4-16-2013

Joint optimization of manifold learning and sparse representations for face and gesture analysis

Raymond Ptucha

R·I·T

Joint Optimization of Manifold Learning and Sparse Representations for Face and Gesture Analysis

By

Raymond Ptucha

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY, COMPUTING AND INFORMATION SCIENCES

Dissertation Advisor:

Dr. Andreas Savakis, Professor, Computer Engineering, RIT

Dissertation Committee:

Dr. Nathan Cahill, Associate Professor, School of Mathematical Sciences, RIT

Dr. Joe Geigel, Associate Professor, Computer Science, RIT

Dr. Linwei Wang, Assistant Professor, Ph.D. Program, GCCIS, RIT

Defense Chair:

Dr. Marcin Lukowiak, Associate Professor, Computer Engineering, RIT

B. THOMAS GOLISANO COLLEGE OF COMPUTING AND INFORMATION SCIENCES

DEPARTMENT OF COMPUTING AND INFORMATION SCIENCES-PHD

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

APRIL 16, 2013

B. THOMAS GOLISANO COLLEGE OF
COMPUTING AND INFORMATION SCIENCES
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

The Ph.D. Degree Dissertation of Raymond Ptucha has been examined and approved by the dissertation committee as complete and satisfactory for the dissertation requirement for Ph.D. degree in Computing and Information Sciences

_____

Dr. Andreas Savakis, Chair (date)

_____

Dr. Nathan Cahill, Member (date)

_____

Dr. Joe Geigel, Member (date)

_____

Dr. Linwei Wang, Member (date)

_____

Dr. Marcin Lukowiak, Defense Chair (date)

_____

Dr. Pengcheng Shi, Ph.D. Program Director (date)

B. Thomas College of Computing and Information Sciences

Joint Optimization of Manifold Learning and Sparse Representations for Face and Gesture Analysis

By

Raymond Ptucha

April 2013

Abstract

Face and gesture understanding algorithms are powerful enablers in intelligent vision systems for surveillance, security, entertainment, and smart spaces. In the future, complex networks of sensors and cameras may disperse directions to lost tourists, perform directory lookups in the office lobby, or contact the proper authorities in case of an emergency. To be effective, these systems will need to embrace human subtleties while interacting with people in their natural conditions. Computer vision and machine learning techniques have recently become adept at solving face and gesture tasks using posed datasets in controlled conditions. However, spontaneous human behavior under unconstrained conditions, or in the wild, is more complex and is subject to considerable variability from one person to the next. Uncontrolled conditions such as lighting, resolution, noise, occlusions, pose, and temporal variations complicate the matter further. This thesis advances the field of face and gesture analysis by introducing a new machine learning framework based upon dimensionality reduction and sparse representations that is shown to be robust in posed as well as natural conditions.

Dimensionality reduction methods take complex objects, such as facial images, and attempt to learn lower dimensional representations embedded in the higher dimensional data. These alternate feature spaces are computationally more efficient and often more discriminative. The performance of various dimensionality reduction methods on geometric and appearance based facial attributes are studied leading to robust facial pose and expression recognition models.

The parsimonious nature of sparse representations (SR) has successfully been exploited for the development of highly accurate classifiers for various applications. Despite the successes of SR techniques, large dictionaries and high dimensional data can make these classifiers computationally demanding. Further, sparse classifiers are subject to the adverse effects of a phenomenon known as coefficient contamination, where for example variations in pose may affect identity and expression recognition. This thesis analyzes the interaction between dimensionality reduction and sparse

representations to present a unified sparse representation classification framework that addresses both issues of computational complexity and coefficient contamination.

Semi-supervised dimensionality reduction is shown to mitigate the coefficient contamination problems associated with SR classifiers. The combination of semi-supervised dimensionality reduction with SR systems forms the cornerstone for a new face and gesture framework called Manifold based Sparse Representations (MSR). MSR is shown to deliver state-of-the-art facial understanding capabilities. To demonstrate the applicability of MSR to new domains, MSR is expanded to include temporal dynamics.

The joint optimization of dimensionality reduction and SRs for classification purposes is a relatively new field. The combination of both concepts into a single objective function produce a relation that is neither convex, nor directly solvable. This thesis studies this problem to introduce a new jointly optimized framework. This framework, termed LGE-KSVD, utilizes variants of Linear extension of Graph Embedding (LGE) along with modified K-SVD dictionary learning to jointly learn the dimensionality reduction matrix, sparse representation dictionary, sparse coefficients, and sparsity-based classifier. By injecting LGE concepts directly into the K-SVD learning procedure, this research removes the support constraints K-SVD imparts on dictionary element discovery. Results are shown for facial recognition, facial expression recognition, human activity analysis, and with the addition of a concept called active difference signatures, delivers robust gesture recognition from Kinect or similar depth cameras.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Andreas Savakis, for without his guidance, mentorship, and encouragement, this thesis and the research behind it, would have not been possible. Dr. Savakis' faith and confidence in me from day one served as a beacon of hope throughout the Ph.D. process. I like to think his technical breadth, skill, and experience turned me into an academic professional. I would like to thank my committee, Dr. Nathan Cahill, Dr. Joe Geigel, Dr. Andreas Savakis, and Dr. Linwei Wang for their invaluable advice and assistance in making this thesis possible. I would like to thank Dr. Mrityunjay Kumar, for serving on my committee until he moved away from Rochester. I would like to thank Dr. Marcin Lukowiak for serving as the Defense Chair, keeping the thesis presentation and discussion both on track and of high academic merit. I would like to thank Dr. Pengcheng Shi for admitting me into the B. Thomas College of Computing and Information Science Ph.D. program, encouraging my progress, and then ensuring successful completion.

I would like to thank the numerous friends and colleagues I have developed over the years. Joyce Hart, Kathryn Stefanik, Pamela Steinkirchner, Anne DiFelice, and Richard Tolleson gave me great academic support over the years in too many ways to mention. Grigorios Tsagkatakis, Sherif Azary, Yuheng Wang served as great classmates with whom I have had many technical discussions. Dr. Justin Domke, Dr. Andrew Gallagher, Dr. Harvey Rhody, and Dr. Michael Yacci served as great technical liaisons. I would like to thank the Eastman Kodak Company for allowing me to return to school, and the great support I got from Brian Mittelstaedt, Rodney Miller, David Kloosterman, Larry Wolfe, Mike Devoy, and Terry Taber.

I would like to thank my wonderful children Kelsey, Stephen, Paige, and Alex for their patience and understanding as I attempted to be a good father and role model while going to school. Finally, I would like to thank my best friend and sole-mate, my wife Sandy. Her encouragement, support, and love make me the luckiest man.

Table of Contents

1 Introduction

In the not too distant future, we may live amongst a plethora of sensors and cameras situated precariously to aid and interact with humans. In public spaces, these systems will help shoppers trying to find the perfect gift, assist commuters looking for the next train, entertain customers waiting in line, detect and report crime, and serve as general informational dispensers. In private situations such as the home, office, and car, systems will learn individualized lifestyle and daily routines of its owners. Such private systems not only will greet occupants, but will also serve as personal assistants, placing calls, sending texts, managing calendars, and even giving personal advice. This technology, called pervasive or ambient intelligence will make life more efficient, informative, safer, and eventually weave itself into the fabric of everyday existence.

There has been much research on improving both the efficiency and overall experience of Human Computer Interaction (HCI) systems [1-3]. The study of computing that recognizes, interprets, and influences human emotions has spawned an entire field of study called affective computing [4]. Lew [3] argues that in order to achieve effective human to computer communication, the computer needs to interact with the human. Pantic [2] found that human judges relied on facial expression more than body gestures or vocal expression in the judgment of behavioral cues. The goal of HCI systems is twofold: 1) to have the computer engage and embrace all the human subtleties, that as a whole, convey the true underlying message; and 2) to interact with the human in his/her natural setting, eliminating ambiguous or awkward input modalities. Just as humans have adapted to the keyboard, mouse, and touchpad, a new modality will arise from which humans will communicate with computers. Kaplan [5] introduced a gesture based system to interact with everyday computers.

The introduction of low cost depth cameras such as Microsoft Kinect, along with advances in computer vision, has spawned an exciting new era for human-computer interaction. Depth sensors provide more salient information than RGB cameras for gesture recognition. Depth facilitates the extraction of objects against complex backgrounds and reduces the tracking of objects from a highly compute intensive task to one that is both simpler and more robust. Shotten et al. [6] have shown how Kinect depth images are segmented into body clouds, then converted to body parts, and finally to skeletal joints in real time. These depth cameras, which are capable of video resolution frame rates, have given the gesture recognition community a revolutionary leap in controller-less capability [7]. People can now become fully engaged with computers in natural settings without awkward input modalities. Recent progress in facial understanding and affective computing, coupled with gesture interpretation, are setting the stage for unprecedented human to computer experiences.

The notion of Sparse Representations (SRs), or finding sparse solutions to underdetermined systems, has found applications in a variety of scientific fields. The resulting sparse models are similar in nature to the network of neurons in V1, the first layer of the visual cortex in the human, and more generally, the mammalian brain [8, 9]. Patterns of light are represented by a series of innate or learned basis functions whereby sparse linear combinations form a surrogate input stimuli to the brain. Similarly, for many input signals of interest, a small number of exemplars can form a surrogate representation for a new test signal. Unfortunately, SR systems are not only compute intensive, but will be shown to suffer from a weakness known as coefficient contamination.

This thesis research introduces a new SR based machine learning architecture intent on overcoming these weaknesses with the goal of making SR systems suitable for interactive or ambient intelligence systems. After a brief discussion on related technologies, this thesis introduces the fields of dimensionality reduction and sparse representations. The combination of the two concepts into a single architecture is investigated with a method called Manifold based Sparse Representations (MSR). MSR optimizes each concept individually, and combines the two methods to achieve exciting results. The learnings from MSR research are then used to develop a more advanced and novel architecture called LGE-KSVD which jointly optimizes both manifold learning and sparse representation concepts into a single framework. This research focus was primarily developed for facial understanding, but the identical LGE-KSVD framework is utilized for both gesture and activity recognition.

With regards to facial understanding, localized key facial feature points, geometric, appearance, and hybrid methods are used in conjunction with supervised machine learning to resolve facial pose, expression, gender, race, age, and identity. Facial pose is listed first, as it has been shown that images of a single person under multiple poses has greater variation than images of different people at a single pose [10]. Expression is listed second, as faces are deformable objects. If we are to correctly classify the identity of a given face, two strategies mitigate the facial deformation and pose problem: 1) We may choose to first unwarp the pose and expression of the given face to some canonical representation; or 2) We may choose to pre-learn all possible pose↔expression combinations for each individual. While it is not fully understood how the human eye-brain accomplishes such an identification task so efficiently, the latter strategy is formidable in unconstrained environments. Facial understanding, including facial detection, tracking, feature extraction, and follow-on inferences is arguably one of the most widely researched fields in the computer vision industry. Although there is no consensus, a hybrid strategy often yields the best performance.

Correctly estimating human pose and expression will not only enable interactive displays, but the ability to normalize a face back to a frontal neutral expression enables better gender, race, age, and

identification. The implications for HCI and ambient intelligence systems are enormous. In addition to understanding the single frame instance of the human face, temporal behavior has been shown to include important clues in human to human interaction. For example, the Facial Action Coding System (FACS) [11] objectively characterizes 46 Action Units (AUs), each of which correspond to an independent motion of the face. Information such as onset, duration, and offset of each facial motion has been characterized by behavioral scientists in an effort to understand the complex nature of human emotion. When used in conjunction with multimodal systems, and when the semantical context of the emotion is understood, HCI, and ambient intelligence systems in general, will achieve unprecedented levels of intellect.

The contributions of this thesis will concentrate on the facial pose and expression portions of human-computer interaction, but is extended to general facial understanding and gesture recognition in the spatial and temporal domains.

1.1 Thesis Contributions

The key contributions of this thesis are:

1) Detailed analysis into the usage of dimensionality reduction methodologies for the purposes of facial understanding.
2) Detailed analysis of the necessary components and variation of such components used in combination with sparse representations for the purpose of face and gesture understanding.
3) The first method to robustly tackle coefficient contamination associated with sparse representation classification.
4) Introduction of LGE-KSVD, a machine learning framework that jointly optimizes semi-supervised variants of Linear extension of Graph Embedding with K-SVD dictionary learning.
5) The first published technique to jointly learn the dimensionality reduction matrix, sparse representation dictionary, sparse coefficients, and sparsity-based linear classifier.
6) While initially developed for static facial expression recognition, this work has been expanded to generic facial understanding problems in both static and temporal domains, and with the introduction of a novel concept called active difference signatures, has been adapted successfully to activity and gesture recognition.
7) These same methods can be directly applied to other domains in computer vision, pattern recognition, robotics, networking, and computing in general.

1.2 Thesis Overview

Chapter 1 presents motivation for the proposed research. Parsimonious behavior in biology inspired the technical selection of sparse representations. The development of semi-supervised dimensionality reduction minimized both coefficient contamination and compute resources. This classification methodology was applied to other facial understanding domains such as gender, race, and identification. The joint optimization of manifold learning with sparse representations has brought this thesis research to the forefront and the scope was expanded to include temporal aspects as well as gesture and activity recognition.

Chapter 2 covers several areas that are core to face and gesture recognition. This chapter begins with face detection and facial feature point localization. Facial pose and facial expression are studied further, with an emphasis on facial expression. Temporal processing is used to extend the methodologies which are used for face, gesture, and activity recognition. A section on depth cameras covers basics of people and skeleton extraction, and a follow-up chapter gives an introduction to gesture recognition. The last section covers notations used in this thesis.

Chapter 3 presents the fundamental concepts of dimensionality reduction. It begins with an overview of PCA and LDA, and then migrates to manifold learning and linear approximations to non-linear manifolds.

Chapter 4 presents several experimental studies that benefit from the dimensionality reduction techniques covered in Chapter 3. Facial pose and expression are independently investigated using localized facial feature points as well as facial image pixels. The chapter concludes with a joint manifold investigation that integrates both pose and facial expression into a single concept.

Chapter 5 presents the fundamental concepts of sparse representations. After a decomposition of sparse methodologies, this chapter reviews methods of using sparse coefficients as input into sparse classifiers.

Chapter 6 presents several experimental studies that utilize manifold learning and sparse representations into a single framework. After a description of coefficient contamination, a comprehensive analysis of various dimensionality reduction and sparse representation methodologies is performed. This section presents the Manifold based Sparse Representation (MSR) framework which is a state-of-the-art sparse representation classifier. Results are shown for facial expression and expanded to other facial understanding attributes. With the introduction of a temporal attributes, facial expression classification is improved further. With the incorporation of active difference signatures, MSR is shown to give excellent results on gesture recognition.

Chapter 7 presents LGE-KSVD, the first published method that simultaneously optimizes the dimensionality reduction matrix, sparse coefficients, sparse dictionary, and sparse coefficient classifier. This method also uses the novel method of infusing LGE concepts into the K-SVD framework to remove fixed support restrictions on K-SVD dictionary learning. The LGE-KSVD method is demonstrated to produce excellent results across a diverse set of problems including posed vs. natural datasets, small vs. large number of classes, static vs. temporal processing, and the recognition of expression, identity, and actions.

The **Conclusion** summarizes key findings and proposes several areas to pursue for future research on this topic.

Appendix I presents a brief discussion of the standard datasets used in this thesis, has a discussion on the posed vs. natural datasets, describes cross-validation methodologies, and concludes with a discussion on error metrics.

Appendix II presents the pixel processing techniques used, including facial bounding box schematics. This section also describes the pixel normalization used in the experiments.

Appendix III presents an overview of classification methodologies including linear regression, logistic regression, k-Nearest Neighbor, artificial neural nets, and support vector machines.

Appendix IV provides a list of software libraries used for this research that are freely available for download on the web. Two "hello world" sample codes are provided with the digital version of this thesis, one explaining how to use dimensionality reduction, cross-validation, and support vector machines; the other describing how to use sparse representations and sparse representation classifiers.

2 Background

There are several underlying technologies and concepts used in this thesis. A brief description of topics that have been essential to completing this research are included in this chapter. This section starts with an overview of face detection, followed by facial feature point localization, facial pose, facial expression, temporal processing, depth cameras, gesture recognition, and a final section describing notations used in this thesis.

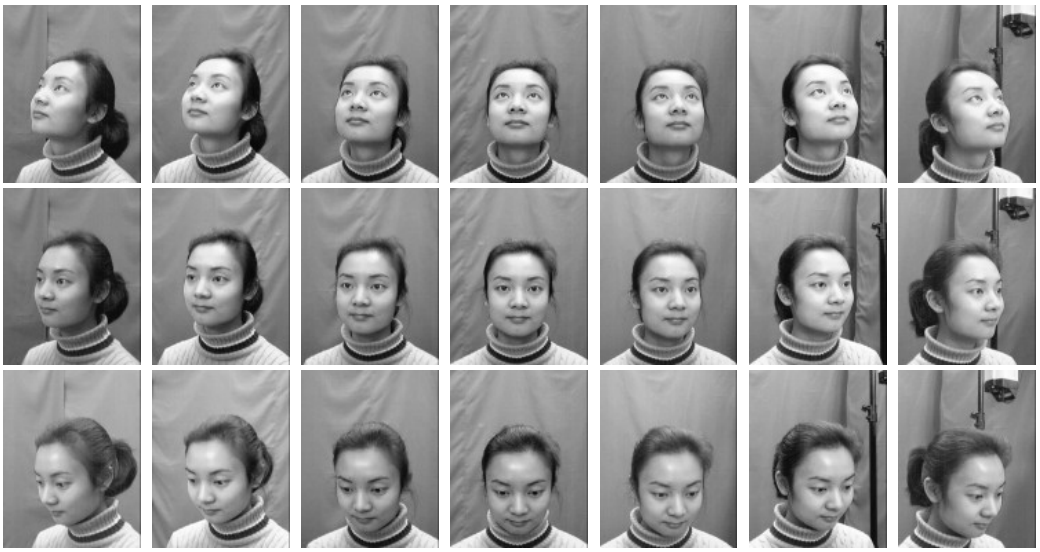
2.1 Face Detection

The first task for a facial understanding system is to detect faces. In [12], Yang presented a survey of traditional face detection methods and in [13] Lewis investigated several theories as to how the human eye-brain detects faces. The Viola-Jones approach [14], with enhancements by Lienhart [15], is a popular face detection method and is commonly used because of its low computational requirements and high detection rates. The Viola-Jones method utilizes simple rectangular difference pairs, similar to Haar basis functions. Each selected rectangular wavelet pair is a weak classifier in that it is only slightly better than 50% at distinguishing faces from non-faces. The AdaBoost [16] learning algorithm selects the most effective weak classifiers. When these weak classifiers are combined, they form a strong classifier. By converting the original image to an integral image, these weak features are computed quickly; and when these features are organized into a cascade, obvious non-facial regions are rejected quickly, while harder to classify regions are subject to further feature scrutiny as necessary. A face detection window of variable size is slid over all possible locations, evaluating the cascade approximately 225,000 times on a typical video-based frame.

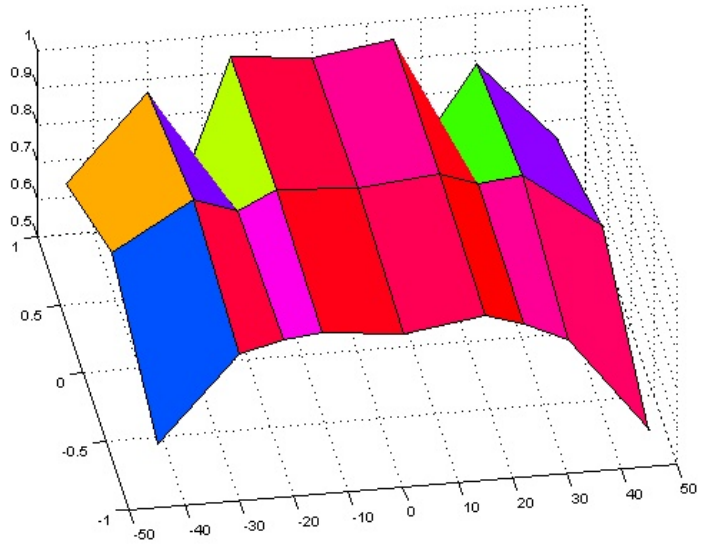
The Viola-Jones method is trained for locating faces of limited pose variation. As such, multiple passes over the image are required to find all faces, with each pass searching for a targeted range of pose. Empirical testing has shown that the default haarcascade_frontalface_alt2.xml classifier distributed with OpenCV [17] does a good job at detecting most faces over reasonable pose ranges. To demonstrate the effectiveness of the OpenCV implementation of face detection, the CAS-PEAL-R1 face database collected under the sponsor of the Chinese National Hi-Tech Program and ISVISION Tech. Co. Ltd. [18] was employed. This database contains multiple subjects photographed at three different levels of pitch and seven different yaw positions. An example subject from the CAS-PEAL-R1 face database is shown in Figure 2.1. (See Appendix I for a description of datasets used in this thesis.)

Figure 2.1(b) shows the performance of the default OpenCV haarcascade_frontalface_alt2.xml classifier when used on all 1042 subjects in the CAS-PEAL-R1 facial database. The X-axis is yaw angle.

The Y-axis is pitch with a +1 meaning the subject was looking upward (approximately +30^o pitch), a 0 meaning the subject was looking straight-ahead, and a -1 meaning the subject was looking downward (approximately -30^o pitch). This classifier exhibited detection rates above 90% at near frontal yaw angles of all pitches, but performance degrades quickly with respect to pose. The two dips in performance at upward pitch and yaw of +/- 22^o were due primarily to a limited number of samples at that particular pose position. Recent face detection methods incorporate pose and statistical models to improve the accuracy, efficiency, or robustness [19].



(a)



(b)

Fig. 2.1. (a) Sample subject from CAS-PEAL-R1 dataset. 21 pose positions are used to demonstrate face detection capability. Yaw angles correspond to [-45,-30,-15,0,+15,+30,+45]. Pitch angles correspond to [+30 ,0,-30]. (b) Performance of haarcascade_frontalface_alt2.xml classifier on the CAS-PEAL-R1 dataset. Over 21,000 faces at

various yaw angle (X-axis) and pitch (Y-axis). The pitch on the Y-axis is defined as "1" looking up, '0' looking straight, '-1' looking down.

2.2 Facial Feature Point Localization

After face detection, facial feature localization plays a prominent role in facial understanding. Facial feature detection algorithms include template matching methods [20], edge-based approaches, holistic methods [21], and shape models [22]. Shape models have the advantage that the shape of the face can be constrained to precisely locate key features of the face in the presence of occlusions. Statistical models capture shape variation, pose variation, and non-rigid deformations and envelope them via a linear model. Active Shape Models (ASMs), initially introduced by Cootes [23], have been enhanced by Bolin [24], and Milborrow [25]. Active Appearance models, initially introduced by [26] have been enhanced by Matthews [27], and are available as an open source software library (VOSM http://www.visionopen.com/). Constrained Local Models (CLMs) [28] are similar to AAM models, but, they encode considerable pose and deformable face variations into a single model. Blanz [29] introduced the first 3D morphable models, and numerous 3D versions of each of the above shape models have since been proposed in the literature. Although the usage of 3D information often achieves slightly higher localization accuracy with only modest increase in compute power, 3D training data is still difficult to capture accurately for training and testing.

The ASM method localizes key facial landmarks, constraining each by plausible location learned from the training set. The AAM method furthers the ASM method by including location and appearance information while constraining each landmark, where the appearance information is defined by pixel intensity information defined by Delaunay triangulation between landmark points.

Eye centroid estimates can be found by using an eye detector or (more commonly) by using the average human eye locations based upon the face bounding box and anthropometry [30] of the human face. (See Appendix II for more information.) These eye centers are fed into the ASM/AAM algorithm, which in turn generates a vector of average feature positions for the eyes, eyebrows, nose, mouth, and face boundary. The location of each feature point is placed where the texture has salient features, such as edges and corners of key facial features. The number of ASM/AAM points may vary depending on the application. More points can be added to stress the importance of a particular feature, but as the number of points increases, so does the computational load. Figure 2.2 shows a sample facial image automatically annotated with 82 feature points (Bolin [24]) on the left and another sample face automatically annotated with 68 feature points (Milborrow [25]) on the right.

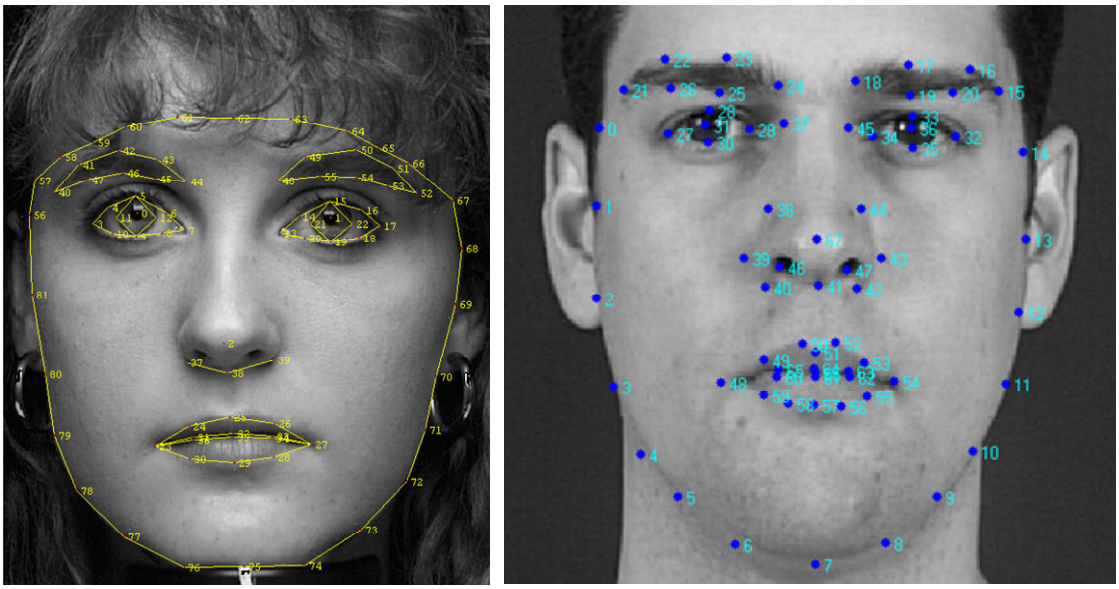


Fig. 2.2. Sample feature points produced by the ASM algorithm using Bolin [24] and Milborrow [25].

During training, the texture profile surrounding each feature point is learned from manually annotated ground truth subjects. The spatial relationship amongst feature locations is further used to develop a model which restricts the plausible region of each feature point. Figure 2.3 shows how the texture and statistical models are used by ASM during runtime. After face detection and eye localization, the eye centroid estimates along with the average face proportions are used to determine the approximate starting location for each feature point. The exact location of each feature point is determined by finding the point within the neighborhood search area that best matches the learned texture profile.

As shown in Figure 2.3, texture profiles are often defined as a 1D RGB gradient normal to the shape boundary. However, a second 1D gradient perpendicular to the first, and/or the surrounding neighborhood pixels of each feature point is also commonly used. The point with the smallest Mahalanobis distance with respect to the training data is selected as the search result, or feature point location. After all feature points are independently localized, the last step uses a holistic global shape model to constrain the overall shape. The global shape model transforms feature point locations to PCA space, where each dimension is constrained to be within +/- $3\sqrt{\lambda_i}$, where λ_i is the eigenvalue that corresponds to the i^{th} eigenvector. The last three steps are repeated and generally converge after three to five iterations.

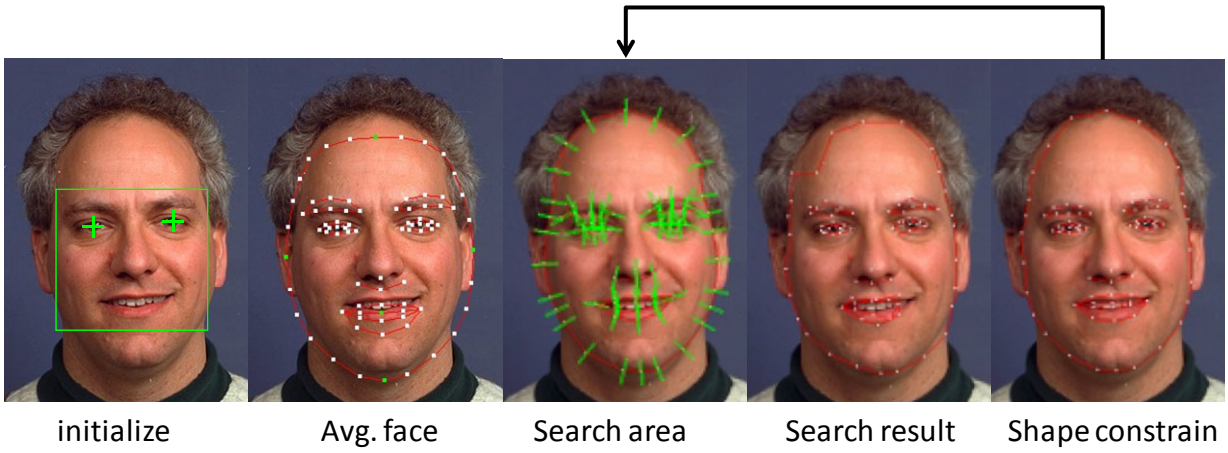


Fig. 2.3. Sample feature points produced by the ASM algorithm.

2.3 Facial Pose Estimation

Upon detection of the size and location of each face, there are many ways to perform facial pose estimation. A comprehensive survey of pose estimation techniques was documented by Muprhy-Chutorian [31]. This survey covered over 90 methods, loosely grouping them into eight categories: 1) Multiple face detectors, each tuned to a specific pose; 2) Direct comparison of filtered face to training exemplars; 3) Mapping of face features to pose classification or regression models; 4) ASM/AAM feature extraction to pose models; 5) Geometric models based upon eye/mouth/nose/etc landmarks; 6) Projection of facial features onto manifold surfaces; 7) Optical flow estimation from one video to the next; 8) Hybrid methods.

The pose estimation methods in this thesis rely heavily on dimensionality reduction. Performing dimensionality reduction on face pixels to estimate pose was demonstrated by [32-36]. Kanaujia [37] has shown that if we segment training samples by pose, and then utilize an intermediate step of first localizing facial features, we can build a family of principal components that enable a mixture of regression models.

Facial feature locations, such as corners of eyes, bottom of chin, etc., generally produce more accurate pose estimations than raw image pixels. ASMs were used for pose estimation in [38, 39] because they offer good feature localization and are robust over appearance variations and partial occlusions. Given accurate facial feature point locations, facial symmetry is very useful for pose estimation. For example, Figure 2.4 (left) shows the left eye length / right eye length, and Figure 2.4 (right) shows the left cheek area / right cheek area. Both are excellent predictors of yaw as they are generally invariant to pitch.

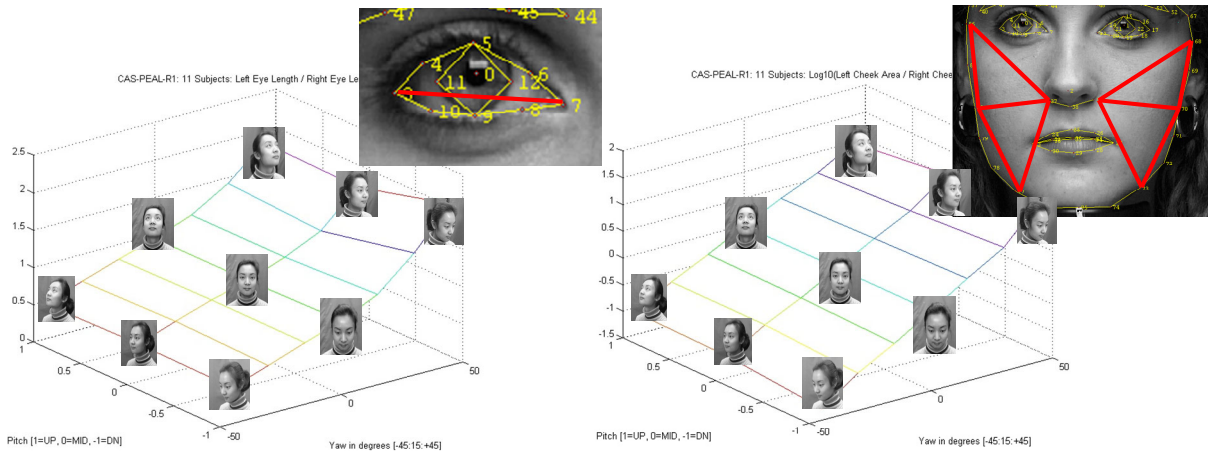


Fig. 2.4. (left) Using ASM eye points, calculate left eye length divided by right eye length for faces of various pose. (right) Using ASM points, calculate left cheek area divided by right cheek area for faces of various pose.

Recent pose detection approaches utilize 2D projection of 3D faces [40-42], 3D modeling of faces [43-45], and hybrid variations [46]. It remains a challenge to create techniques that are robust, accurate, and amenable to real time implementation. For example, most techniques fail in the presence of facial occlusions or harsh illumination, while others are computationally intensive.

2.4 Facial Expression Recognition

Facial expression recognition is the task of autonomously analyzing the human face to estimate a person's emotional state, mood or other form of facial communication. The ability to automatically extract facial semantic information has widespread implications on a number of industries including security, entertainment, and human computer interfaces. State-of-the-art techniques have become adept at recognizing posed expressions in laboratory conditions and have migrated to recognizing spontaneous expressions in uncontrolled settings. These new techniques have the potential to improve the quality of life by offering easier and more efficient interfaces to machines as well as enabling new and exciting connections between humans.

The importance of facial expressions was recognized in Charles Darwin's 1872 book "The Expression of the Emotions in Man and Animals". Darwin suggested that all mammals possess the innate ability to display and understand emotion through faces. While it is not completely appreciated why mammals exhibit facial expressions, there is no shortage of hypotheses. For example, without facial expression, a baby's ability to communication with their mother would be quite limited. If two humans cannot see each other's faces, it is hard to tell if someone is lying or telling the truth, happy or sad, needing affection or wanting to be left alone, or being friendly or hostile. Reasons range from basic

survival necessity to the ability to fall in love and sustain relationships. Regardless of how they evolved, facial expressions have weaved their way into every corner of human to human communication.

In the late 1960's, psychologist Paul Ekman, motivated by legendary professor Silvan Tomkins's uncanny ability to read people's faces, travelled the world and watched hundreds of thousands of facial film footage to understand human expressions. Ekman and his collaborator Wallace Friesen empirically proved all cultures exhibit the six universal expressions of fear, sadness, happiness, anger, disgust, and surprise. Ekman, Friesen, and their colleagues then created a taxonomy of facial expressions and documented forty-three facial movements constituting over ten thousand facial expressions. They discovered facial expressions consisted of both voluntary and involuntary muscle contractions, noted differences between genuine and posed expressions, and documented quick bursts of involuntary facial expressions called microexpressions. They further discovered that not only does mood or emotion involuntarily trigger facial muscle movements, but also performing an expression for extended periods evokes emotions related to that expression.

Computing power in the 1970's and early 80's was not powerful enough to tackle autonomous facial expression algorithms; but by the mid-1990's computing advances enabled significant improvements in face detection and facial tracking, which reinvigorated interest in facial understanding research. Affective computing, or computing that deliberately senses and influences emotion, was brought to the forefront by Rosalind Picard's book "Affective Computing" [4]. Gains in understanding facial expression and emotion have spawned a new era in human computer interfaces. Today, techniques are focusing on recognizing facial expression in unconstrained conditions that include variations of facial pose, facial occlusions, illumination, image fidelity, and background clutter.

2.4.1 Facial Expression Topology

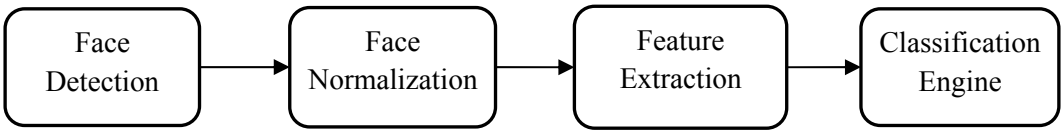


Fig. 2.5. Minimal steps necessary for facial expression recognition.

The study of the six culture agnostic emotions, i.e. fear, sadness, happiness, anger, disgust, and surprise, has made great strides in recent years from constrained frontal posed faces to unconstrained faces in natural conditions [47-49]. Figure 2.5 shows the minimal steps necessary for a facial expression recognition system. Face detection, as described in Section 2.1, is often accomplished with the Viola-Jones [14] approach because of its low computational requirements and high detection rates. Following face detection, faces are normalized to a reference shape and size. Typically, the eye and mouth corners

are localized, and an affine warp to canonical frontal face is defined. More complex methods utilize multiple canonical representations at various predefined pose representations [50].

Facial expression methods can be broadly categorized as geometric or appearance-based [51-53]. Geometric methods [54, 55] localize facial landmarks such as the outline of eyes, lips, nose, etc.. Appearance-based methods [56] work holistically with facial pixels enabling the capture of facial muscle subtleties such as nose wrinkles or dimple formation.

Geometric methods require computing size, shape, and location of key facial features such as the eyes, mouth, and eyebrows. Active Shape Model (ASM) or Active Appearance Model (AAM), as described in Section 2.2, are two of the most popular facial landmark localization methods [26]. Given enough training data and accurate facial landmark localization, shape models perform very well for expression classification [54, 55]. When the training set is not sufficiently rich, the trained ASM models may fail to capture the variance of individual expressions, which leads to reduced recognition performance. Furthermore, errors in point localization degrade expression classification accuracy further. Ptucha [57] demonstrated the effects of manually annotated vs. automatic ASM landmark placement on expression classification performance. Lucey [58] minimized this problem by applying neutral frame subtraction to the AAM points.

Appearance based methods often compute intermediate representations of images using features such as Gabor wavelets [59, 60] or Local Binary Patterns (LBP) [56, 61]. Gabor wavelets compute directional band pass filters based on the human visual system, but are slow and memory intensive. LBP capture various texture primitives and are quite tolerant to illumination changes. Shan [56] has shown that LBP slightly outperforms Gabor filters for expression recognition both in terms of speed and accuracy.

The classification engine for facial processing has been studied extensively [62]. With proper feature extraction, common methods such as k-Nearest Neighbor (k-NN), Support Vector Machines (SVMs), Logistic Regression, Adaboost, regression trees, and artificial neural networks yield acceptable results. Nonlinear methods or kernel-based methods generally offer small improvements. Sparse Representations (SRs) have proven to be effective at facial recognition, and recently have been adopted for facial expression classification [63].

Facial expression analysis can be classified as judgment-based or sign-based [52]. The former directly maps test subjects or attributes of test subjects to a predefined number of expression classes (happy, sad, angry, etc.). The latter first deciphers facial motion into action classes such as Facial Action Coding System (FACS) [11], whereby groupings of muscles in the face form Action Units (AUs), the

motions and combinations of which enable final classification. Rather than attempt to interpret the facial emotion, FACS captures all the possible atomic facial signals that can then be used as features into a reasoning engine. For example, human subjects exhibiting AU six (contraction of orbicularis oculi and pars orbitalis, or the cheek raiser muscles) in combination with AU twelve (pulling up of the zygomatic major, or the corners of the lips) are generally experiencing happiness. Interestingly, Ekman discovered that if someone is asked to act as if they are happy, they perform only AU twelve. He found it almost impossible for subjects to exercise the orbicularis oculi and pars orbitalis properly on command. Equally intriguing, it was just as difficult for humans to stop those muscles from contracting when they were genuinely happy. Similar to FACS, the Moving Pictures Experts Group (MPEG) developed the Facial Animation Parameters (FAP) specification as part of the MPEG-4 international standard. FAPs are focused on animation of facial expression, but are strongly correlated with AUs used in FACS.

While it is not fully understood how the human brain determines the emotions of other faces, temporal evidence has been shown to significantly aid the true comprehension of the emotional state of faces [2, 64, 65]. Recent works have shown that temporal dynamics can improve AU detection considerably [66].

2.4.2 Facial Expression Classification

After face detection, ASM and AAM geometric methods automatically localize key facial features such as the eyes, mouth, and eyebrow boundaries. A generalized Procrustes analysis [67] on facial feature points compensates for translation, scaling (head size), and rotation (head roll), effectively transforming the set of facial points to a normalized canonical representation suitable for classification.

Normalized facial feature points can be used to train a classification engine for facial expression recognition. In general, the selection of facial features and normalization of such features are more important than the choice of classifier [62].

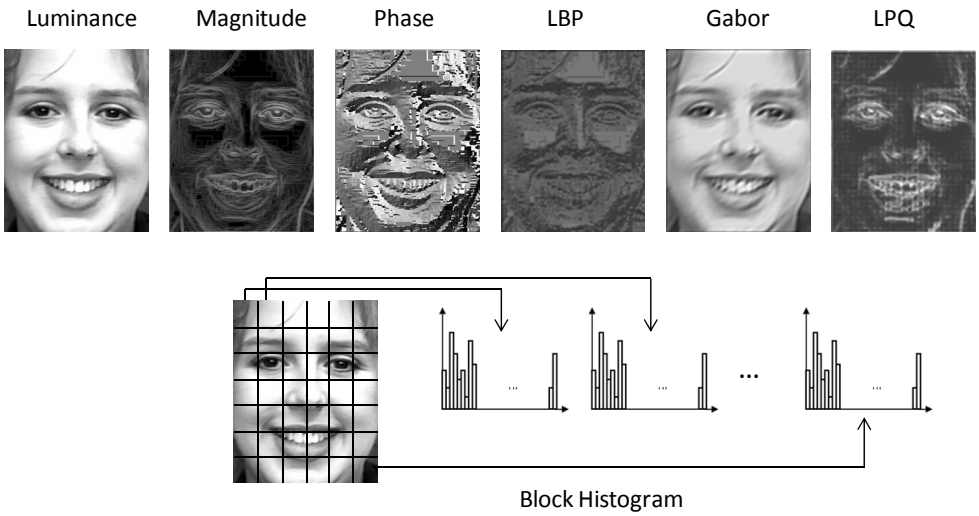


Fig. 2.6. Sample face exhibiting six pixel processing variants on top and block histograms on bottom. On the top, from left to right are luminance, edge magnitude, edge phase, LBP, Gabor, and LPQ.

Appearance based methods use eye and mouth corner points to define an affine warp transformation to a frontal canonical face representation. Edge gradients [68], Gabor wavelets [59, 60], Local Binary Patterns (LBP) [56, 61], and Local Phase Quantization (LPQ) [69] are common processing methods. Edge gradients typically use horizontal and vertical Sobel filters to extract edges of the face, resulting in the outline of eyes, nose, lips, etc. Gabor wavelets compute directional band pass filters inspired by properties of the human visual system. Typically, a multi-phase, multi-frequency bank of filters is used. Perhaps the most common configuration is eight equally spaced phases at five frequencies for a total of 40 Gabor representations. The Gabor representation in Figure 2.6 shows a mid-frequency filter at 45^{o}. LBP captures various texture primitives and is quite tolerant to illumination changes. The LBP of a given pixel is obtained by comparing it with all pixels in a window centered at that pixel; if a window pixel is greater than the center pixel, it is encoded as a '1'. The concatenation of these binary comparisons forms a single binary number whose decimal equivalent is used as the LBP feature. LPQ has been shown to be quite tolerant to changes in image sharpness and illumination. LPQ computes the phases of low frequency coefficients whose histogram is used to derive the final feature representation. (Appendix II has a summary of pixel processing techniques used in this thesis.)

Another common feature used in appearance based facial expression models is block histograms. Block histograms divide the face into $n_r \times n_c$ (generally non-overlapping) regions, where n_r and n_c represent the number of blocks down and across the face respectively. The histogram for each region is tabulated, and the histograms for all regions are concatenated to generate one large feature descriptor.

A facial region extraction and dimensionality reduction step may be added before feature extraction to make the model more general. The facial region extraction (see Section 6.2) allows the model to mask portions of the face. Processing individual facial regions is motivated by the need to

improve classification accuracy in the presence of occlusions and allows independent processing on each region of the face. When facial regions are used, block histograms are often unnecessary and each masked region is independently trained to obtain a localized model for every face region.

Regardless of whether geometric or appearance-based methods are considered, the features used in facial understanding often reside in representations of high dimensionality. These high dimensional feature spaces are inefficient and computationally intensive. Further, the artificially high dimensionality often masks the discriminative signal embedded in the data. As such, dimensionality reduction techniques are often utilized.

2.4.3 Dimensionality Reduction and Sparse Representations

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two effective techniques for obtaining a lower dimensional representation of the input data. PCA is used for unsupervised datasets and is optimal in the sense of reconstruction error. LDA is used for supervised datasets and is optimal in the sense of classification error. Both methods assume a linear mixture of Gaussian distributions. This may be limiting when modeling the behavior of complex imagery such as face representations.

Manifold learning techniques reduce the dimensionality of input data by identifying a non-linear lower dimensional space where the data resides [70, 71]. Popular methods include Isomap [72] and Locally Linear Embedding (LLE) [73]. In order to support the extension of the manifold model to new examples, linearized techniques such as Linear extension of Graph Embedding (LGE) [74], solve a linear approximation of the non-linear object. The dimensionality reduction offered by the LGE techniques generally affords greater dimensionality reduction than linear methods such as PCA or LDA.

The family of LGE techniques yields a multitude of dimensionality reduction techniques such as Locality Preserving Projections (LPP) [75] and Neighborhood Preserving Embedding (NPE) [76]. State-of-the-art techniques use a blend of supervised and non-supervised LGE techniques. The supervised techniques excel at class discrimination, and the non-supervised techniques more closely mimic the complex topology of spontaneous faces in uncontrolled conditions.

