Improving Facial Expression Analysis via Intrinsic Normalization of Surfaces

Eric Robert Yudin
Improving Facial Expression Analysis via Intrinsic Normalization of Surfaces

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Eric Robert Yudin

Submitted to the Senate of the Technion — Israel Institute of Technology
Adar 5775 Haifa February 2015
This research was carried out under the supervision of Professor Ron Kimmel, in the Faculty of Computer Science.

Acknowledgements

First and foremost, I would like to thank my advisor, Ron Kimmel, for his wisdom and guidance throughout my research. Thanks also go out to Matan Sela, Aaron Wetzler, Nastya Dubrovina and the rest of the GIP lab for their technical, theoretical and moral support, and to Omri Azencot for help with Hebrew translation.

Thanks to my local friends and family, especially Rouven, Gideon, Jimy, Eitan, Nir, the Ritz’s, the Kahtans, the Meirovich’s and many others, for helping make Israel my home away from home.

And of course, my unending love and gratitude go to Mom, Dad, Alanna, Joelle, Brian and all of my friends back home. I could not have done it without their love, support, encouragement and patience.

The Technion’s funding of this research is hereby acknowledged.
# Contents

## List of Figures

## Abstract

### 1 Introduction

1.1 FACS and AU Detection ............................................. 3
1.2 Classification and Normalization .................................. 6
1.3 Previous Work ....................................................... 11

### 2 Preliminaries

2.1 Differential Geometry ............................................. 17
2.2 Manifold .............................................................. 17
2.3 Riemannian Manifold & Metric ..................................... 19
2.4 Gaussian Curvature .................................................. 19
2.5 Metric Space .......................................................... 20
2.6 Embedding Spaces and Isometries .................................. 21
2.7 Intrinsic and Extrinsic Geometry ................................... 22

### 3 Method: Intrinsic Normalization

3.1 Introduction ......................................................... 23
3.2 Isometric Model & Formal Constraints ............................ 24
3.3 Implementation ..................................................... 26
3.3.1 Deformation Transfer .............................................. 27
3.3.2 Non-Rigid Registration ........................................... 28
3.3.3 Laplace-Beltrami Smoothing ..................................... 28
3.3.4 Boundary Constraints ............................................ 29
3.3.5 Template Face ..................................................... 30
3.3.6 Intrinsic Normalization Algorithm ............................ 30

### 4 Supporting Algorithms

4.1 Introduction ......................................................... 33
4.2 Deformation Transfer ................................................ 33
4.2.1 Vertex-based Approach .......................................... 34
4.2.2 Poisson-based Approach ........................................... 35
4.3 Non-rigid Registration .................................................. 38

5 Experimental Results .................................................... 41
  5.1 Introduction .......................................................... 41
  5.2 Goal & Strategy ...................................................... 41
  5.3 Algorithm Implementation ........................................... 41
  5.4 Cross Validation ...................................................... 43
  5.5 Results Comparison .................................................. 43

6 Conclusions and Discussion ............................................ 53
  6.1 Conclusion .......................................................... 53
  6.2 Limitations .......................................................... 53
    6.2.1 Topological Differences ....................................... 53
    6.2.2 Processing Time ................................................. 54
    6.2.3 Manual Constraints ............................................. 54
  6.3 Future Directions .................................................... 54
    6.3.1 Topological Adjustment ........................................ 54
    6.3.2 Ideal Target Manifold ......................................... 55
    6.3.3 Generalization .................................................. 55
    6.3.4 Localization .................................................... 55

Hebrew Abstract .......................................................... i
List of Figures

1.1 FACS intensity scale. [EF78] ................................................. 4
1.2 Examples of multiple subjects performing identical AU combinations [SAD+08]. ....................................................... 5
1.3 Classification (a) training and (b) testing processes. .................. 6
1.4 Demonstration of alignment variability across several faces making a happy expression. (a)-(c) illustrates the sampled scanline. (d) shows depth values for each face. ........................................... 8
1.5 A Receiver Operator Characteristic curve. ............................. 10
1.6 The basic LBP operator. .................................................. 13
1.7 (a) Construction of the orientation-based CS-3DLBP operator. (b) First four Ratio-Based Geometry features. ......................... 14

2.1 String B (a) is a one-dimensional manifold. String C (b) is not, as it fuses with itself. ............................................. 18

3.1 The geodesic follows the in-surface shortest path. ....................... 25
3.2 Illustration of a discrete mesh with six vertices and five faces. ....... 27
3.3 Example Laplace-Beltrami smoothing step of a single simplex. ....... 29
3.4 A rendering of the template mesh. ...................................... 30

4.1 Left to right: Hat basis functions $\phi_i$, $\phi_j$ and $\phi_k$ for the same triangle. The height represents the value of the function. For each index $q$ of the mesh there is a piecewise linear function $\phi_q$ that is of unity value at vertex $v_q$, that is of zero value at all other vertices, and that linearly interpolates in between. .................................................. 36
4.2 Illustration of $\alpha_{i,j}$ and $\beta_{i,j}$ as specified by the cotangent weight scheme. 37

5.1 Examples of non-normalized faces (left) and their corresponding normalized faces (right). ........................................... 45
5.2 More examples of non-normalized faces (left) and their corresponding normalized faces (right). .................................. 46
5.3 Graph comparing AuC for normalized and non-normalized input data for first half of AU set. ............................................. 47
5.4 Graph comparing AuC for normalized and non-normalized input data for second half of AU set. ................................. 48
5.5 Graph comparing AuC error for normalized and non-normalized input data for selected AUs. ................................. 50
5.6 Graph comparing AuC error for normalized and non-normalized input data for the rest of the AU set. ................................. 51
Abstract

When comparing two or more sets of data it is prudent to first normalize them. This process typically involves subtracting the mean from each example and dividing by its standard deviation or variance. Such a practice brings all data examples to similar scale and range, and can also serve to reduce dimensionality. Further, if all data examples truly represent a single class, normalization can reduce their variability, without sacrificing variance between classes.

Here we employ a geometric framework to extend the concept of data normalization to the domain of functions that lie on manifolds. We pose normalization in this context as an embedding of all examples into manifolds nearly isometric to one another. Using novel geometric tools, we propose an implementation for the case of discretized functions on triangulated two-dimensional manifolds.

We apply the proposed Intrinsic Normalization technique to the task of automatic facial action unit detection. This problem has received much attention in the literature ever since Ekman and Friesen introduced the Facial Action Coding System (FACS) in an attempt to codify facial expressions via modular components called action units. By normalizing cross-subject examples to a common template face, we are able to improve the results of state-of-the-art action unit detection algorithms.
Chapter 1

Introduction

1.1 FACS and AU Detection

Facial expressions constitute a significant component of human communication. They contain non-verbal elements that enhance and provide deeper context to the words themselves. They can convey meaning in the absence of words altogether. They even potentially provide more insight into an individual’s thoughts than he or she may be consciously aware. Indeed, utilizing and deciphering facial expressions are an integral part of the human experience. Doing so is therefore so crucial on a societal level that those who lack this ability have difficulty communicating properly. This is probably one reason why newer, digital forms of communication such as chat, SMS and email, and even voice-only conversation feel constrained and “missing something”.

The importance of facial expressions led scientists Paul Ekman and Wallace Friesen to study them in detail [Ekm70]. They identified six universal expressions – universal in that they do not vary across cultures. These are happiness, sadness, surprise, fear, disgust and anger. This cultural invariance has significant implications, specifically, that there may be a genetic basis for facial expression.

Aside from these six basic expressions, however, Ekman & Friesen found that no other correspondence between facial configuration and communicative intent remained valid universally. This meant that such correspondences could be culturally dependent, or even individually dependent.

FACS

In their studies, Ekman and Friesen recognized the need to codify all possible facial configurations. The language of facial expression could not be reduced to any fixed set; rather, it required its own alphabet whose letters, together as words, could describe any facial configuration the human face is capable of making. The Facial Action Coding System (FACS) [EF78] came from this insight. FACS considers the face as groups of muscles, and specifically and modularly describes each local movement as one of 46 possible action units, or AUs. For example, Action Unit 2, usually written AU2,
indicates the raising of the outer eyebrow; AU15 indicates depression of the lip corner; and AU6 indicates raising of the cheek and compression of the eyelid. Together with a range of five possible intensities, AUs form the building blocks for any configuration expressible by the human face.

In FACS, intensity itself is scored on a scale from A, which indicates trace presence, to E, which indicates maximum presence. The scale is non-linear, meaning that there is more dynamic range allotted to some letters than to others. Figure 1.1 is a graphical representation of the intensity scale. However, this image is more of a conceptual indicator of intensity; in many cases intensities are simply not well-defined geometrically. The manual often has a very heuristic means of scoring certain intensities of AUs, often described in terms of whether specific features are apparent or not.

![FACS intensity scale](image.png)

Figure 1.1: FACS intensity scale. [EF78]

Action units are defined in terms of how much effort a certain muscle group is making with respect to no facial action at all. That is, FACS defines their presence and intensity with respect to a given subject’s neutral face. This implies that FACS is defined independently of physiognomy, the inherent structure and characteristics of an individual’s face. As an example, the same AUs and intensities will comprise a slight smile on both a long-faced woman and a heavily-jowled man. Figure 1.2 shows several examples of multiple subjects performing identical action unit combinations. Note how the differences in physiognomy could affect geometric properties for what FACS considers identical.

