Model compression of SDM-based face alignment for mobile applications

Yehu Shen1,2, Quansheng Jiang1,2, Yong Yang1,2, Bangfu Wang1,2, Qixin Zhu1,2

1College of Mechanical Engineering, Suzhou University of Science and Technology, Suzhou, People’s Republic of China
2Suzhou Key Laboratory of Precision and Efficient Machining Technology, Suzhou, People’s Republic of China
E-mail: yehushen@mail.usts.edu.cn

Abstract: Face alignment could be widely used in face recognition, expression recognition, face-based AR applications etc. Cascaded-regression-based face alignment algorithms have been popular in recent years for their low computational costs and impressive results in uncontrolled scenarios. Unfortunately, the size of the trained model is quite large for cascaded-regression-based methods which makes it unsuitable for commercial applications on mobile phones. In this study, the authors proposed a data compression method for the trained model of the supervised descent method (SDM). Firstly, the distribution of the model data was estimated using a non-parametric method. Then an adaptive quantisation algorithm was proposed to quantise the model data. Finally, their adaptive quantisation algorithm was tightly coupled with the SDM training process to fine tune the results. The quantitative experimental results proved that their proposed method could compress the data model to <20% of its original size without hurting the performances. The proposed method has been integrated into a mobile AR application, subjective evaluations proved that the proposed compression method could provide similar visual effects compared with the uncompressed counterpart.

1 Introduction

Face alignment is important for facial image analysis. It uses a face image as an input and automatically localises facial feature points such as eyes, nose, eyebrows mouth etc. Fig. 1 shows an example of face alignment with the supervised descent method (SDM) algorithm [2].

Current methods could be roughly divided into two categories: generative methods and discriminative methods [3].

Generative methods explicitly construct generative models for the shape and appearance of the face. Feature locations are derived according to the best fit of the model to the test image. The classical active appearance models [4] which contain shape model, appearance model and motion model belong to this category. Its defect is that this method is not robust against occlusions. Cootes et al. [5] propose to calculate an appearance model for every facial part separately. In [6], the authors propose Gauss–Newton deformable part model which could construct generative models for each facial part simultaneously.

Discriminative methods aim to estimate the mapping between the facial appearances and the feature locations directly. Constrained local models (CLM) are able to learn independent local detectors for each feature point [7]. Then a shape model is used to regularise these local models. Yang et al. [8] propose a nonlinear correlation filter which can be used for individual face landmark detection and as a local expert for face alignment algorithms. Different from CLM, cascaded regression methods learn a vectorial regression function directly to calculate the face shape stage-by-stage. Explicit shape regression [9] is one of the first cascaded regression algorithms. It is a two-level boosted regression framework. Burgos-Artizzu et al. [10] introduce occlusion information into the regression process. Kazemi and Josephine [1] propose to use regression trees instead of random ferns and achieve super speed. Besides the abovementioned two-level boosted regression framework, Xiong and De la Torre [2] present a cascaded linear regression method using hand-crafted features. The contribution of [2] is a provable SDM. They also extend SDM to global SDM in order to cope up with the problem of conflicting gradient directions [11]. The SDM is a popular face alignment method especially for mobile applications since it could achieve state-of-the-art results in real 2D scenarios and still achieve real-time performances.

With the development of deep learning technology, deep neural networks have been successfully applied in many computer vision tasks in recent years. Sun et al. [12] are the first to use deep convolutional network cascade for face alignment. Trigeorgis et al. [13] propose a recurrent neural network approach. Recently, there have been studies to achieve 3D face alignment by fitting a 3D morphable model through convolutional neural networks [14, 15]. Bulat and Tzimiropoulos [16] propose a 3D face alignment network by stacking four hourglass networks. Although these methods perform better on the images with large head poses than traditional methods, they are still not easy to be applied in mobile platforms. The main reasons are as follows: (i) Deep learning model needs huge amount of training data which are not easy to collect for the face alignment task. (ii) The deep learning model is...
Input: $M$ training images $I_i (i=1\ldots M)$, ground truth landmarks $s_i (i=1\ldots M)$ for $M$ training images; initial estimations of the landmarks $s'_i (i=1\ldots M)$ for $M$ training images.

Output: Learned regressors $r_d (d=1\ldots D)$ for $D$ steps.

For $d = 1$ to $D$

1. Calculate $f(I, s'_{d-1}) (i=1\ldots M)$ for each training image.
2. Calculate $\Delta s'_{d-1} (i=1\ldots M)$ for each training image according to Eq. (3).
3. Solve for $r_d = \{A_d, b_d\}$ according to Eq. (2).
4. Calculate $s'_d (i=1\ldots M)$ for each training image according to Eq. (1).

