Exemplar based approaches on face fiducial detection and frontalization

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS by Research
in
Computer Science & Engineering

by

Mallikarjun B R
201307681
kiran.raj@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, India
February 2016

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Exemplar based approaches for face fiducial detection and frontalization" by Mallikarjun B R, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. C V Jawahar

To My Family

Acknowledgments

Acknowledgments

Abstract

Computer vision problems concerning faces play a vital role in security and other commercial applications. There has been decades of research work done by the vision community in solving the fundamental problems on faces such as face fiducial detection, face recognition, gender detection etc. Solving these problems leads to interesting applications in survellience, animation industries.

In our work, we consider two fundamental problems on faces. First, we discuss the problem of face fiducial detection and secondly we approach the problem of face frontalization. Face fiducial detection is one of the main pre-processing step done for face recognition, animation, gender detection, gaze identification and expression recognition systems. Approach suggested for face frontalization can be used in many of the facial animation application.

Face fiducial detection involves detecting key points on the faces such as eye corner, nose tip, mouth tips etc. Number of different approaches like active shape models [**?**], regression based methods [**?**], cascaded neural networks [**?**], tree based methods [**?**] and exemplar based approaches [**?**] have been proposed in the recent past. Many of these algorithms only address part of the problems in this area. We propose an exemplar based approach which takes adavantage of the complimentarity of different approaches and achieve consistently superior performance over the state-of-the-art methods. We provide extensive experiments over three popular datasets.

Face frontalization is the process of synthesizing frontal view of the face given a non-frontal view. Frontalization is used in intelligent photo editing tools and also aids in improving the accuracy of face recognition systems. Methods previously proposed involve estimating the 3D model or assuming a generic 3D model of the face. Esitmating an accurate 3D model of the face is not a completely solved problem and assumption of generic 3D model of the face results in loss of crucial shape cues. We propose an exemplar based approach which doesn't involve estimating a 3D model. We show that our method performs consistently better than the other approach considered and also efficient.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation and Objectives

Computer vision as a field includes making inference out of images. There has been decades of research happening in this field. We also have seen fruitful applications achieved in various domains. Be it reconstructing of 3D world which helps in automatic car navigation, analysis of medical images for early detection of diseases, analysis of astronimical images to study universe in general, biometric systems for security purposes using finger print or face images. Even though there has been decades of research, problems mentioned above are not completely solved as there is demand for more precision and efficient solutions. Technological advancement from other ends such as camera capablilites, computational capacity aids in solving problems which were not feasible earlier.

In this thesis, we are going to concentrate on approaches and applications on face images. We have seen a lot of generic problems such as face detection, face recognition, expression detection, gaze identification, face renactment etc currently being under research and also deployed in various systems. Most of the above system depends on pre-processing steps such as representation, face fiducial detection, tranformation etc which influences the effectiveness to a great extent. Specifically in this thesis we consider the problem of face fiducial detection and face frontalization whose approach are used as pre-processing steps.

Consider a basic face recognition system. To have the representation of an individual's face as a model, we need to be sure the representation of face has to be aligned to get a meaningful model. Face fiducial detection is used as a pre-processing step to extract the features at the corresponding locations while building the model. Similarly for an expression detection system, statistical distribution facial landmarks with each other helps in predecting accurate expression. Even the gender detection system would need facial landmarks as their initial point as fiducials are in general differently distributed for different gender.

Other pre-processing step which we discuss in this thesis is face frontalization. Given a profile view face, a frontal view face is synthesized. This approach helps in improving the accuracy of face

recognition system as the profile view faces would result in degraded performance. It is also used for face renactment where video of a person can be mimicked by replacing the face with another.

1.2 Methodology

> *Represent all the data with a nonparametric model rather than trying to summarize it with a parametric model, because with very large data sources, the data holds a lot of detail... Now go out and gather some data, and see what it can do.*
>
> – Alon Halevy, Peter Norvig, and Fernando Pereira, The Unreasonable Effectiveness of Data (Google, 2009)

The above statement is more so true when the data at hand is diverse at many fronts. In our case, the data is the face images which can have variations in terms of age, gender, expression, pose, facial structure etc. Trying to come up with a parametric model which holds all these information leads to poorly performing systems. Also whenever there is new data, the model has to be updated which is not so efficient. Rather its better to go with the data driven solutions.

Consider coming up with a generic model which represents an eye. Since the model has to perform in all different scenarios, we need to train the model with all possible variations. The samples could have variations with respect to the type of eyes, open or closed, visible or occluded, in frontal or profile view. When these samples are represented in appearance space, they end up in subspaces far apart. Training a model to capture all the above variations will be difficult. We could think of coming up with separate model for each of the variation above, but coming up with the labelled data for all these variations is tedious task.

For face fiducial detection, we employ exemplar based approach to to select the best solution from among outputs of regression and mixture of trees based algorithms (which we call candidate algorithms). We show that by using a very simple SIFT and HOG based descriptor, it is possible to identify the most accurate fiducial outputs from a set of results produced by candidate algorithms on any given test image. We also propose two different ways in which the exemplars can be selected and provide analysis how the preformance affects in choosing between two methods.

For face frontalization We employ an exemplar based approach to find the transformation that relates the profile view to the frontal view, and use it to generate realistic frontalizations. Our method does not involve estimating 3D model of the face, which is a common approach in previous work in this area. This leads to an efficient solution, since we avoid the complexity of adding one more dimension to the problem.

1.3 Contributions and Novelties

In this thesis, we propose exemplar based approaches for two fundamental problems related to face images. In both the cases, we provide extensive experimental analysis to show that the proposed approaches perform superior to the state-of-the-art methods on popular datasets. Face fiducial detection approach manifests as two algorithms, one based on optimizing an objective function with quadratic terms and the other based on simple kNN. Proposed face frontalization approach can be used either as a pre-processing step in face recognition, gender identification algorithms or also in rendering face video for face reenactment.

Proposed face fiducial is *initialization-insensitive, pose/occlusion and expression-robust* approach with the following characteristics,

- Our approach attempts the problem of fiducial detection as a classification problem of differentiating between the best vs the rest among fiducial detection outputs of state-of-the-art algorithms. To our knowledge, this is the first time such an approach has been attempted.

- Since we only focus on selecting from a variety of solution candidates, this allows our pre-processing routine to generate outputs corresponding to a variety of face detector initialization, thus rendering our algorithm insensitive to initialization unlike other approaches.

- Combining approaches better geared for sub-pixel accuracy and algorithms designed for robustness leads to our approach outperforming state-of-the-art in both accuracy and robustness.

We compare our approach with five of *state-of-the-art* methods on three popular datasets such as LFPW, COFW and AFLW. In some cases, we report as much as 10 improvement in the accuracy.
For face frontalization, we employ an exemplar based approach to find the transformation that relates the profile view to the frontal view, and use it to generate realistic frontalizations. In specific,

- Our method does not involve estimating 3D model of the face, which is a common approach in previous work in this area. This leads to an efficient solution, since we avoid the complexity of adding one more dimension to the problem

- Our method also retains the structural information of the individual as compared to that of a recent method, which assumes a generic 3D model for synthesis

We compare our approach with a recent *state-of-the-art* method. We provide qualitative comparision on various faces extracted from the videos available online. We also provide quantitative result on a face recognition dataset by frontalizing all the faces before the recognition task and show that our method performs significantly better and efficient.

1.4 Thesis Overview

In this chapter, we introduced the problem this thesis addresses and also the motivation in choosing the aforementioned problems along with the contributions. The rest of the thesis is divided into four more chapters. Chapter 2 introduces all the fundamental concepts briefly which are used in the thesis. Chapter 3 describes in detail about the face fiducial detection which includes defining the problem, related work, approaches proposed and quantitative comparitive experiments. Similarly Chapter 4 deals with face frontalization in detail pertaining the problem definition, previous methods proposed, our approach, qualitative and quantitative results. And finally we end with conclusion which illustrates the future direction of the work in Chapter 5.

2

Concepts

2.1 KNN

KNN is one of the simplest and effective classification algorithm used in the community. If one doesn't know the distribution of data at hand, it is generally preferred to go with kNN as wrong assumption of distribution in other algorithms leads to bad performance. kNN is an instance based learning or lazy learning as it depends on the samples from a small neighbourhood. All the training data is carried over till testing phase, where the label of the unknown test data is classified to a label represented by the majority label of its k-nearest neighbours. Figure **??** shows the example for labeling the test sample when K is equal to one and four.

The performance of KNN is dependent on the chosen value of K and also the distance metric used. The neighbourhood distance of the sample depends on the Kth nearest neighbour. Different K results in different distances and different conditional probablities. If value of K is very small, sample ends up with a small neighbourhood and could result in poor performance because of data sparseness, noise, ambiguous or poorly labelled data. If we try to smoothen bad effects by increasing the value of K, it results in introduction of outliers from other class and result in over smoothing.

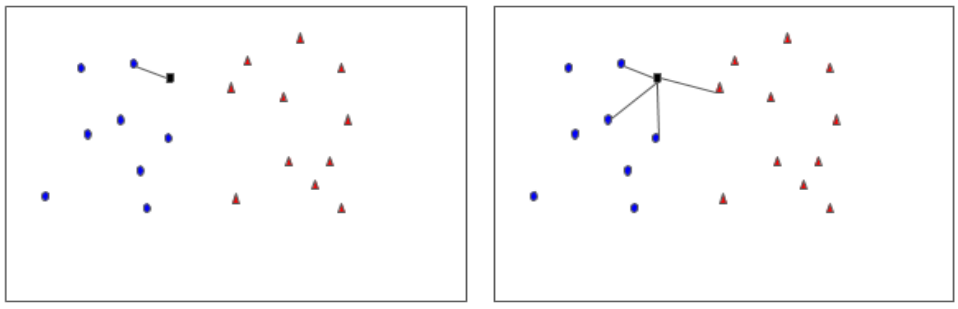


Figure 2.1: Left hand figure shows an example for 1-NN decision rule and the right hand figure shows the example for 4-NN

In our experiments, we use KNN in classification setting. Assuming face fiducial detection is a function given an image, we want to select one best performing function from among the functions in the appearance space where we have the training samples. The distance metric would give the degree of dissimilarity between the points.

2.2 Metric Learning

Type of metric used to measure between two points in KNN algorithm plays an important role in determining its performance. If no prior knowledge of the data is available, KNN algorithm is generally used with Euclidean distance measure. Since Euclidean metric doesn't hold any statistical measure of data with respect to labels, it would lead to sub-optimal solutions. [**?**], [**?**], [**?**], [**?**] shows that the performance of KNN improves by learning an appropriate distance metric. For example, distance metric learnt for face recognition task and the gender detection task would be significantly different.