Defining AUs with respect to neutral face also implies that there is no guarantee of linearity. Put differently, we cannot simply model the effect of an AU as the addition of, say, a field of deformation vectors. Were this the case, the required field would need to vary with physiognomy, defeating the purpose of a universally-applicable facial expression taxonomy.

Mathematically speaking, action units form a sparse basis that spans the space of facial expressions, in that with FACS, one can describe any facial expression as a weighted combination of AUs. As indicated above, this basis is not necessarily linear. Compared with a naïve representation like the vector-based one above, FACS significantly lowers the dimensionality. The difficulty, of course, is correctly classifying the action units.

**Automatic Facial Action Unit Detection**

Though there has been much research in automating the coding process, and even companies based around automatic AU detection technology, current algorithms are
still not reliable enough to leave the human coder fully uninvolved. Hence to this day, there is still a demand for manual FACS coding. The process is time consuming and very prone to human error. In addition, becoming a FACS coder requires learning from an extensive specification and getting certified. Automating the process would therefore hugely reduce the human burden. Having machines perform this task tirelessly and more quickly means whole video streams of people could be coded with little human effort. The data produced by such a system could open doors for a variety of applications.

The field of human-computer interaction (HCI) relies on several different kinds of data generated by humans. Examples are voice, brainwaves, body movement, hand gesture, heartbeat, and muscular activity. The ability to detect facial expression adds yet another dimension of information for an HCI system to integrate and interpret. This higher dimensional view can lead to better user experience through more natural interfaces. A more advanced system that can detect expression information as well as vocal and body pose data may be equipped to read social cues, which could give a more human-like understanding of humans. Interaction with something more human-like could ease the feeling of interacting with an automated system.

In addition, the extra dimension of data could benefit the medical community. There may be correlations with the presence of mental disorders and the execution of certain facial expressions or movements (or lack thereof) [MPP98]. Current diagnoses are an
extremely manual process, requiring long interviews and depending on expert knowledge of the disease. While it may be some time before such diagnosis can be fully automated, expert systems in the near future could help by raising flags.

1.2 Classification and Normalization

Currently, the most popular way to approach the problem of action unit detection is to use supervised machine learning. Machine learning in its essence is a set of computational tools that can be used to make accurate predictions based on previous observations of the world [MRT12]. In supervised machine learning, inference process is based on a prior training session in which data examples $x^{(i)}$ are associated with values $y^{(i)}$ known to be ground truth. After training, the system “goes online” and attempts to predict values for previously unseen examples. Figure 1.3 illustrates these phases.

**Feature extraction and Classification**

One type of problem to which this can be applied is classification, in which the goal is to associate input examples with a specific category, or class of phenomena. The functions that describe the association, together with the algorithms that create them, are called classifiers. Many different classifiers exist, and each comes with its advantages and disadvantages. Examples include Support Vector Machines, Neural Networks, Decision Trees, and Random Forests.
The inputs to classifiers are called features. Features are numerical descriptions of a data example, usually in the form a vector of values. Thus a classifier can be described as a function \( f : x^{(i)} \in \mathbb{R}^N \rightarrow y^{(i)} \in \mathbb{I} \), mapping these vectors to some indexable category. Feature vectors should be as informative yet succinct as possible, such that they can use less information to describe the example well enough to properly classify.

**Feature Scaling and Normalization**

When performing machine learning tasks or most kinds of data analysis it is prudent first to normalize the data. Indeed, most of the methods above begin by applying some transformation to get all examples into some canonical form. The intent when normalizing is to remove specific kinds of variability from each example. This variability is known to be due to the way in which the data is measured or represented, and not part of the distribution of possible configurations or structure of the class of interest.

2D digit recognition can serve as a concrete motivating example. Suppose we are attempting to recognize the digit ‘8’. Examples of this digit will vary in many different ways: shape, smoothness, connectivity, and more. With ever more examples our classifier builds a better picture of exactly how this class can vary. However, there is some variability that does not interest us. If two examples have been drawn identically but appear shifted within the image, we would still like to consider them the same version of the digit. We therefore center all input examples, as the shifting should not contribute to the classifier’s knowledge of what an ‘8’ looks like. In fact, considering the shift unnecessarily increases dimensionality of the space, requiring much more data for accurate classification. The same notion applies for rotation and scale, though only to an extent; over-normalization can cause false alarms by allowing the classifier to recognize members of different classes. For example, if we allowed rotational normalization of 90°, our classifier would now accept the infinity symbol ‘∞’ as part of our class. Good normalization attempts to reduce unnecessary variation as much as possible, but only as far as it can maintain the boundaries of the class of interest.

For the application of facial expression recognition, it is prudent to center, align and scale along the axes. This removes dimensionality that does little to inform a classifier of a classifier’s inherent structure, and does not act to create false positives in the process. However, we can do better. As we are trying to isolate all manifestations of an expression independent of subject, we would like to normalize facial geometry in such a way as to remove all variability related to an individual’s physiognomy.

**Subject Independence**

In facial expression analysis we typically have many different examples from many different subjects, such that examples of a certain phenomenon span many different subjects. In these cases, variability due to physiognomy is undesirable. Because of this, we prefer to make the data subject independent. The standard 9 degree of freedom
Figure 1.4: Demonstration of alignment variability across several faces making a happy expression. (a)-(c) illustrates the sampled scanline. (d) shows depth values for each face.

(DOF) normalization mentioned above does take care of some aspects of physiognomy, such as facial width, height and to some extent depth. However, most of the finer aspects of physiognomy are typically allowed to persist in the data, and this can affect the learning procedure. There are at least two ways in which this variability can adversely affect the learning process.

**Alignment Variability** First, measurements that in some way rely on ambient coordinates, or that are not “surface aware” can cause what we will call **alignment variability**. Even if the boundaries of the face are aligned during the normalization process, individual facial features may not line up with one another. Any learning features that rely on ambient coordinates to make their measurements may suffer from jitter due to these misalignments. As an example, if we were to align several faces via the nine DOF mentioned above, choose a point on, say, the tip of the cheek, and then use this ambient coordinate to find the same point on the other faces, there will likely be jitter across all faces with respect to this point, as this representation has no notion of corresponding in-surface features other than their ambient coordinates. Figure 1.4 illustrates this case. A more “surface-aware” representation such as a parameterization based on facial features would help reduce this jitter.

**Measurement Variability** Measurements based on characteristics of the surface itself produce a second kind of variability we will call **measurement variability**. Even assuming that all surfaces are in correspondence with one another, that is, that we can always choose the same semantic point on all faces, different physiognomies induce different surface characteristics. This implies that even given the same expression, neutral or otherwise, the values of features (e.g. curvature, etc.) will be different. Consequently, the variabilities introduced by each unique physiognomy becomes coupled in with the variability brought by the structure of the expressions themselves.
Ideal Subject-Independent Setup

We can now begin to ask what the best setup for analyzing facial expressions would look like. Ideally, all possible variations of a given facial expression would be performed by a single subject. Note that it would not merely include every way that a particular person performs the facial expression. Rather, it would consider how all people perform the expression, but would replace all of their physiognomies with that of a single subject. Doing so would guarantee subject independence by definition. The contribution in this work is an algorithm to produce such a setup.

Area Under ROC Curve (AuC)

We require an effective way of measuring success of a given algorithm. With our ground truth (in our case supplied by the Bosphorus database), we know a priori what is and is not correct. However, in binary classification problems there are many different ways to judge success. For any given example classification, there are four different kinds of results. True positives (resp. false positives) are correct (resp. incorrect) guesses by the classifier that the example belongs to the class of interest, while true negatives (resp. false negatives) are correct (resp. incorrect) guesses that the example does not belong. With this multidimensional way of describing results, there are multiple ways of describing success versus failure of a classification algorithm. The following are a few:

1. Recognition Percent measures the number of correct positive and negative classifications.

2. True Positive Rate, (also known as Hit Rate, recall) measures the number of true positives against total actual positives.

3. False Positive Rate, (also known as False Alarm Rate, fall-out) measures the number of false positives against actual negatives.

Each of these measures helps illustrate a piece of the classification picture, but none encompasses all aspects. Recognition Percent gives an idea of the overall effectiveness of the algorithm. However, an example shows what we miss. Say we have a set of 100 examples. The ground truth indicates that instances 1-90 are in the negative class, while instances 91-100 are in the positive class. Meanwhile, our detector is not very sensitive, so it associates examples 1-90 with the negative class, as well as 5 of the last examples 91-100. Recognition Percent will be 95%, a seemingly impressive measure. However, an equally (if not more) telling number is the Hit Rate, which will be a dismal 50%, since the detector only recognized 5 of the 10 true positives.

To analyze the system graphically it is useful to draw the Receiver Operator Characteristic (ROC). The ROC curve plots True Positive Rate against False Positive Rate while varying the threshold of the classifier. Figure 1.5 shows an example.
Figure 1.5: A Receiver Operator Characteristic curve.

We build this curve by changing the threshold, counting how many instances the classifier passes. When the threshold for classification is very high, very few examples will be declared belonging to the class, and we will get points close to the lower left corner. Likewise, a very low threshold will pass many or all examples, resulting in points close to the top right corner. It is important to note that increasing the threshold only allows more positives to pass, and so for a given change in threshold, the curve can only extend right, up or diagonal. Consequently, the curve is monotonic.

By looking at the ROC curve, one can gain an intuitive feel for how a classifier behaves from all perspectives. For example, the ideal classifier produces a curve with three points: (0,0) when the threshold is at its highest, (1,1) when the threshold is at its lowest, and (0,1) everywhere in between. This illustrates a classifier that outside of the boundary threshold conditions, always guesses perfectly.

