Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: donner@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-97 November 23, 2005

Fast parameter prediction for Active Appearance Models using Canonical Correlation Analysis – An AAM Matlab Implementation

René Donner, Georg Langs, Michael Reiter, Horst Bischof, Robert Sablatnig[1]

Abstract

Active Appearance Models (AAM) provide a compact statistical model of data encompassing both shape and texture variations. This report introduces a novel and fast search algorithm for AAMs based on *canonical correlation analysis (CCA)*. In contrast to the standard AAM matching approach CCA exploits the correlation between texture residuals and model parameters more efficiently. In a set of experiments using face and medical images we show that CCA based search consistently outperforms the convergence speed of the standard method by a factor of four. The time needed for training is reduced by 80%. Since the implementation effort of the standard approach and CCA are similar our results suggest that CCA can replace the standard AAM search.

Keywords. Active Appearance Models, Statistical Image Models, Canonical Correlation Analysis, Medical Imaging

[1]This work has been supported by the Austrian Science Fund under Grant P17083-N04 (AAMIR). Part of this work has been carried out as part of the K-plus Competence center ADVANCED COMPUTER VISION funded under the K plus program.

Contents

1 Introduction

This report documents the efforts to implement a flexible framework for active appearance models (AAMs). It was then used for the investigation of advanced parameter prediction techniques using canonical correlation analysis (CCA).

1.1 Motivation

Active appearance models, first introduced by Cootes et al. [8] in 1998, build a linear model of both shape and texture out of a set of training images and can then be used to find an object in a new image. Areas of successful applications include face recognition [8] and many areas of medical imaging, e.g. in spine radio-graphs [26], diaphragm *computer tomography* (CT) images [2] and hand radio-graphs [15].

During search, AAMs follow an iterative gradient descent approach to find the optimal model parameters. Hence their search speed and the accuracy of their results depend heavily on the quality of the parameter prediction at each iteration.

In this report we investigate the potential of canonical correlation analysis for improving this prediction step. CCA finds the directions of maximum correlation between two sets of data, thus forming a powerful regression tool.

1.2 Active Appearance Models

Appearance based image processing approaches represent both the internal structure of the object to be modeled like shape and texture changes as well as external influences like illumination and reflectance properties. Furthermore the encoded a priori knowledge leads to a robust search behaviour and ensures the results to be reasonable.

Several different approaches to modeling appearance exist. Turk and Pentland [33] employed an eigenspace representation of faces, introducing the term *eigenfaces* for this method.

Active shape models (ASMs) [12], although they do not model the whole texture, are the direct predecessors of AAMs. ASMs build a statistical linear model of the shape of the objects to be modeled as a set of points along the outline. In addition, they build a mean representation of a small texture patch orthonormal to the landmarks. The texture differences between these patches and the image are used during search to iteratively update the positions of all landmarks and then confine these positions to the model. The result is a good approximation of the objects' border.

AAMs use the same shape information as ASMs, but use the entire texture of the object. Each training object is warped to the mean shape, and out of the resulting shape-free textures a texture model is built. So instead of using only the mean texture information, the texture variability is modeled, too. The shape and texture models are then combined, yielding the active appearance model, which is able to construct photo-realistic images of the training objects and all linear interpolations using only a handful of parameters, which are the PCA coefficients of the underlying model plus additional modes for rotation, scaling and contrast/brightness. During search these parameters are

estimated using an iterative gradient descent optimization, with the texture difference between model and image as objective.

AAM History and New Developments Since their initial proposition AAMs many modifications and improvements were developed. Cootes in [10] gives a brief time line of the milestones in AAM evolution. The training scheme introduced in [13], which has replaced the initial *multivariate linear regression* (MLR) approach, will be investigated in this report. Novel texture representations were investigated [31], leading to the incorporation of feature based approaches to AAMs [28].

Several publications focus on how to model and combine the shape and the texture information. *Direct AAMs* [21] store only the texture parameters as model parameters and try to implicitly map the according shape parameters. *Shape AAMs* [11] also separate shape and texture parameters, as they only update the shape parameters trough regression and then map the encountered texture to the texture parameters directly. Non-linear models utilising kernel-PCA (*principal component analysis*) were proposed in [27] . Interesting work has also been conducted on the topic of robust AAMs [3]. Small adjustments to the search routine aimed at improving robustness [7].

Due to the limited variance of global models several attempts have been undertaken to incorporate additional degrees of freedom into the optimization process, like the use of linked sub-models [26].

Applications AAMs have been employed in various domains like face modelling [16], studying human behavior [22] and human computer interaction [1]. The main area of application is medical imaging, like the segmentation of cardiac images obtained through *magnetic resonance imaging* (MRI) [24] or the diaphragm in CT data [2], and registration in functional heart imaging [30]. In [29] an extensive overview of existing applications is given.

1.3 Canonical Correlation Analysis

Canonical Correlation Analysis a versatile method that is especially well suited for relating two sets of measurements (signals) [4]. Like PCA, CCA also reduces the dimensionality of the original signals, since only a few factor-pairs are normally needed to represent the relevant information; unlike PCA, however, CCA takes into account the relationship between two signal spaces (in the correlation sense), which makes them better suited for regression tasks than PCA.

Recent work on CCA includes the formulation of a unified approach to PCA, *partial least squares* (PLS), MLR and CCA [4]. Investigations into developing robust CCA were conducted by [17].

1.4 Contributions and Structure

The main contribution of this report is the utilization of CCA for parameter prediction in AAM search. The accuracy of the predictions outperforms the standard approach, leading to a more compact regression model and faster search convergence.

Sec. 2 provides a comprehensive description of active appearance models. It describes the prerequisites for building AAMs and how to obtain the model, mainly through the utilization of normalization, *Procrustes analysis* for alignment and *principal component analysis* for dimensionality reduction. Then the process of regression on the difference images and the parameter displacements by the standard regression approach is described, leading to a formulation of the search procedure.

Sec. 3 first introduces canonical correlation analysis in Sec. 3.1 in order to derive an improved parameter prediction scheme for AAMs in Sec. 3.2. Empirical results will be presented in Sec. 4. Sec. 5 provides a summary and an outlook. Finally, the appendix contains the documentation of the AAM software implementation.

1.5 Nomenclature

CCA Related Symbols

N	number of samples
p	dimensionality of the input space
q	dimensionality of the output space
k	number of factors or eigenspace dimension
$\mathbf{w}$	parameter vector
$\rho(\mathbf{w})$	objective function
$\mathbf{w}^*$	extremum (stationary) point of $\rho(\mathbf{w})$
$\mathbf{X}$	$p \times N$ training sample matrix
$\mathbf{Y}$	$q \times N$ output sample matrix
$\mathbf{x} = (x_1, \ldots, x_p)^T$	random vector
$\mathbf{x}_{[k]} = (x_1, \ldots, x_k)^T$	projection onto the first k dimensions

AAM Related Symbols

n	number of training samples
w	width of training images
h	height of training images
$\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$	matrix containing the training shapes
$\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_n)$	matrix containing the training textures
$\mathbf{P}_s$	eigenvectors of the shape eigenspace
$\mathbf{P}_g$	eigenvectors of the texture eigenspace
$\mathbf{P}_c$	eigenvectors of the combined shape/texture eigenspace
$\mathbf{p} = (\mathbf{c}^T \vert \mathbf{t}^T \vert \mathbf{u}^T)$	model parameter vector consisting of sub-vectors $\mathbf{c}, \mathbf{t}$ and $\mathbf{u}$
$\mathbf{R}$	regression matrix
n	number of training samples
n	number of training samples
$\mathbf{r}(\mathbf{p})$	texture residual

2 Active Appearance Models

Active appearance models are a method to build a statistical model of a priori knowledge and then to utilize this model to perform a robust search on a hitherto unseen image [14]. This section will detail the steps involved in training and using AAMs.

2.1 Using Models in Image Segmentation

Active appearance models belong to the family of methods based on the alignment approach, which try to maximize a certain measure of fit (e.g. texture residual) between the representation given by the model and the image. The necessary transformation can either be applied to the image or the model, the later being the case for AAMs.

The type of model employed in AAMs is a statistical one, the parameters of which are learnt during training. The advantage of statistical models over more generally deformable ones is that applying constraints on the parameters, which are estimated from training data, ensures that the model output remains reasonable, e.g. the image still looks like a human face. This leads to higher robustness to noise in the images.

In addition to that, the model is a global one, capturing the correlation of different parts of the model. This allows to predict the expected appearance of parts of the object which are missing or occluded in the image.

2.2 Training

The concept of active appearance models as described in [9] is based on the idea of combining both shape and texture information of objects to be modeled into a common statistical model, that can express the variations encountered in the training data.

For fitting the model the difference between the current model state to the image is used as objective function, which is assumed to be approximable by a quadratic function. This assumption leads to the use of an iterative Gauss-Newton optimization procedure, resulting in a search result closely resembling the underlying image.