Consider x and y as two points in a space. The Euclidean distance, d between these two points is defined as

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{2.1}$$

Even simple linear transformation learnt in supervised manner can lead to better performance in KNN classification as shown by [**?**] and [**?**]. Consider a linear transformation matrix, L which satisfies pseudo-metric properties such as triangular inequality, non-negativity, symmetry and uniqueness. The distance measured in the transformed space can be represented by,

$$d_L = ||L(x - y)||^2 \tag{2.2}$$

The generalization of Euclidean metric is the Mahalanobis metric. The Mahalanobis distance between two vectors is defined as,

$$d_M = \sqrt{(x - y)^T M (x - y)} \tag{2.3}$$

Where M has to be positive semi-definite matrix. Euclidean metric is a special case with $M = I$.

In our approaches we use distance metric to define degree of dissimilarity and also as probablity. The vector under consideration for our approaches are to deal with the appearance of key feature on the face or structural representation of the key features on the face. Clearly there is some statistical information embedded across various points derieved from these features. For example, we could see significant correlation within the points representing an eye corner or a nose tip. Also correlation is different when comparing between the eye corner and nose tip points. We would like to use these statistical information residing in the labelled data to derive a suitable distance metric to improve the accuracy of our KNN based algorithm.

More specifically we use *large margin nearest neighbour* (LMNN) classification algorithm to learn a Mahalanobis metric specifically designed for KNN classification. LMNN works on the intuitive idea that the test sample would be classified correctly if it lies near to samples of same label. LMNN learns a linear transformation by minimizing a loss function that consists of two terms, where first term ensures the samples matching the labels are pulled together and second term pushes the samples with non-matching labels with large margin.

2.3 Bayes' Theorem

Bayesian methods helps in providing coherent reasons in the face of uncertainity. Its based on mathematically handling uncertainity proposed by Bayes and Laplace in 18^th century and developed further by statisticians and philosophers in the 20^th century. Bayesian methods have emerged as popular models in the field of multisensory integration, motor learning, , neural computation and as the base of machine learning.

Bayes rule states that,

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} \tag{2.4}$$

It can be derived from basic probability theory. Here x can be considered as the data point and the θ as the model parameters. $P(\theta)$ is the probability of θ and is refered as the prior. Prior is obtained before observing any information on x. $P(x|\theta)$ is considered as *likelihood* and is the probablity of x conditioned on θ. $P(\theta|x)$ is considered as *posterior* probability of θ after observing x. $P(x)$ is the normalizing factor.

For a dataset of N points, $D = x_1, x_2, ..., x_N$, and model m with model parameters θ:

$$P(m|D) = \frac{P(D|m)P(m)}{P(D)} \tag{2.5}$$

We compute the above quantity for many different models m and select the one with highest posterior probability as the best model for our data.

$$P(D|m) = \sum_i P(D|\theta_i, m)P(\theta_i|m) \tag{2.6}$$

and is called the *marginal likelihood*.

To predict the probability of new data points, x^*, which have not been observed yet,

$$P(x^*|D, m) = \sum_i P(x|\theta_i)P(\theta_i|D, m) \tag{2.7}$$

where

$$P(\theta|D, m) = \frac{P(D|\theta, m)P(\theta|m)}{P(D|m)} \tag{2.8}$$

is the posterior probability of model parameters θ conditioned on the data D and is based on Bayes rule.

2.4 Convex Optimization

Optimization in mathematical sense is to select a particular sample out of all possible samples which yields best solution in some sense. Based on the sample spaces and the type of function which defines the outcome, we can categorize the optimization problem in various types. We describe few types of optimization and its solutions which are relevant to this thesis in the following section.

Linear optimization is special case of mathematical optimization in which the solution space is defined by the linear equality and inequality constraints with a linear objective function. The solution space is a convex polytope. Linear optimization problem can be canonically represented as

$$O(x) = \arg\min_{x}(c_T x) \tag{2.9}$$

subjected to $Ax < b$ and $x >= 0$

2.5 Affine Transformation

3

Fiducial detection

3.1 Introduction

Facial fiducial detection is an important problem with applications in facial expression recognition, gaze identification, face recognition etc. The task of identifying *several* locations for different components of a face in an image like ears, nose, mouth etc. becomes very daunting considering that each part might have a much more non-distinctive appearance profile than an entire face, and could also be subject to complete occlusion (Figure **??**, second row, eyes), drastic appearance and illumination variation (Figure **??**, third row, pose) or expression variation (Figure **??**, first row, mouth). Though there is no consensus yet on even the number of fiducial points assigned to a face [**?**], there is a broad realization among recent papers for the necessity to reduce failure rates and increase the accuracy of fiducial detection in a wide variety of challenging examples [**?**, **?**, **?**, **?**, **?**, **?**], since it automatically lends to better performance of systems that rely on fiducial detection.

While a number of different approaches like active shape models [**?**], regression based methods [**?**], cascaded neural networks [**?**], tree based methods [**?**] and exemplar based approaches [**?**] have been proposed in the recent past, many of these algorithms only address part of the problems in this area. Since datasets available today like COFW [**?**] , LFPW [**?**] (Figure 3.4) and AFLW [**?**] offer images varying widely in appearance, pose, expression, illumination and occlusion, each of these algorithms demonstrate their strengths in specific areas like occlusion handling [**?**], or robust performance in the case of profile views [**?**]. Indeed, while regression based approaches are better suited to perform well on metrics that measure pixel-wise accuracy of detection [**?**, **?**], exemplar or mixture-of-trees based approaches [**?**, **?**] are better suited to be more robust to pose change.

The surprising finding of our work is that many of these algorithms show decent complementarity in performance, which could be identified and exploited. In this paper, we present two algorithms that build on top of recent results in this space. Our kNN based algorithm is simple and effective, while our optimization algorithm provides a flexible framework to incorporate complicated models. Specifically, our algorithms use several state-of-the-art candidate algorithms [**?**, **?**, **?**, **?**, **?**] to generate fiducial points on a given image, and pose the detection problem as one of *selecting* the best result from the obtained

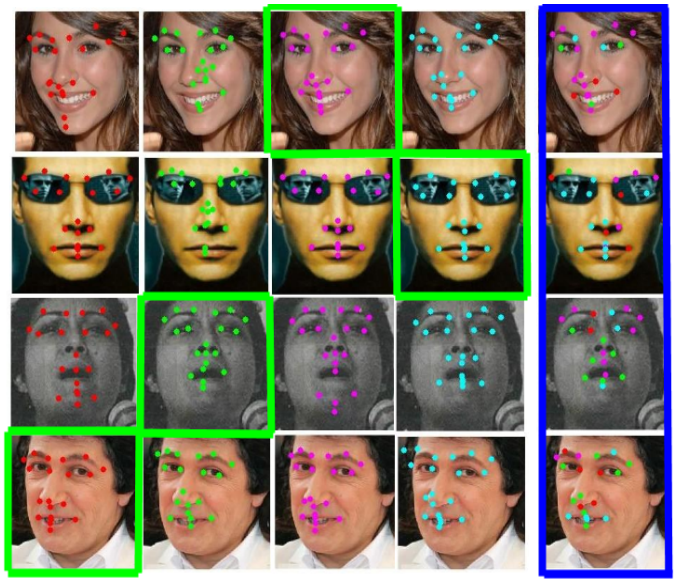


Figure 3.1: Fiducial detection of Chehra [3](red points), Zhu et al. [32](green points), Intraface [24](magenta points) and RCPR [8](cyan points) can be observed in column 1, 2, 3 and 4 respectively. Output selection by kNN is highlighted in green boxes. Last column shows the output selection by optimization highlighted in blue box. Best viewed in color.

outputs. By using several candidate algorithms, we ensure that we have access to the output of different approaches to fiducial detection, and thus reduce our problem to that of *classifying* between accurate and inaccurate fit to the data.

More formally, we propose an *initialization-insensitive*, *pose/occlusion and expression-robust* approach to face fiducial detection with the following characteristics

- Our approach attempts the problem of fiducial detection as a *classification* problem of differentiating between the best vs the rest among fiducial detection outputs of state-of-the-art algorithms. To our knowledge, this is the first time such an approach has been attempted.

- Since we only focus on selecting from a variety of solution candidates, this allows our preprocessing routine to generate outputs corresponding to a variety of face detector initialization, thus rendering our algorithm insensitive to initialization unlike other approaches.

- Combining approaches better geared for sub-pixel accuracy and algorithms designed for robustness leads to our approach outperforming state-of-the-art in *both* accuracy and robustness.

The outline of our paper is as follows. In section 3.2, we review related work with a perspective to distill out complimentary advantages of different approaches to fiducial detection. This is followed in section **??** by the formulation in section 3.3.1 and outline of our approach with focus on exemplar selection (section 3.3.4), output selection (section 3.3.5 for the kNN algorithm, section 3.3.6 for the optimization algorithm) and implementation details (section 3.3.9). We then follow up with an extensive

experimental section **??**, where we first show results on all the popular datasets like AFLW, COFW, LFPW and in each case present both mean part-wise pixel accuracy and failure-rate comparisons of our approach with the state-of-the-art.

3.2 Related Work

In this section, we categorize recent facial fiducial detection algorithms and discuss their advantages in brief.

3.2.1 Active Appearance Models (AAM):

The AAM framework has existed for almost two decades [**?**, **?**] and the traditional AAM based methods have not been suitable for fiducial detection *in the wild* [**?**, **?**]. However, some recent methods that deviate from the traditional pixel-value based texture model have shown new promise [**?**, **?**].

3.2.2 Constrained Local Models (CLMs):

The CLM framework has existed for a decade [**?**, **?**] and has been shown to be more capable of handling *in the wild* settings. In short, CLM is a part-based approach that relies on the locally trained detectors to generate response maps for each fiducial point followed by a simple Gauss-Newton method based optimization [**?**] for facial shape estimation. A regression based strategy for CLM optimization has also been proposed recently [**?**].

3.2.3 Exemplar Methods:

Exemplar based approaches have been popular since Belhumeur *et al*.'s work [**?**]. Zhao *et al*. [**?**] use gray scale pixel values and HOG features to select k-nearest neighbor training faces, from which they construct a target-specific AAM at runtime. Smith *et al*. [**?**] and Shen *et al*. [**?**] perform Hough voting using k-NN exemplar faces, which provides robustness to variations in appearance due to occlusion, illumination and expression. Finally, Zhou *et al*. [**?**] combine an exemplar-based approach with graph-matching for robust facial fiducial localization. Since, we build upon outputs of candidate algorithms, we take inherent advantage of the shape based regularization schemes employed by individual approaches and thus either side-step this problem (section 3.3.5) or *smoothen* candidate outputs using optimization (Figure 3.6) in our algorithms.

3.2.4 Cascaded Regression Based Methods:

Cascaded regression based methods are considered to be the current state-of-the-art for facial fiducial detection [**?**, **?**, **?**, **?**, **?**]. All these methods are capable for robustly handling *in the wild* settings in

real-time. In general, the training strategy is to synthetically perturb each of the ground truth shapes and extract robust image features (SIFT or HOGs) around each of the perturbed fiducial points. The regression is then used to learn a mapping from these features to the shape perturbation w.r.t the ground truth shape. Generally, a cascaded regression based strategy is adopted to learn this mapping and has been shown to converge in 4-5 iterations [**?**, **?**].