End for

Fig. 2 Training process of SDM algorithm

usually several hundred MB which is too big for mobile applications. (ii) The computational cost is still quite high and can hardly achieve real-time processing on the mobile platform.

As mentioned before, SDM can achieve satisfactory results while maintaining relatively low computational costs. However, the trained model size of SDM could easily achieve more than 80 MB. It is still unacceptable for a commercial mobile application to perform the trained model of the SDM which could reduce the size to $\frac{1}{5}$ of the original size without obvious performance reduction.

In recent years, some research studies about deep learning network compression emerge. Iandola et al. [17] propose new kinds of convolutional operations so that parameters could be reduced. Li et al. [18] compress the network by pruning unimportant filters according to weight analysis. Rastegari et al. [19] and Courbariaux et al. [20] transfer the weights into binary values in order to reduce the size of the model. Most of the above methods are aimed at specific tasks or network structures and are questionable to be applied directly to face alignment networks.

This paper fails as follows: Section 2 briefly describes the SDM algorithm; Section 3 explains our proposed algorithm in detail; Section 4 shows qualitative and quantitative experimental results; and Section 5 draws the conclusion.

2 Brief introduction of SDM

In order to make this paper easy to follow, we briefly introduce the workflow of the SDM in this section. Please refer to [2] for details.

We assume that the face feature points are represented by $N$ 2D landmarks $s = [x_1, y_1, \ldots, x_68, y_68]^T$. Given a face image $I$ and the initial 2D landmarks $s_0$, we define a series of regressors:

$$ R = r_1 \ldots r_D $$

where $r_d = \{A_d, b_d\} (d=1\ldots D)$, $A_d$ is the projection matrix which could also be called the descent direction. $b_d$ is the bias term. In the SDM, $D$ is usually chosen between four and six. Thus the estimated 2D landmarks in the $d$th step can be calculated through:

$$ s_d = s_{d-1} + A_d \cdot f(I, s_{d-1}) + b_d $$

(1)

$f(I, s_{d-1})$ is the shape-related features which are SIFT values [21] or Histograms of Oriented Gradients (HoG) features [22] for better performances [23]. It could be calculated from landmarks $s_{d-1}$ in the image $I$. The final estimated face landmarks are $s_D$.

Given the $M$ training images, the SDM aims to minimise a series of the following equations and gets $r_d$:

$$ \arg \min_{A_d, b_d} \sum_{i=1}^{M} \| \Delta s'_{d-1,i} - A_d \cdot f(I, s'_{d-1,i}) - b_d \|^2 $$

(2)

where $\Delta s'_{d}$ are the shape residuals of the $i$th training image at the $d$th regression step:

$$ \Delta s'_{d} = s'_d - s'_{d-1} = s'_d - s_{d-1} - A_d \cdot f(I, s'_{d-1}) - b_d $$

(3)

$s_i$ are the ground truth landmark locations of the $i$th training image. Equation (2) is a standard linear least squares problem and could be solved in the closed form.

The training process of the SDM is listed as follows (See Fig. 2):

3 Data compression algorithm

The key components of the SDM are $r_d$ for all $D$ steps. Thus the trained model of the SDM stores all the information of $r_d$ for all $D$ steps. In a typical scenario, face landmarks include 68 points which is to say $N=68$. Fig. 3 shows the definition of the face landmarks.

Since we concatenate features calculated from each of these landmarks together and make a large feature vector, the dimension of the feature vector could easily achieve about 27,200 if we use HoG features. So the dimensions of $A_d$ are 136 $\times$ 27,200 and $b_d$ is a 136-dimensional vector. Each component in $A_d$ and $b_d$ is represented by a single-precision floating point number. Table 1 shows the data ranges of $A_d$ and $b_d$ for a typical HoG-based six-step regressor.

Since the data ranges vary in different regression steps, we compress the data separately for each step. Furthermore, since the data range of $A_d$ is very different from that of $b_d$, and $b_d$ only contains 136 floating point numbers in each step which is much smaller (only about 3KB for all six steps) than the size of $A_d$. In this paper, we propose to only compress the data in $A_d$.

