Real-Time Generic Face Tracking in the Wild with CUDA

Shiyang Cheng[1], Akshay Asthana[1], Stefanos Zafeiriou[1], Jie Shen[1] and Maja Pantic[1,2]
[1]Department of Computing, Imperial College London, United Kingdom
[2]EEMCS, University of Twente, Netherlands
{shiyang.cheng11, a.asthana, s.zafeiriou, jie.shen07, m.pantic}@imperial.ac.uk

ABSTRACT

We present a robust real-time face tracking system based on the Constrained Local Models framework by adopting the novel regression-based Discriminative Response Map Fitting (DRMF) method. By exploiting the algorithm's potential parallelism, we present a hybrid CPU-GPU implementation capable of achieving real-time performance at 30 to 45 FPS, on ordinary consumer-grade computers. We have made the software publicly available for research purposes[1]

Categories and Subject Descriptors

I.5.4 [**Image Processing and Computer Vision**]: Applications—*Computer Vision*

General Terms

Performance,Experimentation.

Keywords

Face Tracking, Constrained Local Model, HOG, SVM, CUDA.

1. INTRODUCTION

The problem of facial landmark point detection and tracking has been studied rigorously for decades, and several model-based methods have been proposed [3, 5, 9]. Active Shape Model (ASM), proposed by Cootes et al [3], was one of the earliest part-based model for detecting the facial landmark points which utilized the global shape constraints and local appearance model (in form of image patches cropped around each landmark point). This methodology was extended in Active Appearance Model (AAM), proposed by Edwards et al [5], which modeled both the shape and appearance in a holistic manner. AAM used the global shape constraints but modeled the appearance based on a global texture model that was computed by warping the actual facial texture in a piece-wise affine manner to the mean facial shape of the 2D shape model.

In recent years, the Constrained Local Model (CLM) framework [1, 9], which originated from the ASM framework, has shown state-of-the-art results under uncontrolled natural settings. This method models the variations around each landmark point by training a local detector (known as *patch experts*) for each landmark point. Among the fitting optimization strategies for CLM, the Discriminative Response Map Fitting (DRMF) method [1] is the current state-of-the-art method in terms of fitting accuracy but the Regularized Landmark Mean-Shift (RLMS) [9] has been shown capable of real-time performance.

In this paper, we present a real-time CPU-GPU hybrid implementation of the DRMF framework [1], powered by Graphical Processing Unit (GPU), and explain the architecture of the system in detail. We conduct generic facial landmark localization experiments on the Multi-PIE [6], LFPW [2] and Helen [7] databases, and show that our DRMF implementation outperforms the RLMS framework in fitting accuracy by a margin. Furthermore, we evaluate the performance of our implementation on multiple computers with different hardware configurations. The current system is capable of tracking the face and estimating the 3D head-pose under uncontrolled natural settings at 30 to 45 FPS on conventional consumer-grade computers. Another benefit of the system is that it allows for the direct use of 3D shape model during the alignment procedure and therefore, can be easily used for applications that require 3D facial information. We demonstrate this via fully-automatic and real-time facial expression recognition application built on top of the current face tracking system.

2. OVERVIEW OF CLM FRAMWORK

The Constrained Local Model (CLM) is a typical part-based deformable model which utilizes the response maps generated by the local patch experts and optimizes the shape model parameters based on these responses. The advantages of CLM can be summarized as (1) by optimizing on response maps instead of the actual facial texture, unseen variations (such as pose, illumination and expression) can be well generalized, (2) the influence of partial occlusion is reduced since our interest lies only in local parts.

The standard CLM framework can be represented by the model $M = \{\mathcal{S}, \mathcal{D}\}$ where $\mathcal{S}$ is the shape model and $\mathcal{D}$ is the set of trained patch experts for each facial point in shape model. It can be described as a group of linear classifiers $\mathcal{D} = \{\mathbf{w}_i, b_i\}_{i=1}^{n}$, where $\mathbf{w}_i$ and b_i are the weights and biases of the i^{th} patch expert. The probabilistic response map (i.e. $p\left(l_i = 1 \mid \mathbf{x}, \mathcal{I}\right)$) for the i^{th} landmark being correctly aligned ($l_i = 1$) at location $\mathbf{x}$ of image $\mathcal{I}$ is modeled by a logistic function [1, 9]:

$$p\left(l_i = 1 \mid \mathbf{x}, \mathcal{I}\right) = \frac{1}{1 + e^{\{d(\mathbf{w}_i^T \mathbf{h}(\mathbf{x}; \mathcal{I}) + b_i) + c\}}}, \qquad (1)$$

where d and c are computed through a cross-validation procedure. $\mathbf{h}$ is a feature extraction function, and $\mathbf{h}(\mathbf{x}; \mathcal{I})$ denote the features computed from the local areas centered

[1] **Software and Videos:** http://ibug.doc.ic.ac.uk/resources.

