations, which do not include occluding squares. For comparison, first the standard non-robust batch (*nr_batch*) and incremental (*nr_incA* and *nr_incX*) methods were applied.

Then, we assumed that the positions of outliers were known and that their deviations from the correct values were approximately known as well. Thus, we applied the algorithms for the batch and the incremental weighted learning. Each algorithm was applied two times using two sets of weights. If we denote the value of a pixel in an original image with x and $\tilde{x}$ denotes the value of the corresponding pixel in the occluded image, then the corresponding weight was set to $w_1 = \frac{255-|x-\tilde{x}|}{255}$ in the first case and to $w_2 = \frac{1}{|x-\tilde{x}|+1}$ in the second case. Thus, in the first case (*W1_batch*, *W1_incA*, and *W1_incX*) the influence of the outlying values was still rather large (proportional to the minus absolute deviation of the pixel values), while in the second case (*W2_batch*, *W2_incA*, and *W2_incX*) the weights were inversely proportioned to the absolute deviation of the pixel values meaning that the influence of the outlying values was significantly reduced.

Next, the outliers were considered as missing pixels. First, an ad-hoc mean substitution was performed by substituting each missing value with the mean value of all corresponding non-missing pixels. The filled training images were then processed using the standard batch (*ms_batch*) and incremental (*ms_incA* and *ms_incX*) methods. Then the proposed methods for learning from incomplete data were applied: the EM algorithm with temporal smoothing (*MP_EMts*), the batch algorithm with iterative reconstruction of missing pixels (*MP_itRec*) and the incremental algorithm for learning from incomplete data (*MP_incA* and *MP_incX*).

Finally, we processed the occluded images with the proposed robust algorithms, which were supposed to detect outliers by themselves. We only indicated the percentage of outliers in each image to enable a fair comparison between the robust algorithms. The one-stage (*Rob_batch1*) and two-stage (*Rob_batch2*) batch algorithms as well as the proposed robust incremental algorithm (*Rob_incA* and *Rob_incX*) were evaluated.

6.3.3 Evaluation results

In all cases the goal was to build a subspace, which encompassed at least 70% of the energy. To achieve this, we retained 11 principal axes for modeling objects, 18 principal axes for modeling faces and 2 principal axes for modeling background. The methods were evaluated with the mean squared reconstruction error, which was calculated by comparing the reconstructed images with the original outlier-free training images.

The errors are presented in Table 6.4. Due to different characteristics of the training images, the reconstruction errors of three different types of images differ significantly. To make the evaluation of the performance clearer, we compare all results with the optimal results, which were obtained using the batch method on the original non-occluded images (*GT_batch*). The percentage of the degradation of the results for each learning approach is presented in Table 6.5.

From these results we will draw some important conclusions about the properties of the proposed methods. To make the explanations clearer we will additionally visualize the results in plots in Fig. 6.6 and Fig. 6.7.

Fig. 6.6(a) depicts the degradation of the results of the incremental method in comparison with the standard batch method. The results for all three types of images and both image orders are shown. As we already established in Section 3.3, better results are obtained when the training images are kept until the end of the learning process (*incX*) than when the obtained coefficients are maintained throughout the learning process and the training images are discarded (*incA*). We also know from previous experiments that the image order plays a significant role. The degradation of the results is larger when the images entered in the incremental learning process in the sequential order. This is the most noticeable in the case of the background modeling, it is less apparent in the case of the object modeling, while the image order influences the results of the modeling of faces the least. This is, as one could expect, completely in accordance with the degree to which changes in the appearance through the image sequence are gradual. We can again observe that the incremental method does not degrade the results a lot. Except in the case when the background was modeled using sequentially ordered input images (which is really an extreme case), all the degradation of the results are fairly below 10%.

When the representations were built from the occluded images, the reconstruction error increased significantly. This can be observed in Fig. 6.6(b), which depicts the reconstruction errors obtained using representations built from the original and the occluded images. The occluded squares were included in the representation, therefore they appear in the reconstructed images as well (see, e.g., Fig. 6.2(i)). The deviation from the optimal reconstruction is thus rather large.

By using the algorithms for weighted learning, we significantly improved the results. Fig. 6.6(c) compares reconstruction errors obtained by using the standard batch method and the weighted method considering two sets of weights. Since additional knowledge was presented to the system, the weighted algorithm produced smaller reconstruction errors. The reconstruction error was considerably smaller when the weights of outliers were set to very small values (*W2*), thus diminishing the influence of the outlying values. Very similar results can be observed in the case of the incremental weighted algorithm as well (see Fig. 6.6(d)). Better results were obtained when the blending step of the weighted incremental algorithm relied more on the values yielded by the model than on the outlying values.

It is evident from Figs. 6.6(e,f) that the image order influences the results also in the case of weighted incremental algorithm and that this influence is larger when the outliers are significantly down-weighted (*W2*). In this case, the new values in the outliers are mainly estimated from the values yielded by the model. If the model roughly encompasses various appearances in the beginning already, it is more capable to reconstruct the values in outliers correctly.

At this point we have to recall that the first 40 images were not occluded. In this way, the incremental methods were able to build at the beginning of the learning process a reliable representation, which was subsequently used for improving the values in outliers. If the initial images were occluded as well, then the incremental methods would not be able to successfully reconstruct the outliers, since no previous reliable knowledge would be available. For that reason, the results would not be as good as in these experiments.

Fig. 6.7(a) depicts the results of the experiments, which considered outlying squares as missing pixels. The results are similar to the results of the weighted methods. Since the positions of outliers were known, they were mainly successfully reconstructed, and excellent representations were built. However, the role of the image order was still very important. In this case it influenced the results of the batch algorithm as well. Since the batch algorithm *MP_EMts* assumes the smoothness prior, i.e., it assumes that adjacent images are similar, significantly worse results were obtained when the input images were aligned in an alternate order, which broke the smoothness assumption. This effect can be observed in Fig. 6.7(b). The effect of the image order on the efficiency of the incremental algorithms is opposite. For the same reasons as in the case of the weighted incremental method, significantly better results were obtained when the images entered into the learning process in the alternate order (see Fig. 6.7(c)).

Finally, we analyze the results of the robust method. The results considerably depend on the properties of input images. The robust modeling of objects was unsuccessful. As depicted in Fig. 6.7(d), the robust algorithms did not improve the results of the non-robust method. The main reason is that the images of the objects are not temporally correlated to the extent, which would enable the reliable detection of outliers. Outliers were detected by checking the consistency of individual pixels through the image sequence. Since the training images were not highly correlated, not only the outliers were non-consistent but also some other pixels in each individual image, which significantly differed from the values of the corresponding pixels in other images, i.e., the pixels on the contour of a rotating object (see Fig. 6.5). Such pixels were considered as outliers and their values were reconstructed from the current model, meaning that no novel information was encoded in the representation. For that reason, the object was not correctly modeled and the reconstruction error was large.

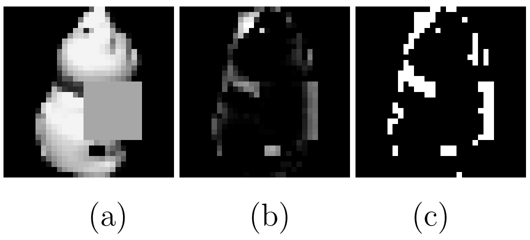


Figure 6.5: (a) Occluded input image, (b) reconstruction error, (c) detected outliers.

Better results were achieved when the robust algorithms were used for building representations of faces (Fig. 6.7(e)). Since face images are more temporally coherent, a significant amount of outliers was successfully detected and not included in the representation. Thus, the reconstructed images mainly did not contain the outlying squares (see, e.g., Fig. 6.2(j)) and the reconstruction error was reduced.

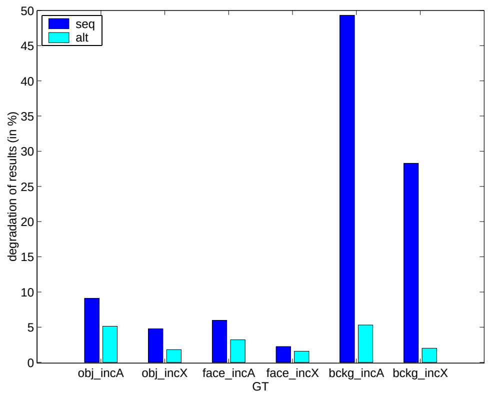
The improvement of the results was even more noticeable in the case of the background modeling. These images are very temporally coherent, therefore the detection of outliers was very reliable and the results were excellent, as can be observed in Fig. 6.7(f). The algorithms indeed succeeded to build a model of the background with varying illumination and without outlying squares. However, also in this case the image order proved to be crucial. As it was already shown in Section 5.5, the results of modeling background with the incremental robust learning algorithm are very poor if the initial eigenspaces do not encompass sufficient variability of background appearances. This condition was perfectly fulfilled when the input images entered into the learning process in the alternate order, which resulted in a rather small reconstruction error.

		objects		faces		background	
		seq	alt	seq	alt	seq	alt
GT	batch	1302.5	1302.5	443.5	443.5	69.7	69.7
	incA	1421.0	1369.4	470.0	457.8	104.0	73.4
	incX	1364.8	1326.2	453.5	450.6	89.4	71.1
nr	batch	2099.3	2099.3	1075.6	1075.6	507.0	507.0
	incA	2176.5	2092.5	1077.5	1068.3	557.8	506.9
	incX	2139.2	2091.9	1071.9	1072.7	517.2	505.8
W1	batch	1669.8	1669.8	908.5	908.5	133.8	133.8
	incA	1731.9	1644.5	835.2	821.7	245.5	119.0
	incX	1713.6	1635.5	866.9	866.7	220.0	119.2
W2	batch	1370.0	1370.0	461.4	461.4	70.2	70.2
	incA	1488.9	1438.5	483.6	470.5	106.5	70.7
	incX	1437.9	1397.3	468.5	464.0	94.8	70.5
ms	batch	1512.2	1512.2	509.3	509.3	90.1	90.1
	incA	1633.9	1577.1	532.1	520.6	120.7	93.7
	incX	1582.4	1540.5	519.2	514.9	106.7	91.5
MP	EMts	1396.7	1800.9	509.7	620.6	70.2	106.5
	itRec	1374.6	1374.6	457.4	457.4	70.1	70.1
	incA	1499.4	1445.8	485.2	471.8	106.7	70.7
	incX	1449.5	1406.2	469.9	465.2	95.2	70.5
Rob	batch1	2260.6	2260.6	769.6	769.6	122.1	122.1
	batch2	2330.4	2330.4	1124.4	1124.4	114.5	114.5
	incA	2285.5	2127.2	683.6	677.4	6456.7	103.5
	incX	2345.9	2186.0	825.1	830.3	2588.9	100.5

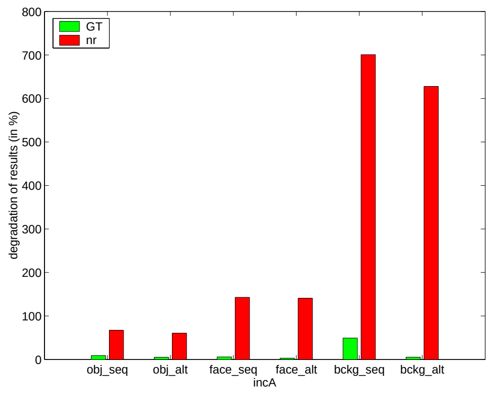
Table 6.4: Mean squared reconstruction errors obtained by modeling three types of training images (objects (*obj*), faces (*face*) and background (*bckg*)) given in two different orders (sequential (*seq*) and alternate (*alt*)) using various learning approaches: learning from original non-occluded images (GT), learning from occluded images using the standard non-robust approach (nr), learning considering known weights (*W1* and *W2*), learning by treating outliers as missing pixels and using the simple mean substitution (ms) technique and using the proposed algorithms for learning from incomplete data (*MP*), and learning using proposed robust methods (*Rob*).

		objects		faces		background	
		seq	alt	seq	alt	seq	alt
GT	batch	0.00	0.00	0.00	0.00	0.00	0.00
	incA	9.10	5.13	5.98	3.22	49.32	5.31
	incX	4.78	1.82	2.26	1.59	28.28	2.01
nr	batch	61.17	61.17	142.52	142.52	627.70	627.70
	incA	67.10	60.65	142.95	140.87	700.60	627.56
	incX	64.24	60.60	141.68	141.85	642.37	625.91
W1	batch	28.20	28.20	104.84	104.84	92.01	92.01
	incA	32.96	26.25	88.32	85.26	252.41	70.84
	incX	31.56	25.56	95.46	95.42	215.72	71.05
W2	batch	5.18	5.18	4.02	4.02	0.77	0.77
	incA	14.31	10.44	9.03	6.08	52.86	1.50
	incX	10.39	7.28	5.63	4.61	36.09	1.14
ms	batch	16.10	16.10	14.83	14.83	29.32	29.32
	incA	25.44	21.08	19.98	17.39	73.23	34.42
	incX	21.49	18.27	17.06	16.08	53.07	31.34
MP	EMts	7.23	38.26	14.92	39.93	0.77	52.78
	itRec	5.53	5.53	3.12	3.12	0.54	0.54
	incA	15.11	11.00	9.38	6.37	53.15	1.49
	incX	11.29	7.96	5.94	4.89	36.57	1.13
Rob	batch1	78.91	78.91	153.50	153.50	64.33	64.33
	batch2	73.56	73.56	73.52	73.52	75.31	75.31
	incA	75.47	63.31	54.12	52.72	9166.98	48.50
	incX	80.10	67.83	86.02	87.21	3615.65	44.26

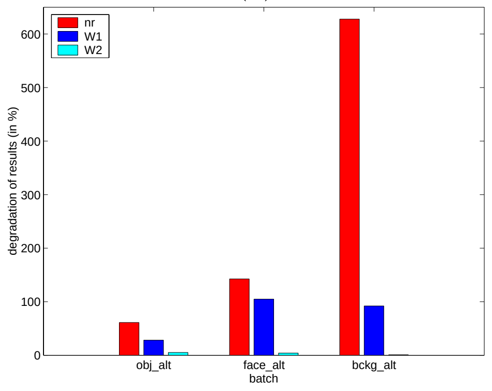
Table 6.5: Degradation of results (in %) of modeling three types of training images (objects (*obj*), faces (*face*) and background (*bckg*)) given in two different orders (sequential (*seq*) and alternate (*alt*)) using various learning approaches: learning from original non-occluded images (*GT*), learning from occluded images using the standard non-robust approach (*nr*), learning considering known weights (*W1* and *W2*), learning by treating outliers as missing pixels and using the simple mean substitution (*ms*) technique and using the proposed algorithms for learning from incomplete data (*MP*), and learning using proposed robust methods (*Rob*).