Zafeiriou [77] used PCA and Sparse Representation (SR) techniques based on Wright [78] for facial expression recognition. Zafeiriou noted that applying the SRs framework in facial expression applications is not a straightforward process because the facial identity of the person is often confused with the facial expression. As a result, the sparse representation will likely contain coefficients corresponding to both similar expression and similar identity. To deal with this problem, Zafeiriou proposed to subtract the neutral expressive image from the test image before processing, emphasizing facial movement and deemphasizing facial identity. The methods introduced in Section 6 overcome this

coefficient contamination issue without the need for neutral frame subtraction [79].

The communication between two humans often carries significant observable information that is best captured in a temporal fashion. Further, AU characterization in the FACS system assumes a neutral reference frame and includes levels of severity, making it ideally suited for temporal analysis. Both sparse and dense optical flow techniques across the human face can be incorporated into expression classifiers. Many of the same temporal techniques used for facial expression analysis pertain to gesture and action recognition. As such, the next section will cover temporal methods described in context of facial expression analysis, but all of which are suitable for gesture and activity analysis.

2.5 Temporal Processing

The communication between humans naturally contains a temporal signature. For example, the rolling of the eyes, or raising of an eyebrow carries significant observable information that is best represented in a temporal fashion. Psychological studies have confirmed that temporal evidence is necessary towards the full comprehension of the emotional state of the face of interest [2, 64]. However, the usage of facial dynamics in the facial expression community is quite limited. As evidence of such, in the 2011 Facial Expression Recognition and Analysis Challenge (FERA2011), only four (out of fifteen) entrants utilized facial dynamics. Popular methods are extensions to static methods such as LBP-TOP [56] and LPQ-TOP [69].

The usage and interpretation of facial dynamics [64] by humans is an active area of research. Facial expressions typically contain an onset, apex, offset, and neutral stage. The timing and duration of each stage are critical to the interpretation of the observed behavior. These temporal dynamics have been used to discern between genuine and acted pain [80], telling the truth vs. lying [81], detection of depression [82], synthesizing facial expressions for avatars [83], and much more.

Facial dynamic methods include temporal tracking of geometric landmarks as well as tracking changes in facial appearance. Salient landmarks, such as corners of the eyes and mouth are generally well behaved and track well using optical flow techniques. Less well defined features, e.g. nose wrinkle, can be tracked with dense optical flow techniques such as Motion History Images (MHI) [84], Free Form Deformations (FFD) [85], or SIFT flow [86]. MHI was initially introduced for human movement recognition, and was later adopted for facial AU detection [66]. FFD was initially introduced for medical image registration and later adopted for facial AU detection [66]. SIFT flow was introduced for generic image registration [86], and adopted for face alignment [87].

Facial expressions or gestures can occur at any point in time and are variable in length. Thus, we define m sliding temporal windows W^{θ}_{l} of duration θ, where θ is the number of frames in an expression

sequence, and $l=1..m$. Each of these temporal windows can be used as inputs to facial dynamic classifiers. Each temporal sliding window W^{θ}_{l} produces one of several estimates of facial expression per video segment. These estimates are stored in a vector and converted into a single expression estimate via voting. Equation (2.1) combines static and temporal features into a single expression estimate:

$$\Psi = mode\left(hist(F_s) \cdot \left({hist(F_t)}/{\sum F_t} \right) \right) \tag{2.1}$$

Where F_s and F_t are the static and temporal vectors of predicted votes; *hist()* is a histogram operation; and *mode()* computes the most frequently occurring prediction. Equation (2.1) weighs the votes of the static model by the confidence values from the temporal model.

2.5.1 Facial Feature Point Tracking

When accurately placed, facial feature locations such as corners of eyebrows, outline of mouth, etc., can produce accurate expression estimations. Active Shape Models (ASMs) and Active Appearance Models (AAMs), initially introduced by [23] were used for expression estimation in [55, 88]. ASM is applied independently on each frame of the temporal window W^{θ}_{l}. For example, using the Bolin [24] ASM, we get 82 facial feature points per frame. Each set of 82 points is transformed to a canonical face representation using a generalized Procrustes analysis [67]. With θ frames per W^{θ}_{l}, we get θ*82*2=θ*164 dimensions per each of the m samples.

Each sample is dimensionality reduced and classified using standard machine learning methods. The dimensionality reduction step not only reduces the upstream compute complexity, but also makes the sample data more discriminative, or more suitable for subsequent classification. The machine learning methods include techniques such as k-NN, regression trees, Support Vector Machines (SVM), neural nets, etc. Alternatively, we can pass the average of the 82 points per W^{θ}_{l} into dimensionality reduction, or convert these points to [Δx,Δy] or [magnitude,phase] motion vectors before dimensionality reduction.

2.5.2 Motion History Images

Motion History Images (MHI) were initially introduced for human movement recognition [84], and were later adopted for facial AU detection [66]. MHI compresses the motion over each sliding temporal window W^{θ}_{l} into a single template. The methods described here are similar to [66], except the conversion from MHI image to motion vectors is modified for improved facial expression performance. MHI initially evaluates the movement between all possible frames f and $f+1$ in W^{θ}_{l}, where $f=1..\theta-1$. For each pair of frames $\{f, f+1\}$ in W^{θ}_{l}, we first calculate motion detection at the pixel level in a binary fashion:

$$d_f(x,y) = \begin{cases} 1 & |g(x,y,f) - g(x,y,f+1) > \gamma| \\ 0 & otherwise \end{cases} \quad (2.2)$$

where $g(x,y,f)$ is a Gaussian filtered version of frame f and γ is a noise threshold. These difference frames are morphologically filtered with an opening operation to remove isolated noise. Each of the m sliding windows produces a single frame called a MHI^{θ}_{l} template:

$$MHI^{\theta}_{l} = \frac{1}{\theta - 1} \max_{f} \{ f\, d_f(x,y) \quad 0 \le f \le \theta - 1 \} \quad (2.3)$$

In this context, more recent movements are assigned higher weights. The MHI^{θ}_{l} templates are converted to motion vectors by replacing each pixel code value with a motion vector that points in the direction of the highest (most recent motion) code value within a 7x7 neighborhood. This neighborhood is constrained such that we may only point in the direction where pixels are monotonically increasing from the center outwards. Further, if there are several pixels of the same value, the average value is used. ΔX, ΔY, magnitude, and phase-magnitude versions of this motion vector image are passed into dimensionality reduction.

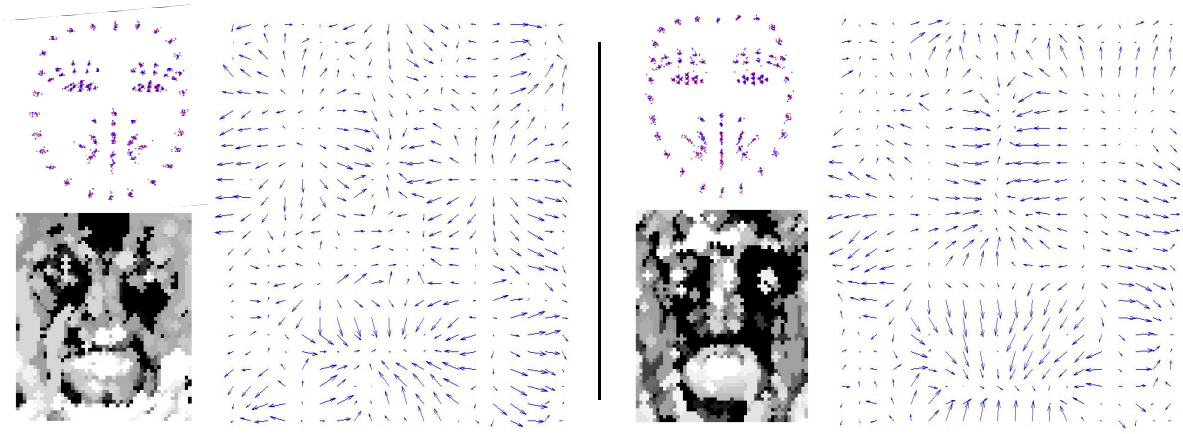


Fig. 2.7. Sample anger (left) and joyful (right) temporal windows, θ=20. For each group: (upper left) movement of ASM facial feature points from first (red) to last (blue) frame; (lower left) corresponding MHI template; (right) final motion vectors from the MHI template.

2.5.3 Free Form Deformations

Free Form Deformations (FFD) are a dense optical flow technique initially introduced by [85] for medical image registration and later adopted for facial AU detection [66]. Given two images, FFD computes a rigid global motion and a non-rigid local motion model representing the movement of each pixel from one frame to the next. The global motion model iteratively solves for a 3x3 affine transformation matrix using convex optimization techniques across all pixels in both images. The local motion model solves for the displacement of a mesh of grid points. Given input pixel $(x,y)_t$ in frame t, we solve for the estimate of its position $(x',y')_{t+1}$ in frame t+1:

$$(x',y')_{t+1} = (x,y)_t + F_t(x,y) \tag{2.4}$$

$F_t(x,y)$ is solved for each pair of neighboring frames $\{f, f+1\}$ in the temporal window W^{θ}_{l}, and pass the set of all $F_t(x,y)$ per W^{θ}_{l} as the input to dimensionality reduction. To solve for each $F_t(x,y)$, FFD solves a gradient descent optimization across a sparse mesh of control points, minimizing sum of square difference of pixels values. Given the deformed sparse mesh of control points, any interpolation method can solve for the dense motion $F_t(x,y)$ at each input pixel location. FFD uses cubic B-splines as the resulting fit is smooth and continuous across mesh vertices. To avoid local minima and make this process more computationally tractable, a hierarchical approach, solving from a low to high resolution mesh is utilized. Each mesh point position is initialized by the lower resolution mesh preceding it. Gaussian blurring with a σ twice the grid spacing at each level in the pyramid ensures robust behavior.

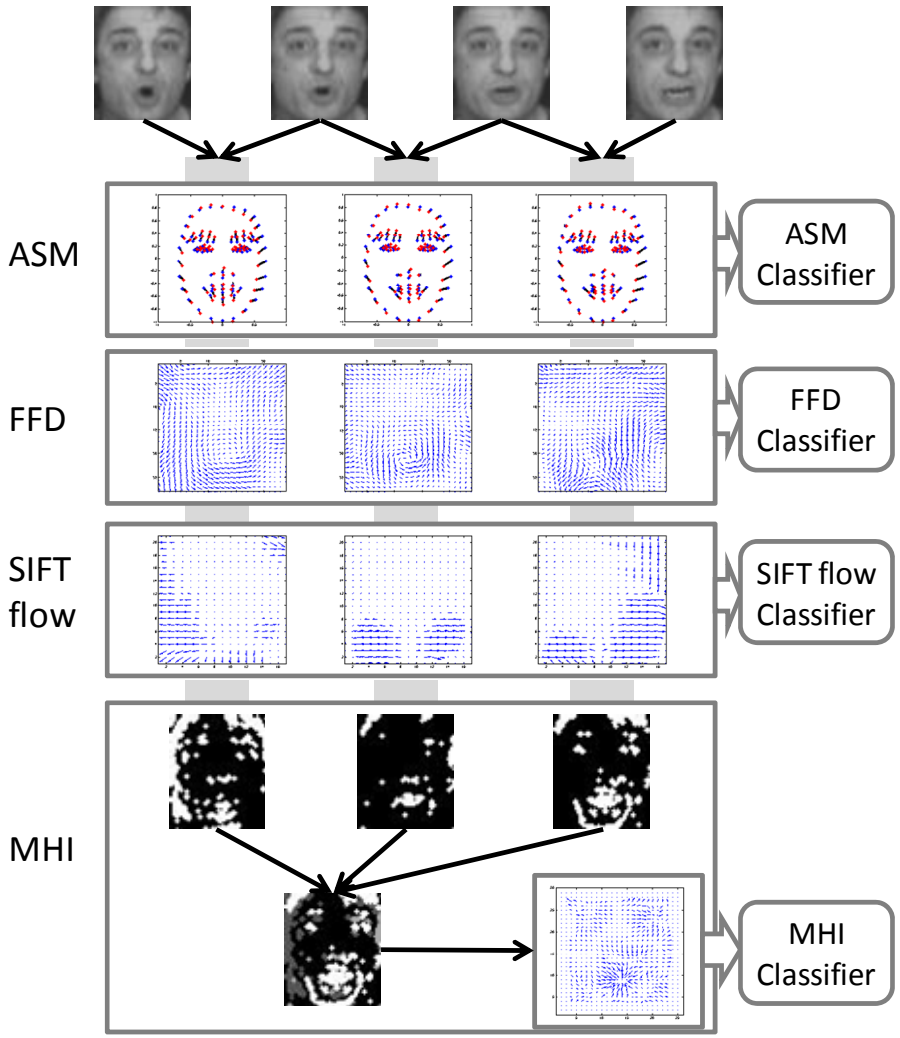


Fig. 2.8. Sample temporal window W^{θ}_{l} , θ=4 frames, from an 'angry' video. From top to bottom we have input 60×51 cropped and affine warped faces, 82 point ASM motion vectors (black arrows have blue tail, red tip), grid of 28×24 dense FFD optical flow vectors, grid of 21x17 dense SIFT flow vectors, and MHI fields. For MHI three 60×51 difference frames form one 60×51 MHI template and one 30×26 dense MHI flow field.

2.5.4 SIFT Flow

SIFT flow [86] is an image alignment algorithm initially introduced to register two similar images and further adopted for facial registration by [86] [87]. Similar to FFD, SIFT flow produces a dense optical flow field between all neighboring frames $\{f, f+1\}$ in the temporal window W^{θ}_{l}. SIFT descriptors are densely computed on every input pixel. The objective function of SIFT flow is similar to optical flow, but minimizes SIFT descriptors rather than RGB values:

$$E(w) = \sum_{p} \left| s1(p) - s2(p+w) \right|_1 + \frac{1}{\sigma^2} \sum_{p} \left(u^2(p) + v^2(p) \right) + \sum_{(p,q)\in\varepsilon} \min(\alpha \left| u(p) - u(q) \right|_1, d) + \min(\alpha \left| v(p) - v(q) \right|_1, d) \tag{2.5}$$

Where p represents all pixels in the image, s_1 and s_2 are the SIFT image for two neighboring frames, w is the flow field in the u and v directions, ε is the local neighborhood of the pixel p with neighbor q. The first regularization term favors small displacements and the second discourages discontinuities in the local field. An iterative belief propagation algorithm is used to solve for w. Silimarly to FFD, a multi-grid hierarchy is used both for speed and robust point matching. For the experiments in this thesis, σ=300, α=0.5, and d=2.

2.6 Depth Cameras

With the introduction of low cost depth cameras such as Microsoft Kinect [89], depth estimation has for the first time become an affordable option for digital interfaces. Depth information provides much more salient information than RGB or grayscale cameras for subject gesture recognition. The extraction of objects against backgrounds, and the tracking of objects have been simplified from a highly compute intensive and error prone task to one that is much more robust and works with much simpler methods [6]. To the average gamer, Kinect may be just another cool interfacing device, but for the computer vision world, Kinect has spurred a revolutionary leap in human-machine interaction.

Extraction of rigid objects against cluttered backgrounds is a difficult task. Allowing for unlimited pose variation complicates matters further. Allowing the object to be deformable makes this task formidable by most RGB camera systems. However, this same task of extracting deformable objects at any pose is straightforward with depth cameras.

Fig. 2.9. The Kinect sensor and standard drivers enable silhouette extraction of multiple users.

For example, Figure 2.9 shows three subjects interacting with a RGB camera on the left and depth camera on the right. Only moving objects are considered for extraction. The depth of the moving blobs against the static background is easy to extract using the depth channel and conventional computer vision algorithms such as temporal thresholding. At this point, any moving object is a candidate for extraction- a cat, dog, moving car, etc., would all be extracted as a contiguous object. Further, humans carrying tools or other objects, or wearing loose clothing or objects such as backpacks would expand the range of the deformable object. For the system to work properly in this thesis, the only moving objects should be humans, and all humans should avoid wearing loose clothing or holding anything that could easily trick the silhouette extraction software.

Given the silhouette of a human, each section of the silhouette blob needs to be assigned to a body part, where kinematic and temporal constraints ensure plausible limb identification. A skeletal model is then fit to localize the ankles, knees, hips, shoulders, elbows, wrist, head and torso. The model employed by Kinect is documented in [6].

The first step is to assign each pixel in the silhouette map to one of thirty-one predefined body parts. To increase saliency, the silhouette map is converted to a depth delta map, where the difference in depth between each pixel and its neighbors is used as a classification feature. The classification engine is a training forest of decision trees, each trained with over one-million manually labeled ground truth samples. Each decision tree is pruned to a depth of twenty. After each pixel in the silhouette is classified independently by the decision forest, neighborhood filtering classifies each pixel as belonging to one of the thirty-one body parts. The joint position can't be determined by 3D centers of probability mass because of sensor noise and shadowing caused by triangulation of the IR emitter and detector used in the depth camera. Instead, a local mode-finding approach based on mean shift with a weighted Gaussian kernel is used.

One interesting fact about Kinect method [6] is that its functionality does not utilize any temporal tracking. During a 2011 IEEE Computer Vision and Pattern Recognition Conference oral presentation,

Shotton claimed that even if all joints were tracked with 99% accuracy, after one minute the skeleton would be grossly misrepresented 50% of the time. As such, an independent joint assignment is made on each video frame, and then kinetic and temporal smoothing constraints are imposed.

2.7 Gesture Recognition

Nonverbal gestures can be used in place of voice commands both because of the natural tendencies of humans, as well as unpredictable sound in crowded or busy environments. Gestures include movement of the arms, hands, and face. Although many gestures are culture agnostic, researchers will no doubt develop new variants of gestures which users find natural and intuitive to perform. These gestures are likely to change by application, environment, and culture. For example, gestures in games may include simulation of shifting a car and turning a steering wheel, while gestures in an airport may only include simple point and select of airline departure times.

By extracting head pose, calculating depth, then using Euler angle geometry [90], we can project a human's visual focus of attention onto the screen. Similarly, by using joint locations and projecting a line fit through the shoulder and arm joints onto the screen, we can estimate where the human is pointing. Empirical studies have shown that head pose is more appropriate for displays of close proximity, while pointing is more intuitive than using the head pose for larger displays, or displays that are further away. Because no two humans are the same size/shape, and because systems are subject to calibration/perspective errors, a fiducial should be placed on the screen, indicating where the actor is looking/pointing.

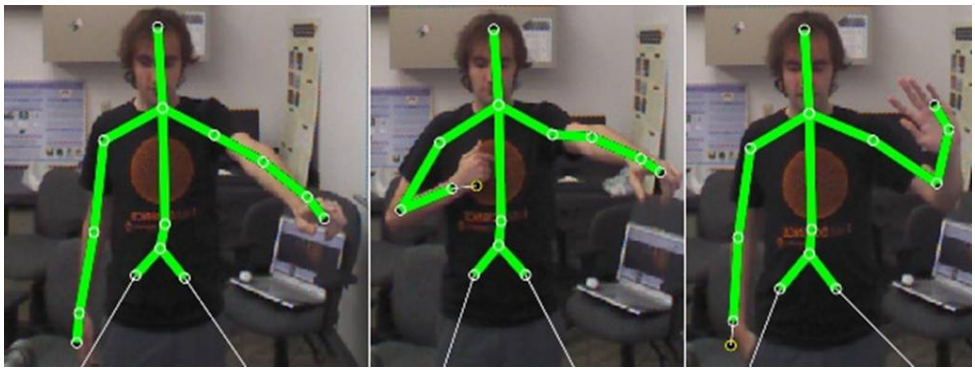


Fig. 2.10. Skeleton overlay on top of RGB image.

Recognizing gestures other than simple pointing or looking is often accomplished by feeding tracked skeletal joints into Hidden Markov Models (HMM) to accurately model sequences of complex gestures [91, 92]. The recognized gesture can then be used to direct interactive large-scale displays alongside facial pose or expression information for a more robust and satisfying user experience [93, 94].

The most intuitive gestures mimic familiar physical interactions, which makes high accuracy and low latency paramount to meeting user's expectations for usability and feedback [95]. Using only tightly constrained body gestures has been shown to reduce the usability of highly interactive applications with many degrees of freedom [96], such as video games. Heydekorn et al. [97] have shown that more constrained application specific gestures are less intuitive than loosely constrained gestures for users. To accomplish these goals, new methods that are capable of both accurate classification and real time implementation are necessary.

Action classification requires descriptors that are both discriminative and computationally efficient. Spatial action representations, such as body models [98], body pose estimations [99], kinematic joint models [100], and stick figures [101] offer intuitive representations, but may not adequately capture the human body's high degree of variability. Spatial parametric image features such as contour/silhouette representations [102] and optical flow [103] don't require body part labeling or tracking, but are more computationally intensive.

Temporal modeling methods for gesture control include temporal tracking of skeletal joints as well as dense tracking of RGB or depth pixels. Dense optical flow techniques include Motion History Images (MHI) [84], Free Form Deformations (FFD) [85], and SIFT flow [86] as reviewed in Section 2.5.

This thesis introduces active difference signatures to select active temporal regions of interest based on both the depth map from a 3D camera along with estimated kinematic joint positions [6]. The skeletal joints are normalized using a reference representation of both the depth image and the joint locations. The difference between the normalized joints and a canonical representation of skeletal joints forms an active difference signature, a salient feature descriptor across the video sequence. This descriptor is dynamically time warped to a fixed temporal duration in preparation for classification.

2.8 Notation

Variables, vectors, and matrices are denoted with italics. Matrices are bold uppercase, and vectors are bold lowercase. Components of vectors are denoted with subscripts such as x_n, for the n^{th} offset of vector x. Matrices are denoted with subscripts such as W_{ij}, to represent the i^{th} row and j^{th} column entry of matrix $\boldsymbol{W}$. Images are treated identically as matrices. Matrices and images are stored in lexicographic form before being input into classifiers. The dimensionality of a vector is denoted as $\boldsymbol{x} \in \mathbf{R}^D$, meaning that vector $\boldsymbol{x}$ has D dimensions or D entries. The dimensionality of a matrix or image area is denoted as $\boldsymbol{X} \in \mathbf{R}^{D \times n}$, meaning that matrix $\boldsymbol{X}$ has n entries, each of D dimensions. The ℓ^0 norm of a vector $\boldsymbol{x} \in \mathbf{R}^D$ is a sparsity measure which counts the number of non-zero vector elements as $\|\boldsymbol{x}\|_0 = \sum_{i=1}^{D} |x[i]|^p$. The ℓ^p norm of a vector $\boldsymbol{x} \in \mathbf{R}^D$, $1 \leq p < \infty$, is defined as $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{D} |x[i]|^p\right)^{1/p}$. The Frobenius norm of a

matrix $\boldsymbol{X} \in \mathbf{R}^{D \times n}$ is defined as $\|\boldsymbol{X}\|_F = \left(\sum_{i=1}^{D} \sum_{j=1}^{n} X[i,j]^2\right)^{1/2}$. Common variables used throughout this thesis include:

n: The number of (training or testing) samples.

k: The number of discrete classes for a dataset.

D: The (high) dimensional of a feature before dimensionality reduction.

d: The (low) dimension of a feature after dimensionality reduction.

$\boldsymbol{U}$: A $D \times d$ dimensionality reduction matrix.

$\boldsymbol{W}$: A $n \times n$ adjacency matrix that represents neighbor to neighbor connections from each of the *n* input samples to all other *n*-1 samples in a dataset.

$\boldsymbol{\Phi}$: A sparse representation dictionary.

$\boldsymbol{a}$: The sparse coefficients, or linear amounts used from each element in the dictionary

m: The number of entries in a sparse representation dictionary Φ.

λ: The regularization parameter used in sparse representation, $0 \leq \lambda \leq 1$. Higher values of λ induce more sparsity.

α: The semi-supervised parameter used in dimensionality reduction, $0 \leq \alpha \leq 1$. Lower values of α use more unsupervised dimensionality reduction. Higher values of α use more supervised dimensionality reduction.

3 Dimensionality Reduction

Dimensionality reduction maps data of high dimension to a lower dimension, and often discards uninformative variance. This process makes the data more compute friendly, removes noise, emphasizes discriminative properties, and enables improved inference for later regression or classification models. Complex objects such as facial images often necessitate representations of high dimensionality. For example, Lucey [58] represents faces using 68 landmark points and 87×93 pixels. As a result, each face resides in $\mathbf{R}^D$ where D=(68×2+87×93)=8,227. This high dimensional feature space is not only inefficient and computationally intensive, but the sheer number of dimensions often masks the discriminative signal embedded in the data.

Formally, the input feature space contains n samples, $x_1, x_2, \ldots x_n$, each sample of dimension D, $x_i \in \mathbf{R}^D$. These n samples are projected onto a lower dimensional representation, yielding $y_1, y_2, \ldots y_n$, each output sample of dimension d, $y_i \in \mathbf{R}^d$. As d is always $\leq D$, we are interested in the case where $d << D$. In matrix notation, the input feature space is described by $n \times D$ matrix $\boldsymbol{X}$, where the i^{th} row of $\boldsymbol{X}$ corresponds to x_i. In the linear projection from D to d dimensions, we have $\boldsymbol{Y}^{\mathrm{T}}=\boldsymbol{X}^{\mathrm{T}}\boldsymbol{U}$, $\boldsymbol{X}\in\mathbf{R}^{D\times n}$, $\boldsymbol{Y}\in\mathbf{R}^{d\times n}$, and $\boldsymbol{U}$ is the $D \times d$ projection matrix.

3.1 PCA

Principal Component Analysis (PCA) is the most often used dimensionality reduction method and often serves as a starting point for many dimensionality reduction methods. PCA is an unsupervised linear dimensionality reduction method that projects data along orthogonal directions of maximum variance. PCA is optimal in the sense of reconstruction error, but PCA basis functions are not usually optimal for feature extraction or discriminative classification.

PCA solves for one dimension at a time according to $y_i = U_i^T x_i$, where each U_i maximizes:

$$\max_u \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{3.1}$$

and

$$\bar{y} = \frac{1}{n} \sum y_i \tag{3.2}$$

This is typically solved via eigenvectors of the $D \times D$ covariance matrix, where the covariance matrix measures the relative spread from the means between any two dimensions, $\boldsymbol{a}$ and $\boldsymbol{b}$. Each entry of the covariance matrix is solved as:

$$cov(a,b) = \frac{\sum_{i=1}^{n}(a_i - \overline{\boldsymbol{a}})(b_i - \overline{\boldsymbol{b}})}{(n-1)} \tag{3.3}$$

As such, $cov(\boldsymbol{a},\boldsymbol{a}) = var(\boldsymbol{a})$, and $cov(\boldsymbol{a},\boldsymbol{b}) = cov(\boldsymbol{b},\boldsymbol{a})$. When $cov(\boldsymbol{a},\boldsymbol{b}) > 0$, it indicates that as $\boldsymbol{a}$ increases, so does $\boldsymbol{b}$. For D-dimensional data, we need to calculate $D!/(2(D-2)!$ covariance values and D variance values. A good way to store these values is in a $D \times D$ matrix. For example, for 3-dimensional data, we have:

$$C = \begin{bmatrix} cov(\boldsymbol{a},\boldsymbol{a}) & cov(\boldsymbol{a},\boldsymbol{b}) & cov(\boldsymbol{a},\boldsymbol{c}) \\ cov(\boldsymbol{b},\boldsymbol{a}) & cov(\boldsymbol{b},\boldsymbol{b}) & cov(\boldsymbol{b},\boldsymbol{c}) \\ cov(\boldsymbol{c},\boldsymbol{a}) & cov(\boldsymbol{c},\boldsymbol{a}) & cov(\boldsymbol{c},\boldsymbol{c}) \end{bmatrix} \tag{3.4}$$

The diagonals are all the variances. Since $cov(\boldsymbol{a},\boldsymbol{b}) = cov(\boldsymbol{b},\boldsymbol{a})$, the matrix is symmetric. The eigenvectors of the covariance matrix $\boldsymbol{C}$ form the basis functions that convert to the alternate space (which in this case maximizes variance). The eigenvector defines the projection vector and the corresponding eigenvalue defines the variance of the data after it is projected onto the eigenvector. D dimensional data yields a $D \times D$ covariance matrix and D eigenvector/eigenvalue pairs. The eigenvector with the largest eigenvalue is called the principal component. Eigenvectors are generally sorted from most to least significant, according to the eigenvalues. As such, the eigenvector matrix is $D \times D$, consisting of D column eigenvectors, where each eigenvector is a $D \times 1$ projection vector. As such, $\boldsymbol{U}$ consists of D eigenvectors, each eigenvector being $D \times 1$.

If we format the n input samples, x, into n column-wise samples, each sample of dimension D, we obtain a $D \times n$ input matrix $\boldsymbol{X}$. Then $\boldsymbol{Y}^T = \boldsymbol{X}^T\boldsymbol{U}$, where $\boldsymbol{Y}$ is the PCA transformed space, $\boldsymbol{U}$ is the matrix of eigenvectors, and $\boldsymbol{X}$ is the input data. Each output sample (row of) $\boldsymbol{Y}$ is a linear combination of the D dimensions of each input sample (columns of $\boldsymbol{X}$), and so PCA is a linear transform. Given $\boldsymbol{Y}$, we can transform back to $\boldsymbol{X}$ using $\boldsymbol{X}^T = \boldsymbol{Y}^T\boldsymbol{U}^{-1}$. Since the inverse of an orthogonal matrix is its transpose, we have $\boldsymbol{X}^T = \boldsymbol{Y}^T\boldsymbol{U}^{\mathrm{T}}$, another useful property of PCA.

The eigenvectors of PCA are alternate basis functions that can be used to represent the input data. Often we only need to use d eigenvectors, and the remaining $(D-d)$ eigenvectors and corresponding eigenvalues are 0. In such cases, we reduce the basis or dimensionality of the data from D to d without information loss.

For usage in computer vision classification, verification, or recognition, it will be shown that an object can be reconstructed from a training dictionary of similar objects:

$$\hat{x} = \mu + \sum_{i=1}^{\emptyset} a_i \Phi_i \tag{3.5}$$

where μ is the mean object from the training dictionary, ϕ is the number of training samples in the dictionary, $\boldsymbol{a}$ is the coefficient weight, and $\boldsymbol{\Phi}$ is the training sample dictionary. This is analogous to the PCA transformation where we construct a linear transformation of D eigenvectors. The key difference is how we weight each eigenvector $\boldsymbol{\Phi}_t$ by a_i. The vector of coefficients $\boldsymbol{a}$, are a unique signature of $\boldsymbol{x}$ and can be used to identify an object. Solving for $\boldsymbol{a}$ is best done via an example.

Assume n=100 samples and each sample consists of 50x50 pixels. We can represent each sample as a one dimensional vector of D=2500 dimensions giving an $D \times n$ feature matrix $\boldsymbol{A}$. Computing the covariance of $\boldsymbol{A}$ results in a 2500×2500 covariance matrix $\boldsymbol{C}=\boldsymbol{A}^T\boldsymbol{A}$ which may cause the computer to run out of memory. Alternatively, we can compute matrix $\boldsymbol{L}=\boldsymbol{A}\boldsymbol{A}^T$, which is only 100×100. We can work with $\boldsymbol{L}$ instead of $\boldsymbol{C}$ because the eigenvectors of $\boldsymbol{L}$ are linear combinations of the eigenvectors of $\boldsymbol{C}$. An alternate representation of the eigenvectors of $\boldsymbol{C}$ is $\boldsymbol{U}_c=\boldsymbol{A}\text{eig}(\boldsymbol{L})$. In this fashion $\boldsymbol{U}_c$ is a 2500×100 transformation matrix instead of 2500×2500 (note: $\boldsymbol{A}$ is $n \times D$ and $\boldsymbol{L}$ is $n \times n$).

To compute the unique signature $\boldsymbol{a}$ for a new sample $\boldsymbol{x}$, we compute $\boldsymbol{a}=(\boldsymbol{x}-\mu)\boldsymbol{U}_c$ or $\boldsymbol{a}=(\boldsymbol{x}-\mu)\boldsymbol{U}$; where $(\boldsymbol{x}-\mu)$ is the mean subtracted test sample (1×2500 in this case), $\boldsymbol{U}$ is the eigenvectors of **C**, $\boldsymbol{U}_c$ is the alternate representation of the eigenvectors of $\boldsymbol{C}$, and $\boldsymbol{a}$ is the 1×100 weighting coefficients for test sample $\boldsymbol{x}$. In this fashion, we can work with $\boldsymbol{U}$ or $\boldsymbol{U}_c$, whichever is smaller. The vector $\boldsymbol{a}$ is a low dimensional representation of $\boldsymbol{x}$, or in this case, $\boldsymbol{a}$ represents the amount of each basis function (eigenvector) needed to reconstruct $\boldsymbol{x}$. If we consider the eigenvectors as dictionary elements, $\boldsymbol{a}$ represents the amount of each dictionary element necessary to reconstruct $\boldsymbol{x}$. The vector $\boldsymbol{a}$ is a unique and compact representation of $\boldsymbol{x}$, where the Euclidean distance between any two $\boldsymbol{a}$ values is referred to as eigenimages. When faces are used, it is referred to as eigenfaces [104]. Eigenfaces formed the foundation of state-of-the-art facial recognition models for many years. Today the best facial recognition systems use multiple features and statistical modeling, whereby some of those features are often eigenfaces.

3.2 LDA

LDA is a linear supervised technique that is optimal in the sense of discrimination of input classes. As done with PCA, LDA also seeks a linear combination of the input dimensions. Once again, the transformation matrix $\boldsymbol{U}$ transforms from one basis space to the next. While PCA optimized this basis set for efficient representation, LDA optimizes this basis set for efficient discrimination. Said differently, upon projection to LDA basis space, input classes are more easily separated into discrete classes. The method LDA uses is based upon the Fisher Linear Discriminant method that chooses to find a set of vectors, such that after projection into this new space, we maximize between-class means and minimize the intra-class variance. LDA solves the linear system above by maximizing:

$$\max_u \frac{u^T S_B u}{u^T S_W u} \tag{3.6}$$

Where:

$$S_B = \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{3.7}$$

and

$$S_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} (x_i - \mu_k)(x_i - \mu_k)^T \tag{3.8}$$

where μ is the overall mean of input samples, K is the number of classes, N_k, is the number of samples in class k, μ_k, is the mean of class k. $\boldsymbol{S_B}$ is called the between class scatter and $\boldsymbol{S_w}$ is called the within class scatter. Referring to the maximizing function, we need to find vectors $\boldsymbol{u}$, to maximize the objective function by setting the derivative to zero.

$$\frac{d}{du}\left[\frac{u^T S_B u}{u^T S_W u}\right] = \frac{\left(\frac{d}{du} u^T S_B u\right) u^T S_w u - \left(\frac{d}{du} u^T S_w u\right) u^T S_B u}{(u^T S_w u)^2} \tag{3.9}$$

Noting:

$$\frac{d}{du}(u^T S_B u) = 2 S_B u \tag{3.10}$$

Substituting (3.10) into (3.9):

$$= \frac{(2S_B u) u^T S_w u - (2S_w u) u^T S_B u}{(u^T S_w u)^2} \tag{3.11}$$

Setting the numerator to 0, and dividing each term by $2u^T S_W u$, we get:

$$\frac{(S_B u) u^T S_w u}{u^T S_w u} - \frac{(S_w u) u^T S_B u}{u^T S_w u} = S_B u - \frac{u^T S_B u}{u^T S_w u} S_W u = S_B u - \lambda S_W u = 0 \tag{3.12}$$

Thus, we get $\boldsymbol{S_B u} = \lambda \boldsymbol{S_W u}$, where λ is a constant. This gives $\lambda \boldsymbol{u} = \boldsymbol{S_W}^{-1} \boldsymbol{S_B u}$ where $\boldsymbol{u}$ is solved as a generalized eigenvector analysis of $\boldsymbol{S_W}^{-1} \boldsymbol{S_B}$. After projecting the input data onto this set of eigenvectors, $\boldsymbol{u}$, the data is processed for efficient discrimination using a classifier, SVM, k-NN, class centers, etc. As with PCA, the eigenvalues in the LDA analysis represent the variance of the projected data onto each eigenvector. Figure 3.1 shows a sample set of points projected onto the first principal component using PCA (on left) and LDA (on right). The PCA principal component is guided by variance while the LDA principal component is guided by class separability.

To enable dimensionality reduction using PCA or LDA, the top *d* eigenvectors that encode most of the data variance are used for the projection matrix *U*. To model the top *T*% variance of the training samples, d is solved such that $\Sigma\lambda_j/\Sigma\lambda_k >= T/100$; where $j=1,2,\ldots d$; $k=1,2,\ldots D$. Additionally, if we were using PCA to extract the signature *a* of a sample, we only need take the top *d* coefficients from the *a* vector for subsequent classification. A technique called whitening scales the elements of *a* by the square root of their corresponding eigenvalues such that more important eigenvector dimensions get weighted more than less important dimensions.

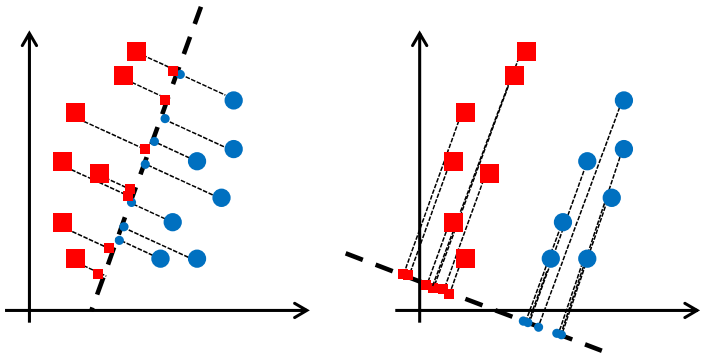


Fig. 3.1. A set of data from two classes (red square and blue circle) projected onto the first principal component as determined by PCA (on left) and LDA (on right).

3.3 Manifold Learning

The underlying linearity assumption of PCA and LDA may not be adequate for efficiently modeling the behavior of the high dimensional face representations. Alternatively, the high dimensional feature space can be parameterized by a lower dimensional embedded manifold discovered using manifold learning. In addition to being more compact, the resulting lower dimensional manifold representation is more discriminative and thus more appropriate for subsequent classification. Manifold learning [70, 71] is a dimensionality reduction technique that strives to reduce the dimension of input data by identifying a non-linear low dimensional space where the data resides. A manifold is a topological space in which every point has either a local or global neighborhood that is homeomorphic in $\mathbf{R}^d$, where *d* is the dimensions of the lower dimensional representation. The topological space is a mathematical structure defining properties of sets and subsets such that any union or intersection of subsets also belongs to the parent set. Homeomorphism between two topological spaces states that there is a continuous and invertible function connecting the two spaces such that all topological properties are preserved.

Manifolds constructed from facial understanding data are inherently non-linear alternate representations of the input space. The basic premise is to preserver neighbor relationships as we go from high to low dimensional. This can be done in a local or global fashion. In a local fashion, the input data is represented in a low dimensional space with many small linear patches spanning the dataset. In a

global context, a globally optimal solution is obtained using the shortest geodesic distances of all input samples.

LLE [73] is a nonlinear manifold learning method that preserves the local neighborhood topology (from input to manifold space) by finding weights that minimize reconstruction error in both $\mathbf{R}^D$ and $\mathbf{R}^d$. Locally linear patches are mapped to a global nonlinear structure. LLE computes the weights that best linearly reconstruct x_i from its k nearest neighbors by minimizing:

$$\left\| x_i - \sum_{j \epsilon N(i)} w_{ij} x_j \right\| \tag{3.13}$$

The resulting weight matrix $\boldsymbol{W}$ is solved in a least squares sense, and represents the local linear geometry of the patches. These locally linear patches are large in linear regions, and smaller in regions where the data is non-linear. This same local geometry is maintained in manifold space by applying basic constraints to $\boldsymbol{W}$, then computing a normalized covariance matrix of $\boldsymbol{W}$, whose eigenvectors complete the transformation to dimension d.

Isomap [72] is a nonlinear manifold learning method that preserves geodesic distances from input to output space. Geodesic distances are the shortest path distances (typically calculated using Dijkstra's shortest path algorithm) from point A to point B, traveling in straight lines through nearest neighbors along the manifold surface. Isomap computes all $n \times n$ pairwise geodesic distances in computing the global error, then uses a variant of multi-dimensional scaling to map from high to low dimensional representation. Isomap weighs short distances more than longer ones, then solves the eigenvectors of the geodesic distance matrix to convert the input features $\boldsymbol{X}$ to a lower dimensional manifold $\boldsymbol{Y}$.

3.3.1 Linear extension of Graph Embedding (LGE)

During manifold learning a fully connected graph of the input space is constructed, where each of the n input samples or nodes is connected to all other (n-1) input samples with a weight, $0 \leq w_{ij} \leq 1$, $i,j = 1 \ldots n$. The resulting connection matrix $\boldsymbol{W}$, $\boldsymbol{W} \in \mathbf{R}^{n \times n}$, is called the adjacency matrix and the connections or weights w_{ij} can be solved several ways. For example, w_{ij} is set to 1 if x_i is amongst the z nearest neighbors of x_j, 0 otherwise. Alternatively, w_{ij} is set to 1 if $||x_i - x_j|| < \varepsilon$, and 0 otherwise. Setting w_{ij} to continuous values between 0 and 1 offers more control in describing the connections.

Linear extension of Graph Embedding (LGE) is a general framework for graph based subspace learning. LGE techniques find a projection that respects the $\boldsymbol{W}$ graph structure, preserving the similarities amongst neighbors in both high and low dimensional spaces. LGE describes a family of such transformations, which are based on the geometric structure of the input space and perform a linear fit to a

non-linear phenomenon. Two popular methods include Locality Preserving Projections (LPP) [105] and Neighborhood Preserving Embedding (NPE) [76]. For example, LGE manifold learning using LPP is found by solving a linear approximation to the nonlinear Laplacian Eigenmap. As such, LPP shares many of the data topology properties of nonlinear techniques, but is linear and defined throughout space (not just on the trained data points as in LLE). For facial expression recognition, Xiao [106] and Section 4 of this thesis show that LPP performs better than PCA and LDA.

LGE creates an adjacency map of the top k neighbors for each feature point x_i, weighting each neighbor by distance to form $n \times n$ adjacency matrix $\boldsymbol{W}$ with entries w_{ij}. Close or similar neighbors have high values, far neighbors set $w_{ij}=0$. The primary goal of LGE is to preserve the neighborhood structure; as such LGE minimizes the function:

$$\sum_{i,j} \|y_i - y_j\|^2 w_{ij} \tag{3.14}$$

W is defined similarly for X and Y, such that if neighbors x_i and x_j are close, y_i and y_j are also close. LGE seeks a linear approximation to this nonlinear problem of the form $y_i=ax_i$.

$$\frac{1}{2}\sum_{i,j} \|y_i - y_j\|^2 w_{ij} = \frac{1}{2}\sum_{i,j} \|a^T x_i - a^T x_j\|^2 w_{ij} \tag{3.15}$$

$$= \sum_i a^T x_i D_{ii} x_i^T a - \sum_{ij} a^T x_i w_{ij} x_i^T a \tag{3.16}$$

$$= a^T X(D - W)X^T a \tag{3.17}$$

where $\boldsymbol{D}$, $\boldsymbol{D} \in \mathbf{R}^{n \times n}$, is a diagonal matrix of the column sums of $\boldsymbol{W}$, and $\boldsymbol{L}$, $\boldsymbol{L} \in \mathbf{R}^{n \times n}$, is the Laplacian matrix, $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{W}$. The Laplacian has its roots in graph theory and reveals all sorts of interesting connectivity aspects. The eigenvectors of the Laplacian are a linear approximation to this non-linear concept. $\boldsymbol{D}$ is a natural measure of the importance of the data points; the larger D_{ii}, the more important y_i is. As such, the constraint may be added:

$$y^T D y = 1 \implies a^T X D X^T a = 1 \tag{3.18}$$

Giving us a final objective function:

$$\min_a a^T X L X^T a \qquad s.t.\ a^T X D X^T a = 1 \tag{3.19}$$

The Lagrangian formulation of (3.19) is as follows:

$$\mathcal{L} = a^T X L X^T a - \lambda a^T X D X^T a \tag{3.20}$$

Setting the Lagrangian to zero and replacing $\boldsymbol{a}$ with the projection matrix $\boldsymbol{U}$, LGE computes eigenvectors of the generalized eigenvector problem:

$$\boldsymbol{XLX^T U} = \lambda \boldsymbol{XDX^T U} \tag{3.21}$$

The dimensionality reduction matrix $\boldsymbol{U} \in \mathbf{R}^{D \times d}$, and $\boldsymbol{XLX^T}$ and $\boldsymbol{XDX^T}$ are both $\in \mathbf{R}^{D \times D}$, were D is the input dimension and d is output reduced dimension. Typically, $n>D$, making LGE methods more computationally tractable than nonlinear manifold methods.

Unlike PCA, the resulting LGE transformation is not orthogonal. As such, the reduced dimensions contain some degree of redundancy and correlation. For example, several efforts to improve LPP include orthogonal LPP (OLPP) [107], uncorrelated LPP (ULPP) [108], and optimal (orthogonal and uncorrelated) LPP [109].