To provide the necessary parameter predictions at each iteration the relation between the difference images (between model and image) and the corresponding parameter offsets has to be learnt.

2.2.1 Modelling Shape

Landmarks and annotation The prerequisites for building an active appearance model are images of the objects to be modeled. In addition to the images, which provide the information about the texture (gray level intensities), information about the shape of the object is needed. This has to be provided through a list of points corresponding to samples of the contours of interest.

These contours have to be provided by a human expert, who manually selects the contours or points of interest, preferably with semi-automated tools.

For further processing the contours of each image i first have to be encoded as sets of points in the form of $\mathbf{x}_i = (x_1, \ldots, x_m, y_1, \ldots, y_m)^T$. To extract equi-distanced points

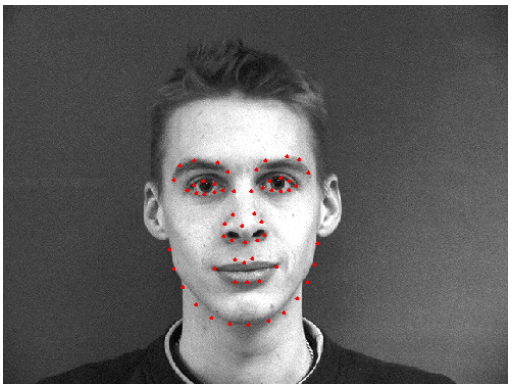
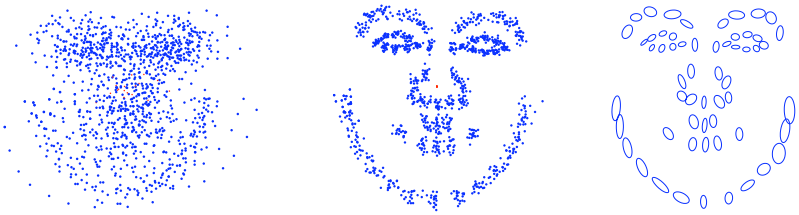


Figure 1: Example of an annotated face image. Image originating from [25].



(a) (b) (c)

Figure 2: Landmarks for 20 face images (a) before and (b) after alignment. In (c) the variance of the individual landmarks is depicted. Ellipses fitted using [18].

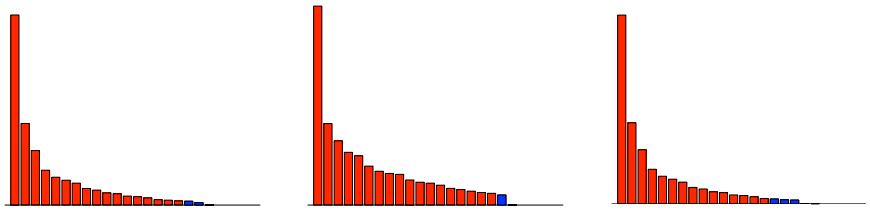
from curves, which only contain information orthogonal to the curve, i.e. do not move along the curve between different training samples, the approach described in [32] is used.

The resulting m points are called landmarks and contain all the shape information that is used by the AAM. For example, on annotated face images usually the contours of the eyes, the nose and mouth and the chin are marked, while for the metacarpal bones the outline is annotated (Fig. 1).

Shape alignment using Procrustes analysis The m landmarks, i.e. the shape, of the n training images are stored in vectors $\mathbf{s}_i = (x_1, \ldots, x_m, y_1, \ldots, y_m)^T$. Their length gets normalized before they are aligned using Procrustes analysis [20]. Procrustes analysis[1] ensures that the distribution of positions of one landmark is not due to translation, rotation or scale. See Fig. 2 for a display of the unaligned and aligned landmarks.

All shapes can now be combined in an $m \times n$ matrix $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$, whose dimensionality will be reduced in the next step, yielding the parameters of the shape model.

[1]Procrustes analysis is named after Procrustes, a character of Greek mythology, who stood at the sides of roads with a bed, waiting for tired travelers. As soon as they lied down in his bed, he would torture them by pulling at or cutting off their limbs, until they would fit his bed.



(a) (b) (c)

Figure 3: Eigenvalues corresponding to (a) the shape eigenvectors $\mathbf{P}_s$, (b) the texture eigenvectors $\mathbf{P}_g$ and (c) the combined model eigenvectors $\mathbf{P}_c$, with the eigenvalues responsible for 98% variance highlighted

Formulating the shape model By applying PCA to the data set $\mathbf{S}$ we obtain the eigenvectors $\mathbf{P}_s$ which are sorted by descending eigenvalues. As stated above the eigenvalues are proportional to the variance of the signal. In image processing it is generally assumed (for images that display little noise), that the eigenvectors corresponding to the smallest eigenvalues model the noise. So a index k is chosen so that

$$\sum_{i=1}^{k} \lambda_i \geq \alpha \sum \lambda_i,$$

with α commonly set to 98%, and only the projections of $\mathbf{S}$ onto the first k eigenvectors are subsequently used. A bar plot of the eigenvalues with an indication of the 98% variance limit is shown in Fig. 3.

2.2.2 Modelling Texture

The mean shape $\bar{\mathbf{s}}$ is needed to establish a so-called normalized reference frame. The training images get warped to the mean shape using piecewise affine warping or more elaborated warping methods like b-spline warping, yielding shape free images which are called the *texture* of the objects. For an overview of warping methods see [19]. The textures are normalized yielding the texture vectors $\mathbf{g_i}$, which are of equal length, so they can be stored in a $(w * h \times n)$ matrix $\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_i, \ldots, \mathbf{g}_n)$, where w and h denote the common width and height of the shape free images. Fig. 4 depicts 2 input images and their shape free representation.

By applying PCA to the data sets $\mathbf{S}$ and $\mathbf{G}$ we obtain the eigenvectors $\mathbf{P}_s$ and $\mathbf{P}_g$, which are sorted by descending eigenvalues (Fig. 3). Only the leading a_s respectively a_g eigenvectors are used which cover 98% variance of the data, as the remaining 2% are considered as noise.

A given training image $\langle \mathbf{s}_i, \mathbf{g}_i \rangle$ can now be described by parameters $\langle \mathbf{b}^s, \mathbf{b}^g \rangle$ using a linear model:

$$\begin{aligned} \mathbf{s}_i &= \bar{\mathbf{s}} + \mathbf{P}_s \mathbf{b}_i^s \\ \mathbf{g}_i &= \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_i^g \end{aligned}$$

(a) (b) (c) (d)

Figure 4: 2 input faces before (a,c) and after (b,d) being warped to the mean shape. Note that the brightness of the images appears different due to a subsequent normalization step.

where $\overline{\mathbf{s}}, \overline{\mathbf{g}}$ are the mean vectors and $\mathbf{P}_s, \mathbf{P}_g$ constitute the orthogonal modes of variation.

The parameter vectors $\mathbf{b}_i^s$ and $\mathbf{b}_i^g$ are much shorter than the original vectors $\mathbf{s}_i, \mathbf{g}_i$, illustrating the dimensionality reduction performed by the PCA. A typical example would be $\mathbf{b}_i^s$ and $\mathbf{b}_i^g$ being of lengths 50 and 10000, while $\mathbf{s}_i$ and $\mathbf{g}_i$ being of lengths 20 and 30.

2.2.3 Combining the Shape and Texture Models

The resulting shape and texture parameters may still be correlated, so to eliminate the correlations and gain an even more compact model PCA is applied once more to the concatenated vector

$$\mathbf{b}_i = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_i^s \\ \mathbf{b}_i^g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{P}_s^T (\mathbf{s}_i - \overline{\mathbf{s}_i}) \\ \mathbf{P}_g^T (\mathbf{g}_i - \overline{\mathbf{g}_i}) \end{pmatrix}$$

where $\mathbf{W}_s$ is a diagonal matrix scaling the shape parameters to allow for the difference in range between shape and texture space. $\mathbf{W}_s$ is set to the ratio of the total texture variation to the total shape variation, $\mathbf{W}_s = \mathbf{I} \sum \lambda^g / \sum \lambda^s$.