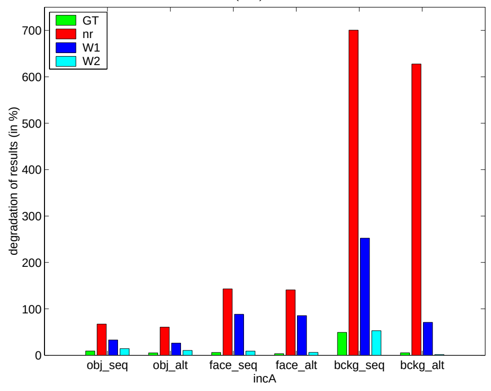
(a)



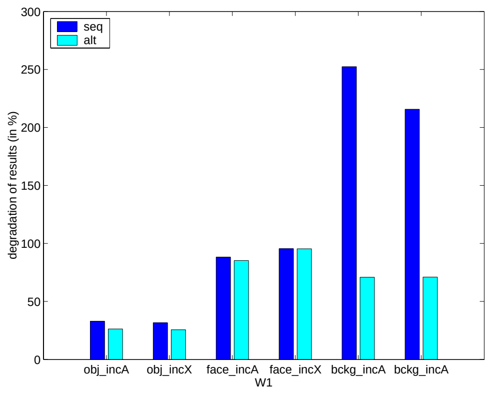
(b)



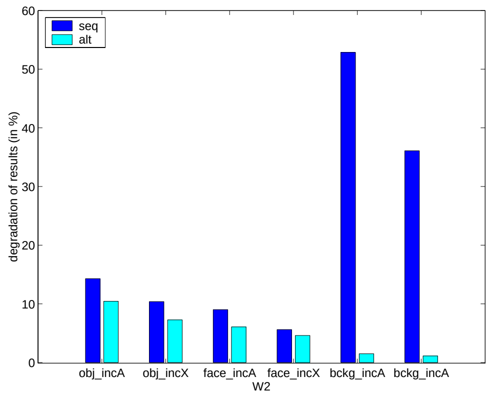
(c)



(d)



(e)



(f)

Figure 6.6: Evaluation of incremental and weighted methods. (a) Influence of the image order on the results of the incremental method. (b) Degradation of the results when outliers are added. Improvements of the results using weighted (c) batch and (d) incremental methods. (e,f) Influence of the image order on the results of the weighted incremental method.

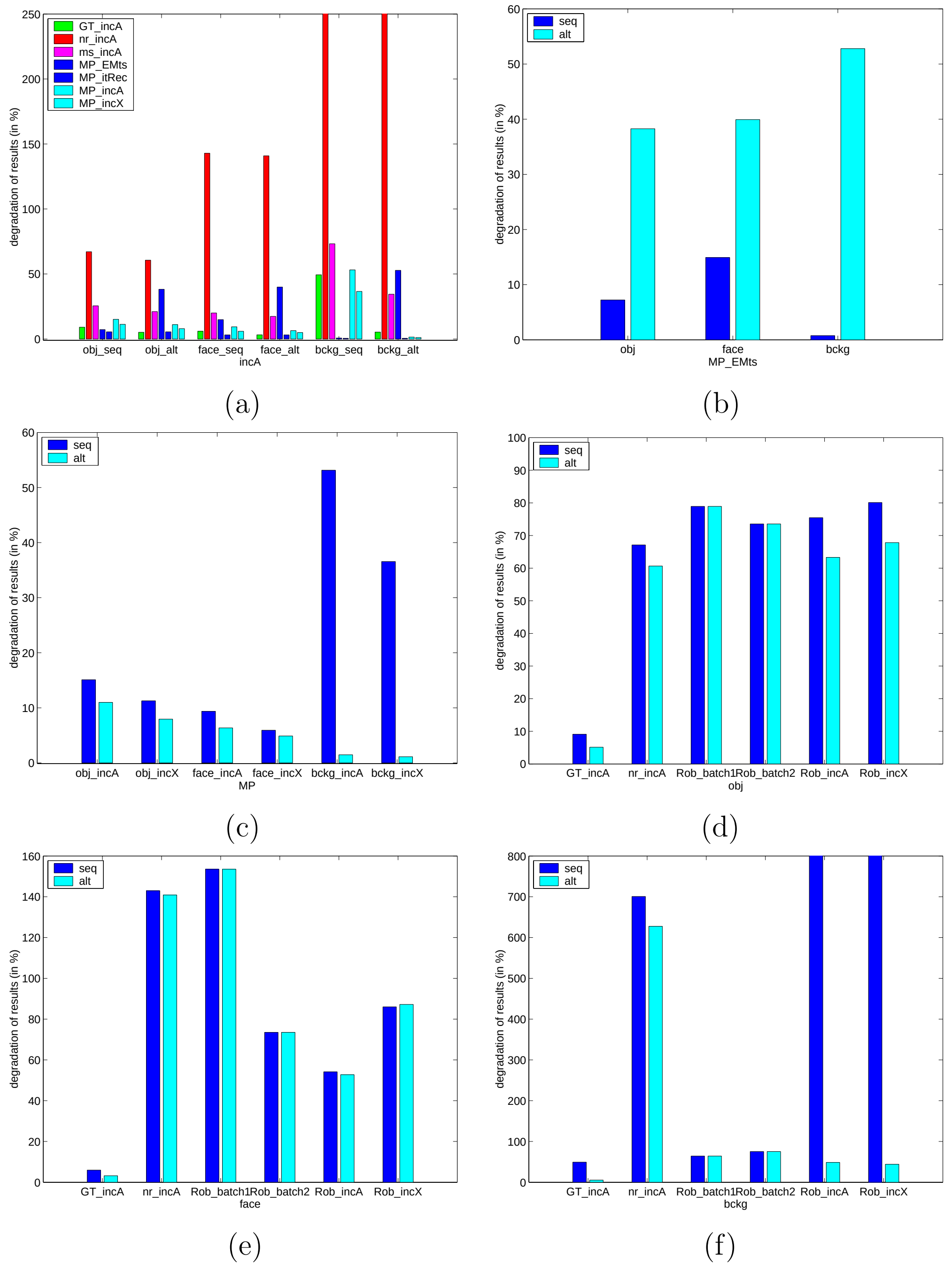


Figure 6.7: Evaluation of the methods for learning from incomplete data and robust methods. (a) Comparison of several algorithms, which consider missing pixels. Influence of the image order on the results of (b) the method *MP_EMts* and (c) the incremental *MP* method. Comparison of robust algorithms on different image domains: (d) objects, (e) faces, and (f) background.

6.3.4 Conclusions

Several conclusions can be drawn from the results of the performance evaluation. First of all, in all experiments the image order proved to be important to achieve good results with incremental methods. This was evident for all incremental algorithms, however this was more pronounced when the weighted incremental algorithm was used, and even more in the case of the robust incremental algorithm. In these experiments the alternate image order was rather optimal to assure diverse images at the beginning of the training sequence. In practice, such ideal situations are less likely to occur, but they are usually not necessary. Nevertheless, to enable efficient incremental learning, especially by using the robust approach, one has to assure that the visual variability of the entire image sequence is encompassed in the initial images already at least at a very coarse level.

From the results we can also conclude that the proposed incremental and weighted approaches as well as the algorithms for learning from incomplete data perform very well. In the robust framework this means that the algorithms are capable to correctly reconstruct outlying values given that their positions in images are known. If the positions of outliers are not known, they have to be detected using the robust algorithms. When the input images are not highly correlated, as in the case of images of several objects, the input sequence does not contain enough information for the reliable detection of outliers using the reconstruction error only. For that reason the robust learning algorithm does not perform well. In such cases an alternative algorithm for outlier detection should be used or an additional knowledge about outliers obtained from higher-level processes should be utilized to detect outliers more reliably. The results could also be improved by enhancing the redundancy in the data by increasing the number of input images or by splitting the training image set to several temporally more coherent sequences. Once the outliers are detected more reliably, the results would improve significantly.

Such alternative measures are not necessarily required for the robust modeling of temporally more coherent image sequences, such as faces or background images. In these cases the outliers can be detected by considering reconstruction error, which suffice to achieve a good performance of the robust algorithms.

The main conclusions about the performance of the proposed algorithms with regard to different image domains are summarized in Table 6.6.

	objects	faces	background
significance of image order	high	low	very high
performance of incremental method	very high	very high	very high
performance of weighted methods	high	high	very high
performance of robust methods	very low	high	very high

Table 6.6: Characteristic results on three types of images.

6.4 Chapter summary

In this chapter we presented some additional experimental results, which give a global view to all algorithms. First, we illustrated the performance of the incremental algorithms for building the representations of faces. Then we evaluated most of the proposed algorithms on three image domains with different characteristics. We analyzed the influence of the image order on the results as well as the influence of different properties of input images. From the experimental results we drew some conclusions about applicability of the proposed algorithms.

Chapter 7

Summary and conclusions

7.1 Dissertation summary

The appearance-based approach to visual learning and recognition has been an active area of research in the recent years. Using this approach, objects are modeled as a set of views captured during a systematic observation of each object. Recognition is then performed by directly matching an unknown image with the stored training images.

This approach of direct matching would be prohibitive (computationally and in terms of space) unless the views were compressed in a compact representation and the matching was performed in an efficient way. The most common approach is to compress the set of views in an eigenspace representation built from all images of objects using principal component analysis. Recognition of an unknown object is then performed by projecting its image into the eigenspace and finding the nearest projected training image.

The main advantage of the appearance-based approach is that it does not require 3-D reconstruction of objects. It uses raw image data directly without any extraction of geometric features. Learning and recognition can thus be performed by using simple and well known methods from linear algebra. For that reason several successful applications have been developed ranging from object and face recognition to background modeling.

However, this simplicity poses several limitations, such as the combinatorial explosion of the number of required training images, poor across-class generalization, shift and scale sensitivity and non-robustness to non-gaussian noise. In this dissertation we focused on the specific problems regarding incremental, weighted and robust learning.

The standard PCA approach is usually performed in a batch mode, i.e., all training images are processed simultaneously, which means that all of them have to be given in advance. In this dissertation we proposed a method for incremental learning, which processes images sequentially one by one and updates the principal subspace accordingly. Although several methods for incremental learning have already been proposed, our approach is the most suitable for the extension to a weighted and robust method.

The next shortcoming of the standard PCA approach is that it treats all pixels of an image equally. Also, all training images have equal influence on the estimation of principal axes. This is not well suited for applications where some images or some parts of images should be considered more or less important than other. In this dissertation, we presented a generalized PCA approach, which estimates principal axes and principal components considering weighted pixels and images, enabling selective influence of training images as well as pixels in individual images to the process of learning. We considered also the special case of the weighted learning, where some pixels are totally unreliable and proposed batch and incremental methods for learning from incomplete data.

And finally, we put the learning algorithms into a robust framework. Since the images of objects and scenes are not always ideal and as such they may contain noise or occlusions, a robust approach is required. Although many methods for robust recognition have already been proposed, robust learning has been tackled very rarely. Thus, we proposed the robust methods for eigenspace learning, which detect outliers and estimate the principal subspace from the consistent data only.

All the proposed methods were experimentally evaluated. The experimental results were used in the discussion about properties of the proposed methods and their applicability in different scenarios.

7.2 Contributions of dissertation

The main goal of this dissertation was to develop methods, which would strengthen the subspace approach to visual learning and recognition under non-ideal conditions. The emphasis was on the incremental, weighted, and robust learning of eigenspaces. We studied related problems and theoretically derived suitable methods, which tend to overcome the drawbacks of the standard PCA approach.

We proposed a novel **incremental approach**, which enables continuous learning. We also presented a **generalized weighted PCA approach**, which considers individual images and pixels selectively. We also developed a **robust approach**, which aims to build consistent representations even in non-ideal training conditions. Finally, we embedded weighted and robust methods in the incremental framework to enable the weighted and robust learning in an evolving environment as well.

For all these approaches we developed appropriate algorithms. To summarize the contributions of the dissertation, we list the proposed algorithms and highlight their main characteristics:

- We proposed an algorithm for **incremental learning**. It maintains low-dimensional representations of the previously learned images throughout the entire learning stage, meaning that each training image can be discarded immediately after the update. Furthermore, it is able to treat different images differently, which enables to advance it into a weighted incremental method.

- We presented two algorithms for **batch weighted learning**. The first one considers only temporal weights and is based on the weighted covariance matrix, while the other, a modified EM algorithm, considers general weights. The latter algorithm was then adapted for **batch learning from incomplete data**. Two algorithms were proposed; one algorithm constrains the reconstructed values in missing pixels considering a smoothness prior, the other is based on iterative reconstruction of missing pixels.

- The incremental algorithm was also extended into the algorithm for **incremental weighted learning**, which considers arbitrary temporal and spatial weights. A reduced version of this algorithm adapted for **incremental learning from incomplete data** was also presented.

- Next, we proposed the algorithm for **batch robust learning**, which iteratively detects outliers in training images, reconstructs their values and estimates the robust principal subspace. Since the initial step of this algorithm is still non-robust, a **robust subsampling-based initialization** was also proposed.

- And finally, the robust approach was augmented in the incremental method, resulting in an algorithm for **robust incremental learning**. It sequentially updates the representation using the previously acquired knowledge for determining consistencies and discarding inconsistencies in the input images.

We experimentally evaluated all the proposed algorithms on **three image domains** with different characteristics and determined the applicability of the methods in different scenarios.

7.3 Future work

In this dissertation we proposed several algorithms, which overcome some deficiencies of the standard eigenspace approach to the appearance-based modeling. However, many open problems still remain to be solved. In this very final section of the dissertation, we will identify some of these open problems and try to give directions for their solutions.

The proposed robust methods, especially the incremental version, do not prove effective when training images are not highly temporally correlated. In such cases the outlier detection is less reliable since it is very difficult to differentiate between the inconsistencies in the new images, which are result of the object rotation and the inconsistencies due to occlusions or other outliers. One way to alleviate this problem is to treat outliers, which are detected on the boundary of the object, differently, since it is very likely, that those pixels are inconsistent with the previous pixel values due to rotation of the object and are not outliers.

We can approach this problem also by increasing the number of training images by, for example, processing a video sequence. In this case, the appearance changes very smoothly, the redundancy in the data is larger, the images are more temporally correlated and the detection of outliers is more reliable. By using the incremental algorithm for eigenspace learning such an approach is feasible in terms of space as well as in terms of computational requirements.

Outliers are detected by checking the consistencies over a sequence of images, thus analyzing the correlation through the time. However, one could also check consistency in other directions of a spatio-temporal volume. This is especially suitable when the correlation across rows or columns of the training images is large. Furthermore, the consistency could also be checked using smaller spatio-temporal sub-volumes to explore local properties of the data. Such approach would make the detection of outliers more robust and reliable.

The temporal correlation can also be increased by registering training images. In the case of face recognition, the registration can easily be accomplished, since all training and test images are registered to the same template [9]. It is not obvious, however, how this registration can be performed in the case of recognition of general objects. Nevertheless, to bound the number of required training images and to improve the results of the robust algorithm, a kind of registration, at least at a very coarse level, should be performed.

In this dissertation PCA approach was always used for processing grey level images. It can be used, however, for processing images of other modalities as well. Moreover, it can be applied to any set of vectors that are temporally correlated. Instead of images, feature vectors obtained using various methods for feature extraction, can be used. All the algorithms, which were developed in this dissertation, can be applied on such data domains as well. In this way, a pure holistic view-based approach can be extended by exploiting local geometric features and structural information. In fact, current research trends have been taking this approach, which has a potential to overcome many drawbacks of view-based methods enabling efficient visual learning and recognition.

Appendix A

Notation

In this dissertation we use the following notation:

- Scalars are denoted with the italic typeface (e.g., M, i).
- Vectors are denoted with small letters in the bold typeface (e.g., $\mathbf{a}$).
- Matrices are denoted with capital letters in the bold typeface (e.g., $\mathbf{A}$).
- The elements of a vector $\mathbf{a}$ are denoted as a_i, thus $\mathbf{a} = [a_1, a_2, \ldots, a_N]^\top$.
- The elements of a matrix $\mathbf{A}$ are denoted as a_{ij}, thus
$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \ldots & a_{MN} \end{bmatrix} .$$
- The columns of a matrix $\mathbf{A}$ are denoted as vectors $\mathbf{a}_i$, thus $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N]$.
- The rows of a matrix $\mathbf{A}$ are denoted as row vectors $\mathbf{a}_{i:}$, thus
$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{1:} \\ \mathbf{a}_{2:} \\ \vdots \\ \mathbf{a}_{M:} \end{bmatrix} .$$
- $\mathbf{1}_{M \times N}$ denotes a matrix of the dimension $M \times N$, where every element equals 1.