Different choices of $\boldsymbol{W}$ yield a multitude of dimensionality reduction techniques including LDA, LPP, and NPE. For each approach, $\boldsymbol{W}$ is initialized to all zeros, and then connected w_{ij} entries are determined by similarity. For LDA, we define a $\boldsymbol{W_{LDA}}$ kernel such that nodes i and j are connected if they are from the same class. $\boldsymbol{W_{LDA}}$ is initialized to all zeros, then connected w_{ij} entries are set to $1/k_n$, where k_n is the number of samples per their shared class:

$$w_{ij} = 1/k_n \tag{3.22}$$

For LPP, we define a $\boldsymbol{W_{Gaussian}}$ kernel such that if nodes i and j are connected, then:

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\tau}} \tag{3.23}$$

For NPE, we define a $\boldsymbol{W_{NPE}}$ kernel such that if nodes i and j are connected, we solve the following objective function for element i in local reconstruction matrix $\boldsymbol{M} \in \mathbf{R}^{nxn}$ as a function of z nearest neighbors of x_i, $N_z(x_i)$:

$$min \left\| x_i - \sum_{j \in N_{z(x_i)}} M_{ij} x_j \right\|^2, \quad \sum_{j \in N_{z(x_i)}} M_{ij} = 1 \tag{3.24}$$

Then

$$\boldsymbol{W_{NPE}} = \boldsymbol{M} + \boldsymbol{M^T} - \boldsymbol{M^T M} \tag{3.25}$$

Similarly to LDA, both LPP and NPE can be used in supervised mode by defining connected neighbors as the neighbors that share similar class labels. Supervision can be quite helpful, but it needs to be used cautiously. For example, although the LDA kernel can produce excellent results, the rank of the matrix is $k-1$, where k is the number of classes, and thus the maximum number of eigenvectors is $k-1$, restricting d

$\leq (k\text{-}1)$. When k is small, say for binary gender classification, such extreme dimensionality reduction is not practical.

To maintain input topology and increase the rank of $\boldsymbol{W}$, [110] proposed adding the LDA between-class and LDA within-class matrix to the LPP objective function (3.14). Recognizing that some problems are linearly separable, while others are not, this thesis proposes utilizing a convex combination of the LDA and Gaussian kernels:

$$\boldsymbol{W} = \alpha \boldsymbol{W}_{\boldsymbol{LDA}} + (1 - \alpha)\boldsymbol{W}_{\boldsymbol{Gaussian}} \tag{3.26}$$

where $0 \leq \alpha \leq 1$. $\boldsymbol{W}$, $\boldsymbol{W}_{\boldsymbol{LDA}}$, and $\boldsymbol{W}_{\boldsymbol{Gaussian}}$ are $n \times n$ matrices, where n is the number of input samples. For posed imagery, or datasets with exaggerated differences between classes, the choice of α is forgiving $\forall$ $\alpha > 0$. For natural imagery, any value of α (other than 0 or 1) produces improved results compared to the Gaussian or LDA kernel. Conceptually, adding a percentage of the Gaussian kernel to the LDA kernel, mimics the local non-linear input topology while simultaneously increases the matrix rank. For datasets that are difficult to separate by class, α should be decreased to learn the local topology. Additionally, for LDA kernels of low rank, the performance improvement can be dramatic. The greater k, the higher the LDA kernel rank becomes, and the less the improvement obtained. In this thesis, we introduce this form of semi-supervised LPP, termed SLPP.

To demonstrate the power of SLPP manifold modeling, 1072 faces of five expressions were taken from the CK [111] dataset. Each face was cropped and resampled down to 26x20 pixels, for an input D=520. These faces were trained via PCA and SLPP, where only the top three eigenvectors from each were utilized for the projection matrix, U, a 520×3 projection matrix. Each of the 1072 faces was projected down to 3 dimensions using each method, and the resulting scatter plots are shown in Figure 3.2. Figure 3.2 demonstrates the advantage supervised dimensionality reduction methods such as SLPP can have over unsupervised methods when performing subspace clustering. In Chapter 6, we will see that good subspace clustering is critical to minimizing coefficient contamination of sparse coefficients.

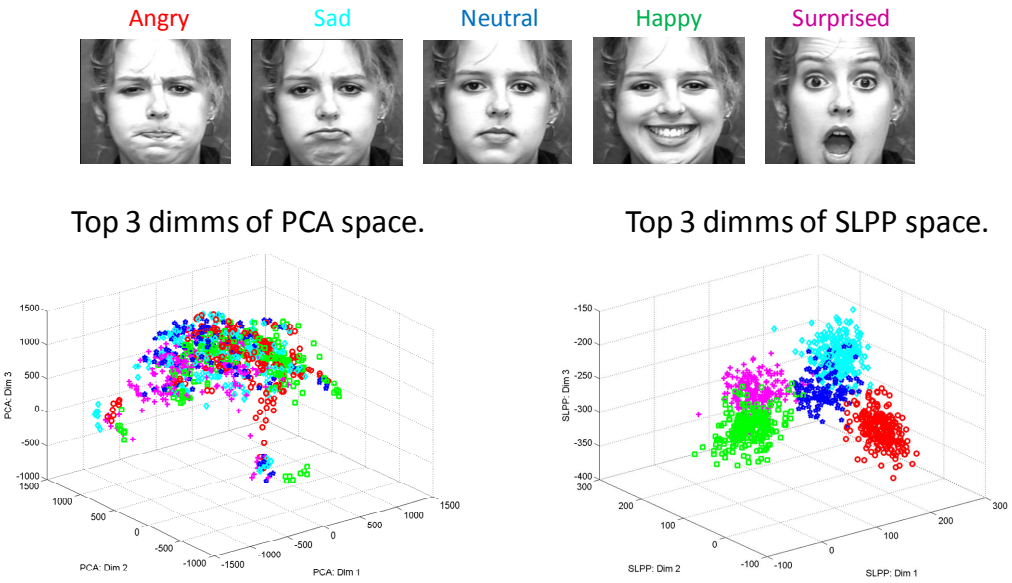


Fig. 3.2. Top row: A sample subject at the five training expressions. Bottom row: Each point represents one of 29 subjects exhibiting one of the 5 expressions after reduction to 3 dimensions using PCA (bottom left) and SLPP (bottom right). Expressions are color coded as shown above the training samples.

4 Applications of Dimensionality Reduction

4.1 Pose Estimation

4.1.1 Pose Estimation Introduction

Pose variation is the largest factor as we examine the differences from one 2D representation of a human face to the next. This needs to be addressed first when performing facial understanding of natural faces. It will be shown in Section 4.2 that accurate facial feature locations, such as corners of eyes, bottom of chin, etc., generally produce more accurate pose estimations than raw image pixels. Active Shape Models (ASMs), were used for pose estimation in [38, 39] because they offer good feature localization and are robust over appearance variations and partial occlusions. Geometric modeling of pose [38] uses the ASM feature points directly to estimate pose. For example, the left eye width divided by the right eye width varies as a function of head yaw in a surprisingly predictable fashion independently of pitch. This section uses the Bolin [24] ASM algorithm which produces 82 feature points in 2-D space to yield a 164 dimension input space

Manifold learning on ASM points for pose estimation is more effective than manifold learning operating on raw image pixels. Several dimensionality reduction techniques are utilized to contrast linear vs. nonlinear methods and unsupervised vs. supervised methods. A single manifold surface is trained across all facial pose input samples. This section will demonstrate that pose estimation in lower dimensional space using regression models offers advantages over the traditionally used k-nearest neighbor methods.

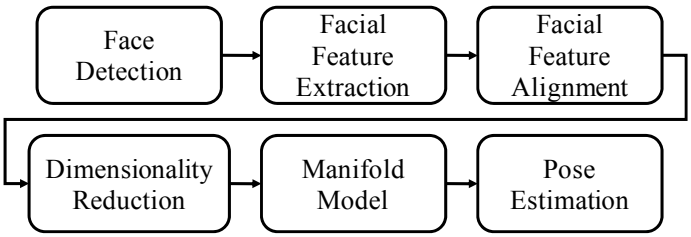


Fig. 4.1. Flowchart of pose estimation algorithm.

Figure 4.1 shows a flowchart illustrating the pose estimation approach. Faces are detected, facial features are extracted and normalized, the normalized facial features are projected onto a low dimensional manifold space, and a manifold model is used for pose estimation.

4.1.2 Pose Estimation Method

Face detection over a range of poses is accomplished using the Viola-Jones (or similar) method, which detects most faces over ±22^{o} yaw and reasonable pitch ranges. The ASM algorithm is initialized using

anthropometry guesses (based on facial bounding box coordinates) of the eye centroids and generates a vector of 82 feature positions for the eyes, eyebrows, nose, mouth, and face boundary. Scale, translation, and roll are removed via a generalized Procrustes analysis [67]. The CAS-PEAL-R1 pose [18] face database was used as the primary ground truth data source. This analysis uses 21 CAS-PEAL-R1 subjects, each at 21 poses, corresponding to three different pitch positions ([-30°, 0°, 30°]) and seven different yaw positions([-45°,-30°,-15°,0°,15°,30°,45°]. A total of 82 ASM feature points were manually annotated for each of the (21 subjects × 21 poses =) 441 faces. Figure 4.2 shows the corresponding ASM points from the sample CAS-PEAL-R1 subject shown in Figure 2.1.

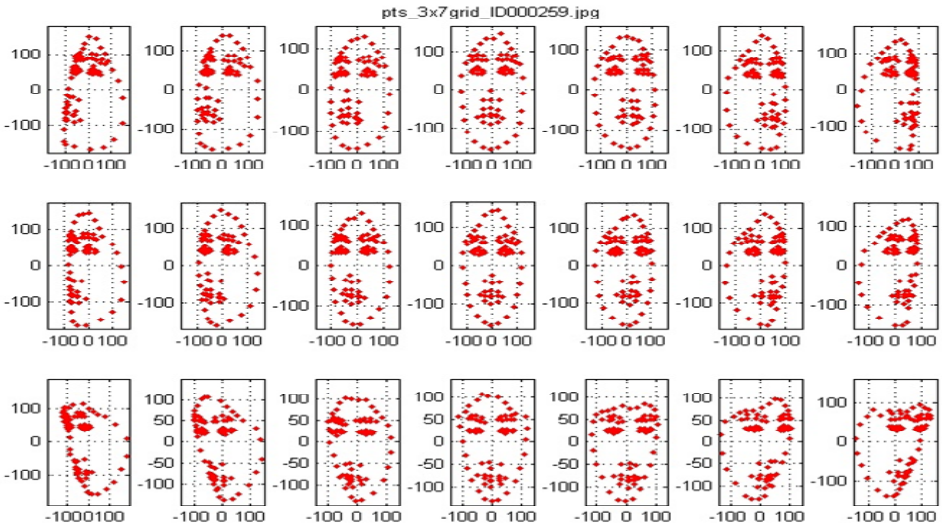


Fig. 4.2. ASM points for the 21 poses in Figure 2.1.

For the pixel based experiments, a square facial bounding box of width twice the inter-ocular distance, centered on the centroid of the eye-mouth triangle is cropped from each image. As the human face turns side to side, the 2D projection of inter-ocular distance varies as a function of yaw. This has the unfortunate side effect of increasing the facial bounding box with respect to head size. The top row of Figure 4.3 shows a sample subject from the CAS-PEAL-R1 dataset at seven different yaw positions. The middle row of Figure 4.3 shows the average left eye (red), right eye (green), and mouth (blue) location for all 1,042 subjects in the dataset. The number above each plot in the middle row is the inter-ocular distance at each yaw position. The bottom plot in Figure 4.3 shows the inter-ocular distance as a function of yaw. By inverting this curve, we can form a normalized yaw boost. For the raw image pixel experiments, inter-ocular distance was normalized by yaw using:

$$\Delta' = \Delta \sum_{j=0}^{4} c_j \text{yaw}^j \tag{4.1}$$

where Δ is the inter-ocular distance in the image, and Δ' is the boosted inter-ocular distance as if the subject were facing forward. The parameter c was solved by Ptucha [90] as [1.0, 0.0, 0.00014943, 0.0, 0.00000003].

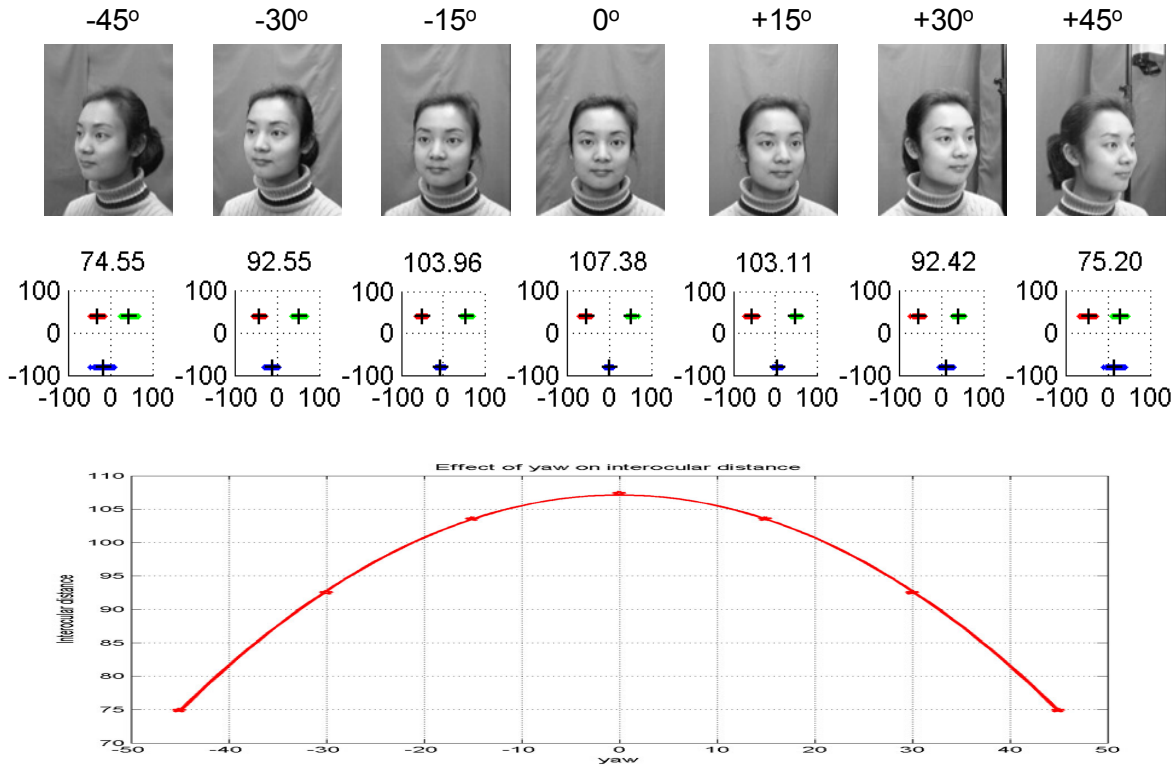


Fig. 4.3. Top row: 7 different yaw positions. Middle row: average location of eye and mouth coordinates for all subjects in CAS-PEAL-R1 dataset. Bottom row: Inter-ocular distance plotted vs. yaw.

Regarding ASM representations, such as the 82 point ASM, each face is represented by a 164 point vector. This high dimensionality feature space is parameterized by a lower dimensional embedded manifold discovered using manifold learning. The resulting lower dimensional manifold representation is more compact, more receptive to subsequent classification analysis, and easier to visualize. The training points on the surface of this manifold have known pose. Test faces are identically projected to this manifold surface, where the ground truth training points are used to estimate pose. Using PCA, LDA, LPP, supervised LPP, Isomap, or LLE dimensionality reduction, we can reduce the normalized 164

dimensional ASM points down to as few as three dimensions. Figure 4.4 shows a scatter plot of all 441 training faces reduced to three dimensions using PCA.

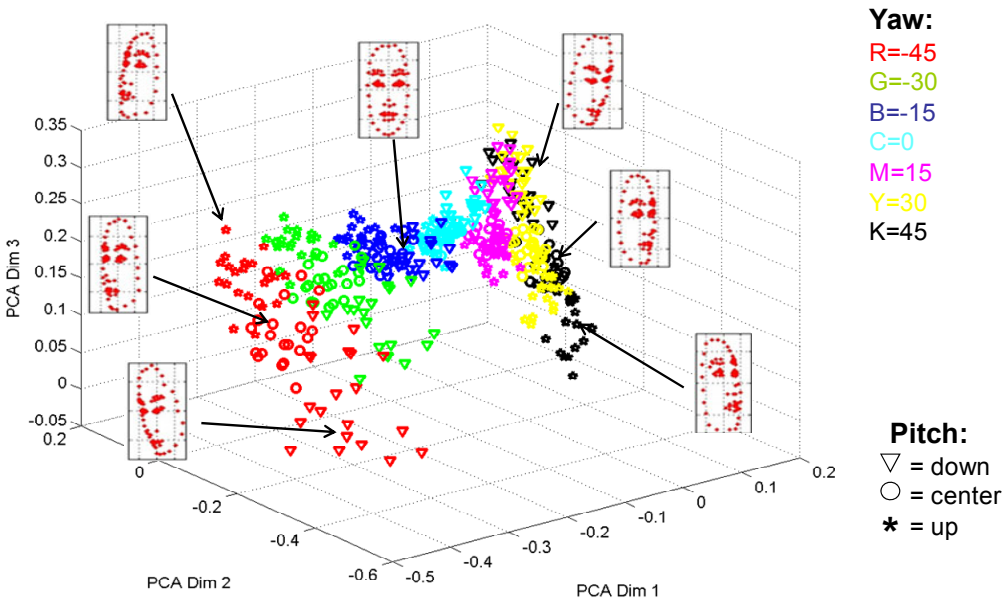


Fig. 4.4. Each 164 dimensional ASM face is mapped down to 1 point in this plot. Seven such ASM face positions in low dimensionality space are shown. Dimensionality reduction is done by using the top three PCA eigenvectors.

To resolve pose from low dimensional training data, we could use weighted k-nearest neighbors, multi-class SVM, or several other classification techniques. Alternatively, regressions can be performed on the low dimensional training data. Regression models perform necessary smoothing of data, are visually verifiable, and enable quick and efficient pose estimation for any new test point mapped to the manifold space. Separate manifold regression models were created for pitch and yaw, each of the general form:

$$\varphi_i = \sum_{j=0}^{m} c_{ij} y_i^j \tag{4.2}$$

where φ_i is our estimate of pitch and yaw, y_i^j is the dimensionality reduced term raised to various powers of j, coefficients c_{ij} and number of terms m are determined by the number of dimensions, d, in the low dimension projection space and the goodness of fit of the manifold surface to the ground truth training samples.

Polynomials of various dimensions and orders were created using pseudoinverse matrix regression. Letting $\boldsymbol{g} = \boldsymbol{Fc}$, where $\boldsymbol{F}$, the feature vector is created such that $\boldsymbol{F} \in \mathbf{R}^{nxt}$, where n is the number of training samples, and t is the number of terms. The terms used are the low dimensional values, the cross products of each, and higher order combinations. For example, for a three dimensional polynomial of order two, row i of $\boldsymbol{F}$ (corresponding to training sample i) would consist of 10 terms $[y_{i,1}\ y_{i,2}\ y_{i,3}\ y_{i,1}^2\ y_{i,2}^2\ y_{i,3}^2\ y_{i,1}y_{i,2}\ y_{i,1}y_{i,3}\ y_{i,2}y_{i,3}\ 1]$, i =1..n. $\boldsymbol{g} \in \mathbf{R}^{nx1}$, the ground truth vector, containing a single pitch or yaw value per training sample. We solve for separate pitch and yaw regression coefficients, $\boldsymbol{c} = \boldsymbol{F}^{-1}\boldsymbol{g}$, where $\boldsymbol{c} \in \mathbf{R}^{tx1}$. Because $\boldsymbol{F}$ is not square, we solve for the pseudoinverse as: $\boldsymbol{F}^{-1} \approx \boldsymbol{F}^{\dagger} = (\boldsymbol{F}'\boldsymbol{F})^{-1}\boldsymbol{F}'$. To compute pose estimates, we first construct the feature matrix for the test sample, $\boldsymbol{F}_s$, then use the learned value of $\boldsymbol{c}$ from the training samples to compute the predicted pose estimate using $\boldsymbol{g}'$ = $\boldsymbol{F}_s\boldsymbol{c}$. To determine if multiple polynomials would model the manifold surface better than one polynomial, the manifold was divided into p regions, defining p independent, but overlapping polynomial models; the selection of which manifold model to use is determined by the region where y_i is located in the manifold space.

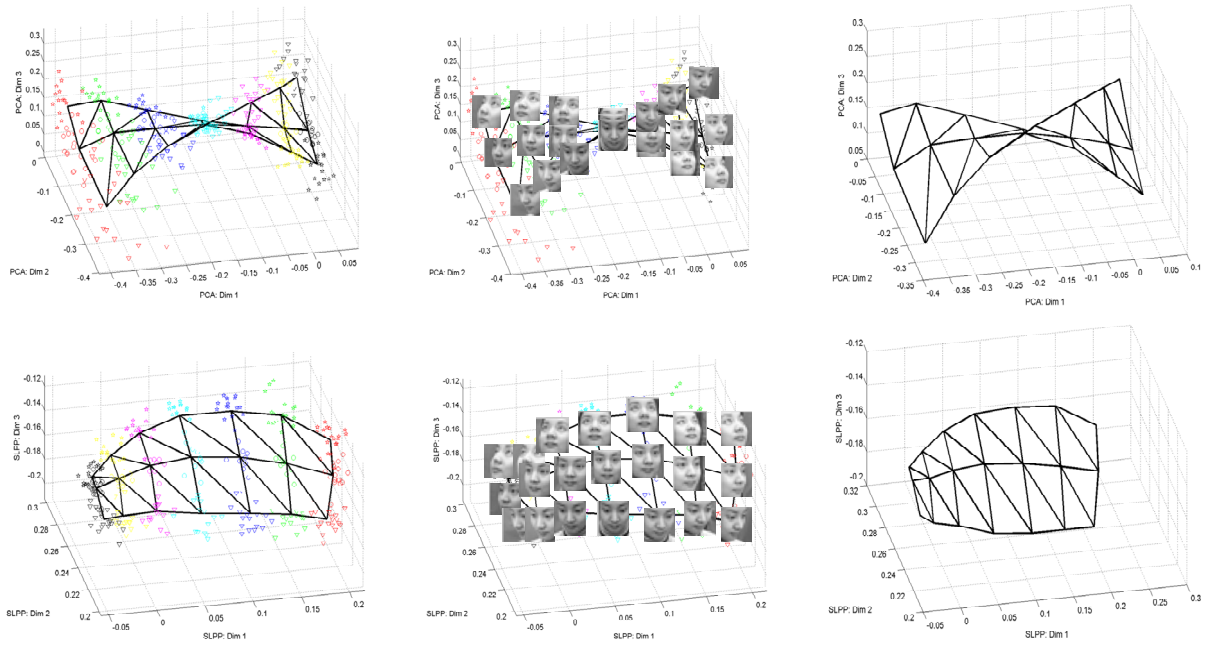


Fig. 4.5. Two different projection spaces: PCA on top, supervised LPP on bottom. On left, raw data with polynomial grid overlay. On center, faces overlaid at each vertex for visual clarity. On right, polynomial model used for pose estimation.

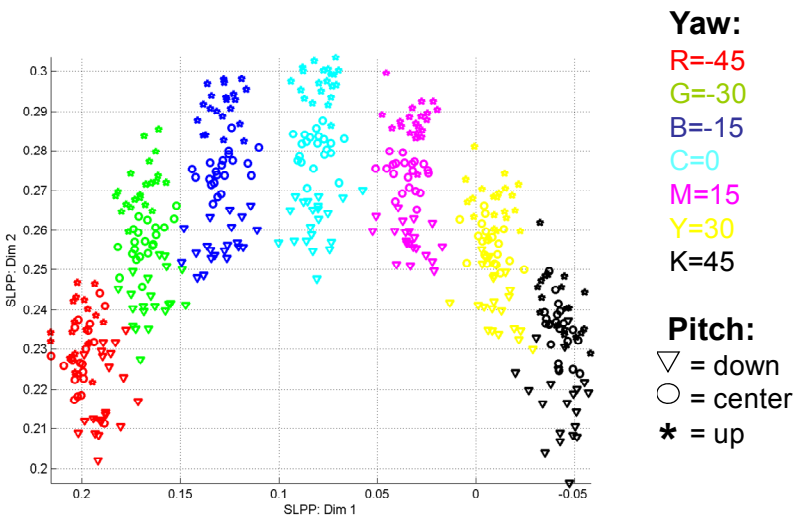


Fig. 4.6. First two dimensions of the supervised LPP projection space for each of the 441 training subjects.

Using the three most important dimensions, Figure 4.5 compares the low dimensional surface of PCA vs. the supervised LPP manifold surface. Each vertex is the average pitch/yaw from the 21 subjects, where averaging was done in the lower dimension projection space. While it would be quite difficult to discern pitch or yaw from the 164 dimensions, the ease and elegance of predicting such values in the lower dimension space is evident. To illustrate further, Figure 4.6 plots *d1* vs. *d2* for the supervised LPP projection method. There are 441 points shown in Figure 4.6, 21 subjects, each at 21 pose positions.

4.1.3 Pose Results

The pose estimation results were generated by the leave-one-out cross validation methodology (see Appendix I for discussion of cross validation methodologies). The average of 21 iterations is computed, where, for each iteration, there were 420 training faces and 21 test faces. Using geometric models as described in [38], a pitch root mean square percent error (RMSE) of 5.79^{o} and yaw RMSE of 4.82^{o} was found. Using supervised LPP along with Gaussian weighted k-nearest neighbor, different Gaussian widths give different results. For supervised LPP, optimal results were obtained for σ=0.007; resulting in RMSE errors of 6.57^{o} for pitch and 3.14^{o} for yaw.

For the manifold methods Isomap and LLE, out of sample computations were done using Nyström approximations. Table 4.1 summarizes the results of the six dimensionality reduction techniques when a three dimensional quadratic polynomial is used to model the low dimensional manifold surface.

TABLE 4.1. COMPARISON OF SIX DIFFERENT PROJECTION METHODS FOR POSE ESTIMATION. 82 POINT ASM POINTS USED AS INPUT; THREE DIMENSIONAL QUADRATIC POLYNOMIALS USED TO ESTIMATE POSE.

Technique	Pitch RMSE	Yaw RMSE
PCA, 82 ASM, top 3 dim	7.35	3.99
LDA, 82 ASM, top 3 dim	8.72	5.49
LPP, 82 ASM, top 3 dim	7.66	4.06
SLPP, 82 ASM, top 3 dim	6.27	4.24
Isomap, 82 ASM, top 3 dim	7.61	5.34
LLE, 82 ASM, top 3 dim	7.28	4.82

Geometric models and weighted k-nearest neighbors are quite accurate. However, considerable time and effort are required to test and retrain new geometric models; while polynomials are much faster than weighted k-nearest neighbors for real-time applications.

Using more dimensions or higher order polynomials in the manifold regression model often yields better results at the cost of algorithm complexity. Table 4.2 shows the results of varying the input dimensions and order of the polynomial model for the supervised LPP (SLPP) test cases. Additionally, the supervised LPP manifold was divided into seven regions according to pitch, where seven independent but overlapping quadratic polynomial models were defined, yielding an RMSE of 5.51^{o} for pitch.

TABLE 4.2. POSE ESTIMATION SUPERVISED LPP RMSE PROJECTION ERROR. POLYNOMIAL VARIED FROM 2 TO 5 DIMENSIONS, USING BOTH QUADRATIC AND CUBIC TERMS.

Num Dim	Pitch RMSE quadratic	Pitch RMSE cubic	Yaw RMSE quadratic	Yaw RMSE cubic
2	9.52	9.60	4.39	4.40
3	6.27	6.30	4.42	4.38
4	5.26	5.35	4.38	4.35
5	5.37	5.54	5.52	4.49

The yaw manifold surface is adequately defined by two dimensions. The pitch manifold surface is more complex, and requires four dimensions. The results illustrate that quadratic regression models are sufficient to model the manifold surface.

TABLE 4.3. POSE ESTIMATION USING IMAGE PIXELS INSTEAD OF ASM POINTS AS INPUT DIMENSIONAL DATA TO SUPERVISED LPP (SLPP).

Technique	Pitch RMSE	Yaw RMSE
SLPP, 20x20 faces	8.43	12.91
SLPP, 20x20 faces, Laplacian	8.61	12.06
SLPP, 20x20 faces, LOG	8.65	12.55
SLPP, 50x50 faces	9.23	12.12

Table 4.3 shows the results obtained with supervised LPP using various representations of image pixels. From top to bottom we have the baseline 20x20 image pixels, Laplacian high pass filtered pixels, Laplacian of Gaussian high pass filtered pixels, and baseline 50x50 image pixels. None of the raw image inputs performed as well as their ASM point counterparts. Although face pixels were normalized and in some cases enhanced, the information per input dimension is less than that obtained with ASM.

In summary, accurately placed ASM features are more accurate than pixel based methods when estimating pose. supervised LPP is the preferred dimensionality reduction method because of its high accuracy and execution speed. Employing manifold methods instead of geometrical models takes the guess work out of feature extraction and enables fast and accurate retraining. The use of polynomial regression methods, rather than weighted k-nearest neighbors, overcomes the need to solve for the optimal weighting function, and is faster to compute at runtime.

4.2 Expression Classification

4.2.1 Expression Classification Introduction

The manifold methods used for pose estimation in Section 4.1 can be applied to facial expression recognition. Again, it needs to be determined if facial pixels or ASM/AAM feature points discriminate expression classes better. Erroneous facial landmark assignments decrease the effectiveness of any expression classifier. The usage of facial landmarks alone ignores skin wrinkles associated with certain expressions and the process of locating these facial landmarks can be compute intensive. As such, there are reasons to investigate bypassing the ASM procedure and performing expression recognition directly on image pixels. He [112] used raw pixels along with PCA and a multi-class Minmax Probability Machine (MPM) to perform classification. Chang [113] used Isomap to define manifold surfaces for expression estimation. Shan [114] used Locality Preserving Projections (LPP) [105] dimensionality reduction along with k-NN to do expression recognition. Depending upon the subject conditions, some classifiers perform better than others. Ying [115] has proposed a system to fuse the output of multiple classifiers into a single optimal classification.

Figure 4.7 shows a flowchart illustrating the expression recognition approach. Faces are detected, pixels are cropped, facial features are extracted, the data is normalized and projected onto a low dimensional manifold space, and a manifold model is used for expression recognition.

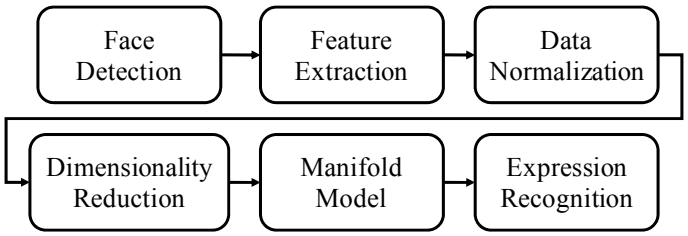


Fig. 4.7. Flowchart of the facial expression recognition framework.

4.2.2 Expression Classification Method

Face detection and ASM point localization are performed as was done in Section 4.1 on pose. For this work, faces are limited to small variation in pitch and yaw (< 10^{o}). Image pixel feature extraction is different, as we do not have to contend with wide variations in pose, and we need to accommodate elongated chin dropped facial expressions. As such, the square facial bounding box often used for face detection needs to be stretched to a 1:1.3 aspect ratio centered on the eyes horizontally. In the vertical direction, the top of the crop box is above the eye centerline by an amount equal to two-thirds the inter-ocular distance. This crop box is then resampled to a normalized size based on inter-ocular distance. Appendix II describes facial bounding box dimensions for various facial understanding tasks.

Figure 4.8 shows the family of cropped and processed raw pixels for a sample surprised expression. The pixel inputs were: (from left to right) no processing; normalized such that each had a mean of 128 and std dev of 100; normalized, then oversharpened with a 3x3 circular symmetric FIR filter of boost 3.0; edge detection with 5x5 Laplacian of Gaussian; edge detection with Canny Edge detector; and edge detection with Sobel detector (see Appendix-II for details).

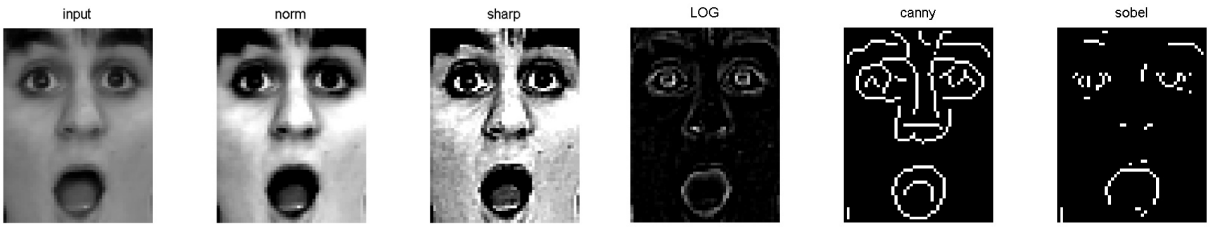


Fig. 4.8. Six variants of image pixel features studied in this section.

Both Stasm [25] ASM and Bolin [24] ASM are evaluated as each ASM implementation has its own cost-benefit trade-offs with regards to speed vs. accuracy. The Stasm ASM algorithm [12] produces 68 feature points in 2-D space to yield a 136 dimension input space. The Bolin ASM algorithm [13] produces 82 feature points in 2-D space to yield a 164 dimension input space. The Stasm ASM runs about 5x the speed of the Bolin ASM, but has smaller facial landmark templates and smaller search

ranges. The processed image pixels produce anywhere from 20x26 to 50x65 images, yielding 520 to 3250 dimension input space.

Training faces represented by image pixels or ASM coordinates are used in conjunction with the supervised LPP manifold learning technique to reveal a low dimensional manifold surface. The training points on the surface of this manifold have known facial expressions of angry, happy, neutral, sad, and surprised. Test faces are projected to this manifold surface, where a classifier estimates expression. To resolve expression from this low dimensional space, nearest class center, k-nearest neighbors, and Gaussian weighted k-nearest neighbors are evaluated. The class centers are class centroids solved during training. During runtime, the distance between the test point and each class center is calculated. The class with the smallest Euclidean distance is assigned to each point. Using Gaussian weighted k-nearest neighbors, different Gaussian widths give different results. For this work, optimum results were obtained for σ=0.003.

4.2.3 Expression Results

The Cohn-Kanade Facial Expression Database [111] was used as the primary ground truth data source. Although this database contains short video segments of 92 subjects, only 29 exhibit angry, happy, neutral, sad, and surprised expressions. These 29 subjects were used for this study. The training set contains 1072 image frames (232 for angry, happy, sad, surprised; 144 for neutral). Each image frame has 56 manually annotated facial feature points.

Stasm ASM and Bolin ASM were automatically run on all 1072 images. This enables the calculation of three sets of ASM points:

1) Cohn-Kanade 56 manually annotated facial feature landmark points. These points have no position error, aside from human placement error.
2) Automatically generated Stasm ASM. 68 facial feature landmark points with position error.
3) Automatically generated Bolin ASM. 82 facial feature landmark points with position error.

A generalized Procrustes analysis [67] on ASM point removes scaling (head size), translation, and rotation (head roll) variation amongst all subjects. Figure 4.9 shows sample facial landmark points for each of the three ASM point sets after Procustes alignment. The 56 perfectly placed Cohn-Kanade points are unfairly advantaged in that both the training and testing sets had perfectly placed landmark points. The two ASM sets of points are more representative of performance expected in a natural setting.

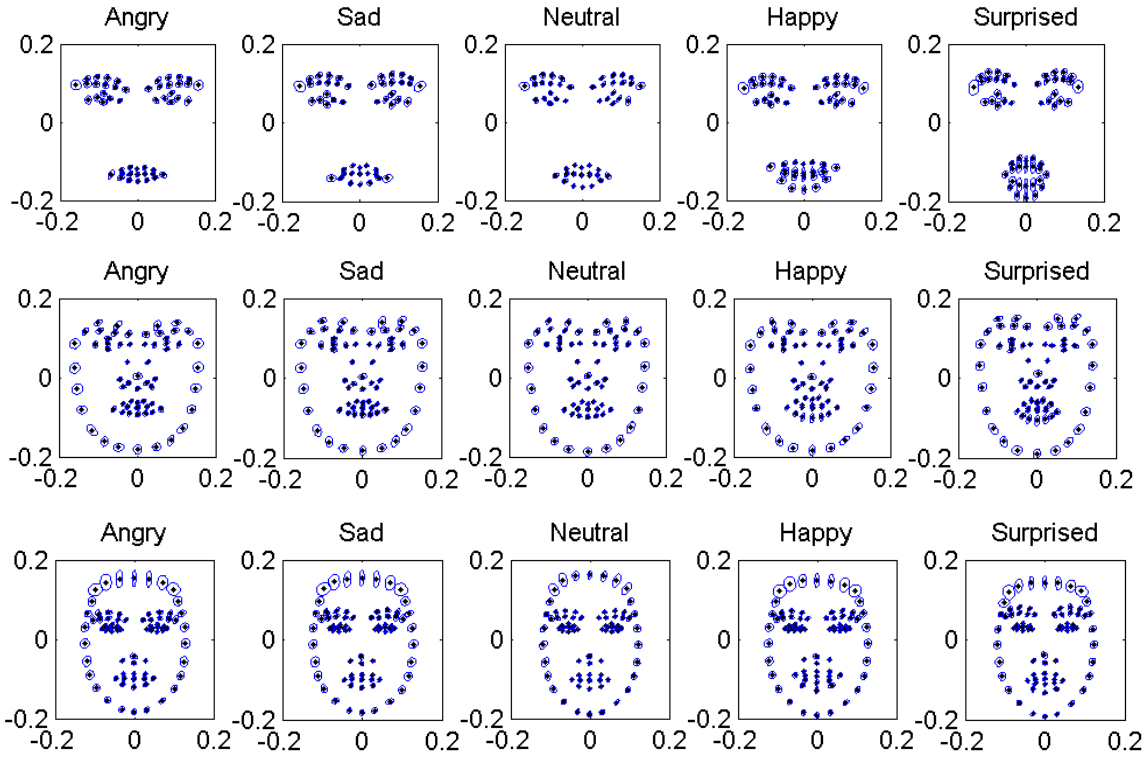


Fig. 4.9. Mean ground truth landmark points and two standard deviation limits for facial expressions. Top row is 56 point Cohn-Kanade. Middle row is 68 point Stasm. Bottom row is 82 point Bolin.

When using the 56 perfectly placed Cohn-Kanade ground truth points, a 3-dimensional, k=3 k-NN gives an overall accuracy of 87.34% with a leave-one-subject out cross validation. Each training sequence starts from neutral, and then gradually develops to full expression, which makes it difficult to distinguish between neutral and sad on many of the subjects. When automatically generated ASM points are used to localize both training and test facial landmarks, the mean classification performance drops to 50.93% for Stasm and 65.35% for Bolin.

The leave-one-subject out cross validation method was repeated for several types of classification schemes. Table 4.4 shows the results of nearest class centroid matching, weighted Gaussian distance, and several k-NN methods for the Bolin ASM method using 3 and 4 supervised LPP (SLPP) dimensions. Supervised LPP was not used in higher than 4 dimensions as the eigenvalues of the 5th and higher dimensions were not significant.

TABLE. 4.4. MEAN OF DIAGONAL OF CONFUSION MATRIX FOR VARIOUS LEAVE 1-SUBJECT OUT CROSS-VALIDATION EXPERIMENTS USING THE BOLIN ASM DATA FOR TRAINING AND TESTING.

	3 SLPP Dims	**4 SLPP Dims**

Class Centroid	68.55	66.07
Weighted Distance	65.84	65.90
1 K-NN	63.53	64.47
3 K-NN	65.35	64.92
5 K-NN	64.31	67.17
7 K-NN	64.05	64.79
9 K-NN	64.51	66.17

With regards to image pixels, interocular distances from 10 to 25 pixels were evaluated. This resulted in pixel crop boxes ranging in size from 20x26 to 50x65 pixels. Figure 4.10 shows the normalized and sharpened images when reduced to 3 dimensions by supervised LPP. Tables 4.5 and 4.6 show the results with supervised LPP constrained to 4 dimensions using nearest class centers and 3 k-NN classification respectively. SVM classifiers with both linear and radial basis functions were found to yield similar results, but the fast and robust nearest class centers are preferred with the class separation as shown in Figure 4.10. Table 4.7 shows the resulting confusion matrix for the normalized and sharpened 50x65 images using 3 k-NN in 4 dimensional supervised LPP space.

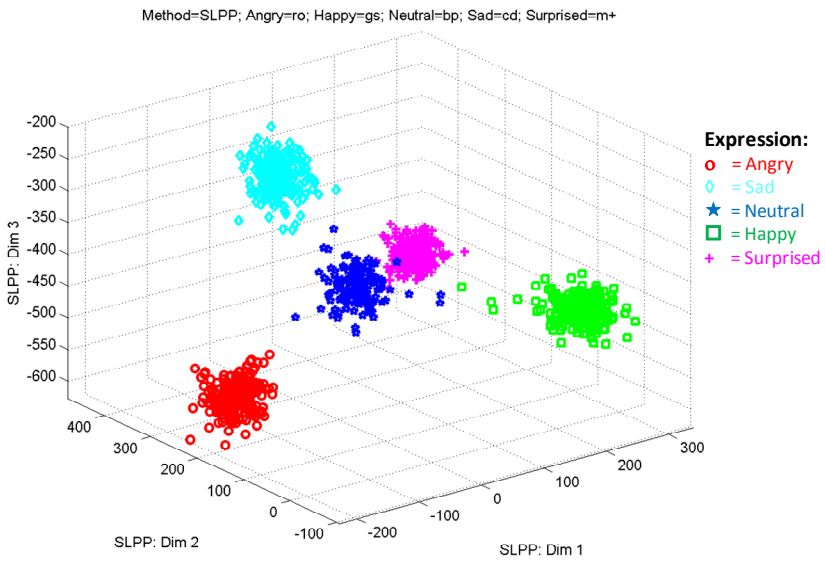


Fig. 4.10. Normalized and oversharpened raw pixels reduced to 3 dimensions using supervised LPP.

TABLE 4.5. MEAN OF DIAGONAL OF CONFUSION MATRIX WHEN RAW PIXELS REDUCED TO 4 DIMENSIONS AND USING NEAREST CLASS CENTER CLASSIFICATIONS.

	20x26	**30x39**	**40x52**	**50x65**
Raw	71.72	67.66	71.41	72.03
Norm	70.63	71.56	72.81	72.97

NormSharp	69.06	66.88	71.56	74.84
LOG	66.25	63.13	62.66	64.53
Canny	50.63	51.88	50.47	55.94
Gabor	66.41	70.78	71.56	74.22

TABLE 4.6. MEAN OF DIAGONAL OF CONFUSION MATRIX WHEN RAW PIXELS REDUCED TO 4 DIMENSIONS AND USING 3 K-NN CLASSIFICATIONS.

	20x26	**30x39**	**40x52**	**50x65**
Raw	66.72	65.78	69.84	70.63
Norm	68.28	72.81	72.50	73.59
NormSharp	68.13	70.31	73.28	75.47
LOG	64.84	60.47	62.03	65.47
Canny	50.16	51.25	51.56	55.78
Gabor	66.09	68.13	71.09	74.06

TABLE 4.7. CONFUSION MATRIX USING 50X65 NORMALIZED AND SHARPENED PIXELS ON ALL TRAINING AND TEST SUBJECTS. LEAVE 1-SUBJECT OUT CROSS-VALIDATION. 3 K-NN CLASSIFICATION. MEAN OF DIAGONAL IS 75.47%.

	Angry	**Sad**	**Neutral**	**Happy**	**Surprised**
Angry	78.13	7.03	11.72	3.13	0.00
Sad	7.03	75.00	15.63	0.00	2.34
Neutral	7.81	23.44	64.84	1.56	2.34
Happy	0.78	3.13	7.03	89.06	0.00
Surprised	0.00	12.50	6.25	10.94	70.31

Figure 4.10 shows a clear intra-class separation in 3-dimensional supervised LPP space. With such clean separation, one might expect confusion matrices to be nearly perfect. Furthermore, previous publications, such as [15] report classification accuracies around 90%. This difference can be rectified via cross validation methodology. The commonly used k-fold cross validation selects the test and training sets randomly. Table 4.8 shows a comparison of 16-fold cross validation vs. leave-one-subject out cross validation. The results demonstrate that the method performs very well under 16-fold cross validation testing, while the leave-one-subject validation out is more challenging and does not generalize as well when the number of subjects is relatively small.

TABLE 4.8. CLASSIFICATION ACCURACY USING LEAVE 1-SUBJECT OUT VS. 16-FOLD CROSS VALIDATION METHODOLOGIES. MEAN OF DIAGONAL OF CONFUSION MATRIX WHEN NORMSHAP RAW PIXELS (20X26 AND 50X65 PIXEL FACES SHOWN) REDUCED TO 4 DIMENSIONS VIA SUPERVISED LPP AND CLASSIFIED USING NEAREST CLASS CENTER, 3 K-NN, AND SVM WITH LINEAR KERNEL.

	Nearest Class Center		3 K-NN		SVM	
	20x26	**50x65**	**20x26**	**50x65**	**20x26**	**50x65**
Leave 1-	69.06	74.84	68.13	75.47	64.69	75.31

subject out						
16-fold cross validation	92.64	94.71	92.56	94.95	96.16	97.94

In summary, normalized and sharpened pixel data delivered good results using 3 k-NN and SVM, however, the much faster nearest class centroid classification is the method of choice for real time processing, as it provides comparable performance while execution speed is much faster. Neither ASM method performed as well as the pixel based methods, perhaps because neither was trained with faces of varying expressions. The most confusion lies between neutral and sad expressions.

4.3 Mixed Pose and Expression

4.3.1 Mixed Pose and Expression Introduction

Recent work [50, 116-118] has demonstrated promising results for non-frontal facial expression recognition. These methods jointly analyze facial pose and expression necessitating the need to discriminate between the rigid motion of the head and the non-rigid motion of the face. Zhu [118] localizes facial landmarks using analytical models to predict pose and expression simultaneously. 3D datasets have enabled [50, 116, 117] to use manually annotated facial feature points from 3D models to predict pose as well as expression. An interesting note is that in Hu [116], the authors report that machine learning methods achieve higher performance in recognizing expression at non-frontal yaw positions compared to frontal yaw positions. One hypothesis for that finding is that human faces are symmetric, whereby the left and right side of the face presents the same redundant (albeit more robust to noise, lighting, occlusions, etc.) information. As yaw is introduced, the information content is increased as we get to see a new dimension of the face.

