Additive Update Predictors in Active Appearance Models

Philip A Tresadern
philip.tresadern@manchester.ac.uk
Patrick Sauer
patrick.sauer@postgrad.manchester.ac.uk
Tim F Cootes
tim.cootes@manchester.ac.uk

Imaging Science and Biomedical Engineering
University of Manchester
Manchester, UK
M13 9PT

Abstract

The Active Appearance Model (AAM) provides an efficient method for localizing objects that vary in both shape and texture, and uses a linear regressor to predict updates to model parameters based on current image residuals. This study investigates using additive (or 'boosted') predictors, both linear and non-linear, as a substitute for the linear predictor in order to improve accuracy and efficiency. We demonstrate: (a) a method for training additive models that is several times faster than the standard approach without sacrificing accuracy; (b) that linear additive models can serve as an effective substitute for linear regression; (c) that linear models are as effective as non-linear models when close to the true solution. Based on these observations, we compare a 'hybrid' AAM to the standard AAM for both the XM2VTS and BioID datasets, including cross-dataset evaluations.

1 Introduction

Localizing and tracking objects in images and video – a central problem in Machine Vision – is complicated if the objects may vary in both shape and texture, as is often the case over a population of individuals from a given biological group. Determining the underlying parameters of this variation is useful for analysis and interpretation of the scene, such as in face analysis where texture strongly indicates identity and shape can indicate pose or expression. Similarly, in medical image analysis the shape and texture of internal organs can indicate the presence or nature of disease.

With the aim of recovering these shape and texture parameters from image data, the Active Appearance Model (AAM) was introduced [7] and has been further developed over the years [11]. In short, the AAM samples image pixels within a region defined by a set of control points (that are in turn defined by the parameters of a shape model) and feeds the sampled image data into a regressor that predicts incremental updates to the shape parameters. The shape parameters are updated and the process repeats either for a fixed numer of iterations or until convergence (when parameter updates approach zero).

One characteristic of the original AAM [7] is that it uses a linear regressor to predict shape parameter updates from sampled image pixels. More recently, additive non-linear

predictors – strong learners expressed as a weighted sum of weak learners that are trained via greedy function approximation [10] – have shown better performance [21, 28]. The learning process needed to build an additive model, however, is notoriously slow and has parameters (particularly 'shrinkage') that are tricky to determine. This paper investigates the properties of additive models in AAMs and makes the following contributions:

- A method that pools weak learners when training an additive ensemble, resulting in significantly faster training;
- A comparison between methods of estimating the linear predictor, including an additive approach that identifies useful image features;
- Assessment of a 'hybrid' model fitting approach that uses non-linear additive models for robustness to local minima and linear additive models for accuracy when close to the true solution.

1.1 Related Work

The texture model used in an AAM has its origins in the 'Eigenfaces' recognition system [23] that parameterized faces images by their projections in a linear subspace [15]. A 'Point Distribution Model' [5] parameterized shape in a similar way. The Active Appearance Model [7] then presented an efficient method that used a linear predictor to update shape and texture parameters until they reached a local optimum.

Originally, the linear predictor was learned by directly solving a linear system of equations [7]. This was later replaced by a more efficient system that used a Gauss-Newton update step [3]. Unlike the AAM which predicts both shape and texture (collectively known as 'appearance') at each iteration, the 'Project Out' [18] (or 'Shape AAM' [6]) method predicts only shape parameters and is therefore more efficient. Inverse compositional alignment [1, 18] is even more efficient whereas non-linear predictors can also be effective [21, 28]. Coarse-to-fine model fitting has long been encouraged via an image pyramid [7, 14] and using a hierarchy of training data can also improve performance [22].

The limitations of representing shape by its projection onto a linear subspace are evident when large variation is present in the training set (*e.g.* due to rigid head motion) where the flexibility needed by the model to represent the training variation also permits non-valid shapes. Methods such as mixture models [2, 4], non-linear manifolds [20] and combined 2D+3D models [19] make the model more specific where needed.

Describing texture by its projection onto a linear subspace mixes variation due to identity, expression and lighting. Though we can separate these modes of variation out with post-processing [9], multilinear models do a better job implicitly [16]. Discriminative methods can be trained to recognize good matches [17] or the better of two matches [26].