In the situations where the value of a variable is time dependent, the superscript denotes the time instant (step) which the value is related to. E.g., $\mathbf{A}^{(n)}$ denotes the values of $\mathbf{A}$ at the step n.
The following list enumerates some frequently used variables and their typical dimensions:

- M – number of pixels in image
- N – number of training images
- k – dimension of eigenspace
- $\mathbf{x} \in \mathbb{R}^M$ – image
- $\mathbf{X} \in \mathbb{R}^{M \times N}$ – input data matrix (training images)
- $\boldsymbol{\mu} \in \mathbb{R}^M$ – mean image
- $\hat{\mathbf{X}} \in \mathbb{R}^{M \times N}$ – mean centered input data matrix
- $\mathbf{U} \in \mathbb{R}^{M \times k}$ – matrix of principal vectors
- $\mathbf{A} \in \mathbb{R}^{k \times N}$ – matrix of coefficient vectors
- $\mathbf{Y} \in \mathbb{R}^{M \times N}$ – matrix of reconstructed training images
- $\boldsymbol{\lambda} \in \mathbb{R}^N$ – vector containing eigenvalues
- $\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$ – diagonal matrix containing eigenvalues on diagonal
- $\mathbf{C}$ – covariance matrix
- $\mathbf{C}'$ – inner product matrix
- $\mathbf{W} \in \mathbb{R}^{M \times N}$ – weight matrix
- ${}^{\mathrm{t}}\mathbf{w} \in \mathbb{R}^{1 \times N}$ – temporal weights
- ${}^{\mathrm{s}}\mathbf{w} \in \mathbb{R}^M$ – spatial weights

A special notation for dealing with missing pixels:

- $\mathcal{I}_j^{\bullet}$ – set of indices of non-missing pixels in j-th image
- $\mathcal{I}_j^{\circ}$ – set of indices of missing pixels in j-th image

- $\mathcal{I}_{i:}^{\bullet}$ – set of indices of non-missing pixels in i-th row of data matrix $\mathbf{X}$
- $\mathcal{I}_{i:}^{\circ}$ – set of indices of missing pixels in i-th row of data matrix $\mathbf{X}$
- $\hat{\mathbf{x}}_j^{\bullet} \in \mathrm{I\!R}^{M^{\bullet}}$ – subvector of non-missing pixels in image $\hat{\mathbf{x}}_j$
- $\mathbf{U}^{\bullet} \in \mathrm{I\!R}^{M^{\bullet} \times N}$ – rows of $\mathbf{U}$, which correspond to $\hat{\mathbf{x}}_j^{\bullet}$
- $\hat{\mathbf{x}}_j^{\circ} \in \mathrm{I\!R}^{M^{\circ}}$ – subvector of missing pixels in image $\hat{\mathbf{x}}_j$
- $\mathbf{U}^{\circ} \in \mathrm{I\!R}^{M^{\circ} \times N}$ – rows of $\mathbf{U}$, which correspond to $\hat{\mathbf{x}}_j^{\circ}$
- $\hat{\mathbf{x}}_{i:}^{\bullet} \in \mathrm{I\!R}^{1 \times N^{\bullet}}$ – row subvector of non-missing pixels in row vector $\hat{\mathbf{x}}_{i:}$
- $\mathbf{A}^{\bullet} \in \mathrm{I\!R}^{k \times N^{\bullet}}$ – columns of $\mathbf{A}$, which correspond to $\hat{\mathbf{x}}_{i:}^{\bullet}$
- $\hat{\mathbf{x}}_{i:}^{\circ} \in \mathrm{I\!R}^{1 \times N^{\circ}}$ – row subvector of missing pixels in row vector $\hat{\mathbf{x}}_{i:}$
- $\mathbf{A}^{\circ} \in \mathrm{I\!R}^{k \times N^{\circ}}$ – columns of $\mathbf{A}$, which correspond to $\hat{\mathbf{x}}_{i:}^{\circ}$

Some non-standard operators:

- $\langle \mathbf{a}, \mathbf{b} \rangle$ – dot product between two vectors of equal dimension
- $\mathbf{A}^{\dagger}$ – pseudoinverse
- $.\sqrt{\mathbf{A}}$ – point wise square root (calculates the square root of each element of the matrix)
- $\mathbf{A} \circ \mathbf{B}$ – point wise (Hadamard) product between two matrices of equal dimension

The following expressions are mainly used as synonyms:

- principal subspace = eigenspace
- principal axes = principal vectors = principal directions = eigenvectors
- principal components = coefficients
- vector of principal components = coefficient vector = projection

Appendix B

Related topics from linear algebra

B.1 Solving system of linear equations and pseudoinverse

In this appendix we will show the relation between the solution of an overdetermined system of linear equations and the pseudoinverse [26].

Let us suppose that we want to solve the following system of M equations with N unknowns:

$$\sum_{j=1}^{N} a_j x_{ij} = b_i \ ; \ \ i = 1 \dots M \ , \tag{B.1}$$

where x_{ij} and b_i are known and a_j unknown variables. This problem can be conveniently rewritten in the matrix notation. We are looking for a vector $\mathbf{a}$ that minimizes the sum of the squared errors

$$E = \|\mathbf{X}\mathbf{a} - \mathbf{b}\|^2 = \sum_{i=1}^{M} (\mathbf{x}_{i:}\mathbf{a} - b_i)^2 \ . \tag{B.2}$$

By setting the gradient of the error to zero

$$\nabla E = \sum_{i=1}^{M} 2(\mathbf{x}_{i:}\mathbf{a} - b_i)\mathbf{x}_{i:} = 2\mathbf{X}^\top(\mathbf{X}\mathbf{a} - \mathbf{b}) = \mathbf{0} \tag{B.3}$$

we obtain the following equation:

$$\mathbf{X}^\top\mathbf{X}\mathbf{a} = \mathbf{X}^\top\mathbf{b} \ . \tag{B.4}$$

$\mathbf{X}^\top\mathbf{X}$ is a square matrix and if it is nonsingular, we can solve for $\mathbf{a}$ uniquely as

$$\mathbf{a} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{b} = \mathbf{X}^\dagger\mathbf{b} \ . \tag{B.5}$$

Here, the $N \times M$ matrix

$$\mathbf{X}^\dagger = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \tag{B.6}$$

is called the *pseudoinverse* of $\mathbf{X}$.

Therefore, an overdetermined system of linear equations (B.1) can be solved in the least squares sense using the pseudoinverse (B.6).

B.2 Eigenvalue decomposition and singular value decomposition

In many places in this dissertation we perform eigenvalue decomposition (EVD) of a matrix. Here, we will show that for symmetric real matrices eigenvalue decomposition is equivalent to singular value decomposition (SVD) [58, 46].

Let us assume that $\mathbf{A} \in \mathbb{R}^{M \times M}$ is a square matrix. If for some vector $\mathbf{e}$ and a scalar λ holds that

$$\mathbf{A}\mathbf{e} = \lambda \mathbf{e} \ , \tag{B.7}$$

then $\mathbf{e}$ is called the *eigenvector* and λ is the *eigenvalue* of $\mathbf{A}$. Let $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_M]$ be composed from all M eigenvectors of $\mathbf{A}$ and let $\mathbf{\Lambda}$ be a diagonal matrix with the corresponding eigenvalues $\lambda_i, \ \ i = 1 \ldots M$ on the diagonal. Then it holds that

$$\mathbf{A}\mathbf{E} = \mathbf{E}\mathbf{\Lambda} \ . \tag{B.8}$$

If $\mathbf{E}$ is invertible we can rewrite (B.8) as

$$\mathbf{A} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1} \ . \tag{B.9}$$

The diagonalization in (B.9) is known as *eigenvalue decomposition* of $\mathbf{A}$.

In this regard, symmetric real matrices have some very nice properties:

- Any two eigenvectors belonging to different eigenvalues are orthogonal.
- All eigenvalues are real.
- Every symmetric matrix can be diagonalized according to (B.9).
- If a real symmetric matrix is of the form $\mathbf{B}\mathbf{B}^\top$, all its eigenvalues are nonnegative.

The other very useful and common matrix decomposition is *singular value decomposition.* SVD decomposes a matrix $\mathbf{A}$ in the following way:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top \ , \tag{B.10}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and $\mathbf{D}$ is a diagonal matrix. If $\mathbf{A}$ is symmetric, thus $\mathbf{A}^\top = \mathbf{A}$, we can rewrite (B.10) as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^\top \ . \tag{B.11}$$

Since $\mathbf{U}$ is orthonormal, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$, thus $\mathbf{U}^\top = \mathbf{U}^{-1}$. Therefore, we can rewrite (B.11) as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1} \ , \tag{B.12}$$

which is equivalent to (B.9). Thus, eigenvalue decomposition and singular value decomposition of a real symmetric matrix are equivalent.

If the matrix $\mathbf{A} \in \mathbb{R}^{M\times M}$ is a square matrix, then it is decomposed to the matrices of the same size $\mathbf{U} \in \mathbb{R}^{M\times M}$ and $\mathbf{D} \in \mathbb{R}^{M\times M}$. However, if it is of the form $\mathbf{A} = \mathbf{B}\mathbf{B}^\top$, where $\mathbf{B} \in \mathbb{R}^{M\times N}$ and $N < M$, then the rank of $\mathbf{A}$ is less or equal to N and the matrix $\mathbf{A}$ has at most N non-zero eigenvalues. Hence, by performing the SVD (B.11), we can decompose the matrix $\mathbf{A} \in \mathbb{R}^{M\times M}$ on a smaller orthonormal matrix $\mathbf{U} \in \mathbb{R}^{M\times N}$ and a diagonal matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$.

Appendix C

Comparison with related work

C.1 Incremental PCA

In this section we will prove that the method for incremental PCA proposed in Chapter 3 estimates an identical subspace as the algorithm introduced by Hall et al. [32]. The core of both algorithms is SVD of a covariance matrix, which is, however, obtained in different ways. To prove the equivalence of two algorithms we will show that the covariance matrix is equal in both cases.

Our algorithm performs PCA on $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ by applying SVD on its covariance matrix. To make the derivation of the proof of the equivalence between both algorithms easier, we will rewrite the covariance matrix of $\mathbf{A}'$ in the following form:

$$\mathbf{C} = \frac{1}{n+1}\mathbf{A}'\mathbf{A}'^\top - \boldsymbol{\mu}''\boldsymbol{\mu}''^\top \ .$$

Now, we will derive the product $\mathbf{A}'\mathbf{A}'^\top$:

$$\mathbf{A}'\mathbf{A}'^\top = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix} \begin{bmatrix} \mathbf{A}^{(n)\top} & \mathbf{0} \\ \mathbf{a}^\top & \|\mathbf{r}\| \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{(n)}\mathbf{A}^{(n)\top} + \mathbf{a}\mathbf{a}^\top & \|\mathbf{r}\|\mathbf{a} \\ \|\mathbf{r}\|\mathbf{a}^\top & \|\mathbf{r}\|^2 \end{bmatrix} \ .$$

Since the coefficients $\mathbf{A}^{(n)}$ are mean centered, $\sum_{j=1}^{n}\mathbf{a}_j = 0$, and

$$\boldsymbol{\mu}'' = \frac{1}{n+1}\sum_{j=1}^{n+1}\mathbf{a}'_j = \frac{1}{n+1}\mathbf{a}'_{n+1} = \frac{1}{n+1}\begin{bmatrix} \mathbf{a} \\ \|\mathbf{r}\| \end{bmatrix} \ ,$$

thus

$$\boldsymbol{\mu}''\boldsymbol{\mu}''^\top = \frac{1}{(n+1)^2}\begin{bmatrix} \mathbf{a} \\ \|\mathbf{r}\| \end{bmatrix}\begin{bmatrix} \mathbf{a}^\top & \|\mathbf{r}\| \end{bmatrix} = \frac{1}{(n+1)^2}\begin{bmatrix} \mathbf{a}\mathbf{a}^\top & \|\mathbf{r}\|\mathbf{a} \\ \|\mathbf{r}\|\mathbf{a}^\top & \|\mathbf{r}\|^2 \end{bmatrix} \ .$$

Now we can rewrite $\mathbf{C}$ as

$$\begin{aligned}\mathbf{C} &= \frac{1}{n+1}\begin{bmatrix}\mathbf{A}^{(n)}\mathbf{A}^{(n)\top}+\mathbf{a}\mathbf{a}^\top & \|\mathbf{r}\|\mathbf{a}\\ \|\mathbf{r}\|\mathbf{a}^\top & \|\mathbf{r}\|^2\end{bmatrix} - \frac{1}{(n+1)^2}\begin{bmatrix}\mathbf{a}\mathbf{a}^\top & \|\mathbf{r}\|\mathbf{a}\\ \|\mathbf{r}\|\mathbf{a}^\top & \|\mathbf{r}\|^2\end{bmatrix} = \\ &= \frac{n}{n+1}\begin{bmatrix}\frac{1}{n}\mathbf{A}^{(n)}\mathbf{A}^{(n)\top} & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix} + \frac{n}{(n+1)^2}\begin{bmatrix}\mathbf{a}\mathbf{a}^\top & \|\mathbf{r}\|\mathbf{a}\\ \|\mathbf{r}\|\mathbf{a}^\top & \|\mathbf{r}\|^2\end{bmatrix} .\end{aligned}$$

The main difference of the Hall's method is that it passes eigenvalues between the learning steps instead of coefficients. Let $\mathbf{\Lambda}$ be a diagonal matrix containing eigenvalues on its diagonal and $\mathbf{h}$ a unit residue vector $\mathbf{h} = \frac{\mathbf{r}}{\|\mathbf{r}\|}$. Then, if we define

$$\gamma = \mathbf{h}^\top(\mathbf{x} - \boldsymbol{\mu}^{(n)}) \ ,$$

the matrix $\mathbf{C}$ in Hall's method is composed from matrices

$$\mathbf{B}_1 = \begin{bmatrix}\mathbf{\Lambda} & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix} \text{ and } \mathbf{B}_2 = \begin{bmatrix}\mathbf{a}\mathbf{a}^\top & \gamma\mathbf{a}\\ \gamma\mathbf{a}^\top & \gamma^2\end{bmatrix}$$

in the following way:

$$\mathbf{C} = \frac{n}{n+1}\mathbf{B}_1 + \frac{n}{(n+1)^2}\mathbf{B}_2 \ .$$

If we compose $\mathbf{C}$ from submatrices in the following way:

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_1+\mathbf{C}_2 & \mathbf{c}_3\\ \mathbf{c}_3^\top & c_4\end{bmatrix} \ ,$$

we obtain the following expressions for our and Hall's method:

$$\begin{aligned}\mathbf{C}_{1our} &= \frac{n}{n+1}\frac{1}{n}\mathbf{A}^{(n)}\mathbf{A}^{(n)\top}\\ \mathbf{C}_{2our} &= \frac{n}{n+1}\mathbf{a}\mathbf{a}^\top\\ \mathbf{c}_{3our} &= \frac{n}{(n+1)^2}\|\mathbf{r}\|\mathbf{a}\\ c_{4our} &= \frac{n}{(n+1)^2}\|\mathbf{r}\|^2\\ \mathbf{C}_{1hall} &= \frac{n}{n+1}\mathbf{\Lambda}^{(n)}\\ \mathbf{C}_{2hall} &= \frac{n}{n+1}\mathbf{a}\mathbf{a}^\top\\ \mathbf{c}_{3hall} &= \frac{n}{(n+1)^2}\gamma\mathbf{a}\\ c_{4hall} &= \frac{n}{(n+1)^2}\gamma^2 \ .\end{aligned}$$