We can go a step further and use the area under the ROC curve (AuC) as a single convenient number to express performance of binary classification. In the best possible situation like the one above, the area will be 1, while in the worst, the area will be 0.5, since the worst case ROC curve is a straight line connecting (0,0) to (1,1). This number therefore characterizes how close to perfect a classifier behaves, including both notions of true and false positives in the process.

**Bosphorus Database**

In order to evaluate algorithms and to properly compare methods, we require a common dataset. That is, if we are to build a new algorithm or improve an existing one, best practices compel us to compare the performance using the same input data. In this vein, domain-specific databases are often created to promote algorithm development related to a given field.
One of the most popular facial expression databases is currently the Bosphorus Database \cite{SAD08}. This database consists of 105 subjects, with each subject providing multiple facial expressions. Each example provides multiple modalities of data: a high-resolution depth scan; a high resolution RGB image; about 22 landmarks; and most importantly, FACS codes describing the action units and their intensities, coded by a certified FACS coder.

Each example is labelled in the form <subject>._<expression> although <expression> does not necessarily correspond exactly to which AUs are present. It is instead more of a request to the subject; the FACS codes, though, are intended to reflect exactly what resulted from this request. For instance, the example labeled bs000_LFAU_9_0 indicates "Lower Face Action Unit 9" for subject 000, although it was coded as 7D+9D+24D.

The Bosphorus database has enjoyed popularity recently because of its wealth of data, quality and consistency of acquisition and its possession of FACS labels for each face. All of these aspects make it an invaluable tool for modern facial expression research.

1.3 Previous Work

Action Unit Detection

For decades, research in action unit detection was restricted primarily to analysis of two-dimensional RGB images. Works such as \cite{KP07}, and others mentioned in \cite{Bet12} represent the pinnacle of 2D action unit detection and show that this analysis met with much success. Indeed, there are even commercial enterprises that rely on 2D-based analysis \cite{Kha15}.

The advent of high-resolution, efficient, cheap 3D scanners, graphics processing hardware (GPUs) and accompanying high-level programming languages in the early-to mid-2000s brought interest and practicality to 3D data processing. While this kind of data is typically somewhat more time-consuming and complicated to process, and while 3D scanners do introduce their own errors and sensor noise, the advantages are significant. First, illumination dependence is a problem that plagues traditional 2D color processing, and has generated much research on its own. Many 3D depth scanning techniques, meanwhile, work independently of visible light, and as such are illumination independent. Processing can thus occur in any kind of lighting conditions, making the data acquisition process much more flexible. Secondly, 2D-based analysis describes facial movements as deformations. Some even go beyond describing these deformations in terms of the image plane, and attempt to reconstruct 3D properties based on the color values and perhaps some prior facial model. While these have met with some success, deformations are better described with the language of surfaces and geometry existing in three-dimensional ambient space. Further, true 3D data allows a facial analysis system to calculate more accurate values for properties such as depth and curvature, instead of
relying on rough approximations based on 2D RGB images. Indeed, [SST12] shows that employing 3D data only enhances results over pure 2D-based methods.

The first set of works in 3D facial expression analysis concerned expressions themselves. That is, detectors were trained to recognize overall expressions like the six universal expressions, as opposed to action units. This provided a base of research, but was still limited in scope. Action unit recognition using 3D sensors therefore came to the scene only very recently.

Sun et al. [SRY08] were one of the first to use 3D data for action unit detection. They used an active appearance model to track 83 predefined positions and then a set of Hidden Markov Models to classify the displacements of these points across a video of 6 frames. They achieved a recognition rate of 87.1% with the tracked points, and 89.5% with manually labelled points across eight action units. Moreover, confirming the findings of [SST12], they achieved better results with 3D data than 2D data. Zhao et al. [ZDCS10] used a patch-based ‘Statistical Facial FeAture Model’ (SFAM) to build features based on three types of data: landmark configuration, local texture, and local geometry. Their classification scheme used principal component analysis (PCA) to generate a set of coefficients characterizing the current example, and then a score for each action unit indicating how close the example is in the feature space. Savran & Sarkur [SS09] also consider geometric features, but first align their data examples with a non-rigid registration algorithm, such that changes are properly measured with respect to consistent points on the face. They then include a host of both 2D and 3D features in their feature vector. In addition, they use GenBoost, a technique that discovers and emphasizes features that more profoundly influence detection accuracy. This technique achieves an average AuC of 96.2% over 22 action units.

Finally, several works have devised a family of features based on the Local Binary Patterns (LBP) model [OPM02]. This paradigm, originally developed for use with 2D images, has proven quite successful, generating impressive results with a features that are relatively simple to compute. The original Local Binary Patterns feature was defined as

\[
LBP_{P,R} = \sum_{p=0}^{P-1} (g_p - g_c)^{2^p} \\
s(x) = \begin{cases} 
1 & x \geq 0 \\
0 & x < 0 
\end{cases},
\]

where \( P \) is the number of surrounding pixels (typically 8) and \( R \) is the radius (typically 1). The feature forms a bit pattern based on comparison of pixel values of surrounding pixels and center pixels as seen in figure 1.6. The representation is compact, and, because it only considers sign, as opposed to real values, it is relatively illumination invariant, though still desirably sensitive to texture. Later works took this core idea and extended it to the 3D setting. 3D Local Binary Patterns of Huang et al. [HWT06]
observes differences in depth value instead of color value. Sandbach et al. proposed the Local Normal Binary Patterns [SZP12b] feature, which takes surface normals and compare the angle differences between those of the the surrounding depth pixels and the center pixels. The same team later suggested a wide array of variations on the LBP theme for use in the 3D setting including Local Azimuthal Binary Patterns, Local Phase Quantizers, Gabor Binary Patterns, and Monogenic Binary Patterns [SZP12a]. More recently, Bayramoğlu et al. achieved state-of-the-art results using Center-Symmetric 3D Local Binary Patterns (CS-3DLBP) and Ratio-Based Geometry (RBG) features in conjunction with the Random Forests classifier [BZP13]. CS-3DLBP itself measures fluctuations of normal angles, and ‘Center-Symmetric’ refers to its comparison of normals on opposing sides of the pixel of interest, reducing the length of the bit pattern to $N/2$.

CS-3DLBP is defined as

$$CS-3DLBP_{R,N,\theta_t}(p) = \sum_{i=0}^{N/2-1} s(\vec{n}_i, \vec{n}_{i+N/2})2^i$$ (1.2)

where $\vec{n}_i$ are surface normals at $N$ equally-spaced points on a circle of radius $R$ surrounding point $p$, and $\theta_t$ is a threshold angle. The result is an $N/2$-bit binary number associated with $p$. See figure 1.7a. The method does not use these values directly, however; the input depth map is instead broken into a $5 \times 5$ grid, and then histograms of the CS-3DLBP values in each grid location are concatenated to form the final feature vector. Since Equation (1.2) can generate $2^{N/2}$ unique bit patterns, feature vectors for the common case of $N = 8$ will contain $16 \times 5 \times 5 = 400$ entries.

In addition to this, RBG consists of 24 values that are based on ratios of certain distances, areas and angles on each input face. Ratios are used in an attempt to make this feature subject-independent. Some examples are shown in figure 1.7b. Finally, this work uses Random Forests for classification. This classifier has gained popularity recently due to its relative efficiency, as well as its build-in feature boosting properties. Using both features in unison and classifying them with the Random Forests classifier,
Figure 1.7: (a) Construction of the orientation-based CS-3DLBP operator. (b) First four Ratio-Based Geometry features.

the team achieves state-of-the-art results.

**Normalization**

As we have seen, variability due to physiognomy does generally contribute undesired variance in the data. For this reason much of the literature discusses the notion of subject independence. Here we discuss some previous approaches.

The most popular method for attempting subject independence is to use six- or nine-DOF alignment. Works such as [SRY08], [LB05], [LCKL98] and [ZDCS10] preprocess the data with an affine transform to remove variations in scale, rotation and position. The transformation can be calculated directly using facial landmarks with known positions, via Principle Component Analysis (PCA), or via Iterated Closest Point (ICP). This approach is so widely used because it is relatively simple to implement and
provides imperfect but satisfactory alignment between faces.

Savran & Sankur make an impressive attempt to reduce subject variability in [SS09]. They first map each example depth map to the plane, and then use a 2D non-rigid registration algorithm to achieve a dense correspondence between points on a reference example and the current input example. They then resample the latter in the space of the former. This allows them to be sure that points associated with a given example face are always compared to properly corresponding points on other example faces.

This process allows them to reduce the first kind of variability of the two mentioned above, by ensuring a pointwise alignment between any two given faces. However, this technique does not take care of measurement variability, the second kind of physiognomy-based variability. Roughly speaking, their method compares the proper points, but the values of the features associated with those points are still influenced by the physiognomy.

Some previous research has attempted to attack this issue not at the raw data but at the feature design stage. For example, Soyel & Demirel take their feature vector and divide by the horizontal spanning distance [SD07]. This technique transforms all features to be measured in units of some singular measurement on the face. While this works for features based on Euclidean distances, it will distort non-linear differences between two physiognomies. Filateri & Malassiotis’s [TM10] rule-based method is based on ratios of measurements, as is [BZP13], who devised their twenty-four Ratio Based Geometry features off explicit ratios of distances, areas and angles. This helps improve classification accuracy as well for similar reasons, and the fact that there are so many helps their informativeness and discrimination. Ultimately though, these are features based on specific facial locations, and while effective, their construction is heavily engineered and somewhat ad-hoc. There must be a more natural solution to the problem of subject independence. This in fact is the problem we propose to solve. As such, the contribution in this work is not so much an algorithm for detecting action units as it is a pre-processing step to normalize the data towards natural subject independence.
Chapter 2

Preliminaries

The current work describes a technique for normalization of facial surfaces to approach ideal subject independence. To describe what we mean by normalization on a surface, we need to provide a foundation of mathematical and geometric concepts. The current chapter is intended to give intuitive feel for the mathematical concepts at work.