2 Active Appearance Models

The 'vanilla' AAM assumes a linear subspace model for both shape, $\mathbf{x}$, and texture, $\mathbf{g}$, in a normalized model frame such that:

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s\mathbf{b} \qquad \text{and} \qquad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g\mathbf{q} \tag{1}$$

where $\mathbf{x} = (x_1, y_1, \ldots, x_P, y_P)^T$ is the vector of P landmark point positions that is characterized by a mean shape, $\bar{\mathbf{x}}$, plus modes of variation that are defined by the B columns of $\mathbf{P}_s$ and weighted by coefficients $\mathbf{b}$; this normalized shape is then transferred to the image frame via a similarity transformation, $T_s(\mathbf{x};\theta_s)$ where θ_s are parameters that correspond to scale, orientation and translation.

Similarly, $\mathbf{g}$ is the normalized vector of G 'shape-free' greyscale pixel values that are defined by a mean, $\bar{\mathbf{g}}$, and modes of variation $\mathbf{P}_g$ with coefficients $\mathbf{q}$; global illumination effects are then applied via a 1D affine transformation, $T_g(\mathbf{g};\theta_g)$ where θ_g are parameters corresponding to brightness and contrast. The modes of variation, $\mathbf{P}_s$ and $\mathbf{P}_g$, correspond to the eigenvectors of a training set and are estimated via Principal Component Analysis.

In the original implementation [7], shape and texture parameters were coupled into a single 'appearance' parameter vector. For the purposes of this work, however, we follow the more efficient 'Project Out' [18] (or 'Shape AAM' [6]) method. Given values for the shape parameters, $\mathbf{p} = \{\mathbf{b}, \theta_s\}$, we compute the shape in the image frame and warp the image data (via a piecewise affine transform) back to the 'shape-free' texture frame. There we compute an image residual with respect to current texture parameters that we use to predict and apply incremental parameter updates (*e.g.* via gradient descent) until some error measure (*e.g.* sum of squared difference) reaches a local minimum.

2.1 Computing Model Parameter Updates

Matching the model parameters to the image via off-the-shelf optimizers such as gradient descent is computationally expensive since the gradient (and sometimes Hessian) must be computed at each iteration. However, it can be shown that the relationship between the parameter residuals, $\delta\mathbf{b}$, and measurement residuals, $\delta\mathbf{g}$, is approximately linear when close to the true solution such that:

$$\delta\mathbf{b} = \mathbf{R} \cdot \delta\mathbf{g} \tag{2}$$

where $\mathbf{R}$ is a matrix of regression coefficients that is estimated from a set of N labelled training examples $\{\delta\mathbf{b}_i\}$ and $\{\delta\mathbf{g}_i\}$. These examples are generated by fitting the shape model to images with known landmark locations, applying a perturbation $\delta\mathbf{b}$ to the optimal parameters and computing the resulting image residual, $\delta\mathbf{g}$.

Originally [7], $\mathbf{R}$ was estimated directly by stacking training examples into matrices:

$$\mathbf{B} = \begin{bmatrix} \delta\mathbf{b}_1 & \delta\mathbf{b}_2 & \cdots & \delta\mathbf{b}_N \end{bmatrix} \qquad \text{and} \qquad \mathbf{G} = \begin{bmatrix} \delta\mathbf{g}_1 & \delta\mathbf{g}_2 & \cdots & \delta\mathbf{g}_N \end{bmatrix}, \tag{3}$$

and solving the linear equations:

$$\mathbf{B} = \mathbf{R}\mathbf{G} \qquad \Rightarrow \qquad \mathbf{R} = \mathbf{B}\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}. \tag{4}$$

This, however, means inverting the $G \times G$ matrix, $\mathbf{G}\mathbf{G}^T$, which is computationally demanding. Principal Component Regression [12] solves this problem by projecting the residual images onto a lower dimensional subspace before solving the linear system [13], thus also regularizing the solution if it is not well constrained (*e.g.* when $N < G$). Canonical Correlation Analysis projects the images onto directions of maximal covariance between the residuals and parameter updates [8], in contrast to Principal Component Analysis that ignores the

variables we aim to predict. The most popular alternative [3], however, was proposed based on the Gauss-Newton approximation:

$$\delta\mathbf{g} \approx \frac{\partial \mathbf{g}}{\partial \mathbf{b}} \cdot \delta\mathbf{b} \qquad \Rightarrow \qquad \delta\mathbf{b} \approx (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\delta\mathbf{g} \tag{5}$$

where we have used the shorthand

$$\mathbf{J} = \frac{\partial \mathbf{g}}{\partial \mathbf{b}} \tag{6}$$

to denote the $G\times B$ Jacobian. A weighted average, $\mathbf{J}$, is computed via finite differencing over the training set and inverting the $B\times B$ matrix, $\mathbf{J}^T\mathbf{J}$, is simple since $B \ll G$. Despite its theoretical basis, however, this approximation is a poor substitute for $\mathbf{R}$ and does not generalize well; its use is limited to tracking one person for whom the model was trained.