3.1 Data distribution estimation

The data distribution of $A_d$ could not be described by a parametric model. So in this paper, we apply non-parametric methods to estimate the data distribution of $A_d$. Specifically, we use Parzen window method [24]. Assuming the number of elements in $A_d$ is $T$, window size is $h$, the probability density function (PDF) of the elements $x$ in $A_d$ could be estimated through:

$$ p(x) = \frac{1}{T} \sum_{i=1}^{T} \frac{1}{h} h\left(\frac{x - s_i}{h}\right) $$

(4)

Fig. 3 Definition of the face landmarks
In order to compress the element which is represented by a single-bit, the probability density distribution of the data. Thus, we have made it inefficient to use a uniform quantiser. In this paper, we propose an adaptive data quantisation method which is described before. Since there are different quantised levels with single-precision floating point numbers. There are usually much smaller than 32-bit floating point number. Through results. In order to constrain the errors introduced by data distribution inside the quantised region is not uniform. Instead, we set the quantised value as the mean value of all the elements falling in the same quantised region as follows. First, we calculate the set of all the elements which belong to the kth quantisation step: $V_k = \{ A_d(i) | v_{k-1} < A_d(i) \leq v_k, 1 \leq i \leq T \}$

Then the quantised value of the kth quantisation step could be calculated as follows:

$$q_k = \frac{1}{|U_k|} \sum_{i \in U_k} U(i)$$  (9)

### 3.3 Compressed model data storage organisation

The final compressed model consists of the stacked version of all $D$ steps of regressors $r_d$. Each regressor $r_d$ consists of three parts: the quantised values for all $Q$ quantisation levels; the quantised projection matrix $AQ_d$ which corresponds to $A_d$; the bias term $b_d$.

The quantised values are stored in the order of quantisation levels with single-precision floating point numbers. There are $Q$ floating point numbers for each regression step. The quantised projection matrix $AQ_d$ has the same dimension as that of $A_d$. Each element in $AQ_d$ is an index to the quantised value which is described before. Since there are $Q$ different quantised values, each element in $AQ_d$ only consists of log$_2$ $Q$ bits which is usually much smaller than 32-bit floating point number. Through $AQ_d$, we could use the index to get the corresponding quantised value and reconstruct the approximate projection matrix. In such a way, we achieve the data compression purpose. Throughout this paper, we choose $Q = 64$ which is justified in the experimental results section.

The bias term $b_d$ is stored directly as floating point numbers.

### 3.4 Training process embedding

If we quantise the final learned result of $A_d$, the quantisation process will introduce extra errors to the feature localisation results. In order to constrain the errors introduced by data storage.
quantisation, we modify the traditional training algorithm of the SDM described in Section 2 and propose the following algorithm (See Fig. 7).

In the above algorithm, we embed the data quantisation process into the training process so that the errors caused by quantisation in the previous step will be propagated into the next regression step. As a result, the projection matrix in the next step could partially correct the errors introduced by quantisation and the final results could be improved.

4 Experimental results

In this section, we compare our method with the standard SDM on the 300 W dataset [25]. We use the open source implementation of the standard SDM by Patrik Huber [26].

4.1 Choice of Q

In this section, we evaluate the face alignment accuracy according to the average distance between the detected landmarks and the ground-truth, normalised by the inter-ocular distance [25]. We vary the number of bits for each element in $AQ_d$ and calculate their corresponding normalised mean error loss against the standard SDM algorithm [26]. Fig. 8 shows the result.

From the above figure we can conclude that if each element in $AQ_d$ uses more than six bits, the loss is small and changes smoothly. However, if each element is represented by <6 bits, the error loss increases rapidly. In this paper, we choose 6 bits considering the balance between the error loss and compression efficiency. This means $Q = 2^6 = 64$.

In this paper, we use HoG features and localise 68 feature points. The number of regression steps is six. So the dimensions of $AQ_d$ and $A_d$ are both $136 \times 27,200$. The dimension of the bias vector $b_d$ is 136. For the standard SDM, all the elements are represented by 32-bit single precision floating point numbers. So the total space needed for the training model is

$$(136 \times 27,200 + 136) \times 4 \times 6 = 88,784,064 \text{ bytes}$$

With our proposed method, we need to store 64 single precision floating point quantised values, the $136 \times 27,200$ 6-bit matrix $AQ_d$ and 136-dimensional single precision floating point vector $b_d$ for each regression step. So our storage consumption for the whole training data is:

$$(64 \times 4 + 136 \times 27,200 \times 6/8 + 136 \times 4) \times 6 = 16,651,200 \text{ bytes}$$

Our model size is only about 18.75% of the standard one which means the compression rate of our proposed method is about 5.3×.