4.3.2 Mixed Pose and Expression Classification Method

For simultaneous pose and expression recognition, we focus on the supervised LPP dimensionality reduction method and consider three methods for the construction of the adjacency matrix $\boldsymbol{W}$. The first method groups all pose variation of the same expression under the same class label, giving k classes as done previously with (3.22). One drawback of this method is that resulting adjacency matrix $\boldsymbol{W}$ has rank k, where k is the number of classes. This limits the number of eigenvalues to be <= k. With small k, this can hinder the supervised LPP performance.

The second method identifies $k \times p \times y$ classes, where p is the number of pitch sub-classes, and y is the number of yaw sub-classes. This gives supervised LPP more discriminating power and allows the rank of the $\boldsymbol{W}$ matrix to increase dramatically.

The third method learns separate expression manifolds, one at each pose position, where each manifold predicts a single expression. As in [39], a test input face is first passed through a pitch and yaw manifold to get a pitch and yaw value. The four manifolds surrounding the input face pitch and yaw are used for expression classification, where a voting scheme determines final classification. To model the distribution of votes, a method similar to the cubic convolution filter is employed. Defining $[pt, yt]$ as the pitch and yaw of the test sample and $[pm_i, ym_i]$, $i=1..z$ as the pitch and yaw for each of the z manifolds, we define the votes per manifold as:

$$\begin{aligned} \text{pitch\_scale}_i &= 1.5|\, pt - pm_i \,|^3 - 2.5|\, pt - pm_i \,|^2 + 1 \\ \text{yaw\_scale}_i &= 1.5|\, yt - ym_i \,|^3 - 2.5|\, yt - ym_i \,|^2 + 1 \\ \text{votes}_i &= \text{round}(100 * \text{pitch\_scale}_i \bullet \text{yaw\_scale}_i) \end{aligned} \tag{4.3}$$

4.3.3 Mixed Pose and Expression Results

Given the lack of freely available datasets exhibiting expressions at various pose and yaw positions, we make use of the Singular Inversions' FaceGen Modeler [http://www.facegen.com/modeller.htm] to create a dataset of 1,200 images. Twelve subjects, each at 5 pitch (-30, -15, 0, +15, +30), 4 yaw (0, +15, +30, +45), and 5 expressions (angry, fear, happy, sad, surprised) were used for the analysis that follows. Figure 4.11 shows a sample subject at 20 pose positions and 5 sample expressions of differing subjects.

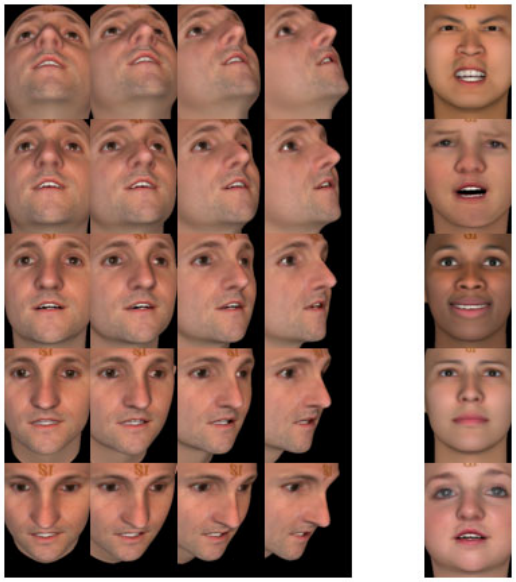


Fig. 4.11. Twenty ([+30, +15, 0, -15, -30] pitch × [0, +15, +30, +45] yaw) pose positions on the left. Five (angry, fear, happy, sad, surprised) facial expressions on the right.

The facial pixels are cropped and normalized as per the previous expression experiments down to 20x26 pixels, yielding a 520 dimension input space. In all experimental results, the leave-one-subject-out

cross validation methodology was used. The cumulative confusion matrix is formed and the mean of the diagonal is reported as a measure of accuracy.

The 1,200 images, each $\in \mathbf{R}^{520}$, are entered into supervised LPP to form a single manifold. Labeling the ground truth by expression (angry, fear, happy, sad, surprised) reduces the manifold dimensionality to $\in \mathbf{R}^{5}$. Using multiclass linear SVM, the resulting confusion matrix has an average diagonal accuracy of *76.1%*. When the manifolds are trained to classify pitch (-30, -15, 0, +15, +30), the accuracy increases to *97.17%*. When the manifolds are trained to classify yaw (0, +15, +30, +45), the accuracy is *94.3%*. This indicates that pose is the dominant signal, while expression is secondary.

To further understand the interaction between pose and expression, a single manifold was created with 100 (5 expressions × 5 pitch × 4 yaw) class labels. Figure 4.12 shows the first two dimensions of this manifold. The increased number of labels enables finer discrimination by supervised LPP and increases the rank of the adjacency matrix. The overall confusion matrix accuracy is *85.3%*. However, if we are only interested in expression, we can group all 20 of each pose variations per expression into one class (i.e. there are 20 sub-classes of type angry, 20 of fear, etc.), giving a 5 entry confusion matrix for expression. The expression accuracy of this matrix is *89.0%*. Therefore, by including more discriminate class labels, we increase accuracy by *12.9%*.

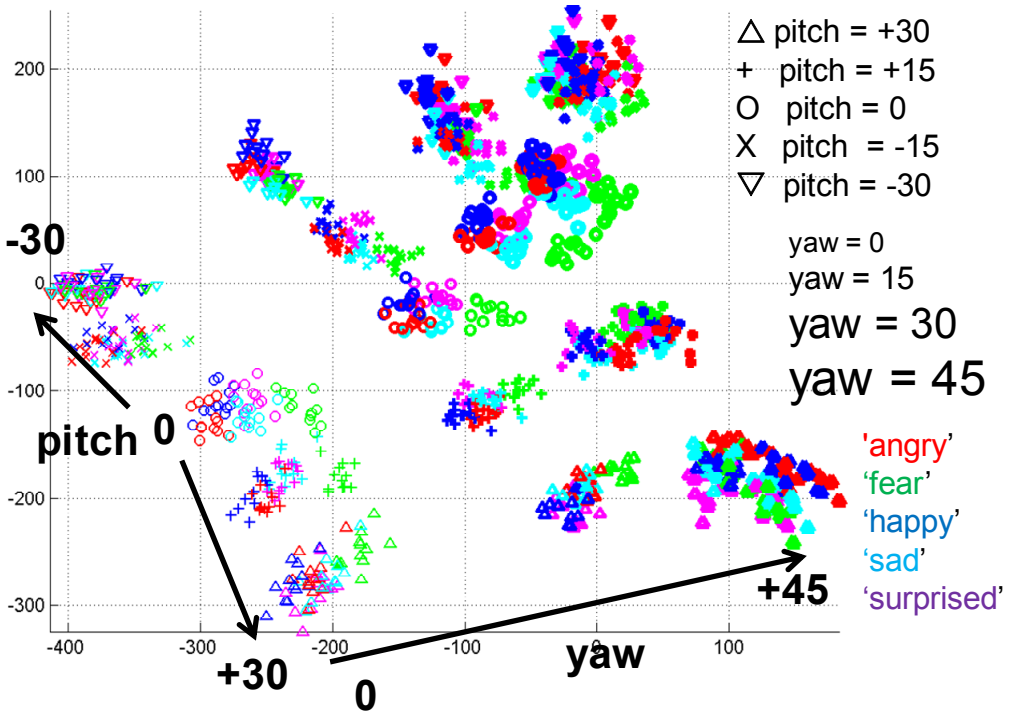


Fig. 4.12. First 2 dimensions of 100 class centers supervised LPP manifold. The 20 large clusters are arranged by pose. Within each of the 20 large clusters are 5 smaller expression clusters.

TABLE 4.9. EXPRESSION CONFUSION MATRIX ACCURACY RESULTS FOR 5 AND 100 CLASS GROUND TRUTH CENTERS.

	PCA	LDA	SLPP
5 class	70.4	63.9	76.1
100 class	87.0	84.7	89.0

In order to evaluate the performance of supervised LPP (SLPP), Table 4.9 shows a comparison of PCA, LDA, and supervised LPP methods for both 5 class and 100 mixed class ground truth labeling. In addition to the method used for dimensionality reduction, we compare different approaches in feature extraction. We consider adjusted pixel values, Local Binary Patterns (LBP), and Gabor filters. More specifically, we examine the following five techniques: 1) pixel values normalized to μ=128 and σ=100 (Norm); 2) over sharpened pixels (Sharp); 3) $LBP_{8,1}$ image (LBP); 4) 3x5 concatenated block histogram of $LBP_{8,1}^{u2}$ (LBP-h); and 5) Gabor processed images (Gabor) (See Appendix-II for details). Each of these 5 image types is passed into the supervised LPP framework, where manifolds are built for test and training images identically. Table 4.10 shows that Norm is comparable in performance to the more computationally expensive pre-processing techniques.

TABLE 4.10. SUPERVISED LPP EXPRESSION CONFUSION MATRIX ACCURACY RESULTS USING 5 DIFFERENT TYPES OF PRE-PROCESSING ON THE RAW IMAGE PIXELS.

	Norm	Sharp	LBP	LBP-h	Gabor
5 class	76.1	73.8	73.2	73.6	73.7
100 class	89.0	90.8	90.1	85.5	88.8

For the final test, twenty separate expression manifolds are created, one at each pose position. Test input faces are first passed through a pitch and yaw manifold where the resulting pitch and yaw indicates which manifold is used for expression classification. For $|\text{pitch}|<30^o$ and $|\text{yaw}|<45^o$, the four surrounding manifolds are evaluated for expression. For pose values outside this range, only the two boundary manifolds are used. Using Eq. (4.3), the resulting classification accuracy on this 20 manifold model is *89.3%*. This indicates that although better results may be possible with multiple manifolds, using mixed classes is equally as effective.

In summary, the proposed approach can simultaneously classify pose and expression on low resolution images without the need for sophisticated pre-processing or computationally expensive facial feature point localization. By understanding the strengths and weaknesses of supervised manifold learning techniques, expression classification is performed much faster and is as good as techniques that are constrained to frontal pose positions. Mixed class pose and expression manifold techniques perform

better than expression-only manifold techniques and work just as well as fusing together results from multiple manifolds.

5 Sparse Representations

The Sparse Representations (SRs) framework was inspired by studies suggesting selective firing of neurons in V1, the first layer of the primate visual cortex [8, 9]. In the SRs framework, a test signal is represented as a sparse linear combination of training exemplars. Initially developed for reconstructive purposes on audio, image, and video data, recent work has focused on making the SR framework more discriminative. It has been shown that under typical conditions, the minimal solution is the sparsest one [119, 120]. There have been several studies optimizing both the ℓ^1 minimization [121, 122] as well as the selection of dictionary elements [123-125]. Furthermore, simple-cells in the visual cortex appear to exhibit nonnegative properties leading to the Nonnegative Matrix Factorization (NMF) algorithm [18]. Wright [78] detailed the use of sparse representations for facial identification further fueling the desire to use SR frameworks for face and gesture tasks.

5.1 Sparse Representation Theory

Many scientific problems can be reformulated as finding a linear representation of an input signal from a dictionary, Φ, of training examples. Letting the input signal be $\boldsymbol{y}\in\mathbf{R}^d$ (note: y is the low dimension representation of $\boldsymbol{x}\in\mathbf{R}^D$) and the dictionary of examples $\Phi\in\mathbf{R}^{dxn}$, a natural way to represent y from Φ is by an approximation obtained through solving the system of linear equations $\hat{y}=\Phi\boldsymbol{a}$, where $\boldsymbol{a}\in\mathbf{R}^n$ is the weight of each training exemplar in dictionary Φ. In general, n, the number of training samples can be quite large, and d, the dimension of each sample is smaller, $d<n$, yielding an overcomplete dictionary. Further, the number of non-zero coefficients in $\boldsymbol{a}$ is also small, $||\boldsymbol{a}||_o<n$, meaning that each test sample can be represented with only a few elements in Φ. In most practical cases, the linear system of equations has either no solution or multiple solutions. This problem is usually tackled by applying a least squares regression:

$$\hat{\boldsymbol{a}} = \arg\min ||\boldsymbol{a}||_2 \quad \text{s.t. } \hat{\boldsymbol{y}} = \Phi\boldsymbol{a} \tag{5.1}$$

where $||\boldsymbol{a}||_2$ is the ℓ^2 norm and the reconstruction coefficients $\boldsymbol{a}$ are given by $\boldsymbol{a}=\Phi^{\dagger}\boldsymbol{y}$, where $\Phi^{\dagger}=(\Phi^{T}\Phi)^{-1}\Phi^{T}$ is the Moore-Penrose pseudo-inverse of Φ. The ℓ^2 solution offers two significant benefits, a closed form solution and the generation of a unique solution. Although regularizing the energy of the solution has been successfully applied in many problems, minimizing the ℓ^2 norm may not lead to the optimum solution for specific types of signals such as images.

For many input signals of interest, such as natural images, only a small number of dictionary elements are needed to represent them. For sparse signals, the objective of SRs is to identify the smallest

number of nonzero coefficients $\boldsymbol{a} \in \mathbf{R}^n$ such that $\hat{\boldsymbol{y}} = \Phi \boldsymbol{a}$. It has been shown that under typical conditions, the minimal solution is the sparsest one [119, 120]. The solution to this problem can be obtained by solving the following optimization problem:

$$\hat{\boldsymbol{a}} = \min \|\boldsymbol{a}\|_0 \quad s.t. \ \hat{\boldsymbol{y}} = \Phi \boldsymbol{a} \tag{5.2}$$

where the sparsity constraint is given by the zero-norm $\|\cdot\|_0$ which counts the non-zeros elements- perhaps the best measure of sparsity. Unfortunately, the problem in Eq. (5.2) is non-convex and therefore difficult to solve for practical problems. Two approaches have been presented for reformulating the intractable problem in Eq. (5.2) into an efficient optimization: convex relaxation and greedy algorithms.

The convex relaxation approach was introduced in the pioneering work of Donoho [119] and Candes [120], where it was shown that if the solution satisfies certain constraints, such as the sparsity of the representation, the solution is equivalent to the solution of the following LASSO (Least Absolute Shrinkage and Selection Operator, often referred to as Lasso) regression problem in statistics:

$$\hat{\boldsymbol{a}} = \min \|\boldsymbol{a}\|_1 \quad s.t. \ \hat{\boldsymbol{y}} = \Phi \boldsymbol{a} \tag{5.3}$$

where $\|\boldsymbol{a}\|_1 = \Sigma |\boldsymbol{a}|$. The benefit of using the ℓ^1 minimization is that the problem can be efficiently solved using convex optimization algorithms. When noise is present in the signal, a perfect reconstruction using Eq. (3) may not be feasible. Therefore, we require that the reconstruction is within an error tolerance. This optimization, called Basis Pursuit Denoising (BPDN), reformulates Eq. (5.3) as:

$$\hat{\boldsymbol{a}} = \min \|\boldsymbol{a}\|_1 \quad s.t. \|\hat{\boldsymbol{y}} - \Phi \boldsymbol{a}\|_2^2 \leq \varepsilon \tag{5.4}$$

Often Eq. (5.4) is approximated by loosening the error constraints and reconfigured to specifically include a regularization term, λ which encourages sparseness by incurring a penalty on the resulting coefficients:

$$\hat{\boldsymbol{a}} = \min\{\|\hat{\boldsymbol{y}} - \Phi \boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1\} \tag{5.5}$$

To ensure arbitrarily large dictionary elements won't get paired up with very small values of a, the elements of Φ are forced to have ℓ^2 norms equal to one. Perhaps the most widely used method to solve the ℓ^1 minimization of (5.4) and (5.5) is Orthogonal Matching Pursuit (OMP) [126]. OMP selects one dictionary element at a time in a greedy fashion and can quickly converge to a solution. In particular, OMP picks the dictionary element with the largest correlation with the test sample. With the first element selected, an orthogonal residual error vector is formed. This becomes the new response for which we are trying to solve. In similar fashion we select the next dictionary element with the largest correlation with the orthogonal residual error. The process continues until the error residual falls below a predetermined

threshold. Solving for the orthogonal residual error requires a costly matrix inversion. More efficient methods use a progressive Cholesky or QR update mechanism [127].

Forward stagewise is a cautious version of forward selection method. Instead of taking a few large steps, perhaps a thousand smaller steps are taken. Least Angle Regression with laSso (LARS) [122] attempts to strike a balance between forward selection and forward stagewise methods. As used in this research, LARS is very similar to Lasso regression, where Lasso is a constrained version of ordinary least square regression. In fact, both Lasso and forward stagewise are basic variants of LARS. Parsimony is actually a side effect of Lasso, and thus a side effect of LARS.

Similar to forward selection, the LARS algorithm selects one dictionary element at a time. Specifically, we begin by selecting the dictionary element that has the largest correlation with the test sample. However, instead of stepping entirely in this direction, we go as far as we can until some other dictionary element has as big of a residual error with the current residual. Instead of proceeding in an orthogonal direction, LARS now proceeds in a direction that is equiangular with the two elements. As it moves in this direction, it is continually computing the residual error. It continues to step in this direction until the residual error is equaled by a third dictionary element. LARS then continues the process by stepping in the direction equiangular between the three elements. This process continues until the error residual falls below a predetermined threshold. After t steps are made, we have t non-zero coefficients. Although the geometry is complex, [122] has also described a highly efficient one pass implementation which is used for this research.

5.2 Sparse Representation Classification

Given the sparse representation coefficients $\hat{\boldsymbol{a}}$ of a test sample using the dictionary Φ, various techniques can be used for identifying the class of the test image. To motivate the selection of a classification scheme based on SRs, Figure 5.1 shows a sample image along with its six non-zero $\hat{\boldsymbol{a}}$ coefficients from ℓ^1 sparse representation. The 330 dictionary elements are grouped by class and labeled on the x-axis from left to right as angry, contempt, disgust, fear, happy, sad, and surprised. The test face belongs to the *sad* class, but it is readily apparent that dictionary elements come from four classes.

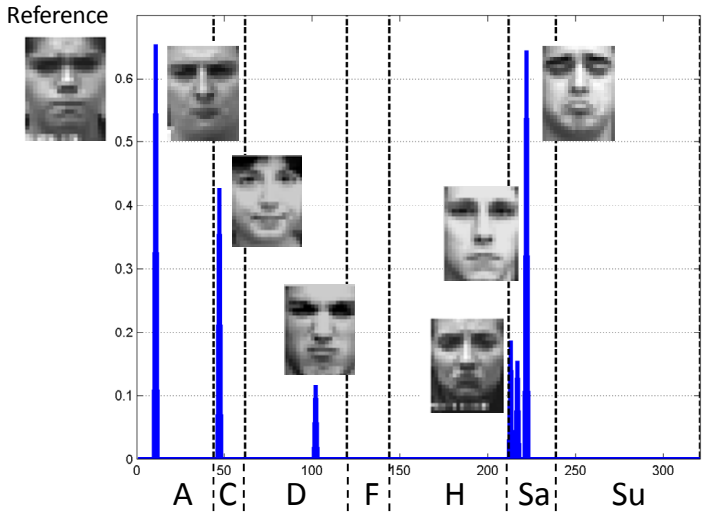


Fig. 5.1. Sample sad face and its top 6 $\hat{a}$ coefficients. Coefficient categories are Anger (A), Contempt (C), Disgust (D), Fear (F), Happy (H), Sadness (Sa), and Surprise (Su).

Using the set of sparse representation coefficients $\hat{\boldsymbol{a}}$ as input, there are a variety of ways to perform classification. Referring again to Figure 5.1, we can assign the test sample to the class with the most significant $\hat{\boldsymbol{a}}$ coefficient, the class with the most non-zero $\hat{\boldsymbol{a}}$ coefficients, or to the class with the greatest energy (sum or quadratic sum) of all $\hat{\boldsymbol{a}}$ coefficients. The first strategy would have incorrectly classified the exemplar Sad face as Angry, while the latter two would have resulted in a correct classification. Experiments have concluded that using a reconstruction error outperforms the aforementioned methods. The reconstruction error method estimates the class c^* of a query sample $\boldsymbol{y}$ by comparing the reconstructed sample using sparse coefficients $\boldsymbol{a}$ from all classes to the reconstructed sample using coefficients $\boldsymbol{a}^c$ from each respective class as:

$$c^* = \arg\min_{c=1\ldots k} \|\boldsymbol{y} - \Phi\, \boldsymbol{a}^c\|_2 \tag{5.6}$$

Equation (5.6) assigns the test sample classification to the class most similar to the fully reconstructed sample. Similar to (5.6), one can construct k independent dictionaries, one dictionary for each class, and then assign the test sample to the class whose dictionary minimizes the reconstruction error.

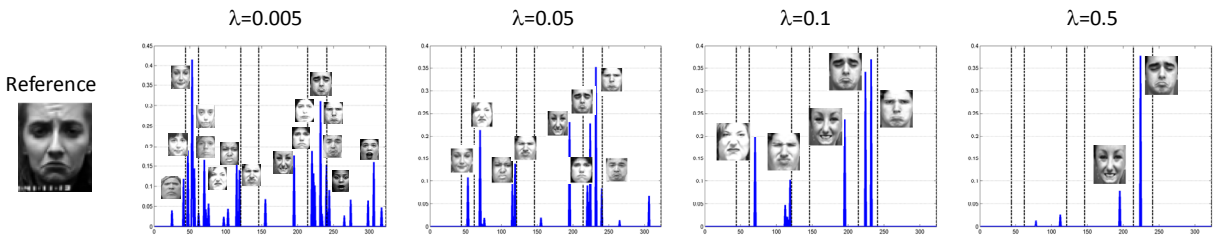
The coefficients in Figure 5.1 have been constrained to be non-negative and y is estimated by using all coefficients, $\mathbf{y} \approx \hat{\boldsymbol{y}} = \Phi \boldsymbol{a}$. Empirical studies in Section 6.3 will show improved performance of the nonnegative SR method over ℓ^1 SR. This is consistent with previous findings and also mimics the behavior of simple-cells in the visual cortex. Intuitively, the meaning of negative coefficients for

classification purposes is not obvious, since a negative weight does not provide direct information as to which class a sample should belong.

The greater λ in Eq. (5.5), the sparser the solution and the fewer atoms from Φ that are used to estimate $\hat{y}$. The top row of Figure 5.2 illustrates the effect λ has on the solution of a sample face. Starting with a dictionary size of 330 faces, the regularization parameter λ determines how many atoms are used to represent the reference face. The parsimonious notion of sparsity would indicate that generous values of λ are tolerable. To prevent over fitting to the training data, one can either increase the size of the training dictionary or increase λ. On sufficiently over-complete dictionaries, empirical testing has shown $0.05 \leq \lambda \leq 0.25$ yields a good tradeoff between classification accuracy and processing complexity.

To gain further intuition for the LPP adjacency matrix weight α in (3.26), and the ℓ^1 regularization parameter λ in (5.5), the bottom row of Figure 5.2 shows the effect of varying α and λ to different dataset and classification problems. The CK+ dataset is a posed dataset with rather large class to class separation in supervised LPP space. The three LFW datasets are natural datasets with considerably more overlap between class distributions in supervised LPP space. (See Appendix I for discussion on datasets.)

With respect to the Expression CK+ plot in the bottom of Figure 5.2, the inherent data is linearly separable. As long as the adjacency matrix includes some amount of supervision, the accuracy is rather flat. With respect to the LFW plots, it is evident that neither the Gaussian or LDA adjacency matrix performs as well as the convex combinations of the two. Furthermore, while the 2-class gender and 5-class race plots have large drops when α=1, the 10-class race-gender plot is not as pronounced because the rank of $\boldsymbol{W}$ is higher. Additionally, less regularization on the LFW datasets gives preferred results due to considerable sample to sample variability in this dataset, which makes it difficult to have an over-complete dictionary.



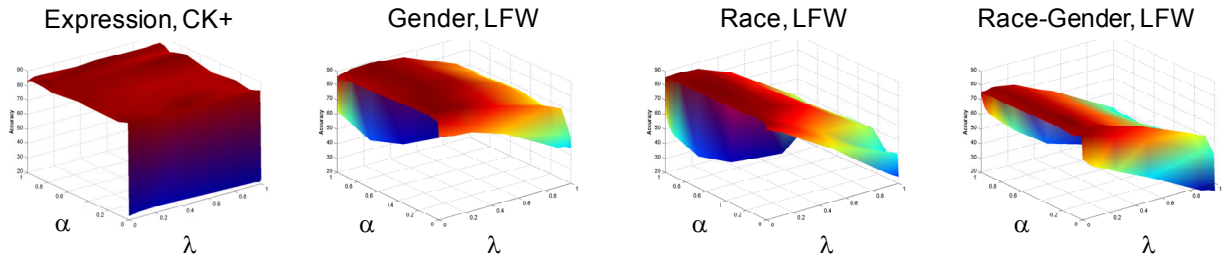


Figure 5.2. Top row: Varying the regularization parameter λ affects the sparseness of the ℓ^1 coefficients. More regularization creates fewer non-zero $\hat{\boldsymbol{a}}$ coefficients. Bottom row: Classification accuracy (vertical axis) obtained by varying α, the adjacency matrix blend parameter along with ℓ^1 regularization parameter λ on varying datasets and classification problems. The left-most plot is a 7-class posed dataset. The three remaining plots are 2-class gender, 5-class race, and 10-class race-gender natural datasets. As λ goes to the right, we increase regulation. As α goes to the left, we increase LDA discriminative embedding.

5.3 Sparse Representation Dictionaries

When constructing Φ the goal is to generate an over-complete dictionary with more samples than dimensions per sample. This allows the necessary degrees of freedom to choose the sparsest solution and produces smooth and graceful coefficient activity across diverse test samples [128]. For efficiency, we would like the dictionary size to be small, necessitating the need for linearly independent or decorrelated samples. To minimize reconstruction error, we often need to increase the size of the dictionary, however too many samples in the dictionary result in unstable estimates for $\hat{\boldsymbol{a}}$ as well as greater computational burden. Wright *et al.* [78, 129] used all n training samples in Φ. Engan *et al.* [130] introduced the Method of Optimal Directions (MOD), while Aharan introduced K-SVD [131] to learn an over-complete but small dictionary.

Both MOD and K-SVD are iterative techniques, where each iteration first sparsely codes each training sample using the current dictionary estimate (using OMP or similar), and then solves for the dictionary estimate. Given sparse codes, the updating of each dictionary element can be solved directly via SVD decomposition, details of which are explained in Section 7.4. MOD updates all dictionary elements simultaneously, while K-SVD updates one dictionary element at a time. By updating one element at a time, K-SVD is also able to update the sparse coefficients that use that dictionary element, making the method converge faster than MOD. The K-SVD algorithm becomes computationally burdening as the number of samples or the sample dimension increases. Rubinstein [132] implemented an efficient implementation of K-SVD using Batch Orthogonal Matching Pursuit. They show that the SVD decomposition only converges to a local minimum, and thus can substitute it with and n OMP projections, where n is the number of training samples.

Tuning of the dictionary is dependent on both the number of atomic elements in Φ as well as the target non-zero coefficients for $\hat{\boldsymbol{a}}$. Figure 5.3 shows the RMSE reconstruction error for a sample dataset,

varying both the sparsity of $\hat{\boldsymbol{a}}$ as well as the dictionary size of Φ. The dataset contained 1072 faces of varying expression; each face was 26x20 pixels. It is not uncommon to use 5-10 non-zero coefficients for $\hat{\boldsymbol{a}}$, so for the data in Figure 4, a dictionary of >200 samples is necessitated.

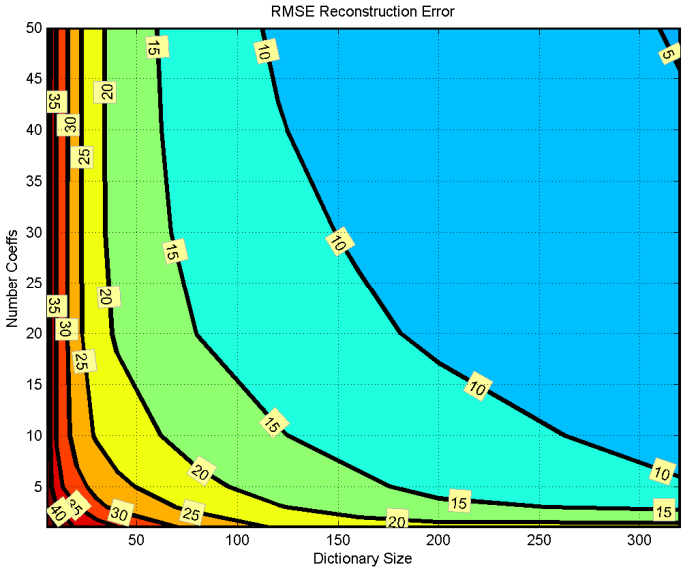


Fig. 5.3. RMSE reconstruction error of a sample dictionary using the Batch Orthogonal Matching Pursuit approximation of the K-SVD algorithm for dictionary selection. Lower RMSE is achieved by using more coefficients and larger dictionary size.

6 Applications of Sparse Representations

6.1 Manifold Based Sparse Representation Introduction

Although the SR framework was designed for reconstruction purposes, it has successfully been adopted for classification problems [79]. Wright *et al.* [78, 129] populated a dictionary with all n training samples, then passed the $\boldsymbol{a}$ coefficients of test samples into a minimum reconstruction error classifier to achieve state-of-the-art results. In this framework, the dominant signal always prevails, but it could produce some unintended effects. For example, when trying to extract facial identity, pose variation may contaminate or even dominate the sparse coefficients. Coefficient contamination is unfortunate yet important, as it has been shown that images of a single person under multiple poses exhibit greater variation than images of different people at a single pose [10]. Had the dataset used in [78, 129] contained faces with varying pose and facial expression, recognition results would have suffered.

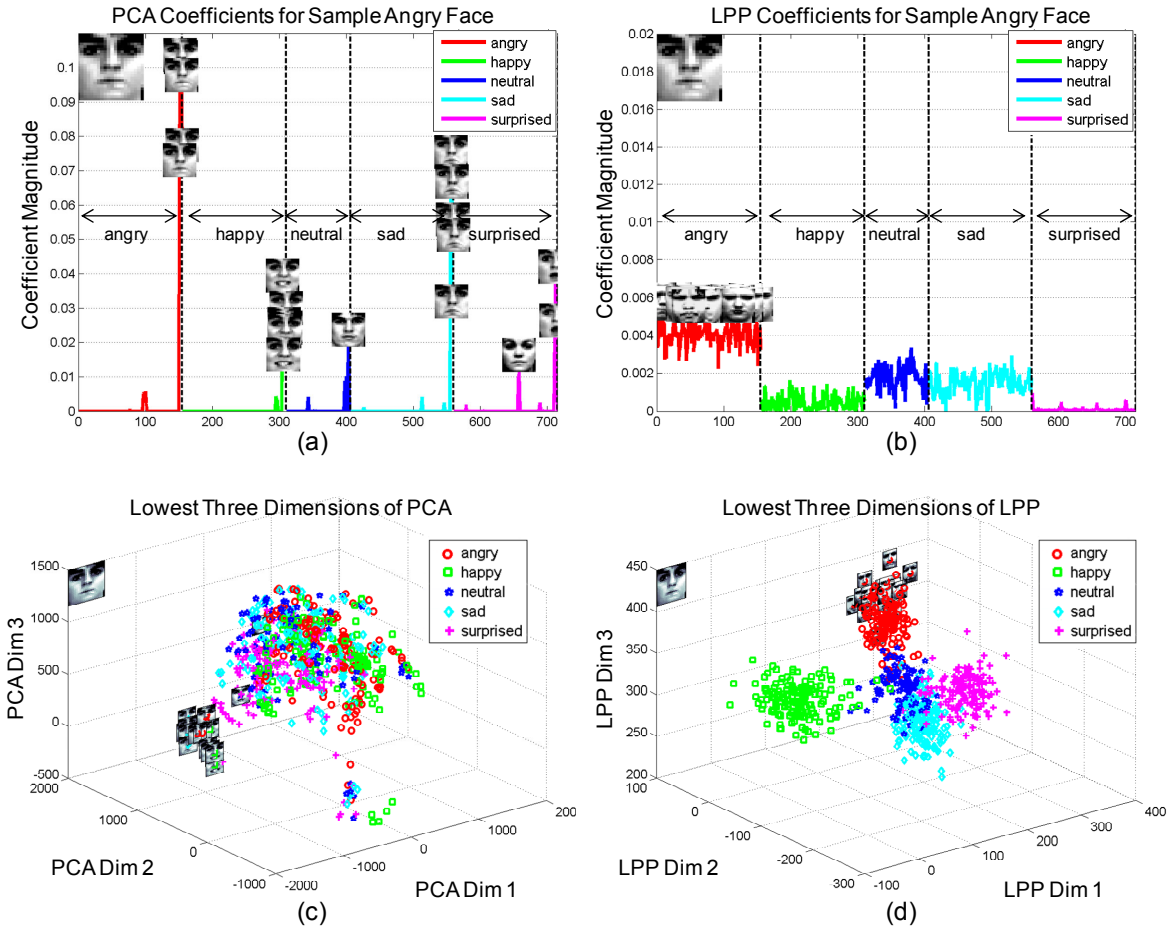


Fig. 6.1 Sample decomposition of an angry face using PCA on left and SLPP on right.

For example, Fig. 6.1 shows the spare coefficients from a test angry face using PCA (on left) and SLPP (on right). The top of Fig. 6.1 overlays pictures on top of the top 20 $\boldsymbol{a}$ coefficients. Those same 20 images are shown on the bottom, which depicts the most important three dimensions of the PCA and SLPP spaces. The dictionary is made up of approximately 700 faces, grouped from left to right with the expressions angry, happy, neutral, sad, and surprised. The PCA coefficients appear to be made up of several expressions, while the SLPP appears to cluster nicely with only angry faces. PCA is clustering both by expression and subject identity. As evidence of such, the peaks in the happy, neutral, sad, and surprised categories are mostly from the same identity as the test face. Fig. 6.2 demonstrates the same phenomenon, this time with a happy face.

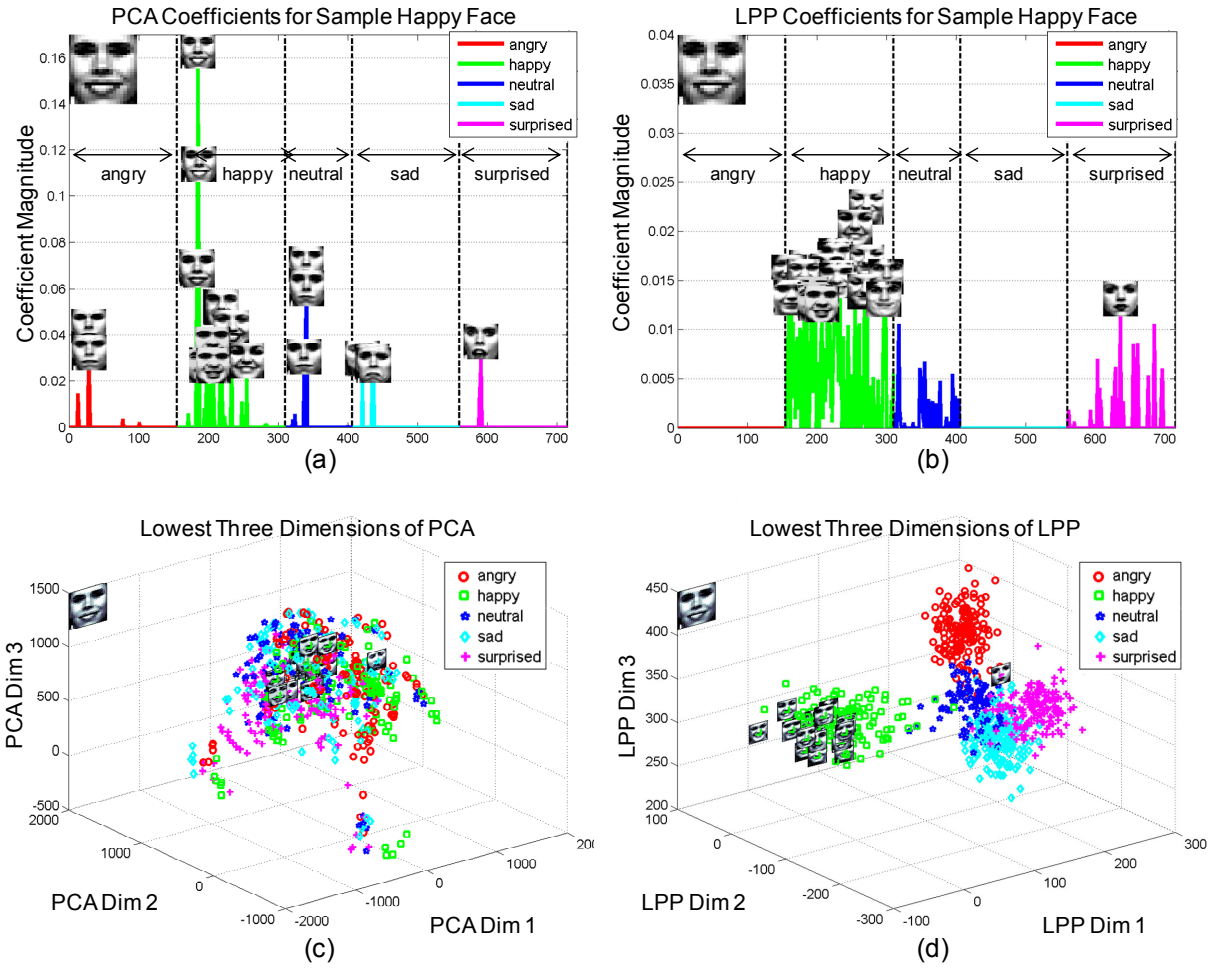


Fig. 6.2 Sample decomposition of a happy face using PCA on left and SLPP on right.

To make the coefficient learning computationally tractable Wright *et al.* [78, 129] used random projections to reduce the dimensionality of the data. Tzimiropoulos *et al.* [133] and Zafeiriou and Petrou

[77] used Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) respectively, along with SR techniques based on [78] to demonstrate further computational and accuracy improvements. The work in [77] experienced the adverse effects of coefficient contamination, and noted that applying Wright's framework is not a straightforward process because the facial identity of the person is often confused with facial expression. Ptucha *et al*. [63, 79, 134] addressed the coefficient contamination problem by preprocessing the data with supervised manifold learning. Similar to subspace clustering [135], supervision in manifold learning encourages clustering of sample images in accordance with their classification labels. Further, by training local feature regions defined across the face, the ability to resolve difficult expressions is enhanced. By introducing a "Don't Know" classification along with statistical mixture models, tolerance to mouth, eye, or nose occlusion is increased dramatically.

The steps of the expression recognition approach used in this section are shown in Figure 6.3. After face detection, eye and mouth corner points define an affine mapping to a normalized face size. Localized facial regions are extracted, processed, and then mapped onto a low dimensional manifold surface learned by semi-supervised LPP (SLPP). Sparse representations encode the test face regions projected onto the manifold as a nonnegative linear combination of dictionary elements.

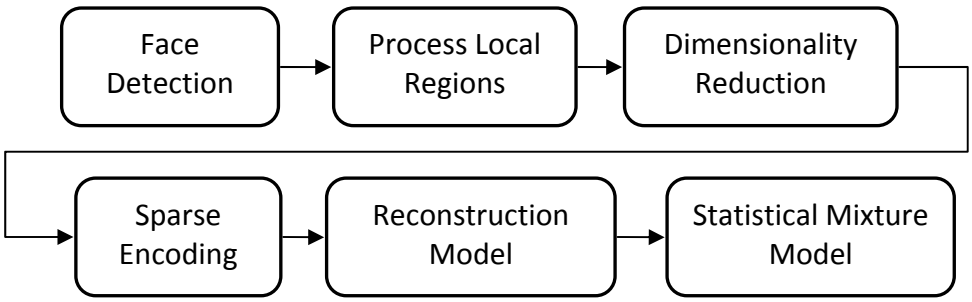


Fig. 6.3. Flowchart of expression recognition approach.

A reconstruction model compares the fully reconstructed facial region, using sparse coefficients from all classes, to the reconstructed facial region using only coefficients from each respective class. The class with the smallest reconstruction error to the fully reconstructed facial region is indicative of the facial expression. This approach is termed MSR, for Manifold based Sparse Representation. By examining several facial features simultaneously, statistical mixture models determine final expression classification. If a test face is occluded, for example if the face is covered with a scarf, the mouth region triggers a "Don't Know" classification, and that part of the face is excluded from facial classification.

6.2 Manifold Based Sparse Representation Method

Eye and mouth corner points extracted from the frame of interest enable precise facial localization for

both test and training images. These 6 points are projection mapped to a canonical reference face of 60x51 pixels, with left eye coordinates of {(12,21),(20,21)}, right eye coordinates of {(34,21),(42,21)}, and mouth coordinates of {(17,45),(36,45)}. Figure 6.4 shows a sample face before and after this projection mapping. Because each face has unique eye and mouth corner points, the projection matrix is solved on an image by image basis. The projection matrix used for Figure 6.4 is:

$$\begin{bmatrix} 0.8869 & 0.0397 & 0.0014 \\ -0.1984 & 0.5171 & -0.0056 \\ -3.7488 & 0.1577 & 0.9909 \end{bmatrix}$$

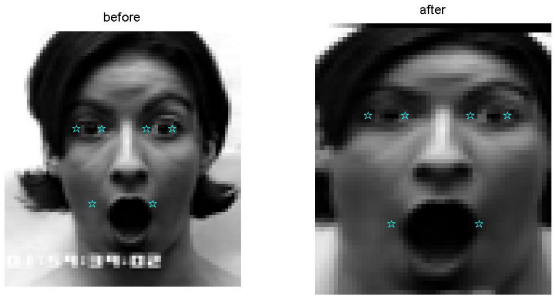


Fig. 6.4. Image on left is a zoomed out face, centered on the 60x51 mask area before the projection transform. Image on the right is the 60x51 masked area after the projection transform. The cyan fiducials show the desired locations of the eye and mouth corner points.

Several variants of pixel processing were evaluated as part of this research. The five methods that are used in the model that follows include luminance, edge magnitude, edge phase, LBP, and Gabor. Figure 6.5 shows each on a sample face from the CK+ dataset. (See Appendix-II for details.)

Fig. 6.5. Sample CK+ face exhibiting the five pixel processing variants used in this proposal. From left to right are luminance, edge magnitude, edge phase, LBP, and Gabor.

SLPP dimensionality reduction followed pixel processing. In most cases, this reduced the dimensionality from 51x60 $\in \mathbf{R}^{3060}$ to $\in \mathbf{R}^{5}$. This step not only closely clustered like expressions together, but, increased processing speed by over 10x factor. With each face represented by a 5 dimensional vector, sparse signal representation was used to enable parsimonious representations of each test face.

The training faces served as the dictionary, Φ, from which to select. Sparse representations represented each new test face as a linear combination of training faces from Φ. This sparse vector of coefficients is denoted as $\hat{\boldsymbol{a}}$.

The robustness of (5.6) is directly tied to the discriminative nature of the $\hat{\boldsymbol{a}}$ coefficients. The α parameter from (3.26) of SLPP is a driving factor. The closer α is to 1.0, the stronger the discriminant nature between classes. To illustrate this effect, Figure 6.6 shows two sample faces from the CK+ dataset. Each row shows the $\hat{\boldsymbol{a}}$ coefficients for PCA, LPP, SLPP, and LDA LPP. As α is increased, we observe the tightening of the $\hat{\boldsymbol{a}}$ coefficients to their respective class categories. This tightening minimizes coefficient contamination. As such, SR classification methods benefit from a previous subspace clustering step. Too much tightening (e.g. α =1.0) is likely to cause failure on natural datasets with limited dictionary Φ size.

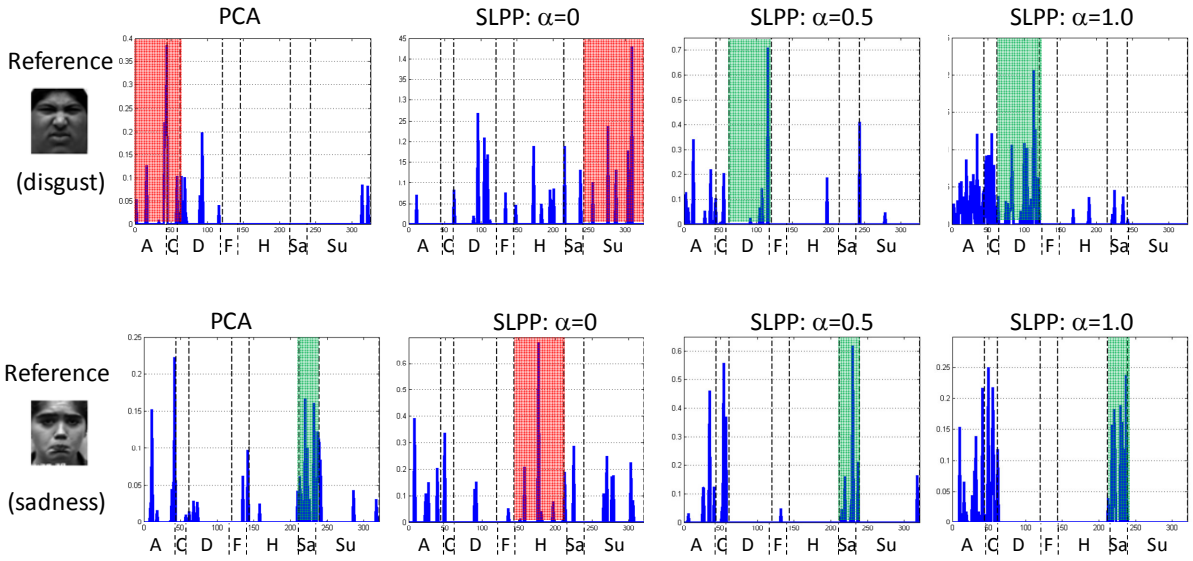


Fig. 6.6. Sample faces (disgust in top row and sadness in bottom row) along with their top $\hat{a}$ coefficients using PCA dimensionality reduction, LPP, SLPP, and LDA LPP. Coefficients are grouped by CK+ categories of Anger (A), Contempt (C), Disgust (D), Fear (F), Happy (H), Sadness (Sa), and Surprise (Su). Shaded rectangle shows classification made via Equation (4.6)- red is incorrect, while green is correct.