More recent studies have investigated non-linear (rather than linear) predictors for updating parameters. Examples include the Relevance Vector Machine for rigid object tracking [25] and additive piecewise constant functions applied to AAM fitting [21]. In this study, we further examine the additive model approach to predicting parameter updates.

3 Additive Models for Parameter Updates

Predicting parameter updates from sampled image data has recently been demonstrated [21, 28] using additive models (often referred to as boosted regressors) of the form

$$\delta\mathbf{b} = \sum_{m=1}^{M} \lambda \mathbf{f}_m(h_m(\mathbf{g})) \tag{7}$$

where: $h_m()$ computes a scalar feature of the sampled image data, $\mathbf{g}$; the *weak learner*, $\mathbf{f}_m()$, uses h_m to predict each of the parameter updates; and $0 \leq \lambda \leq 1$ is a 'shrinkage' parameter that scales each weak learner's contribution to the final output [10]. Note that it is common to predict parameter updates from the sampled image data, $\mathbf{g}$, rather than image residual, $\delta\mathbf{g}$.

In this study, we use Haar-like features due to their efficient computation from the integral image [24]. Possible choices for the class of weak learner include piecewise constant [21] and piecewise linear [27] functions. It is also not uncommon for $\mathbf{f}_m()$ to predict each parameter, b_j, independently via a scalar function, $f_{mj}()$.

When training a weak learner (Algorithm 1), finding the best feature-function pair is computationally expensive because the number of possible Haar-like features is very large.

Algorithm 1 Algorithm for training a weak learner.

1. Initialize residual for all training examples, i: $\mathbf{r}_i = \delta\mathbf{b}_i$
2. For $m = 1 \ldots M$ 'rounds' of boosting:
   (a) Find feature-function pair that best predicts current residuals:
   $(\mathbf{f}_m, h_m) = \arg\max_{\mathbf{f},h} \sum_i [\mathbf{r}_i - \mathbf{f}(h(\mathbf{g}))]^2$
   (b) Add feature-function pair to ensemble and update residuals:
   $\mathbf{r}_i - \lambda \mathbf{f}_m(h_m(\mathbf{g})) \rightarrow \mathbf{r}_i$

However, if we use a class of weak learners that can be fitted to the data quickly (*e.g.* piecewise constant) the search becomes more practical, albeit very slow. In the following, we show how to make this process faster and how it may relate to the standard linear approach.

3.1 Pooling Weak Learners

If a small number of weak learners happen to predict the training data particularly well, the strong learner will overfit and not generalize well to new data. To avoid this, the accepted practice is to use a large ensemble of diverse learners with a small shrinkage parameter (*e.g.* $\lambda \approx 0.05$) to prevent any one learner from dominating [10]. However, this approach has two considerable drawbacks:

- Although Haar-like features and primitive functions can be evaluated quickly, the large numbers involved mean it is not uncommon for the learning process to take hours or even days [28]. Deliberately increasing the number of weak learners, M, results in even longer training times.
- Estimating the optimal value of λ is not trivial – too high a value results in overfitting, too low a value results in systematically underestimating the output – and systems do tend to be sensitive to its value. A stopping criterion based on increasing validation error makes the training process more complex and reduces efficiency. As a result, λ and M are usually selected empirically and are often suboptimal.