Applying PCA to these vectors yields the final combined linear model

$$\mathbf{b} = \mathbf{P}_c \mathbf{c}. \tag{1}$$

The shape free images and the corresponding shapes defining the deformation of the texture can thus be expressed directly using $\mathbf{c}$ by

$$\begin{aligned} \mathbf{s} &= \overline{\mathbf{s}} + \mathbf{P}_s \mathbf{W}_s^{-1} \mathbf{P}_{cs} \mathbf{c}_i \\ \mathbf{g} &= \overline{\mathbf{g}} + \mathbf{P}_g \mathbf{P}_{cg} \mathbf{c}, \end{aligned}$$

where

$$\mathbf{P}_c = \begin{pmatrix} \mathbf{P}_{cs} \\ \mathbf{P}_{cg} \end{pmatrix}.$$

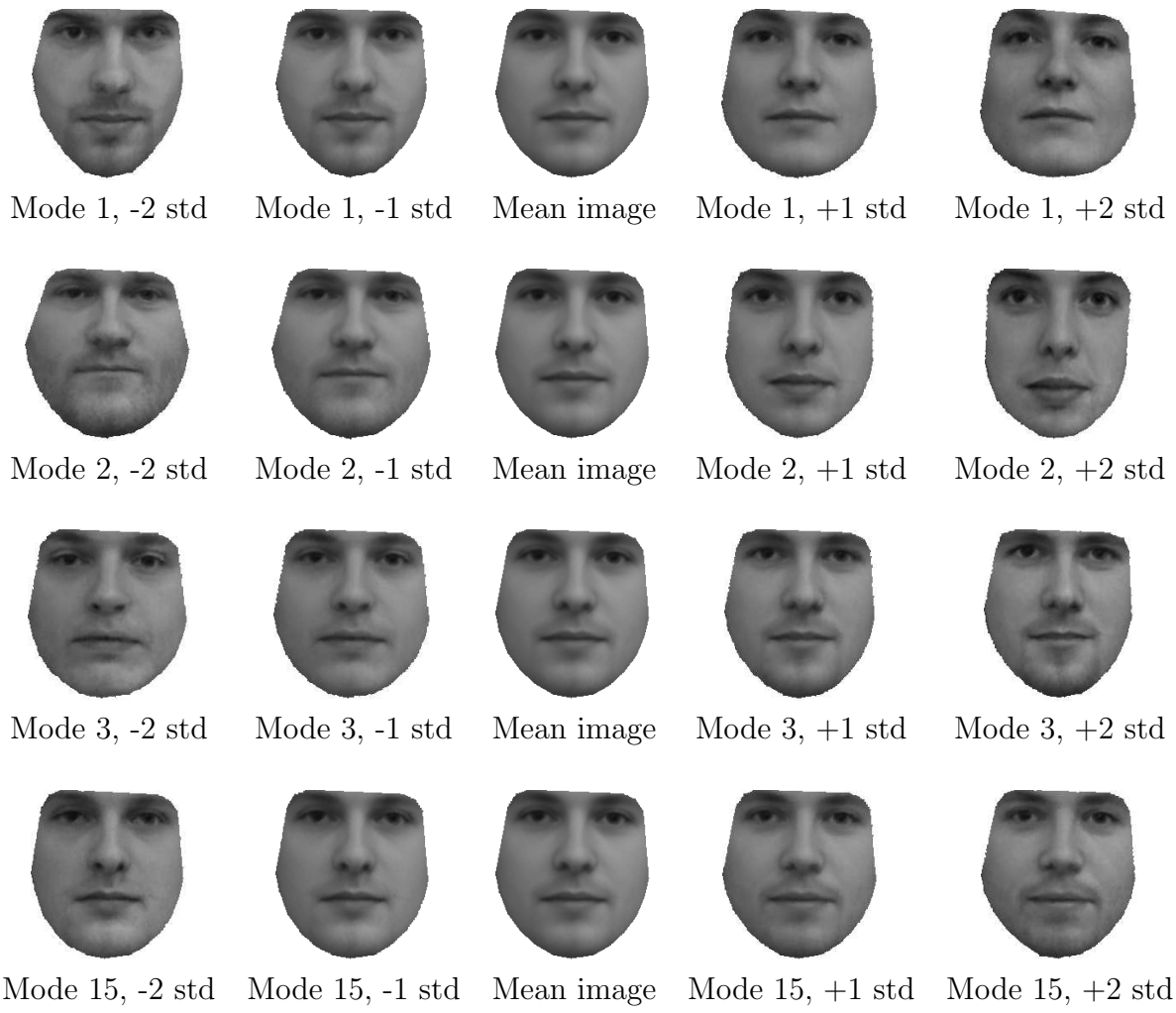


Figure 5: Modes of the resulting AAM model. Mode 1, encompassing the largest variance, is depicted at the top, followed by mode 2, 3 and the last mode (15). For each mode the resulting appearance at parameter values of (-2std, -1std, 1std, 2std) are shown.

The resulting model (modes depicted in Fig. 5) represents the shape and texture variation of the modeled objects utilizing a single parameter vector $\mathbf{c}$. Each element of $\mathbf{c}$ controls one mode of combined texture and shape variation, with the first modes being responsible for the highest variation, in descending order. Examples for the first three and the last mode are depicted in (Fig. 5).

In addition to these modes we need to account for basic operations as translation, scaling and rotation, further texture contrast and brightness. We thus introduce additional parameter vectors $\mathbf{t}$ and $\mathbf{u}$, resulting in a combined parameter vector

$$\mathbf{p} = (\mathbf{c}^T, \mathbf{t}^T, \mathbf{u}^T). \tag{2}$$

$\mathbf{c}$ denotes the parameter vector from Eq. (1) controlling shape and texture.

$\mathbf{t} = (s_x, s_y, t_x, t_y)^T$ is a linear parameter vector controlling rotation θ, scaling s and

translation (t_x, t_y), with $s_x = (s\cos\theta - 1)$ and $s_y = s\sin\theta$.

$\mathbf{u} = (u_1, u_2)^T = (\alpha - 1, \beta)^T$ controls gray-level image contrast α and brightness β of the texture according to $\mathbf{g}_u(\mathbf{g}) = (1 + u_1)\mathbf{g} + u_2\mathbf{I}$.

Note that all parameters express linear relations, i. e. AAMs are linear models.

2.3 AAM Training Search

Provided we have a trained AAM where model parameters $\mathbf{p}$ generate synthetic images $\mathbf{I}_{model}(\mathbf{p})$. The standard search for an optimal match minimizes the difference between a given image $\mathbf{I}_{image}$ and the reconstructed image $\mathbf{I}_{model}(\mathbf{p})$. The search for the model parameters $\mathbf{p}$ can be guided by using knowledge about how the difference images correlate with the parameter displacements. This knowledge is obtained during training.

During each search step the current image residual between the model texture $\mathbf{g}_m(\mathbf{p})$ and the sampled image patch $\mathbf{g}_s(\mathbf{p})$ (warped to the mean shape) is computed using

$$\mathbf{r}(\mathbf{p}) = \mathbf{g}_s(\mathbf{p}) - \mathbf{g}_m(\mathbf{p}). \tag{3}$$

The search procedure aims at minimizing the sum of square (pixel) error

$$\frac{1}{2}\mathbf{r}(\mathbf{p})^T\mathbf{r}(\mathbf{p}). \tag{4}$$

Following the standard Gauss-Newton optimization method one approximates (linearizes) Eq. 3 using the first-order Taylor expansion

$$\mathbf{r}(\mathbf{p} + \delta\mathbf{p}) \approx \mathbf{r}(\mathbf{p}) + \frac{\partial\mathbf{r}}{\partial\mathbf{p}}\delta\mathbf{p},$$

with the ij^{th} element of matrix $\frac{\partial\mathbf{r}}{\partial\mathbf{p}}$ being $\frac{\partial r_i}{\partial p_j}$.

Building the derivative of Eq. 4 w.r.t. $\mathbf{p}$ and setting it to zero gives

$$\delta\mathbf{p} = -\mathbf{R}\,\mathbf{r}(\mathbf{p}), \quad \text{where} \quad \mathbf{R} = \left(\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T\frac{\partial\mathbf{r}}{\partial\mathbf{p}}\right)^{-1}\frac{\partial\mathbf{r}}{\partial\mathbf{p}}^T = \left(\frac{\partial\mathbf{r}}{\partial\mathbf{p}}\right)^{\dagger}, \tag{5}$$

with † denoting the pseudo-inverse. Instead of recalculating $\frac{\partial\mathbf{r}}{\partial\mathbf{p}}$ at every step it is computed once during training using numeric differentiation.

During training each parameter is displaced from its optimal value in h steps from -1 to +1 standard deviations, and a weighted average of the resulting difference images over the training set is built:

$$\frac{dr_i}{dp_j} = \sum_h \omega(\delta p_{jh})\frac{(r_i(\mathbf{p} + \delta p_{jh}) - r_i(\mathbf{p}))}{\delta p_{jh}}$$

During the actual search, each iteration updates the model parameters using $\mathbf{p}_{next}(s) = \mathbf{p}_{current} - s\,\delta\mathbf{p}_{predicted}$, with $\delta\mathbf{p}_{predicted} = -\mathbf{R}\,\mathbf{r}_{current}$ and s being a scaling factor sequentially chosen from $\mathbf{s}_{steps} = \langle 1, 0.5, 1.5, 0.25, 0.1, 2, 0.025, 0.01\rangle$, as proposed in [14]. At each of these scaling steps, the image patch is compared to the synthesized image $\mathbf{I}_{model}$, which is computationally expensive. Let $E(\mathbf{p}_{current}) = |\mathbf{r}(\mathbf{p}_{current})|^2 = |\mathbf{g}_s - \mathbf{g}_m|^2$ be the error of

the current model. An iteration is declared successful for the first step s to produce an error $E(\mathbf{p}_{next}(s)) < E(\mathbf{p}_{current})$. $\mathbf{p}_{current}$ is then set to $\mathbf{p}_{next}(s)$ and the search continues with the next iteration. If no $\mathbf{p}_{next}(s)$ better then $\mathbf{p}_{current}$ can be found, convergence is declared and $\mathbf{p}_{current}$ is the best estimate for the model parameters. As will be shown in subsection 4.1 our approach eliminates the need for using different step sizes, as the parameter predictions are more accurate.