2.1 Differential Geometry

Much of the mathematical machinery underlying the computational geometric tools used in this work comes from the field of Differential Geometry. In this branch of mathematics, we study the inherent characteristics of curves, surfaces, and higher dimensional geometric objects, and how to perform calculations and manipulations on them using concepts from calculus and linear algebra.

Whereas standard calculus considers functions on flat or orthogonal domains such as \( \mathbb{R}^N \), Differential Geometry enables the generalization of these concepts such that our function domains can be curved surfaces or higher dimensional spaces. This advance provides a rigorous mathematical framework that solidifies the notion of functions “living” on surfaces and other spaces. Applications have included fluid dynamics, Einstein’s General Theory of Relativity, and shape analysis.

2.2 Manifold

One of the most important objects in differential geometry is the manifold. A manifold of dimension \( N \) is an object that at any point can be charted unambiguously with \( N \) variables. Put formally, it is a topological space homeomorphic (i.e. it can be bijectively mapped) to \( \mathbb{R}^N \) near each point. In other words, if we zoom in close enough to an \( N \)-dimensional manifold, the arrangement of points looks like that of \( N \)-dimensional Euclidean space, or \( \mathbb{R}^N \).

A topological space, moreover, is a set of objects with an abstract notion of proximity and therefore connectedness of these objects, but without an explicit notion of distance.
or measure. Since a manifold is merely a topological space, it bears no notion of shape, rather only connectivity of the contained points with respect to one another. In fact, this connectivity defines the manifold. For a given manifold, we are allowed to stretch, bend, crumple and distort it to our heart’s consent; so long as we don’t change connectivity, it is considered not to have changed.

To gain an intuitive feel for this abstract object, we can picture an (infinitely thin) string $\mathcal{B}$ tangled up into a ball as in Figure 2.1(a). Even though $\mathcal{B}$ it is completely tangled, each point on the string aside from the boundary is connected to exactly two other points on the string. With some effort we can tease apart the string at a neighborhood around a given point and compare it with, or “hold it up against” the number line $\mathbb{R}^1$. Similarly, if we were to parameterize the string with the one-dimensional variable $u$, then for any given point on the string $u_0$, and an arbitrarily small distance $\varepsilon$, it is clear where $u_0 + \varepsilon$ and $u_0 - \varepsilon$ lie. The string $\mathcal{C}$ in 2.1(b), however, meets and fuses with itself. At the intersection point we would need more than one dimension of reals to navigate. It is therefore not a one-dimensional manifold. Nor is it a two-dimensional manifold, since to map it properly to $\mathbb{R}^2$, it would require two full dimensions worth of possible directions to travel at each point, yet this is not the case anywhere on $\mathcal{C}$.

Moving to two dimensions, we can picture an inflatable beach toy $\mathcal{G}$ with its air let out. Clearly, the shape is in disarray, much like $\mathcal{B}$ above. Yet no matter how much we squash, crumple, or twist it, we could theoretically still choose any point $p$ and a small neighborhood $\mathcal{N}_\varepsilon(p)$ surrounding it, and tease $\mathcal{G}$ apart such that $\mathcal{N}_\varepsilon(p)$ looks like a plane in $\mathbb{R}^2$. Since at no point does $\mathcal{G}$ fuse with itself, only two dimensions suffice to parameterize it. On the other hand, if $\mathcal{G}$ were made of a material like clay, and began fusing with itself at certain points, say at $(u_0, v_0)$ then at those points it is unclear where $(u_0 + \varepsilon, v_0 + \varepsilon)$ and $(u_0 - \varepsilon, v_0 - \varepsilon)$ would be located. In this case we cannot define a parametrization in $\mathbb{R}^2$ that is consistent and unambiguous at all points on $\mathcal{G}$.

It is also for this reason why a plane in $\mathbb{R}^2$ is not topologically equivalent to a space-filling curve in $\mathbb{R}^1$. On the latter, the points are not connected in the second dimension, and so for any given point, there is only one dimension in which to navigate. We can therefore only map it onto $\mathbb{R}^1$. This concept applies to higher dimensions as well.
Despite that the range of the space-filling function covers the same subset of Euclidean space as a manifold in the next-higher dimension, its connectivity dictates its actual dimensionality.

### 2.3 Riemannian Manifold & Metric

As we have seen, a manifold on its own is a set of points or objects with a specific connectivity, but without any notion of shape. The Riemannian manifold introduces the concept of shape by assigning an object $g$ to each point on the manifold. Specifically, it gives a notion of length to the manifold. For this reason, $g$ is also known as the metric tensor, or simply, the metric. Let $\mathcal{S} : \mathbb{R}^2 \to \mathbb{R}^3$ be a surface parameterized as $\mathcal{S}(u,v) = \{x(u,v), y(u,v), z(u,v)\}$. For a given distance $(du, dv)$ traversed in the parameter space, $g$ informs us of the length $ds$ we will travel along the surface. Without loss of generality, we can confine our analysis to a curve $c(t) = \mathcal{S}(u(t), v(t))$ along our surface and express length along this curve at a point $p = c(0)$ via the arclength element $ds = ||c'(0)||dt$ at that point. We only need $c$ to be differentiable on $t \in [-\varepsilon, \varepsilon]$ to ensure that derivatives exist. Then, via the chain rule we have

$$ds = \left\| S_u \frac{du}{dt} + S_v \frac{dv}{dt} \right\| dt$$

$$= \sqrt{\langle S_u du + S_v dv, S_u du + S_v dv \rangle}$$

$$= \sqrt{\langle S_u, S_u \rangle (du)^2 + 2\langle S_u, S_v \rangle dudv + \langle S_v, S_v \rangle (dv)^2}, \quad (2.1)$$

where $S_u$ and $S_v$ are the partial derivatives of $\mathcal{S}$ with respect to $u$ and $v$, respectively. $S_u$ and $S_v$, moreover, are vectors that form, or span, the tangent plane at $p \in \mathcal{S} \setminus T_p\mathcal{S}$.

Equation (2.1) can be written in matrix form as

$$ds^2 = [du \ dv] J^T J \begin{bmatrix} du \\ dv \end{bmatrix} \quad (2.2)$$

where $J = [S_u \ S_v]$ is the Jacobian. The Jacobian itself holds the locally-linear slopes scaling each dimension in the domain to each dimension in the range. $J^T J$, meanwhile, is our metric $g$. Hence it is now clear how the metric encodes length transformations between the parameter space and the manifold.

### 2.4 Gaussian Curvature

A fundamental property of a given space is how curved it is. On a space curve (a one-dimensional manifold), curvature is defined as the magnitude of the rate of change of the tangent vector with respect to an arclength (unit speed) parametrization, and expressed as $\kappa(s)$. It can also be expressed as the reciprocal of the radius of the instantaneously circumscribable, or osculating, circle.
Moving to surfaces, at any given point $p$, we can choose a direction on the tangent plane and measure the curvature of a curve embedded on the surface that passes through point $p$ and whose tangent vector is parallel to our chosen direction. For any $p$, the directions of maximal and minimal curvature are known as the principal directions, and are known to be orthogonal to one another [DC76]. Further, the curvatures themselves, $\kappa_{\text{max}}$ and $\kappa_{\text{min}}$, are known as the principal curvatures.

There are actually many different ways to define curvature on a surface. Perhaps the most important is called Gaussian curvature, and is given by the product of the principle curvatures, $\kappa_{\text{max}}\kappa_{\text{min}}$. Also noteworthy is the mean curvature $H = \frac{1}{2}(\kappa_{\text{max}} + \kappa_{\text{min}})$.

A plane has zero Gaussian curvature. This seems intuitive, but so does a cylinder, since curves along its length are flat. A sphere, moreover, has the same non-zero Gaussian curvature at every point, while that of a saddle point assumes negative values.

The cylinder is an interesting case, since like the plane it also has zero Gaussian curvature everywhere (notice that this is not the case for mean curvature). The fact that these two seemingly different surfaces have the same Gaussian curvature is significant; it can be shown that two surfaces with identical Gaussian curvature can be transformed into one another without stretching or bending, specifically, without altering lengths. This is usually not the case; for example, one cannot perform this transformation from a sphere to a plane. This is why there are so many different techniques to draw a map of the Earth: each technique typically tries to minimize a different kind of distortion while projecting the distances on the globe to a planar map. Indeed, the Gaussian curvature is a special measurement as we will see in the coming sections.

### 2.5 Metric Space

Now that we have established a notion of length on the manifold, we can begin to perform measurements on it. In this sense, the Riemannian manifold is a metric space. A metric space is a set of objects equipped with a notion of distance between these objects. More formally, it is a pair $(X, d)$, where $X$ is a topological space, and $d$ is a distance. Distance is defined as a function $d : M \times M \to \mathbb{R}$ that upholds the following properties [Cho92]:

1. **Non-negativity:** $d(x, y) \geq 0 \ \forall x, y \in M$.
2. **Identity of discernibles:** $d(x, y) = 0 \iff x = y \ \forall x, y \in M$.
3. **Symmetry:** $d(x, y) = d(y, x) \ \forall x, y \in M$.
4. **Triangle inequality preservation:** $d(x, z) \leq d(x, y) + d(y, z) \ \forall x, y, z \in M$.