Based on these observations, we made two hypotheses: (a) a large and diverse ensemble is required to provide some immunity to overfitting; (b) if λ is small, the residual is reduced by a small amount at each round such that successive weak learners are likely to be highly correlated. We propose to retain these properties while improving efficiency by adding the best $K > 1$ learners, appropriately weighted (*e.g.* with weight $1/K$), at each of the M rounds:

$$\delta\mathbf{b} = \sum_{m=1}^{M} \frac{\lambda}{K} \sum_{k=1}^{K} \mathbf{f}_{m,k}(h_{m,k}(\mathbf{g})). \tag{8}$$

As a result, the number of rounds of boosting, M, that are required for an ensemble of equal size is reduced by a factor of K. Unlike repeating a round of boosting K times, however, maintaining a list of the K best learners at each round imposes almost no extra computational cost. Inevitably, this approximation incurs a penalty in accuracy compared to the standard approach; one aim of this work was to find ways to maintain efficiency while minimizing the loss of accuracy.

3.2 Linear Additive Models with Haar-Like Features

Armed with an efficient training scheme for additive learners, we now consider what happens if the weak learners are themselves linear. Specifically, since Haar-like features are linear transformations of the image, the resulting strong learner is also linear and therefore equivalent to a regression matrix, $\mathbf{R}$. For large image samples, the additive model may be faster to evaluate than the linear regressor whereas an additive model with many weak learners may be evaluated more efficiently using the equivalent linear model for small image samples. To compare performance of the linear predictor when estimated in different ways, we conducted an experiment that not only measured error for different methods but also compared their equivalent basis images for further insight into their differences.

3.3 Linear vs. Non-Linear Regression

When far from the optimal solution, the Taylor series approximation begins to break down and a linear update relationship no longer holds. Recent studies have therefore replaced the linear regressor, $\delta\mathbf{b} = \mathbf{R} \cdot \delta\mathbf{g}$, with a non-linear one, $\delta\mathbf{b} = \mathbf{f}(\delta\mathbf{g})$, such as an additive piecewise constant function [21] or a Relevance Vector Machine [25]. Since a non-linear regressor is better equipped to cope with these situations, it should perform better than a linear predictor. Given that other studies have noted the strong performance of well-trained linear models compared to non-linear ones [22], we invesigated this expectation by comparing a non-linear with several linear methods over a range of initial displacements.

4 Results

In the following experiments, we trained and tested on the publicly available XM2VTS dataset (and the BioID dataset in some cases). Training data for the XM2VTS model consisted of two samples, randomly perturbed from ground truth parameter values, for each of the images corresponding to subjects 000-163 from the first session (588 images in total); testing data consisted of 25 samples for each of the images corresponding to subjects 164-371 from the first session (7400 in total).

4.1 Pooling Weak Learners

This first experiment compared additive models that build a strong predictor $K = 1$ weak learner at a time over M rounds against those that add $K > 1$ weak learners at each round, using the non-linear piecewise constant function as our weak learner. Keeping the total number of weak learners fixed at $M \times K = 100$, we computed the test error for values of λ around the optimum (determined by trial and error) for three cases: 1×100, 10×10 and 100×1. Two more cases were tested, also with $M = 10$ rounds but with $K = 20$ and $K = 100$.

When varying K with $M \times K = 100$, our results (Figure 1) show that error was lowest for $K = 1$ (the standard case). Higher values such as $K = 10$ did not increase error substantially even though they required a fraction of the time to train. Increasing K beyond some point,

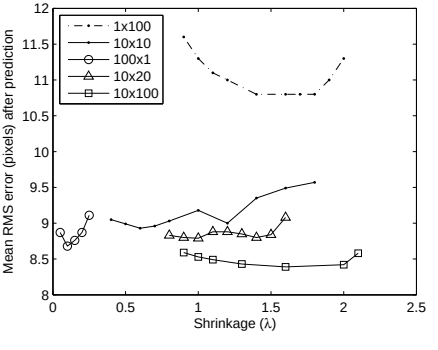


Figure 1: Error with respect to shrinkage for different ($M \times K$) cases. Increasing M improves performance but at additional cost during training; increasing K for a fixed M also improves performance at a smaller additional cost during testing.

however, increased errors significantly – in particular, the errors for $K = 100$ suggest that multiple rounds of boosting are important. When varying K with a fixed $M = 10$ (such that training time is largely unchanged), errors were even lower than those achieved for the 100×1 case yet required far less time to train. Pooling learners also appears to be less sensitive to the exact value of λ and using $\lambda = 1$ for $K > 1$ often gives good results. Some examples also show the difficulty in finding an 'optimal' value for λ (*e.g.* 10×20 has two local minima).