3 Advanced Parameter Prediction

In this section a new parameter prediction scheme based on CCA will be developed which leads to more accurate predictions and thus faster convergence of the search process.

The most crucial part in AAMs apart from the actual model itself is the issue of parameter prediction. To minimize the texture residual the parameter update has to be computed from the difference image at each iteration. Different approaches used for this prediction will lead to different behavior in robustness, performance and accuracy.

Canonical correlation analysis, being a method to find the optimal relation in the sense of maximized correlation of the projections, is very well suited to perform this regression on the difference images and the corresponding parameter differences.

3.1 Canonical Correlation Analysis

Given two zero-mean random variables $\mathbf{x} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$, CCA finds pairs of directions $\mathbf{w}_x$ and $\mathbf{w}_y$ that maximize the correlation between the projections $x = \mathbf{w}_x^T\mathbf{x}$ and $y = \mathbf{w}_y^T\mathbf{y}$ (in the context of CCA, the projections x and y are also referred to as *canonical variates*). This is illustrated in Fig. 6.

More formally, the directions can be found as maxima of the function

$$\rho = \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}} = \frac{E[\mathbf{w}_x^T\mathbf{x}\mathbf{y}^T\mathbf{w}_y]}{\sqrt{E[\mathbf{w}_x^T\mathbf{x}\mathbf{x}^T\mathbf{w}_x]E[\mathbf{w}_y^T\mathbf{y}\mathbf{y}^T\mathbf{w}_y]}}, \tag{6}$$

$$\rho = \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y}{\sqrt{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{C}_{yy}\mathbf{w}_y}}. \tag{7}$$

whereby $\mathbf{C}_{xx} \in \mathbb{R}^{p\times p}$ and $\mathbf{C}_{yy} \in \mathbb{R}^{q\times q}$ are the *within-set covariance matrices* of $\mathbf{x}$ and $\mathbf{y}$, respectively, while $\mathbf{C}_{xy} \in \mathbb{R}^{p\times q}$ denotes their *between-set covariance matrix*. A number of at most $k = \min(p, q)$ factor pairs $\langle \mathbf{w}_x^i, \mathbf{w}_y^i \rangle, i = 1, \ldots, k$ can be obtained by successively solving

$$\mathbf{w}^i = (\mathbf{w}_x^{iT}, \mathbf{w}_y^{iT})^T = \arg\max_{(\mathbf{w}_x^i, \mathbf{w}_y^i)}\{\rho\} \tag{8}$$

$$\text{subject to} \quad \rho(\mathbf{w}_x^j, \mathbf{w}_y^i) = \rho(\mathbf{w}_x^i, \mathbf{w}_y^j) = 0 \;\; j = 1, \ldots, i-1 \tag{9}$$

The factor pairs $\mathbf{w}^i$ can be obtained as solutions (i.e., eigenvectors) of a generalized eigenproblem (for details see e.g., [23]). The extremum values $\rho(\mathbf{w}^i)$, which are referred to as *canonical correlations*, are obtained as the corresponding eigenvalues.

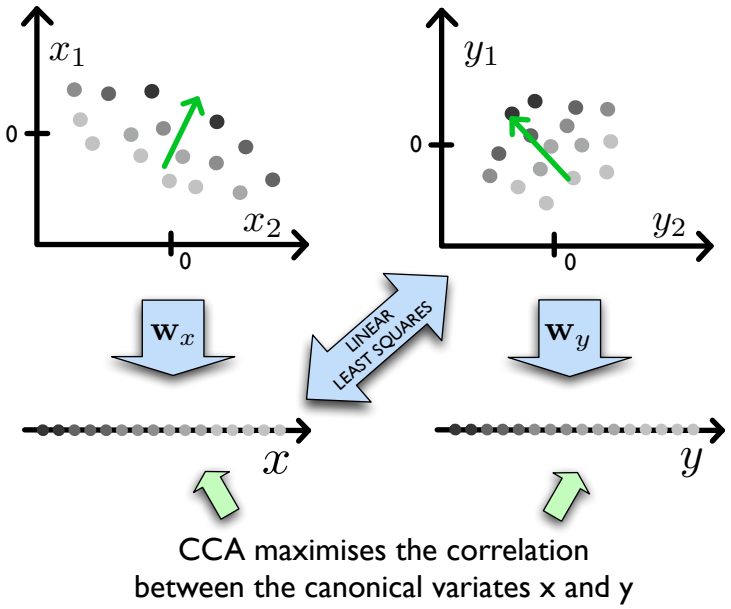


Figure 6: Principle of canonical correlation analysis. CCA finds the subspaces of maximum between-set correlation. Regression is performed on the projections of signal space x and the signal space y (not projected).

By employing CCA, we perform regression on only a small number (compared to the original dimensionality of the data) of linear features, i.e. derived linear combinations of original response variables $\mathbf{y}$. Thus, CCA can be used to compute the (reduced) rank-m regression parameter matrix by using only $m < k$ factor pairs. Thereby, in contrast to standard multivariate regression CCA takes advantage of the correlations between the response variables to improve predictive accuracy [6].

Analogously to standard multivariate regression, CCA can also directly be formulated as a linear least squares problem. It can be easily shown that minimizing

$$\begin{aligned} RSS(\mathbf{w}_x, \mathbf{w}_y) &= E[(\mathbf{w}_x^T\mathbf{x} - \mathbf{w}_y^T\mathbf{y})^2] && (10)\\ &= E[\mathbf{w}_x^T\mathbf{x}\mathbf{x}^T\mathbf{w}_x] && (11)\\ &\quad -2E[\mathbf{w}_x^T\mathbf{x}\mathbf{y}^T\mathbf{w}_y] + E[\mathbf{w}_y^T\mathbf{y}\mathbf{y}^T\mathbf{w}_y] && (12)\\ &= \mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x && (13)\\ &\quad -2\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y + \mathbf{w}_y^T\mathbf{C}_{yy}\mathbf{w}_y && (14) \end{aligned}$$

subject to the constraints

$$\begin{aligned} \mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x &= 1, && (15)\\ \mathbf{w}_y^T\mathbf{C}_{yy}\mathbf{w}_y &= 1. && (16) \end{aligned}$$

yields the first canonical factor pair. An iterative (online) CCA algorithm based on the above formulation is described in [5]. This formulation also allows to successively obtain multiple factor pairs.

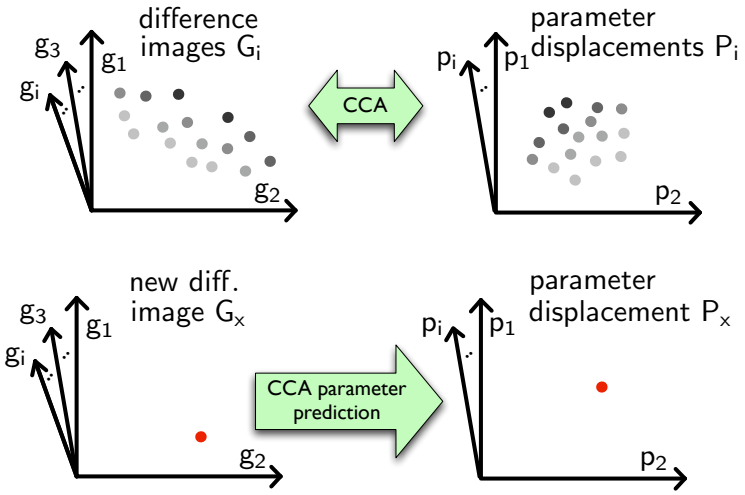


Figure 7: Using CCA for parameter prediction in AAMs. Regression is performed during training (top), allowing to obtain a prediction for a given difference image during search (bottom).

3.2 Utilizing CCA for AAM Parameter Prediction

Our concept is based on the idea of finding the directions of maximum correlation between the difference images $\mathbf{G}$ and the displacement predictions $\delta\mathbf{P}$. The resulting CCA prediction function $\delta\mathbf{p}(\mathbf{r})$ is used during the AAM search, which is otherwise analogous to the standard method.