Processing individual facial regions is motivated to improve classification accuracy and enable classification in the presence of occlusions. Referring to Figure 6.7, the left-most image shows each pixel of the face color coded by its ability to classify facial expression (if its nearest 11x11 neighbors were used for expression classification using our MSR technique). The mouth area, and specifically, the mouth corners, offer the most salient information.

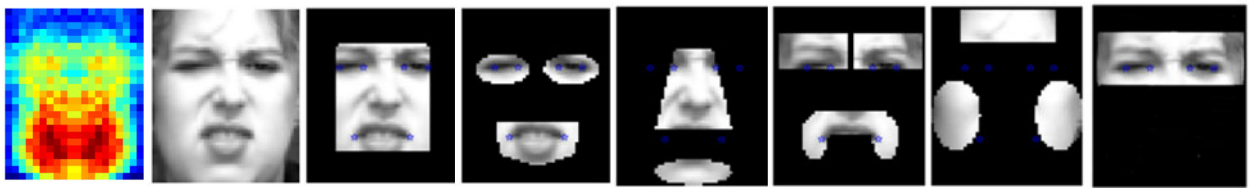


Fig. 6.7. Face importance map and eleven facial areas: Fullimg, face, eyes, mouth, nose, chin, eyebrows, mustache, forehead, cheeks, eyereg.

Referring again to Figure 6.7, the seven right-most figures show the eleven facial regions studied, many inspired by Kumar *et al.* [136]. Each masked region was independently trained to get eleven different SLPP manifold models and eleven sets of dictionary elements. The final classification includes predictions from multiple regions, where the following statistical mixture model determines the predicted class:

$$\hat{c} = \text{argmax}_{c=1\ldots z} \sum P_f c_f^* I[c = c^*] \tag{6.1}$$

where the summation is done over all included face regions, each region predicting one of the z discrete classifications. P_f is the prior probability of accuracy for the particular face region on the training set (see Tables 6.4 and 6.5 for example P_f values), c_f* is the predicted class from the particular face region, and I[c=c*] is the indicator function that returns 1 when c=c*, otherwise 0. As such, (6.1) allows each facial region a weighted vote for a particular class, $c_1..c_z$, and the test face is assigned to the class with the most weighted votes.

A feature harvesting process is employed for local feature selection, as it is entirely conceivable that different regions of the face may be better described by different types of pixel processing. Six types of pixel processing are demonstrated: luminance, edge intensity based upon the magnitude of the high pass image, edge direction based upon the phase of the high pass image, Local Binary Patterns (LBP) [56, 61], Gabor filters [60, 137], and Local Phase Quantization (LPQ) [69] pixels. For each region, mean normalization, $\boldsymbol{x}' = \boldsymbol{x}/\mu$, consistently gave superior results over other normalization techniques.

Because none of the training faces have occlusions, the SR framework with reconstruction error is ideally suited to detecting occlusions over portions of a face. If a test face is partially occluded, the reconstruction error will be large in the occluded regions. For example, it would be difficult to represent an occluded test sample mouth region (e.g. with a hand in front of it) using the training mouth regions, since none of the training samples have occluded mouth regions. The MSR method searches the eye and mouth regions first. If the reconstruction error is greater than κ, the algorithm omits those regions from the statistical merging in (6.1). Thus, if the mouth is occluded, the nose and eye regions can still be used for final classification. This instructs the algorithm to ignore occluded regions of the face and concentrate

on areas with small reconstruction errors. If no occlusions are detected, all face regions are eligible for usage in (6.1).

6.3 Manifold Based Sparse Representation Results

The classification results reported in this section use the Cohn-Kanade (CK) [111] dataset, the extended Cohn-Kanade (CK+) [58] dataset, a selection of portrayals from the Geneva Multimodal Emotion Portrayals (GEMEP) corpus [138] used for the Facial Expression Recognition and Analysis Challenge (FERA2011), referred to as the GEMEP-FERA [139] dataset, and the Labeled Faces in the Wild (LFW) [140] dataset. (See Appendix I for dataset details.)

For each face, a projection matrix maps eye and mouth corner points to a reference canonical position. The resulting images are cropped to 60x51 pixels in preparation for mask extraction. In all experimental results, the leave-one-subject-out cross validation methodology was used.

To mimic real-world occlusion problems, a set of over 250 eye or mouth occluded images were downloaded from the internet. Each image was affine mapped to the canonical 60x51 pixel image, where the occlusion was manually extracted in Photoshop to create an occlusion mask. In this fashion, any occlusion mask can be overlaid on top of any test image. Figure 6.8 shows a sample CK+ image with various eye and mouth occlusions.

Fig. 6.8. Example of eye and mouth occlusions. Original images on top, masks in middle, modified CK+ image on bottom.

6.3.1 Justification of Dimensionality Reduction and Sparse Representation Methods

To demonstrate the capabilities and limitations of the MSR technique and understand its parameter selection, the 7 class CK+ expression dataset, 2 class LFW gender dataset, 5 class LFW race dataset, and 10 class LFW gender-race datasets are contrasted. The 10 class race-gender LFW dataset assigns each face as one of the 5 races × 2 gender types described in the dataset section. The CK+ results are leave-

one-subject-out and the LFW results use the training set for training and (the non-overlapping) test set for testing. Table 6.1 shows the classification accuracy of the four ℓ^1 classification strategies discussed in Section 5.2 under various dimensionality reduction techniques. Dimensionality reduction techniques included no dimensionality reduction (No DR), PCA, Gaussian kernel LPP (LPP), SLPP, α=0.5, and LDA LPP. Non-negative ℓ^1 SR coefficients were used as input to the classification method. Extended faces, luminance processing, with mean normalization are used. The $\mathbf{R}^d$ column is the number of dimensions after dimensionality reduction, and time column is the total time for processing (lower numbers are better).

Table 6.1 illustrates the beneficial performance of the minimum reconstruction error [78] for facial classification (labeled as RE in Table 6.1). With regards to the dimensionality reduction methods, the No DR case skips the dimensionality reduction step, passing all 60×51 pixels into each of the four SR models. For the PCA case, the eigenvectors corresponding to 99% of the variance were kept. For the LPP cases, the top 30 dimensions were used. The addition of supervision to the LPP framework via (3.26) minimizes coefficient contamination by forcing samples to be reconstructed from dictionary elements of identical class. This minimizes the reconstruction error for that class, enabling dramatic classification accuracy improvements. The lower dimensions resulting from LPP speed up the resulting classification dramatically. The time column in Table 6.1 is the total time in seconds to process the entire dataset, using appropriate cross-validation, on a quad-core I-7 computer inside the Matlab programming environment. The MP %Acc, nonZ %Acc, Energy %Acc, and RE %Acc columns represent the SR classification accuracy using the methods of Max Peak , Most nonzero, Energy ($\Sigma \hat{a}$), and reconstruction error as defined in Section 5.2.

TABLE 6.1A. SEVEN CLASS EXPRESSION ACCURACY ON THE CK+ DATASET. COMPARISON OF DIMENSIONALITY REDUCTION METHOD VS. SR COEFFICIENT CLASSIFICATION METHOD.

Method	$\mathbf{R}^d$	Time	MP %Acc	nonZ %Acc	Energy %Acc	RE %Acc
No DR	3060	643	54.9	49.1	62.2	62.0
PCA	199	1180	58.5	70.1	73.7	69.2
LPP	6	192	21.7	21.1	22.9	23.7
SLPP,α=0.95	6	211	62.5	72.5	79.4	84.1
LDA LPP	6	192	65.2	82.6	81.2	**85.2**

TABLE 6.1B. TWO CLASS GENDER ACCURACY ON THE LFW DATASET. COMPARISON OF DIMENSIONALITY REDUCTION METHOD VS. SR COEFFICIENT CLASSIFICATION METHOD.

Method	$\mathbf{R}^d$	Time	MP %Acc	nonZ %Acc	Energy %Acc	RE %Acc
No DR	3060	760	76.7	77.3	79.9	80.2
PCA	411	92	75.6	77.5	79.8	82.8

LPP	30	29	62.2	76.4	76.0	74.9
SLPP,α=0.95	30	31	80.8	84.9	85.9	**88.9**
LDA LPP	1	29	51.9	51.9	51.9	51.9

TABLE 6.1C. FIVE CLASS RACE ACCURACY ON THE LFW DATASET. COMPARISON OF DIMENSIONALITY REDUCTION METHOD VS. SR COEFFICIENT CLASSIFICATION METHOD.

Method	**R^d**	**Time**	**MP %Acc**	**nonZ %Acc**	**Energy %Acc**	**RE %Acc**
No DR	3060	700	53.9	81.0	81.8	81.8
PCA	405	80	67.1	81.8	83.4	83.7
LPP	30	33	50.8	80.7	78.7	77.9
SLPP,α=0.95	30	34	73.0	82.9	83.7	**86.0**
LDA LPP	4	29	32.2	47.6	43.7	40.7

TABLE 6.1D. TEN CLASS RACE-GENDER ACCURACY ON THE LFW DATASET. COMPARISON OF DIMENSIONALITY REDUCTION METHOD VS. SR COEFFICIENT CLASSIFICATION METHOD.

Method	**R^d**	**Time**	**MP %Acc**	**nonZ %Acc**	**Energy %Acc**	**RE %Acc**
No DR	3060	724	40.9	62.6	64.9	65.1
PCA	405	80	50.4	63.4	66.7	69.1
LPP	30	33	29.7	59.2	55.9	53.4
SLPP,α=0.95	30	33	56.4	71.1	72.9	**76.1**
LDA LPP	9	30	44.5	64.1	60.9	55.4

Continuing with the same four datasets, Table 6.2 compares multiclass linear SVM to four SR methods (LARS ℓ^1, NMF LARS ℓ^1, regularized ℓ^1, NMF regularized ℓ^1). The LARS methods are based upon SparseLab 2.1 (http://sparselab.stanford.edu/), and the regularized methods are based upon SLEP 4.0 (http://www.public.asu.edu/~jye02/Software/SLEP/) toolkits. (See Appendix IV for more information software toolkits used.) All tests were done with SLPP (α=0.95) dimensionality reduction. The SR methods used the minimum reconstruction error model to assign the final class. Regularized ℓ^1, although approximately 25% slower than LARS, is 13% more accurate. The NMF regularized ℓ^1 techniques are convincingly better for all categories with the exception for the 2 class gender LFW dataset.

TABLE 6.2. SEVEN CLASS EXPRESSION ACCURACY ON THE CK+ DATASET USING LINEAR SVM VS. SR METHODS.

	Linear SVM	**LARS ℓ^1 SR**	**NMF LARS ℓ^1 NMF SR**	**regularized ℓ^1 SR**	**NMF regularized ℓ^1 SR**
7 class CK+	82.5	79.4	82.8	79.5	**84.1**
2 class gender LFW	**89.5**	86.7	85.2	88.0	88.9
5 class race LFW	84.6	81.0	64.6	84.9	**86.2**
10 class race-gender LFW	75.7	65.7	43.4	75.2	**76.3**

The accuracy of the SR methods is strongly influenced by the regularization parameter λ, and the adjacency matrix $\boldsymbol{W}$ blend parameter, α. Table 6.3 continues with the same four datasets, varying λ from 0 (no regularization) to 1 (full regularization), and varying α from 0 (full heat kernel or only local topology) to 1 (full supervised LDA adjacency matrix). The blended adjacency matrix method is superior to the LDA or Gaussian adjacency matrix for the three natural LFW datasets, but not the CK+ posed expression dataset. For the CK+ dataset, large differences between exaggerated expressions are linearly separable using a linear LDA method and the local topology offered by the Gaussian kernel offers little value. As long as some amount of the LDA kernel is utilized, results are surprisingly flat. Further, because the CK+ dataset has exaggerated sample to sample differences, the regularization parameter λ has little effect. For the LFW datasets, too much regularization generally prevents important $\hat{\boldsymbol{a}}$ coefficients from being passed into the reconstruction model classifier, while too little regularization overfits to the training set.

Because posed datasets with large separations between classes are tolerant to large changes in λ and α, we can concentrate on natural datasets such as LFW. Experimentation has found that constraining $0.05 \leq \lambda \leq 0.25$, and $0.25 \leq \alpha \leq 0.95$ is generally sufficient to meet a cross spectrum of classification needs. Tougher classification problems require less regularization (which essentially oversmoothes the model), and more local topology (lower α). Further, when k, the number of classes is high, the rank of W is high, and the benefit of using the Gaussian heat kernel diminishes.

TABLE 6.3A. SEVEN CLASS EXPRESSION ACCURACY ON THE CK+ DATASET. COMPARISON OF SLPP BLEND PARAMETER α TO REGULARIZATION PARAMETER λ.

		CK+ Expression, NMF=1													
		<-- Less Regularization					λ				More Regularization -->				
	α \ λ	0.0000	0.0001	0.0010	0.0050	0.0250	0.0500	0.0750	0.1000	0.2500	0.5000	0.9000	0.9500	0.9900	1.0000
α (Heat) --> <-- (LDA)	0.000	23.3	23.1	22.8	22.6	23.1	23.2	23.6	23.7	24.3	23.4	22.3	22.3	21.7	22.9
	0.010	**86.1**	**86.1**	**86.1**	**86.1**	**86.1**	**86.1**	**86.1**	85.9	85.1	84.9	84.3	84.3	84.3	84.3
	0.100	83.8	83.8	83.8	84.6	84.8	85.6	84.8	84.6	84.1	85.2	82.2	82.2	82.2	81.4
	0.250	84.1	84.1	84.1	84.1	84.1	84.1	84.1	84.1	84.1	84.1	82.9	83.2	83.2	83.2
	0.500	84.8	84.8	84.8	84.8	84.5	84.8	84.8	84.8	83.6	82.9	82.3	82.3	81.8	81.2
	0.750	83.5	83.5	82.9	82.9	83.3	84.1	84.3	84.6	84.6	84.6	85.1	85.1	85.1	85.1
	0.850	84.4	85.0	84.4	85.0	85.0	85.3	85.0	85.0	85.0	83.4	83.2	83.2	83.2	83.2
	0.900	84.6	85.4	84.6	85.4	84.9	85.4	85.4	84.9	85.4	84.3	81.9	81.9	81.9	81.9
	0.950	84.6	84.6	84.6	84.6	84.6	84.6	84.6	84.1	84.6	85.7	84.4	84.4	84.4	84.4
	0.990	83.7	83.7	83.7	83.7	83.7	83.7	83.7	84.0	83.6	85.3	84.5	85.0	85.0	85.0
	0.999	82.3	82.1	82.9	82.9	82.9	82.9	82.9	82.4	83.4	84.5	84.4	85.0	85.0	85.0
	1.000	83.5	83.5	83.5	83.5	84.3	85.2	85.2	85.2	84.4	83.3	81.3	79.7	79.7	79.7

TABLE 6.3B. TWO CLASS GENDER ACCURACY ON THE LFW DATASET. COMPARISON OF SLPP BLEND PARAMETER α TO REGULARIZATION PARAMETER λ.

		LFW, gender, NMF=1													
		<-- Less Regularization					λ							More Regularization -->	
	α \ λ	0.0000	0.0001	0.0010	0.0050	0.0250	0.0500	0.0750	0.1000	0.2500	0.5000	0.9000	0.9500	0.9900	1.0000
α (Heat) --> <-- (LDA)	0.000	76.5	76.5	76.5	76.5	76.4	76.0	75.5	74.9	73.8	66.5	47.4	47.0	45.0	44.8
	0.010	86.6	86.7	86.8	86.6	85.9	84.9	84.2	83.4	77.2	68.6	56.1	54.5	53.1	52.9
	0.100	88.2	88.1	88.2	88.1	88.5	89.1	89.9	**90.1**	88.1	82.4	72.0	70.4	68.8	68.7
	0.250	88.3	88.4	88.4	88.5	88.8	89.4	89.7	89.8	88.9	84.6	69.4	67.2	65.0	64.3
	0.500	87.2	87.1	87.2	87.4	87.9	88.2	88.5	89.0	90.0	87.7	73.1	70.1	68.3	67.6
	0.750	86.5	86.5	86.6	86.5	87.3	87.7	87.9	88.5	89.9	86.6	71.8	69.2	67.0	66.9
	0.850	86.4	86.3	86.5	86.4	87.2	87.6	87.6	88.4	**90.1**	86.7	72.1	68.8	66.8	66.3
	0.900	86.0	86.1	86.0	86.0	86.9	88.1	88.6	88.8	89.1	86.7	71.4	69.8	66.9	65.9
	0.950	85.6	85.6	85.6	85.9	87.6	88.0	88.2	88.9	89.6	86.9	72.8	70.0	67.7	67.0
	0.990	86.2	86.2	86.4	86.2	87.2	87.7	88.2	88.9	89.5	87.2	73.0	69.2	66.4	65.8
	0.999	86.3	86.4	86.3	86.4	87.4	87.7	88.2	89.1	89.5	87.4	72.9	68.8	66.7	66.1
	1.000	62.5	62.5	62.5	61.8	59.3	56.9	54.6	51.9	36.3	27.5	24.2	23.8	23.8	23.8

TABLE 6.3C. FIVE CLASS RACE ACCURACY ON THE LFW DATASET. COMPARISON OF SLPP BLEND PARAMETER α TO REGULARIZATION PARAMETER λ.

		LFW, race NMF=1													
		<-- Less Regularization					λ							More Regularization -->	
	α \ λ	0.0000	0.0001	0.0010	0.0050	0.0250	0.0500	0.0750	0.1000	0.2500	0.5000	0.9000	0.9500	0.9900	1.0000
α (Heat) --> <-- (LDA)	0.000	81.3	81.3	81.3	81.3	80.7	79.3	78.5	77.9	69.6	56.4	29.9	27.0	25.1	24.5
	0.010	85.4	85.3	85.7	85.3	84.8	83.9	83.2	82.2	74.6	61.0	44.3	42.7	41.0	40.6
	0.100	86.5	86.6	86.4	86.4	86.0	85.1	84.8	83.7	77.2	64.6	41.6	39.3	36.9	36.8
	0.250	85.9	85.8	86.1	85.9	86.2	86.1	85.9	85.4	84.0	77.7	55.0	52.6	50.5	49.8
	0.500	86.1	85.8	86.0	86.1	86.7	87.3	87.4	**87.5**	84.9	77.1	54.9	51.6	49.6	49.1
	0.750	86.0	86.0	86.0	86.0	86.5	86.9	86.9	86.9	85.3	76.0	53.1	48.6	45.5	44.9
	0.850	85.7	85.7	85.5	85.4	85.9	86.4	86.5	86.4	85.1	77.3	51.9	48.7	46.4	45.5
	0.900	85.7	85.7	85.7	85.7	85.8	86.1	86.2	86.0	86.0	76.9	54.4	50.7	47.3	46.4
	0.950	85.5	85.5	85.5	85.7	85.9	86.3	86.0	86.0	84.9	76.8	53.1	50.0	47.4	46.8
	0.990	85.8	85.8	85.8	85.8	86.0	86.2	86.3	86.2	85.4	77.0	53.4	49.1	46.0	45.3
	0.999	85.8	85.8	85.8	85.8	86.0	86.1	86.4	86.4	85.5	76.9	52.7	48.9	46.1	45.5
	1.000	77.2	76.7	69.2	60.4	52.1	46.8	43.3	40.7	25.5	15.1	11.6	11.3	11.2	11.2

TABLE 6.3D. TEN CLASS RACE-GENDER ACCURACY ON THE LFW DATASET. COMPARISON OF SLPP BLEND PARAMETER α TO REGULARIZATION PARAMETER λ.

		LFW, race-gender, NMF=1													
		<-- Less Regularization					λ							More Regularization -->	
	α \ λ	0.0000	0.0001	0.0010	0.0050	0.0250	0.0500	0.0750	0.1000	0.2500	0.5000	0.9000	0.9500	0.9900	1.0000
α (Heat) --> <-- (LDA)	0.000	61.0	61.0	61.0	60.9	59.1	56.6	54.8	53.4	46.7	36.4	17.7	16.8	15.4	15.0
	0.010	77.5	77.3	77.1	77.3	77.0	76.0	75.5	75.1	70.4	61.6	43.9	42.5	41.1	40.8
	0.100	74.7	74.8	74.8	74.6	75.3	75.7	75.5	76.2	75.1	68.7	54.1	51.6	50.3	50.1
	0.250	75.9	76.0	75.9	76.0	76.2	77.0	**78.1**	77.5	72.9	62.5	45.2	42.2	40.6	39.9
	0.500	75.8	75.6	75.6	75.9	76.6	76.6	76.8	76.6	73.2	64.2	45.6	42.8	41.8	41.0
	0.750	75.1	75.4	75.2	75.3	75.4	76.0	76.1	76.8	73.9	64.0	43.9	40.8	38.3	38.0
	0.850	75.4	75.2	75.3	75.6	75.5	75.8	76.0	76.4	73.8	64.5	43.7	40.7	39.1	38.9
	0.900	75.2	75.3	75.5	75.6	75.5	75.6	76.3	76.2	73.6	64.3	43.7	40.8	39.4	38.8
	0.950	74.8	74.8	74.9	75.3	75.6	76.1	76.4	76.1	73.8	65.3	43.8	41.1	39.1	38.9
	0.990	75.2	75.1	75.3	75.3	75.5	75.8	76.3	76.3	73.6	63.3	45.1	42.5	40.8	39.6
	0.999	75.3	75.1	75.2	75.6	75.4	75.7	76.4	76.4	74.1	63.7	45.2	42.3	40.3	40.1
	1.000	73.9	73.9	72.8	71.0	64.3	60.4	57.5	55.4	43.3	27.9	17.3	16.0	15.0	15.0

The MSR research has consistently found:

- The improved performance of the reconstruction error classifier over the max peak, most non-zero, and energy classifiers was found in just about every test case.
- The improved performance of the nonnegative (ℓ^1 NMF) SR method over ℓ^1 SR is consistent with our previous classification studies that ultimately use the sparse coefficients for classification purposes.
- The single most important factor towards reducing coefficient contamination is to use some form of subspace clustering. We have found that supervised methods such as LDA and SLPP outperform unsupervised methods only because unsupervised methods indiscriminately cluster by similarities that often differ from the attribute that is being classified. We have tried many supervised and unsupervised dimensionality reduction techniques, and SLPP is consistently one of the best.
- The improved performance of the nonnegative SR method over the linear SVM classification justifies the inclusion of SR for facial classification problems. While the combination of manifold learning and sparse representations works best for this set of facial understanding problems, we need to explore their efficacy in other domains.

In summary, the combination of Manifold learning with Sparse Representations (MSR) has been found to yield the excellent facial understanding classification accuracy at a low computational cost. Specifically, the MSR technique takes normalized images, passed through SLPP, followed by nonnegative SR, and finally classified using the minimum reconstruction error.

6.3.2. Choice of Pixel Processing and Facial Parts Selection

Table 6.4 displays various types of pixel processing over eleven facial regions for the CK+ dataset with no occlusions. The six types of pixel processing include luminance image, edge magnitude, edge phase, $LBP_{8,1}^{u2}$ pixels, Gabor filters, and LPQ. (See Appendix II for a discussion on pixel processing techniques.) For each region and pixel processing type, normalized pixels are passed into SLPP dimensionality reduction, where ℓ^1 coefficients are classified according to the reconstruction model using (5.6). The classification accuracies in Table 6.4 are used as the P_f values for (6.1).

The formation of facial expressions involves multiple areas of the face simultaneously. As such, the top features for our Manifold based Sparse Representation (MSR) method are the extended face (extface) and face regions. Surprisingly, for the CK dataset, using only the 'mouth' and 'mustache' regions sacrifice only 10-15 percentage points of accuracy. This is followed by 'nose' and 'eyereg' which sacrifice 20 and 30 percentage points respectively. The forehead, eyes, eyebrows, chin, and cheeks regions were found to be less valuable.

TABLE 6.4. CK+ DATASET CLASSIFICATION ACCURACY FOR FACIAL REGIONS ACROSS SIX DIFFERENT PIXEL PROCESSING TECHNIQUES.

	 extface	 face	 nose	 must.	 mouth	 chin	 cheeks	 eyes	 eyebrow	 eye reg	 forehead
Lum	83.8	85.8	63.3	**76.6**	**81.0**	43.2	53.2	58.0	54.3	56.8	30.8
Mag	86.6	84.3	72.6	70.3	69.0	30.6	53.2	54.3	45.6	56.8	35.9
Phase	**88.0**	81.2	64.4	66.0	64.3	38.0	47.7	43.2	45.1	53.2	23.7
LBP	85.4	80.2	58.1	55.5	60.0	26.1	45.2	44.1	47.9	48.4	22.1
Gabor	83.6	**88.0**	**70.7**	**76.6**	79.7	**44.1**	**56.8**	**62.9**	**57.6**	**64.8**	**33.9**
LPQ	84.0	76.7	61.3	61.8	67.6	38.5	42.7	43.4	48.5	57.7	28.4

With regards to pixel processing, Gabor processing slightly outperformed luminance and edge magnitude. The edge phase and LBP was 8 and 13 percentage points behind Gabor processing on average. Different regions and pixel processing can be combined for increased accuracy. For example, if the top five feature regions are selected from Table 6.4, Eq. (6.1) improves the overall classification accuracy to 91.4%.

Table 6.5 contrasts the results from the CK+ posed dataset in Table 6.4 with the GEMEP-FERA natural dataset. Table 6.5 was created by applying the MSR methods to the GEMEP-FERA training set, as no ground truth was available for the test set. A comparison of Table 6.4 to Table 6.5 (along with the CK+ and GEMEP-FERA images in Figure 6.9) show evidence towards the difference in facial region effectiveness for posed vs. natural datasets. Posed datasets emphasize the mouth region, while natural datasets emphasize the eye regions.

TABLE 6.5. MSR MODEL APPLIED TO ALL 8865 FRAMES OF THE GEMEP-FERA TRAINING DATASET. NUMBERS SHOWN ARE CLASSIFICATION ACCURACY SHOWN FOR SAMPLE FACIAL REGIONS ACROSS FIVE DIFFERENT PIXEL PROCESSING TECHNIQUES.

	 extface	 face	 nose	 must.	 mouth	 chin	 cheeks	 eyes	 eyebrow	 eye reg	 forehead
Lum	91.9	86.5	72.8	62.2	62.6	40.3	65.8	69.1	67.9	77.1	52.2
Mag	92.9	**89.0**	76.6	**68.7**	**65.5**	40.4	**70.7**	**81.5**	**85.4**	**90.4**	71.2
Phase	90.5	80.4	67.2	62.1	54.9	**45.3**	64.4	64.9	75.3	76.2	61.0
LBP	88.3	77.0	66.1	58.2	52.6	36.0	63.2	67.5	76.0	78.8	57.8
Gabor	**93.5**	87.9	**74.2**	61.7	61.0	44.2	67.4	66.9	74.2	80.0	**51.9**

6.3.3. Benchmarking MSR Performance for Facial Expression

The performance of the MSR technique is now compared to other works for the 6-class expression CK dataset, the 7-class expression CK+ dataset, and the 5-class emotion GEMEP-FERA dataset. Table 6.6

compares our MSR approach to two other recently published methods that use SRs for facial expression. The first two entries of Table 6.6 show MSR with the top 1 and 5 features respectively. Zafeiriou *et al*. [77] used neutral subtracted frames, PCA dimensionality reduction, nonnegative matrix factorization, and most nonzero classification method. Zhi *et al*. [141] used a nonnegative based SR without neutral frame subtraction using k-fold cross validation, which is less challenging and less realistic than leave-one-subject-out testing.

TABLE 6.6. COMPARISON OF EXPRESSION CLASSIFICATION PERFORMANCE OF THE MSR METHODS INTRODUCED IN THIS PAPER TO OTHER SR METHODS USING THE CK DATASET.

Method	Accuracy	Neutral Subtract	Dim	Cross Valid
MSR, 1 feature	92.0	N	6	Subj
MSR, 5 features	94.6	N	6	Subj
Zafeiriou [77]	81.0	Y	40	Subj
Zhi-35dim [141]	90.2	N	35	k-fold

Using Eq. (6.1), MSR can automatically detect occluded regions. Table 6.7 shows the expression classification performance of the MSR technique on the CK dataset in the presence of eye and mouth occlusions. In place of the real-world occlusion dataset used in this paper, [77] used black box rectangles for occlusions to the eye region, and as before used neutral frame subtraction.

TABLE 6.7. EXPRESSION CLASSIFICATION ACCURACY ON CK DATASET WITHOUT AND WITH EYE AND MOUTH OCCLUSIONS.

	MSR no occl.	MSR Mouth	MSR eye	Zafeiriou [77] eye
Accuracy	94.6	64.5	84.4	78.7

Table 6.8 compares leave-one-subject-out expression classification accuracy of our MSR method on the 7-class expression CK+ dataset to other recently published results. Lucey *et al*. [58] used neutral frame subtraction, AAM points, and face pixels along with a linear SVM classification model. Chew *et al*. [142] used normalized pixels from constrained local models along with linear SVM. Jain *et al*. [143] used AAM points, PCA, and linear SVM. The last two columns of Table 6.8 show the expression classification performance of the MSR technique on the CK+ dataset in the presence of simulated eye and mouth occlusions.

TABLE 6.8. EXPRESSION CLASSIFICATION ACCURACY ON CK+ DATASET WITHOUT AND WITH EYE AND MOUTH OCCLUSIONS.

	MSR no	Lucey [58] no	Chew [142]	Jain [143]	MSR mouth	MSR eye

	occl.	occl.	no occl.	no occl.	occl.	occl.
Accuracy	**91.4**	83.3	74.4	84.1	60.5	70.9

The MSR model was then applied to the GEMEP-FERA dataset. As this dataset is temporal in nature, the MSR model was applied to each frame of each test video sequence, and then majority voting determined final emotion classification for each video. Results returned from the FERA2011 organizers are done three ways: 1) Person dependent: subjects in test set are in the training set; 2) Person independent: subjects in test set are not in the training set; and 3) Overall.

As part of the FERA2011 challenge, 14 papers were submitted, including a baseline. The MSR method scored 98.5% on the person dependent, or 2nd place; 56.6% on the person independent test, or 10th place; and 73.5%, or 5th place overall. The poor performance on the person independent results is a common failure mode of sparse representations when there are insufficient training exemplars in Φ. With only 7 subjects in the training set, this is not surprising. The excellent performance on the person dependent results shows the power of SR methods with sufficiently populated dictionaries.

6.3.4. MSR Performance for Other Facial Attributes

To further test the robustness of the MSR technique to other types of facial understanding and classification problems, the facial attributes of gender, glasses, facial hair, and race were tested on the LFW dataset. Table 6.9 compares multi-class linear SVM and MSR classification accuracies with and without occlusions. Both the SVM and MSR results in Table 6.9 use SLPP dimensionality reduction. To ensure that both SVM and MSR were utilized to the best of their abilities, optimal values for α and λ values were independently solved and used for each. Radial basis function SVM's were tested, but offered no statistical improvement above linear SVM.

In general, the MSR technique compares favorably to SVM with or without occlusions. When occlusions are not present, MSR outperforms SVM consistently. When occlusions have a strong impact on a classification, such as mouth occlusions with facial hair detection, the MSR technique performs significantly better. This demonstrates the power of the minimum reconstruction error working in conjunction with the statistical inference model. When occlusions don't have a strong impact on classification, such as facial hair with eye occlusion, SVM and MSR are comparable. While one can introduce occlusion detection into the SVM model, it is not as simple or elegant as the sparse model. For example, none of the training subjects had their hands over their mouth. If we now take a test sample with a hand over their mouth, the minimum reconstruction error is very high for all classes, a clear indication of occlusion. In the SVM paradigm, we will get a class prediction and can get confidence

levels for each class. Unfortunately, there is no guarantee that confidence levels will be low for all classes in the presence of occlusions- in fact, we often map occluded samples far away from decision boundaries, which are falsely reported as high levels of confidence.

TABLE 6.9. GENDER, GLASSES, FACIAL HAIR, RACE, AND MIXED RACE-GENDER CLASSIFICATION ACCURACY ON LFW DATASET USING SVM AND THE MSR TECHNIQUE WITHOUT AND WITH EYE AND MOUTH OCCLUSIONS.

	SVM no occl.	MSR no occl.	SVM mouth occl.	MSR mouth occl.	SVM eye occl.	MSR eye occl.
Gender	89.6	**90.8**	89.8	**90.3**	80.5	**80.8**
Glasses	85.0	**87.9**	84.3	**85.0**	71.8	**79.6**
Hair	86.9	**87.7**	80.8	**85.6**	87.3	**87.4**
Race	85.1	**87.5**	**85.0**	84.3	78.7	**82.0**
Mixed	75.9	**78.5**	76.2	**76.6**	64.6	**66.5**
Avg.	84.5	**86.5**	83.2	**84.4**	76.6	**79.3**

6.3.5. Facial Expression Recognition on Posed vs. Natural Datasets

The unique region based pixel processing methods used by MSR allow for any size facial region of any location to be evaluated for its effectiveness in facial classification. Referring to Figure 6.9, the CK+ and GEMEP-FERA images show each area of the face color coded (if its nearest 11x11 neighbors were used for expression classification using our MSR technique on 60x51 face images) by its ability to classify facial expression. The mouth area, specifically the mouth corners, offers the most salient information for the CK+ dataset. The upper cheek and eye region offers the most salient information for the GEMEP-FERA dataset. The subjects in the GEMEP-FERA dataset were talking as they were exhibiting expressions. Talking not only varied the mouth position, but constrained the mouth from exhibiting exaggerated positions. As such, the mouth importance dropped significantly. Equally intriguing is the increase in upper cheek and eye classification capability. Concerning facial expression, by comparing Tables 6.4 and 6.5 along with Figure 6.9, we can unequivocally state that posed datasets favor the mouth region, while natural datasets favor the eye, eyebrow, and upper cheek regions. Similar findings have been noted in the facial expression literature [144].

The Gender, Glasses, Hair, and Race plots in Figure 6.9 show the importance of individual regions of the face on natural datasets with regards to their respective classification problems. Gender classification favors the eye and eyebrow regions. Glasses detection favors the nose, but not eye regions. Facial hair classification favors the mustache region, but not the chin region. The sides of the nose, the corners of the mouth, and the inner eye sockets play a prominent role in race classification.

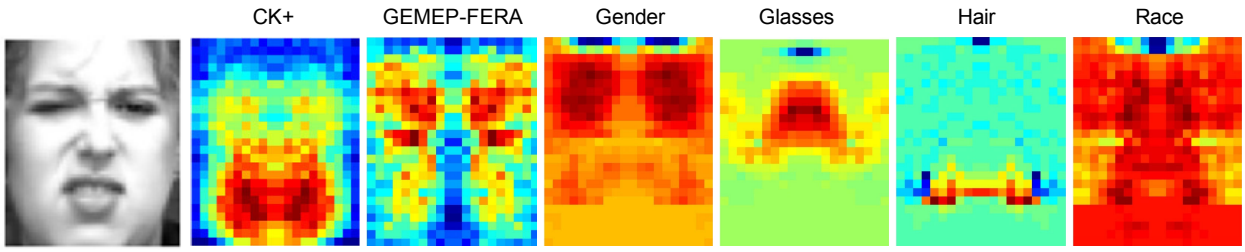


Fig. 6.9. Sample face and face importance demonstrating discriminative regions of the face for (left to right): expression on CK+ dataset; expression on the GEMEP-FERA dataset; Gender, Glasses, Facial Hair, and Race classification on the LFW dataset.

6.4 Temporal Facial Expression Sparse Representation Results

The SR results demonstrated in previous sections have been on static data. For temporal analysis, the GEMEP-FERA [139] dataset is once again used, but this time the optical flow based methods introduced in Section 2.5 will be contrasted. This GEMEP-FERA dataset is representative of a natural dataset- faces are of varying pose, expression, gender, race, facial hair, glasses, and occlusion.

ASM was used to automatically extract facial feature points, which define a warp to a canonical facial mask region. Optional dimensionality reduction was followed by SVM and MSR classification. Table 6.10 shows varying dimensionality reduction techniques for multi-class linear SVM and the SR techniques. The LPP LDA kernel and SLPP methods performed much better than no dimensionality reduction, PCA, or LPP Gaussian kernel. While SVM was quite tolerant to the blend parameter α, the SR method preferred α=1. This is because the SR training dictionary is extremely non-linear and a linear dimensionality reduced space prevents overfitting to the training data. The other benefit to using the three LPP methods over no dimensionality reduction or PCA is speed, as the LPP methods were consistently faster by 2-5X. If we allow the SR LPP-LDA method to use multiple facial regions as in [63], the accuracy increases to 74.5%.

TABLE 6.10. COMPARISON OF DIMENSIONALITY REDUCTION TECHNIQUES ON STATIC PROCESSING USING MULTI-CLASS LINEAR SVM AS WELL AS THE SR METHOD. NUMBERS ARE ACCURACY ON THE GEMEP-FERA TEST SET.

	None	PCA	LPP heat	LPP LDA	SLPP
SVM	64.9	64.2	58.2	70.9	**71.6**
SR	57.5	67.9	56.0	**71.6**	64.2

To compare the temporal techniques introduced in Section 2.5, Table 6.11 shows the ASM and MHI methods generally outperform the FFD method which generally outperforms the SIFT flow method. Temporal methods in the literature often utilize the knowledge of a neutral face. In this work, we have no neutral frames, perhaps putting the temporal methods at a disadvantage. For example, if the starting frame of the temporal window W_1^θ is a frown, upward movement of the corner of the mouth may be

classified as happiness, when in reality it would be indicative of neutral. For the ASM points, we have the canonical set of facial points from which each frame is warped via Procrustes analysis [67]. Using this canonical set of points as a neutral expression, we can form motion vectors between the raw ASM points after warping and the corresponding points on the canonical face, which we call Δ ASM vectors. The set of all Δ ASM vectors per W^{θ}_{l} is used as input features in Table 6.11.

TABLE 6.11. TEMPORAL TECHNIQUE CLASSIFICATION RESULTS. NUMBERS ARE PERCENT ACCURACY ON THE GEMEP-FERA TEST SET.

	ASM	MHI	FFD	SIFT flow
Accuracy	71.6	**73.1**	63.4	44.8

For the MHI results the [ΔX, ΔY] motion vectors from the single MHI template per window W^{θ}_{l} is used. For the FFD and SIFT flow results, the set of m [ΔX, ΔY] motion vectors from each $F_t(x,y)$ per W^{θ}_{l} is used as input features. The dense flow field used in MHI, FFD, and SIFT flow was evaluated on a mesh grid of resolution 30x26, 28x24, and 21x17 respectively.

The sliding window size θ affected performance significantly. Quick actions such as eye-blink favor small θ, while longer actions such as smile favor longer values of θ. Window sizes of θ={2,4,8,12,16,20} were implemented for each method. The results in Table 6.12 show the optimal window size θ for each method which was 20 frames for ASM and SIFT flow and 16 frames for MHI and FFD.

TABLE 6.12. COMPARISON OF WINDOW SIZE θ ON MHI CLASSIFICATION PERFORMANCE USING MAGNITUDE-PHASE AS WELL AS XY COORDINATES. NUMBERS ARE ACCURACY ON THE GEMEP-FERA TEST SET USING SLPP DIMENSIONALITY REDUCTION AND MULTI-CLASS LINEAR SVM.

Feature	Window size θ					
	2	4	8	12	16	20
Magphase	61.2	57.5	64.2	61.2	59.7	57.5
ΔX,ΔY	53.7	63.4	67.9	62.9	**73.1**	67.2

To gauge how well the above methods compare to state-of-the-art, the median submission in the 2011 FERA challenge reported 70% accuracy. Compared to the FERA challenge results, the SR static or MHI temporal methods score in 5th and 6th place respectively (out of 15 submissions). In an effort to improve the overall performance further, the static and temporal methods can be combined using Eq. (2.1). For example, if we combine the sparse static and MHI temporal methods, the overall accuracy increases to 79.1%, placing it 2nd in the FERA challenge.

The static methods perform best at happy and angry where characteristic mouth and eyebrow positions induced during these emotions enable accurate classification by humans and machine alike. The temporal methods perform best at relief and sadness where characteristically slow movements enabled accurate classifications. Fear was difficult for all methods, although it confused human viewers most often as well.

6.4 Temporal Gesture Recognition Using Active Difference Signatures

Figure 6.10 outlines the gesture extraction framework developed for this thesis. The input to the system is depth frames annotated with XY, and depth Z estimation for 20 kinematic joints of the human body [6]. The output of the system is an estimate of the gesture being performed. Gesture boundaries are detected, the XYZ coordinates of the skeletal joints are normalized, and the normalized joints are converted into an active difference signature attribute. For HCI displays, these active difference signature attributes are computed on every frame and are fed directly into a static gesture classifier. For pre-recorded datasets, the beginning and end of the gesture is used for dynamic time warping, creating a temporal joint attribute that is passed into the temporal gesture classifier. The classifier uses dimensionality reduction followed by a Sparse Representation Classifier (SRC).

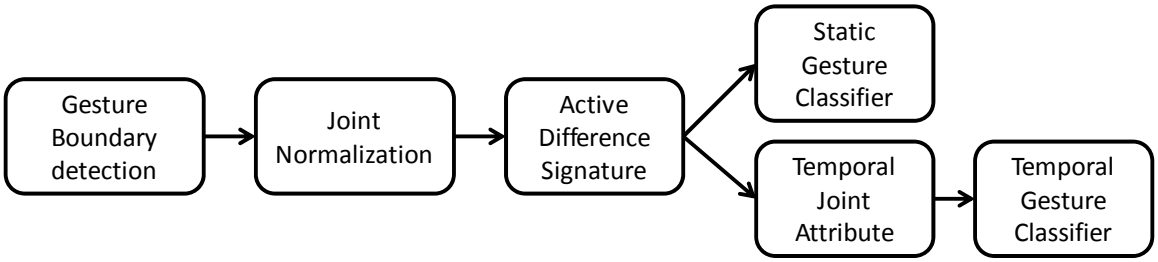


Figure 6.10. Overview of the active difference signature framework.

Gesture boundary detection finds frames which indicate the beginning and end of a gesture. Inspired by the ChaLearn Gesture Challenge one-shot-learning gesture dataset [145], this detection scheme skips over frames for which there is no information content worth analyzing. As soon as motion is detected in the scene, we activate the start of a gesture. If gestures are separated by a user returning to a resting position, motion detection along with a measure of the difference between the frame and the resting position are good markers for gesture boundaries. For example, Figure 6.11 shows a time sequence of a user executing four gestures in a single video frame from the ChaLearn Gesture Challenge.

The solid blue line in Figure 6.11 is an indicator of frame to frame motion. The dashed black line records the difference of the current frame to a resting frame. The gray regions pictorially illustrate the non-gesture regions, or the resting regions. The white regions are active regions where a gesture is being performed. The formation of the blue motion and black difference curves is done via a variation of MHI.

Specifically, depth frames are first resampled down to 60x80 pixels, and then they are converted to a difference frame using (2.2). This difference frame is subsequently morphologically filtered with an opening operation to remove noise, and all pixels greater than a threshold are tagged as difference pixels. The sum of the difference pixels forms the motion indicator. Higher values indicate more motion from one frame to the next. Any motion value greater than 0.05*max(*runningMotion*) is considered significant motion marking the beginning of a gesture, where *runningMotion* is the motion value for the past *f* frames in an HCI device, or all the frames in the video sequence if using a test or training sample from a dataset.

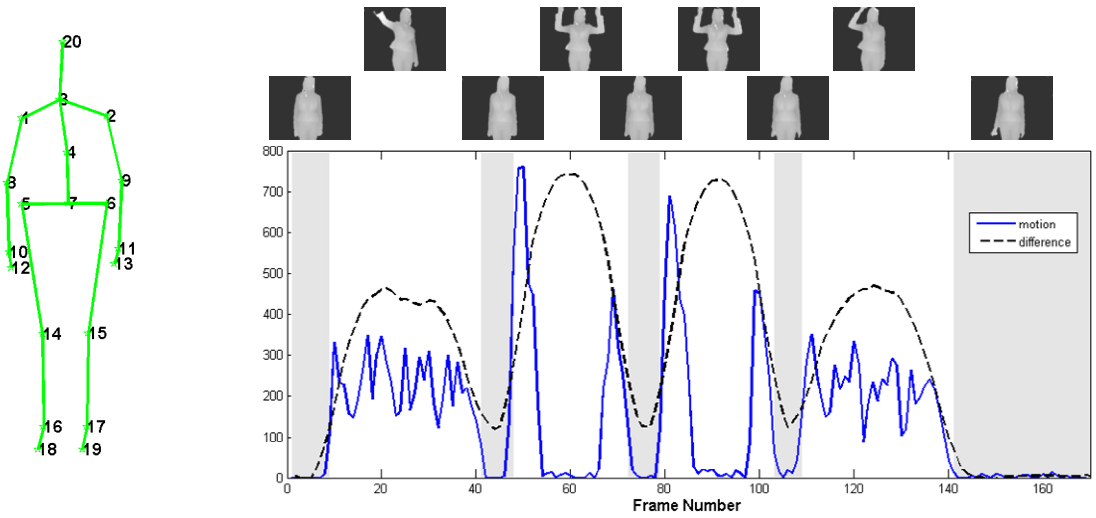


Figure 6.11. (left) The canonical skeleton used showing the 20 joints, each having XY and depth coordinate Z (right) Multi-gesture video sequence with 4 active gesture areas separated by five non-gesture regions indicated in gray areas. The solid blue curve is a frame to frame difference signature; the black dotted line is a frame-canonical depth image signature. The images at top show the depth frame at the center of the non-gesture and gesture regions.