A recent work of Smith *et al*. [**?**] addresses the problem of analyzing the quality of facial fiducial results using an exemplars based approach. However, several difference exist between our approaches. Firstly, they work on a completely different problem of *aggregating* fiducials from different datasets and transferring them to a target dataset through Hough based feature *detection* [**?**], while the goal of the work presented in this paper is to *select* the best locations for each fiducial among the candidate locations provided by various candidate algorithms on every image. Secondly, they use algorithms like graph matching to ensure that the detected fiducials resemble a face [**?**], while we either side-step such issues (section 3.3.5) or handle them using optimization (section 3.3.6).

Recently, some promising attempts have also been made to approach the problem of facial fiducial detection in the deep-learning framework [**?**]. However, most of the proposed deep-learning based models work on low resolution images [**?**, **?**]. This prevents us from getting accurate fiducials on actual data. In this paper, we present a fully-automatic and principled approach for selecting the best fiducial location by combining results from multiple candidate algorithms for every image.

3.3 Algorithm

In this section, we first outline our formulation in section 3.3.1, followed by our algorithm for fiducial detection. Briefly, given an input image, candidate algorithms return vectors of locations of various fiducials for that image. Given the output of each of the candidate algorithms, our task is to identify a set of fiducials that best represent the face in the input image. This can be done by either selecting the *entire* output of *one* of the candidate algorithms, or by selecting *individual* fiducials from the various outputs of candidate algorithms to form a facial structure of our own. In order to do this, we first identify a set of *exemplars* from the training dataset, that serve as guidelines on how a face should look like, both in shape and appearance. Our approach is to then *match* candidate algorithm outputs to exemplars from the training dataset, in order to *select* the best output for the given image. Our algorithm has two main components: *exemplar selection* (section 3.3.4) and *output selection* (section 3.3.5, section 3.3.6). A flowchart of our approach is illustrated in Figure 3.2.

3.3.1 Formulation

Let $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ be a variable that represents the n locations of a set of fiducials. Let $\hat{X}$ denote the true locations of fiducial features in any given image I, while X_k refer to ground truth fiducials in the exemplar set used in our algorithm, where $k = 1 \ldots K$ indexes into the set of exemplars

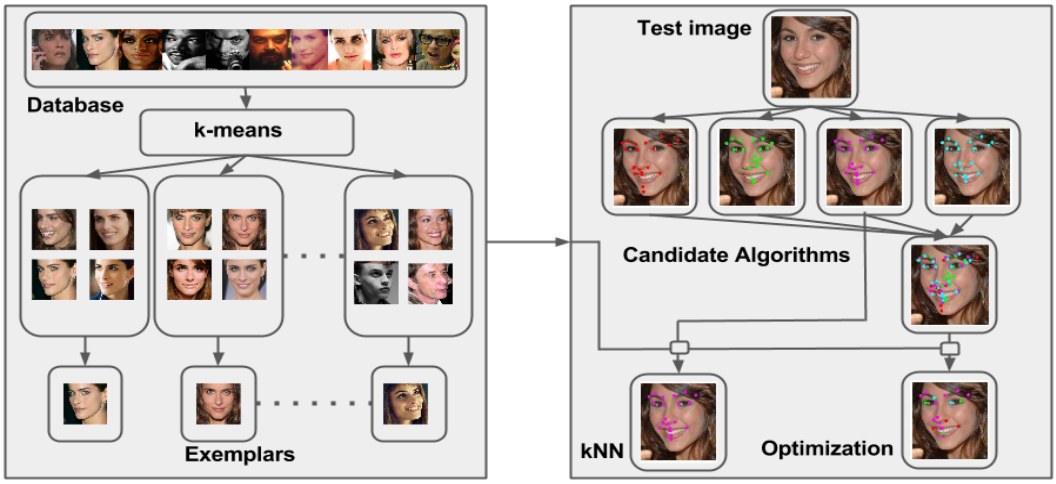


Figure 3.2: Left box pictorially represents exemplars selection. Right box represents our two algorithms for output selection. One by using kNN approach and other using optimization. Best viewed in color.

in consideration. In this paper, we consider $K = 20,\ \&\ n = 20$ since that is the set of common fiducials detected by algorithms presented in recent literature [**?**, **?**, **?**, **?**, **?**]. Note that recent approaches [**?**] offer a way to increase the number of common fiducial locations, and thus our assumption is not restrictive. Let $R = \{\mathbf{r}^1, \ldots, \mathbf{r}^m\}$ represent features extracted at m pixels on the image. We would like to optimize the following function to obtain the fiducial locations at the current image

$$X^* = \arg\max_{\tilde{X}} P(X \mid R) \tag{3.1}$$

Note that $\tilde{X}$ is the space of all possible sets of fiducial locations. It is a huge (40 dimensional) space, and sampling all of it is impractical. Instead, let us assume that we have been given some candidate locations where probability of a correct result is higher, and assume we will pick X from one of these locations. Let us depict these locations with the variable $\mathcal{X} = \{\bar{X}_1, \ldots, \bar{X}_l\}$, where $\bar{X}_i, i = 1 \ldots l$ are the number of candidates we have selected. We can now re-write equation 3.1 as

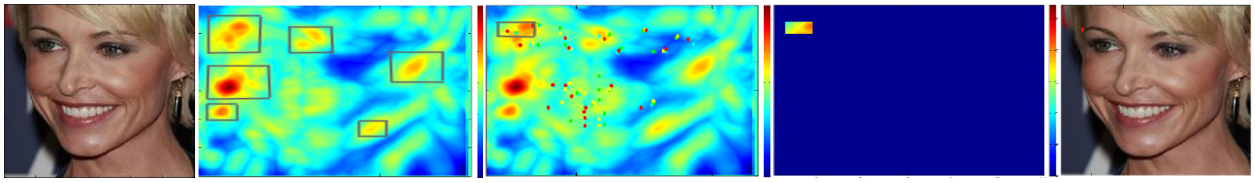
$$X^* = \arg\max_{\tilde{X}} P(X \mid R, \mathcal{X}) = \arg\max_{i} P(\bar{X}_i | R) \tag{3.2}$$

where we assume that the probability of selecting fiducials not represented by candidate algorithms is negligible. Using Bayes rule, and adopting a similar strategy of marginalizing over exemplars used in [**?**], equation 3.2 can now be elaborated as

$$P(\bar{X}_i \mid R) \propto P(R \mid \bar{X}_i) \tag{3.3}$$

$$\propto \sum_{k \in K} P(R \mid X_k, \bar{X}_i) P(X_k \mid \bar{X}_i) \tag{3.4}$$

where we marginalize over all exemplars X_k. Note that equation 3.4 *splits* the probability into comparison between *appearances* of our candidates and exemplars (first term), and comparison between



(a) Input Image (b) Distance from Exemplars (c) Output of Fiducials (d) Constrained Distance (e) Final Result

Figure 3.3: An example of fiducial detection of eye corner in a test image. Best viewed in color.

their shapes (term 2). Further, given structure is preserved in the way these two sets of candidates are generated, we can breakdown the above equation into parts

$$P(\bar{X}_i \mid R) \propto \sum_{k \in K} \prod_{j} P(R \mid \mathbf{x}_k^j, \bar{\mathbf{x}}_i^j) P(\mathbf{x}_k^j \mid \bar{\mathbf{x}}_i^j) \tag{3.5}$$

We denote individual probabilities for shape and appearance using the following functions

$$P(R \mid \mathbf{x}_k^j, \bar{\mathbf{x}}_i^j) = (1/\alpha) \exp(-\|F_k^j - F_i^j\|^2) \tag{3.6}$$

$$P(\mathbf{x}_k^j \mid \bar{\mathbf{x}}_i^j) = (1/\beta)\ dist(\mathbf{x}_k^j, \bar{\mathbf{x}}_i^j) \tag{3.7}$$

where F denotes concatenation SIFT and HOG features, while $dist$ is a scaled inverse Euclidean distance function and α, β are normalization constants to ensure both equations represent valid probabilities. Note that evaluating equation 3.5 entails summing over SIFT and HOG distances between candidate and exemplar fiducials. Finally, one could alternatively choose to optimize equation 3.2 using an optimization function as outlined in section 3.3.6. In this work, candidates are generated using algorithms of Zhu *et al.* [**?**], Xiong *et al.* [**?**], Asthana *et al.* [**?**], Artizzu *et al.* [**?**], and Tzimiropouluos *et al.* [**?**].

3.3.2 Example

In equation 3.5, the term $P(R \mid \mathbf{x}_k^j, \bar{\mathbf{x}}_i^j)$ can be seen as the term that *selects* appropriate exemplars given fiducial candidates using a *shape/appearance constraint* represented by equation 3.6. This is better illustrated with an example. In Figure 3.3, we show an input image for which the *minimum distance* in SIFT+HOG space from a set of exemplars is shown in Figure 3.3b, for a single fiducial (eye corner). Note how there are several minima in the distance map (marked by bounding boxes). Running candidate detection algorithms, however, generates eye fiducial candidates only in a specific region (Figure 3.3c, with bounding box), which is then selected and isolated using equation 3.7 (Figure 3.3d), leading to a correct location of the eye fiducial in the final output (Figure 3.3e).

3.3.3 Algorithm Outline

As explained earlier, our algorithm is divided into two main sub-parts: *exemplar selection* and *output selection*. The task in exemplar selection is to select a subset of face images with ground truth anno-

tations from the training dataset, that are representative of the *variation of pose, appearance including occlusion, expression etc*.of the dataset in consideration. Algorithm **??** gives an outline of our approach to exemplar selection. Note that while, we could use the entire training dataset annotations as exemplars, it suffices to have this limited set, as we will show in section 3.4.5.

This subset of annotated images then serve as our basis for differentiating between the various candidate algorithm outputs on any test image. The process of selecting the best fitting fiducials on any test image, given the exemplars, is called output selection.

3.3.4 Exemplar Selection

Exemplar selection is the process of selecting a subset of the training images along with fiducial annotations that represent the range of variations in pose/expression/occlusion in the dataset. We term the set of images selected eventually as the *exemplar set*. Ideally we would like the exemplar set to be representative of the training set in that we would like to be able to describe the pose/appearance of all images in the training set as some combinations of images in the exemplar set, in a specific representation space. For example, given annotations of fiducial locations in the training set, we would like have an exemplar set such that the shape of any training image annotation (represented as an ordered list of pixel coordinates of various fiducial points) is a *linear* combination of the annotations in the exemplar set.

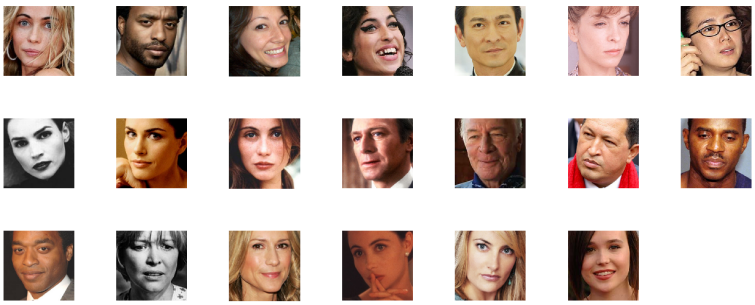


Figure 3.4: Examples automatically selected by our clustering approach in Section 3.3.4 for LFPW dataset. Best viewed in color.