Applying CCA to this data means to maximize

$$\rho = \frac{\mathbf{w}_g^T \mathbf{C}_{gp} \mathbf{w}_p}{\sqrt{\mathbf{w}_g^T \mathbf{C}_{gg} \mathbf{w}_g \mathbf{w}_p^T \mathbf{C}_{pp} \mathbf{w}_p}}, \tag{17}$$

yielding the canonical factors pairs $\mathbf{W}_g = (\mathbf{w}_g^1, \ldots, \mathbf{w}_g^k)$ and $\mathbf{W}_p = (\mathbf{w}_p^1, \ldots, \mathbf{w}_p^k)$, where $k = \min(l, n, m)$.

To improve generalization we use only the first k^* factors pairs, setting k^* so that only correlations larger than a certain threshold are used. The resulting factors $\mathbf{W}_g = (\mathbf{w}_g^1, \ldots, \mathbf{w}_g^{k^*})$ and $\mathbf{W}_p = (\mathbf{w}_p^1, \ldots, \mathbf{w}_p^{k^*})$ are the linear combinations which predict $\delta\mathbf{p}$ best. This means that we are performing rank-k^* regression on the leading canonical variates.

To be able to later match new difference images to displacement vectors, the least squares mapping depicted in Fig. 7 is used:

$$\mathbf{l} = \mathbf{G}_{proj}^{\dagger} \mathbf{P} \tag{18}$$

where $\mathbf{G}_{proj}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{G}_{proj}$.

During search a new parameter prediction has to be obtained at each iteration. Instead of using Eq. 3, the prediction can now be obtained by

$$\delta\mathbf{p} = \mathbf{l}\mathbf{r}_{proj} \quad \text{where} \quad \mathbf{r}_{proj} = \mathbf{W}_g^T \mathbf{r}. \tag{19}$$

This yields the displacement prediction function $\delta\mathbf{p}(\mathbf{r})$:

$$\delta\mathbf{p}(\mathbf{r}) = \mathbf{l}\mathbf{W}_g^T\mathbf{r}. \tag{20}$$

As $\mathbf{R}_{cca} = \mathbf{l}\mathbf{W}_g^T$ can be computed once during training the final formulation of the prediction function is

$$\delta\mathbf{p}(\mathbf{r}) = \mathbf{R}_{cca}\mathbf{r}, \tag{21}$$

allowing for an AAM search utilizing the correlations between parameter displacement and image difference as captured by CCA. The computation of these predictions is as fast as for the standard approach, therefore one step of an iteration of CCA search is as fast as one step using the standard approach.

4 Experiments

Number of images and cross validation To be able to build a meaningful statistical model of shape and texture the number of required images depends on the variation of both properties throughout the images.

The images of the data set are separated into training and test sets, and only the training images are used to build and train the AAM. If training images were to be used for testing the result would ideally be a perfect fit, but as with all pattern recognition methods and model based approaches the main interest lies in the performance of the algorithm on unseen data, i.e. the test set.

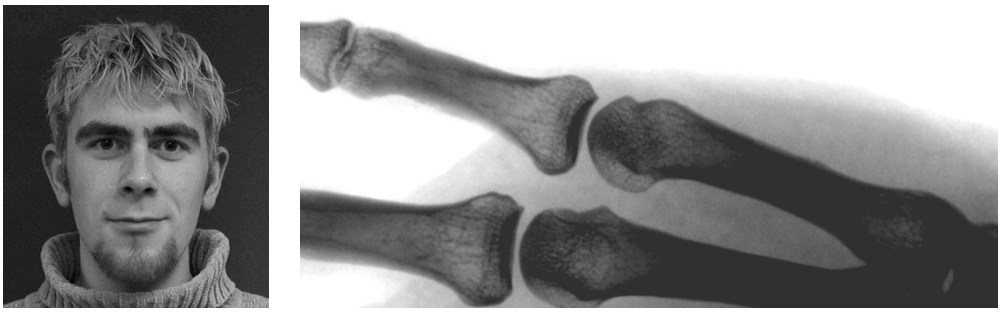
In our experiments we used at least 30 images to build the models while testing was performed on the remaining images, while for final numerical results we run leave-one-out tests.

4.1 Setup

Experiments were conducted on 36 face images [25] and 36 metacarpal bone images manually annotated by a medical expert (Fig. 8). The algorithm was evaluated using 4-fold cross validation. Following the standard AAM training scheme, a set of difference images and corresponding parameter displacements were obtained by randomly perturbing the AAM modes in the interval -1 to +1 standard deviation. While the calculation of $\mathbf{R}$ (cf. Eq. 5) by numerical differentiation requires separate variation of each AAM mode, CCA-AAM training allows simultaneous variation of all modes.

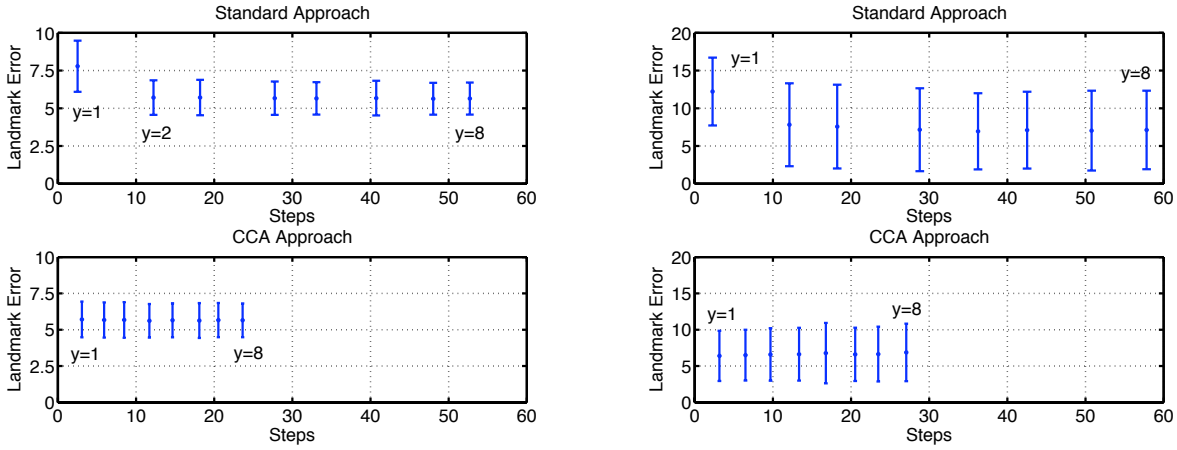
To compare search performance in both cases AAM search was performed on the test data using varying lengths of $\mathbf{s}_{steps}$. Scaling factors available during search are chosen by using the first y elements of $\mathbf{s}_{steps}$. As a performance measure we use the total number of steps accumulated over all iterations (cf. subsection 2.3).

Searches were initialized using equal initialization (randomly generated by translations of up to 10 pixels and mean shape and texture) for both approaches. 180 search results provide the data for each of the result bars plotted.



(a) (b)

Figure 8: Types of data used for evaluation. (a) Faces images and (b) medical images



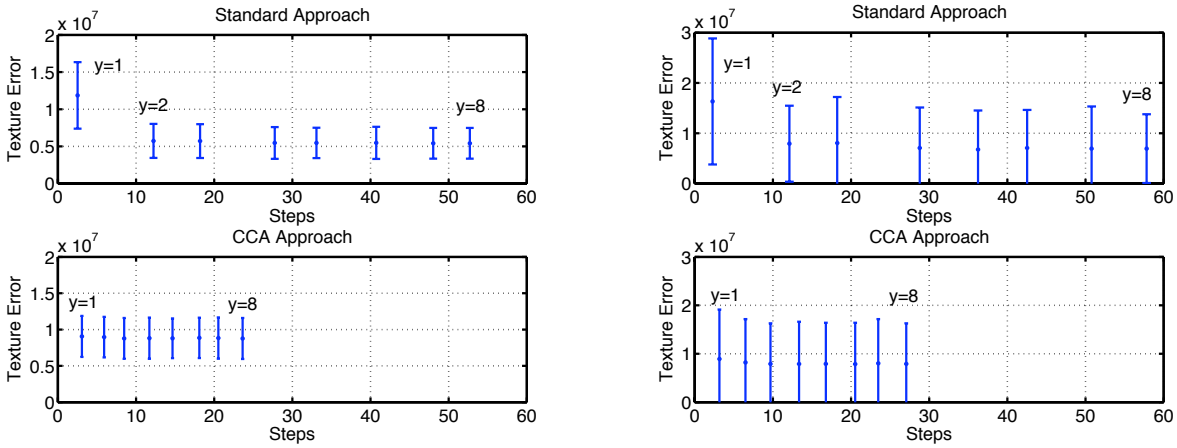
(a) Faces (b) Metacarpal bones

Figure 9: Comparison of landmark errors. The 8 bars correspond to y, i.e. the length of s_{steps}, ranging from 1 to 8, from left to right. Note how the CCA yields better (bones) or equal (faces) results faster (at ≈3 steps) then the standard approach (at ≈12 steps).