Sice the coefficients are decorrelated, i.e., $\mathbf{a}_{i:}^{(n)}\mathbf{a}_{j:}^{(n)\top} = 0$ if $i \neq j$, and the eigenvalues are equal to the variances in the directions of the principal axes, i.e., $\frac{1}{n}\mathbf{a}_{i:}^{(n)}\mathbf{a}_{i:}^{(n)\top} = \lambda_i^{(n)}$, we can write

$$\mathbf{C}_{1our} = \frac{n}{n+1}\frac{1}{n}\mathbf{A}^{(n)}\mathbf{A}^{(n)\top} = \frac{n}{n+1}\mathbf{\Lambda}^{(n)} = \mathbf{C}_{1hall} \ .$$

The pair of second equations is already equal:

$$\mathbf{C}_{2our} = \frac{n}{n+1}\mathbf{a}\mathbf{a}^\top = \mathbf{C}_{2hall} \ .$$

Now we will derive the value of γ in the Hall's method. Let $\hat{\mathbf{y}}$ be the reconstruction of $\mathbf{a}$. Then $\mathbf{x} - \boldsymbol{\mu}^{(n)} = \mathbf{r} + \hat{\mathbf{y}}$ and $\mathbf{r}$ is orthogonal to $\hat{\mathbf{y}}$. It follows that

$$\gamma = \mathbf{h}^\top(\mathbf{x} - \boldsymbol{\mu}^{(n)}) = \frac{\mathbf{r}^\top}{\|\mathbf{r}\|}(\mathbf{x} - \boldsymbol{\mu}^{(n)}) = \frac{\mathbf{r}^\top}{\|\mathbf{r}\|}(\mathbf{r} + \hat{\mathbf{y}}) = \frac{\mathbf{r}^\top\mathbf{r}}{\|\mathbf{r}\|} + \frac{\mathbf{r}^\top\hat{\mathbf{y}}}{\|\mathbf{r}\|} = \frac{\|\mathbf{r}\|^2}{\|\mathbf{r}\|} = \|\mathbf{r}\| \ .$$

Thus, the value of γ is equal to the length of the residual vector $\mathbf{r}$. Now, it is trivial to show that

$$\mathbf{c}_{3hall} = \frac{n}{(n+1)^2}\gamma\mathbf{a} = \frac{n}{(n+1)^2}\|\mathbf{r}\|\mathbf{a} = \mathbf{c}_{3our}$$

and

$$c_{4hall} = \frac{n}{(n+1)^2}\gamma^2 = \frac{n}{(n+1)^2}\|\mathbf{r}\|^2 = c_{4our} \ .$$

Thus, the obtained covariance matrix $\mathbf{C}$ is equal in both algorithms, therefore the estimated subspace is identical.

C.2 Weighted incremental PCA

In Section 4.4 we presented a method for weighted incremental learning. In the literature, the most related work is the algorithm for eigenspace updating with temporal weighting proposed by Liu and Chen [41]. In this section we will show that their algorithm can be considered as a special case of ours when the temporal weights in our method form a geometric sequence.

Their algorithm is very simple. At each step n it first estimates the mean vector:

$$\boldsymbol{\mu}_{liu}^{(n)} = \frac{\mathbf{x}_n + \alpha\mathbf{x}_{n-1} + \alpha^2\mathbf{x}_{n-2} + \dots}{1 + \alpha + \alpha^2 + \dots} \ ,$$

where α is a decay parameter, which controls how much the previous samples should contribute to the estimation of the current mean. Since α is in the range from 0 to 1, we have $1 + \alpha + \alpha^2 + \ldots = \frac{1}{1-\alpha}$. Thus, the mean can be estimated as

$$\boldsymbol{\mu}_{liu}^{(n)} = \alpha\boldsymbol{\mu}^{(n-1)} + (1-\alpha)\mathbf{x}_n \ .$$

Similarly, the covariance matrix can be estimated as

$$\mathbf{C}^{(n)}_{liu} = \beta \mathbf{C}^{(n-1)} + (1-\beta)(\mathbf{x} - \boldsymbol{\mu}^{(n)})(\mathbf{x} - \boldsymbol{\mu}^{(n)})^\top \ ,$$

where β is a decay parameter for the covariance matrix.

Once the mean vector and the covariance matrix are estimated, the eigenvectors are obtained using SVD or EVD. The estimated covariance matrix and the mean vector are later input in the next step of the updating algorithm. Thus, the entire covariance matrix is calculated and being maintained throughout the process of incremental learning. For the cases where the vector size is larger than number of the input vectors they proposed also a similar algorithm, based on the inner-product matrix, which is smaller than the covariance matrix.

We will show that the results produced by both methods are almost equal when we set the weights in our method to

$$w_i = \alpha^{n-i} \ ,$$

where n is the number of the training images. To prove this, we have to show that the mean vector and the covariance matrix of both algorithms are equal in both methods at any step.

Our algorithm estimates the mean at the step $n+1$ by the following equation:

$$\boldsymbol{\mu}^{(n+1)}_{our} = \frac{1}{\sum_{i=1}^{n+1} w_i} \sum_{i=1}^{n+1} w_i \mathbf{a}'_i \ .$$

We can rewrite this equation in the recursive manner:

$$\boldsymbol{\mu}^{(n+1)}_{our} = \frac{1}{\sum_{i=1}^{n+1} w_i} \left(\sum_{i=1}^{n} w_i \boldsymbol{\mu}^{(n)} + w_{n+1} \mathbf{a}'_{n+1} \right) \ .$$

Inserting the values of the weights we obtain

$$\boldsymbol{\mu}^{(n+1)}_{our} = \frac{1}{\sum_{i=1}^{n+1} \alpha^{n+1-i}} \left(\sum_{i=1}^{n} \alpha^{n+1-i} \boldsymbol{\mu}^{(n)} + \mathbf{a}'_{n+1} \right) \ .$$

Considering that $\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$, we obtain for *large values* of n the same expression as in the case of the method Liu and Chen:

$$\boldsymbol{\mu}^{(n+1)}_{our} = \alpha \boldsymbol{\mu}^{(n)} + (1-\alpha)\mathbf{x}_{n+1} = \boldsymbol{\mu}^{(n+1)}_{liu} \ .$$

The derivation of the covariance matrix $\mathbf{C}$ is analogous. However, the difference is, that their method estimates $\mathbf{C}$ from input images $\mathbf{X}$, while our method estimates

C from their representations $\mathbf{A}'$. If during the process of incremental learning we do not discard any eigenvector, we obtain in both cases exactly the same covariance matrix, since $\mathbf{A}'$ represents $\mathbf{X}$ without losing any information, i.e., they are the same points expressed in different coordinate frames. If during the process of incremental learning we discard least significant eigenvectors, the covariance matrix of $\mathbf{A}'$ is as good approximation of the covariance matrix of $\mathbf{X}$ as possible. Therefore, we can conclude that

$$\mathbf{C}_{our} \approx \mathbf{C}_{liu} \ ,$$

which indicates that the algorithm of Liu and Chen can be considered as the special case of ours.

C.3 Incremental PCA on incomplete data

Here we will show that the rule for imputing missing pixels in the incremental algorithm, which we presented in Section 4.5, is equivalent to the imputation rule proposed by M. Brand [14].

Our rule for imputation of missing pixels is:

$$\hat{\mathbf{x}}^{\circ}_{our} = \mathbf{U}^{\circ}\mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^{\bullet} \ . \tag{C.1}$$

Brand's rule for imputation of missing pixels [14] is:

$$\hat{\mathbf{x}}^{\circ}_{brand} = \mathbf{U}^{\circ}\mathbf{\Lambda}(\mathbf{U}^{\bullet}\mathbf{\Lambda})^{\dagger}\hat{\mathbf{x}}^{\bullet} \ , \tag{C.2}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues on its diagonal. By expanding the pseudoinverse and considering the diagonality of $\mathbf{\Lambda}$, (C.2) can be rewritten as

$$\begin{aligned}
\hat{\mathbf{x}}^{\circ}_{brand} &= \mathbf{U}^{\circ}\mathbf{\Lambda}(\mathbf{\Lambda}^{\top}\mathbf{U}^{\bullet\top}\mathbf{U}^{\bullet}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}^{\top}\mathbf{U}^{\bullet\top}\hat{\mathbf{x}}^{\bullet} = \\
&= \mathbf{U}^{\circ}\mathbf{\Lambda}(\mathbf{\Lambda}\mathbf{U}^{\bullet\top}\mathbf{U}^{\bullet}\mathbf{\Lambda})^{-1}\mathbf{\Lambda}\mathbf{U}^{\bullet\top}\hat{\mathbf{x}}^{\bullet} = \\
&= \mathbf{U}^{\circ}(\mathbf{U}^{\bullet\top}\mathbf{U}^{\bullet})^{-1}\mathbf{U}^{\bullet\top}\hat{\mathbf{x}}^{\bullet} = \mathbf{U}^{\circ}\mathbf{U}^{\bullet\dagger}\hat{\mathbf{x}}^{\bullet} = \hat{\mathbf{x}}^{\circ}_{our} \ ,
\end{aligned} \tag{C.3}$$

Therefore, if we neglect the numerical issues, the Brand's rule for imputation of missing pixels is equivalent to our rule.

Appendix D

Extended summary in Slovenian

D.1 Uvod

D.1.1 Učenje, predstavitev, razpoznavanje

Vid je najpomembnejši človekov čut. Oskrbuje nas z veliko količino podatkov o naši okolici, o bližnjih predmetih in osebah ter njihovih dejanjih. Toda vizualni podatki, ki jih zajame mrežnica človeškega očesa, dobijo pomen šele, ko so razumljeni v naših možganih, torej, ko vsaj do določene mere razpoznamo predmete, ki nas obkrožajo. Že sama beseda "razpoznavanje" implicira na dejstvo, da moramo nek predmet najprej poznati, da ga lahko potem raz-poznamo. Najprej se moramo torej *naučiti*, kako nek predmet izgleda, zapomniti si moramo njegovo *predstavitev*, šele potem lahko ta predmet *razpoznamo*. Učenje, predstavitev in razpoznavanje so torej trije tesno povezani deli vizualnega zaznavanja.

Čeprav je vid oz. interpretacija vizualne informacije zelo enostavna naloga za človeka, je to zelo zahtevno opravilo za računalnik. Raziskovalci na področju računalniškega vida se že zelo dolgo trudijo, da bi zmogljivosti računalniških sistemov za vizualno zaznavanje vsaj deloma približali sposobnostim človeka. V osemdesetih letih prejšnjega stoletja je tako prevladovalo prepričanje, da je za uspešno razpoznavanje potrebno predmete predstaviti kot *predmetno-osrediščene* (object-centered) 3-dimenzionalne modele, ki so invariantni na smer pogleda (t.i. Marrova paradigma [43]). Potem, ko so različne psihofizične, neurofiziološke in behavioristične študije postregle z dokazi, da je človeško razpoznavanje objektov zelo odvisno od smeri pogleda na predmet [73, 77, 15, 35, 72, 42], se je tudi v računalniškem vidu bolj uveljavil *glediščno-osnovani* (view-based) pristop do učenja in razpoznavanja objektov in scen. Pri tem pristopu se objekti ne modelirajo kot glediščno invariantni 3-D modeli, temveč je predstavitev sestavljena iz množice 2-D pogledov. Predmet je lahko tako predstavljen kar z množico 2-D slik. Tak pristop lahko imenujemo tudi *slikovno-osnovani* (image-based) ali *na izgledu osnovani* (appearance-based) pristop.

D.1.2 Na izgledu osnovani pristop

Izgled predmeta je odvisen od skupnega učinka oblike, odbojnih lastnosti površine predmeta, njegovega položaja in osvetlitve [55]. Učenje pri na izgledu osnovanem pristopu poteka tako, da vseh teh faktorjev ne poskušamo ločiti med seboj, ampak sistematično zajamemo vse (vsaj v grobem) možne izglede predmeta z vseh zornih kotov in pri različnih osvetlitvah. Razpoznavanje neznanega objekta oz. prizora na sliki se nato poenostavi v iskanje najbolj podobne slike zajete v fazi učenja. Celoten proces modeliranja ter razpoznavanja torej poteka v domeni 2-D slik. Tak pristop bi bil seveda časovno in prostorsko preveč potraten, zato je potrebno vse vhodne slike zajete z različnih pogledov učinkovito predstaviti v zgoščeni predstavitvi, ki omogoča hitro iskanje oz. neposredno ujemanje testnih slik z učnimi.

V ta namen se pogostokrat uporabljajo metode na osnovi podprostorov, kot je *metoda glavnih komponent PCA* (principal component analysis) [34]. Osnovna ideja metode PCA je, da se visoko-dimenzijski vhodni podatki na učinkovit način preslikajo v nizko-dimenzijski podprostor in pri tem ohranijo kar največ informacije. V visoko-dimenzijskem prostoru vhodnih podatkov se tako poiščejo smeri, v katerih je varianca podatkov največja. Smeri, v katerih je varianca vhodnih podatkov majhna, lahko zanemarimo in tako zmanjšamo razsežnost prostora. Napaka, ki jo naredimo pri preslikavi vhodnih podatkov v podprostor, ki ima glavne smeri za bazne vektorje, je minimalna (v smislu napake najmanjših kvadratov) med vsemi linearnimi preslikavami v podprostor enake dimenzije.

Poleg metode glavnih komponent poznamo še nekatere druge metode za zmanjševanje dimenzionalnosti, kot so *linearna diskriminantna analiza* (LDA) [8, 44], *analiza kanonične korelacije* (CCA) [47, 46], *analiza neodvisnih komponent* (ICA) [36, 6] ter *ne-negativna faktorizacija matrik* (NMF) [39], kot tudi nelinearne izvedenke vseh teh metod [20, 65, 48, 47]. Ker pa je PCA zaradi svojih lastnosti najbolj primerna in uporabljana metoda za vizualno učenje in razpoznavanje, se bomo v tem delu osredotočili nanjo.

Nasploh pa je vizualno modeliranje na osnovi podprostorov postalo zelo priljubljen in pogosto uporabljan pristop za parametrizacijo oblike, izgleda in gibanja [21, 13, 49, 53, 76]. To je vodilo do uspešnih aplikacij, ki rešujejo probleme, kot so razpoznavanje obrazov [76, 9], vizualni nadzor [80], vizualno pozicioniranje in sledenje robotskim manipulatorjem [54], načrtovanje osvetlitve [51], samolokalizacija robota [37], modeliranje ozadja [57], idr.

D.1.3 Definicija problema

Kljub enostavnosti (ali pa ravno zaradi tega) ima na izgledu osnovani pristop, temelječ na analizi glavnih komponent, kar precej pomanjkljivosti. Najprej se bomo posvetili eni izmed pomanjkljivosti standardne metode PCA. Glavni podprostor se ponavadi računa s paketno metodo; vse učne slike se procesirajo naenkrat, kar pomeni, da morajo biti vse učne slike podane vnaprej. To pa ni izvedljivo v mnogo realističnih situacijah, kjer učne slike prihajajo zaporedno ena za drugo. Zato bomo v disertaciji predlagali metodo za *inkrementalno učenje*.