MMSys '14, Mar 19-21 2014, Singapore, Singapore.
ACM 978-1-4503-2705-3/14/03.
http://dx.doi.org/10.1145/2557642.2579369

around location $\mathbf{x}$ in image $\mathcal{I}$. In this paper, we mainly use Histogram of Oriented Gradient (HOG) [4] features for its robustness against illumination and geometric transformation. We empirically set the parameters as follows: bin size is 9, cell size is 5 and block size equal to 2, and the clipping values for normalization is 0.2. For each patch expert, we compute HOG features on the set of positive and negative samples extracted from training images, and use Linear Support Vector Machine (SVM) to train the classifier due to its computational advantage on large scale datasets. Cross-validation is performed during the training process to select the best parameters of Linear SVM.

For the CLM based fitting, Asthana et al. proposed a discriminative approach named as Discriminative Response Map Fitting (DRMF) [1] to estimate the shape model parameters. Different from generative fitting method like RLMS [9], which maximizes the probability of a reconstructed shape, given that all landmark points are correctly aligned in the image, DRMF aims to find a mapping from the current response map estimate to the shape model parameter updates. Specifically, a perturbation of $\Delta\mathbf{p}$ is introduced and for the local area around i^{th} landmark point of this perturbed shape, we compute the response estimate $\mathbf{R}_i(\Delta\mathbf{p}) = p(l_i = 1 \mid \mathbf{x} + \mathbf{x}_i(\Delta\mathbf{p}))$. After that, from these perturbed response estimates $\{\mathbf{R}_i(\Delta\mathbf{p})\}_{i=1}^n$, we want to learn a mapping function f such that $f(\{\mathbf{R}_i(\Delta\mathbf{p})\}_{i=1}^n) = \Delta\mathbf{p}$.

The training procedure of DRMF [1] contains two main steps and can be summarized as follows. The first step is to learn a dictionary for the approximation of the response map, so as to extract the relevant feature for learning the fitting update model. PCA would be applied during the dictionary training procedure. The second step aims to learn the parameter update model iteratively which can be achieved by a cascaded regression based procedure. In [1], Linear Support Vector Regression (SVR) was used.

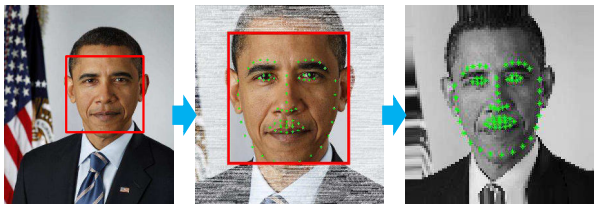
3. IMPLEMENTATION DETAILS

Our implementation of DRMF framework takes the advantage of several pre-processing steps, and exploits the potential parallelism in feature extraction and response map calculation using Compute Unified Device Architecture (CUDA) framework. The current system can easily perform the fitting at 30 to 45 FPS on conventional consumer-grade computers. The system is based on a fast CPU-GPU hybrid design of the HOG-CLM framework with DRMF method and consists of two main parts. The first part aims at reducing the computational complexity at CPU level. The second part is focused on how to utilize the CUDA framework to parallelize the algorithm efficiently.

3.1 Modified DRMF fitting

To reduce the computational cost, it is essential to remove some redundancies within the CLM framework while preserving the accuracy.

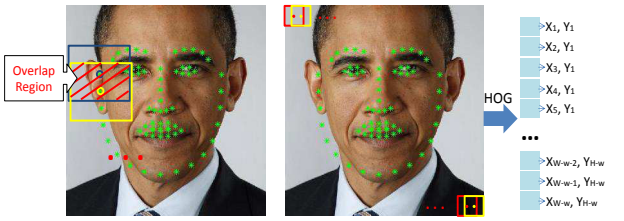
Image Resolution: In the CLM training procedure, there is a pre-processing step to register all the training images to a reference 2D shape, so as to remove any 2D rigid movement. During the fitting step, the same registration process will be applied as well. In general, the size of the reference shape (obtained by training the shape model on the images from Multi-PIE [6]) is 160×160 in the original scale. Within the local $N \times N$ region around each landmark point, we convolve a $w \times w$ patch expert to get the alignment probability response map. The complexity for computing responses of n points is $O(nN^2)$ without considering the HOG feature computation time. The size of local region is mainly determined by that of the reference shape as it needs to be large enough to capture discriminative facial structure and to achieve an optimal fitting performance. Instead of using the original scale, we empirically found that the 50×50 reference shape, with the local region size of 31×31 and the patch-size of 11×11, had the best balance between the accuracy and the computational cost. This pre-processing step is displayed in Figure 1.