This formulation affords us some flexibility and creativity, in that different distances can be defined that are better suited to certain applications and analysis. For example, the most popular distance is the Euclidean one, typically expressed as the $\ell_2$ norm.
\|x\|_2 = \sqrt{\sum_i x_i^2}. The Pythagorean Theorem is a specific case of this. However, other applications find use of the \( \ell_1 \) norm \( \|x\|_1 = \sum_i |x_i| \), otherwise known as the *Manhattan* or *taxicab* distance. When we begin speaking of curved spaces, the notions of near and far vary wildly depending on how we define distance. Consider a very tall, yet thin wall, with two friends situated on either side. The Euclidean distance between them informs us that they are very close to one another. However, to unite, they must climb over the wall. In this case, the shortest path between them is much larger. This notion of in-surface shortest path between two points is known as the geodesic. It is induced by the metric structure of the surface; since the metric describes infinitessimal lengths at each point, it also helps define which paths between two points are the shortest. Each pairing of a manifold with a specific kind of distance is considered a different metric space, and we are free to use whichever combination best suits the needs of a given problem. The metric space is therefore a powerful mathematical tool providing us with a rigorous definition of what it means to measure.

### 2.6 Embedding Spaces and Isometries

In mathematical and geometric analysis it is often convenient to move certain objects to and from different spaces, or to express them in more convenient ways. An embedding is a means to do so. Embeddings are injective mappings \( f : X \rightarrow Y \) that preserve connectivity, algebraic properties, or some other structure, where the notion of “structure” is domain dependent. They allow us to take the same mathematical object and apply some transform to it while still retaining its essence. The injectivity of the embedding requires that after the mapping, distinct objects in the source space remain distinct in the target space. In addition, sometimes the embedding is reversible, meaning that we can reconstruct the original object from the embedding space.

In typical scenarios, we analyze surfaces in terms of their coordinates, say \((x, y, z)\) in Euclidean space. Surfaces essentially two-dimensional manifolds, since as we saw above, we can unambiguously parameterize them in \(\mathbb{R}^2\). In this case, we say that \(S\) is *embedded* in \(\mathbb{R}^3\). Further, we say that \(\mathbb{R}^3\) is the *ambient space*. The object in question is fundamentally two-dimensional, (sometimes called *bivariate*), yet we expressing it using three dimensions.

An embedding that preserves distances is called an *isometry*, or *isometric* embedding. Naturally, because of their length-preserving property, embeddings of this type also preserve the Riemannian metric. Examples of isometries include rotation and translation. In Chapter 3, we will work with the embedding of surfaces representing faces into other spaces. These embeddings will transform a given face to a form more conducive to deformation analysis while retaining certain properties of the surface such as its topology and latent deformation (i.e., facial expression).
2.7 Intrinsic and Extrinsic Geometry

Differential Geometry describes many different properties of geometric objects. Many of these are calculable using the ambient space into which an object is embedded. Some even depend on it. It turns out, however, that there are some properties that are inherent to the object itself; that is, they are invariant to the ambient space under isometries. The metric is one of these properties; Gaussian curvature is another [DC76]. Properties like these are called *intrinsic* as they are inherent to the structure of the object, and do not depend on the ambient space. Meanwhile, properties that do vary with ambient space are called *extrinsic*. Coordinates themselves are one example in that we can isometrically transform the object, say, via rotation or translation, and the names of the coordinates at each point will change. Mean curvature is another [BKP+10]. We can see this by observing the unrolling of a cylinder into a plane. This operation is an isometry, yet the mean curvature changes. The concepts of intrinsic and extrinsic form the basis for this work. We will see that the physiognomy of a face can be regarded as nearly intrinsic, while the facial expression is approximately extrinsic.
Chapter 3

Method: Intrinsic Normalization

3.1 Introduction

Normalization is known to be an effective means of reducing the variance of observations of a class of phenomena \([\text{BK08}]\). It brings examples into a canonical form by removing variations which are not manifestations of the class of interest. In doing so, it reduces the dimensionality and the number of DOFs necessary for a mathematical model to successfully identify an instance of the class. Because which dimensions are important varies application to application, so too the normalization technique is application dependent. By normalizing data we are essentially deciding on a canonical form for our class of interest and transforming all instances such that if a canonical instance ever appeared in the data set, its normalized form would look as we expect.

Example: 2D Digit Recognition

2D digit recognition can serve as a concrete motivating example. Suppose we are attempting to recognize the digit ‘8’. Examples of this digit will vary in many different ways: shape, smoothness, connectivity, and more. With ever more examples our classifier builds a better picture of exactly how this class can vary. However, there is some variability that does not interest us. If two examples have been drawn identically but appear shifted within the image, we would still like to consider them the same version of the digit, i.e. map to the same exact point in feature space. We therefore center all input examples, as the offset should not contribute to the classifier’s knowledge of what an ‘8’ looks like. In fact, considering this offset unnecessarily increases dimensionality of the space, requiring much more data for accurate classification. The same notion applies for rotation and scale, though only to an extent; over-normalization can cause false alarms by allowing the classifier to recognize members of different classes. For example, if we allowed rotational normalization of 90°, our classifier would now accept the infinity symbol ‘∞’ as part of our class. Good normalization attempts to reduce unnecessary variation as much as possible, but only as far as it can maintain the boundaries of the class of interest.
3.2 Isometric Model & Formal Constraints

In chapter 1 we discussed how physiognomy contributes to both alignment variability and measurement variability. We also discussed how previous attempts at subject independent analysis have only roughly tackled these issues. In this chapter we utilize the background we have provided to derive and establish a model for true subject-independent normalization.

Given the problem, we can ask ourselves what would the ideal setup would be if we wished to achieve true subject independence. Clearly, the answer is that we have a single subject perform all possible performances of a given action unit. A set of examples lacking physiognomic variety is subject independent by definition because there is no variation with respect to the subject. Hence, if we could somehow normalize the physiognomy to a single subject, we could create this scenario. But is this possible?

To understand what this scenario means geometrically, we begin by considering the surface of a face as a Riemannian manifold \( M = (T, g) \), where \( T \) is a topological space describing point connectivity, and \( g \) is the metric tensor. As discussed in Chapter 2, the metric tensor describes how lengths on the surface are stretched with respect to its parameterization. \( g \) therefore provides us with a notion of scale associated with each point on \( M \).

Bronstein et al. relate the pointwise metric tensor to the concept of physiognomy via their Isometric Model of Facial Expressions [BBK06a]. This model assumes only that for any given facial expression, the skin stretches only slightly. Geometrically, this assumption implies that for a given subject, the surface of a face during the execution of any facial expression more or less retains its metric structure. Conceptually, it implies that physiognomy is roughly defined by the intrinsic geometry of the face. The contribution due to expression, therefore, can be viewed as a near-isometric deformation of the ambient space in which the geometry is embedded.

To formally describe this model the authors use the geodesic distances \( d_S(\xi_1, \xi_2) \) between any two points \( \xi_1 \) and \( \xi_2 \) on \( S \). Since geodesic distances are induced by the metric structure, one can equivalently describe the metric structure in terms of them. A transformation \( \psi : S \to S' \) is called an isometry if

\[
d_S(\xi_1, \xi_2) = d_{S'}(\psi(\xi_1), \psi(\xi_2)) \tag{3.1}
\]

Using the notion of isometry, the Isometric Model declares that for any expression of a face, the geodesic distance between two isometrically mapped points \( \psi(\xi_1) \) and \( \psi(\xi_2) \) on \( S' \) is approximately the same as the geodesic distance between the original points \( \xi_1 \) and \( \xi_2 \) on the original surface. Figure 3.1 illustrates this concept.

The work in [BBK06a] focuses on finding an expression-invariant representation of faces for the purposes of geometry-based facial recognition. To apply the model to the task, the authors embed the facial manifold in a space using only the geodesic distances
Figure 3.1: The geodesic follows the in-surface shortest path.

(which, again, are induced by the metric structure). They do this via Multidimensional Scaling, an algorithm that finds a set of points in Euclidean space, such that the values of the Euclidean distances between them match the geodesic distances on the original surface. Hence it is an embedding of the distances into Euclidean space. Such an embedding provides a representation that ignores the extrinsic phenomena related to expression. As a result, in this representation all faces of a given subject (i.e. constant physiognomy), regardless of expression, map to roughly the same coordinate in the new space.

We can conceptualize this more easily using our paper roll example from Chapter 2. Suppose we drew several points \( \{p_i\}_{i=1}^{N} \) on this piece of paper. We then proceed to roll it up. Doing so changes the coordinates in the ambient space. However, embedding the geodesic distances in Euclidean space would always produce the same configuration of points (up to a rotation and/or translation). In fact, such an embedding after any isometric manipulation would produce the same point configuration.

In facial expression analysis, we are concerned with the complementary problem. We are trying to learn the distribution of facial expressions, and so we welcome the variability they contribute. The unwanted variance comes then not from the facial expression but from the structure of the physiognomy itself. Ideally we would transform all subjects in our testing and training set into the same subject. Geometrically therefore, we would like to transform our geometry to some space where all faces of a given expression end up mapped to coordinates very close to one another.