Standardna metoda za vizualno učenje ravno tako ne upošteva dejstva, da imajo lahko tako različni deli slik kot tudi celotne posamezne slike različen vpliv na proces učenja, čeprav je to v praksi zelo verjetno. Zato bomo v disertaciji predlagali posplošeni pristop do analize glavnih komponent, ki bo *selektivno* obravnaval posamezne slike in tudi slikovne elemente. V praktičnih aplikacijah se tudi pogosto zgodi, da del podatkov ni na voljo, t.j., da slike vsebujejo t.i. *manjkajoče slikovne elemente* (missing pixels). Zato bomo algoritme za uteženo učenje prilagodili tako, da bodo lahko uspešno reševali tudi ta problem.

Slike, ki jih zajamemo z različnimi vizualnimi senzorji, pa pogosto tudi niso idealne in vsebujejo različne nezaželene dodatke, kot so šum ali delna zakrivanja. V disertaciji bomo zato predstavili tudi *robustne algoritme*, ki bodo zaznali nekonsistentnosti v vhodnih podatkih in zgradili robustne predstavitve objektov samo iz konsistentnih podatkov.

Za vse omenjene pristope (uteženo učenje, učenje iz delnih podatkov, robustno učenje) bomo razvili paketne in inkrementalne metode, ter analizirali za kakšne vrste slik in pod kakšnimi pogoji so te metode uporabne.

D.1.4 Pregled sorodnih del

V preteklosti je bilo že predlaganih nekaj metod za inkrementalno učenje na osnovi glavnih podprostorov. Prvi tak algoritem na področju računalniškega vida sta predlagala Murakami in Kumar [50]. Nato so Chandrasekeran in dr. predlagali algoritem, ki temelji na nadgrajevanju razcepa s singularnimi vrednostmi (SVD) [19], kar je bila tudi tema nekaterih drugih del [16, 31, 14]. Vsi ti algoritmi pa ne upoštevajo, da se povprečna vrednost slik lahko spreminja skozi čas. S tem problemom so se prvi spopadli Hall in dr. [32, 33]. Njihova metoda sicer daje ekvivalentne rezultate kot algoritem, ki ga bomo predstavili v tej disertaciji. Vendar pa je naš algoritem zasnovan na tak način, da omogoča tudi uteženo učenje in da ni potrebno učnih slik shranjevati, ampak jih lahko zavržemo takoj po uporabi.

Večina algoritmov za gradnjo glavnih podprostorov temelji na dekompoziciji kovariančne matrike vhodnih podatkov. Možni pa so tudi drugi pristopi. Na PCA lahko gledamo tudi z verjetnostnega vidika, kot na mejni primer linearnih gaussovih modelov, kjer je šum neskončno majhen in enak v vseh smereh [63]. Iz tega lahko izpeljemo algoritem za računanje glavnih smeri, ki temelji na EM (expectation-maximization) algoritmu [25, 63, 75]. Bazne vektorje glavnega podprostora, t.j. glavne smeri prostora vhodnih podatkov, lahko iščemo tudi tako, da minimiziramo kvadratno rekonstrukcijsko napako vseh vhodnih vektorjev. Če hočemo, da posamezni slikovni elementi selektivno vplivajo na učenje, lahko minimiziramo uteženo rekonstrukcijsko napako. Takšne pristope so predlagali npr. Wiberg [78], Shum in dr. [66], Gabriel and Zamir [30], Sidenbladh in dr.[67] ter De la Torre in Black [24]. Brand [14] ter Liu in Chen [41] so predstavili tudi inkrementalne algoritme za uteženo učenje, ki so primerni za učenje ob določenih predpostavkah. Algoritmi, ki jih bomo predlagali v tem delu, bodo bolj splošni.

Velika pomanjkljivost omenjenega osnovnega pristopa vizualnega modeliranja na osnovi podprostorov je nerobustnost na šum, delna zakrivanja in nepravilno ozadje. Zato je bilo predlaganih precej metod za izboljšanje robustnosti razpoznavanja: modularni lastni prostori [59], lastna okna (eigenwindows) [56], iskalno okno (search-window) [52], adaptivne maske [27], M-estimacija [12], hierarhični pristop [61] ter pristop na osnovi podvzorčenja [40]. Vse te metode pa se ukvarjajo z robustnostjo v fazi *razpoznavanja* in predpostavljajo, da so bile slike v fazi učenja nepokvarjene, da je torej vizualni model pravilen.

Robustno učenje je veliko težji problem. Njegova naloga je v fazi učenja zgraditi konsistenten model iz neidealnih učnih slik. Ker pa model objekta oz. prizora šele gradimo, nimamo veliko predhodnega znanja, na osnovi katerega bi lahko določili odstopajoče slikovne elemente oz. dele slik. Nekaj avtorjev se je spoprijelo tudi s tem problemom. Xu in Yuille sta predlagala algoritem [79], ki je uvedel robustnost na nivoju slik in pri gradnji modela ni upošteval slik, ki so bile nekonsistentne z ostalimi. V praktičnih problemih pa to ni dovolj. Zagotoviti moramo namreč robustnost na nivoju slikovnih elementov. Gabriel in Zamir sta ta problem rešila z uteženim razcepom s singularnimi vrednostmi (SVD) [30]. Še korak dlje sta šla De la Torre in Black, ki sta predlagala metodo za robustno analizo glavnih komponent, ki temelji na M-estimaciji [23, 24]. Ta metoda daje zadovoljive rezultate na slikah z dovolj veliko časovno konsistentnostjo, je pa časovno zelo zahtevna. Metoda, ki jo bomo predstavili v disertaciji, je časovno precej manj potratna, daje pa zelo podobne rezultate. Poleg tega bomo predstavili tudi inkrementalno robustno metodo, ki se deloma nanaša na nedavno objavljeno metodo avtorjev Aanæsa in dr. [3].

D.2 Osnovna PCA

D.2.1 Kaj je PCA?

Metoda glavnih komponent je linearna transformacija iz visoko-dimenzijskega prostora vhodnih podatkov (input space) v nizko-dimenzijski prostor značilk (feature space), ki med vsemi linearnimi transformacijami zagotavlja najboljšo predstavitev visoko-dimenzijskih vhodnih vektorjev z nizko-dimenzijskimi vektorji koeficientov v prostoru značilk. Koordinatni sistem obrne v takšno smer, da je variabilnost vhodnih vektorjev lahko opisana že z majhnim številom baznih vektorjev.

D.2.2 Izpeljava in lastnosti PCA

Zelo pomembna lastnost analize glavnih komponent je, da maksimizira varianco preslikav vhodnih vektorjev v glavni podprostor. Iz te lastnosti lahko izpeljemo tudi algoritem za računanje glavnega podprostora.

Recimo, da imamo N M-dimenzionalnih vektorjev $\mathbf{x}_j$ zloženih v matriko podatkov $\mathbf{X} \in \mathbb{R}^{M\times N}$. Naš cilj je najti takšno smer $\mathbf{u}$ (vektor dolžine 1) v vhodnem prostoru $\mathbb{R}^M$, ki maksimizira varianco preslikav vseh vhodnih vektorjev na $\mathbf{u}$. Ker lahko projekcijo j-tega vektorja $\mathbf{x}_j$ na vektor $\mathbf{u}$ izračunamo kot $a_j = \langle \mathbf{x}_j, \mathbf{u}\rangle = \mathbf{u}^\top \mathbf{x}_j = \sum_{i=1}^{M} u_i x_{ij}$, lahko varianco vseh vektorjev koeficientov $a_j, j = 1 \dots N$ izrazimo s kovariančno matriko vhodnih vektorjev $\mathbf{C}$:

$$\sigma^2 = \mathbf{u}^\top \mathbf{C}\mathbf{u} \ . \tag{D.1}$$

Izkaže se, da moramo za maksimizacijo σ^2 rešiti klasičen problem razcepa z lastnimi vrednostmi (EVD):

$$\mathbf{C}\mathbf{u} = \lambda\mathbf{u} \ , \tag{D.2}$$

ki ga lahko rešimo tudi z razcepom s singularnimi vrednostmi (SVD):

$$\mathbf{C} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top \ , \tag{D.3}$$

tako da ortonormalna matrika $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N] \in \mathbb{R}^{M\times N}$ vsebuje po stolpcih lastne vektorje $\mathbf{u}_1, \ldots, \mathbf{u}_N$, diagonalna matrika $\boldsymbol{\Lambda} \in \mathbb{R}^{N\times N}$ pa na diagonali lastne vrednosti $\lambda_1, \ldots, \lambda_N$. Predpostavili bomo, da so lastne vrednosti in pripadajoči lastni vektorji urejeni po padajočem vrstnem redu lastnih vrednosti, torej $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$. Tako je večina variabilnosti vhodnih vektorjev vsebovana že v prvih lastnih vektorjih, ki jih imenujemo tudi *glavni vektorji* ali *glavne osi*. Ohranimo torej samo prvih $k, k \ll N$ lastnih vektorjev, torej $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \mathbb{R}^{M\times k}$.

Vhodni vektor lahko sedaj projiciramo na k-dimenzionalni glavni podprostor z uporabo transformacijske matrike $\mathbf{U}^\top : \mathbb{R}^M \to \mathbb{R}^k$:

$$\begin{aligned} \mathbf{a} &= \mathbf{U}^\top\hat{\mathbf{x}} = \mathbf{U}^\top(\mathbf{x} - \boldsymbol{\mu}) \\ a_j &= \langle \hat{\mathbf{x}}, \mathbf{u}_j \rangle = \sum_{i=1}^{M} \mathbf{u}_{ij}\hat{\mathbf{x}}_i = \sum_{i=1}^{M} \mathbf{u}_{ij}(x_i - \mu_i) \ , \ j = 1 \ldots k \end{aligned} \tag{D.4}$$

in ga nazaj rekonstruiramo z uporabo transformacijske matrike $\mathbf{U} : \mathbb{R}^k \to \mathbb{R}^M$:

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{U}\mathbf{a} = \sum_{j=1}^{k} a_j\mathbf{u}_j \\ \mathbf{y} &= \hat{\mathbf{y}} + \mu \ . \end{aligned} \tag{D.5}$$

Vhodni vektor torej predstavimo z linearno kombinacijo prvih k glavnih vektorjev.

Takšna predstavitev je še posebej primerna za modeliranje množice vhodnih vektorjev, ki so med seboj zelo korelirani, saj jih dekorelira in s tem odpravi redundanco v vhodnih podatkih ter tako zelo zmanjša količino potrebnih podatkov za predstavitev. Tako je PCA optimalna v smislu zgoščevanja podatkov, saj pri določeni stopnji kompresije (številu baznih vektorjev) zagotavlja minimalno rekonstrukcijsko napako

$$e = \sum_{i=1}^{M}\sum_{j=1}^{N}\left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il}a_{lj}\right)^2 \ . \tag{D.6}$$

Na PCA pa lahko gledamo tudi kot na mejni primer gaussovega modela, ko je šum neskončno majhen in enak v vseh smereh. Roweis [63] je iz te opazke izpeljal algoritem za izračun glavnega podpostora, ki temelji na EM algoritmu [25, 63, 75]. Algoritem izračuna glavne smeri in glavne komponente na iterativen način z izmeničnim izvajanjem naslednjih E in M korakov:

- **E-korak**: $\mathbf{A} = (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \hat{\mathbf{X}}$
- **M-korak**: $\mathbf{U} = \hat{\mathbf{X}} \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1}$.

D.2.3 PCA za vizualno učenje in razpoznavanje

Zaradi vseh omenjenih lastnosti je analiza glavnih komponent zelo primerna za vizualno učenje. Pri na izgledu osnovanem pristopu namreč modeliramo predmet z veliko množico slik tega predmeta, ki so med seboj korelirane, zato jih lahko z uporabo PCA zelo zgostimo. V kontekstu vizualnega modeliranja gledamo na sliko kot na vektor v zelo visoko-dimenzionalnem vhodnem prostoru slik. Vse slike nato z uporabo PCA preslikamo v nizko-dimenzijski prostor značilk. Izkaže se, da lahko korelacijo med dvema slikama v prostoru slik aproksimiramo z razdaljo med njihovima projekcijama v prostoru značilk.

Učenje in razpoznavanje predmetov lahko tako izvedemo na naslednji način. Objekt modeliramo z množico slik, ki jih zajamemo z različnih pogledov med sistematičnim opazovanjem objekta. Primer je prikazan na sliki D.1(a). Iz množice slik nato izračunamo lastne vektorje (oz. lastne slike, glej sliko D.1(b)) in zgradimo glavni podprostor ter vanj preslikamo vse vhodne slike tako, da dobimo njihove nizko-dimenzijske predstavitve. Razpoznavanje objektov se tako poenostavi na projiciranje vhodne slike na nizko-dimenzijski podprostor in iskanje najbližje projekcije neke učne slike v tem podprostoru. Ker so vhodne slike ponavadi zelo korelirane, je dimenzija podprostora majhna, kar zagotavlja zgoščeno predstavitev objektov oz. prizorov in hitro iskanje. Poleg tega pa lahko z interpolacijo med projekcijami v prostoru značilk generaliziramo zajete učne slike tudi na še nevidene poglede.

D.3 Inkrementalna PCA

Večina algoritmov za gradnjo glavnega podprostora procesira vse učne slike naenkrat. To seveda ni primerno za situacije, kjer vse učne slike niso znane vnaprej, ampak je

(a)
(b)

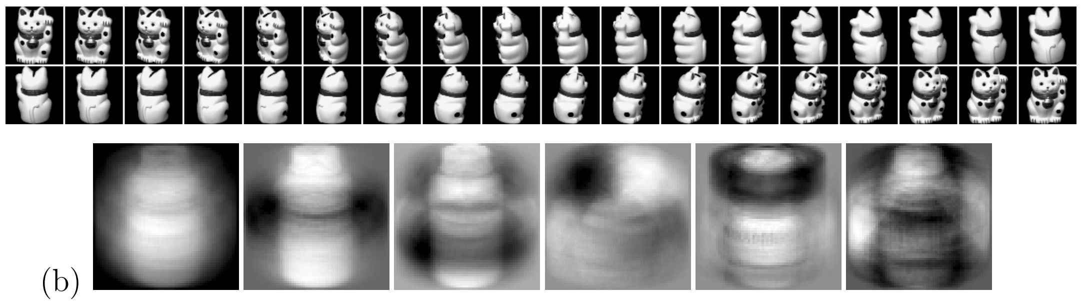



Figure D.1: (a) Slike, s katerimi modeliramo predmet. (b) Povprečna slika in prvih pet lastnih slik.

na voljo le ena slika naenkrat, druga za drugo. Zato bomo predstavili inkrementalni algoritem, ki zgradi glavni podprostor postopno po korakih.

D.3.1 Algoritem

Recimo, da imamo že izgrajen glavni podprostor iz prvih n slik. V koraku $n+1$ bi lahko izračunali novi glavni podprostor iz *rekonstrukcij* prvih n slik in nove slike z običajno paketno metodo. Seveda pa bi bila računska zahtevnost takšnega inkrementalnega algoritma previsoka, saj bi morali na vsakem koraku izvesti paketno metodo PCA na množici visoko-dimenzijskih podatkov. Identične rezultate pa lahko dobimo z uporabo nizko-dimenzijskih *vektorjev koeficientov* (predstavitev) prvih n slik namesto njihovih visoko-dimenzijskih rekonstrukcij, saj so vektorji koeficientov in njihove rekonstrukcije v bistvu iste točke predstavljene v različnih koordinatnih sistemih. Ker je dimenzija glavnega podprostora majhna, je takšen algoritem zelo učinkovit. En korak tega algoritma je povzet v Algoritmu 15.