(a) (b) (c)

Figure 1: Image Pre-processing Procedure. (a) Original image with bounding box. (b) Cropped image with an extended bounding box in the original scale. (c) Registered image with the Reference Shape of size 50×50.

HOG Feature Computation: For the HOG based DRMF method, we calculate the HOG feature of every $w \times w$ patch within the search region. For n landmark points, a total number of $n \times (N - w)^2$ feature vectors have to be computed. However, these features have significant redundancy as one pixel can contribute to several regions (See Figure 2(a)). To reduce the redundancy, we follow a look-up table based approach to retrieve the HOG instances within the designated region. We calculate the HOG features once for every patch within the $W \times H$ registered image, and create an index table to store the position of each window. The number of feature vectors to compute in this case becomes $(W - w) \times (H - w)$ (Figure 1(c)). To make sure that $(W \times H) \ll nN^2$, we crop the original image according to either the face bounding box provided by [10] or previous frame's tracking result. Although this approach introduces a new overhead of searching the look-up table, the overall reduction achieved in the HOG computation is much more significant. Moreover, due to the previously discussed registration step, the whole computation is invariant to the image resolution. The procedure this outlined in Figure 2.



(a) (b)

Figure 2: Two HOG Computation Schemes. (a) Sliding windows in n local $N \times N$ search regions. (b) $w \times w$ sliding window over $W \times H$ image and HOG feature look-up table.

3.2 CPU-GPU Hybrid Implementation

Our test shows the baseline implementation of the DRMF fitting procedure can only achieve 10 frames-per-second on

consumer-grade computers. This processing speed is insufficient for real-time facial expression analysis, which often requires at least 30 FPS to capture the fast changes in face region. Hence we identify the performance bottleneck in the algorithm and accelerate these steps by exploiting their potential parallelism using CUDA.

The per-frame DRMF fitting procedure consists of four steps: pre-processing (image cropping and coarse registration), HOG feature extraction, response map computation, and regression. We tested our baseline implementation on 7030 images from Multi-PIE database [6] on four consumer-grade computers. The average time spent on each step is shown in Table 1. The result shows that HOG feature extraction and response map computation are the most time-consuming steps. Together they take up roughly 88% of the execution time. Nonetheless, since the CLM assumes conditionally independence of local patch experts and the computation of HOG feature and regression are intrinsically parallelizable, these two step can by accelerated by parallelizing most of their computations using multiple threads running on GPU.

Table 1: Average per-frame processing time (ms) breakdown for Multi-PIE database on 4 test machines.

	A[a]	**B**[b]	**C**[c]	**D**[d]
Pre-process	0.99	0.87	0.89	0.74
HOG Feature	58.80	56.02	51.19	47.57
Response Map	50.73	43.97	43.70	36.15
Regression	14.31	12.38	12.53	10.37
Total	124.83	113.24	108.32	94.83

[a] Intel Core i7-2860QM with NVidia Quadro 5010M and 24GB RAM.
[b] Intel Xeon E5-1620 with NVidia GeForce GTS 450 and 24GB RAM.
[c] Intel Xeon E5-1620 with Zotac GeForce GTX 660 and 32GB RAM.
[d] Intel Core i7-4900MQ with NVidia GeForce GTX 780M and 16GB RAM.

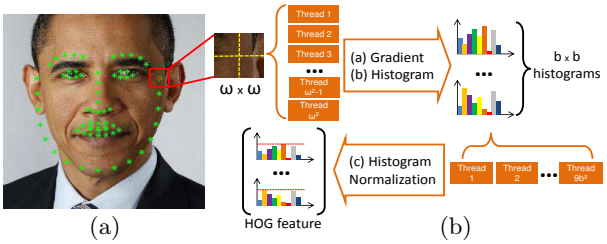


Figure 3: HOG Computation for a single window with CUDA. (a) Extract $w \times w$ patch from the image and assign it as a block. (b) Computation step for HOG in CUDA.

HOG feature computation with CUDA: The HOG features are extracted in in three steps: gradient computation, cell histogram creation with gradient binning, and block normalization. In the DRMF framework, response maps are computed only over a local region around each landmark point. Therefore, we require the HOG features instead of the responses from all the patches.