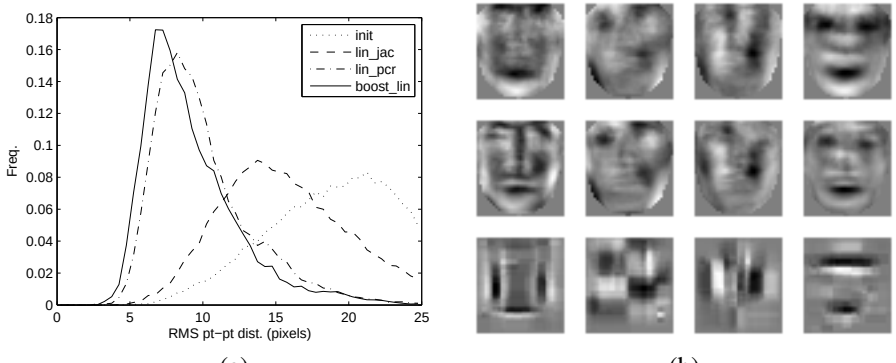
During testing, a basic AAM with $M\times K = 100$ learners required around 0.2 milliseconds for two iterations; with $M\times K = 1000$ this increased to 1.1 milliseconds. If a linear weak learner is used, however, the equivalent linear predictor may be used thus placing an upper bound on the time required for testing. An future study will investigate methods of post-processing the 'shortlist' at each round to remove highly correlated features from the pool.

The significance of these result lies not in an improvement in accuracy but in the speed at which accurate models can be trained compared with the one-at-a-time approach. Training time is roughly proportional to the number of rounds of boosting, M, which pooling reduces significantly without compromising accuracy. As a result, models can be built and tested in a fraction of the time previously required. With this modification, it is also considerably easier (and quicker) to find a sensible value of K and λ.

4.2 Linear Additive Models with Haar-Like Features

Since an additive model can be used to approximate the linear regression matrix, $\mathbf{R}$, it is useful to determine what effect (if any) this has on performance. Therefore, we compared the performance of a linear predictor where $\mathbf{R}$ was estimated using three different methods: the Gauss-Newton approximation [3]; Principal Component Regression [12, 13] where training images were first projected onto a subspace that captured 80% of the original variance; and an additive model with 100 linear weak learners.

A comparison of error between the methods (Figure 2) revealed little difference between Principal Component Regression (*lin_pcr*) and the linear additive model (*boost_lin*).



(a) (b)

Figure 2: (a) Error distributions before prediction (*init*) and following two iterations of prediction with three methods: Gauss-Newton approximation (*lin_jac*), Principal Component Regression (*lin_pcr*), linear additive model with Haar-like features (*boost_lin*); (b) Equivalent basis images for the same three methods, corresponding to (from left) scale, orientation, x translation and y translation.

For moderate perturbations, however, the Gauss-Newton approximation (*lin_jac*) performed poorly compared with the other methods. We also see that the equivalent basis images (Figure 2) are qualitatively similar. For example, those corresponding to orientation all exhibit a 'checkerboard' pattern and the linear additive model places more emphasis than the others on the regions of high gradient around the eyes and mouth for y translation.

This experiment suggests that an additive linear model could be used as a cheap approximation to a more complete linear solution, rather than the Gauss-Newton approximation that performs poorly. It also highlights more clearly the most important regions of the image when predicting parameter updates.

4.3 Linear vs. Non-Linear Regression

Intuitively, non-linear predictors should perform better than linear predictors for large initial displacements where the linear relationship breaks down. To test this intuition, we compared the performance of the three linear predictors (*lin_jac*, *lin_pcr* and *boost_lin*) with that of the non-linear predictor (*boost_nonlin*) for initial displacements of increasing size. The results (Figure 3) suggest that non-linear methods outperformed linear ones for large displacements but the benefit was lost when close to the solution. Again, there was only a small difference between the Principal Component Regression solution and the linear approximation from an additive model. The Gauss-Newton approximation performed worst for all displacements.

4.4 Cross-Dataset Evaluation

Finally, we compared the performance of a basic AAM with that of an advanced AAM that used several recently proposed improvements. The 'baseline' AAM consisted of a single model of around $G = 3000$ pixels, contained four pose parameters and 18 shape modes, and was fit to image data by updating parameters with a linear predictor estimated via Principal Component Regression. Four iterations were applied for each image.