Algorithm **??** illustrates our basic exemplar selection algorithm. The function `ComputeClusters` performs the operation of `kmeans` clustering in the vector space of fiducials, or feature vectors depending upon its input arguments. While the algorithm outputs two datasets for shape based and appearance based exemplars, note that shape based exemplars can be further divided into pose and expression classes and appearance based exemplars can also be tuned to include some examples of occlusion. However, we found that `kmeans` inadvertently does this since it clusters fiducials of the same pose but varying expression (shape clustering) or occlusion (appearance clustering) into one cluster.

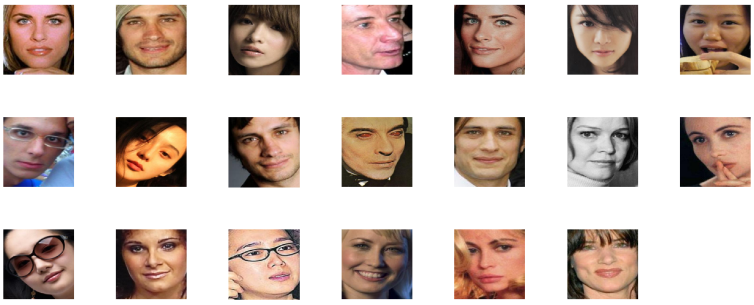


Figure 3.5: Examples automatically selected by our clustering approach in Section 3.3.4 for COFW dataset. Best viewed in color.

3.3.5 Output selection by kNN

Once the fiducial detection of the state-of-the-art candidate algorithms are obtained for an input image, we compute appearance vectors for an image patch around each fiducial location. Appearance vectors are represented in HOG and SIFT space. We concatenate these features them to form the feature vector.

We then compare these candidate algorithm feature vectors to the exemplars chosen from the previous approach, and choose the candidate algorithm-exemplar image output that minimizes the sum of euclidean distance between common features (equation 3.5). Note that this is a simple kNN based approach, where k=1. Alternatively, we also consider the idea of selecting individual fiducials from various candidate algorithm outputs, to form our own facial structure that minimizes an objective function. This is explained in the following section.

3.3.6 Output selection by Optimization

Instead of selecting fiducals from one method for all the parts as explained in earlier section, here we propose a method which selects fiducials for each part from best performing method. We first collect fiducials from all the candidate algorithms on an input image. Our task is now to select a subset of these fiducials for our output.

We propose an optimization framework based on equation 3.2, where we minimize a function based on appearence and structural costs. The appearance cost forces the areas around the fiducial locations in the input image to "look" like a face, while the structural cost ensures that the outline of fiducial locations resembles a facial structure. We define a quadratic objective function with unary and binary terms that enforce these constraints. Unary terms enforce appearance costs, while binary terms enforce structural costs.

The selection of the j^{th} fiducial from the i^{th} method is represented by the binary variable x_i^j. Let u_i^j be its appearence cost. Let y_{cd}^{ab} be the selection variable which will be 1 when both x_c^a and x_d^b are 1. And, p_{cd}^{ab} define the structural cost when y_{cd}^{ab} is 1. Thus y_{cd}^{ab} is the binary variable that represents *joint selection* of fiducials corresponding to unary variables x_c^a and x_d^b.

3.3.7 Appearance Costs:

We would want the fiducial prediction for each part to look similar to the corresponding fiducial of *one* of the exemplars. To do this, we compare the appearance feature vectors (using SIFT and HOG) between the fiducial x_i^j and that of the corresponding fiducials in the exemplar database. Let $f(x_i^j)$ represent the appearance feature vector corresponding to the j^{th} fiducial produced by the i^{th} method. We define the unary costs as

$$u_i^j = \arg\min_k \|f(x_i^j) - f(\mathcal{E}_k^j)\|^2 \tag{3.8}$$

where $\mathcal{E}_k^j$ denotes the j^{th} part of the k^{th} exemplar. Let $m(j,i)$ represent the exemplar index that has the fiducial closest in appearance to that of x_i^j. That is, let $u_i^j = \|f(x_i^j) - f(\mathcal{E}_{m(j,i)}^j)\|^2$.

3.3.8 Structural Costs:

We would also want to preserve the facial structure while selecting fiducials. This is most naturally enforced in the binary variable cost p_{cd}^{ab}. The importance of this cost is depicted in Figure 3.6. We enforce structural consistency by ensuring that if two fiducials x_c^a and x_d^b are selected, their corresponding closest exemplars (given by indices $m(a,c)$ and $m(b,d)$ as mentioned above) are as close to each other in shape as possible. Thus, we define the structural cost p_{cd}^{ab} as the euclidean distance between the shape of exemplars $\mathcal{E}_{m(a,c)}$ and $\mathcal{E}_{m(b,d)}$. Note that the structural cost is only defined between two variables that *do not* represent the same fiducial. That is

$$p_{cd}^{ab} = \|s(\mathcal{E}_{m(a,c)}) - s(\mathcal{E}_{m(b,d)})\|^2, \qquad a \neq b \tag{3.9}$$

where $s(\cdot)$ is the function that denotes the shape of a set of fiducials (represented as a vector of fiducial locations). Additionally, we also want to enforce the constraint that the same fiducial from different methods should not be simultaneously selected. This is easily enforced by the constraint

$$\sum_i x_i^j = 1 \tag{3.10}$$

Combining all the above, we want to minimize the following function function,

$$O(X,Y) = \sum_{i=1}^{5}\sum_{j=1}^{20}(x_i^j \times u_i^j) + \sum_{c=1}^{20}\sum_{d=c+1}^{20}\sum_{a=1}^{5}\sum_{b=1}^{5}(y_{cd}^{ab} \times p_{cd}^{ab}) \tag{3.11}$$

subjected to constraints, $x_i^j \in \{0,1\}$, $y_{cd}^{ab} \in \{0,1\}$, $\sum_{i=1}^{5} x_i^j = 1$, $y_{cd}^{ab} = x_c^a \times x_d^b$

Since the above problem has quadratic constraints and can not be solved in polynomial time, as the solutions are in integers, we relax the constraints **[?]** to get: $0 \leq x_i^j \leq 1$, $0 \leq y_{cd}^{ab} \leq 1$, $x_c^a \geq y_{cd}^{ab}$,

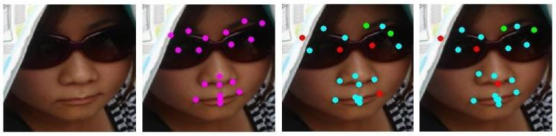


Figure 3.6: From left to right, we observe input test image, output selection by kNN, output selection by optimization without structural costs and output selection by optimization with structural costs. Observe that the left eye prediction suffers in third image because of not considering structural costs for optimization.

$x_d^b \geq y_{cd}^{ab}$, $x_c^a + x_d^b \leq y_{cd}^{ab} + 1$. Thus, we obtain the final linear optimization problem as

$$\begin{aligned} O(X,Y) \quad = \quad & \sum_{i=1}^{5}\sum_{j=1}^{20}(x_i^j \times u_i^j) + \\ & \sum_{c=1}^{20}\sum_{d=c+1}^{20}\sum_{a=1}^{5}\sum_{b=1}^{5}(y_{cd}^{ab} \times p_{cd}^{ab}) \\ & 0 \leq x_i^j, y_{cd}^{ab} \leq 1, x_c^a \geq y_{cd}^{ab}, x_d^b \geq y_{cd}^{ab} \\ & x_c^a + x_d^b \leq y_{cd}^{ab} + 1 \end{aligned} \tag{3.12}$$

We use `MOSEK` wrapper in MATLAB to solve the above optimization problem. Sometimes, because of the non-linear nature of the problem, we get non-integer solutions for x_i^j. In such cases, we take our fiducial location to be the average position of the top two selected outputs for the j^{th} part.

3.3.9 Implementation Details

In this section, we present some implementation details of the paper along with threshold values.

To compute the appearance vector around each fiducial part, we take 10x10 pixel patches and extract HOG features with a cell size of 3. We also compute the SIFT features around facial fiducial locations at two different scales of 5 and 8 pixels. After concatenating both the features, we obtain a vector of dimension 535 for each part. This is repeated for all the fiducial parts for both candidate algorithms and exemplars. For the experimentation, we used 20 clusters in k-means algorithm to automatically choose the trainnig samples to be used for kNN selection.

We took the author released code for candidate algorithms [**?**, **?**, **?**, **?**, **?**] along with the trained models. Experiments were conducted on the same test split for candidate and our algorithms for all the datasets.

3.4 Results

Thus far, we have outlined our approaches to fiducial detection in the previous sections. In this sections, we evaluate our algorithms on three state of the art datasets LFPW, COFW and AFLW. Before

we present the quantitative result (produced in Table 3.1) in the remaining part of this section, we describe the 3 datasets in brief below.

We have chosen 3 popular datasets to test the performance of our algorithm for several reasons.

3.4.1 LFPW

is the oldest dataset we consider [**?**], and contains faces of several people in "wild" settings, with lots of occlusions and pose / expression variation. It contains 1035 images, out of which 811 are used for training and 224 are used for testing purposes. Ground truth annotation of training images in the form of 68 fiducial locations for each face is available to us. This dataset has been standard for some time, but current algorithms give very good performance on it.

3.4.2 COFW

is a dataset released by Burgos-Artizzu *et al*. [**?**], and is specialzed to highlight situations where faces are occluded in a manner that hinders accurate fiducial detection by state-of-the-art algorithms. It contains 1852 images, out of which 1345 are used for training and 507 are used for testing purposes. Ground truth annotation of training images in the form of 29 fiducial locations for each face is available to us. This dataset is relatively new, and moderate performances have been reported on it.

3.4.3 AFLW

is a dataset released by [**?**], and contains several annotated face images in extreme settings. It is considered one of the toughest datasets in fiducial detection literature [**?**, **?**], as it has larger pose variations, partial occlusions and illumination variation compared to other datasets. Like [**?**], we sample 1000 training images and 3000 testing images randomly from the dataset, while ensuring no overlap between the two sets.

3.4.4 Quatitative Results

In this section, we outline the basis for future experiments detailed in the next sections. Table 3.1 shows results of our approach on LFPW, COFW and AFLW datasets. To produce these results, we first resize *all* images (training and testing) to a size of 300×300, and compute a set of 20 exemplars for each dataset using Algorithm **??**, equally divided between shape and appearance. Figure 3.4 illustrates our results of exemplar selection on the LFPW dataset. SIFT features for each fiducial are calculated at the scale of 5 and 8 pixels, which roughly translates to 4% and and 6% of the interocular distance. Once this is done, we proceed to the output selection by kNN and optimization based algorithms.