Algoritem poveča dimenzijo glavnega podprostora za eno. Če hočemo ohraniti dimenzijo podprostora nespremenjeno, lahko zavržemo najmanj pomemben bazni vektor. Za začetni podprostor lahko postavimo kar prvo učno sliko, torej $\boldsymbol{\mu}^{(1)} = \mathbf{x}_1$, $\mathbf{U}^{(1)} = \mathbf{0}_{M\times 1}$, in $\mathbf{A}^{(1)} = 0$. Tako je algoritem popolnoma inkrementalen, saj na vsakem koraku zahteva samo eno sliko. Ker se predstavitve slik hranijo in obnavljajo skozi celoten proces učenja, lahko namreč vsako učno sliko takoj po uporabi zavržemo.

Algorithm 15 : Inkrementalna PCA

```
Input: trenutni povprečni vektor $\boldsymbol{\mu}^{(n)}$, trenutni lastni vektorji $\mathbf{U}^{(n)}$, trenutni koeficienti $\mathbf{A}^{(n)}$, nova vhodna slika $\mathbf{x}$.
Output: novi povprečni vektor $\boldsymbol{\mu}^{(n+1)}$, novi lastni vektorji $\mathbf{U}^{(n+1)}$, novi koeficienti $\mathbf{A}^{(n+1)}$, nove lastne vrednosti $\boldsymbol{\lambda}^{(n+1)}$.
1: Projiciraj novo sliko $\mathbf{x}$ v trenutni lastni prostor: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$ .
2: Rekonstruiraj novo sliko: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
3: Izračunaj vektor razlike: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ je pravokoten na $\mathbf{U}^{(n)}$.
4: Dodaj $\mathbf{r}$ kot nov bazni vektor: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{bmatrix}$ .
5: Določi vrednosti koeficientov v novi bazi: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & \|\mathbf{r}\| \end{bmatrix}$ .
6: Izvedi PCA na $\mathbf{A}'$. Dobi povprečno vrednost $\boldsymbol{\mu}''$, lastne vektorje $\mathbf{U}''$, in lastne vrednosti $\boldsymbol{\lambda}''$.
7: Projiciraj vektorje koeficientov na novo bazo: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$ .
8: Zavrti podprostor $\mathbf{U}'$ za $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$ .
9: Popravi povprečni vektor: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$ .
10: Nove lastne vrednosti: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$ .
```

D.3.2 Lastnosti inkrementalne gradnje

Paketna metoda PCA je optimalna v smislu minimalne kvadratne rekonstrukcijske napake. Inkrementalna metoda operira s približki vhodnih slik, zato so njeni rezultati nujno nekoliko slabši. Toda kolikšno je poslabšanje rezultatov? So rezultati še vedno uporabni? Kateri faktorji so pri tem odločilni? Da bi dobili odgovore na ta vprašanja in ugotovili nekatere lastnosti inkrementalne gradnje smo izvedli različne poizkuse na množici 720 slik dvajsetih predmetov iz slikovne baze COIL20 [53]. V nadaljevanju navajamo najpomembnejše ugotovitve.

Zavračanje slik

Ugotovili smo, da je potem, ko ob koncu inkrementalnega učenja preslikamo vse učne slike v zgrajeni glavni podprostor in jih nato rekonstruiramo, kvadratna rekonstrukcijska napaka zelo primerljiva z napako paketne metode (v povprečju je večja za 3,1 %). Kot smo že omenili, pa predlagani algoritem omogoča, da vsako sliko takoj po uporabi zavržemo in nato operiramo samo z njeno predstavitvijo v glavnem podprostoru. To seveda nekoliko poslabša rezultate, ki pa so v povprečju slabši od paketne metode še vedno samo za 8,6 %. Zaključimo torej lahko, da so rezultati inkrementalne metode povsem primerljivi z rezultati paketne metode in da so uporabni za večino aplikacij.

Vrstni red slik

Rezultati inkrementalne gradnje so še precej boljši, če podajamo slike v naključnem in ne v zaporednem vrstnem redu. V primerjavi s paketno metodo se rezultati poslabšajo samo za 1,3 %, če slik ne zavračamo, oz. 3,1 %, če jih. Že na začetku se namreč zgradi podprostor, ki zelo na grobo zajema izglede celotnega zaporedja slik in se potem skozi fazo učenja samo še rahlo prilagaja novim slikam, kar omogoča boljše predstavitve.

Povečevanje dimenzije podprostora

Analizirali smo tudi, kako vpliva na rezultate strategija povečevanja dimenzije glavnega podprostora. Če že vnaprej vemo, kakšna naj bi bila ciljna dimenzija podprostora, je najbolje, da na začetku na vsakem koraku povečamo dimenzijo podprostora za 1, dokler ne doseže ciljne dimenzije, nato pa to dimenzijo ohranjamo do konca inkrementalnega učenja.

Energija podprostora

Želeli smo tudi ugotoviti, kako vpliva na rezultate količina energije, ki jo zajema podprostor (količnik med vsoto ohranjenih in vsoto vseh lastnih vrednosti). Po pričakovanju se rekonstrukcijska napaka manjša z večanjem dimenzije podprostora (oz. s povečevanjem količine zajete energije). Ugotovili pa smo tudi, da se relativno povečanje napake v primerjavi s paketno metodo rahlo povečuje z rastjo dimenzije podprostora.

Število učnih slik

Ugotovili smo, da število učnih slik ne vpliva bistveno na povprečno poslabšanje rekonstrukcijske napake v primerjavi s paketno metodo. Pač pa je lahko poslabšanje zelo različno pri slikah z različnih delov vhodnega zaporedja, kar pa je predvsem odvisno od kriterija za povečevanje dimenzije glavnega podprostora.

Kompresija

Na koncu smo želeli ugotoviti še, kako z naraščanjem števila vhodnih slik narašča število lastnih vektorjev, ki so potrebni za ohranitev določenega deleža energije. Ugotovili smo, da to število narašča samo do določene meje. Ko podprostor opisuje določeno število slik, zajema takšno vizualno variabilnost, da z dodajanjem novih slik ni več potrebno povečevati dimenzije lastnega prostora. Kljub temu pa z dodajanjem novih slik izboljšujemo trenutni podprostor (ga nekoliko vrtimo), tako da rekonstrukcijska napaka slik precej pade po tem, ko so le-te dodane v model. Čeprav se podprostor ne napihuje več, se še vedno prilagaja novim slikam. Ker pa kljub povečevanju števila vhodnih slik dimenzija podprostora ostaja enaka, je kompresija vedno boljša.

D.4 Utežena PCA

D.4.1 Paketna metoda za uteženo PCA

Standardna metoda za analizo glavnih komponent se lahko prevede na uteženo PCA, če v minimizacijsko funkcijo (D.6) dodamo uteži. Naj matrika $\mathbf{W} \in \mathbb{R}^{M\times N}$ vsebuje uteži, tako da je w_{ij} utež i-tega slikovnega elementa v j-ti sliki. Cilj je minimizirati *uteženo* kvadratno rekonstrukcijsko napako

$$\mathcal{E} = \sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij}\left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il}a_{lj}\right)^2 \quad . \tag{D.7}$$

Vrednosti matrike $\hat{\mathbf{X}}$ so v tem primeru dobljene z odštevanjem *uteženega* povprečnega vektorja $\boldsymbol{\mu}$ od učnih slik $\mathbf{x}_i$.

V praksi se ponavadi srečamo z dvema tipoma uteži: s *časovnimi* utežmi ${}^{\mathbf{t}}\mathbf{w} \in \mathbb{R}^{1\times N}$, ki utežijo posamezne slike, in s *prostorskimi* utežmi ${}^{\mathbf{s}}\mathbf{w} \in \mathbb{R}^{M}$, ki utežijo posamezne slikovne elemente neke slike.

Časovne uteži

Časovne uteži določajo vpliv posameznih slik na proces učenja. Če želimo neki sliki dodeliti večjo vlogo pri določanju glavnih osi, ji priredimo večjo utež. V tem primeru torej minimiziramo uteženo kvadratno rekonstrukcijsko napako oz. maksimiziramo uteženo varianco projekcij. Iz slednje lastnosti lahko izpeljemo algoritem za uteženo učenje, ki upošteva časovne uteži. Lastni prostor lahko tako izračunamo z razcepom s singularnimi vrednostmi *utežene* kovariančne matrike.

Splošne uteži

Algoritem za izračun glavnega podprostora z upoštevanjem splošnih uteži, ko ima lahko vsak posamezni slikovni element na katerikoli sliki poljubno utež, bomo izpeljali iz EM algoritma. Spomnimo se, da v E-koraku tega algoritma izračunamo nove vrednosti $\mathbf{a}_j$ na sledeči način: $\mathbf{a}_j = (\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top\hat{\mathbf{x}}_j$. Ta izraz pa lahko izrazimo tudi s pseudoinverzom $\mathbf{a}_j = \mathbf{U}^\dagger\hat{\mathbf{x}}_j$, ki pa je po drugi strani ekvivalenten reševanju ustreznega sistema linearnih enačb. Podobne opazke veljajo tudi za M-korak EM algoritma. Če posamezne enačbe ustrezno utežimo, lahko EM algoritem priredimo za iskanje takšnega podprostora, ki minimizira uteženo rekonstrukcijsko napako. Koraka E in M sedaj izgledata takole:

- **E-korak**: Izračunaj $\mathbf{A}$ na naslednji način: Za vsako sliko j, $j = 1 \dots N$, reši naslednji sistem linearnih enačb v smislu napake najmanjših kvadratov:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}}\sum_{p=1}^{k} u_{ip}a_{pj} \ , \ \ i = 1 \dots M \ . \tag{D.8}$$

- **M-korak**: Izračunaj $\mathbf{U}$ na naslednji način: Za vsak slikovni element i, $i = 1 \dots M$, reši naslednji sistem linearnih enačb v smislu napake najmanjših kvadratov:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}}\sum_{p=1}^{k} u_{ip}a_{pj} \ , \ \ j = 1 \dots N \ . \tag{D.9}$$

D.4.2 Paketna metoda za PCA na delnih podatkih

Prirejen EM algoritem

V praktičnih aplikacijah se pogosto zgodi, da vrednosti nekaterih slikovnih elementov niso definirane. Takšne situacije lahko obravnavamo kot poseben primer uteženega učenja, kjer postavimo uteži manjkajočih elementov na nič. V primeru EM algoritma to pomeni, da postavimo v sisteme linearnih enačb (D.8) in (D.9) samo tiste enačbe, ki izhajajo iz poznanih slikovnih elementov. Takšno reševanje tega problema pa je slabo pogojeno. Algoritem sicer res minimizira rekonstrukcijsko napako v znanih slikovnih elementih, vendar so hkrati lahko rekonstruirane vrednosti manjkajočih slikovnih elementov zelo daleč od pravih vrednosti, ker ima algoritem precej slabe generalizacijske sposobnosti. Zato predlagamo, da se v primerih, ko so učne slike posnete v zaporedju, tako da so sosednje slike med seboj precej podobne, uporabi še dodatna omejitev. V M-koraku poleg minimizacije rekonstrukcijske napake v znanih slikovnih elementih hkrati minimiziramo tudi drugi odvod rekonstruiranih vrednosti manjkajočih slikovnih elementov. To zagotavlja, da bodo vrednosti rekonstruiranih manjkajočih slikovnih elementov gladke skozi čas.

Iterativna rekonstrukcija slikovnih elementov

Alternativno lahko vrednosti $\mathbf{U}$ v M-koraku izračunamo tudi z uporabo navadne paketne metode PCA na celotni matriki vhodnih podatkov, pri pogoju, da so manjkajoči elementi ustrezno zapolnjeni. Vprašanje je, katera je optimalna metoda za polnjenje manjkajočih elementov. Ker ne poznamo vrednosti vseh slikovnih elementov neke slike, so nekatere koordinate ustrezne točke v prostoru slik nedefinirane. Pravilni položaj točke se torej nahaja nekje v podprostoru, ki ga določajo manjkajoče koordinate. Ob poznavanju glavnega podprostora $\mathbf{U}$, ki modelira vhodne podatke, je optimalna lokacija točka v podprostoru manjkajočih elementov, ki je najbližje glavnemu podprostoru. Dobimo jo tako, da manjkajoče slikovne elemente zapolnimo z rekonstruiranimi vrednostmi, ki jih izračunamo iz koeficientov dobljenih v E-koraku trenutne iteracije z uporabo glavnih osi dobljenih v M-koraku predhodne iteracije.

D.4.3 Inkrementalna utežena PCA

Inkrementalni algoritem lahko dopolnimo s principi uteženega učenja in tako dobimo algoritem za inkrementalno uteženo učenje.

Časovne uteži je zelo enostavno vključiti v inkrementalni algoritem. Jedro tega algoritma je še vedno standardna paketna metoda PCA na koeficientih. Vse, kar moramo storiti je, da to metodo zamenjamo z uteženo paketno metodo.

Prostorske uteži pa uvedemo v inkrementalno gradnjo z dodatnim korakom. Naj bodo prostorske uteži vrednosti z intervala med 0 in 1. Če je vrednost uteži 1, to pomeni, da se na vrednost slikovnega elementa popolnoma zanesemo in jo uporabimo takšno, kot je. Če je vrednost uteži 0, to pomeni, da je vrednost slikovnega elementa popolnoma nerelevantna in se nanjo popolnoma nič ne zanesemo. Takšno vrednost lahko rekonstruiramo z upoštevanjem trenutnega modela, ki smo ga zgradili v predhodnih korakih. S postavljanjem uteži med 0 in 1 lahko uravnavamo vpliv vrednosti slikovnega elementa in vpliv trenutnega modela (rekonstruirane vrednosti). Na začetku inkrementalnega algoritma torej uvedemo nov korak, ki najprej izračuna koeficiente nove slike **x** na utežen način (D.8) in te koeficiente rekonstruira v sliko **y**. Nato dobimo novo sliko kot uteženo vsoto originalne in rekonstruirane slike

$$x_i^{new} = {}^{s}w_i x_i + (1 - {}^{s}w_i) y_i \ , \quad i = 1 \dots M \ , \tag{D.10}$$

ki jo potem uporabimo za posodobitev trenutnega glavnega podprostora z inkrementalnim algoritmom.

D.4.4 Inkrementalna PCA na delnih podatkih

V primeru algoritma za računanje glavnega podprostora v prisotnosti manjkajočih slikovnih elementov se inkrementalni uteženi algoritem poenostavi tako, da imajo uteži lahko samo vrednosti 0 (manjkajoči slikovni elementi) ali 1 (znani slikovni elementi). Korak z uteženo vsoto se torej poenostavi na polnjenje vrednosti manjkajočih slikovnih elementov z rekonstruiranimi vrednostmi na podoben način, kot pri paketni metodi, ki temelji na iterativni rekonstrukciji manjkajočih slikovnih elementov.

D.5 Robustna PCA

Pri uteženi metodi za izračunavanje glavnega podprostora predpostavljamo, da so uteži (ali manjkajoči slikovni elementi) poznane. V praktičnih primerih pa to velikokrat ne drži, tako da mora algoritem sam odkriti odstopajoče slikovne elemente (outliers). To je naloga robustnih algoritmov.