A canonical resting image is formulated by averaging all frames of a video sequence in which no motion is detected, and serves as a reference image for comparison. Using this canonical resting frame, a difference indicator is calculated in similar fashion as the motion indicator, except instead of calculating the difference from one frame to the next in (2.2), each frame is differenced from a single canonical resting frame. Although these motion and difference indicators are good attributes to pass into a gesture classifier, better performance is achieved by incorporating the XYZ coordinates of the 20 skeletal joints for active frames in the temporal sequence.

The 20 XYZ skeletal joint coordinates in each active frame are normalized akin to a Procrustes analysis [67] in preparation for subsequent processing. This normalization makes our technique invariant to subject distance from the camera, subject size, and subject location within the frame. Setting *s* equal to

a vector of XYZ skeleton joints from a test frame and *c* equal to a vector of XYZ canonical skeleton joints determined in the laboratory (and pictorially showed in Fig. 6.11):

$$s' = s - \sum_{i=1}^{n_j} {}^{s_{ij}}/_{n_j} \qquad j \in 1..3; \quad i \in 1..20 \tag{6.2}$$

$$s'' = s' \left(\frac{size(canonicalSkeleton)}{size(s')} \right) \tag{6.3}$$

$$s''' = s'' + \left(\sum_{i=i}^{n_j} {}^{c_{ij}}/_{n_j} - \sum_{i=1}^{n_j} {}^{s''_{ij}}/_{n_j} \right) \quad j \in 1..3; \quad i \in [3,4,7,20] \tag{6.4}$$

The skeleton joints are first shifted such that the centroid is at (0,0,0). The *size* of a skeleton is the joint to joint geodesic measure (i.e. the 3D length of the 20 green lines in Figure 6.11). After scaling, the skeleton is shifted back to the canonical skeleton location using a centroid calculation of only the head and spine joints (joints numbed 3, 4, 7, and 20). Omitting arm and leg joints enables the body mass to remain stationary even if a subject's arm or leg is fully extended.

After joint normalization, the active difference signature attribute is formed by differencing the 20 normalized skeleton joints of each frame with the 20 canonical skeleton joint locations. For static gesture estimation, these joint difference attributes, a 20x3 feature $x \in \mathbf{R}^{60}$, can be passed directly into the classifier stage. For temporal gesture estimation, the joint attributes for all active frames between gesture boundaries are dynamically time warped to form a virtual window of 25 frames. This is done separately for the X, Y, and Z direction by forming an image of θx20, where θ is the number of frames in the active window and 20 is the number of skeletal joints.

After prefiltering to prevent aliasing, the image is resampled to 25x20 pixels. This procedure yields 25 frames with 20 skeletal joints per frame, where the combination of the X, Y, and Z prefiltered images form the temporal joint attribute feature $x \in \mathbf{R}^{1500}$. Not only does the dynamic time warping convert each active sequence to a fixed number of dimensions in preparation for classification, but it also removes high frequency noise between frames. Figure 6.12 shows an active difference signature on the left and a difference signature on the right. Each of the 20 lines shows the temporal movement of each of the 20 skeletal joints across the dynamically warped 25 frame timeline. The gesture in Figure 6.12 did not start until after frame 10, and thus the signature on the right does not generalize well. Perhaps the biggest benefit of the dynamically warped active difference signatures is that they are invariant to the speed at which the subject is performing gestures.

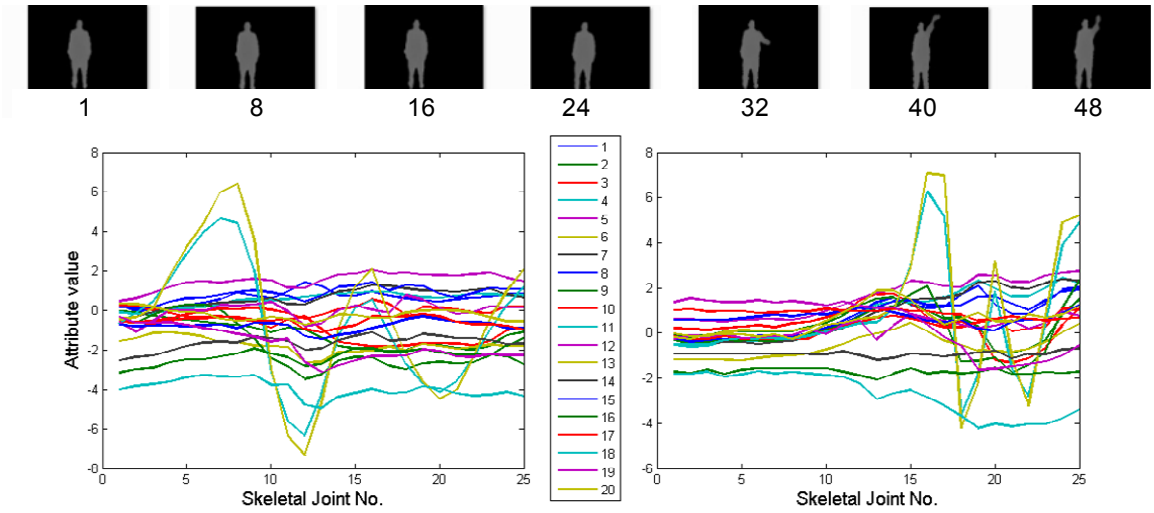


Figure 6.12. (left) Comparison of an active difference signature (left) vs. a difference signature (right) for the waving gesture from sample 1 of the MSR3D dataset. Each of the lines in the two figures shows the temporal displacement of one of the 20 skeletal joints from the canonical skeletal frame (see Figure 6.11 for point annotation). Sample 1 was a left hand wave as shown by the samples frame thumbnails atop, and as such, kinematic joints 11 and 13 showed the most displacement from the canonical skeleton. The numbers below each frame indicate the frame number from the dataset video.

Classification can be done using methods such as k-NN, neural nets, or support vector machines (SVMs). Our testing has consistently shown that Manifold based Sparse Representations (MSR) offers the highest accuracy and provides moderate robustness to outliers and partial occlusions. In particular, we use equation (3.26) to solve for $\boldsymbol{W}$, and LPP to generate dimensionality reduction matrix $\boldsymbol{U}$ using (3.21), and solve for the low dimensional sample, $\boldsymbol{y}^T = \boldsymbol{x}^T\boldsymbol{U}$. Sparse Representation (SR) techniques convert y into a gesture estimate by solving the ℓ^1 minimization of low dimensional sample y, using (5.5), and making a class estimate on sparse coefficients $\boldsymbol{a}$ using a minimum reconstruction error (5.6).

6.5 Interactive Display Gestures

A bakery HCI application was developed to prove feasibility of interactive gestural interfaces. This interactive display allowed users to point, select (two handed gesture- while one hand points, the second hand is brought to the chest as an adoption of American Sign Language symbol meaning “mine”), and release (arm waving over head) products from a bakery showcase. The continuous nature of pointing and waving prohibit the usage of active gesture areas, as these gestures can go on without a start, middle, or end. Instead of utilizing the temporal joint attribute, we treat each frame as active and utilize the static gesture classifier. Several training videos were shot of each gesture in isolation. The canonical skeleton was formed from resting frames extracted from training videos. Difference signatures from this canonical skeleton were fed into SLPP manifold learning, whereby a linear SVM model is created.

During runtime, the skeletal joints are normalized as per (6.2-6.4), differenced from the canonical skeleton, projected onto the SLPP manifold, and passed into the SVM classifier. System overhead is less than 2 msec per frame. To collect quantitative results, a session of a user interacting with the device was recorded and passed through our framework. Figure 6.13 illustrates the performance of the HCI system on this test video of 369 frames. The accuracy was 92.7%, with most errors coming from vague interpretations of gesture transitions. We also had a small number of frames at the start/end of a wave gesture classified as pointing. In practice, these errors do not affect system performance and the overall user experience.

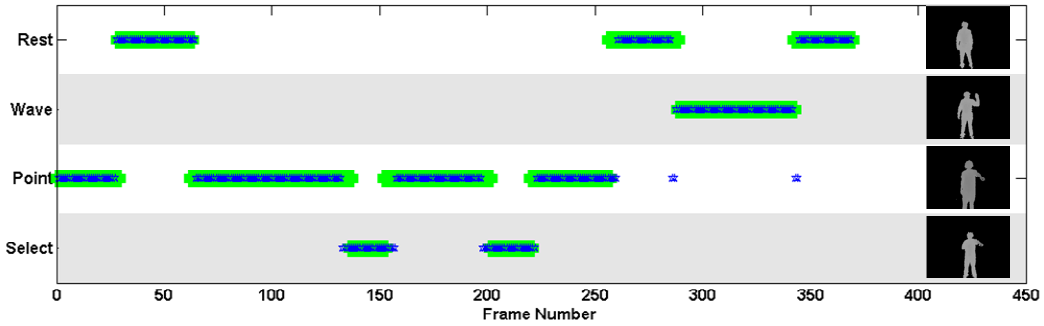


Figure 6.13. Sample test sequence from the HCI bakery. Ground truth is recorded with large green blobs, and difference signature predictions from the static gesture classifier are the blue pentagrams. The user started with a pointing gesture, then rested, then back to pointing followed by select, etc. The frames on the right are random frames chosen from each gesture type.

6.6 Active Difference Signature Results

6.6.1 MSR3D Dataset

The Microsoft Action 3D (MSR3D) dataset proposed by Li et al. [146] is the only known standard dataset that contains both depth maps and corresponding skeletal joint locations. It consists of depth map sequences with a resolution of 320x240 pixels recorded with a depth sensor at 15 FPS. There are ten subjects performing twenty actions two to three times with a total of 567 depth map sequences. The dataset actions are: high arm wave, horizontal arm wave, hammer, catch, tennis swing, forward punch, high throw, draw 'X', draw tick, tennis serve, draw circle, hand clap, two hand wave, side boxing, golf swing, side boxing bend, forward kick, side kick, jogging, and pick up and throw. No corresponding RGB information is available, however twenty (see Fig. 6.11) 3D kinematic joint positions are provided for each frame.

We benchmark our method against state-of-the-art techniques and then examine the performance of the active difference signature attribute across several variants of techniques utilizing dimensionality reduction and sparse representations. The dimensionality reduction techniques include PCA, LDA, LPP [75], and NPE [76].

6.6.2 Experimental Methodologies

A leave-one-subject out cross-validation methodology was used to separate the MSR3D dataset into separate training and test sets. Each test subject is validated against the remaining nine subjects and the process is repeated until all subjects have been used for training and testing. The results from each subject are averaged to give a final performance result. The dimensionality reduction techniques capture 99% of the data variance. LDA uses (3.22), LPP uses (3.23), and NPE uses (3.24) and (3.25). The SLPP method uses α=0.5 in creation of $\boldsymbol{W}$ using (3.26). The sparse coefficients from test samples are generated using (5.5), setting λ=0.15. The low dimensional projection of all training samples in the cross-validation training split forms the training dictionary in the SR techniques. The corresponding sparse coefficients of test samples use (5.6) to make a final classification estimate.

6.6.3 Experimental Results

Table 6.13 shows the classification results of our method against other state-of-the-art gesture recognition techniques. The first three techniques are existing temporal techniques we adopted for gesture recognition, techniques '4' and '5' were published results on the MSR3D dataset, and technique '6' is the active difference signatures method proposed in this paper. The first three techniques were: Motion History Images (MHI) were initially introduced for human movement recognition [84], and were later adopted for facial AU detection [66]; SIFT flow [86] is an image alignment algorithm introduced to register two similar images; Optical Flow of Skeletal Joints tracks the skeletal joints frame by frame, forming the difference between each joint coordinate and a canonical skeletal coordinate. Bag of Features [146] uses action graphs to model the dynamics of the actions and a bag of features to encode the action. Spatio-Temporal Joint Descriptor [147] encodes the difference between each skeletal joint and the centroid of the skeleton, and then uses dynamic time warping to generate gesture attributes.

TABLE 6.13. CLASSIFICATION ACCURACY ON THE MSR3D DATASET FOR VARIOUS GESTURE RECOGNITION TECHNIQUES.

Method	Classifier	Accuracy
MHI [66]	SRC	62.1
SIFT flow [86]	SRC	40.8
Optical Flow of Skeletal Joints	SVM	40.9
Bag of Features [146]	NERF	74.7 *
Spatio-Temporal Joint Descriptor [147]	SRC	72.3
Active Difference Signatures (our work)	SRC	82.6

*NOTE: THE CLASSIFICATION ACCURACY FOR (LI. ET AL., 2010) CANNOT BE DIRECTLY COMPARED TO OTHER TECHNIQUES BECAUSE A DIFFERENT VALIDATION SCHEME WAS USED.

The results in Table 6.13 show the significant advantage of active difference signatures on final classification rates. In particular, by concentrating only on active frames, the attributes passed into the

classifier are more discriminative. For example, methods '5' and '6' in Table 6.13 both used similar sparse representation classifiers. Methods '1', '2', and '4' used depth pixels as the primary feature, while methods '3', '5', and '6' used the 3D skeletal joint coordinates as the primary feature. It should be noted that the results for method '4' used half the subjects for training, the other half for testing, which is not directly comparable to the leave-one-subject-out cross validation used by the other five methods. Nonetheless, we include method '4', as Li et al., introduced the MSR3D dataset. Under the classifier column, SRC is the Sparse Representation Classifier, and NERF is a fuzzy spectral clustering method that classified the test sample according to the training sample with the minimum Hausdorff distance.

Table 6.14 demonstrates the effectiveness of our active difference signature attribute across various dimensionality reduction and classification techniques. All techniques except for spectral regression perform quite well on this dataset. The Sparse Representation Classification (SRC) minimum reconstruction error classifier performs slightly better than the multi-class linear SVM classifier across the several dimensionality reduction techniques. Aside from being slightly advantaged from a classification point of view, the SR techniques are ideal for HCI applications because new gestures can easily be added to the dictionary Φ without any new retraining. Further, SR methods are advantaged if there are very few data samples per a particular gesture class, as the reconstruction error (5.6) favors these classes significantly when solving the ℓ^1 minimization for test sample y.

TABLE 6.14. CLASSIFICATION ACCURACY FOR TEMPORAL JOINT ATTRIBUTES ON THE MSR3D DATASET. LINEAR SVM AND SPARSE REPRESENTATION CLASSIFICATION TECHNIQUES ARE CONTRASTED OVER VARIOUS DIMENSIONALITY REDUCTION TECHNIQUES. *D* IS THE OUTPUT DIMENSIONS FROM DIMENSIONALITY REDUCTION.

Method	*d*	SVM	SRC
PCA	141	77.7	81.7
LDA	19	76.7	77.9
LPP [75]	141	75.9	79.8
SLPP [75]	141	79.5	82.6
NPE [76]	100	76.8	83.6
AVG		73.0	76.6

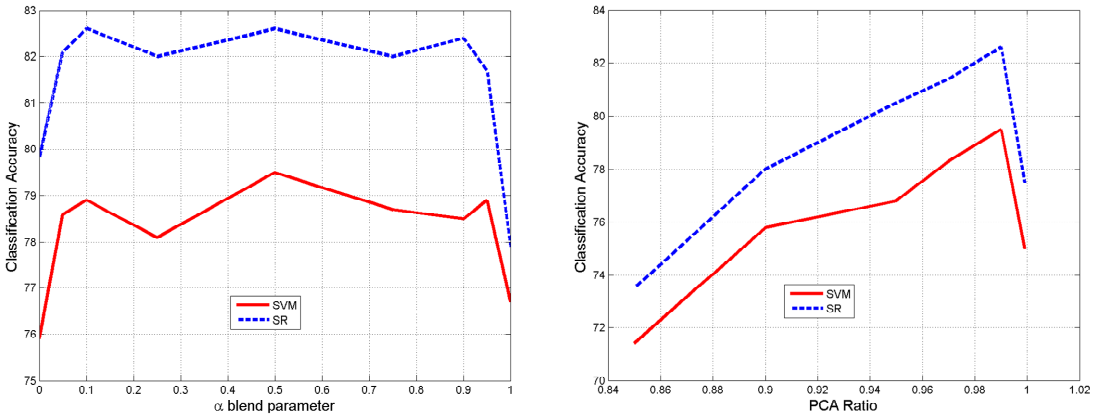


Figure 6.14. (left) Classification accuracy as a function of the α blend parameter for SLPP dimensionality reduction. (right) Classification accuracy as a function of the percent of variance of eigenvalues kept in SLPP dimensionality reduction.

The plot on the left of Figure 6.14 shows the robustness of the SLPP technique as a function of LDA parameter α in (3.26). As α approaches 0, SLPP becomes unsupervised, and as α approaches 1.0, SLPP approaches the fully supervised LDA technique. The plot on the right of Figure 6.14 shows the classification performance of SLPP as more or less eigenvectors are utilized in dimensionality reduction matrix $\boldsymbol{U}$. The PCA ratio is the proportion of variance maintained in $\boldsymbol{U}$. Lower values mean fewer eigenvalues and more generalization. Higher numbers mean more eigenvalues that are subject to over fitting. The SR approach outperforms SVM for all α blend values and all PCA ratio values.

7 Joint Optimization of Manifold Learning and Sparse Representations

7.1 Introduction to LGE-KSVD

Research in manifold learning has influenced the SR community and vice-versa. The Sparsity Preserving Projections approach [148] replaces the adjacency matrix used in LGE techniques with SR sparse coefficients. Mairal *et al*. [149] injects a multiclass logistic regression term to the sparse energy function to make dictionary learning have both reconstructive and discriminative properties. Discriminative Sparse Coding [150] uses sparse coefficients in an LDA framework. Graph Regularized Sparse Coding [151] adds the LGE objective function on sparse coefficients to the traditional ℓ^1 sparse objective function as it jointly learns the sparse coefficients and dictionary terms.

The works of [149, 152, 153] jointly optimize dictionary learning and classifier training to select exemplars that minimize both reconstructive and discriminative errors. Jiang *et al*. [123] devised efficient methods for choosing Φ from a set of training exemplars by simultaneously minimizing both reconstruction and classification errors. The work in [123] encourages input samples from the same class to have similar sparse codes.

Although methods for populating the adjacency matrix $\boldsymbol{W}$ vary, sparseness is one common characteristic across all techniques. Sparsity Preserving Projections (SPP) [148] is similar to NPE, but uses sparse coefficients instead of local topology when solving for $\boldsymbol{W}$. Global Sparse Representation Projections [154] modifies the dimensionality reduction function in SPP to simultaneously maximize supervised class separability and minimize sparse representation error. [150] uses the sparse coefficients to populate matrix $\boldsymbol{W}$, then adds supervised similarity and dissimilarity matrices akin to LDA. [151] replaces the $\boldsymbol{y}$ terms in (3.14) with coefficients $\hat{\boldsymbol{a}}$, claiming that nearby samples should have similar coefficients. Each of the above methods introduces a new dimensionality reduction technique or a new SR technique. What lacks is a unified approach that optimizes dimensionality reduction projection matrix $\boldsymbol{U}$ with dictionary Φ, and coefficients $\hat{\boldsymbol{a}}$.

This thesis furthers the field of sparse representation classification by solidifying the relationship between manifold learning and SRs. In particular, this section:

i. Proposes a method to jointly optimize dimensionality reduction, sparse dictionary learning, sparse coefficients, and a sparsity-based classifier in an elegant framework.

ii. Employs the SR concept in a dimensionality-reduced space obtained by a semi-supervised variant of LGE. This dimensionality-reduced space minimizes coefficient contamination, is suitable for posed and natural datasets, static and temporal datasets, problems with small and large number of

classes, and is computationally efficient.

iii. Redefines the atom optimization process in K-SVD to allow variable support using graph embedding techniques.

This new framework is called LGE-KSVD, which stands for the optimization and infusion of Linear extension of Graph Embedding with K-SVD dictionary learning.

7.2 LGE-KSVD Method

Classification frameworks based on SR concepts have been found to suffer from:

a. Coefficient contamination that compromises classification accuracy, and
b. Computational inefficiencies due to high dimensional features and large dictionaries.

The usage of LGE-KSVD aims to address both limitations via semi-supervised dimensionality reduction and K-SVD dictionary learning. We have considered many ways to combine LGE concepts with K-SVD, and the proposed LGE-KSVD approach presents a framework that provides robust and accurate classification across a diverse field of computer vision problems.

We begin by combining the dimensionality reduction matrix $\boldsymbol{U}$ from (3.21) with a method to learn a dictionary Φ and sparse coefficients $\hat{\boldsymbol{a}}$. K-SVD solves:

$$\{\widehat{\Phi}, \hat{a}\} = \arg \min \|x - \Phi a\|_2^2 \;\; s.t. \|a\|_0 \leq \delta \tag{7.1}$$

Combining (3.21) with (7.1), we get:

$$\{\widehat{U}, \widehat{\Phi}, \hat{a}\} = \arg \min \|X^T U - \Phi a\|_2^2 \; + \frac{U^T X L X^T U}{U^T X D X^T U} \tag{7.2}$$

$$s.t. \|a\|_0 \leq \delta$$

The first term performs K-SVD optimization in low dimensional space, and the second term is the LGE dimensionality reduction objective function. Since we are solving a classification problem, we need to convert test sample coefficients $\hat{\boldsymbol{a}}$ into a classification label estimate. Because dictionary elements from K-SVD are a linear combination of input samples, we cannot use the minimum reconstruction error in (5.6). Alternatively, we can pass the coefficients $\hat{\boldsymbol{a}}$ into any regression based or machine learning based classifier. Our research suggests to perform classification with coefficient transformation matrix $\boldsymbol{C}$, $\boldsymbol{C} \in \mathbf{R}^{mxk}$, where k is the number of classes and m is the number of dictionary elements.

We define $\boldsymbol{H}$ as a sparse ground truth matrix, $\boldsymbol{H} \in \mathbf{R}^{kxn}$. Each column of $\boldsymbol{H}$ corresponds to a training sample, where the k^{th} element is set to 1 if y_i belongs to class k, 0 otherwise. Coefficients $\hat{\boldsymbol{a}}$ from each training example are stored into matrix $\boldsymbol{A}$, $\boldsymbol{A} \in \mathbf{R}^{mxn}$. This problem is formulated as:

$$\hat{C} = \arg \min \|H - C^T A\|_2^2 \tag{7.3}$$

The above can be solved directly via ridge regression:

$$C = (AA^T)^{-1} AH^T \tag{7.4}$$

7.3 Training Procedure for LGE-KSVD

Equation (7.2) is neither directly solvable nor convex. Using K-SVD with n training samples, we learn a dictionary Φ of m atoms, where $m \leq n$ via an implicit transformation $\boldsymbol{T}$, $\boldsymbol{T} \in \mathbf{R}^{mxn}$, resulting in $\Phi = \boldsymbol{TY}^T = \boldsymbol{TX}^T \boldsymbol{U}$. As such, the dictionary transformation function $\boldsymbol{T}$ and the dimensionality reduction transformation function $\boldsymbol{U}$ will oscillate if we indiscriminately iterate one after the other.

For smooth and reliable results, we desire an overcomplete dictionary in which the number of samples, m is greater than the number of dimensions d. $\boldsymbol{T}$ has rank $\leq m$, $\boldsymbol{U}$ has rank $\leq d$. Since $m \geq d$, $\boldsymbol{T}$ in general has more degrees of freedom and it is preferable to iterate on $\boldsymbol{T}$ more often than $\boldsymbol{U}$. As we minimize reconstruction errors in (7.2), coefficients $\hat{\boldsymbol{a}}$ develop into more accurate representation of $\boldsymbol{X}$, resulting in lower classification errors.

In [63] it was sown that supervised dimensionality reduction minimizes SR coefficient contamination by enforcing class separation. A discriminative dictionary was utilized in [123, 148, 150, 154]. We find better results if the SR energy function minimizes reconstruction errors and the LGE energy function encourages class discrimination. Not only does this offer superior classification results, but because we are operating in a low dimensional space, the resulting framework minimizes compute resources.

After an initial dimensionality reduction matrix $\boldsymbol{U}$ is obtained via semi-supervised LGE, we propose a double nested iterative training procedure. The outer loop updates $\boldsymbol{U}$ based upon the current best estimates of Φ and $\boldsymbol{a}$, and the inner loop uses K-SVD to iteratively update $\boldsymbol{a}$, then Φ, based upon the current best estimate of $\boldsymbol{U}$.

To get an updated estimate of $\boldsymbol{U}$, coefficients $\boldsymbol{a}$ from each training example are stored into matrix $\boldsymbol{A}$, $\boldsymbol{A} \in \mathbf{R}^{mxn}$. The update problem is then formulated as:

$$\hat{U} = \arg \min \left\| X^T U - A^T \Phi^T \right\|_2^2 \tag{7.5}$$

and is solved directly:

$$U = (XX^T)^{-1} XA^T \Phi^T \tag{7.6}$$

Fig. 1 summarizes the training procedure.

WHILE ε has not converged or ε > τ

IF firstIteration

1a. Calculate $\boldsymbol{U}$ using LGE.

ELSE

1b. Calculate $\boldsymbol{U}$ using (3.16).

ENDIF

2. Calculate low dimensional samples $\boldsymbol{Y}= \boldsymbol{X}^T\boldsymbol{U}$.

3. Initialize the m samples of Φ randomly from the n low dimensional training samples.

4. Calculate $\{\boldsymbol{A}, \Phi\}$ using modified K-SVD, substituting $\boldsymbol{Y}$ for $\boldsymbol{X}$.

5. Calculate $\boldsymbol{C}$ using (7.4).

6. Calculate verification set error, $\varepsilon = \|\boldsymbol{H} - \boldsymbol{C}^T\boldsymbol{A}\|_2^2$.

ENDWHILE

Fig. 7.1. Training procedure for LGE-KSVD.

The choice of LGE in Step 1a of the training procedure should be a discriminative embedding that maintains input topology. The best approach we have found uses a convex combination of supervised and unsupervised adjacency matrices $\boldsymbol{W}_{\boldsymbol{LDA}}$ and $\boldsymbol{W}_{\boldsymbol{Gaussian}}$ corresponding to (3.22) and (3.23) respectively. The two are combined into a single $\boldsymbol{W}$ using (3.26). For posed datasets that are linearly separated, $\boldsymbol{W}_{\boldsymbol{LDA}}$ is weighted higher. For natural datasets or problems where the number of classes is small, we emphasize $\boldsymbol{W}_{\boldsymbol{Gaussian}}$.

The initialization Step 3 in the training procedure is done such that each of the k classes is represented in proportion to the number of samples that it occupies in the training set. For example, if the training set has 30% of the samples from class A, 30% of Φ is initialized using training samples from class A.

7.4 Modified K-SVD

The K-SVD penalty term of (7.1) can be rewritten as:

$$\begin{aligned} \|X - \Phi A\|_F^2 &= \left\| X - \sum_{z=1}^{m} \Phi_z A_T^z \right\|_F^2 \\ &= \left\| \left(X - \sum_{z \neq j} \Phi_z A_T^z \right) - \Phi_j A_T^j \right\|_F^2 \\ &= \left\| E_j - \Phi_j A_T^j \right\|_F^2 \end{aligned} \tag{7.7}$$

Where $\Phi \boldsymbol{A}$ is decomposed into the sum of m rank-1 matrices $\in \mathbf{R}^{dxn}$, each representing one dictionary element, with $\Phi \in \mathbf{R}^{dxm}$ and $\boldsymbol{A} \in \mathbf{R}^{mxn}$, and $\|\cdot\|_F$ is the Frobenius norm. The error, E_j is the total error for all n training samples with the j^{th} dictionary element removed. The step of updating dictionary elements sequentially updates one element at a time. While updating element j, K-SVD assumes that dictionary matrix Φ and sparse coefficient matrix $\boldsymbol{A}$ are fixed except for (column) element j of Φ, $\Phi_j \in \mathbf{R}^{dx1}$, and the corresponding coefficients for Φ_j, which comprise row j of coefficient matrix $\boldsymbol{A}$, $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}} \in \mathbf{R}^{lxn}$.

WHILE ε has not converged or ε > τ
1. Calculate coefficients $\boldsymbol{A}$ using (5.4) or (5.5).
2. Update dictionary Φ:
  FOREACH element j in dictionary
    2a. Calculate Φ_j, $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$, and E^R_j.
  END FOREACH
END WHILE

Fig. 7.2. The K-SVD Algorithm.

Singular Value Decomposition (SVD) solves the closest rank-1 matrix that approximates E_j, yielding the optimal Φ_j and $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$ directly; our new best estimates for dictionary element j and its corresponding coefficients. Unfortunately $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$ would tend to use many, if not all, training samples, resulting in a non-sparse $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$. K-SVD enforces sparsity by fixing the support of Φ_j to only those training entries with non-zero coefficients of element j; as such, at initialization, each dictionary element is paired up with a list of training samples that can never change. A vector $\boldsymbol{\eta_j}$ is initialized with zeros, then corresponding entries with non-zero $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$ are used as indices to x_i that use dictionary element Φ_j:

$$\eta_j = \{i | 1 \le i \le J, A_T^j(i) \ne 0\} \tag{7.8}$$

Vector $\boldsymbol{\eta_j}$ is used to build matrix $\boldsymbol{\Omega_j} \in \mathbf{R}^{nx|\eta j|}$. By multiplying $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$ and E_j by $\boldsymbol{\Omega_j}$, we remove zero entries, using only the subset of dictionary samples that use element Φ_j, forming restricted row vector $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{R}}$ and error matrix $\boldsymbol{E}^{\boldsymbol{R}}_{\boldsymbol{j}}$:

$$\left\| E_j \Omega_j - \Phi_j A_T^j \Omega_j \right\|_F^2 = \left\| E_j^R - \Phi_j A_R^j \right\|_F^2 \tag{7.9}$$

By minimizing (7.9), K-SVD forces the solution of Φ_j and $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{R}}$ to have identical support to the original $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{T}}$. SVD directly decomposes the rank-1 matrix $\boldsymbol{E}^{\boldsymbol{R}}_{\boldsymbol{j}} = \mathbf{U}\Delta\mathbf{V}^{\mathrm{T}}$. Φ_j is the first column of $\boldsymbol{U}$, and $\boldsymbol{A}^{\mathbf{j}}_{\mathbf{R}}$ is the first column of $\boldsymbol{V}$ multiplied by $\Delta(1,1)$.

K-SVD can be modified by incorporating supervision into the updating of dictionary element *j*. Each dictionary element is assigned a class using a minimum reconstruction error via (5.6), and then the element's support, controlled via $\boldsymbol{\eta_j}$, is restricted to the training samples of its own class.

K-SVD without or with supervision fixes the support of each element at the time of dictionary initialization. The former makes K-SVD dependent upon a good initialization guess; the latter may seem good from a classification perspective, but is poor from a reconstruction perspective, as test samples often necessitate elements from more than one training class.

An improvement over both approaches is to let the training samples that contribute to each dictionary element be governed by sample-to-sample similarity and class labels. Further, as long as sparsity is maintained, it is desired for the support of each element, $\boldsymbol{\eta_j}$, to change at each iteration. We propose to use semi-supervised LGE adjacency matrix $\boldsymbol{W}$ as per (3.26) to regulate the support of each dictionary element. In particular, the support of dictionary element *j* may:

a) Expand: Modify the support of element *j* by adding (union) all training entries *similar* to element *j*.

b) Contract: Modify the support of element *j* by removing (intersection) training entries *not similar* to element *j*.

c) Redefine: Set the support of element *j* to be only training samples *similar* to element *j*.

d) Fixed: Maintain the support of element *j*, as in the K-SVD algorithm.

In LGE-KSVD, *similar* is defined in terms of the LGE adjacency matrix $\boldsymbol{W}$, i.e., training samples respect all samples of same class or nearest neighbors, as defined by $\boldsymbol{W}$. During the updating of dictionary element *j*, LGE-KSVD modifies the support of $\boldsymbol{E_j}$ using the expand, contract, redefine, and fixed operations, creating a different $\boldsymbol{\eta_j}$ and $\boldsymbol{E^R_j}$ for each condition. Given four sets of $\boldsymbol{\eta_j}$, $\boldsymbol{E^R_j}$, Φ_j, and $\boldsymbol{A^j_T}$ we choose the Φ_j and $\boldsymbol{A^j_T}$ that minimize the penalty term (7.7) as:

$$\{\widehat{\Phi_J}, \widehat{A_T^J}\} = arg \min_{i=1:4} \left(\left\| E_j^R{}_i - \Phi_j{}_i A_{Ri}^j \right\|_F^2 \right) \tag{7.10}$$

Rather than assigning a single class to each element *j*, LGE-KSVD uses the top coefficients from $\boldsymbol{A^j_R}$. Each of those top coefficients is used as a look-up into adjacency matrix $\boldsymbol{W}$. All training samples similar to each of those top coefficients (as defined by $\boldsymbol{W}$) are used to expand, contract, or redefine $\boldsymbol{E^R_j}$ before the SVD decomposition. The top coefficients can be solved by keeping the top percentile of total energy, looking for the steepest slope of the cumulative energy distribution of $\boldsymbol{A^j_T}$, or a combination thereof.

Deciding which neighbors are similar, based on the top coefficients, is dependent upon the technique used to build $\boldsymbol{W}$. By using a sparse $\boldsymbol{W}$ matrix, any non-zero entry is a similar neighbor to j. For example, by setting τ=1.0 in (3.23), similar neighbors are those with either the same ground truth class, or neighbors that have a small ℓ^2 norm in low dimensional space. If $\boldsymbol{W}$ is non-sparse, only w_{ij} entries above a threshold are considered similar. Although it may seem desirable to use the same $\boldsymbol{W}$ for dimensionality reduction as well as element neighbor similarity determination, there are advantages in maintaining some degree of flexibility. For example, it is often desirable to decrease α in (3.26) or increase τ in (3.23) for element neighbor similarity determination. Both modifications make $\boldsymbol{W}$ less sparse, and perhaps less discriminate, but simultaneously make $\boldsymbol{W}$ more open to finding relationships between diverse training samples. The modified K-SVD algorithm is summarized in Figure 7.3.

WHILE ε has not converged or $\varepsilon > \tau$
1. Calculate coefficients $\boldsymbol{A}$ using (5.4) or (5.5).
2. Update dictionary Φ:
  FOREACH element j in dictionary
    2a. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in expand mode.
    2b. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in contract mode.
    2c. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in redefine mode.
    2d. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in fixed mode.
    2e. Select the $\boldsymbol{E}^{\boldsymbol{R}}_j$ and corresponding Φ_j and $\boldsymbol{A}^j_{\mathbf{R}}$ that minimize (7.10).
  END FOREACH
END WHILE

Fig. 7.3. The modified K-SVD Algorithm.

7.5 Testing Procedure for LGE-KSVD

The training procedure solves for dimensionality reduction matrix $\boldsymbol{U}$, dictionary, Φ, and coefficient transformation matrix $\boldsymbol{C}$. Given a test sample x, along with $\boldsymbol{U}$, Φ, and C, Fig. 7.4 summarizes the testing procedure. The coefficient transformation matrix $\boldsymbol{C}$ in Step 3 of the testing procedure converts the sparse coefficients $\hat{\boldsymbol{a}}$ to class label estimates. Ideally, $\boldsymbol{l}$ would contain most of its energy in one class. The normalized energy of l_i and other $\boldsymbol{l}$ values provide confidence indicators as to which class x belongs.

1. Calculate low dimensional sample $\boldsymbol{y}=\boldsymbol{x}^T\boldsymbol{U}$.
2. Calculate sparse coefficients $\hat{\boldsymbol{a}}$ using (5.5).
3. Use C along with $\hat{\boldsymbol{a}}$ to estimate class label vector $\boldsymbol{l}\in\mathbf{R}^{kx1}$ where the maximum value of $\boldsymbol{l}$ is used as a class predictor.

$$\hat{l} = \max_{i=1:k}(l = C^T\hat{a}) \qquad (7.11)$$

Fig. 7.4. Testing procedure for LGE-KSVD.

7.6 LGE-KSVD Method Overview and Parameter Selection

Because there are many aspects to LGE-KSVD, the choices and intuition for each step along with the optimization of the necessary tuning parameters are briefly summarized in this section along with Fig. 7.5.

We begin with dimensionality reduction on the input data to minimize compute resources. We gravitate to the LGE family of methods of which LPP generally outperforms other methods. The introduction of supervision to LGE is necessary to minimize coefficient contamination, but, without some unsupervised component it is difficult to capture class to class similarities. At this point, we need to decide what flavor of SR to use, a value for τ in LPP, and the value of α to generate a semi-supervised dimensionality reduction.

Regarding the selection of a SR technique, several pursuit algorithms are quite good at solving (5.4) or (5.5). We found that it is optimal to use (5.5) with $0.1\leq\lambda\leq0.3$. Furthermore, we use non-negative coefficients for SR classification purposes. Regarding the value of τ in (3.23), we generally require a small amount of supervision, and thus we select $\tau=1$, which provides favorable results for all experiments that follow. Regarding the selection of α in (3.26), as long as $\alpha\neq0$ and $\alpha\neq1$, results are reasonable. We recommend α values in the range $0.1\leq\alpha\leq0.9$, and use $\alpha=0.5$ for all LGE-KSVD experiments.

Our experiments show that the introduction of K-SVD not only shrinks the size of the dictionary, but also results in higher classification accuracy. Another decision to make is selecting m, the size of the dictionary. This will vary with the dataset size n, and we typically start by setting $m=n/2$.

Motivated to remove the support restrictions imposed by K-SVD, LGE concepts are introduced into the K-SVD algorithm. It is necessary to increase τ in the creation of the adjacency matrix used by K-SVD to find similar neighbors. A value of $\tau=100$ is a good rule of thumb which will be used in all experimental results.

WHILE1 ε has not converged

IF firstIteration

1a. Calculate $\boldsymbol{U}$ using LGE.

ELSE

1b. Calculate $\boldsymbol{U}$ using (3.16).

ENDIF

2. Calculate low dimensional samples $\boldsymbol{Y}^T = \boldsymbol{X}^T\boldsymbol{U}$.

3. Initialize the m samples of Φ randomly from the n low dimensional training samples.

4. Calculate $\{\boldsymbol{A}, \Phi\}$ using modified K-SVD:

WHILE2 ξ has not converged

4.1. Calculate coefficients $\boldsymbol{A}$ using (5.5).

4.2. Update dictionary Φ:

FOREACH element j in dictionary

4.2a. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in expand.

4.2b. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in contract.

4.2c. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in redefine.

4.2d. Calculate Φ_j, $\boldsymbol{A}^j_{\mathbf{R}}$, and $\boldsymbol{E}^{\boldsymbol{R}}_j$ in fixed.

4.2e. Select the $\boldsymbol{E}^{\boldsymbol{R}}_j$ and corresponding Φ_j and $\boldsymbol{A}^j_{\mathbf{R}}$ that minimize (7.10).

END FOREACH

END WHILE2

5. Calculate $\boldsymbol{C}$ using (7.4).

6. Calculate verification set error, $\varepsilon = \|\boldsymbol{H} - \boldsymbol{C}^T\boldsymbol{A}\|_2^2$.

END WHILE1

Fig. 7.5. Full LGE-KSVD training procedure.

The selection of top coefficients is done by picking any training sample that contains 50% or more of the total coefficient energy. Similar neighbors are those elements whose Euclidean distance in the W matrix includes 99% of all related training samples. This accounts for 100% of samples of the same class, and other samples that are deemed to be similar by the unsupervised LPP adjacency matrix.

7.7 LGE-KSVD Experiments

The LGE-KSVD approach was evaluated on four widely used databases: the extended Cohn-Kanade (CK+) facial expression dataset [58], the extended Yale B facial recognition database [155], the Facial Expression Recognition and Analysis Challenge (FERA2011) GEMEP-FERA [139] dataset, and the i3DPost multi-view activity recognition dataset [156]. We test each dataset across three categories of (i)

dimensionality reduction; (ii) sparse representation; and (iii) combined techniques. The dimensionality reduction techniques include PCA, LDA, LPP [75], NPE [76], and Sparsity Preserving Projections (SPP) [148]. The sparse representation methods include K-SVD [131], LC-KSVD1 and LC-KSVD2 [123]. The combined methods include Sparse Representation-based Classification (SRC) [78], Manifold based Sparse Representation (MSR) [63], and the proposed LGE-KSVD method.

7.7.1 LGE-KSVD Testing Datasets

An Active Appearance Model (AAM) automatically localizes 68 points on each face of the CK+ [58] expression dataset. The AAM eye and mouth corner points are used to define an affine warp to a canonical face of 60x51 pixels. As such, from this dataset we compare two variants: D=68x2=136 (AAM point based), and D=60x51=3060 (pixel based). Each has 164 training and 163 testing faces (chosen randomly), and the K-SVD methods use a dictionary size of m=63 elements.

Each face of Extended YaleB facial recognition dataset is 192x168 pixels which are reduced to D=504 via random projections following [78]. The test set contains 1216 training faces and 1198 testing faces. The K-SVD methods use a dictionary size of m=570 elements.

Automatically localized eye and mouth corner points define an affine warp to a canonical face of 60x51 pixels per each frame of the GEMEP-FERA temporal expression dataset. A sequence of 16 frames at the 1/3rd and 2/3rd mark of each video is fed into Motion History Image (MHI) [84] analysis adapted for facial expression analysis [157]. As per [157] (described in Section 2.5), MHI yields a 24x20 dense optical flow per sequence. The X and Y coordinates at each 24x20 grid point for each of the two sequences formed the D=1920 input dimensions per sample. The dataset contains 155 training and 134 test videos. The K-SVD methods use a dictionary size of m=75 elements.

Each video of the i3DPost multi-view [156] activity recognition dataset is processed to extract MHI features, giving 125 MHI sequences where each sequence contains 1500 motion vector points. PCA yielded 767 dimensions per video. The dataset contains 512 training videos and 256 testing videos. The K-SVD methods use a dictionary size of m=450 elements. (See Appendix I for more information on these datasets.)

7.7.2 LGE-KSVD Testing Methodologies

The LGE-KSVD method is compared against three types of classification approaches: dimensionality reduction followed by SVM, K-SVD approaches, and sparse representation classification with dimensionality reduction.

Dimensionality reduction techniques capture 99.9% of the data variance and are followed by multi-class linear SVM classifiers. LDA uses equation (3.22), LPP uses (3.23), and NPE uses (3.24) and

(3.25). SPP is similar to NPE, but modifies equation (3.24) to use sparse coefficients.

The sparse representation techniques all use K-SVD to define a training dictionary of size m, where $m<n$. Coefficient transformation matrix $\boldsymbol{C}$ is generated from the training set as per (7.4). Test samples use the m element dictionary to generate sparse coefficients using (5.5), setting $\lambda=0.25$. These sparse coefficients are converted to a class estimate using (7.11). LC-KSVD1 modifies the K-SVD objective function to favor clustering of coefficients by class and LC-KSVD2 further modifies the K-SVD objective function to include the solution of coefficient transformation matrix $\boldsymbol{C}$.

The SRC method uses random projection matrices for dimensionality reduction. The low dimensional projection of all training samples forms the training dictionary. The corresponding sparse coefficients of test samples use (5.6) to make a final classification estimate. The MSR method is identical to SRC, except the random projection dimensionality reduction is replaced with LPP.

All LPP methods use $\alpha=0.5$ in creation of W using (3.26). The LGE-KSVD method uses $\tau=1$ for dimensionality reduction $\boldsymbol{W}$ and $\tau=100$ for element neighbor similarity $\boldsymbol{W}$. The LGE-KSVD method keeps the top coefficients which make up 50% of the total energy from $\boldsymbol{A}^{j}_{T}$.

7.8 LGE-KSVD Experimental Results

Table 7.1 demonstrates the performance of the five dimensionality reduction methods, the three sparsity based methods, and the two combined methods against LGE-KSVD on the 7-class CK+ dataset using the 68 AAM points. Because the data has only 136 dimensions, no dimensionality reduction is used for K-SVD, LC-KSVD1, LC-KSVD2, or SRC. This is a posed dataset, and as such LDA performs the best out of the dimensionality reduction techniques. The LGE-KSVD method has two numbers in the accuracy entry for Tables 7.1-7.5. The first is with iterative convergence turned off (1 iteration), and the second is the accuracy after convergence. The value in (·) after the second accuracy entry is the number of iterations required for convergence.

Table 7.2 uses the same CK+ dataset from Table 7.1, but uses 60x51 images as input. This higher dimensional space is not as discriminative as the 68 AAM points, but all methods do well because of the large class to class separation.

Table 7.3 uses the 38-class YaleB facial recognition dataset. The 504 random projection input for all methods was further reduced in dimensionality as indicated by the d column, where d is the dimension where classification is performed. The SR methods are advantaged over the dimensionality reduction methods, while the MSR and LGE-KSVD methods perform the best.

Table 7.4 uses the 5-class GEMEP-FERA emotion dataset. Two MHI optical flow sequences per video were used as input. The dimensionality reduction methods are advantaged over the SR methods,

and the combined methods perform better than the dimensionality reduction methods.

Table 7.5 uses the 12-class i3DPost multi-view activity recognition dataset. The 767 PCA projection input for all methods was further reduced in dimensionality as indicated by the *d* column. While there is no clear winner on this dataset, the semi-supervised LPP methods performed the best.