4.2 Results

Faster training CCA-AAM training needs fewer synthetic difference images. Using 24 modes for face data and 18 for bone data, 6480 synthetic face images and 4860 synthetic bone images were generated for standard training. For CCA training no improvement was observed when using more than 200 synthetic difference images. Thus, although the computation of the CCA is expensive, training is still considerably faster than standard training. For a Matlab implementation on a PowerMac G5 1.8GHz a speed-up factor of 4.9 and 3.5 was achieved.

Faster convergence with equal accuracy In Fig. 9 the mean landmark error (point to point distance) over the corresponding number of overall search steps until convergence is depicted. Error bars are 1 standard deviation. The 8 results plotted correspond to y ranging from 1 to 8 as stated above. In contrast to full rank of 24 (faces) and 18 (bones)



(a) Faces (b) Metacarpal bones

Figure 10: Comparison of texture errors. The 8 bars correspond to y (length of s_{steps}), ranging from 1 to 8, from left to right. Again, the CCA approach yields its best results already at $y = 1$ at ≈3 steps while the standard approach needs ≈12 steps for equal error levels.

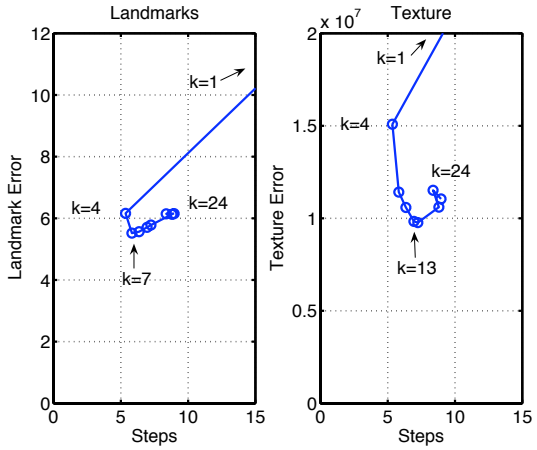


Figure 11: Influence of CCA regression rank. Mean landmark error against the number of steps for different choices of k (number of factors for CCA regression) for the face data set.

CCA employed ranks of 10 and 9 respectively.

It can be observed that the CCA performance is considerably better than the one of the R-matrix. Already with $\mathbf{s}_{steps} = \langle 1 \rangle$, i.e. no scaling of $\delta\mathbf{p}$, the CCA approach yields its best result, in 3.07 steps for the faces data and 3.16 steps for the metacarpals, respectively. The standard approach needs at least $\mathbf{s}_{steps} = \langle 1, 0.5 \rangle$ to perform equally well, requiring 12.23 and 12.11 steps. The CCA approach is thus 3.98 and 3.83 times faster. The mean texture errors in Fig. 10 show a similar picture. Again, the CCA approach yields its best results already at $y = 1$. The standard approach is dependent on the availability of further

Faces	Standard	CCA
Training samples	6480	200
Mean landmark error	5.7	5.7
Mean texture error ($\cdot 10^6$)	5.7	9.1
Necessary search steps	12.23	3.07
Search speed-up	1.00	**3.98**

Bones	Standard	CCA
Training samples	4860	200
Mean landmark error	7.8	6.4
Mean texture error ($\cdot 10^6$)	7.9	8.9
Necessary search steps	12.11	3.16
Search speed-up	1.00	**3.83**

Table 1: Result summary. Mean landmark and texture errors and corresponding number of search steps for both data sets.

scaling factors ($\mathbf{s}_{steps} = \langle 1, 0.5 \rangle$) to equal this performance. The results are summarized in Tab. 1.

Influence of rank reduction In a separate experiment the influence of rank reduction by CCA was investigated. In Fig. 11 the dependency of the mean landmark errors after search convergence is depicted for rank k set to $1, 4, \ldots, 24$ for the face data set. It can be seen that for $k = 7$ the search yields the lowest landmark errors, and for $k = 13$ the lowest texture errors. The number of necessary steps is lower than for full rank in both cases.

5 Conclusion and Outlook

CCA-AAMs introduce a search algorithm based on *canonical correlation analysis (CCA)*. CCA efficiently models the dependencies between image residuals and parameter correction. Taking advantage of the correlations between these two signal spaces CCA makes sensible rank reduction possible. It accounts for noise in the training data and thereby yields significant improvements of the AAM search performance in comparison to the standard search approach. After computing CCA, linear regression is performed on a small number of linear features which leads to a more accurate parameter prediction during search, eliminating the need for the expensive variable step size search scheme employed in the standard approach.

Empirical evaluation on two data sets shows that the CCA-AAM search approach is up to 4 times faster than the standard approach. As fewer training samples are needed, training is up to 5 times faster. Our approach can be adopted in most of the existing extensions of the original AAM search approach based on linear regression. Future research will focus on the use of non-linear CCA (Kernel CCA) and the application of CCA to obtain more compact and more descriptive active appearance models.

A Software Documentation and Users Guide – `matlAAM` Documentation

In the following section you will find a quick-start guide for the AAM implemented in Matlab. This will enable you to build and test your first AAM within an hour. Section **??** provides more insight into `matlAAM`, while *all* source code file are extensively documented. Please see `matlaam/htmldocs/index.html` for an automatically generated, hyper-linked version of the source code documentation!

The implementation closely follows the AAM description in [9].

A.1 See an Example in Action

1. If you use `matlAAM` for the first time and you are using a system other than Windows or Mac OS X, please compile `matlAAM/AAM/cwarp.c` by issuing `mex cwarp.c`.
2. Add `matlAAM/AAM` to your Matlab path using `addpath`.
3. Run `matlAAM/AAM/examples/example1/aamRun_BuildModel.m`, then `aamRun_standard`![2] This will build and train your AAM before running 4-fold cross validation runs.
4. Take a look at the `aamResults/Images` directory, where the search results are stored. Use `aamResults` for a graphical representation of the numerical results.

A.2 Quick-start Guide

1. If you use `matlAAM` for the first time and you are using a system other than Windows or Mac OS X, please compile `matlAAM/AAM/cwarp.c` by issuing `mex cwarp.c`.
2. Add `matlAAM/AAM` to your Matlab path using `addpath`.
3. Create a directory of your choice (e. g. `yourdir`) and copy the contents of `matlAAM/examples/te` to `yourdir`.
4. To build an AAM, you need the following input data, which should be stored in `yourdir/data`:

   - Input images, which may be in `JPEG` (any other image file format understood by Matlab) or `DICOM` format. Create a file called `filenames.txt` containing the names of the images, with one filename per line:

     ```
     image1.jpg
     image2.jpg
     ...
     ```

[2] The annotated images used in this example were taken from [25]

- Landmark data for the input images. Landmarks are stored in a (n, m)-matrix, where n is the number of landmarks and m the number of images. Coordinates are stored as complex numbers, $x+y*i$, x being in the range of [1,image width] and y in the range of [1,image height]. Save the matrix as `landmarks.mat`.
- Initialization data for the AAM search. Create a $(4, m)$-matrix where m is the number of images. In each $(4, 1)$-column store the following information:
  (a) x coordinate
  (b) y coordinate
  (c) scale estimate, default is 0
  (d) rotation estimate, default is 0

5. Run `yourdir/model/aamRun_Buildmodel.m`, followed by `aamRun_standard.m`! The AAM will get built and you can observe the search working. Take a look at `aamRun.m` to familiarize yourself with the structure of `matlAAM`!

A.3 The Configuration and Management File `aamRun`

The highlighted function `aamRun` will usually be the only file you need to change when building your own AAMs. The sample included in `matlAAM/examples/example/aamRun.m` looks like this:

```
p = aamInit;
for set = 1:aamGet(p,'manyfold')
    p = aamLoadPFromModel(p,set);

    p = aamSet(p,'method','standard');
    aamTraining(p);

    p = aamSet(p,'recordsearchmovie',1);
    p = aamSet(p,'saveoverlaidresult',1);
    aamRunSearch(p, loadedimages);
end
```

`matlAAM` is built around the concept of using one configuration variable that contains all the information necessary to build an AAM and to run a search with it. Using this configuration variable, called `p` in our example, you can even control implementation details, thus generally you don't have to edit any other file than `aamRun.m`.

In the above example we first initialize `p` by executing `aamInit.m`:

```
function p = aamInit()

p={};
p = aamSet(p,'datapath', '../faces/');
p = aamSet(p,'levels',[1]);
p = aamSet(p,'manyfold',4);
```

This sets the path where the data is to be found, that it should only work on full resolution images (and not any other level of a Gaussian pyramid) and that 4-fold cross validation should be used. The number of images used, 36, is extracted from the number of lines of `filenames.txt`.

Back in `aamRun_standard`, we start the training procedure to estimate the regression matrix.

With the next two statements we configure that each search will be recorded as an `avi`-movie (it will be shown on screen during search, too) and that the converged result of the AAM search overlaid on the unseen image shall be saved as a `jepg` image.