The 'hybrid' AAM contained four models with 1000, 1000, 3000 and 3000 pixels, respectively. The four models included four pose parameters, and 0, 6, 12 and 18 shape modes so that those related to fine detail were predicted last, once the model was almost aligned. The first model updated parameters with an additive non-linear predictor whereas subsequent

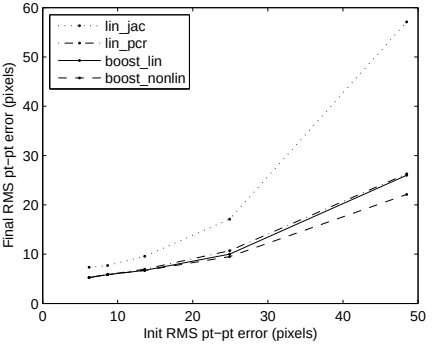


Figure 3: Final RMS pt-pt errors for all four methods with respect to increasing initial displacement. Standard error bars are so small as to be almost invisible.

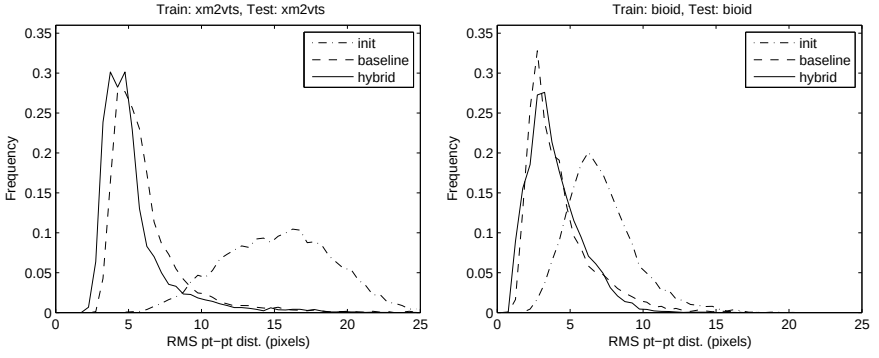


Figure 4: Distribution of RMS error when using the same dataset for training and testing.

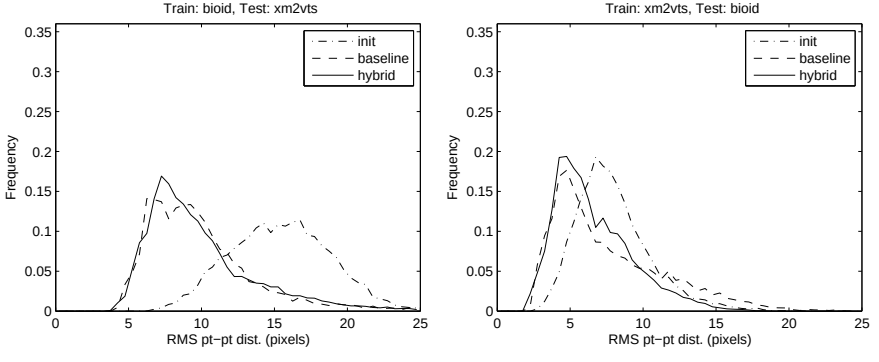


Figure 5: Distribution of RMS error when using different datasets for training and testing.

models used the additive linear model (Section 3), each applied for one iteration. The models were also trained sequentially [22, 29]: for each model in the sequence, samples were drawn from the same generating distribution and updated with all preceding models that had already been trained. This ensured that the distribution of training examples for each model was representative of the uncertainty in the sequence of predictions up to that iteration.

In addition to the XM2VTS data, we trained and tested models with the BioID datasets. Training data for the BioID model consisted of two samples, randomly perturbed from ground truth parameter values, for images 0000-0749 (1500 in total); testing data contained 10 samples for each of the images 0750-1519 (7690 in total). Initial errors are smaller for the BioID experiments due to differences in scale between datasets (BioID faces are smaller).

The results for models that were trained and tested on the same dataset (Figure 4) suggest that the standard AAM performs well but can be improved further using advances such as sequential training and non-linear regression at early stages. These cross-dataset experiments (Figure 5), absent in many studies, suggest that the benefits of the 'hybrid' model may only apply for similar training and testing data. For example, the sequential training scheme [22] may fail for new datasets where the spread of predictions at one stage is greater than expected such that the following stage is inadequately trained. These experiments highlight the importance of cross-dataset evaluation.