TABLE 7.1. 7-CLASS CK+ EXPRESSION DATASET, 68 AAM POINTS. 164 TRAINING AND 163 TESTING SAMPLES

Method	*d*	*m*	*% Accuracy*
PCA	62	-	82.2
LDA	6	-	89.6
LPP	62	-	83.4
NPE	24	-	80.4
SPP	48	-	87.7
K-SVD	136	63	79.1
LC-KSVD1	136	63	79.1
LC-KSVD2	136	63	75.5
SRC	136	164	43.6
MSR	62	164	75.5
LGE-KSVD	62	63	**90.2/92.0** (2)

TABLE 7.2. 7-CLASS CK+ EXPRESSION DATASET, 60X51 IMAGES. 164 TRAINING AND 163 TESTING SAMPLES.

Method	*d*	*m*	*% Accuracy*
PCA	162	-	82.8
LDA	6	-	**86.5**
LPP	163	-	84.7
NPE	71	-	84.0
SPP	80	-	77.9
K-SVD	3060	63	84.0
LC-KSVD1	3060	63	85.9
LC-KSVD2	3060	63	84.7
SRC	500	164	71.8
MSR	163	164	79.1
LGE-KSVD	163	63	**86.5/87.1** (5)

TABLE 7.3. 38-CLASS YALEB RECOGNITION DATASET. 192X168 PIXEL IMAGES REDUCED TO 504 DIMENSIONS VIA RANDOM PROJECTIONS. 1216 TRAINING IMAGES, 1198 TESTING IMAGES.

Method	*d*	*m*	*% Accuracy*
PCA	477	-	89.1
LDA	37	-	90.3
LPP	477	-	89.3
NPE	271	-	91.2
SPP	288	-	88.7
K-SVD	504	570	93.2
LC-KSVD1	504	570	93.7
LC-KSVD2	504	570	93.4
SRC	504	1216	86.1
MSR	477	1216	**96.5**
LGE-KSVD	477	570	95.7/95.7 (1)

TABLE 7.4. 5-CLASS GEMEP-FERA EMOTION DATASET. MHI MOTION VECTORS. 155 TRAINING VIDEOS, 134 TESTING VIDEOS.

Method	*d*	*m*	*% Accuracy*
PCA	154	-	55.2
LDA	4	-	55.2
LPP	154	-	55.2
NPE	66	-	56.7
SPP	75	-	52.2
K-SVD	1920	75	51.5
LC-KSVD1	1920	75	53.7
LC-KSVD2	1920	75	51.5
SRC	500	155	57.5
MSR	154	155	56.0
LGE-KSVD	154	75	**58.2/61.2** (9)

The results in Tables 7.1-7.5 show impressive accuracy performance of LGE-KSVD across a wide variety of problem sets. We attribute this to the discriminative strengths of dimensionality reduction, the classification power of SR methods, along with the integration of LGE into the K-SVD dictionary learning architecture.

When SR methods have insufficient training exemplars in Φ, their performance lags behind SVM classification methods. When datasets are posed, LDA dimensionality reduction is preferred; when datasets are natural, semi-supervised LPP or NPE methods are preferred. The LGE-KSVD representation offers the discriminative properties of LDA while maintaining the local topology of complex data representations in the low dimensional manifold representations. As such, LGE-KSVD has been shown to be robust over datasets with few vs. many classes, high vs. low dimensionality, posed vs. spontaneous faces, static vs. temporal features, and across the classification problems of facial expression, facial recognition, and human activity recognition.

TABLE 7.5. 12-CLASS I3DPOST MULTI-VIEW ACTIVITY RECOGNITION DATASET. 512 TRAINING VIDEOS, 256 TESTING VIDEOS.

Method	*d*	*m*	*% Accuracy*
PCA	510	-	94.9
LDA	510	-	94.5
LPP	510	-	96.1
NPE	224	-	94.9
SPP	241	-	91.0
K-SVD	767	450	94.1
LC-KSVD1	767	450	95.3
LC-KSVD2	767	450	93.8
SRC	767	512	88.7
MSR	510	512	95.3
LGE-KSVD	510	450	**96.9/96.9** (1)

7.8 Analysis of LGE-KSVD

By utilizing a semi-supervised variant of LGE in (3.26), we get excellent results across posed and natural datasets. Fig. 7.5 shows the effect of the α blend parameter used in (3.26). A value of α=1.0 is fully supervised, and a value of α=0 is fully unsupervised. We have consistently found that LGE-KSVD using (3.26) is quite robust for all $0.1 \leq \alpha \leq 0.9$.

The LGE-KSVD framework was designed to minimize the number of iterations- ideally the goal is to have one iteration. Since (7.2) is not directly solvable, we need to solve iteratively for the SR and LGE parameters. Table 7.6 shows the percent improvement from the first iteration of LGE-KSVD to the stopping condition for each of the five datasets. The GEMEP_FERA dataset has the lowest starting accuracy, and thus the potential for largest improvement. The number of LGE-KSVD iterations is often small, as the LGE-KSVD method converges quickly.

Fig. 7.6 shows the effect of the dictionary size *m* on the i3Dpost multi-view dataset. While the performance of other techniques decreases noticeably with smaller dictionary sizes, LGE-KSVD remains robust to dictionaries as small as *m*=50. We attribute this to the clustering abilities of LGE before performing SR classification.

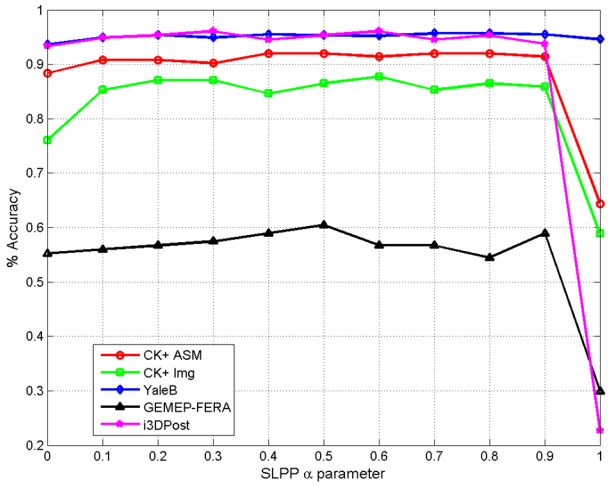


Figure 7.5. Accuracy of LGE-KSVD as a function of adjacency matrix W parameter α in (3.26).

TABLE 7.6. ACCURACY IMPROVEMENTS BY LGE-KSVD ITERATIONS.

Dataset	*Number of iterations*	*% Accuracy improvement*
CK+ ASM	2	2.00
CK+ Img	5	0.70
YaleB	1	0
GEMEP-FERA	9	5.15
I3DPost	1	0

Equation (7.2) minimizes the reconstruction error in a dimensionality-reduced space, while our end goal is to minimize the classification error. One benefit of successive reductions in reconstruction errors is that the input data is more faithfully represented and generally yields lower classification errors. Fig. 7.7 demonstrates this effect over eight iterations of the LGE-KSVD algorithm on the CK+ dataset using image pixels. Each iteration is separated by gray vertical bars and contains 20 modified K-SVD iterations. The blue line shows the RMSE reconstruction error. The numbers in (·) show the resulting classification accuracy at the end of each LGE-KSVD iteration. While these numbers are not monotonically decreasing, there is a strong trend towards achieving better classification accuracy with each iteration.

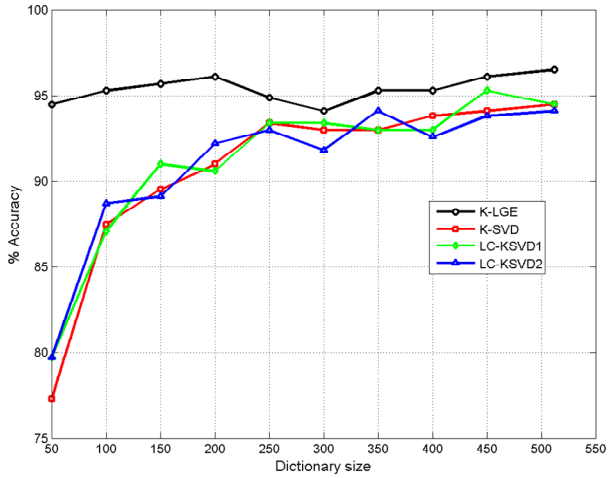


Fig. 7.6. Performance of the four K-SVD methods as a function of dictionary size on the i3DPost dataset.

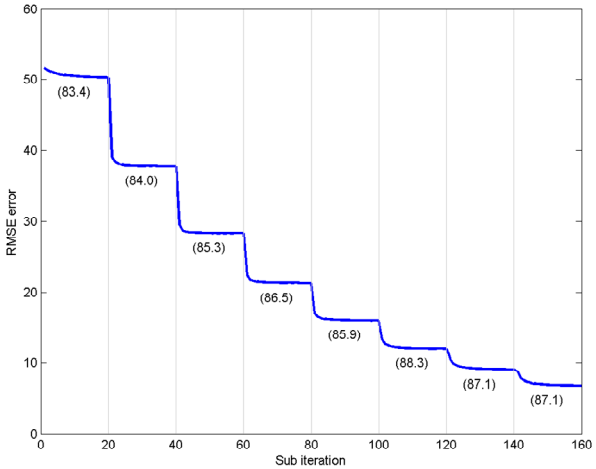


Fig. 7.7. RMSE Error across 160 sub iterations of LGE-KSVD on the CK+ dataset using image pixels. Each iteration of LGE_KSVD has 20 K-SVD iterations. Each vertical line denotes the end of an iteration of the LGE-KSVD procedure with the corresponding test set classification accuracy denoted in parenthesis.

The restriction of dictionary element support in the K-SVD algorithm was devised by [131] to

ensure the solutions of each element remained sparse; this is desirable in order to achieve an optimal solution with an overcomplete dictionary. The modified K-SVD algorithm presented in this thesis enables the support of each dictionary element to grow or shrink with each call to SVD. While this enables LGE-KSVD to converge faster to minimal reconstruction errors, we may ask whether this is done at the risk of sacrificing sparsity. In our experiments, we demonstrate that across varied datasets sparsity is maintained. Further, because we use (5.5) to calculate sparse coefficients, given the current dictionary estimate at the beginning of each modified K-SVD iteration, sparsity is kept in check.

To control sparsity using the modified K-SVD algorithm, the τ value in (3.23) should be kept high. This enables the $\boldsymbol{W}$ matrix to more easily find and discriminate between similar neighbors, and only the most similar are kept in each iteration. During dictionary initialization, each element is initialized with training samples from a single class. With each successive iteration, only training neighbors that are of the same class or have small distances are considered similar, and only similar training samples contribute toward each dictionary element.

Figs. 7.8-7.9 demonstrate the change in support across twenty modified K-SVD iterations. The restrict mode is in red, add is in green, subtract is in blue, and the fixed default is in black. The mean support for the restrict, add, and subtract method per each iteration is shown in Fig. 7.9 and overlaid in thicker solid colored lines in Fig. 7.8. The support can go as high as the number of training samples, which for the CK+ dataset is 164 samples, meaning that a dictionary element can, in theory, be a linear combination of all 164 samples. With 67 dictionary elements, there are 67 updates per each modified K-SVD iteration. Fig. 7.9 shows 20 modified K-SVD iterations (1340 steps in Fig. 7.8). After an initial increase in support (or decrease in sparsity), each of the modified K-SVD methods level off after just a few iterations. Similar results are found on the other datasets.

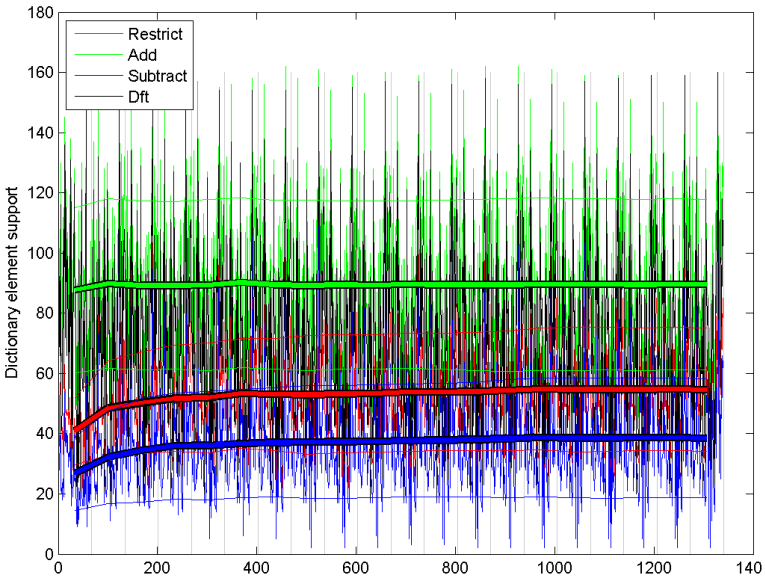


Fig. 7.8. The change in dictionary element support for the CK+ dataset using image pixels across 20 modified K-SVD iterations in the restrict, add, subtract, and default modes. The dictionary size is 67, yielding 67 steps in each of the 20 iterations.

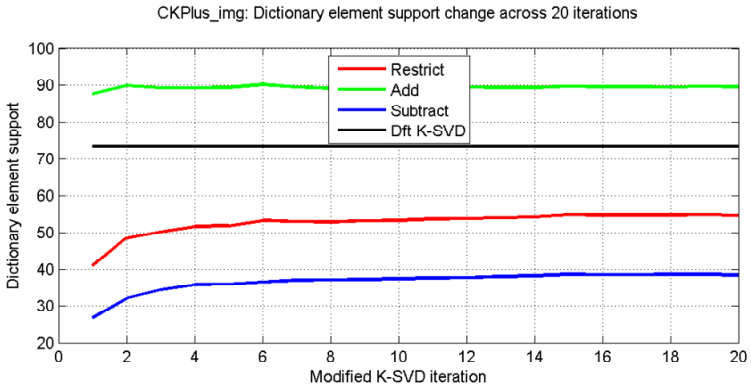


Fig. 7.9. The mean change in dictionary element support for the CK+ dataset using image pixels across 20 modified K-SVD iterations in the restrict, add, subtract, and default modes.

To provide better insight on the algorithm operation during each K-SVD iteration, Fig. 7.10 shows the first 4 modified K-SVD iterations from Fig 7.8. Fig. 7.8 used a semi-supervised $\boldsymbol{W}$ matrix to initialize the dictionary by setting α=0.5 in (3.26). To clarify this point, Fig. 7.10 uses only the LDA component, by setting α=1.0 in (3.26). The CK+ dataset has seven classes. The 67 element dictionary is initially divided amongst those classes to reflect the distribution in the training set. Fig. 7.10 shows these classes sorted in the first iteration with the numbers in (·) above the red curve indicating the breakdown of the distribution of elements for each class. Class one was given 9/67 dictionary elements, class two was giving 4/67, etc. During the first iteration, the restrict mode constrains elements to belong to the initial

class with the height of each bar reflecting the number of training samples for each class. There are 167 training samples to pick from- the sum of the seven bar heights in the first iteration.

In between each iteration, we re-solve for the coefficients $\boldsymbol{A}$ given the most current dictionary estimate. In the second iteration, one of the nine dictionary elements for class one had significant energy (defined by magnitude of coefficient) from a training sample from class seven. As such, this particular dictionary element was expanded to include this new training sample, thus expanding its support to samples from class one and seven. With each successive iteration, elements that are similar to one another are added or removed to make the most efficient use of the dictionary. We use τ=100 for element neighbor similarity $\boldsymbol{W}$ to ensure the support of each dictionary element remains small. After five or six iterations, the activity settles down and only minor changes occur. After 20 modified K-SVD iterations, the clear delineation between classes in the first iteration is all but replaced with a more efficient representation that more clearly represents the topology of the training set.

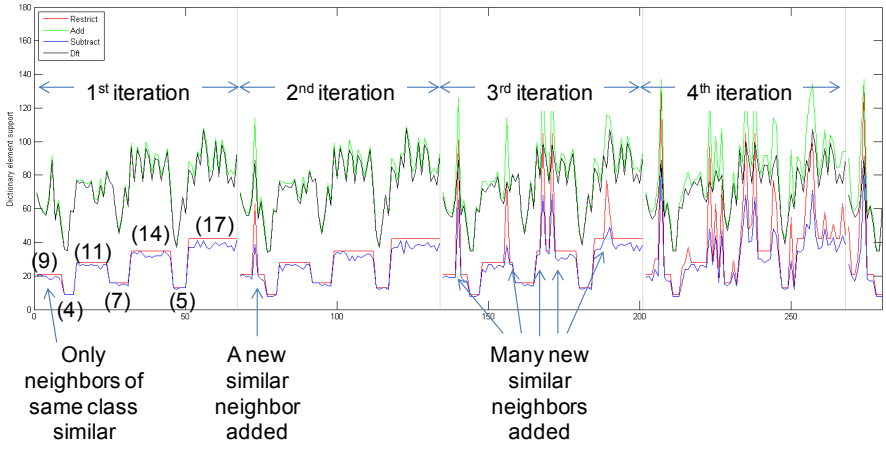


Fig. 7.10. The change in dictionary element support for the CK+ dataset using image pixels across 4 modified K-SVD iterations in the restrict, add, subtract, and default modes.

8 Conclusions

The task of training machines to interpret human face and gestures requires accurate object detection (finding the faces and/or people), robust normalization (warping faces or body to a canonical representation), feature extraction (salient feature point extraction and/or image pixel representations), and subject classification. This thesis contains both theory and experiments that explore each of these facets, and concentrates on the last step of subject classification. Accurate subject classification is predicated upon the extraction of discriminative features and robust statistical reasoning.

This thesis research finds that dimensionality reduction methods can further the fields of face and gesture analysis by simultaneously reducing compute complexity and increasing discriminative properties for improved classification inference. Specifically, Locality Preserving Projections (LPP), which is a linear approximation to the non-linear Laplacian Eigenmap manifold, has consistently demonstrated superior performance as compared to other dimensionality reduction methods. Results demonstrate that LPP can be beneficial in continuous problem spaces, such as facial pose, discrete problem spaces, such as facial expression, as well as hybrid problem spaces, where the two metrics are combined into a single manifold.

Motivated by the successes of Sparse Representations (SRs) in many scientific fields, SR concepts were applied to the field of face and gesture understanding. It was found that coefficient contamination limited SR classifiers to posed datasets where a single dominant signal prevailed. When used in a discriminative fashion, supervised dimensionality reduction methods can encourage signals to cluster based upon a primary signal of interest, reducing secondary signatures to a minimal. Unfortunately, this research found this clustering weakens the system's ability to perform in natural conditions where generalization is paramount. Using an architecture called Manifold based Sparse Representations (MSR), results show that using semi-supervised dimensionality reduction in conjunction with sparse representations produces sufficient discrimination between input classes to minimize coefficient contamination, while preserving enough geometric detail from the high dimensional space for real world imagery. Experiments show that MSR produces excellent results across a broad variety of facial understanding and gesture recognition problems in the spatial and temporal domains.

An advanced framework called LGE-KSVD is introduced that advances the MSR framework by co-optimizing dimensionality reduction matrix $\boldsymbol{U}$, dictionary Φ, training coefficients $\boldsymbol{A}$, and coefficient transformation matrix $\boldsymbol{C}$ through successive iterations. LGE-KSVD further leverages LGE

dimensionality reduction concepts to modify the K-SVD dictionary learning algorithm such that the support of dictionary elements remains sparse, but is no longer fixed.

The modification of the dictionary element update step in K-SVD improves reconstruction and classification accuracy while making K-SVD more generally applicable to a broader spectrum of problems and less reliant on a good initialization. Results demonstrate that the proposed framework provides significant advantages over other state-of-the-art techniques across a wide variety of facial and activity recognition problems.

Future Research

The proposed MSR and LGE-KSVD frameworks have been shown to be robust across datasets with few vs. many classes, static vs. temporal attributes, posed vs. natural tendencies, and across a broad spectrum of facial understanding, gesture recognition, and activity recognition. Future work needs to expand this research to new disciplines such as medical, bioinformatics, entertainment, and robotics fields. For example, it would be very interesting to see how LGE-KSVD would perform on the task of medical diagnosis or gene therapy.

Deploying the proposed techniques in the field might require classification beyond tens of classes as done in this thesis, up to hundreds, thousands, or even tens of thousands of classes simultaneously. It would be interesting to explore how the proposed frameworks might scale with the number of classes. Although Section 4.3 showed that the mixture of two concepts (pose and expression) can be merged successfully into the same framework, it seems unwieldy to consider implementing the limitless classification and reasoning abilities of the human brain in this framework. The study of parsimony in the human brain may lead to important clues on how to make this happen.

Sparsity is in effect very parsimonious in that complex signals can be represented as a linear combination of simple basis functions. It is not clear what the best organization is for these basis functions, although recent deep learning experiments collaborate with brain researchers in believing a hierarchical framework may be one plausible solution. For example, images from the visual cortex may be decomposed to a summation of Gabor filters in the most primitive layer, followed by edges in the next layer, followed by corners, curves, and textures in the next, followed by object building blocks in the next, followed by object parts, followed by objects themselves. It would be very interesting to implement the proposed frameworks in a hierarchical fashion, to see if they can be more broadly disseminated across varied problem sets.

Because the MSR and LGE-KSVD techniques are models which require upfront training, it may be difficult to incorporate either framework into autonomous agents which need to learn over time. The

investigation of using alternate architectures, or borrowing concepts from methods that don't require a full retraining each time new exemplars arise would make the proposed research more applicable to a broader spectrum of applications.

With regards to sparse dictionaries, it would be interesting to develop a method that not only allows dictionary elements to change over time, but allow dictionaries that grow and shrink over time. For example, a hand writing recognition system may initially be trained for the average user. Upon deployment into an office, the dictionary should learn specific traits from its users. If one worker should use the system less over time, the system should eventually deemphasize that person's handwriting strokes and perhaps devote more attention to the frequent users. Or, if all users stopped using the handwriting system, but instead started using a voice recognition system, the hand writing dictionary would shrink back to only a necessary set of basis functions, while the voice recognition dictionary would grow in proportion to its frequency of usage. The current K-SVD dictionary learning framework does not support weights on training samples, the addition of such may help achieve living dictionaries.

Appendix I- Datasets Used

CAS-PEAL , http://www.jdl.ac.cn/peal/index.html: The CAS-PEAL-R1 [18] face database was collected under the sponsor of the Chinese National Hi-Tech Program and ISVISION Tech. Co. Ltd.. This database contains 1040 subjects, 445 females, 595 males, most of Mongolian descent. Seven simultaneous cameras captured seven yaw angles (-45:15:45 degrees). Each subject was instructed to look down, straight ahead, and up as three sets of seven photographs were taken at three different pitch angles (-30, 0 , +30 degrees). An example subject at the 21 resulting pose angles is shown in Figure 2.1.

Cohn-Kanade, http://www.pitt.edu/~jeffcohn/CKandCK+.htm: The Cohh-Kanade (CK) [111] face database was collected by Carnegie Mellon University. This database contains 92 subjects in 229 expression sequences, each displaying one of six universal expression sequences. The expressions were anger, disgust, fear, sad, happy, and, surprised. Each sequence was 10-25 frames of video going from neutral to fully articulated expression. Figure 3.2 is an example of subject exhibiting the five of the expressions from the Cohn-Kanade (CK) dataset.

Extended Cohn-Kanade (CK+), http://www.pitt.edu/~jeffcohn/CKandCK+.htm: The Extended Cohn-Kanade (CK+) [58] dataset is an expanded version of the CK database along with specific testing protocols in an effort to standardize facial expression benchmarking efforts. The dataset contains 118 subjects in 327 expression sequences. In addition to the 6 universal expressions, the CK+ dataset includes the contempt expression. Figure AI.1 shows samples of the 7 expressions from the CK+ dataset. One unique aspect of the CK+ database is that any expression sequence that appeared to be fake or unidentifiable by a panel of judges was excluded. The CK+ benchmarking protocol stipulates a leave-one-subject-out cross-validation yielding 118 different training and testing sets. Only the last frame of each sequence is used for each expression. The first frame may optionally be used as a reference frame if neutral frame subtraction is necessary.

Fig. AI.1. Example of 7 facial expressions from the CK+ dataset.

Extended YaleB, http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html: The Extended YaleB facial recognition dataset [155] contains 2,414 frontal images of 38 people under varying illumination and facial expression. Each face is 192x168 pixels which are reduced to D=504 via random projections following [78]. The test set contains 1216 training faces and 1198 testing faces.

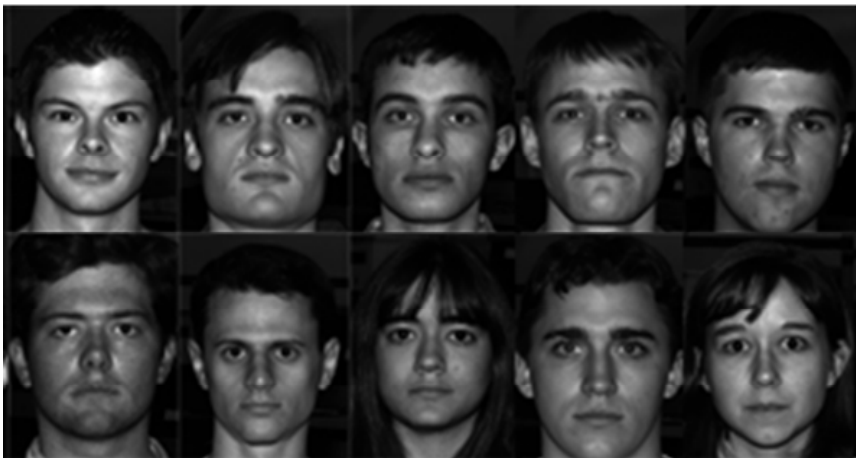


Fig. AI.2. Example faces from the YaleB dataset.

GEMEP-FERA, http://gemep-db.sspnet.eu/: The GEMEP-FERA [139] dataset was introduced at the 9th IEEE International Conference on Automatic Face and Gesture Recognition in 2011. Also referred to as FERA2011, it was organized to help researchers compare and benchmark facial action unit (AU) and emotion recognition classification algorithms. The emotion portion of the dataset consists of 10 actors exhibiting the five emotions of anger, fear, joy, relief, and sadness. The training set contains 7 actors over 155 videos. The test set contains 6 actors (half of which were not present in the test set) over 134 videos. Training video sequences varied from 20-128 frames, with a median of 56 frames. Testing video sequences varied from 26-150 frames, with a median of 51 frames. During most video sequences, subjects uttered one of two pseudo-linguistic phoneme sequences. For the remaining sequences, subjects uttered the sustained vowel 'aaa'.

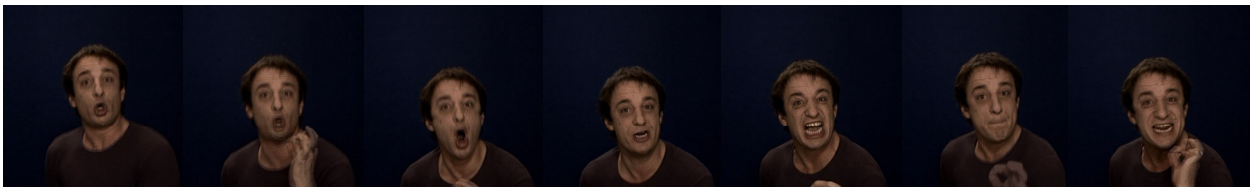


Fig. AI.3. Sample frames from Test 44 of the GEMEP-FERA dataset.

i3DPost, http://kahlan.eps.surrey.ac.uk/i3dpost_action/: The i3DPost multi-view [156] activity recognition dataset contains 768 videos of 8 people performing 12 actions from 8 views. The 12 activities are walk, run, jump, bend, hand-wave, jump in place, sit-stand, run-fall, walk-sit, run-jump-walk, handshake, and pull.

Fig. AI.4. Eight views from a sample frame from the i3DPost dataset.

LFW, http://vis-www.cs.umass.edu/lfw/: The LFW [140] dataset contains 13,233 images of 5,749 unique faces. Each face has been downloaded from the web, and as such is representative of an unconstrained natural pose. Although all faces were found by the Viola-Jones face detector, considerable variability in pose, expression, fidelity, and occlusions exist. The LFW website only releases ground truth for identity, but we have collected ground truth for race {Asian, Black, Caucasian, Hispanic, Indian, other}, gender {female, male}, spectacles {none, reading, eye, sun}, and facial hair {none, beard and mustache, beard only, mustache only, goatee, other}. In the race and facial hair classification results, faces classified as other (< 5% of samples), were removed from the analysis. Although gender and glasses are generally agreeable by most judges, there is considerable disagreement over race and facial hair on many samples.

Fig. AI.5. Three subject pairs from the LFW dataset.

Posed vs. Natural Datasets

In the past few years, the academic community has started to concentrate less on posed datasets and more on spontaneous datasets. Posed datasets are characterized by subjects being instructed what expressions or gestures to mimic, and then they are recorded under tightly controlled conditions. Figure AI.6 is an example of a subject exhibiting five expressions from the Cohn-Kanade (CK) [111] dataset. Some may

say the expressions in this dataset are either not representative of real-life expressions, or they are exaggerated. The extended Cohn-Kanade (CK+) [58] dataset addressed the former issue by excluding fake or unidentifiable expressions by a panel of judges, but not the latter.

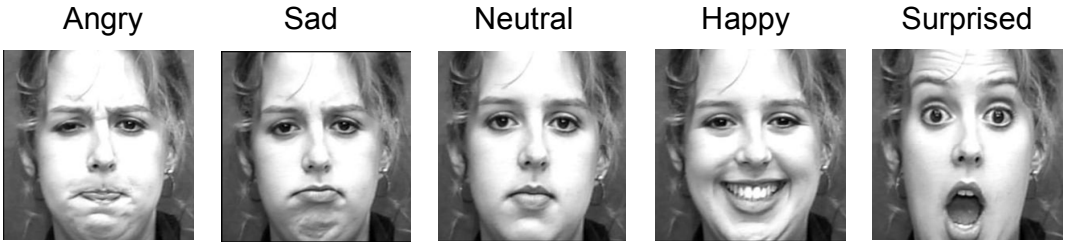


Fig. AI.6. Sample subject from the Cohn-Kanade dataset exhibiting 5 facial expressions.

Not only are the faces in Figure AI.6 exaggerated, they are all frontal poses, taken with identical illumination, and against a neutral background. Although LFW was not the first natural dataset, it was by far the largest at the time and spawned new excitement. Since the release of LFW, several new datasets such as FGnet, GEMEP, Morph-II, Cave, and PubFig all attempt to be more realistic by using spontaneous, or non-posed imagery.

Cross Validation

Machine learning methods learn patterns from training exemplars. These developed methods are then evaluated on test sets to determine performance. If the developed methods fail to generalize to new test samples not in the training set, the resulting performance will be poor. To minimize the risk of over-training and facilitate procedures for fair and equal comparison of alternate processing methods, datasets are broken down into segments or blocks of exemplars used for training, verification, and testing.

To motivate this discussion, the following toy example consists of 19 training points which relate the yaw of the human face to a distance scalar used in an HCI device. Given some unforeseen yaw value, we would like to predict the scalar. A least squares regression model, h_θ is fit to the training data, where θ represents the degree of the polynomial. Figure AI.7 shows the yaw on the x-axis, and HCI scalar on the y-axis. The left most plot is a linear least-squares fit θ=1, the other three plots have θ=3, 10, and 18 going left to right. The model error can be computed as:

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2 \qquad (AI.1)$$

The resulting model error for the for θ values [1, 3, 10, 18] is [10.9, 10.2, 3.0, and 0]. It might be tempting to use the model with θ=18; this model exactly matches each of our training samples. Hopefully it is readily evident that this model is fitting to measurement noise and will not generalize well to new test samples.

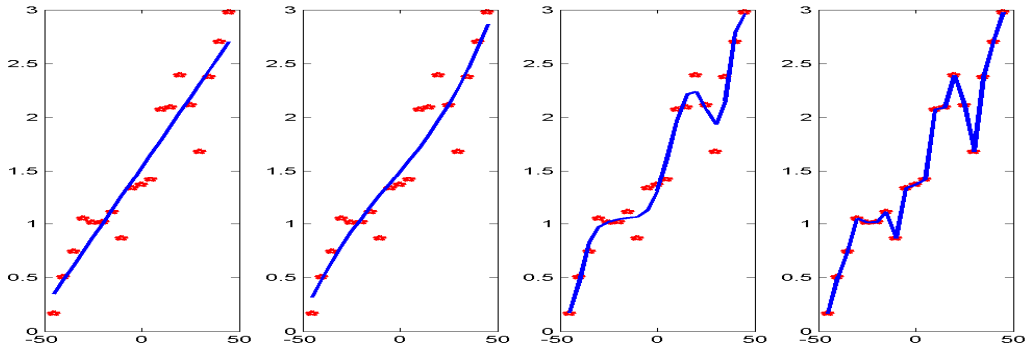


Fig. AI.7. Sample data fit with polynomials of order = 1, 3, 10, and 18 degrees.

The common solution is to randomly partition the given data into a training set, evaluation (or cross validation) set, and test set. The training set generally comprises 60% of data, and the evaluation and test sets each comprise 20% of the data. The idea is to build the model on the training set, and optimize its performance on the evaluation set. Then, the report the final model accuracy on the test set. To ensure fair comparison amongst independent researchers, identical dataset partitions need to be specified. When this approach is used on this toy dataset, a polynomial degree of three gives the lowest error on the test dataset.

Regarding over fitting of data, the terms bias and variance are used. A model that is under fit is said to have high bias; while a model that is over fit is said to have high variance. Figure AI.8 demonstrates the typical performance one may expect. As the model complexity increases (degree of polynomial in our example), the training error continually decreases monotonically, but the test error has a minimal value. The model selection process selects the model which corresponds to minimal testing error.

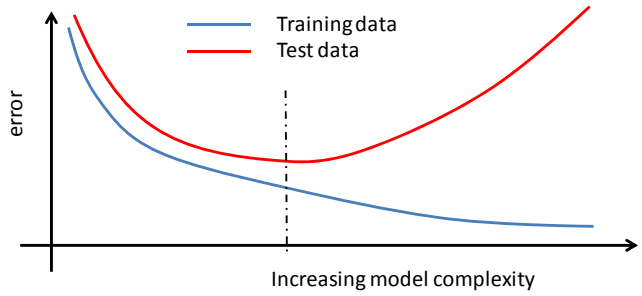


Fig. AI.8. Typical relationship between model complexity and model error for training data (blue) and test/evaluation data (red).

In practice, the partitioning of the dataset into training, evaluation, and testing is often simplified to just a training and testing subset. However, neither of these methods ever utilize testing samples for training purposes. In many studies, collection of ground truth data is labor intensive, and it seems wasteful to not use all the data available. The solution is to adopt a cross validation strategy. In k-fold cross-validation, the data is split into *k*-folds, whereby folds 2… *k* are used for training and fold 1 is used for testing, then the process is repeated using folds 1,3,…*k* for training and fold 2 for testing, and so on. When done, the average of the *k* train/test partitions is reported.

In facial analysis, a dataset may consist of *p* subjects exhibiting *q* expressions. As such, there is often a strong correlation between identity and expression, and researchers use what is referred to as leave-one-subject-out cross validation. This necessitates that the data be split into *p* folds, where partition boundaries are done by subject. Folds 2… *p* (corresponding to subjects 2…*p*) are used for training and fold 1 (corresponding to subject 1) is used for testing, then the process is repeated using folds 1,3,…*p* for training and fold 2 for testing, and so on. When done, the average of the *p* train/test partitions is reported.

Cross-validation not only is necessary for benchmarking, but it also determines the appropriate amount of model complexity or the appropriate tradeoff between bias and variance. Once the model complexity is determined, it is often common in industry to use all training data to build one final model at the determined complexity for implementation into say a commercial application.

Error Metrics and Confusion Matrices

For continuous regressions, Root Mean Squared Error (RMSE) error over the test set or average cross validation sets are an obvious choice for reporting errors. For discrete problems, errors are determined by the number of correct detections or classifications. For an object detector, say a face detector, we need to determine how many of the potential faces were correctly classified, and how many of the faces were missed by the classifier. Figure AI.9 describes the possible outcomes from such a classifier.

		Predicted Class	
		Face	**Non-Face**
Actual Class	**face**	TP (true positive)	FN (false negative) (missed a face)
	Non-face	FP (false positive) (found non-face)	TN (true negative)

Fig. AI.9. Possible outcomes for object detection such as face detection.

Referring to Figure AI.9, for each face that is found, we first refer to the Face column in the predicted class. Found faces can fall in the true positives (TP) category (faces that were correctly found) or false positive category (the face detector is saying it found a face where there is none). Similarly, for each face that is exists in the ground truth data set, we refer to the Face row in the actual class. Actual faces can fall in the true positives (TP) category (actual faces that were correctly found) or false negative category (actual faces that were missed). As such, by running our face detector over our training set, we produce the three values of:

TP: Predicted face corresponds to an actual face;

FP: Predicted face where there is none;

FN: Actual face that was not found by the detector.

There is also a forth category of true negative, or TN. This would be non-face locations that were (correctly) not identified as a face region by the detector. This category is ignored as its meaning carries no significant value. The three most common metrics used for object detectors are:

$$Recall: \quad \frac{TP}{(TP + FN)} \tag{AI.2}$$

$$Precision: \quad \frac{TP}{(TP + FP)} \tag{AI.3}$$

$$F1: \quad \frac{2 * Precision * Recall}{(Precision + Recall)} \tag{AI.4}$$

Where:

Recall describes what percentages of real faces are found by the detector;

Precision states, of the detected faces, what percentage are actually real faces; and

F1 is a value that measures the classification performance in a single number.

When performing classifications, we use a similar concept as the table in Figure AI.9, but now, for each object, we need to determine which of the k classes it belongs to. In this case, each object in our training set has an actual or ground truth class, and a predicted class as determined by the classifier. Figure AI.10 shows this for the case of a two-class gender detector. The values on the diagonals are correct classifications by the gender classifier, and values off diagonals are incorrect classifications. For example the value in the upper right corner of the confusion matrix in Figure AI.10 represents female test subjects that were incorrectly classified as male by our gender classifier. The extension of the confusion matrix to k>2 classes is identical. To report classification accuracy, we have:

$$Accuracy: \quad \frac{\sum_i C_{ii}}{\sum_{ij} C_{ij}} \qquad (AI.5)$$

Where $\boldsymbol{C}$ is the confusion matrix and the numerator in (AI.5) is the sum of the diagonal elements, and the denominator in (AI.5) is the sum of all elements.

		Predicted Class	
		female	**male**
Actual Class	**female**	Correct	Incorrect
	male	Incorrect	Correct

Fig. AI.10. A two class confusion matrix.

Appendix II- Pixel Processing

The studies in this report utilized several types of pixel processing and normalization. Motivated by anthropometry of the human face [30] depicted in Fig. A.II.1, schematic bounding boxes used for the human face used in this thesis are described in Fig. A.II.2. Typical face detectors use the square cropping strategy shown on the left of Fig. A.II.2. For expression classification, this bounding box omitted important mouth and chin areas, especially if the mouth is open. As such, the center bounding box is used in this paper, but, there is no such standard. For initial studies done on age, race, and gender, the hair is a also a critical component. As such, the right-most bounding box is used, but, once again, there is no widely adopted standard.

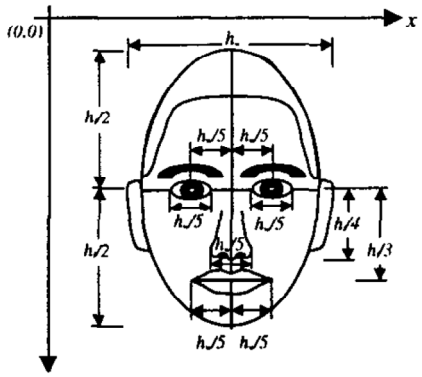


Fig. A.II.1. Anthropometry of the human face as reported by [30].

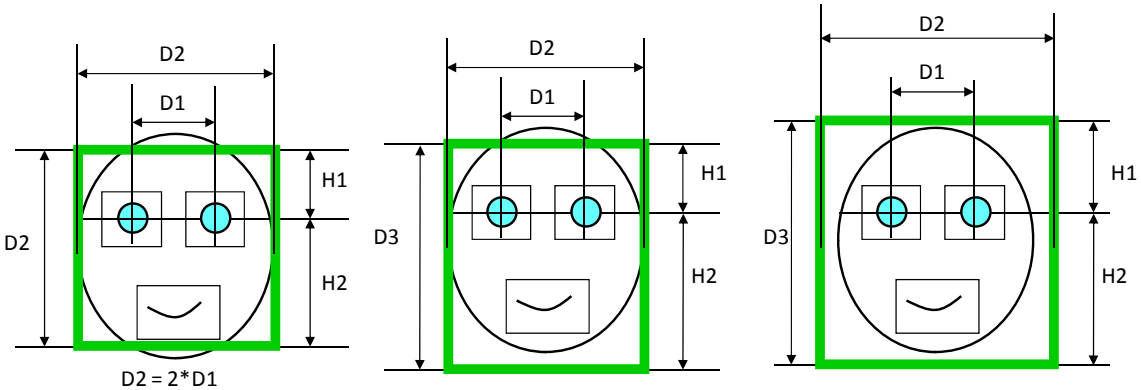


Fig. A.II.2. Three variants of facial bounding box. Left is for face detection, center is for expression recognition, and right is for gender, age, race detection.

Pixel Processing Techniques

None: Use the RGB or grayscale pixels unaltered

Luminance: Convert RGB pixels to grayscale using, lum=(0.2989*R + 0.5870*G + 0.1140*B)

Sobel: Edge detector that computes an approximation to the grayscale gradient. The following vertical and horizontal edge detection kernels are convolved with the image:

$$S_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}; \quad S_h = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

The final image is computed by combining the two filtered images with:

$$\boldsymbol{I' = \sqrt{(I * S_v)^2 + (I * S_h)^2}}$$

Edge Magnitude: Same as Sobel filter, but each pixel in the edge image is passed through a 1/2.4 gamma 1D look-up table to make the existing range fall within 0:255.

Edge Phase: The phase of the image is computed using:

$$\boldsymbol{\theta = \tan^{-1}\left(\frac{I * S_v}{I * S_h}\right)}$$

Where tan-1 is the quandrant specific (atan2) function. The pixels are then linearly scaled from -512:512 to 0:255.

Sharpened: CPU friendly sharpening kernel:

$$\frac{1}{64}\begin{bmatrix} -8 & -32 & -8 \\ -32 & 224 & -32 \\ -8 & -32 & -8 \end{bmatrix}$$

LOG: Laplacian of Gaussian Filter. Look for zero order crossings of 2nd derivative calculated via the approximation 5x5 convolution kernel of all -1's except for a 24 in the center tap.

Canny: Canny Edge detection algorithm which 1) blurs the image; 2) computes 1D derivatives; 3) Sum of square of derivatives; 4) Threshold image using hysteresis thresholding, where hysteresis thresholding starts from one corner of the image, visit pixels until one exceeds an upper threshold, then follow chains of maxima along edges until value drops below lower threshold; mark and save all visited values as a connected contour.

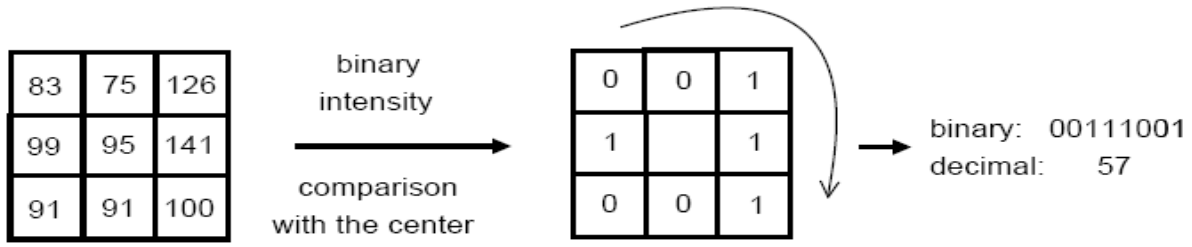


Fig. A.II.3. $LBP_{8,1}$ conversion from a 3x3 pixel neighborhood to a single pixel value of 57.

LBP: Local Binary Patters are a family of texture extracting filters for pattern matching. The full notation is $LBP_{8,1}^{u2}$ local binary pattern. The $_{8,1}$ says we are comparing the center pixel to the 8 nearest neighbors, radius=1. It has been shown that uniform patterns (patterns with <= 2 transitions) contain most of the salient information. If all patterns > 2 transitions are mapped to a single code value, we can represent LPB images in a compressed fashion. In this instance, the 256 unique LBP patterns are reduced to 56 codevalues. The ^{u2} signifies converting the 256 value LBP to these 56 salient combinations.

LBP-h: Calculate the histogram of LBP values for *mxn* tiles across the image. The histogram of each tile concatenated as a 1D vector is the feature space used for classification. Figure 2.6 pictorially shows an example of a concatenated block histogram.

Gabor: Gabor banks are multi-phase, multi-frequency filtered images based upon studies of the human visual response function. The Gabor filter is essentially a rotated sinusoid attenuated by an exponential function. Formally, the real component of a Gabor image is filtered via a kernel described by:

$$G_{\emptyset,\eta}(x,y) = exp\left(-\frac{\left(x_{\emptyset}^2 + \zeta^2 y_{\emptyset}^2\right)}{2\sigma^2}\right)\cos\left(\frac{2\pi}{\eta}x_{\phi}\right);$$

$$x_{\phi} = x\cos\phi + y\sin\phi\,;\;\; y_{\phi} = -x\sin\phi + y\cos\phi$$

Gabor filtered images in this paper are a single channel weighted sum of four Gabor processed ϕ= (0^o, $\pi/4^o$, $\pi/2^o$, $3\pi/4^o$) images, all at a single frequency of η= 0.3 cyc./sample. The 4 planes are summed to a single plane by weighting the 0 and $\pi/2$ planes by 1/3, and the π /4 and 3π /4 planes by 1/6. Figure A.II.4 shows an input image, its 4 Gabor planes, and the final weighted output on the right.