We begin by defining \( \mathcal{F} \) to be the set of all possible faces of all possible subjects, and \( \mathcal{F}_s \subset \mathcal{F} \) to be the set of all possible faces of a given subject \( s \). To normalize a given example, we first choose some target canonical physiognomy metric \( g_c \), associated with faces \( \mathcal{F}_c \) which describes the intrinsic structure and therefore equivalently the geodesic distances. We now seek a function

\[
\varphi_{g_c} : \mathcal{F}_i \rightarrow \mathcal{F}_j
\]

such that \( g_j \), the metric associated with \( \mathcal{F}_j \), is similar to that of \( \mathcal{F}_c \). Further, for a given target physiognomy metric \( g_c \) all possible outputs \( \mathcal{F}_j \) should be intrinsically equivalent to one another. In other words, given two faces \( S_1 \in \mathcal{F}_i \) and \( S_2 \in \mathcal{F}_j \) from any subject,
if we normalize them as

\[ T_1 = \varphi_{g_c}(S_1) \]
\[ T_2 = \varphi_{g_c}(S_2), \]  

(3.3)

we would like \( T_1 \) to be intrinsically similar to \( T_2 \) and to any face with metric structure \( g_c \). More formally, given an isometry \( \psi : T_1 \to T_2 \),

\[ d_{T_2}(\psi(a), \psi(b)) \approx d_{T_1}(a, b), \]

(3.4)

where \( d_{T_1} \) and \( d_{T_2} \) are the geodesic distance functions on \( T_1 \) and \( T_2 \) respectively.

The above formulation accounts for the normalization of physiognomy, but imposes no constraints to maintain facial expression. For simplicity, we can characterize facial expression by deviation from some accepted reference. We will use the neutral face, FACS coded as 0, as this reference.

To formalize this notion, we declare \( \mathcal{N} \) the set of neutral faces for all subjects, and \( \mathcal{N}_s \) the element of \( \mathcal{N} \) representing the neutral face for subject \( s \). (Note that \( \mathcal{N}_s \) is an element, i.e., a face, whereas \( \mathcal{F}_s \) represents a set of faces). Finally, we express as \( D_{i,f} \) the nearly-isometric transformation required to deform \( \mathcal{N}_i \) into a face \( f \in \mathcal{F}_i \). We update our function with the appropriate parameters as

\[ \varphi_{g_c} : D_{i,f} \times \mathcal{F}_j \to \mathcal{F}_j \]

(3.5)

to incorporate the constraints just discussed. Equation (3.5) gives a clearer picture of the operation. The instance of \( \mathcal{F}_j \) associated with the domain is used to impose the physiognomy constraint, while \( D_{i,f} \) imposes the expression constraint. The function maps an instance of \( \mathcal{F}_j \) to another instance of \( \mathcal{F}_j \) using the deformation specified by \( D_{i,f} \). We note that \( \varphi_{g_c} \) is agnostic to source physiognomy as the deformation from neutral to the expression face is sufficient information.

If we assume that the neutral face is the canonical expression, we may wish to use it instead of \( \mathcal{F}_j \) to constrain the physiognomy. In this case our final function can be expressed as

\[ \varphi_{g_c} : D_{i,f} \times \mathcal{N}_j \to \mathcal{F}_j. \]

(3.6)

3.3 Implementation

So far we have formally constrained the types of solutions we would accept as viable means of normalizing physiognomy. We now require an algorithm for implementation. Fortunately, advances in computational geometry and digital geometry processing in the past decade have afforded us with tools that can aid in our approach.

In the practical setting, we must discretize the problem. Shapes are described by
Triangulated meshes, which consist of a set of vertices \( V \) and set of polygonal faces, or triangles \( T \), each of which references a set of three vertices from \( V \). The configuration of polygonal faces is referred to as the triangulation. The order of indices in a face matters; typically indices are specified in counter-clockwise order to indicate an outward-facing face. Specifying this convention provides us with a means to have connectivity produce a triangulation that gives consistent orientation.

In addition, there are two kinds of normals associated with these meshes. First, face normals are orthogonal to their corresponding triangles. They can be computed by taking the cross product of the edges of their corresponding triangles. The order here again matters as cross products are taken following the right-hand rule. Second, vertex normals ascribe an orientation to each vertex. These are not well-defined, and different applications will call for different computations of vertex normals. However, a typical scheme is to take the average of the surrounding face normals, and weight each by the area of its associated triangle. See figure 3.2 for an illustration.

With this setup, we have meshes that are piecewise linear approximations to our surfaces \( S, S', T, T', \ldots \). Considering them as such provides a coherent mathematical framework for discretization of solutions originally proposed in the continuous setting.

### 3.3.1 Deformation Transfer

Deformation transfer is a useful technique developed over the past decade. It is defined as follows. Given a source shape \( S \), a deformed source shape \( S' \) and a target shape \( T \), find an analogously-deformed \( T' \). Deformation transfer was recently applied in [WLVP09], a synthesis problem attempting to transfer expression. The authors create an intermediate neutral face with which to transfer expression to an arbitrary avatar. Here, we apply it to our analysis problem. If \( S \) represents the source subject’s neutral face, \( S' \) the source subject’s expression face, and \( T \) the target subject’s neutral face, we can see that deformation transfer is a viable solution. Assuming we properly capture the deformation
between $S$ and $S'$, via the Isometric Model, this deformation is approximately an isometry. In this sense, deformation transfer successfully embeds the expression face in a new manifold that is approximately isometric to faces with the desired physiognomy, and whose expression similarly deviates from neutral. Over the past decade, the problem of deformation transfer has seen some viable solutions proposed. We used [BSPG06] in this work. We discuss derivations and other solutions to the deformation transfer problem in Chapter 4.

### 3.3.2 Non-Rigid Registration

Deformation transfer requires all input shapes to be in dense correspondence with one another. This implies not only that $S$, $S'$ and $T$ have the same triangulation, but that any vertex $v_i$ is associated with the same point on all corresponded surfaces.

Registration has its origins in 2D computer vision as a technique to find dense, pixel-to-pixel correspondences between images. More recent research has investigated the problem in the 3D domain. We use one such solution to build dense correspondences between faces we will use in the deformation transfer algorithm.

The solution we use is based on the non-rigid registration algorithm of Li et al. [LSP08] to align all face instances. Non-rigid registration operates by deforming one mesh into another, producing a dense point correspondence, or alignment, in the process. Here, we deform our chosen template face $T$ to align with all input examples. Hence, for each vertex on $T$, the algorithm gives us the nearest corresponding vertex on the example face. In the process, we receive a new shape that represents the example face in terms of $T$’s triangulation. Non-rigid registration will be further discussed in Chapter 4.

### 3.3.3 Laplace-Beltrami Smoothing

Roughly speaking, deformation transfer “copies” gradient deformations from one pair of meshes and “pastes” them onto a second. It is evident that any noise in either or both of the source meshes will cause sharp gradients, and this will propagate to the target. moreover, if the target is also somewhat noisy, the noise will be compounded during algorithm execution. We must therefore perform smoothing on all surfaces. However, we must be wary of order of operations. Running non-rigid registration after smoothing risks re-introducing noise into the mesh. We therefore smooth immediately before deformation transfer. However, we must also impose the constraint that such a smoothing process should not significantly change the geometry, lest we distort the alignment generated by the registration process. To this end, we use Laplace-Beltrami smoothing [BKP+10][DMSB99]. This method updates the $N$ vertex positions \($\{v_1, ..., v_N\}$ of a given mesh via the algorithm

\[
v_i = v_i + h\lambda\Delta v_i.
\]

(3.7)
Here, $\Delta v_i = -2Hn$, where $H$ is the mean curvature as seen in Chapter 2, $n$ is the normal to the surface at $v_i$, $h$ is a sufficiently small time step, and $\lambda$ is a scalar diffusion coefficient. $\Delta$ is the Laplace-Beltrami operator, the generalization of the Laplacian for arbitrary Riemannian manifolds. Each $v_i$ is therefore displaced in the direction of $n$, and undergoes no translation in the tangential direction (see fig. 3.3). This is a crucial property for our purposes as the orthogonal projection of each triangle remains constant. We thus suppress sharp deviations from the surface while mostly retaining triangle shape. In practice, we do not calculate $H$ explicitly. Rather, we implement this method by discretizing the Laplace-Beltrami operator using the popular cotangent weight scheme (see Chapter 4) [MDSB03][BS08]. Using these weights, $v_i$ is moved in the expected normal direction and the desired smoothing is achieved.

Finally, to keep the overall geometry the same size, we impose the additional constraint that the boundary of the shape stays the same during smoothing. Hence we require

$$\partial S_{\text{smooth}} = \partial S \quad (3.8)$$
$$\partial S'_{\text{smooth}} = \partial S' \quad (3.9)$$
$$\partial T_{\text{smooth}} = \partial T. \quad (3.10)$$

### 3.3.4 Boundary Constraints

Our implementation for deformation transfer solves a Poisson equation via constraints on the gradients of the mesh faces (see Chapter 4). Because we are using non-water-tight shapes (i.e., shapes with boundaries), we must constrain behavior at the boundaries. There are two ways of imposing boundary constraints; Dirichlet conditions specify values the solution itself needs to take, while Neumann conditions specify values the derivative of the solution must take. We used Dirichlet boundary conditions, as we would like the vertices of the solution to conform to a boundary we specify. Practically, this implies specifying that the boundary of the resulting $T'$ should be the same boundary as the
original template face $T$. Thus, we impose the constraint

$$ \partial T' = \partial T. \quad (3.11) $$

3.3.5 Template Face

The final piece of our framework is the target $T$, which specifies the physiognomy to which we normalize. To this end, we constructed a smooth template face in Autodesk Maya and exported it as a depth map. We then imported this depth map into MATLAB where we adjusted its scale to roughly match the examples in the Bosphorus Database. Figure 3.4 shows a rendering of the template face.