For each test dataset in Table 3.1, mean errors and failure rates in locating fiducials over the entire dataset are shown. For each fiducial, we first compute the ratio of its Euclidean distance from the ground truth and the interocular distance for that image. We then average this ratio over the entire image and

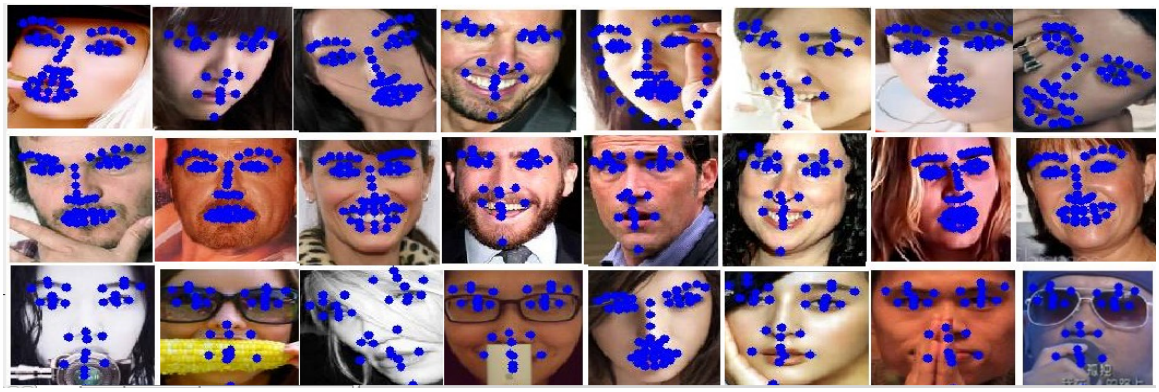


Figure 3.7: Results with varying pose (Row 1), expression (Row 2) and occlusion (Row 3). Best viewed in color.

Dataset	Chehra	Zhu	Intraface	RCPR	PO	Ours (kNN)	Ours (Opt)
LFPW	7.21	7.60	7.79	9.28	**4.82**	**4.31**	4.83
COFW	7.95	15.76	7.22	7.30	6.73	**5.98**	**6.28**
AFLW	40.44	**25.88**	47.98	39.78	46.67	**19.93**	32.08

Table 3.1: Table shows the mean error and failure rate for three datasets. In each row, top two algorithms are highlighted for both mean error and failure rate. Opt in the table represents output selection by optimization. Observe that both of our algorithms consistently perform better than state-of-the-art algorithms.

Dataset	Chehra	Zhu	Intraface	RCPR	PO	Ours (kNN)	Ours (Opt)
LFPW	20.98	15.62	17.41	17.41	**3.57**	**3.57**	5.8
COFW	21.89	49.70	18.15	14.20	9.27	**7.49**	**7.88**
AFLW	80.52	**71.28**	79.80	82.12	75.20	**59.03**	76.30

Table 3.2: Table shows the mean error and failure rate for three datasets. In each row, top two algorithms are highlighted for both mean error and failure rate. Opt in the table represents output selection by optimization. Observe that both of our algorithms consistently perform better than state-of-the-art algorithms.

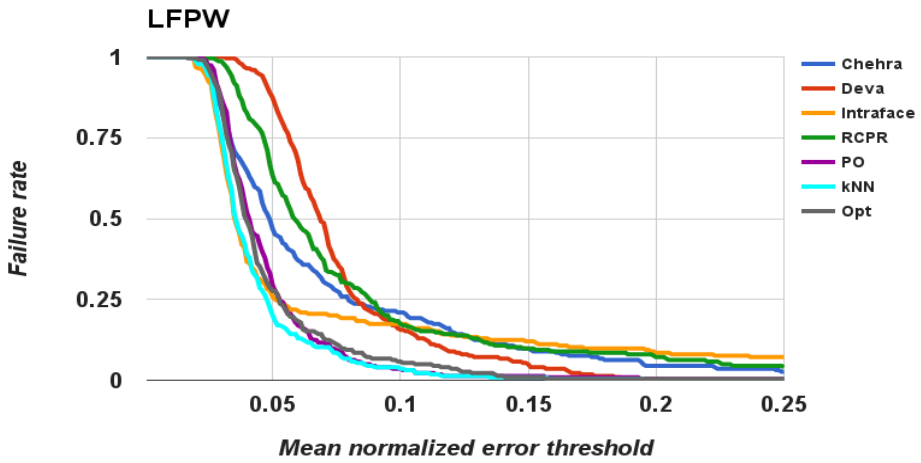


Figure 3.8: Results of our approach on (a) LFPW, (b) COFW, and (c) AFLW datasets. Drop in failure rate with the change in cut-off threshold of mean error normalized with interocular distance. Lower curve means more accurate results. Best viewed in color.

over the entire dataset. Thus the first table represents the *average ratio of fiducial error and interocular distance over the entire dataset*. The failure rate is the fraction of images in the entire dataset, for which this ratio is more than 0.1 (10% error). Thus, while mean error gives an idea of the accuracy of our algorithm, the failure rate gives an idea of its robustness.

A more detailed quantitative comparison of our approach with candidate algorithms is presented in Figure 3.10. Each point on the x-axis of this figure represents a cut-off threshold, and each corresponding point on the y-axis of this figure represents the fraction of images that have mean normalized error greater than this cut-off. Thus, graphs that dip quickly are more accurate. The mean normalized error is the mean of all interocular distance normalized errors over the entire dataset. We notice that both of our algorithms consistently perform better compared to other five algorithms at almost all cut-off ranges. Figure 3.7 illustrates some qualitative results using our approach.

3.4.5 Experimental Analysis

In the previous section, we outlined our basic algorithm and illustrated its results that show superior performance compared to state-of-the-art on three datasets. In this section we analyze various components of our algorithm to illustrate how our approach performs under different settings. Detailed results are provided in the website.

3.4.6 Runtime

For both approaches, candidate algorithms can be run in parallel and hence the total time taken by them on an input image is the maximum time of any algorithm. As an overhead, we compute SIFT/HOG based features on the output of these algorithms, which measures in milliseconds since

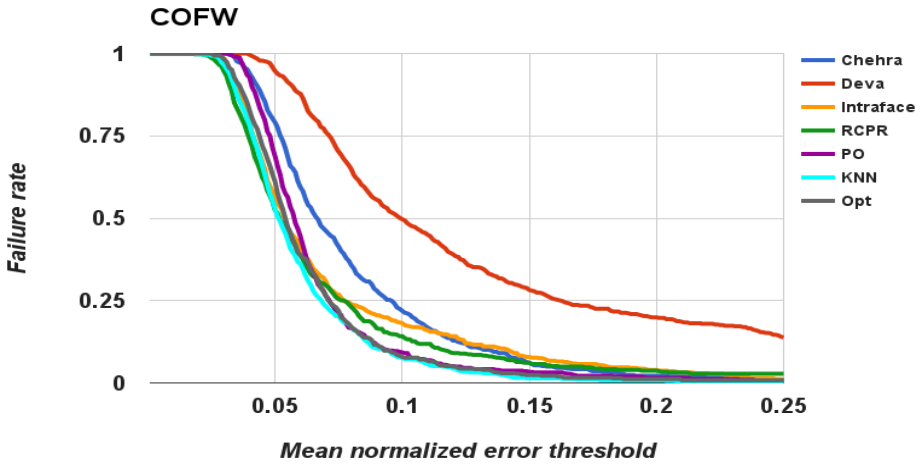


Figure 3.9: Results of our approach on (a) LFPW, (b) COFW, and (c) AFLW datasets. Drop in failure rate with the change in cut-off threshold of mean error normalized with interocular distance. Lower curve means more accurate results. Best viewed in color.

fast GPU based approaches are available for such computations. On top of that, the output selection part uses Euclidean distance computation for kNN, which amounts to 5 (candidate algorithms) x 20 (exemplars) distance computations between 535 dimensional vectors (of SIFT/HOG features). Finally, the optimization algorithm takes 0.4 seconds to converge for a single input image on a Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz system.

3.4.7 SIFT vs HoG

In this experiment, we contrast the contribution of SIFT and HOG features for the task of output selection. Results of our experiment comparing mean errors and failure rates on all datasets are shown in Figure 3.11b. Note that SIFT outperforms HOG, and understandably so since SIFT captures appearance details lost to HOG. We get an improvement of 6% using SIFT and 2% using HOG over competing methods.

3.4.8 Varying Number of Exemplars

Varying the number of exemplars ideally affects the accuracy of fiducial location, since more exemplars should typically mean that the nearest neighbor should be more similar to the test image. However if most variations in pose, expression, partial occlusion have been already captured, increasing the number of exemplars will have minimal effect on accuracy. This is precisely what we observe in Figure 3.12.

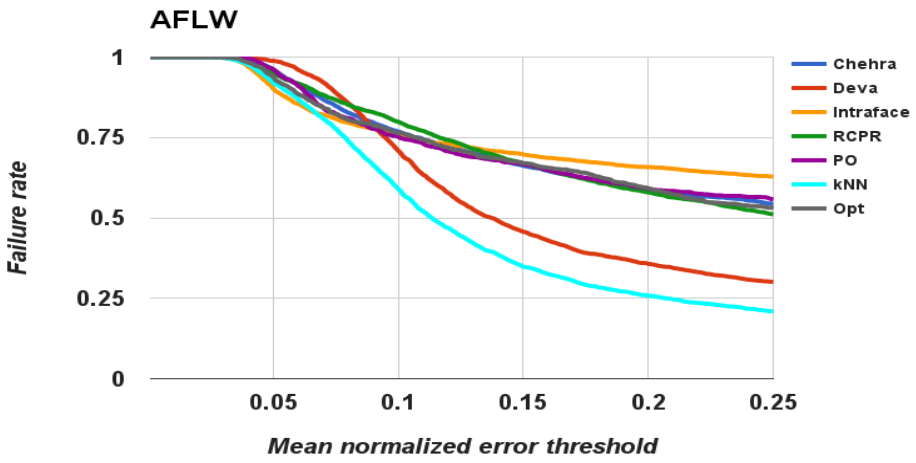


Figure 3.10: Results of our approach on (a) LFPW, (b) COFW, and (c) AFLW datasets. Drop in failure rate with the change in cut-off threshold of mean error normalized with interocular distance. Lower curve means more accurate results. Best viewed in color.

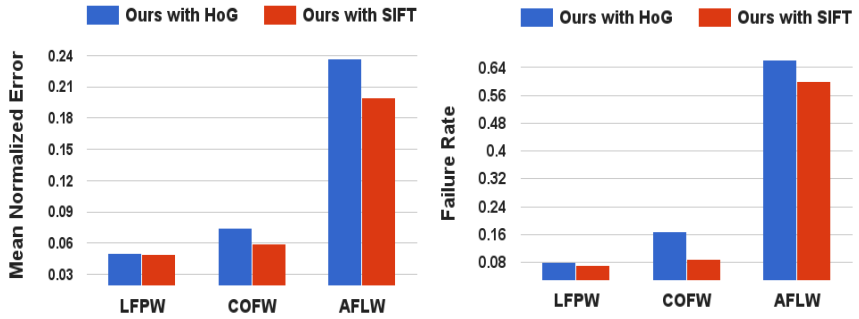


Figure 3.11: Comparison of mean error and failure rate for SIFT vs HOG experiment. Best viewed in color.