For gradient computation and cell histogram creation, we assign each patch to a separate block and, within it, each pixel to an individual thread. Since the patch is reasonably small (for example, 11×11), this assignment does not violate CUDA's per-block thread number limit. To avoid potential aliasing effect, a trilinear interpolation is performed when binning the value into the cell histograms [4]. After obtaining all the cell histograms, we use the L2-Hys norm for block normalization, with each bin of the normalized histogram assigned to an individual thread. This procedure (for a single patch) is illustrated by Figure 3.

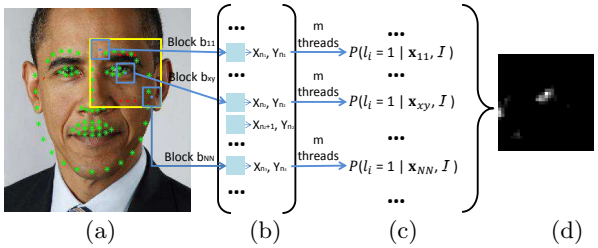


Figure 4: Response Map Computation with CUDA. (a) Extract pixel locations for i^{th} landmark. (b) Extract corresponding HOGs from look-up table. (c) Compute alignment probability. (d) Obtain the response map.

Response Map computation with CUDA: For each landmark point, a pre-trained SVM classifier is applied to compute the response map using HOG feature. In our CUDA implementation, we assign each landmark point to one block. Within the block, each individual thread extracts the feature value and computes decision value for a single feature dimension. Shared memory is used to sum up the decision values of each block and compute the alignment/misalignment likelihood for the landmark point. This procedure (for one landmark point) is illustrated by Figure 4.

Since the PCA projection and the shape parameter update is performed iteratively, with less than 1msec for each iteration (10 iterations in total), it is not very profitable to use CUDA. More importantly, the iterative fitting procedure is obviously not parallelizable. Although each iteration may be implemented using CUDA, the overhead imposed by frequent data transfer between the host (CPU) and the device (GPU) may outweigh the potential benefit of multi-threading. Therefore, the last step in the DRMF fitting remains to run on CPU. As a result, our DRMF implementation is a CPU-GPU hybrid approach. Moreover, the HOG feature and response map computation steps in CUDA have been designed in a more flexible manner as compared to [8] and can easily adapt the changes in HOG parameters.

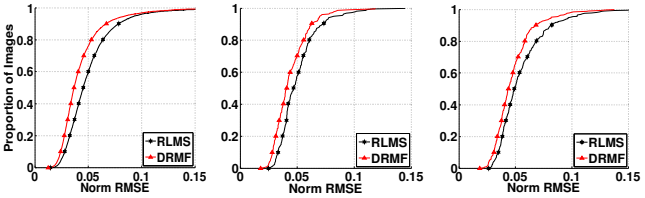
4. EXPERIMENTS

To evaluate the accuracy and speed of our CPU-GPU hybrid implementation of DRMF method, we conducted two main experiments. Firstly, we conducted the generic landmark localization experiment on both controlled (Multi-PIE [6]) and uncontrolled (LFPW [2] and Helen [7]) databases. For the initialization of localization experiment, we used the mean faces centered in the bounding boxes provided the our in-house face detector [1]. We employed Shape Root-Mean Square (RMS) error normalized w.r.t. the inter-ocular distance of the face to measure the fitting performance, which is unbiased for different size of faces. Secondly, we evaluated the CPU-GPU hybrid implementation in terms of its efficiency by showing the speed-up obtained over the single-threaded baseline version.

4.1 Landmark Localization Experiments

The purpose of this experiment is to compare the fitting performance of the DRMF method against the Regularized Landmark Mean-Shift (RLMS) [9] method. For controlled settings' experiments, all 346 subjects from Multi-PIE database were used. Among them, subjects 001-170 with pose range from $-30°$ to $30°$ in yaw direction, all ex-

pressions and two illumination conditions (one frontal illumination and another randomly selected non-frontal illumination) are used as training set. The total number of training images is over 8400. To achieve the best possible performance, we train multi-view models for both methods. The views are divided into three parts according to the yaw direction, $-30°$ to $-15°$, $-15°$ to $15°$ and $15°$ to $30°$. The test set consists of 7030 images from subjects 171-346 with varying expressions, pose and illumination variations. To test the performance of the DRMF method under uncontrolled settings, the Labelled Face Parts in the Wild (LFPW) [2] and Helen [7] database are used. We combine the training set of Multi-PIE with LFPW (811 images) and Helen (2000 images) to train both the RLMS and DRMF methods for this experiment. The test set of LFPW contains 224 images, and Helen includes 330 images.



(a) Multi-PIE (b) LFPW (c) HELEN

Figure 5: Results for Landmark Localization Experiments.