![Figure 3.4: A rendering of the template mesh.](image)

3.3.6 Intrinsic Normalization Algorithm

Our normalization algorithm therefore proceeds as follows. Choose or build a template face $T$, and make sure it is smooth. For each example face $S'$, retrieve the neutral face $S$ of the corresponding subject. Register $S$ and $S'$ with $T$. Run Laplace-Beltrami smoothing on them without modifying the boundary vertices. Finally, run deformation transfer on $S$, $S'$ and $T$, making sure $T'$ keeps the boundary of $T$. Algorithm 3.1 is the procedure performed on each example. Here all inputs $S$, $S'$ and $T$ are triangulated meshes. Note that the step of creating the template face is left out as it is a one-time operation.
Algorithm 3.1 Intrinsic Normalization

1: $\tilde{S} \leftarrow \text{NonRigidRegistration}(T, S)$
2: $\tilde{S}' \leftarrow \text{NonRigidRegistration}(T, S')$
3: $\tilde{S}_{\text{smooth}} \leftarrow \text{LaplaceBeltramiSmooth}(\tilde{S})$
4: $\tilde{S}'_{\text{smooth}} \leftarrow \text{LaplaceBeltramiSmooth}(\tilde{S}')$
5: $T' \leftarrow \text{DefTrans}(\tilde{S}_{\text{smooth}}, \tilde{S}'_{\text{smooth}}, T_{\text{smooth}})$
6: return $T'$
Chapter 4

Supporting Algorithms

4.1 Introduction

While discussing implementation of Intrinsic Normalization in Chapter 3 we mentioned a few geometric methods that have been explored in the literature. One was deformation transfer, which is used to find analogous deformations of two corresponding source shapes. Another was a non-rigid registration algorithm, which helps us find dense correspondences between two source shapes. In this chapter we discuss their derivation and implementation.

4.2 Deformation Transfer

Deformation transfer is a geometric technique that has seen a few treatments over the past several years \cite{SP04,BSPG06,BCWG09}. The goal of the technique can be stated as follows: Given two analogous initial shapes $S$ and $T$, and $S'$, representing a deformed version of the first, find $T'$, the analogously deformed version of the second. $S$ and $S'$ are called the source and deformed source meshes respectively, while $T$ and $T'$ are called the target and deformed target meshes respectively. In our context, all shapes are triangulated meshes, meaning that each has a set of vertices $V$, as well as a set of faces $F$, where each face $f \in F$ is a triangle consisting of a set of three vertices $\{v_1, v_2, v_3\} \in V$. $F$ is also called the triangulation of the mesh.

This problem as stated is more involved than meets the eye. One cannot simply take vertex displacements from one shape and add them as vectors to the other because the displacements are highly dependent on intrinsic characteristics. Nor can one merely deform the triangles of the target mesh in the same manner as the source because there is no guarantee that all vertices will align on the final shape. In order to successfully find analogous deformations one must impose constraints that take the geometry into consideration.
4.2.1 Vertex-based Approach

Sumner & Popović [SP04] construct a solution that explicitly handles the above two objectives. Namely, to transfer the change each triangle undergoes, and to ensure that triangles remain joined at the same vertices. They represent deformation as a set of affine transformations, each given by a pair \((Q, d)\), where \(Q\) is a \(3 \times 3\) non-translational matrix and \(d\) is a translational vector. In addition, \(v_i\) and \(\tilde{v}_i\), \(i \in 1 \ldots 3\) represent the undeformed and deformed vertices, respectively, of a given triangle. Because we are interested in affine transformations and not simply rotations, three vertices do not suffice. Hence they construct a fourth vertex \(v_4 = v_1 + (v_2 - v_1) \times (v_3 - v_1) / \sqrt{[(v_2 - v_1) \times (v_3 - v_1)]}\), (and similarly for \(\tilde{v}_4\)), which is essentially the first vertex translated unit distance in the direction of the normal.

That \((Q, d)\) is an affine transformation means that

\[ Qv_i + d = \tilde{v}_i, \ i \in 1 \ldots 4. \quad (4.1) \]

Subtracting the first of these from the rest gives us \(QV = \tilde{V}\), where

\[ V = [(v_2 - v_1) (v_3 - v_1) (v_4 - v_1)], \quad (4.2) \]

and likewise for \(\tilde{V}\). \(Q\) is therefore given by \(Q = \tilde{VV}^{-1}\). With this, we now have a way to calculate the source transformations.

Deformation transfer relies on a dense correspondence between shapes. To represent the correspondence, the authors build a mapping \(M = \{(s_1, t_1), \ldots, (s_{|M|}, t_{|M|})\}\), each element of which pairs the source triangle indexed as \(s_i\) with the target triangle indexed as \(t_i\). Moreover, \(S_{s_i}\) is the non-translational component of the affine transformation for triangle \(s_i\) on the source shape, while \(T_{t_i}\) represents the same concept for triangle \(t_i\) on the target shape.

Returning to our requirements, we may now formulate the problem as the optimization of

\[
\min_{(T_1, d_1), \ldots, (T_{|T|}, d_{|T|})} \sum_{j=1}^{|M|} \|S_{s_j} - T_{t_j}\|^2_F
\]

subject to \(T_jv_i + d_j = T_kv_i + d_k, \ \forall i, \forall j, k \in \mathcal{N}(v_i),\)

where \(\mathcal{N}(v_i)\) is the set of all triangles that share vertex \(v_i\). This expression is minimized when all (non-translational) transformations \(T_{t_j}\) are as close as possible to those of their corresponding source triangles. The constraints, meanwhile, enforce that for any two triangles surrounding a given vertex, their associated affine transformation must bring this vertex to the same position.

34
4.2.2 Poisson-based Approach

Botsch et al. also employ a minimization process, but base their method on the gradient field manipulation technique from [YZX+04]. This technique relies on the Poisson equation, which allows us to search for a solution function while imposing constraints on its gradients instead of transforming the vertices or the triangles themselves. This allows more flexible and smooth manipulation.

We find a solution for a surface given \( D_s \), the deformations to the gradients at each point \( s \). This is useful, for example, in manually editing surfaces in a coordinate-independent manner. A graphical user interface could supply the user with a convenient way to modify the gradients at each point of a given surface, and the program would use these transformations as user input to reconstruct the surface.

To apply this method to deformation transfer, our deformations come not from the user but rather from the changes between the source face \( S \) and our deformed source face \( S' \).

**Continuous Derivation**

To derive this technique, we start in the continuous setting by considering a surface as a two dimensional Riemannian manifold, expressible parametrically as \( S : \Omega \in \mathbb{R}^2 \rightarrow \mathbb{R}^3 \), and in terms of its coordinates as \( S(u, v) = \{x(u, v), y(u, v), z(u, v)\} \). At each point \( s \in S \) there exists a tangent plane \( T_sS \) spanned by \( S_u \) and \( S_v \). In this tangent plane also reside the gradients of the coordinate functions at \( s \): \( \nabla x, \nabla y \) and \( \nabla z \). The common description of gradient, path of steepest descent, is extended to non-orthogonal manifolds by expressing it in terms of the metric (see Chapter 2). It is important to note that we treat each of \( x(u, v) \), \( y(u, v) \), and \( z(u, v) \) as any other functions on the surface of the manifold that adhere to its intrinsic characteristics.

We can express a deformation of the surface at each point \( s \) by a transformation \( D_s \) of the gradient field. The same \( D_s \) will operate on all coordinate function gradients at \( s \). Now, we seek new coordinate functions \( \hat{x}, \hat{y}, \hat{z} \) that represent the reconstruction of the surface \( T \). The main constraint on these new coordinate functions is that their gradients should be as close as possible to the deformed versions of \( \nabla x, \nabla y \) and \( \nabla z \), in a least-squares sense. Thus we can formulate the energy to be minimized as

\[
E = \int_S \| \nabla \hat{x}(s) - D_s(\nabla x(s)) \|^2 da(s),
\]

and similarly for \( \hat{y} \) and \( \hat{z} \). Here, \( da(s) \) is an area element about \( s \). For clarity, let \( G(s) = D_s(\nabla x(s)) \). \( G(s) \) is also known as the guidance vector field, since the gradient field of our reconstructed shape will try to approximate it.

The minimizer of the above functional,

\[
\Delta S \hat{x} = \text{div} G,
\]

35
Figure 4.1: Left to right: Hat basis functions $\phi_i, \phi_j$ and $\phi_k$ for the same triangle. The height represents the value of the function. For each index $q$ of the mesh there is a piecewise linear function $\phi_q$ that is of unity value at vertex $v_q$, that is of zero value at all other vertices, and that linearly interpolates in between.

is known as the Poisson equation, and is known to be numerically solvable. Here, $\Delta_S$ is the Laplace-Beltrami operator, the generalization of the Laplacian for non-orthogonal manifolds. The divergence operator here as well is similarly defined on the surface. By solving equation (4.5) for $\hat{y}$ and $\hat{z}$ as well, we retrieve the coordinates of $T$. In the next section we will see how to discretize the gradient, divergence, and Laplace-Beltrami operators for practical use.