We then start the search by invoking `aamRunSearch`. As you can see, `aamRun` contains all commands to control the build-search cycle of an AAM.

A.4 Reference of all Configuration Variables

Following you find a list of all variables available to control the AAM build-search cycle. The default values are set in `matlAAM/aamGetDefaultValue.m`. All variables are changed using

```
p = aamSet(p, 'variablename', 123);
p = aamSet(p, 'variablename', [1 2 3]);
p = aamSet(p, 'variablename', 'astring');
```

depending on the type of variable. To create an empty set of variables set `p = {}`.

`usecwarp` Defines whether image warping should use the implementation in C (`1`, default) or in Matlab (`0`).

`'imagetype'` File format of the input images. Either `'jpg'` (default) or `'dicom'`.

`'flipdicom'` Defines whether DICOM images shall be flipped upside down (`1`) or not (`0`, default).

`'datapath'` The path to the input data, i. e. the images and landmarks. Defaults to `'../data/'`.

`'filenames'` The name of the file that contains the list of image filenames. Defaults to `'filenames.txt'`.

`'landmarks'` The name of the file containing the landmark data. Defaults to `'landmarks.mat'`.

`'searchinitialisations'` The name of the file containing the search initialisation data (see A.2 for details).
Defaults to `'searchinitialisations.mat'`.

`'useconvexhull'` Defines how the image warping routine masks the warped images. If the convex hull of the landmarks will produce a satisfactory mask, set it to `1` (default), as this speeds up mask calculation. In cases where the convex hull is not exactly what we want, i. g. for bones, we set it to `0`.

`'standardtrainingimagecount'` Defines how many synthetic images are used to train the R-matrix. On each of these images all parameters are displaced one-by-one. Default is `20`.

`'standardtrainingdisplacementsteps'` Defines in how many steps the parameters are displaced during training of the R-matrix. Default is `10`.

`'standardsearchstepsizes'` Defines which step sizes are available during search scale the parameter predictions.
Default is `[1 0.5 1.5 0.25 0.1 2 0.025 0.01]`.

`'suffix'` This string is used to differentiate between different models built. When evaluating different training parameters it makes sure that the data of one model is not overwritten by the data of an other model. Default is `''`.

`'searchsuffix'` **and** `'searchsuffix2'` This strings ate used to differentiate between different search results. When evaluating different search parameters it makes sure that the results of one model are not overwritten by the results of an other model. Both default to `''`. There is no difference in meaning between the two, but if you test 2 sets of parameters in 2 nested loops, you can use the one for paramter 1 and the other for parameter 2.

`'method'` Defines whether the R-matrix (the standard AAM training scheme), `'standard'` (default), shall be used or the parameter prediction based on CCA, `'cca'`.

`'showintermediateresults'` Controls whether the current model image, overlaid over the test image, shall be displayed at each iteration (`1`) or not (`0`, default). *Note*: Interferes with `recordsearchmovie`!

`'recordsearchmovie'` Defines whether each search shall be recorded as an avi-movie (`1`) or not (`0`, default).
Note: Interferes with `showintermediateresults`!

`'forceditеration'` Controls whether the first parameter prediction of each hierarchy level shall be accepted even if does not improve the texture error. `1` turn it on, `0` off (default).

`'levels'` Defines the number of levels of the hierarchical AAM implementation used. If, for example, 3 levels shall be used, is has to be set to `[4 2 1]`. Default is `0`, i. e. the number of necessary levels is computer automatically (the highest level has a least a size of 20-by-20 pixels.

`'usesubregion'` **and** `'subregion'` If `'usesubregion'` is set to `1`, only the area defined by the landmarks listed in `'subregion'` will be used to compute the texture error. For example, if your models have 10 landmarks, but you want the search to minimize only the texture error of the area between landmarks 2,4,8 and 10, set `subregion` to `[2 4 8 10]`. Defaults are `0` and `[]`.

`'initxydistortion'`, `'initxydistortioncount'` **and** `'initxydistortionrange'`
If `initxydistortion` is set to `1`, then for each test images not only one search will be start using the initial position given in `searchinitialisations.mat`, but `initxydistortioncount` searches will be performed. Each initial position of these searches will be randomly displaced by up to `initxydistortionrange` pixels, both horizontally and vertically. Defaults are `0`, `0`, `0`.

`'ccaimagecount'` **and** `'ccaincpcaloops'`
For the training of the CCA-search `'ccaimagecount'` times `'ccaincpcaloops'` synthetic difference images are generated. As using all images at the same time would exceed practical memory amount, they are processed in `'ccaincpcaloops'` steps and only their PCA-coefficients are stored for the actual CCA-computation. Defaults are `100` and `10`.

`'ccanoincpca'` If `'ccanoincpca'` is set to `1` no incremental PCA is used when computing the CCA and all `'ccaimagecount'` times `'ccaincpcaloops'` are used directly. *Note*: Using this option might easily increase memory requirements by several hundred MBs. Default is `0`.

`'ccathreshold'` The threshold for the correlations used when computing the CCA. Default is `0.8`.

`'usemotionfield'` If set to `1` motion vectors are used instead of the difference images for parameter prediction. Default is `0`. *Note*: Only works in conjunction with `method` set to `cca`.

`'motionfieldcellsize'` When using motionvectors, controls the size of the tiles on which the motion vectors are computed. Default is `20`. *Note*: Only works with `usemotionfield` enabled.

`'robustnessmode'` If set to `1`, for each tile motion vectors outside of the range of the motion vectors occurring during training will be set to `0`. This is meant to increase stability of the search, i. g. in presence of occlusions. Default is `0`. *Note*: Only works with `usemotionfield` enabled.

`'bordercolor'` pixel value used to designate masked areas when warping images. Default: `10e100`

`'DelaunayTriangulation'` determines wether landmarks are triangulated automatically (delaunay) or by a given triangulation. If set to `1` landmarsk are triangulated automatically, if set to `0` tirangulation ist taken by aamGet(p,'tri') Default: `1`. *Note*: if set to `0` a triangulation has to be defined

`'tri'` works in conjunction with `'DelaunayTriangulation'`. A $n \times 3$ matrix defines the triangulation of the landmarks instead of the automatically determined delaunay triangulation.

A.5 The Inner Workings of `matlAAM`

In this section all functions of `matlAAM` will be discribed, in the order of their execution.

aamRun sets up the configuration for both the build and the search process. For a detailed description please see sections A.3 and A.4.

aamInitCrossValidation is called with information about how many images are used in total and how they should be split into sets containing training and test images. Returns the configuration variable `p` prepared accordingly.

aamBuildModel Builds the actual AAM. Calls the functions to load the landmarks and images, to create the shape and texture eigenspaces and the the combined AMM eigenspace, then calls the training function `aamRTraining` or `aamCCATraining` depending on the method chosen.

aamLoadLandmarks Loads the desired landmarks, which are the ones of the training images if called during training. During search it returns the landmarks of the test images.

aamAlignLandmarks Given a set of landmarks it normlized the corresponding shapes and aligns them using Procrustes analysis.

aamLoadImages Load the desired images, which are the training images during training and the test images during search.

aamWarpImages Warps the training images to the shape defined by the mean of the training landmarks. Also creates the masking image that defines the area where the texture error is computed.

aamRTraining / aamCCATraining Computes the regression between synthetic difference images and the corresponding parameter differences, either using the Jacobian scheme presented in **??** or the CCA.

aamRunSearch Manages the search process. Loads the test images and initial search positions, calles `aamSearch` for each images and stores the search results.

aamSearch Performs the actual search.

Additional `matlAAM` functions not depicted in the flowchart:

A.5.1 Main AAM Funtions

aamCompute_r.m Computes the texture residual. Calls aamSampleFromImage to sample from the image according to the current shape and subtracts it from the current model texture.

aamInitCrossValidation.m Divides the images from the data set into training and test sets, and a parameter set if necessary.

aamLoadImages.m Load the images, which can be JPEG or DICOM.

aamLoadLandmarks.m Load the Landmarks.

aamMotionField.m Computes the image gradient.

aamNormalizeImage.m Normalizes the texture vectors.

aamNormalizeLandmarks.m Normalizes the shape vectors.

aamProcrustes.m Performs the actual alignment of the shapes.

aamRTraining.m Creates a number of difference and the corresponding parameter differences and computes regression on them, i. e. it computes the R-matrix.

aamRunSearch.m Manages the search on the test set, where parameters can be sequentially chosen from a given range, to evaluate the influence of these parameters on the search.

aamSampleFromImage.m Samples from the image at the current model shape.

aamSwitchtolevel.m Sets internal variables when moving from one level of the Gaussian pyramid to another.

aamTestPredictions.m Generates a set of difference images and compares the resulting parameter predictions to the parameter displacements that generated the difference images.