5 Summary

Active Appearance Models have recently been improved to exploit the benefits of additive (boosted) regression models. This paper demonstrates that:

- Training additive models can be made substantially more efficient by pooling several weak learners at each round of boosting. This also appears to reduce the sensitivity to the shrinkage parameter, λ. In future work, we will investigate this property further with a view to evaluating its application on a wider scale.
- A linear predictor can be approximated via an additive linear model with potentially greater efficiency.
- Non-linear regressors are useful for avoiding local minima when far from the true solution; however, linear regressors are more appropriate when close to the true solution. Therefore, a hybrid approach that uses non-linear regression at first then switches to linear updates can offer benefits in performance.

Acknowledgements

This work was supported by European Funded Project FP7-2007-ICT-1: Mobile Biometrics (MoBio).

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on : A unifying framework. part I : The quantity approximated, the warp update rule and the gradient descent approximation. *Int. J. Comput. Vis.*, 2004.

[2] T. Cootes and C. J. Taylor. A mixture model for representing shape variation. *Image Vision Comput.*, 17(8):567–574, 1999.

[3] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2001.

[4] T. Cootes, G. Wheeler, K. Walker, and C. Taylor. View-based active appearance models. *Image Vision Comput.*, 2002.

[5] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models – Their training and application. *Comput. Vis. Image Und.*, 61(1):38–59, January 1995.

[6] T. F. Cootes, G. Edwards, and C. J. Taylor. A comparative evaluation of actuve appearance model algorithms. In *Proc. British Machine Vision Conf.*, 1998.

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conf. on Computer Vision*, 1998.

[8] R. Donner, M. Reitner, G. Langs, P. Peloschek, and H. Bischof. Fast active appearance model search using canonical correlation analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1690–1694, October 2006.

[9] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proc. IEEE Int'l Conf. on Automatic Face and Gesture Recognition*, 1998.

[10] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[11] X. Gao, Y. Su, X. Li, and D. Tao. A review of active appearance models. *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, 40(2):145–158, March 2010.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[13] X. Hou, S. Z. Li, H. Zhang, and Q. Cheng. Direct appearance models. In *Proc. IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2001.

[14] M. J. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *Int. J. Comput. Vis.*, 2(29):107–131, 1998.

[15] M. Kirby and L. Sirovich. Application of the Karhunen-Loève procedure for the characterization of human faces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):103–108, January 1990.

[16] H.-S. Lee and D. Kim. Tensor-based active appearance model. *IEEE Sig. Proc. Lett.*, 15:565–568, 2008.

[17] X. Liu. Discriminative face alignment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31 (11):1941–1954, November 2009.

[18] I. Matthews and S. Baker. Active appearance models revisited. *Int. J. Comput. Vis.*, 26 (10):135–164, October 2004.

[19] I. Matthews, J. Xiao, and S. Baker. 2D vs. 3D deformable face models: Representational power, construction, and real-time fitting. *Int. J. Comput. Vis.*, 75(1):93–113, October 2007.

[20] S. Romdhani, S. Gong, and A. Psarrou. A multi-view nonlinear active shape model using kernal PCA. In *Proc. British Machine Vision Conf.*, 1999.

[21] J. Saragih and R. Goecke. A nonlinear discriminative approach to AAM fitting. In *Proc. IEEE Int'l Conf. on Comp. Vis.*, 2007.

[22] J. Saragih and R. Goecke. Learning AAM fitting through simulation. *Pattern Recogn.*, 42(11):2628–2636, November 2009.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *J. Comput. Neurosci.*, 3(1): 71–86, 1991.

[24] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vis.*, 57(2): 137–154, May 2004.

[25] O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1292–1304, August 2005.

[26] H. Wu, X. Liu, and G. Doretto. Face alignment via boosted ranking model. In *Proc. IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2008.

[27] J. Zhang, S. K. Zhou, D. Comaniciu, and L. McMillan. Conditional density learning via regression with application to deformable shape segmentation. In *Proc. IEEE Conf. on Comp. Vis. and Patt. Recog.*, 2008.

[28] S. K. Zhou and D. Comaniciu. Shape regression machine. In *Proc. Int'l Conf. on Information Processing in Medical Imaging*, 2007.

[29] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(4):677–692, April 2009.