3.4.9 Optimization with structural costs

In this experiment, we show qualitative result of output selection by optimization with and without structural costs. Structural costs help in optimizing to a solution which looks like face. If only appearance costs are used, it leads to just selecting best looking fiducials individually leading to distortion in facial structure which can be observed in third image of Figure 3.6.

3.4.10 Shape vs Appearance

Algorithm **??** outlines our approach of using both shape specific and appearance specific exemplars in output selection. In this experiment, we measure the relative importance of each type of exemplar.

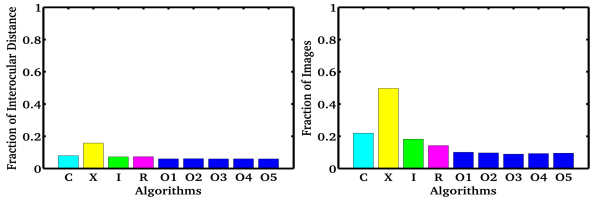
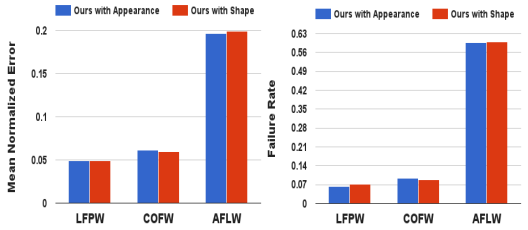


Figure 3.12: Comparison of mean error and failure rate when the number of exemplars is increased. Results O1-O5 correspond to our algorithm with number of exemplars (20, 30, 40, 50, 60) respectively. Best viewed in color.



(a) (b)

Figure 3.13: Comparison of mean error and failure rates for the shape vs appearance experiment. Best viewed in color.

Figure (refer supplementary material) shows results of our experiment, where we find that both have almost equal contributions to the superiority of output selection in comparison to competing methods.

Figure **??** shows our results when only one type of exemplars are used for output selection on the COFW dataset. Shape based and appearance based exemplars perform in a complimentary manner. Shape based exemplars provide robustness to partial occlusion, since they are better at identifying non-occluded fiducials, and generally result in nearest neighbors that are closer in pose to the test image. On the other hand, while appearance based exemplars falter in the presence of occlusion, they are better at identifying more accurate fiducials when all candidate algorithms give accurate outputs. Please refer to supplementary for additional outputs.

3.4.11 Clustering vs Eigenspace Analysis

While `kmeans` has been the preferred choice of clustering method for Algorithm **??**, we also experimented with using principal component analysis (PCA) instead. In order to select exemplars using PCA, we construct a shape matrix where each column represents an exemplar, and find its top 20 principal components. We then select one exemplar per component such that it maximizes its dot product with the corresponding principal component. Results, shown in the supplementary material, show negligible difference between the two approaches. We repeated the same experiment with appearance exemplars with similar results.

4

Frontalization

4.1 Abstract

Face *frontalization* is the process of synthesizing a frontal view of a face, given its non-frontal view. Frontalization is used in intelligent photo editing tools and also aids in improving the accuracy of face recognition systems. For example, in the case of photo editing, faces of persons in a group photo can be corrected to look into the camera, if they are looking elsewhere. Similarly, even though recent methods in face recognition claim accuracy which surpasses that of humans in some cases, performance of recognition systems degrade when profile view of faces are given as input. One way to address this issue is to synthesize frontal views of faces before recognition.

We propose a simple and efficient method to address the face frontalization problem. Our method leverages the fact that faces in general have a definite structure and can be represented in a low dimensional subspace. We employ an exemplar based approach to find the transformation that relates the profile view to the frontal view, and use it to generate realistic frontalizations. Our method does not involve estimating 3D model of the face, which is a common approach in previous work in this area. This leads to an efficient solution, since we avoid the complexity of adding one more dimension to the problem. Our method also retains the structural information of the individual as compared to that of a recent method [**?**], which assumes a generic 3D model for synthesis. We show impressive qualitative and quantitative results in comparison to the state-of-the-art in this field.

4.2 Introduction

Facial analysis in images for recognition/manipulation is a widely addressed and commercially important problem. Its applications range from surveillance to automatic tagging of photos on social websites. Recently, there are papers producing convincing results on *in-the-wild* datasets [**?**, **?**]. These datasets differ from previous ones in their unconstrained nature of image capture. However such methods have two drawbacks. Firstly, a lot of these methods have degraded performance in profile view vs frontal view. Secondly, they require lot of training data [**?**]. One way to alleviate both problems is to

Figure 4.1: Left image shows the profile face. Second image is *face frontalized* by our method. Third image is of Hassner *et al.* [**?**] method. Right image is the natural frontal view of the individual. *Frontalization* helps in face recognition.

be able to generate realistic frontal view faces for any person. This can be achieved, because faces have a definite structure. Eigen analysis [**?**], for example, has shown that faces exist in low dimensional sub spaces and can be represented as linear combinations of other faces. Also, it has been shown earlier that many face characteristics like expressions, hair etc. can be *transferred* from one person to another, in a very realistic manner [**?**].

In this paper, we show that a pre-processing step of synthesizing frontal pose of the face significantly improves the accuracy of face recognition. Face frontalization is the process of synthesizing frontal pose of the face, given a profile view of the face as shown in Figure 4.1. This step helps in simplifying the task of face recognition as recognition systems have more information and less occlusion to work with. Few methods counter this *frontalization* problem, by choosing to extract features only at the salient locations. Unfortunately, this leads to loss of structural relation between various parts of the face. However, as we will show in this paper, our method preserves this information as well.

Apart from aiding face recognition systems, frontalization techniques can also be used to generate a video out of a single image and can find applications in animation [**?**]. For example, if a family photograph has some people looking away from the camera, our approach can be used to correct this discrepancy [**?**].

Recent methods [**?**] [**?**] have proposed different ways of addressing the challenging problems of pose variations in images. Simonyan *et al.* [**?**] [**?**] choose to define features extracted out of large image regions to counter mis-alignments. Wolf *et al.* [**?**] [**?**] choose to align faces before extracting features. Sun *et al.* [**?**] use large datasets to create models robust to these challenges. In line with our approach, some recent works try to counter these challenging conditions of pose variation by synthesizing pose neutral faces from input images. Taigman *et al.* [**?**] try to estimate a 3D model of each input image. They then use this 3D information to synthesize the frontal view. On the other hand, [**?**] assumes a generic 3D model for all input images and produces convincing frontalization results. Even though the approach of [**?**] seems to be good, estimating 3D model from a single image is a hard problem. And assuming a generic 3D model in [**?**], leads to loss of structural information unique to an individual. Thus, in this work, we turn towards an exemplar based approach to fill the 3D information gap required by the previous approaches.

Lately, we have seen a surge of papers [**?**] [**?**] based on exemplar methods for solving computer vision problems. In these type of approaches, exemplars of the problem category are used instead of defining a generic model to solve the the problem at hand. For example, in the case of object detection, [**?**] trains a set of models using one positive exemplar each, instead of all the training set. And they show that the ensemble of such models give surprisingly good generalization. Similarly, our method is based on an exemplar based approach toward face frontalization. Consider a huge dataset of profile, frontal view face pairs of different. Chances of finding individuals with similar face structures to an input profile image is thus very high. Given such a match, the frontal view of the person in the database can then be used to frontalize the input image. Therefore, for our method, we collect a database of profile views and corresponding frontal view of a large number of individuals. We leverage the fact that faces lie in a low dimensional subspace and thus, many characteristics, like pose, expressions, etc. are transferable between people.

4.3 Face Frontalization

Our method takes as input, a profile view face, A_p, and an exemplar database, D, consisting of wide range of profile, frontal pose pairs for different persons. We then proceed to frontalize the face in two steps. First, we run facial landmark detection [**?**] on the input face, and using it we retrieve the most similarly posed face I_p^i and its corresponding frontal view face I_f^i, from database D. When profile views of two faces match, there is high likelihood that the two persons have similar facial structures. We exploit this property to get geometrical transformations required for frontalization of the input face. Simply put, we obtain the frontal view of A_p by using the affine transformations between A_p and I_f. One recently proposed state-of-the-art method [**?**] uses a generic 3D model for computing this transformation. This leads to loss of important discriminate structural information unique to an individual. Since we are finding a nearest profile exemplar and its corresponding frontal view face, structural information is still preserved for an individual face in our case.

Let $P = (X, Y)$, denote the landmark locations on the face, where $X = (x_1, x_2, .., x_{68})$ and $Y = (y_1, y_2, .., y_{68})$ are vectors of X and Y coordinates respectively. We consider 68 landmarks which includes feature points such as eye corners, nose tip, mouth line and jaw line. We use the dlib [**?**] implementation for landmark localization. Note that landmarks of images in D have been pre-computed. We also manually predefine 110 planes for a face using these landmarks. For example, the ends of two eyebrows and the beginning of the nose form a plane (see Figure 4.4). This has to be done only once since these planes connect the same fiducial points irrespective of the face. Each plane is defined by 3 landmark locations.

Let $T = \{t_1, t_2, \ldots, t_{110}\}$ represent the set of planes defined in 3D for a face image. Given planes of one profile-frontal image pair $T_p^m \& T_f^m$, in D, we define $H^m = \{H_1^m, H_2^m, \ldots, H_{110}^m\}$ as the affine transformations between corresponding planes, computed using point correspondences from the facial

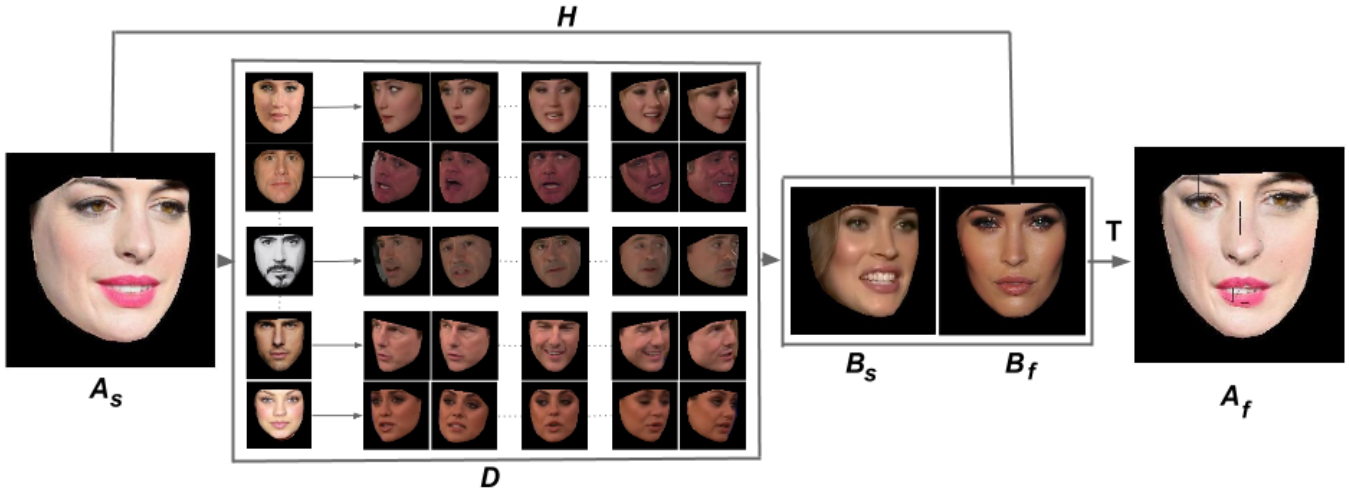


Figure 4.2: Figure shows the generic pipeline used in our approach. Given the input image (left most block) we use the exemplar database (second block) to compute the nearest profile view (third block, first image). We then use the correspondences between the profile and frontal views of the selected exemplar pair (third block) to compute the affine transformation H between the input image and the frontal exemplar, and use it to produce the *frontalized output* (right most block).

landmarks. That is,

$$t_f^{m,i} = t_p^{m,i} \times H_i^m \tag{4.1}$$

where the subscript p denotes profile, and the subscript f denotes frontal views.