From the results in Figure 5, we observe that DRMF shows significant improvement over the RLMS method. The result on uncontrolled testset is particularly impressive considering that these images are captured under challenging real world conditions. It shows that not only does the DRMF method outperforms the RLMS method under controlled settings but also can efficiently handle the challenging variations of pose, illumination, expression and partial occlusion present in the LFPW and Helen database.

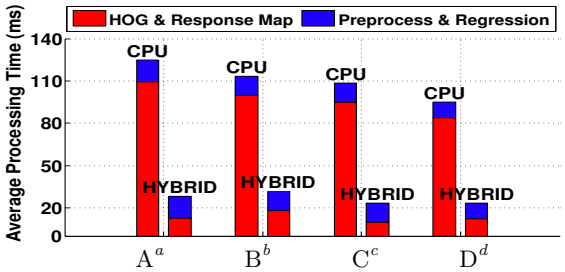


Figure 6: CPU-GPU hybrid implementation vs. Single-threaded CPU implementation on average processing time (ms) break-down. For each sub group, the left bar displays the processing time of CPU baseline version, while the right bar shows the processing time of hybrid implementation.

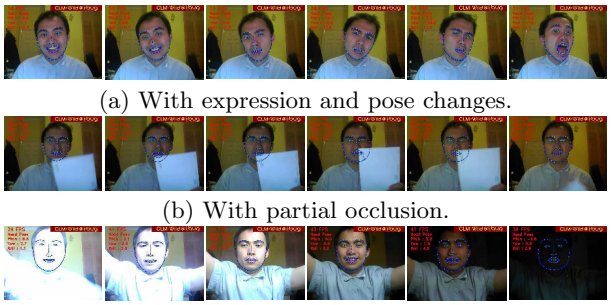
4.2 Computational Speed-up

To demonstrate the speed-up of the CPU-GPU hybrid implementation (comparing to baseline), the average per-frame execution time of the tracker on the Multi-PIE database is measured on the same computers listed in Table 1. The experiment result is presented in Figure 6. As shown in the figure, the CPU-GPU hybrid implementation runs much faster than the baseline version on all test machines. In particular, depending on the computer's specific hardware configuration, the steps implemented using CUDA (HOG feature extraction and response map computation) are 6 to 10 times faster than their counterparts in the baseline version and an overall speed-up of 3 to 5 times is achieved. Comparing to the baseline version's 10 FPS processing speed, the CPU-GPU hybrid is capable of reaching 30 to 45 FPS on our consumer-grade test machines. These results clearly show the advantages of using the GPU to exploit potential parallelism in DRMF to obtain substantial acceleration. See the sample tracking results in Figure 7.

5. THE DEMONSTRATION

In the live demonstration, we will show the CPU-GPU hybrid implementation based real-time system capable of tracking 66 facial landmark points on the face at 30-45 FPS on a conventional consumer-grade computer. Moreover, built on top of this system, a real-time 3D head-pose estimation and facial expression recognition application will also be demonstrated. The system will receive live video stream from a normal consumer-grade web-camera where the attendees will be able to interact with the system in real-time under uncontrolled natural settings. For comparison purposes, the non-parallel CPU implementation of the system will also be demonstrated so that the attendees are able to judge the benefits of using GPU for designing real-time systems, without compromising on accuracy.

Acknowledgement: The work of S. Cheng and S. Zafeiriou is partially funded by the EPSRC project EP/J017787/1 (4D-FAB). The work of A. Asthana is funded by Marie Curie Fellowship under FP7-PEOPLE-2011-IIF Grant agreement no. 302836 (FER in the Wild).



(a) With expression and pose changes.

(b) With partial occlusion.

(c) With illumination change.

Figure 7: Real-time Face Tracking Results.

6. REFERENCES

[1] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Robust discriminative response map fitting with constrained local models. In *IEEE CVPR*, 2013.

[2] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *CVPR*, 2011.

[3] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - their training and applications. *CVIU*, 1995.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[5] G. Edwards, C. Taylor, and T. Cootes. Interpreting Face Images Using Active Appearance Models. In *FG*, 1998.

[6] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. In *IEEE FG*, 2008.

[7] V. Le, J. Brandt, Z. Lin, L. D. Bourdev, and T. S. Huang. Interactive facial feature localization. In *ECCV*, 2012.

[8] V. Prisacariu and I. Reid. Fasthog - a real-time gpu implementation of hog. Technical Report 2310/09, Dept. Engineering Science, Oxford University, 2009.

[9] J. Saragih, S. Lucey, and J. Cohn. Deformable model fitting by regularized landmark mean-shift. *IJCV*, 91(2):200–215, 2011.

[10] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.