4.2 Qualitative experimental results

In this section, we compare our feature localisation results with the uncompressed training model [26]. Fig. 9 shows the results on 300 VW dataset. The left column is the result by running the SDM algorithm with an uncompressed training model. The right column is the result with our compressed training model. Red line in both columns is ground-truth feature locations. From this figure we can conclude that our compressed training model can generate very similar results with an uncompressed counterpart. They can both fit the ground-truth very well.

Fig. 10 shows the feature localisation results on our dataset. Since this dataset contains no ground truth information, we only show results with the uncompressed training model and our proposed compressed training model in the left and right columns, respectively. Again, we can conclude that our compressed training model could get similar results to the uncompressed one and our method is also robust to occlusion scenarios.

4.3 Quantitative analysis

As shown in Fig. 3, we divide the face features into five parts: face contour, eyebrows, eyes, nose and mouth. Face contour contains points No. 1 to No. 17. Eyebrows contain points No. 18 to No. 27. Eyes contain points No. 37 to No. 48. Nose contains points No. 28 to No. 36. Mouth contains points No. 49 to No. 68. We calculate the average value of the normalised mean errors in five parts, respectively, on 300 W test dataset. We compare our compressed training model with the uncompressed training model. The result is shown in Fig. 11. From the figure we can conclude that our
proposed compressed training model could get very close feature points compared with the uncompressed training model especially for eyes and face contour parts. These two parts are very important for AR-based applications.

Fig. 12 shows the cumulative error distribution curve of the two methods. It is obvious that the two curves are very close to each other. This again confirms the similar performance of both methods despite our training model being much smaller.

4.4 Ablation study

In this section, we study the effect of adaptive data quantisation and the training process embedding proposed in Sections 3.2 and 3.4, respectively.

In the first experiment, we compare adaptive data quantisation with uniform quantisation which is to say that the quantisation step is uniformly distributed inside the data range. We use the training process embedding step for both methods. The results are shown in Fig. 13. From the figure we can find that with uniform quantisation, the normalised mean error for each feature point will increase about 1%. Considering that the average of normalised mean error is about 3.5%, the effect of adaptive data quantisation is obvious.

In the second experiment, we analyse the effect of the training process embedding. We compare the results of using the training process embedding with the method without using training process embedding. We apply adaptive data quantisation for both methods. The results are also shown in Fig. 13. From the figure we can conclude that without training process embedding, the normalised mean error for each feature point will increase about 2%. This shows that training process embedding is even more important.

4.5 User study for mobile applications

We develop an AR mobile application with the SDM using our proposed compressed trained model. Sample effects of this application are shown in Fig. 14. This application virtually adds some interesting decorations to the face video in real-time.

We pre-record 20 short face videos of different people. The length of each video is about 30 s. We generate AR effects for
these face videos with our developed mobile application. Each face video generates two output videos: one with the uncompressed model SDM and the other with our proposed compressed model SDM. These two videos have the same AR effect. Different face videos may have different AR effects.

We recruited 20 people to score the results of two methods generated by 20 face videos. The range of the score is from one point to ten points. Half of the people are male and the other half are female. The ages of the test people range from 19 to 40. They are undergraduate students, graduate students and teachers. They have no relationship with this research project.

In this experiment, we show two output videos simultaneously on the monitor. The test subject scores the visual effects. The results demonstrate that 14 out of 20 people score exactly the same for two methods in all the videos. The scores of other six people are listed in Table 2.

From this table we can conclude that the visual effects generated by our compressed model are very similar to those of the uncompressed counterpart. This also proves the effectiveness of our proposed algorithm.

4.6 Computational cost analysis

The computational cost overhead for online feature tracking of our proposed method is to uncompress the data file. The face alignment process is exactly the same for our proposed method and Supervised descent [26]. Fortunately, the uncompressing process only needs to be done once before feature tracking. The computational time of our uncompressing process is about 20 ms on iPhone 6 which is negligible compared with the loading process of the mobile application.

5 Conclusion

This paper proposes an adaptive data compression method for the training model of the SDM face alignment algorithm. We propose an efficient method to quantise the data according to the probability density distribution of the data. Furthermore, we couple our quantisation method with the training process so that the accuracy loss is minimised. The experimental results proved that our proposed method could achieve comparable results with the
standard method while only needs <1/5 of the original storage space. Our proposed method shows potential for the SDM algorithm to be applied in mobile applications.