Given the landmarks P for images in the database D, we now proceed to frontalize using the following steps.

4.3.1 Nearest exemplar selection

To retrieve the closest exemplar to the input face, we first need to define a similarity measure between faces. Let P^m and P^n represent landmarks of two faces. The similarity score between poses of two

Figure 4.3: First row of images are the input profile images. Second row shows the retrieved faces from database.

faces can then be defined as the Euclidean distance between P^m and P^n.

$$ds^{mn} = \sqrt[2]{\sum_{i=1}^{68}((x_i^m - x_i^n)^2 + (y_i^m - y_i^n)^2)} \tag{4.2}$$

However P^m and P^n are defined in different coordinate systems, separated by translation, rotation and scaling. We need to nullify the effect of translation and scaling and bring both sets of landmark positions to one coordinate system. Note that rotation is not considered as it is one of the parameters of pose and our exemplar database is exhaustive enough to take care of rotation variations in the input face. To remove the translational effect, we subtract the mean of X and Y coordinates from both the landmark vectors, $X = (X - \mu_X), Y = (Y - \mu_Y)$. To remove the scaling effect, we multiply P_m by a factor of s, given by

$$s = \frac{\sum_{i=1}^{68}((x_i^m \times x_i^n) + (y_i^m \times y_i^n))}{\sum_{i=1}^{68}((x_i^m)^2 + (y_i^m)^2)} \tag{4.3}$$

which follows from a straightforward optimization procedure that minimizes *root mean squared error* (rmse) error between corresponding landmark positions. The derivation is omitted for brevity.

Pose is accurately defined by the position of landmarks on the face. We concatenate the landmark locations obtained on the input image into a single vector called the pose vector. We then use Euclidean distance as the metric of comparison to retrieve the most similarly posed face from the database D. To get an accurate measure, we convert the pose vector of input face to exemplar face coordinate system. Let P^t represent the landmarks of profile input face and P_p^i represent the landmarks of profile exemplars available in the database. The nearest exemplar is the one which has the least ds^{ti}.

$$i^* = \arg\min_i d^{ti} \tag{4.4}$$

Given the nearest profile image I_p^i, we retrieve its frontal image and pose I_f^i, P_f^i. The first row of Figure 4.3 shows sample input faces and second row shows the nearest exemplars retrieved from D. Observe that men and women have slightly different facial structure, and this captured by our method, since women are retrieved as top exemplar candidates for input images of women.

4.3.2 Triangulation and Transformation

Once the frontal view of the nearest exemplar is obtained, we need to transform the input profile face to a frontal view. To do this, we first *transfer* correspondences between the exemplar pairs to the input image. This is done by replacing positions of the profile exemplar landmarks with those of the input image. Using the landmarks obtained, we define around 110 triangles on the face, each of which can be considered as a plane in the face coordinate system. Since the triangles are defined based on particular set of landmarks, we have correspondences between planes in the input image and corresponding frontal view exemplar. We obtain the affine transformations between the corresponding planes and then synthesize the frontal view of the input image using these transformations.

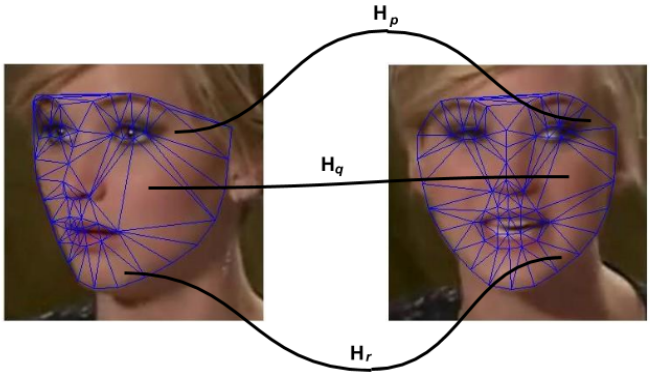


Figure 4.4: Image shows the planes represented as triangles and correspondences between two views of the same face. Note that each plane contains a fixed set of points irrespective of pose. For example, one plane contains two ends of the eyebrows and the top of the nose.

Figure 4.2 pictorially represents our method. For a given profile view input face, A_p, we retrieve most similar exemplar, I_p^i from the D along with its corresponding frontal view face, I_f^i. We then compute affine transformations, H^i, to transform planes of A_p to generate its frontal view.

4.3.3 Face Recognition

Our face recognition pipeline is based on the framework of Write *et al.* [**?**] who claim that faces of a particular individual lie in a low-dimensional subspace. In their method, training samples are represented as a 2D matrix, where each column represents a feature extracted from one training image. The input test sample should then be represented as linear combination of samples from the corresponding training samples of the same class (person). This problem has to be posed as an $l0$ minimization problem as it selects a combination of samples from training set. Based on recent advancement in sparse representation and compressed sensing, the authors claim that when the solution is sparse enough, solving $l1$ minimization is equivalent to the $l0$ minimization problem. Using this insight, a solution can be obtained in polynomial time using linear programming models. We use their implementation as the basis for our experiments, while we train using our dataset.

4.4 Experiments and Results

Content

4.4.1 Exemplar Database

For the exemplar database, we collected various face poses for 22 individuals (11 male and 11 female) from the talk shows available online. We selected sections which have complete swing of pose and expression changes. Approximately 15 exemplars and a frontal view were selected per individual.

Figure 4.5: First row shows the output of our method and the second row is of Hassner *et al.* [**?**] for LFPW [**?**] dataset. Observe ghost like appearances, structure distortion, mirroring effects in Hassner *et al.* [**?**] output.

In total of around 400 exemplars and 22 frontal view faces were collected. For face recognition experiment, we collected approximately 50 training and 50 input faces of 6 celebrities online. We call this dataset the $PoseInTheWildFaceDataSet$ (PIWFDS). We consider a new dataset, as existing datasets do not contain profile-frontal image pairs and even state-of-the-art recognition systems perform poorly on it.

All our experiments were conducted using MATLAB. We used HoG [**?**] feature based face detector to find faces and its output is re-scaled to a 300×300 image for both the database images and our input. This is given to facial landmark detection code based on [**?**], which is publicly made available. This provides 68 landmarks on each face. Using the landmarks we divide the face surface into 110 planes (triangular in shape). Using the corresponding planes between input profile image and the exemplar frontal view image, we compute the homography transformations matrix using publicly available implementation of vgg_Haffine_from_x_MLE. Using this set of homographies, we synthesize the frontal view of the input image.

4.4.2 Comparision with Hassner et al

Figure 4.6 shows the comparative results between our method and that of Hassner *et al.* [**?**] for PIWFDS dataset. To show that our exemplar database is generic enough to extend to standard datasets, we provide qualitative results for LFPW [**?**] dataset in Figure 4.5. We use Hassner *et al.* publicly released code to obtain the result.

Observe ghost like appearances present in most of the cases from Hassner *et al.* [**?**] output. Also take into consideration, that the face structure of the actress in second row has been changed to a generic one. This is because they use a generic 3D model of the face to achieve the result.

Figure 4.6: First row shows the output of our method and the second row is of Hassner *et al.* [**?**] for PIWFDS dataset.

4.4.3 Quantitative Results

For quantitative analysis, we used around 50 testing and 50 training samples of 6 classes for the face recognition task. Each sample is re-sized to a 300x300 image. After converting each sample from color to gray scale, we concatenated gray scale values to form a 90000 dimensional vector. We use Principal Component Analysis to reduce the dimensions to 40 using the training dataset. Each testing sample is also represented as 40 dimension vector as described above. We use publicly available implementation of Wright *et al.* [**?**] to recognize each input face.

Accuracy is calculated as fraction of testing samples classified correctly over the total number of samples. Our method achieved an accuracy of 31%, which is significantly better than 27% achieved by Hassner *et al.*.

4.5 Conclusion

Pre-processing steps such as alignment, pose correction are vital in object recognition systems. We solve one such problem in face recognition domain, by neutralizing the pose of input image. Our method is novel, efficient and simple. These techniques also find applications in intelligent photo editing. We produce decisive qualitative and quantitative results.

5

Conclusions and Future Work

Conclusion

Related Publications

- *Kiran Raj Ramamoorthy, Dip Sankar Banerjee, Kannan Srinathan and Kishore Kothapalli,* A Novel Heterogeneous Algorithm for Multiplying Scale-Free Sparse Matrices, **IEEE - IPDPS, ASHES 2015**.
- *[Submitted] Hardhik Mallipeddi, Kiran Raj Ramamoorthy and Kishore Kothapalli,* Nearly Balanced Work Partitioning via Sampling for Heterogeneous Algorithms, **IEEE - 2016** .

Bibliography

[1] J. Alstott, E. Bullmore, and D. Plenz. Powerlaw: a Python package for analysis of heavy-tailed distributions. PLoS ONE 9(1): e85777, 2014.

[2] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report No. UCB/EECS-2006-183, EECS Department, University of California, Berkeley.

[3] D. A. Bader, V. Agarwal, K. Madduri. On the Design and Analysis of Irregular Algorithms on the Cell Processor: A Case Study of List Ranking. IPDPS, 2007.

[4] D. A. Bader, and K. Madduri. GTgraph: A suite of synthetic graph generators. Available at https://sdm.lbl.gov/kamesh/software/GTgraph/

[5] D. S. Banerjee, K. Kothapalli. Hybrid algorithms for list ranking and graph connected components. HiPC, 2011.

[6] D. S. Banerjee, P. Sakurikar, and K. Kothapalli. Fast, scalable parallel comparison sort on hybrid multi-core architectures. In Proceedings of IPDPS Workshops, 2013.

[7] D. S. Banerjee, S. Sharma, and K. Kothapalli. Work efficient parallel algorithms for large graph exploration. HiPC 2013.