D.5.1 Robustna paketna metoda

Enostopenjska metoda

Paketni robustni algoritem najprej odkrije odstopajoče slikovne elemente na vhodnih slikah. Kot odstopajoči se obravnavajo vsi slikovni elementi, ki niso konsistentni z istoležnimi elementi na ostalih slikah. Konsistenca se preverja s primerjanjem rekonstrukcijskih napak. Algoritem najprej iz vseh (neidealnih) vhodnih slik zgradi glavni podprostor na nerobusten način. Vse slike nato projicira v dobljeni podprostor in jih rekonstruira. Ker odstopajoči slikovni elementi niso konsistentni z ostalimi, je rekonstrukcijska napaka v njih ponavadi večja. Algoritem zato obravnava vse slikovne elemente z veliko rekonstrukcijsko napako kot manjkajoče slikovne elemente in nato izračuna robustni glavni podprostor z eno izmed prej omenjenih metod za učenje iz delnih podatkov. Nato se vrednosti teh slikovnih elementov v matriki vhodnih podatkov zamenjajo z rekonstruiranimi vrednostmi in celoten postopek se ponovi. Algoritem ponavadi zelo hitro skonvergira in že po nekaj iteracijah dobimo v vseh slikovnih elementih vrednosti, ki so konsistentne z ostalimi.

Dvostopenjska metoda

Pomanjkljivost predstavljene robustne metode je v tem, da je prvi korak še vedno standardna nerobustna PCA, ki lahko ob velikem številu odstopajočih slikovnih elementov proizvede slabo začetno rešitev. Če množica učnih slik vsebuje manjše število slik z zelo veliko odstopajočimi slikovnimi elementi, lahko to pomanjkljivost ublažimo tako, da odkrijemo odstopajoče točke v dveh stopnjah. Najprej odkrijemo *odstopajoče slike*, torej slike, v katerih je povprečna rekonstrukcijska napaka slikovnih elementov velika. Nato iz preostalih slik (ki ne vsebujejo veliko odstopajočih slikovnih elementov) zgradimo glavni podprostor. Ta podprostor nam potem v drugi stopnji služi za odkrivanje *odstopajočih slikovnih elementov* na vseh slikah. Na ta način jih lahko veliko bolj zanesljivo odkrijemo. Nadalje postopamo podobno kot v primeru enostopenjskega robustnega algoritma.

Robustna inicializacija z uporabo podvzorčenja

Začetni nerobusten korak robustnega algoritma lahko zamenjamo tudi z bolj robustnim pristopom, ki temelji na podvzorčenju. Najprej naključno izberemo podmnožico indeksov slikovnih elementov in podmnožico slik. Nato iz vseh slikovnih elementov z izbranimi indeksi na izbranih slikah zgradimo glavni podprostor s standardno metodo PCA. Izbrane slikovne elemente nato rekonstruiramo. Vsaka rekonstruirana vrednost hipotetično predstavlja pravo vrednost dotičnega slikovnega elementa. Če smo namreč pri naključnem izbiranju slikovnih elementov izbrali veliko neodstopajočih elementov, le-ti povzročijo, da je tudi rekonstruirana vrednost v odstopajočih slikovnih elementih bolj podobna pravi. Če torej ta postopek naključnega izbiranja slikovnih elementov in rekonstruiranja le-teh velikokrat ponovimo, za vsak slikovni element zgradimo veliko hipotez, med katerimi na koncu izberemo najboljšo, ki nam služi za osnovo pri odkrivanju odstopajočih slikovnih elementov.

D.5.2 Robustna inkrementalna metoda

Robustni inkrementalni algoritem je zelo podoben inkrementalnemu algoritmu za učenje iz delnih podatkov. Slednjemu je dodan samo še korak, ki odkrije odstopajoče slikovne elemente, ki se potem obravnavajo kot manjkajoči slikovni elementi. Nova vhodna slika se torej najprej robustno projicira [40] v trenutni glavni podprostor in nato rekonstruira. Na osnovi rekonstrukcijskih napak se določijo odstopajoči slikovni elementi in njihove vrednosti se zamenjajo z rekonstruiranimi vrednostmi. Tako popravljena slika se nato uporabi za posodobitev glavnega podprostora z inkrementalnim algoritmom.

D.6 Ocena učinkovitosti algoritmov

V doktorski disertaciji smo predlagali množico algoritmov za inkrementalno, uteženo in robustno vizualno učenje in razpoznavanje. Izvedli smo veliko poizkusov, s katerimi smo analizirali učinkovitost predlaganih algoritmov. V nadaljevanju bomo predstavili nekaj najpomembnejših rezultatov in ugotovitev.

D.6.1 Učinkovitost inkrementalnih algoritmov

Najprej bomo na primeru modeliranja obrazov prikazali delovanje vseh predlaganih inkrementalnih algoritmov. Cilj je bil zgraditi čim boljše 25-dimenzionalne predstavitve 400 slik obrazov iz slikovne baze ORL [64] z uporabo inkrementalnega algoritma.

Najprej smo model zgradili iz originalnih slik (slika D.2(a)) s paketno metodo. Rekonstrukcije slik, ki so prikazane na sliki D.2(b), so optimalne glede na to, da so predstavljene samo s 25 koeficienti in lastnimi slikami. Nato smo predstavitve zgradili še z inkrementalnim algoritmom in pri tem zavračali učne slike takoj po uporabi. Rekonstruirane slike, ki so prikazane na sliki D.2(c), so zelo podobne optimalnim, saj je rekonstrukcijska napaka v povprečju večja samo za 2 %. Nato smo zgradili predstavitve z uteženim inkrementalnim algoritmom, tako da smo novejšim slikam (slikam s konca zaporedja) priredili večje uteži. Kot rezultat vidimo, da sta zadnja dva obraza na sliki D.2(d) bolje rekonstruirana in vsebujeta več podrobnosti.

Tudi skupna utežena rekonstrukcijska napaka je manjša in znaša 598 pri standardni in 570 pri uteženi inkrementalni metodi.

Nato smo na vsaki sliki razen na prvih 40 izbrisali četrtino slike (glej sliko D.2(e)). Najprej smo manjkajoče slikovne elemente nadomestili kar s povprečno vrednostjo vseh znanih istoležnih slikovnih elementov in nato predstavitve zgradili iz tako zapolnjenih slik. Rezultati so prikazani na sliki D.2(f). Potem smo uporabili še predlagani inkrementalni algoritem za učenje iz delnih podatkov. Na sliki D.2(g) vidimo, da so obrazi precej bolje rekonstruirani, kar se odraža tudi v manjši rekonstrukcijski napaki (644 proti 760).

Na koncu smo na originalne slike dodali na naključne pozicije kvadratke naključnih intenzitet (slika D.2(h)). Na sliki D.2(i) vidimo, da nerobustna inkrementalna metoda kvadratke vključi v predstavitev, zato se pojavljajo tudi na rekonstruiranih slikah. Če pa za učenje uporabimo robustno inkrementalno metodo, le-ta na vsakem koraku odkrije kvadratke kot odstopajoče slikovne elemente in jih ne vključi v model. Tako so tudi rekonstruirani obrazi, ki so prikazani na sliki D.2(j), veliko bolj podobni optimalnim. Posledično je manjša tudi rekonstrukcijska napaka, ki znaša 915 pri standardni metodi in 710 pri robustni.

D.6.2 Ocena učinkovitosti vseh algoritmov

Na podoben način smo opravili še množico poizkusov, s katerimi smo ocenili večino algoritmov, tako paketnih kot tudi inkrementalnih, ki so bili predlagani v disertaciji. Da bi dobili realistično sliko o uporabnosti predlaganih algoritmov, smo le-te testirali na treh različnih slikovnih domenah: na slikah predmetov, obrazov in ozadja. Za slike predmetov (slika D.3(a)) je namreč značilno, da niso najbolj časovno korelirane, saj se med seboj precej razlikujejo. Obrazi (slika D.3(b)) so veliko bolj časovno koherentni, še najbolj pa so časovno korelirane slike ozadja (slika D.3(c)). Ozadje je namreč statično, spreminja se samo globalna osvetlitev, ki se enakomerno povečuje, in vertikalna senca, ki potuje čez sliko. Za slike ozadja je tudi značilno, da se zelo počasi in enakomerno spreminjajo, kar do neke mere velja tudi za predmete, ker so bile sosednje slike zajete iz sosednjih zornih kotov. To še najmanj velja za slike obrazov, ker le-ti niso bili posneti po kakšnem določenem vrstnem redu poz. Za oceno uteženih in robustnih algoritmov smo nato na vse slike, razen na prvih 40 v obeh vrstnih redih (vseh je bilo 720) naključno dodali kvadratek kot nezaželen in moteč element (slika D.3(d)).

Glavne lastnosti vseh treh tipov slik in učinkovitosti različnih algoritmov so

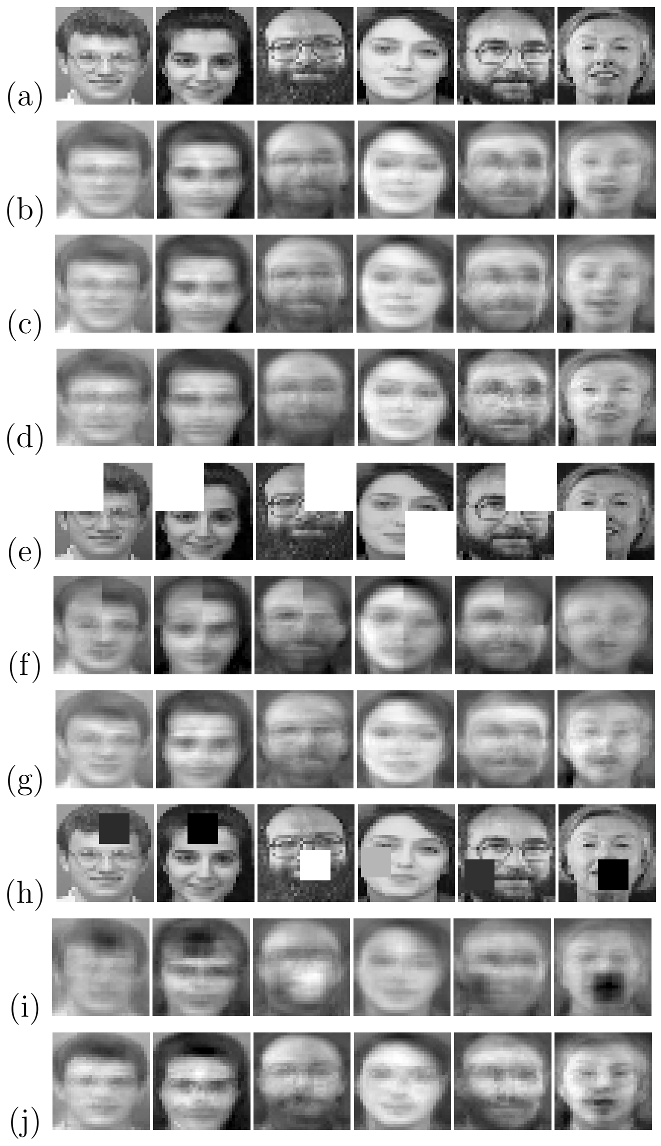


Figure D.2: (a) Učne slike. Rekonstrukcije dobljene z (b) paketno PCA, (c) inkrementalno PCA, (d) uteženo IPCA. (e) Učne slike z manjkajočimi slikovnimi elementi. Rekonstrukcije dobljene s (f) standardno IPCA, (g) IPCA z rekonstrukcijo manjkajočih slikovnih elementov. (h) Deloma prekrite učne slike. Rekonstrukcije dobljene z (i) standardno IPCA, (j) robustno IPCA.

povzete v tabeli D.1. Navedli bomo samo nekaj najbolj značilnih rezultatov. Učne slike smo na vhode algoritmov podajali v dveh različnih vrstnih redih. Najprej smo jih uredili po naravnem zaporednem vrstnem redu - najprej po vrsti vse slike enega predmeta, nato drugega in tako naprej. Nato smo jih uredili v izmeničnem

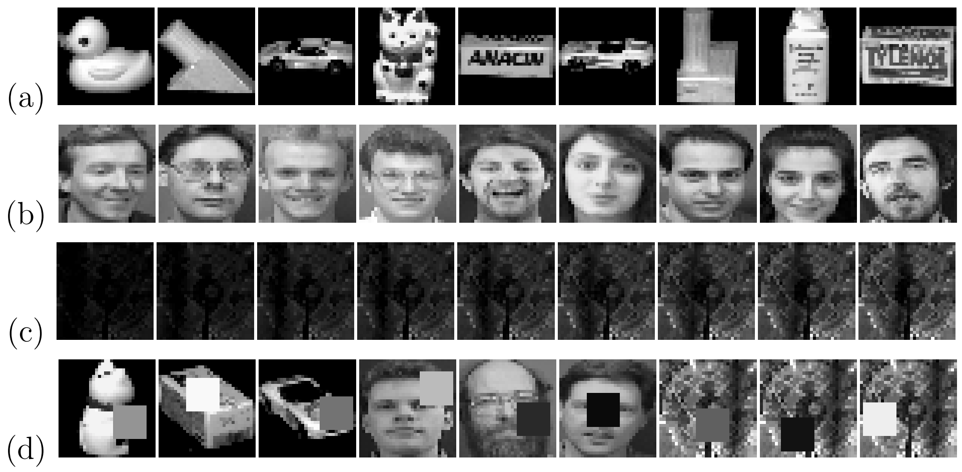


Figure D.3: Nekaj učnih slik: (a) predmeti, (b) obrazi, (c) ozadje. (d) Nekaj deloma prekritih slik.

vrstnem redu - najprej po ena slika vsakega predmeta, nato druga slika vsakega predmeta in tako naprej. Ugotovili smo, da vrstni red pomembno vpliva na rezultate inkrementalnih algoritmov, saj smo pri izmeničnem vrstnem redu vhodnih slik dobili precej boljše rezultate. To velja tako za navadno inkrementalno metodo, kot tudi za uteženo, še najbolj pa za robustno inkrementalno metodo. Pri navadni inkrementalni metodi je izmenični vrstni red ugodnejši, ker nam že na začetku učenja zgradi dovolj dober približek končnega glavnega podprostora, ki se nato samo še izboljšuje. Pri uteženi metodi je ta vpliv še večji, ker se trenutni model uporablja tudi za določanje novih vrednosti slikovnih elementov z majhnimi utežmi na novi sliki. Največji vpliv pa ima vrstni red pri robustni metodi, saj se v tem primeru trenutni model uporablja tudi za odkrivanje odstopajočih slikovnih elementov na novi sliki. Če trenutni model vsaj zelo približno ne zajema tudi izgleda nove slike, se vsi slikovni elementi, ki se na njej razlikujejo od prejšnjih istoležnih elementov, označijo za odstopajoče, torej tudi tisti, ki dejansko to niso, ampak prinašajo samo neko novo informacijo, ki bo sčasoma postala konsistentna. Zato moramo težiti k temu, da algoritmom za inkrementalno učenje (predvsem robustnemu) že na začetku podamo vsaj nekaj slik, ki v grobem predstavljajo različne možne izglede predmeta ali prizora.

Kot smo že ugotovili, je inkrementalna metoda zelo učinkovita, saj bistveno ne poslabša rezultatov paketne metode. V primeru deloma prekritih slik se dobro obnesejo tudi utežene metode in metode za učenje iz deloma manjkajočih podatkov. Pri testiranju teh metod smo namreč predpostavili, da so lokacije odstopajočih slikovnih elementov poznane, zato smo tem slikovnim elementov priredili manjše uteži oz. jih označili za manjkajoče. Predlagane metode so uspešno rekonstruirale prave vrednosti odstopajočih slikovnih elementov in zgradile predstavitve, ki nezaželenih kvadratkov niso vsebovale.