A.5.2 Image Warping Funtions

aamWarp.m Warps an image according to the landmarks specified. Takes care of the exact offsets of the 2 shapes.

aamWarpImages.m Warps all input images to the mean shape.

aamQuickWarp.m Wrapper around the warping function implemented in C. In addition it contains the same algorithm as the C function in matlab.

cwarp.c C image warping function.

A.5.3 Data Visualisation Functions

aamMakeImage.m Returns the current model texture.

aamMakeModeMovie.m Creates a movie of parameter variations.

aamMakeMovie.m Creates a movie of images that are passed as parameter.

aamMakeShapedImage.m Return the current model texture warped to the current shape.

aamsurfg.m Displays a 3D view of a texture vector / a difference image.

aamshowAAMgui.m Allows to investigate an AAM model by providing a graphical interface in which the first 6 modes can be set trough scroll bars. The resulting model output is displayed.

aamshowdicom.m Displays data stemming from a DICOM image correctly.

aamshowimagewithshape.m Displays the current model state.

aamPlotShape.m Plots a shape that is passed as parameter.

aamPlotStatistics.m Plot the results of a search.

A.5.4 Configuration Variable Management Functions

aamDel.m Removes and entry from the configuration variable list.

aamExistsParameter.m Checks for the existence of a parameter in the configuration variable list.

aamGet.m Return the value of a configuration variable form the configuration variable list. If the parameter was not set, aamGetDefaultValue is called.

aamGetDefaultValue.m Returns the default value for a configuration variable.

aamSet.m Set a configuration variable from the configuration variable list to the specified value.

A.5.5 CCA related Functions

aamCCA.m Computes the CCA between the PCA factors of the difference images and the parameter displacements.

aamCCATraining.m Similar to aamRTraining, creates a number of difference and the corresponding parameter differences. Computes the PCA on the difference images using incremental PCA to lower memory requirements, calls aamCCA.

aamCCAmatch.m During search, this function returns the parameter displacement estimate to a given difference image.

A.5.6 Internal Functions

aamsvdpca2.m Computes the PCA using the SVD.

aamcomp2real.m Internal conversion between the complex and the real landmark format. See Sec. A.6 for details.

aamRotate.m Rotates a shape.

aamcenterofgravity.m Computes the center of gravity of a shape.

aamdicom2img.m Adjusts the value range of DICOM images.

aamgausswin.m Creates a vector of specified length with values according to a Gaussian distribution.

aamreal2comp.m Internal conversion between the real and the complex landmark format. See Sec. A.6 for details.

A.5.7 Helper Functions

aamimport_stegmann_asf.m Convert landmark data from the format used by Stegmann to the `matlaam` format.

aamAddLmsCorridor.m Given a set of landmarks, this function adds a border or corridor by duplicating each landmark at a 10 pixel distance outside of the shape.

A.6 Data formats

Througout `matlAAM` the following data structures and formats are being employed:

Landmarks are stored in (n, m)-matrices, where n is the number of landmarks and m is the number of samples. So training landmarks could be stored in a $(64, 30)$-matrix and test landmarks in a $(64, 10)$-matrix, for example.

The coordinates of each landmark are stored as a complex number, with the x-coordinate being the real part and the y-coordinate being the imaginary part. Both coordinates are in the range of [1, image width] and [1, image height], respectively.

Internally landmarks are sometimes stored as $(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)^T$-vectors, but all interfaces use the complex number format.

Images are stored in (n, m)-matrices, where n is the number of pixels (= image width $*$ image height) and m is the number of images. The value range of each pixel is [0, 255]. `DICOM` images are transformed by mapping the minimum pixel value found in the image to 0 and the maximum pixel value to 255.

References

[1] S. M. Aghito, M. B. Stegmann, S. Forchhammer, and B. K. Ersbøll. Segmentation of object-based video of gaze communication. In *HCI International, 11th International Conference on Human-Computer Interaction, Las Vegas, USA*, 2005. 3

[2] R. Beichel, G. Gotschuli, E. Sorantin, F. W. Leberl, and M. Sonka. Diaphragm dome surface segmentation in CT data sets: a 3D active appearance model approach. In *Proc. SPIE Vol. 4684, p. 475-484, Medical Imaging 2002: Image Processing, Milan Sonka; J. Michael Fitzpatrick; Eds.*, pages 475–484, May 2002. 2, 3

[3] Reinhard Beichel, Horst Bischof, Franz Leberl, and Milan Sonka. Robust active appearance model matching, 2005. 3

[4] M. Borga, T. Landelius, and H. Knutsson. A unified approach to PCA, PLS, MLR and CCA. Report LiTH-ISY-R-1992, Department of Electrical Engineering, SE-581 83 Linköping, Sweden, 1997. 3

[5] Magnus Borga. *Learning Multidimensional Signal Processing.* Linköping Studies in Science and Technology, Dissertations, No. 531. Department of Electrical Engineering, Linköping University, Linköping, Sweden, 1998. 12

[6] Leo Breiman and Jerome H. Friedman. Predicting multivariate responses in multiple linear regression. *J.R. Statist. Soc.*, 59(1):3–54, 1997. 12

[7] T. Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *Proc. of BMVC*, volume 2, pages 837–846, 2002. 3

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models, 1998. 2

[9] T. F. Cootes and C. Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, University of Manchester, 2004. 5, 18

[10] T.F. Cootes. Timeline of developments in asms and aams. 3

[11] T.F. Cootes, G. Edwards, and C.J. Taylor. A comperative evaluation of active appearance model algorithms. In *Proceedings of BMVC*, volume 2, pages 680–689, 1998. 3

[12] Tim Cootes. Active shape models - 'smart snakes'. In *Proc. British Machine Vision Conference*, 1992. 2

[13] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Trans. PAMI*, 23(6):681–685, 2001. 3

[14] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Trans. PAMI*, 23(6):681–685, 2001. 5, 10

[15] Rene Donner, Georg Langs, Michael Reiter, and Horst Bischof. Fast active appearance model search based on CCA and its application to hand radiographs. In *Proceedings of the Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition*, 2005. 2

[16] G. Edwards, C.J. Taylor, and T.F. Cootes. Interpreting face images using active appearance models. In *Proc. IEEE International Conference on Automatic Face and Gesture Recognition 1998*, pages 300–305, 1998. 3

[17] P. Filzmoser. Outlier resistant estimators for canonical correlation analysis. Technical report, TU Wien, 2000. 3

[18] M. Fitzgibbon, A. W.and Pilu and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, 1999. 6

[19] C. A. Glasbey and K. V. Mardia. A review of image-warping methods. 25(2):155–??, April 1998. 7

[20] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal fo the Royal Statistical Society, Series B*, 1991. 6

[21] X. Hou, S. Li, H. Zhang, and Q. Cheng. Direct appearance models. In *Computer Vision and Pattern Recognition Conference*, volume volume 1, pages 828–833, 2001. 3

[22] N. Johnson, A. Galata, and D. Hogg. The acquisition and use of interaction behaviour models. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR'98)*, pages 866–871, 1998. 3

[23] Thomas Melzer, Michael Reiter, and Horst Bischof. Appearance models based on kernel canonical correlation analysis. *Pattern Recognition*, 39(9):1961–1973, 2003. 11

[24] Steven C. Mitchell, Johan G. Bosch, Boudewijn P. F. Lelieveldt, Rob J. van der Geest, Johan H. C. Reiber, and Milan Sonka. 3-d active appearance models: Segmentation of cardiac MR and ultrasound images. *IEEE Trans. Med. Imaging*, 21(9):1167–1178, 2002. 3

[25] M. M. Nordstrøm, M. Larsen, J. Sierakowski, and M. B. Stegmann. The IMM face database - an annotated dataset of 240 face images. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, may 2004. 6, 14, 18

[26] M.G. Roberts, T.F. Cootes, and J.E. Adams. Linking sequences of active appearance sub-models via constraints: an application in automated vertebral morphometry. In *Proceedings of the BMVC*, volume 1, pages 349–358, 2003. 2, 3

[27] Sami Romdhani, Alexandra Psarrou, and Shaogang Gong. On utilising template and feature-based correspondence in multi-view appearance models, 2000. 3

[28] Ian Scott. mproving appearance model matching using local image structure. In *Proc. Information Processing in Medical Imaging 2003*, 2003. 3

[29] Stegmann. Generative interpretation of medical images. Master's thesis, Technical University of Denmark, 2004. 3

[30] M. B. Stegmann, H. Ólafsdóttir, and H. B. W. Larsson. Unsupervised motion-compensation of multi-slice cardiac perfusion MRI. *Medical Image Analysis*, 9(4):394–410, aug 2005. 3

[31] Mikkel B. Stegmann and Rasmus Larsen. Multi-band modelling of appearance. In *First International Workshop On Generative Model-Based Vision*, pages 101–106, 2002. 3

[32] Hans Henrik Thodberg. Minimum description length shape and appearance models. In *Proceedings of Information Processing and Medical Imaging; IPMI*, 2003. 6

[33] M.A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings CVPR '91*, 1991. 2