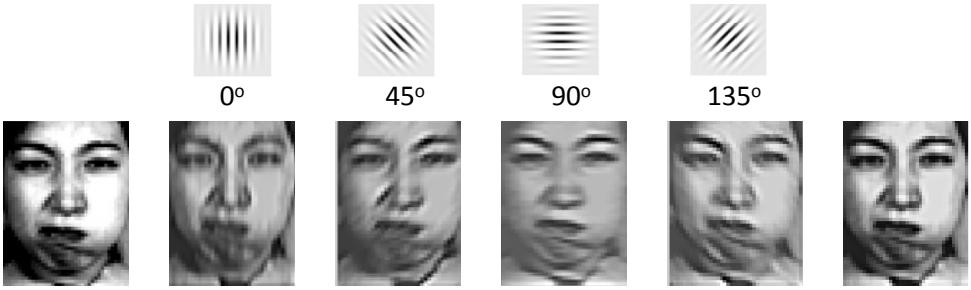


Fig. A.II.4. Sample face on the left. Four Gabor planes in the center. Weighted Gabor plane on the right.

Normalization Methods

None: Use the processed pixels as is.

Norm: Calculate the mean, μ and standard deviation, σ^2 of the image area ***I*** (for example, this can be the entire image, the extracted face area, or a masked face region), then calculate:

$$Norm = \frac{(I - \mu)}{\sigma^2} * 100 + 128$$

Mean: Divide each pixel in an image area by the mean, μ, of the entire image area.

Appendix III- Classification Methodologies

Given a set of input observables and corresponding desired responses, linear regression models the data in a continuous sense and logistic regression models the data in a discrete sense. As examples of each, linear regression would be ideal to model the relationship between attributes of the face and facial pose, while logistic regression would be more suitable to model the relationship between attributes of a face and facial identity (a discrete classification problem).

Linear Regression

In linear regression, a function with variable parameters is fit to the data. Perhaps the simplest example is a polynomial regression of a single variable as seen in Figure AI.7. Our model, of polynomial degree c will be of the form:

$$h_\theta(x) = \sum_{i=1}^{c} \theta(i) x^i \qquad (AIII.1)$$

The vector $\boldsymbol{\theta}$ is solved in a linear least squares sense, minimizing the error function (AI.1). There are several ways to do this, the two most common are gradient descent and direct computation. After an initial estimate for vector $\boldsymbol{\theta}$ (can be all zeros), gradient descent iteratively updates $\boldsymbol{\theta}$ using:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta) \qquad (AIII.2)$$

Where $J(\theta)$ is the cost function defined in (AI.1). Gradient descent typically converges quickly, but if α is too large, it can converge to a local minimum. The direct computation using the normal equation in linear algebra solves $\boldsymbol{y}=\boldsymbol{\theta x}$ over n datapoints simultaneously. In this fashion, $\boldsymbol{y} \in \mathbf{R}^n$, $\boldsymbol{x} \in \mathbf{R}^{n \times c}$, and $\boldsymbol{\theta} \in \mathbf{R}^c$. The direct solution for $\boldsymbol{\theta} = \boldsymbol{y}\boldsymbol{x}^{-1}$, or more generally the pseudo inverse is used since $\boldsymbol{x}$ is not square, $\boldsymbol{\theta} = \boldsymbol{y}\boldsymbol{x}^T(\boldsymbol{x}\boldsymbol{x}^T)^{-1}$.

The gradient descent method requires the specification of α and may require many iterations, but it works well even when n is large. The direct solution does not require the specification of any parameters, has no iterations, but can be slow as the computation of $(\boldsymbol{x}\boldsymbol{x}^T)^{-1}$ is slow if n is large. In general if $n < 5{,}000$ the direct solution is preferable. There are optimized versions of gradient descent that don't require the specification of α and often converge faster (such as conjugate gradient, BFGS, L-BFGS), but these methods are more complex.

Logistic Regression

Logistic regression is suitable for classification problems, where given a set of parameters, we need to determine which of a finite number of classes our sample belongs. In linear regression, $h_\theta(x)$, our estimate for $\boldsymbol{y}$, can take on any value. Logistic regression solves a binary problem, determining for example if a sample is male or female (or 0 or 1). Multi-class classification is done via one vs. all, where all is the rest of the samples. If there are k classes, we generate k models, and assign the class to the model with the maximum class assignment, $h_\theta^i(x)$, where i=1…k. In logistic regression, we will assume binary classification assignment to either class 0 or class 1 and limit $0 \leq h_\theta(x) \leq 1$.

In linear regression, our model in (A.III.1) can be represented as $\boldsymbol{\theta^T x}$. To limit $\boldsymbol{\theta^T x}$, we apply a function $g(\boldsymbol{\theta^T x})$ such that $0 \leq g(\boldsymbol{\theta^T x}) \leq 1$. While there are a handful of functions that can do this nicely, the sigmoid function is perhaps the most commonly found:

$$g(z) = \frac{1}{1+e^{-z}} \qquad (AIII.3)$$

And our logistic regression model becomes:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \qquad (AIII.4)$$

As defined, $h_\theta(x)$ is continuous, $0 \leq h_\theta(x) \leq 1$. If for example we were trying to determine if a face had facial hair from facial attributes, we might get $h_\theta(x) = P(y=1|\boldsymbol{x};\boldsymbol{\theta})$; or the probability that our sample is 1, given attributes $\boldsymbol{x}$, and model $\boldsymbol{\theta}$, or the probability that our face has hair, given attributes $\boldsymbol{x}$, and model $\boldsymbol{\theta}$. Our model $h_\theta(x)$ may return 0.75, so we can say there is a 75% chance the face has facial hair.

Equation (AI.1) defined the cost function for linear regression, and we need to develop a similar equation for logistic regression. The logistic regression model error starts as:

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n} cost(h_\theta(x_i), y_i) \qquad (AIII.5)$$

Where:

$$cost(h_\theta(x), y) = \begin{cases} -log\big(h_\theta(x)\big) & if\ y = 1 \\ -log\big(1 - h_\theta(x)\big) & if\ y = 0 \end{cases} \qquad (AIII.6)$$

The intuition is straight forward if we first consider that y can only take on 0 or 1. If y=1, we hope our model $h_\theta(x)$ agrees with y, so we hope our model is close to 1. As such if y=1 and $h_\theta(x)$ =1, our

cost is 0. However, if $h_\theta(x)$ =0, our cost must be very high, and in this case, we state our cost as infinite. Figure A.III.1 shows this cost function on the left.

If $\boldsymbol{y}$=0, we also hope our model $h_\theta(x)$ agrees with y, so we hope our model is close to 0. As such if $\boldsymbol{y}$=0 and $h_\theta(x)$ =0, our cost is 0. However, if $h_\theta(x)$ =1, our cost must be very high, and in this case, we once again state our cost as infinite. Figure A.III.1 shows this cost function on the right.

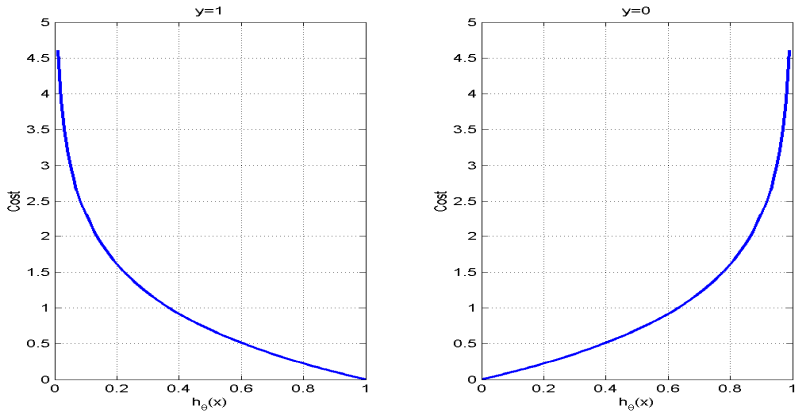


Fig. A.III.1. Logistic regression cost functions for when our ground truth sample, y=1 (on left), and y=0 (on right).

Because each of our samples can only have a class assignment of 0 or 1, we can rewrite our abstract logistic regression cost function in (AIII.5) and (AIII.6) as our new objective function:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_i log\big(h_\theta(x_i)\big) + \ (1-y_i) log\big(1-h_\theta(x_i)\big) \qquad (AIII.7)$$

Note that in (AIII.7) only the first or second term can participate in the summation. Equation (AIII.7) also dropped the 2 in the denominator (for later simplification) and made the summation negative such that this becomes our new objective function we want to minimize. As with linear regression, we need to solve for vector $\boldsymbol{\theta}$. This can once again be done with gradient descent using (AIII.2), and because of the selection of (AIII.7), the solution of which is identical to that done for linear regression, but we substitute $h_\theta(x)= \boldsymbol{\theta}^T\boldsymbol{x}$ with (AIII.4).

k-Nearest Neighbor (k-NN)

The k-nearest neighbor (k-NN) algorithm is a discrete classification scheme that classifies an input sample to be the same class as its nearest neighbor. More generally, the algorithm computes the distance between a test sample and all other training samples, sorts by distance, then assigns the class to the test sample that is the mode of the top *k* nearest neighbors. Often a weight is applied to these neighbors, where the weight is inversely proportional to the distance between the two samples. Further, if we are in

a space where we can assign an importance to each dimension, a whitening operation can be performed by scaling each dimension by the importance value (such as eigenvalue in PCA analysis).

Artificial Neural Nets

The most common type of artificial neural network (neural net for short) is the multi-layer perceptron. Figure AIII.2 shows a schematic of such a network. A variable number of input nodes are input into a multi-layer formation such that the outputs of one layer feed the inputs to the next. The lines connecting layers are weights, and a weighted linear combination of these weights forms the input to each node. Each node applies a non-linear activation function to this input before passing it on to the next layer. This activation function is usually the sigmoid function (AIII.3) or the tanh() function. The output of the network forms the final output values used to determine the class from input attributes. The final output values can use a custom activation function, including the linear activation function, or rounded to a binary value, where the concatenation of all output nodes forms a single binary number.

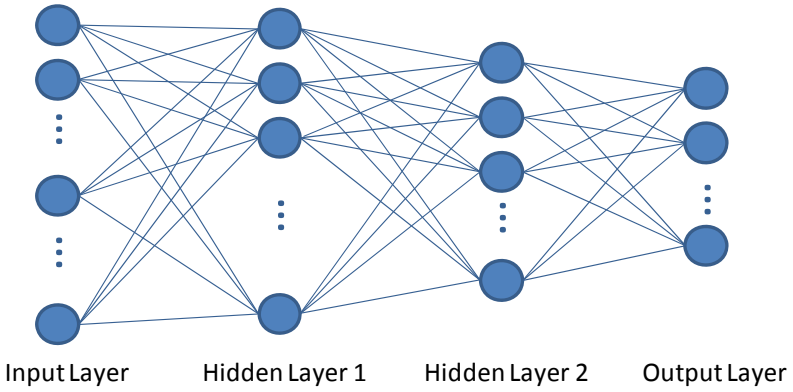


Fig. A.III.2. Schematic of an artificial neural network. Along each line is weight that connects two nodes.

The decision of how many hidden layers and how many nodes per hidden layer is perhaps the biggest drawback of neural networks. There is no robust rule of thumb that applies across varied datasets. The training of neural nets get exponentially slower with the number of nodes.

Training of the neural net involves solving for the weights interconnecting nodes. Starting with a random set of weights, input training samples are passed one at a time, layer to layer, until they get to the output layer, a procedure called feed forward. Once at the output layer, the values at output nodes are compared to the input sample ground truth values. The differences are then passed backwards through the network, updating the weights, through a procedure called back propagation. Once arriving at the input layer, the next sample is passed through the network, repeating the process. Each pass through all training samples is called an epoch. Some neural networks require thousands of epochs before the

weights settle down, taking weeks to classify. Thankfully, most neural nets of reasonable size can be trained in only a few minutes using back propagation.

Support Vector Machines (SVMs)

Like logistic regression, support Vector Machines (SVMs) represent a set of techniques used to do binary classification. As compared to neural networks, SVMs are easy to optimize with minimal parameter tuning. Although the mathematical theory is complex, both the training phase and the resulting classifiers are relatively simple to compute. During training phase, SVM discovers a separating hyperplane that maximizes the margin between samples from two data classes. During classification, SVM predicts which side of the hyperplane the test sample falls on.

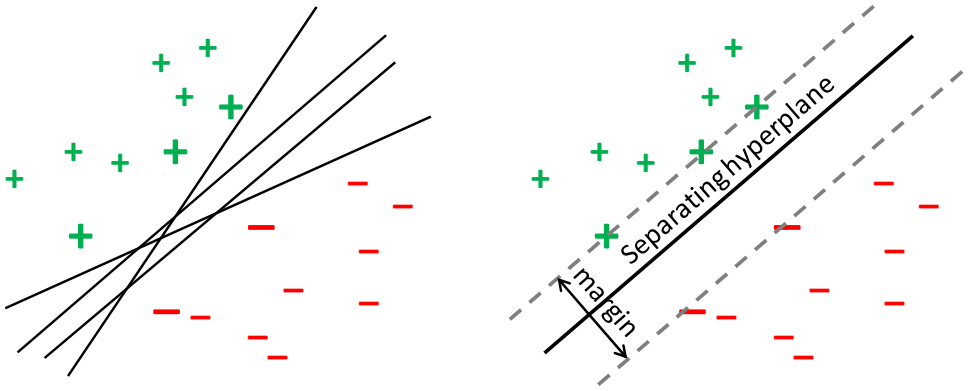


Fig. AIII.3. (left) Several possible separating hyperplane between our positive and negative samples. (right) SVM discovers the separating hyperplane by maximizing margin.

The points touching this separating hyperplane are called support vectors- only they determine the position of the hyperplane, and only they will be used during the classification of new test samples. The mathematical derivation of SVMs is complex and out of the scope of the section, however SVMs start with a regularized logistic regression. To circumvent non-differentiable objective function, non-limiting constraints are introduced such that the Lagrangian duality may be used to minimize our modified objective function. The SVM objective function reduces to:

$$\max_{0\le\alpha\le c} \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j k(x_i, x_j) \qquad (AIII.8)$$

Where $\boldsymbol{\alpha}$ represents the weights we are solving for, $\boldsymbol{y}$ is the ground truth classification for training points (defined as -1 or +1), $k()$ is a kernel function (dot product for now), and x is input attributes. The corresponding classification is done with:

$$f(x) = \sum_i \alpha_i y_i k(x, x_i) \qquad (AIII.9)$$

Where α_i represents the weight we solved for with x_i and y_i being the corresponding attributes and ground truth classification. $\boldsymbol{x}$ is the input test sample, and $k()$ is once again the kernel function. If (AIII.9) is >0, the class is positive, otherwise it is negative. The value of (AIII.9) can be used as confidence values as to which class x belongs to. There are several nice properties of SVMs. The first is that $\boldsymbol{\alpha}$ is often sparse because the objective function is trying to pull it negative, but the constraints are forcing $\boldsymbol{\alpha}$ to be positive. The net result is that many of the terms in (AIII.9) are reduced to 0, and therefore can be dropped, simplifying the classification tremendously. As such, (AIII.9) is only summed over the z support vectors.

Another nice property of SVMs is that the $k(x,x_i)$ function can be replaced with a wide variety of kernel functions. These kernels allow the user to define non-linear mappings of input attributes such that linear hyperplanes can still segment our data into two classes. The default function is the linear inner product of input sample x, with support vector training elements x_i. If we treat $k()$, as a black box function, and if we can somehow get the inner products, we won't need the transform the original x_i data or the test sample, $\boldsymbol{x}$. Essentially this means that we can wrap the transformation of $\boldsymbol{x}$ and x_i to a new non-linear warping as part of our dot product function- a huge computational savings. This kernel trick allows us to replace $k()$ with a family of functions including polynomial functions, radial basis functions, and Gaussian functions. Aside from the linear dot product, radial basis functions are very popular in the machine learning community.

Appendix IV- Software Libraries

Dimensionality reduction (LPP, LGE, LDA, PCA, NPE):

http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html

There are two variants of sparse representation software:

1a) LARS- Least Angle Regression w/ Lasso
SparseLab, version 2.1
http://sparselab.stanford.edu/
function used is: SolveLasso.m

1b) LARS w/ non-negativity matrix factorization
same as '1)', but use 'nnlasso' argument to SolveLasso.m function

2) group sparsity- attempts to group dictionary items based upon class
SLEP, version 4.0
http://www.public.asu.edu/~jye02/Software/SLEP/
function used is: SR_caller.m

K-SVD dictionary learning and optimization toolbox:

http://www.cs.technion.ac.il/~ronrubin/software.html

Label Consistent K-SVD code at:

http://www.umiacs.umd.edu/~zhuolin/projectlcksvd.html

LibSVM:

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Digitally attached with this thesis are two "hello world" examples in a file called LPP_example.zip which utilize dimensionality reduction and sparse representations within the Matlab programming environment. All input image data and ancillary m-files are included in the zip file. The two main examples are:

1) LPP_example.m- Contrasts the difference between PCA and LPP. This code can reproduce the two scatter plots in Figure 3.2. The LPP code is based upon Deng Cai's dimensionality reduction toolkits (see above). To demonstrate classification methodologies, LPP_example.m contains example code which sets up cross-validation experiments and the corresponding accuracy analysis associated with confusion matrices. For classification, the libSVM library (see above) is used with both linear and radial basis function kernels. To

pictorially show how other dimensionality reduction methods compare to PCA and LPP, there is sample code showing how to use LLE and Isomap to create similar scatter plots as in Figure 3.2.

2) LPP_example_w_SR.m- Contains code demonstrating the usage of dimensionality reduction (both PCA and LPP) along with sparse representations for accurate sparse representation classification. This code can reproduce the plots in Figures 6.1 and 6.2, including the automatic overlay of image data on top of normal Matlab plots. The LPP code is based upon Deng Cai's dimensionality reduction toolkits (see above). The sparse representation libraries are based on SLEP 4.0 (see above), and use the minimum reconstruction error from (5.6).

References

[1] S. Afzal, C. Morrison, and P. Robinson, "Intentional affect: an alternative notion of affective interaction with a machine," presented at the Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology, Cambridge, United Kingdom, 2009.

[2] M. Pantic, A. Pentland, A. Nijholt, and T. S. Huang, "Human computing and machine understanding of human behavior: a survey," in *Artifical Intelligence for Human Computing. ICMI 2006 and IJCAI 2007 International Workshops, 3 Nov. 2006*, Berlin, Germany, 2007, pp. 47-71.

[3] M. Lew, E. M. Bakker, N. Sebe, and T. S. Huang, "Human-computer intelligent interaction: A survey," in *4th IEEE International Workshop on Human - Computer Interaction, HCI 2007, October 20, 2007 - October 20, 2007*, Rio de Janeiro, Brazil, 2007, pp. 1-5.

[4] R. W. Picard, *Affective Computing*. Cambridge, MA: MIT Press, ISBN-13 978-0262661157, 1997.

[5] F. Kaplan, "Are gesture-based interfaces the future of human computer interaction?," presented at the Proceedings of the 2009 international conference on Multimodal interfaces, Cambridge, Massachusetts, USA, 2009.

[6] J. Shotton*, et al.*, "Real-Time Human Pose Recognition in Parts from Single Depth Images," in *Computer Vision & Pattern Recognition*, Colorado Springs, CO, 2011.

[7] E. Suma, B. Lange, A. Rizzo, D. Krum, and M. Bolas, "FAAST: The Flexible Action and Articulated Skeleton Toolkit," in *Virtual Reality*, 2011, pp. 247-248.

[8] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1?," *Vision Research,* vol. 37, pp. 3311-25, 1997.

[9] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature,* vol. 381, pp. 607-9, 1996.

[10] J. Sherrah, S. Gong, and E. J. Ong, "Face distributions in similarity space under varying head pose," *Image and Vision Computing,* vol. 19, pp. 807-819, 2001.

[11] P. Ekman and W. V. Friesen, "The Facial Action Coding System," *ed. San Francisco, CA: Consulting Psychologists Press, Inc.*, 1978.

[12] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 34-58, 2002.

[13] M. B. Lewis and H. D. Ellis, "How we detect a face: A survey of psychological evidence," *International Journal of Imaging Systems and Technology,* vol. 13, pp. 3-7, 2003.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition*, 2001, pp. I-511-I-518 vol.1.

[15] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings International Conference on Image Processing*, 2002, pp. 900-3.

[16] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence,* vol. 14, pp. 771-80, 1999.

[17] G. R. Bradski, *Learning OpenCV*, O'Reilly Media, 2008.

[18] W. Gao*, et al.*, "The CAS-PEAL large-scale chinese face database and baseline evaluations," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans,* vol. 38, pp. 149-161, 2008.

[19] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012, June 16, 2012 - June 21, 2012*, Providence, RI, United states, 2012, pp. 2879-2886.

[20] R. Brunelli and T. Poggio, "Face recognition: features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 15, pp. 1042-1052, 1993.

[21] M. Osadchy, Y. Le Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research,* vol. 8, pp. 1197-1215, 2007.

[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," in *Proc. IEEE International Conference on Computer Vision*, New York, NY, USA, 1987, pp. 259-268.

[23] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models - their training and application," *Computer Vision and Image Understanding,* vol. 61, pp. 38-59, 1995.

[24] M. R. Bolin and S. Chen, "An Automatic Facial Feature Finding System for Portrait Images," in *Final Program and Proceedings: IS and T's 55th Annual Conference, April 07,2002 - April 10,2002*, Portland, OR, United states, 2002, pp. 226-231.

[25] S. Milborrow and F. Nicolls, "Locating facial features with an extended active shape model," in *10th European Conference on Computer Vision, ECCV 2008, October 12, 2008 - October 18, 2008*, Marseille, France, 2008, pp. 504-513.

[26] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, pp. 681-5, 2001.

[27] J. Matthews and S. Baker, "Active appearance models revisited," *International Journal of Computer Vision,* vol. 60, pp. 135-164, 2004.

[28] D. Cristinacce and T. Cootes, "Automatic feature localisation with constrained local models," *Pattern Recognition,* vol. 41, pp. 3054-67, 2008.

[29] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of SIGGRAPH 99: 26th International Conference on Computer Graphics and Interactive Techniques, 8-13 Aug. 1999*, New York, NY, USA, 1999, pp. 187-94.

[30] A. M. Alattar and S. A. Rajala, "Facial features localization in front view head and shoulders images," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999, pp. 3557-60.

[31] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, pp. 607-626, 2009.

[32] N. Hu, W. Huang, and S. Ranganath, "Head pose estimation by non-linear embedding and mapping," in *IEEE International Conference on Image Processing 2005, ICIP 2005, September 11, 2005 - September 14, 2005*, Genova, Italy, 2005, pp. 342-345.

[33] B. Raytchev, I. Yoda, and K. Sakaue, "Head pose estimation by nonlinear manifold learning," in *Proceedings of the 17th International Conference on Pattern Recognition, 23-26 Aug. 2004*, Los Alamitos, CA, USA, 2004, pp. 462-6.

[34] S. Z. Li, L. XiaoGuang, H. Xinwen, P. Xianhua, and C. Qiansheng, "Learning multiview face subspaces and facial pose estimation using independent component analysis," *IEEE Transactions on Image Processing,* vol. 14, pp. 705-12, 2005.

[35] J. Myung-Ho and K. Hang-Bong, "Human robot interaction using face pose recognition," in *2007 Digest of Technical Papers. International Conference on Consumer Electronics, 10-14 Jan. 2007*, 2007, p. 2 pp.

[36] H. Zhang, L. Toth, W. Deng, J. Guo, and J. Yang, "Monitoring visual focus of attention via local discriminant projection," in *1st International ACM Conference on Multimedia Information Retrieval, MIR2008, Co-located with the 2008 ACM International Conference on Multimedia, MM'08, August 30, 2008 - August 31, 2008*, Vancouver, BC, Canada, 2008, pp. 18-23.

[37] A. Kanaujia, H. Yuchi, and D. Metaxas, "Tracking facial features using mixture of point distribution models," in *Computer Vision, Graphics and Image Processing. 5th Indian Conference, ICVGIP 2006. Proceedings, 13-16 Dec. 2006*, Berlin, Germany, 2006, pp. 492-503.

[38] R. Ptucha and A. Savakis, "Facial pose estimation using a symmetrical feature model," in *2009 IEEE International Conference on Multimedia and Expo, ICME 2009, June 28, 2009 - July 3, 2009*, New York, NY, United states, 2009, pp. 1664-1667.

[39] R. Ptucha and A. Savakis, "Pose estimation using facial feature points and manifold learning," in *2010 17th IEEE International Conference on Image Processing (ICIP 2010), 26-29 Sept. 2010*, 2010, pp. 3261-4.

[40] R. Stiefelhagen, J. Yang, and A. Waibel, "Model-based gaze tracking system," in *IEEE International Joint Symposia on Intelligence and Systems*, Los Alamitos, CA, USA, 1996, pp. 304-310.

[41] V. Kruger and G. Sommer, "Gabor wavelet networks for efficient head pose estimation," *Image and Vision Computing,* vol. 20, pp. 665-672, 2002.

[42] A. Savakis, M. Erhard, J. Schimmel, and J. Hnatow, "A multi-camera system for real-time pose estimation," in *Proceedings of SPIE, Intelligent Computing: Theory and Applications V*, Orlando, FL, USA, 2007.

[43] S. Ohayon and E. Rivlin, "Robust 3D head tracking using camera pose estimation," in *Proceedings - International Conference on Pattern Recognition*, 2006, pp. 1063-1066.

[44] X. Wang, H. Huang, Z. Ruan, and Z. Lu, "Fast face orientation estimation from an uncalibrated monocular camera," in *Proceedings - 1st International Congress on Image and Signal Processing*, 2008, pp. 186-190.

[45] P. Martins and J. Batista, "Monocular head pose estimation," in *IEEE International Conference on Image Processing*, San Diego, CA, USA, 2008, pp. 357-368.

[46] K. Kinoshita, Y. Ma, S. Lao, and M. Kawaade, "A fast and robust 3D head pose and gaze estimation system," in *8th International Conference on Multimodal Interfaces*, New York, NY, USA, 2006, pp. 137-138.

[47] M. Pantic and L. U. M. Rothkrantz, "Automatic analysis of facial expressions: The state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 1424-1445, 2000.

[48] Z. Zhihong, M. Pantic, G. I. Roisman, and T. S. Huang, "A survey of affect recognition methods: audio, visual, and spontaneous expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, pp. 39-58, 2009.

[49] L. Shuai-Shi, T. Yan-Tao, and L. Dong, "New research advances of facial expression recognition," in *Eighth International Conference on Machine Learning and Cybernetics*, 2009, pp. 1150-5.

[50] O. Rudovic, I. Patras, and M. Pantic, "Coupled Gaussian process regression for pose-invariant facial expression recognition," in *11th European Conference on Computer Vision, ECCV 2010, September 5, 2010 - September 11, 2010*, Heraklion, Crete, Greece, 2010, pp. 350-363.

[51] L. Shuai-Shi, T. Yan-Tao, and L. Dong, "New research advances of facial expression recognition," in *2009 Eighth International Conference on Machine Learning and Cybernetics (ICMLC), 12-15 July 2009*, 2009, pp. 1150-5.

[52] B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey," *Pattern Recognition,* vol. 36, pp. 259-275, 2003.

[53] V. Bettadapura. (2012, Face Expression Recognition and Analysis: The State of the Art. *CoRR abs/1203.6722*.

[54] C. Yeongjae and K. Daijin, "Natural facial expression recognition using differential-AAM and manifold learning," *Pattern Recognition,* vol. 42, pp. 1340-50, 2009.

[55] C. Martin, U. Werner, and H. M. Gross, "A real-time facial expression recognition system based on active appearance models using gray images and edge images," in *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition, 17-19 Sept. 2008*, 2008, p. 6 pp.

[56] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on Local Binary Patterns: A comprehensive study," *Image and Vision Computing,* vol. 27, pp. 803-816, 2009.

[57] R. Ptucha and A. Savakis, "Facial Expression Recognition Using Facial Features and Manifold Learning," in *International Symposium on Visual Computing (ISVC '10)*, Las Vegas, NV, 2010.

[58] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 13-18 June 2010*, Los Alamitos, CA, USA, 2010, p. 8 pp.

[59] Z. Zhengyou, M. Lyons, M. Schuster, and S. Akamatsu, "Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, 14-16 April 1998*, Los Alamitos, CA, USA, 1998, pp. 454-9.

[60] I. Buciu, C. kotropoulos, and I. Pitas, "ICA and Gabor representation for facial expression recognition," in *Proceedings of International Conference on Image Processing, 14-17 Sept. 2003*, 2003, pp. 855-8.

[61] X. Feng, M. Pietikainen, and A. Hadid, "Facial expression recognition based on local binary patterns," *Pattern Recognition and Image Analysis,* vol. 17, pp. 592-598, 2007.

[62] N. Sebe, M. S. Lew, Y. Sun, I. Cohen, T. Gevers, and T. S. Huang, "Authentic facial expression analysis," *Image and Vision Computing,* vol. 25, pp. 1856-63, 12/03 2007.

[63] R. Ptucha, G. Tsagkatakis, and A. Savakis, "Manifold Based Sparse Representation for Robust Expression Recognition without Neutral Subtraction," presented at the BeFIT 2011 Workshop, International Conference on Computer Vision, Barcelona, Spain, 2011.

[64] C. Curio, H. Bulthoff, and M. Giese, *Dynamic Faces: Insights from Experiments and Computation*, 1 ed.: The MIT Press, 2010.

[65] Z. Ambadar, J. W. Schooler, and J. F. Cohn, "Deciphering the Enigmatic Face," *Psychological Science,* vol. 16, pp. 403-410, 2005.

[66] S. Koelstra, M. Pantic, and I. Patras, "A dynamic texture-based approach to recognition of facial actions and their temporal models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, pp. 1940-54, 2010.

[67] J. C. Gower, "Generalized procrustes analysis," *Psychometrika,* vol. 40, pp. 33-51, 1975/03/01 1975.

[68] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, June 20, 2005 - June 25, 2005*, San Diego, CA, United states, 2005, pp. 886-893.

[69] B. Jiang, M. F. Valstar, and M. Pantic, "Action Unit Detection Using Sparse Appearance Descriptors in Space-Time Video Volumes," in *Face and Gesture Recognition*, Santa Barbara, CA, 2011.

[70] A. Ghodsi., "Dimensionality Reduction A Short Tutorial," University of Waterloo, Ontario,m CA, 2006.

[71] L. Cayton., "Algorithms for manifold learning," University of California, San Diego, Tech Rep. CS2008-0923, 2005.

[72] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science,* vol. 290, pp. 2319-23, 2000.

[73] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science,* vol. 290, pp. 2323-6, 2000.

[74] C. Deng, H. Xiaofei, H. Yuxiao, H. Jiawei, and T. Huang, "Learning a spatially smooth subspace for face recognition," in *CVPR '07. IEEE Conference on Computer Vision and Pattern Recognition, 18-23 June 2007*, 2007, pp. 650-6.

[75] X. He and P. Niyogi, "Locality Preserving Projections," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2003.

[76] X. He, D. Cai, S. Yan, and H.-J. Zhang, "Neighborhood preserving embedding," in *Proceedings - 10th IEEE International Conference on Computer Vision, ICCV 2005, October 17, 2005 - October 20, 2005*, Beijing, China, 2005, pp. 1208-1213.

[77] S. Zafeiriou and M. Petrou, "Sparse representations for facial expressions recognition via l1 optimization," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010, June 13, 2010 - June 18, 2010*, San Francisco, CA, United states, 2010, pp. 32-39.

[78] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and M. Yi, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, pp. 210-27, 2009.

[79] R. Ptucha and A. Savakis, "Manifold Based Sparse Representation for Facial Understanding in Natural Images," *Image and Vision Computing*, vol. 31, pp. 365-378, 2013.

[80] P. Lucey, J. Cohn, I. Matthews, K. Prkachin, and P. Solomon, "Painful Data: The UNBC-McMaster Shoulder Pain Expression Archive Database," in *Face and Gesture Recognition*, Santa Barbara, CA, 2011.

[81] N. Bhaskaran, I. Nwogu, M. Frank, and V. Govindaraju, "Lie to Me: Deceit Detection via Online Behavioral Learning," in *Face and Gesture Recognition*, Santa Barbara, CA, 2011.

[82] J. F. Cohn*, et al.*, "Detecting depression from facial actions and vocal prosody," in *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII 2009), 10-12 Sept. 2009*, 2009, p. 7 pp.

[83] J. Saragih, S. Lucey, and J. Cohn, "Real-time Avatar Animation from a Single Image," in *Face and Gesture Recognition*, Santa Barbara, CA, 2011.

[84] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, pp. 257-67, 2001.

[85] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," *IEEE Transactions on Medical Imaging,* vol. 18, pp. 712-721, 1999.

[86] L. Ce, J. Yuen, and A. Torralba, "SIFT flow: dense correspondence across scenes and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, pp. 978-94, 2011.

[87] Y. Songfan and B. Bhanu, "Facial expression recognition using emotion avatar image," in *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG 2011), 21-24 March 2011*, Los Alamitos, CA, USA, 2011, pp. 866-71.

[88] I. Kotsia and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines," *IEEE Transactions on Image Processing,* vol. 16, pp. 172-87, 2007.

[89] *Kinect*. Available: http://www.xbox.com/kinect, 2011.

[90] R. Ptucha and A. Savakis, "Facial Pose Tracking For Interactive Display," in *Western NY Image Processing Workshop*, Rochester, NY, 2009.

[91] R. Munozsalinas, R. Medinacarnicer, F. Madridcuevas, and A. Carmonapoyato, "Depth silhouettes for gesture recognition," *Pattern Recognition Letters,* vol. 29, pp. 319-329, 2008.

[92] K. Nickel and R. Stiefelhagen, "Pointing Gesture Recognition based on 3D-Tracking of Face , Hands and Head Orientation Categories and Subject Descriptors," pp. 140-146.

[93] C. Bellmore, R. Ptucha, and A. Savakis, "Interactive display using depth and RGB sensors for face and gesture control," Rochester, NY, pp. 1-4.

[94] B. Yoo*, et al.*, "3D user interface combining gaze and hand gestures for large-scale display," *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10,* pp. 3709-3709, 2010.

[95] D. Vogel and R. Balakrishnan, "Distant freehand pointing and clicking on very large, high resolution displays," *Proceedings of the 18th annual ACM symposium on User interface software and technology - UIST '05,* pp. 33-33, 2005.

[96] D. Stødle, T. M. S. T.-m. S. Hagen, J. M. Bjørndalen, and O. J. Anshus, "Gesture-Based , Touch-Free Multi-User Gaming on Wall-Sized , High-Resolution Tiled Displays," *Journal of Virtual Reality and Broadcasting,* vol. 5, pp. 0009-6, 2008.

[97] J. Heydekorn, M. Frisch, and R. Dachselt, "Evaluating a User-Elicited Gesture Set for Interactive Displays," *Mensch & Computer 2011: …,* 2011.

[98] H. Ghasemzadeh, V. Loseu, and R. Jafari, "Collaborative Signal Processing for Action Recognition in Body Sensor Networks: A Distributed Classification Algorihtm Using Motion Transcripts," in *9th ACM/IEEE Int. Conf. Inf. Process.*, 2010.

[99] K. Raja, I. Laptev, P. Perez, and L. Oisel, "Joint Pose Estimation and Action Recognition in Image Graphs," in *IEEE International Conference on Image Processing*, 2011.

[100] D. Weinland, E. Boyer, and R. Ronfard, "Action Recognition from Arbitrary Views Using 3D Exemplars," in *International Conference on Computer Vision*, 2007.

[101] s. Maji, L. Bourdev, and J. Malik, "Action Recognition from a Distributed Representation of Pose and Appearance," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[102] Y. Wang and Z. Zhang, "View Invariant Action Recognition in Surveillance Videos," in *Asian Conference on Pattern Recognition*, 2011.

[103] H. Imtiaz, U. Mahbub, and M. A. R. Ahad, "Action Recognition Algorithm Based on Optical Flow and RANSAC in Frequency Domains," in *SICE Annual Conference*, 2011.

[104] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," Journal of Cognitive Science, 1991, pp. 586-591.

[105] X. He and P. Niyogi, "Locality Preserving Projections," in *Advances in Neural Information Processing Systems 16*, Vancouver, Canada, 2003.

[106] X. Rui, Z. Qijun, D. Zhang, and S. Pengfei, "Facial expression recognition on multiple manifolds," *Pattern Recognition,* vol. 44, pp. 107-16, 2011.

[107] C. Deng, H. Xiaofei, H. Jiawei, and Z. Hong-Jiang, "Orthogonal Laplacianfaces for face recognition," *IEEE Transactions on Image Processing,* vol. 15, pp. 3608-14, 2006.

[108] S. Shaoyuan, Z. Haitao, and Y. Huijun, "Discriminant Uncorrelated Locality Preserving Projection," in *2010 3rd International Congress on Image and Signal Processing (CISP 2010), 16-18 Oct. 2010*, 2010, pp. 1849-52.

[109] Z. Haitao and S. Shaoyuan, "Optimal Locality Preserving Projection," in *2010 17th IEEE International Conference on Image Processing (ICIP 2010), 26-29 Sept. 2010*, 2010, pp. 1861-4.

[110] C. Xian-Fa, W. Gui-Hua, W. Jia, and L. Jie, "Enhanced supervised locality preserving projections for face recognition," in *2011 International Conference on Machine Learning and Cybernetics (ICMLC 2011), 10-13 July 2011*, 2011, pp. 1762-6.

[111] T. Kanade, J. F. Cohn, and T. Yingli, "Comprehensive database for facial expression analysis," in *Proceedings of the Fourth International Conference on Automatic Face and Gesture Recognition, 28-30 March 2000*, Los Alamitos, CA, USA, 2000, pp. 46-53.

[112] P. He, G. Pan, Y. Zhou, and H. Zhao, "Facial expression recognition using multi-class minimax probability machine," in *2008 7th World Congress on Intelligent Control and Automation, 25-27 June 2008*, 2008, pp. 5933-6.

[113] Y. Chang, C. Hu, R. Feris, and M. Turk, "Manifold based analysis of facial expression," *Image and Vision Computing,* vol. 24, pp. 605-614, 2006.

[114] C. Shan, S. Gong, and P. W. McOwan, "Appearance manifold of facial expression," in *Computer Vision in Human-Computer Interaction. ICCV 2005 Workshop on HCI. Proceedings, 21 Oct. 2005*, Berlin, Germany, 2005, pp. 221-30.

[115] Z. Ying, J. Li, and Y. Zhang, "Facial expression recognition based on classifier combinations," in *8th International Conference on Signal Processing, ICSP 2006, November 16, 2006 - November 20, 2006*, Guilin, China, 2007, and Chinese Institute of Electronics; IEE; URSI; IEEE Beijing Section; National Natural Science Foundation of China.

[116] Y. Hu, Z. Zeng, L. Yin, X. Wei, X. Zhou, and T. S. Huang, "Multi-view facial expression recognition," in *2008 8th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2008, September 17, 2008 - September 19, 2008*, Amsterdam, Netherlands, 2008.

[117] W. Te-Hsun and J. J. J. Lien, "Facial expression recognition system based on rigid and non-rigid motion separation and 3D pose estimation," *Pattern Recognition,* vol. 42, pp. 962-77, 2009.

[118] Z. Zhu and Q. Ji, "Robust real-time face pose and facial expression recovery," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2006, June 17, 2006 - June 22, 2006*, New York, NY, United states, 2006, pp. 681-688.

[119] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory,* vol. 52, pp. 6-18, 2006.

[120] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory,* vol. 52, pp. 489-509, 2006.

[121] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient Sparse Coding Algorithms," presented at the Advances in Neural Information Processing Systems, 2006.

[122] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least Angle Regression," *Ann. Statist.,* vol. 32, pp. 407--499, 2004.

[123] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, June 20, 2011 - June 25, 2011*, Colorado Springs, CO, United states, 2011, pp. 1697-1704.

[124] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher Discrimination Dictionary Learning for Sparse Representation," in *International Conference on Computer Vision*, Barcelona, Spain, 2011.

[125] M. Julien, B. Francis, P. Jean, and S. Guillermo, "Online Learning for Matrix Factorization and Sparse Coding," *J. Mach. Learn. Res.,* vol. 11, pp. 19-60, 2010.

[126] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers, 1-3 Nov. 1993*, Los Alamitos, CA, USA, 1993, pp. 40-4.

[127] S. F. Cotter, R. Adler, R. D. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection," *IEE Proceedings-Vision, Image and Signal Processing,* vol. 146, pp. 235-44, 1999.

[128] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE Transactions on Information Theory,* vol. 38, pp. 587-607, 1992.

[129] A. Yang, J. Wright, Y. Ma, and S. Sastry, "Feature Selection in Face Recognition: A Sparse Representation Perspective," University of California at Berkely, 2007.

[130] K. Engan, S. O. Aase, and J. H. Husoy, "Frame based signal compression using method of Optimal Directions (MOD)," *Proceedings - IEEE International Symposium on Circuits and Systems,* vol. 4, pp. IV-1-IV-4, 1999.

[131] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing,* vol. 54, pp. 4311-22, 2006.

[132] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient Implementationof the K-SVD Algorithm using Batch Orthogonal Matching Pursuit," Technion, Computer Science Dept., Haifa, Israel2008.

[133] G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "Sparse representations of image gradient orientations for visual recognition and tracking," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2011, June 20, 2011 - June 25, 2011*, Colorado Springs, CO, United states, 2011.

[134] R. Ptucha and A. Savakis, "Joint Optimization of Manifold Learning and Sparse Representations," in *Automatic Face and Gesture Recognition*, Shanghai, China, 2013.

[135] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009, June 20, 2009 - June 25, 2009*, Miami, FL, United states, 2009, pp. 2790-2797.

[136] N. Kumar, P. Belhumeur, and S. Nayar, "FaceTracer: a search engine for large collections of images with faces," in *Computer Vision. 10th European Conference on Computer Vision, ECCV 2008, 12-18 Oct. 2008*, Berlin, Germany, 2008, pp. 340-53.

[137] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, June 20, 2005 - June 25, 2005*, San Diego, CA, United states, 2005, pp. 994-1000.

[138] T. Banziger and K. R. Scherer, "Introducing the Geneva Multimodal Emotion Portrayal (GEMEP) Corpus," in *Blueprint for Affective Computing: A Sourcebook*, ed Oxford: Oxford University Press, 2010, pp. 271-294.

[139] M. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. R. Scherer, "The First Facial Expression Recognition and Analysis Challenge," in *Face and Gesture Recognition*, Santa Barbara, CA, 2011.

[140] G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments," University of Massachusetts, Amherst2007.

[141] Z. Ruicong and R. Qiuqi, "Discriminant sparse nonnegative matrix factorization," in *2009 IEEE International Conference on Multimedia and Expo (ICME), 28 June-3 July 2009*, 2009, pp. 570-3.

[142] S. Chew, P. Lucey, S. Lucey, J. Saragih, J. Cohn, and S. Sridharan, "Person-Independent Facial Expression Detection Using Constrained Local Models," in *Automatic Face and Gesture Recognition*, Santa Barbara, CA, USA, 2011.

[143] S. Jain, C. Hu, and J. K. Aggarwal, "Facial expression recognition with temporal modeling of shapes," in *2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2011, November 6, 2011 - November 13, 2011*, Barcelona, Spain, 2011, pp. 1642-1649.

[144] T. Pfister, L. Xiaobai, Z. Guoying, and M. Pietikainen, "Differentiating spontaneous from posed facial expressions within a generic facial expression recognition framework," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 6-13 Nov. 2011*, 2011, pp. 868-75.

[145] I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hammer, and H. J. Escalente, "ChaLearn Gesture Challenge: Design and First Results," in *Computer Vision and Pattern Recognition Workshops*, 2012.

[146] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *IEEE conference on Computer Vision and Pattern Recognition*, 2010.

[147] S. Azary and A. Savakis, "3D Action Classification Using Sparse Spatio-Temporal Feature Representations," in *International Symposium on Visual Computing (ISVC'12)*, 2012.

[148] L. Qiao, S. Chen, and X. Tan, "Sparsity preserving projections with applications to face recognition," *Pattern Recognition,* vol. 43, pp. 331-341, 2010.

[149] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.

[150] F. Zang and J. Zhang, "Discriminative learning by sparse representation for classification," *Neurocomputing,* vol. 74, pp. 2176-2183, 2011.

[151] M. Zheng*, et al.*, "Graph regularized sparse coding for image representation," *IEEE Transactions on Image Processing,* vol. 20, pp. 1327-1336, 2011.

[152] Z. Qiang and L. Baoxin, "Discriminative K-SVD for Dictionary Learning in Face Recognition," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 13-18 June 2010*, 2010, pp. 2691-8.

[153] W. Jinjun, Y. Jianchao, Y. Kai, L. Fengjun, T. Huang, and G. Yihong, "Locality-constrained linear coding for image classification," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 13-18 June 2010*, 2010, pp. 3360-7.

[154] L. Zhihui, J. Zhong, and Y. Jian, "Global sparse representation projections for feature extraction and classification," in *2009 Chinese Conference on Pattern Recognition. (CCPR 2009) and the First CJK Joint Workshop on Pattern Recognition (CJKPR), 4-6 Nov. 2009*, 2009, p. 5 pp.

[155] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, pp. 643-60, 06/ 2001.

[156] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas, "The i3DPost multi-view and 3D human action/interaction," in *CVMP*, 2009.

[157] R. Ptucha and A. Savakis, "Fusion of Static and Temporal Predictors for Unconstrained Facial Expression Recognition," in *International Conference on Image Processing*, Orlando, FL, 2012.