Rezultati robustne metode pa so bili zelo odvisni od vhodnih slik. Zelo slabo se je obnesla pri modeliranju predmetov. Problem je namreč v tem, da slike predmetov niso najbolj korelirane, tako da določanje odstopajočih slikovnih elementov samo na osnovi konsistentnosti posameznih slikovnih elementov ni bilo dovolj uspešno. Posledično so bili v predstavitve vgrajeni tudi nekateri kvadratki, izvzeti pa nekateri deli predmetov, ki so bili odkriti kot odstopajoči slikovni elementi, čeravno to niso bili (recimo robovi predmetov). Ta problem je bil veliko manj opazen pri slikah obrazov, ki so časovno bolj koherentne. Najboljše rezultate pa je robustni algoritem dosegel pri modeliranju ozadja, kjer je uspešno odkril večino odstopajočih slikovnih elementov in zgradil predstavitev ozadja brez motečih kvadratkov.

	predmeti	obrazi	ozadje
časovna korelacija	majhna	velika	zelo velika
enakomernost spreminjanja slik	velika	majhna	zelo velika
pomen vrstnega reda slik	velik	majhen	zelo velik
učinkovitost inkrementalne metode	zelo velika	zelo velika	zelo velika
učinkovitost utežene metode	velika	velika	zelo velika
učinkovitost robustne metode	zelo majhna	velika	zelo velika

Table D.1: Glavne lastnosti treh tipov slik in učinkovitost algoritmov.

D.7 Zaključek

D.7.1 Prispevki disertacije

V disertaciji smo se v glavnem ukvarjali s povečevanjem robustnosti vizualnega učenja z uporabo podprostorov. Analizirali smo različne probleme, ki se pri tem pojavljajo in predlagali različne rešitve, ki so odvisne od količine informacij, ki nam je na voljo.

V situacijah, v katerih so zanesljivosti posameznih slik in slikovnih elementov že vnaprej poznane, lahko uporabimo predstavljene metode za **uteženo učenje**. Za primere, ko so na slikah nekateri slikovni elementi nedefinirani, smo predlagali algoritme za **učenje ob manjkajočih podatkih**. Za situacije, ko nimamo nobenih dodatnih informacij o vsebini slik, pa smo predlagali algoritme za **robustno učenje**, ki poskušajo sami odkriti odstopajoče slikovne elemente in zgraditi robustni glavni podprostor. Za vse te pristope k vizualnemu učenju na osnovi podprostorov smo predlagali tako **paketne** kot tudi **inkrementalne** algoritme.

D.7.2 Nadaljnje delo

V disertaciji smo predlagali algoritme, ki deloma rešujejo nekatere probleme, ki se pojavljajo pri vizualnem učenju in razpoznavanju na osnovi podprostorov. Še vedno pa na tem področju ostaja veliko odprtih problemov.

Kot smo že ugotovili, se predlagana robustna metoda slabo obnese v primerih, ko vhodne slike niso zelo časovno korelirane, ker je odkrivanje odstopajočih slikovnih elementov v teh primerih zelo nezanesljivo. Ta problem bi lahko omilili tako, da bi povečali število testnih slik, na primer s procesiranjem video sekvence. Na ta način bi se slike počasneje spreminjale, redundanca podatkov bi bila večja, slike bolj časovno korelirane in odkrivanje odstopajočih slikovnih elementov bolj zanesljivo. Z uporabo inkrementalnega algoritma bi bilo procesiranje velikega števila slik tudi dejansko izvedljivo.

Še boljša rešitev pa bi bila poravnava učnih slik še pred učenjem. Tako bi bile slike bolj časovno korelirane, še posebej, če bi zaporedja učnih slik razbili na časovno koherentna podzaporedja in procesirali vsako podzaporedje posebej. V primeru modeliranja obrazov je poravnava preprosta, ker lahko vse slike, tako v fazi učenja kot tudi v fazi razpoznavanja, poravnamo na isto šablono [9], v primeru modeliranja poljubnih predmetov pa bi bil proces poravnave precej bolj zapleten. Vsekakor pa bi bila za bolj robustno in učinkovito gradnjo glediščno-osrediščenih modelov poljubnih predmetov potrebna vsaj zelo groba poravna slik.

Modeliranje z uporabo analize glavnih komponent lahko izvajamo ne le na sivinskih učnih slikah, marveč tudi na slikah drugih modalnosti (binarne, barvne, globinske, infra-rdeče). Slike lahko tudi predhodno predprocesiramo (poiščemo robove, ukrivljenosti), lahko pa na slikah poiščemo tudi razne druge značilke, lokalne ali globalne, ki jih nato uporabimo za modeliranje predmetov s PCA. Vsi algoritmi predlagani v disertaciji so popolnoma uporabni tudi za procesiranje takšnih podatkov. Na ta način bi lahko čisto holistični glediščno-osrediščeni pristop obogatili z lokalnimi geometričnimi značilkami in strukturnimi informacijami. To pa bi lahko bistveno pripomoglo k izboljšanju učinkovitosti na izgledu osnovanega vizualnega učenja in razpoznavanja.

Acknowledgments

First I would like to thank Prof. Aleš Leonardis, who made the completion of this dissertation possible. His devotion to science and broad knowledge were strong motivation and inspiration for my research work and his advices were always there when I needed them most.

I am also grateful to Prof. Franc Solina who invited me to join the Computer vision laboratory and introduced me the wide and interesting field of computer vision.

I would like to thank Matej Artač, Matjaž Jogan, and Dr. Jasna Maver for close collaboration and fruitful brain-storming sessions, which have crystallized many ideas presented in the dissertation. Many thanks also to Dr. Bojan Kverh, Dr. Aleš Jaklič, Peter Peer, Katarina Mele, and all other present and former members of the Computer Vision Laboratory for providing a very nice working atmosphere. It has been really a pleasure working with such a friendly and creative group.

I wish to thank Prof. Horst Bischof for exchanging ideas and for the work we have done together. I would also like to thank him as well as Prof. Saša Divjak and Dr. Blaž Zupan for serving as the members of the dissertation committee.

I want to express my gratitude to my parents, and also to relatives and friends, for encouraging me and helping me in various ways. Last, but not least, the most special thanks go to Iris for her love and support and to Jan for all the moments I have spent writing this dissertation instead of being with him.

Izjava

Izjavljam, da sem doktorsko disertacijo izdelal samostojno pod mentorstvom prof. dr. Aleša Leonardisa. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Bibliography

[1] M. Gökmen A. Yilmaz. Eigenhill vs. eigenface and eigenedge. In *International Conference on Pattern Recognition, ICPR'00*, volume 2, pages 2828–2830. IEEE, September 2000.

[2] T. Amano, S. Hiura, A. Yamaguti, and S. Inokuchi. Eigen space approach for a pose detection with range images. *Proceedings of ICPR'96*, pages 622–626, 1996.

[3] H. Anæs, R. Fisker, K. Åström, and J. M. Carstensen. Robust factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, September 2002.

[4] M. Artač, M. Jogan, and A. Leonardis. Mobile robot localization using an incremental eigenspace model. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pages 1025–1030, Washington DC, USA, May 2002.

[5] A. Bab-Hadiashar and D. Suter. Optic flow calculation using robust statistics. *Proceedings of CVPR'97*, pages 988–993, 1997.

[6] K. Baek, B. Draper, J. R. Beveridge, and K. She. PCA vs. ICA: A comparison on the FERET data set. In *The 6th Joint Conference on Information Sciences*, pages 824–827, Durham, North Carolina, March 8-14 2002.

[7] D. H. Ballard. *Neural Computation.* MIT Press, 1997.

[8] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenspaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[9] D. J. Beymer and T. Poggio. Face recognition from one example view. In *ICCV95*, pages 500–507, 1995.

[10] H. Bischof and A. Leonardis. Robust recognition of scaled eigenimages through hierarchical approach. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 664–670, 1998.

[11] H. Bischof, H. Wildenauer, and A. Leonardis. Illumination insensitive eigenspaces. In *Proc. of IEEE International Conference on Computer Vision*, pages 233–238, 2001.

[12] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 26(1):63–84, January 1998.

[13] M. J. Black, Y. Yacoob, A. D. Jepson, and D. J. Fleet. Learning parameterized models of image motion. In *CVPR97*, pages 561–567, 1997.

[14] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision – ECCV 2002*, volume I, pages 707–720, May 2002.

[15] H. H. Bülthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings of the National Academy of Sciences of the United States of America*, 89:60–64, 1992.

[16] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31:111–129, 1978.

[17] R. J. Campbell and P. J. Flynn. Eigenshapes for 3D object recognition in range data. In *CVPR99*, pages II:505–510, 1999.

[18] R. B. Cattell. The scree test for the number of factors. *Multivariate behavioral research*, 1:245–276, 1966.

[19] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, September 1997.

[20] N. Christianini and J. S. Taylor. *Support vector machines and other kernel-based methods*. Cambridge university press, 2000.

[21] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *PAMI*, 23(6):681–684, June 2001.

[22] R. Dayhot, P. Charbonnier, and F. Heitz. Robust visual recognition of color images. *Proceedings of CVPR-2000*, pages 685–690, June 2000.

[23] F. De la Torre and M. J. Black. Robust principal component analysis for computer vision. In *ICCV'01*, pages I: 362–369, 2001.

[24] F. De la Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, to appear, 2003.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[26] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, 2001.

[27] J. L. Edwards and H. Murase. Coarse-to-fine adaptive masks for appearance matching of occluded scenes. *MVA*, 10(5-6):232–242, April 1998.

[28] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395, 1981.

[29] G. Friedrich and Y. Yeshurun. Seeing people in the dark: Face recognition in infrared images. In *Biologically Motivated Computer Vision, BMCV 2002*, volume LNCS 2525, pages 348–359. Springer, November 2002.

[30] K. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(21):489–498, 1979.

[31] M. Gu and S. T. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Tech. report YALEU/DCS/RR-966, Department of Computer Science, Yale University, New Haven, 1993.

[32] P. Hall, D. Marshall, and R. Martin. Incremental eigenanalysis for classification. In *British Machine Vision Conference*, volume 1, pages 286–295, September 1998.

[33] P. Hall, D. Marshall, and R. Martin. Merging and spliting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1048, 2000.

[34] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[35] G. K. Humphrey and S. C. Khan. Recognizing novel views of three-dimensional objects. *Canadian journal of psychology*, 46:170–190, 1992.

[36] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis.* Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, 2001.

[37] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *ICPR00*, volume IV, pages 136–139, 2000.

[38] M. Jogan and A. Leonardis. Robust localization using the eigenspace of spinning-images. In *IEEE Workshop on Omnidirectional Vision*, pages 37–44, Hilton Head Island, SC, 2000.

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

[40] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78:99–118, 2000.

[41] X. Liu and T. Chen. Shot boundary detection using temporal statistics modeling. In *ICASSP 2002*, Orlando, FL, USA, May 2002.

[42] N. K. Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5(5):552–563, 1995.

[43] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information.* Freeman, San Francisco, CA, 1982.

[44] A. M. Martinez and A. C. Kak. PCA versus LDA. *PAMI*, 23(2):228–233, February 2001.

[45] J. Maver and A. Leonardis. Recognizing 2-tone images in grey-level parametric eigenspaces. *Pattern recognition letters*, 23(14):1631–1640, 2002.

[46] T. Meltzer. *Generalized Canonical Correlation Analysis for Object Recognition.* PhD dissertation, Vienna technical university, Vienna, Austria, October 2002.

[47] T. Meltzer and M. Reiter. Regularized kernel-CCA. In *Proc. of Workshop of the Austrian Association for Pattern Recognition*, pages 119–124, Graz, Austria, September 2002.

[48] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing*, 9:41–48, 1999.

[49] B. Moghaddam and A. P. Pentland. Probabilistic visual learning for object detection. In *ICCV95*, pages 786–793, 1995.

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

[51] H. Murase and S. K. Nayar. Illumination planning for object recognition using parametric eigenspaces. *PAMI*, 16(12):1219–1227, December 1994.

[52] H. Murase and S. K. Nayar. Image spotting of 3D objects using parametric eigenspace representation. In *SCIA95*, pages 325–332, 1995.

[53] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *IJCV*, 14(1):5–24, January 1995.

[54] S. K. Nayar, H. Murase, and S. A. Nene. Learning, positioning, and tracking visual appearance. In *IEEE International Conference on Robotics and Automation*, pages 3237–3244, San Diego, May 1994.

[55] S. K. Nayar, H. Murase, and S. A. Nene. Parametric appearance representation. *Early Visual Learning*, pages 131–160, 1996.

[56] K. Ohba and K. Ikeuchi. Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects. *PAMI*, 19(9):1043–1048, September 1997.

[57] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *PAMI*, 22(8):831–843, August 2000.

[58] V. Omladič. *Uporaba linearne algebre v statistiki.* Metodološki zvezki, FDV, 1997.

[59] A. P. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Seatle, WA, July 1994.

[60] D. I. Perret, M. W. Oram, and E. Ashbridge. Evidence accumulation in cell popoulations responsive to faces: an account of generalisation of recognition withouth mental transformations. In *Object recognition in man, monkey, and machine*, pages 111–145. MIT/Elsevier, 1998.

[61] R. P. N. Rao. Dynamic appearance-based recognition. In *CVPR97*, pages 540–546, 1997.

[62] P. J. Rousseuw and A. M. Leroy. *Robust Regression and Outlier Detection.* Wiley, New York, 1987.

[63] S. Roweis. EM algorithms for PCA and SPCA. In *Neural Information Processing Systems 10 (NIPS'97)*, pages 626–632, 1997.

[64] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, USA, December 1994.

[65] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[66] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(9):854–867, 1995.

[67] H. Sidenbladh, F. de la Torre, and M. J. Black. A framework for modeling the appearance of 3D articulated figures. In *AFGR00*, pages 368–375, 2000.

[68] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. America, A*, 4(3):519–524, 1987.

[69] D. Skočaj and A. Leonardis. Robust recognition and pose determination of 3-D objects using range images in eigenspace approach. In *Third International*

Conference on 3-D Digital Imaging and Modeling: proceedings, pages 171–178, Quebec City, Canada, May 2001.

[70] D. Skočaj, H. Bischof, and A. Leonardis. A robust PCA algorithm for building representations from panoramic images. In *Computer Vision – ECCV 2002*, volume IV, pages 761–775, May 2002.

[71] D. Skočaj and A. Leonardis. Acquiring range images of objects with non-uniform albedo using high-dynamic scale radiance maps. *Proceedings of ICPR'00*, pages 778–781, September 2000.

[72] M. J. Tarr. Rotating objects to recognize them: a case study of the role of viewpoint dependency in the recognition of three-dimensional objects. *Psychonomic Bulletin and Review*, 2(1):55–82, 1995.

[73] M. J. Tarr and H. H. Bülthoff. Image-based object recognition in man, monkey and machine. *Image-based object recognition in man, monkey, and machine*, pages 1–20, 1998.

[74] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. Technical report NCRG/97/010, Neural Computing Research Group, Aston University, September 1997.

[75] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

[76] M. Turk and A. P. Pentland. Eigenfaces for recognition. *CogNeuro*, 3(1):71–96, 1991.

[77] S. Ullman. Three-dimensional object recognition based on the combination of views. *Image-based object recognition in man, monkey, and machine*, pages 21–44, 1998.

[78] T. Wiberg. Computation of principal components when data are missing. In *Proc. Second Symp. Computational Statistics*, pages 229–236, 1976.

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

[80] S. Yoshimura and T. Kanade. Fast template matching based on the normalized correlation by using multiresolution eigenimages. In *Proceedings of IROS'94*, pages 2086–2093, 1994.