[8] M. Baskaran, and R. Bordawekar. Optimizing sparse matrix-vector multiplication on GPUs using compile-time and run-time strategies. Technical report, IBM Technical Report, 2008.

[9] N. Bell, and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report, 2008

[10] N. Bell, and M. Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In Proceedings of SC, 2009.

[11] G. E. Blelloch, J. C. Hardwick, S. Chatterjee, J. Sipelstein, and M. Zagha. Implementation of a portable nested data-parallel language. In Proceedings of PPOPP, 1993.

[12] G. E. Blelloch, M. A. Heroux, and M. Zagha. Segmented operations for sparse matrix computation on vector multiprocessors. Technical report, Pittsburgh, PA, USA, 1993.

[13] M. Boyer, K. Skadron, S. Che, and N. Jayasena. Load Balancing in a Changing World: Dealing with Heterogeneity and Performance Variability. In Proceedings of ACM Computing Frontiers, 2013.

[14] Aydin Buluc and John R. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In The 37th International Conference on Parallel Processing (ICPP), 2008.

[15] J. W. Choi, A. Singh, and R. W. Vuduc. Model-driven autotuning of sparse matrix-vector multiply on GPUs. In Proceedings of PPoPP, 2010.

[16] J. Chhugani, N. Satish, Changkyu Kim, J. Sewall, P. Dubey. Fast and Efficient Graph Traversal Algorithm for CPUs: Maximizing Single-Node Efficiency. IPDPS, 2012.

[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to algorithms. MIT Press, 2001

[18] J.K. Cullum, and R.A. Willoughby. Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Birkhauser, 1985.

[19] T. A. Davis. Direct Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2006.

[20] M. deLorimier, and A. DeHon. Floating-point Sparse Matrix-Vector Multiply for FPGAs. FPGA, 2005.

[21] J. W. Demmel. Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics, 1997.

[22] M. Garland. Sparse matrix computations on many-core GPUs. In Proceedings of DAC, 2008.

[23] R. A. V. D. Geijn, and J. Watts. SUMMA: Scalable universal matrix multiplication algorithm. Concurrency: Practice and Experience, 1997.

[24] A. Gharaibeh, L. B. Costa, E. Santos-Neto, and M. Ripeanu. On graphs, GPUs, and blind dating: A workload to processor matchmaking quest. In Proceedings of IEEE IPDPS, 2013.

[25] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in Matlab: Design and implementation. SIAM Journal of Matrix Analysis and Applications, 1992.

[26] G.H. Golub, and C.F. Van Loan. Matrix Computations. 2nd edition. Johns Hopkins University Press, Baltimore, 1989.

[27] C. Gregg, M. Boyer, K. Hazelwood, and K. Skadron. Dynamic heterogeneous scheduling decisions using historical runtime data. in Workshop on Applications for Multi- and Many-Core Processors, 2011.

[28] J. Greiner. A comparison of parallel algorithms for connected components. In Proceedings of SPAA, 1994.

[29] D. Grewe and Michael F. P. OBoyle. A Static Task Partitioning Approach for Heterogeneous Systems Using openCL. In International Conference on Compiler Construction, 2011.

[30] F. Gustavson. Two fast algorithms for sparse matrices: multiplication and permuted transposition. ACM Transactions on Mathematical Software (TOMS), 1978.

[31] D. S. Hirschberg, A. K. Chandra, D. V .Sarwate. Computing connected components in parallel computers. Communications of the ACM, 1979.

[32] Sungpack Hong, T. Oguntebi, K. Olukotun. Efficient Parallel Graph Exploration on Multi-Core CPU and GPU. PACT, 2011

[33] S. Hong, N. C. Rodia, and K. Olukotun. On Fast Parallel Detection of Strongly Connected Components (SCC) in Small-World Graphs. In Proceedings of SC, 2013.

[34] Q. Hou, K. Zhou, and B. Guo. SPAP: A programming language for heterogeneous many-core systems. Technical report, Zhejiang University, Graphics and Parallel Systems Lab, 2010.

[35] S. Indarapu, M. Maramreddy, and K. Kothapalli. Architecture- and Workload-aware algorithms for Spare Matrix-Vector Multiplication. ACM Compute, 2014.

[36] K. Koer, I. Grasso, B. Cosenza, and T. Fahringer. An Automatic Input-Sensitive Approach for Heterogeneous Task Partitioning. In ACM ICS, 2013.

[37] K. Kothapalli, S. Indarapu, S. Sharma, D. S. Banerjee, and R. Nigam. Workload Aware Algorithms for Heterogeneous Platforms, Under Submission, Available at http://cstar.iiit.ac.in/kkishore/workqueue.pdf, 2013

[38] Zhiling Lan, Valerie E. Taylor, and Greg Bryan. A novel dynamic load balancing scheme for parallel systems. JPDC, 2002.

[39] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. In Proceedings of the 37th annual international symposium on Computer architecture. ISCA 2010.

[40] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In Proceedings of ACM SIGKDD, 2005.

[41] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics, 2009.

[42] H. Ltaief, S. Tomov, R. Nath, and J. Dongarra. Hybrid Multicore Cholesky Factorization with Multiple GPU Accelerators. IEEE Transactions on Parallel and Distributed Computing, 2010.

[43] C. K. Luk, S. Hong, and H. Kim. Qilin: Exploiting Parallelism on Heterogeneous Multiprocessors with Adaptive Mapping. In Proceedings of International Symposium on Microarchitecture (MICRO), 2009.

[44] K. K. Matam, K. Kothapalli. Accelerating Sparse Matrix Vector Multiplication in Iterative Methods Using GPU. ICPP, 2011.

[45] K. K. Matam, S. B. Indarapu, and K. Kothapalli. Sparse matrix-matrix multiplication on modern architectures. HiPC, 2012.

[46] A. Monakov and A. Arutyun. Implementing Blocked Sparse Matrix-Vector Multiplication on NVIDIA GPUs. In Proceedings of SAMOS, 2009.

[47] A. Monakov, A. Avetisyan, and A. Lokhmotov. Automatically tuning sparse matrix-vector multiplication for GPU Architectures. In Proceedings of HiPEAC, 2010.

[48] R. Motwani, and P. Raghavan. Randomized Algorithms. Cambridge University Press, 2000.

[49] S. C. Park, J. P. Draayer, and S. Q. Zheng. Fast sparse matrix multiplication. Computer Physics Communications, 1992.

[50] Pawan Harish, and P. Narayanan. Accelerating Large Graph Algorithms on the GPU Using CUDA. In Proceedings of HiPC, 2007.

[51] G. Penn. Efficient transitive closure of sparse matrices over closed semi-rings. Theoretical Computer Science, 2006.

[52] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical Recipes, The Art of Scientific Computing. 2nd edition. Cambridge University Press, 1992.

[53] M. O. Rabin and V. V. Vazirani. Maximum matchings in general graphs through randomization. Journal of Algorithms, 1989.

[54] N. Satish, M. Harris, and M. Garland. Designing efficient sorting algorithms for many-core GPUs. In Proceedings of IPDPS, 2009.

[55] D. P. Scarpazza, O. Villa, and F. Petrini. Efficient Breadth-First Search on the Cell/BE Processor. IEEE Transactions on Parallel Distributed Systems, 2008.

[56] Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D. Owens. 2007. Scan primitives for GPU computing. In Proceedings of GH, 2007.

[57] Y. Shan, W. Tianji, Y. Wang, B. Wang, Z. Wang, N. Xu, and H. Yang. FPGA and GPU implementation of large scale SpMV. SASP, 2010.

[58] J. Shen, A. L. Varbanescu, P. Zou, Y. Lu, and H. Sips. Improving Performance by Matching Imbalanced Workloads with Heterogeneous Platforms. In Proceedings of ACM ICS, 2014.

[59] Y. Shiloach, and U. Vishkin. An O(log n) parallel connectivity algorithm. Journal of Algorithms, 1982.

[60] J. Siegel, O. Villa, S. Krishnamoorthy, A. Tumeo, Xiaoming Li. Efficient sparse matrix-matrix multiplication on heterogeneous high performance systems. Cluster Computing Workshops and Posters, IEEE 2010.

[61] J. Soman, K. Kothapalli, and P. J. Narayanan. Some GPU Algorithms for Graph Connected Components and Spanning Tree, Parallel Processing Letters, 2010

[62] Fengguang Song, Asim YarKhan, and Jack Dongarra. Dynamic task scheduling for linear algebra algorithms on distributed-memory multi-core systems. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 2009.

[63] P. D. Sulatycke, K. Ghose. Caching-efficient multi-threaded fast multiplication of sparse matrices. In Proceedings of IPDPS, 1998.

[64] S. Tomov, J. Dongarra, and M. Baboulin. Towards dense liner algebra for hybrid GPU accelerated many-core systems. Parallel Computing, 2009.

[65] V. Volkov, and J.W. Demmel. Benchmarking GPUs to tune dense linear algebra. In Proceedings of SC, 2008.

[66] R. Vuduc. Automatic performance tuning of sparse matrix kernels. PhD thesis, UC Berkeley, USA, December 2003.

[67] G. Wang and X. Ren. Power-efficient work distribution method for CPU-GPU heterogeneous system. In Proceedings of ISPA, 2010

[68] Zheng Wei, and Joseph JaJa. Optimization of linked list prefix computations on multi-threaded GPUs using CUDA. In Proceedings of IPDPS 2010.

[69] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel. Optimization of sparse matrix-vector multiplication on emerging multi-core platforms. In Proceedings of ACM SC, 2007.

[70] I. Yamazaki, Tingxing Dong, S. Tomov, J. Dongarra. Tri-diagonalization of a Symmetric Dense Matrix on a GPU Cluster. IPDPSW, 2013

[71] X. Yang, S. Parthasarathy, and P. Sadayappan, Fast Sparse Matrix-Vector Multiplication on GPUs: Implications for Graph Mining. In Proceedings of VLDB, 2011.

[72] R. Yuster, and U. Zwick. Fast sparse matrix multiplication. ACM Transactions on Algorithms, 2005

[73] L. Zhuo, V. K. Prasanna. Scalable and modular algorithms for floating-point matrix multiplication on FPGAs. In Proceedings of IPDPS, 2004.

[74] U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. Journal of the ACM, 2002.

[75] The Top 500 Super-computing sites. Available at http://www.top500.org

[76] Timothy A. Davis, The University of Florida Sparse Matrix Collection, In Proceedings of NA DIGEST 1994.

[77] Stanford Network Analysis Platform Dataset. http://www.cise.ufl.edu/research/sparse/matrices/SNAP/

[78] openMP Application Programming Interface. https://computing.llnl.gov/tutorials/openMP/

[79] CUDA: Compute unified device architecture programming guide. Technical report, NVIDIA.

[80] Intel Math Kernel Library. http://software.intel.com/en-us/articles/intel-mkl/

[81] nVidia cusp-library. http://cusplibrary.github.io/

[82] nVidia cusparse Library. http://developer.nvidia.com/cusparse/