6 Acknowledgments
This work was partially supported by the National Natural Science Foundation of China under contract numbers 61501451, 51375323 and 61563022, the Cooperative Innovation Fund-Prospective of Jiangsu Province under grant BY2016044-01, the Major Program of Natural Science Foundation of Jiangxi Province, China, under grant no. 20152ACB20009, the high level talents of ‘Six Talent Peaks’ in Jiangsu Province, China, under grant DZXX-046, Qing Lan Project of Jiangsu Province, China.

7 References
[1] Kazemi, V., Jone, R. S.: ‘One millisecond face alignment with an ensemble of regression trees’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Columbus, USA, June 2014
[2] Xiong, X., De la Torre, F.: ‘Supervised descent method and its applications to face alignment’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Portland, USA, June 2013, pp. 532–539
[3] Jin, X., Tan, X.: ‘Face alignment in-the-wild: a survey’. Comput. Vis. Image Underst., 2017, 162, pp. 1–22
[4] Cootes, T., Edwards, G., Taylor, C.: ‘Active shape models—their training and application’. Comput. Vis. Image Underst., 1995, 61 (1), pp. 38–59
[5] Tzimiropoulos, G., Panitic, M.: ‘Gauss-newton deformable part models for face alignment in-the-wild’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Columbus, USA, June 2014, pp. 1851–1858
[6] Cristinacce, D., Cootes, T.: ‘Feature detection and tracking with constrained local models’. Proc. British Machine Vision Conf., Edinburgh, UK, September 2006, pp. 929–938
[7] Yang, W., Sun, X., Deng, W., et al.: ‘Fourier locally linear soft constrained MACE for facial landmark localization’, CAI Conf. Technol., 2016, 1, (3), pp. 241–248
[8] Cao, X., Wei, Y., Wen, F., et al.: ‘Face alignment by explicit shape regression’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Providence, USA, June 2012, pp. 2887–2894
[9] Burgos-Artizzu, X.P., Peroza, P., Dollar, P.: ‘Robust face landmark estimation under occlusion’. Proc. Int. Conf. Computer Vision, Sydney, Australia, December 2013, pp. 1515–1520
[10] Xiong, X., De la Torre, F.: ‘Global supervised descent method’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Boston, USA, June 2015, pp. 2664–2673
[11] Trigeorgis, G., Snape, P., Nicolaou, M.A., et al.: ‘Mnemonic descent method: a recurrent process applied for end-to-end face alignment’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Las Vegas, USA, June 2016
[12] Zhu, X., Lei, Z., Liu, X., et al.: ‘Face alignment across large poses: a 3D solution’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Las Vegas, USA, June 2016
[13] Jourabloo, A., Liu, X.: ‘Large-pose face alignment via CNN-based dense 3D model fitting’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Las Vegas, USA, June 2016
[14] Trigeorgis, G., Tzimiropoulos, G.: ‘How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230000 3D facial landmarks)’. Proc. Int. Conf. Computer Vision, Venice, Italy, October 2017, pp. 1021–1030
[15] Tzimiropoulos, G., Panitic, M.: ‘Supervised descent method and its applications to face alignment’. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Portland, USA, June 2013, pp. 3476–3483
[16] Courbariaux, M., Hubara, I., Soudry, D., et al.: ‘Binary neural networks: training deep neural networks with weights and activations constrained to +1 or −1’. CoRR, February 2016
[17] Iandola, F., Han, S., Moskewicz, M., et al.: ‘SqueezeNet: AlexNet-level accuracy with 50X fewer parameters and <0.5MB model size’. CoRR, February 2016
[18] Li, H., Kadav, A., Durdanovic, I., et al.: ‘Pruning filters for efficient convnets’. Proc. Int. Conf. Learning Representations, Toulon, France, April 2017
[19] Bottou, L., Bousquet, O., et al.: ‘Large-scale learning using adadelta’. Proc. Int. Conf. Machine Learning, Vancouver, Canada, June 2011, pp. 73–80
[20] Courbariaux, M., Hubara, I., Soudry, D., et al.: ‘Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or −1’. CoRR, February 2016
[21] Iandola, F., Han, S., Moskewicz, M., et al.: ‘SqueezeNet: AlexNet-level accuracy with 50X fewer parameters and <0.5MB model size’. CoRR, February 2016
[22] Courbariaux, M., Hubara, I., Soudry, D., et al.: ‘Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or −1’. CoRR, February 2016
[23] Li, H., Kadav, A., Durdanovic, I., et al.: ‘Pruning filters for efficient convnets’. Proc. Int. Conf. Learning Representations, Toulon, France, April 2017