**Discretization**

For our purposes, we use deformation transfer on a triangulated mesh. The mesh consists of a set of vertices $v_i \in V, i = 1 \ldots |V|$, and triangles, or faces $f_i \in F, i = 1 \ldots |F|$. The core of our move from continuous to discrete mathematics lies in that we can think of each face as a piecewise linear local approximation to the surface $S$, as is done in finite element analysis. In this way, we can represent our coordinate functions as

$$x(s) = \sum_{i=1}^{|V|} x_i \phi_i(s)$$

(4.6)

and likewise for $y$ and $z$. Here $x_i$ is the $x$-value at vertex $v_i$, $\phi$ is the popular “hat” basis, whose scalar functions $\phi_i$ resolve to unity at corresponding vertex $v_i$, 0 at all other vertices, and linearly interpolated in between. A visual representation of the “hat” basis is given in fig 4.1. Note that functions $\phi_i$ are coordinate weights. This allows for a barycentric interpolation across each triangle, while maintaining that $x(v_i) = x_i$. Then for any given triangle, the function value only depends on the basis functions associated with the three vertices in the triangle. With this representation, we can derive the discrete versions of some useful differential operators [BKP+10]. First, the gradient is expressed as

$$\nabla x(s) = \sum_{i=1}^{|V|} x_i \nabla \phi_i(s),$$

(4.7)

where $\nabla \phi_i$ is a vector in $\mathbb{R}^3$. Moreover, since within each triangle $T$, $\phi_i$ is unity at one vertex and zero at the others, the gradient of basis function $\phi_i$ is orthogonal to the
Figure 4.2: Illustration of $\alpha_{i,j}$ and $\beta_{i,j}$ as specified by the cotangent weight scheme.

opposite to the edge connecting $v_j$ and $v_k$. \cite{BKP+10} defines it as

$$\nabla \phi_i(v) = \frac{(v_k - v_j) \perp}{2 A_T}. \quad (4.8)$$

where $\perp$ indicates a counterclockwise rotation by $90^\circ$ in the plane of the triangle, and $A_T$ is the area of triangle $T$. Note that the gradient lives on the surface of the triangle.

Meanwhile, the divergence of a vector field $w : S \to \mathbb{R}^3$ can be given by

$$\nabla \cdot w(v_i) = \frac{1}{A_i} \sum_{T \in \mathcal{N}_i} \nabla \phi_i \bigg|_T \cdot w_T A_T, \quad (4.9)$$

where $A_i$ is the area of the local Voronoi region of vertex $v_i$, $\mathcal{N}_i$ are the triangles surrounding $v_i$, $w_T$ is the value of $w$ at triangle $T$ and $A_T$ is the area of triangle $T$. For a derivation of this expression, see \cite{TLHD03}.

Finally, the Laplace-Beltrami operator is the generalization of the Laplacian operator for non-orthogonal domains. The most popular discretization is the so-called cotangent weight scheme given by

$$\Delta x(v_i) = \frac{1}{2A_i} \sum_{v_j \in \mathcal{N}_i(v_i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})(x_j - x_i), \quad (4.10)$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the angles opposite the edge connecting vertices $v_i$ and $v_j$ as shown in figure 4.2. For more details on derivations, see \cite{BKP+10} and \cite{TLHD03}.

Typically, the Laplace-Beltrami operator is separated into two matrices $L = A^{-1}W$, where $A$ is an $|F| \times |F|$ matrix with the Voronoi area of each triangle on the diagonal entries, and $W$ is the Laplace-Beltrami weighting matrix consisting of the remainder of Equation (4.10).

The only question that remains is how to implement $J$. Again, this variable represents the gradients of the triangles of the target face deformed via the same transformations that bring $S \to S'$. We see in \cite{BSPG06} that while the gradient-based deformation algorithm implements $J$ as $G'_j := G_j S_j$, where, for triangle $j$, $S_j$ is the rotation/scaling matrix and $G_j$ is the gradient, deformation transfer uses only $S_j^T$, as it leads to the same least squares results. Since $S_j$ is a simple rotation/scaling transformation, it can
be found by solving

\[ S_j = (q'_1 - q'_3, q'_2 - q'_3, n') \cdot (q_1 - q_3, q_2 - q_3, n)^{-1} \]  

(4.11)

where \( q_j \) and \( q'_j \) are corresponding vertices on the source and deformed source triangles, respectively, and \( n \) and \( n' \) are their respective unit triangle normals, which can be found by taking the cross product of their edges.

With this last step we have discretized the Poisson equation of (4.5). We see that our resulting system is a sparse linear system. It is well known that we can solve this Poisson system using the Conjugate Gradient method, and further, there are several packages available that provide their own implementations. MATLAB itself provides a solution using the LMDIVIDE routine, or equivalently its symbol the backslash operator \.

4.3 Non-rigid Registration

The deformation transfer algorithm requires that the three meshes approximating surfaces of interest, \( S, S' \) and \( T \), are in dense correspondence with one another. This means that for any point \( p \) on one surface, we can find its corresponding point \( \tilde{p} \) on the second surface. The problem is somewhat ill-defined as the definition of “corresponding” is semantic and can therefore vary with application. It is also not always clear which points on a target surface should correspond with a given point on the source surface, especially when both surfaces have planar areas. In our case, we would like all features of the face to be associated with those same semantic features on the other face. In our setting, in which each face is represented by a triangulated by a triangulated mesh, we can look to the geometry itself to provide us with unique features to correspond. Fortunately, in our problem, the surfaces almost always have non-zero Gaussian curvature, indicating that there is some feature salience to compare.

This problem has received much treatment in the literature. Li et al. [LSP08] formulate the problem as an energy minimization problem. Shtern et al. uses a statistical approach towards matching in the spectral domain [SK14]. Bronstein et al. treat each surface as a metric space and use their Generalized Multidimensional Scaling to find the embedding of one metric space in the other that minimizes the Gromov-Hausdorff distance [BBK06b].

We use an algorithm based on energy minimization [WLVP09], and include several energy terms. First, the point-to-point metric [BM92], given by

\[ E_{pt} = \sum_{i=1}^{N} \alpha \| v_i - c_i \|_2^2, \]  

(4.12)

is minimized with the spatial difference between corresponding points across all vertices on the surface. Here, \( v_i \) is a vertex on the source mesh, \( c_i \) is a vertex on the target
mesh, and $\alpha$ is a weighting term. Next, the point-to-plane metric
\begin{equation}
E_{\text{plane}} = \sum_{i=1}^{N} |n_{c_i}^T(v_i - c_i)|^2, \tag{4.13}
\end{equation}
originally used by Chen and Medioni [CM91], includes $n_{c_i}$, the normal to the target surface at $c_i$. This term allows flat regions to slide along each other. The vector connecting the two vertices is perpendicular to the normal at the target. This penalizes alignment issues. A membrane energy term
\begin{equation}
E_{\text{memb}} = \sum_{i \in V} \|\Delta_S d_i\|^2 \tag{4.14}
\end{equation}
rewards smoothness. Here $d_i = \tilde{v}_i - v_i$ are displacement vectors, where $\tilde{v}_i$ is the current deformed version of $v_i$. Also, $\Delta_S$ is the Laplace-Beltrami operator, which essentially measures local curvature. The lower the local curvature, the smoother the region.

Finally, a reference point energy term
\begin{equation}
E_{\text{ref}} = \sum_{i} \|v_i - r_i\|^2, \tag{4.15}
\end{equation}
where $r_i$ is one of a sparse set of labeled reference points, is another version of the point-to-point energy, but with points we know a priori to correspond with one another. The Bosphorus database supplies at least 22 landmarks per face that we use as these reference points. In this case they are a result of a manual step, which would prevent us from a fully automatic system. However, recent works such as [PPTK13] have provided advances in automatic facial landmark detection. Assuming these are reliable, we could use the output of these algorithms for a fully automatic system. Thus the total energy of the system is
\begin{equation}
E_{\text{tot}} = E_{\text{pt}} + E_{\text{plane}} + E_{\text{memb}} + E_{\text{ref}}. \tag{4.16}
\end{equation}
The above is formulated as a linear system and solved accordingly over 20 iterations.
Chapter 5

Experimental Results

5.1 Introduction

The previous chapters describe the theory and implementation of Intrinsic Normalization. Here we discuss what steps were taken to demonstrate the effectiveness of this method in improving results in action unit detection.

5.2 Goal & Strategy

Our goal is to demonstrate how using Intrinsic Normalization to refine facial input data to an algorithm for action unit recognition can improve classification accuracy. To do so, we start with a known algorithm, preferably one that produces state-of-the-art results. We implement this algorithm, and run it on input data as described in its original article, hopefully seeing results matching those in the original article. We then take the same exact input instances, normalize them with the Intrinsic Normalization algorithm presented in Chapter 3.1, and run the algorithm again, this time on the normalized data. If the hypothesis is correct, classification accuracy should improve from non-normalized to normalized data. It is important to emphasize that our goal is not to improve the algorithm itself, rather to compare the accuracy of the algorithm with and without normalized data as input.

5.3 Algorithm Implementation

In order to compare classification performance on unnormalized data against that of normalized data, we must integrate normalized data into our pipeline. The most convenient way to do so is to mimic the format of the original dataset so that all processing downstream can be identical.

We begin by taking the original Bosphorus data and triangulating it to create a mesh of the face. We can then process these meshes with the Intrinsic Normalization algorithm of Chapter 3. The output, however, is also a triangulated mesh, while the
processing downstream expects depths maps. Hence we recreate the depth map by matching a surface of the form $Z = F(X,Y)$. To fit the surface, we use the function `mxgridtrimesh.m` [Bri07]. The only degree of freedom is the orientation of the $x$-$y$ plane of the domain, though we simply leave it aligned with the global $x$-$y$ plane, as this corresponds to the depth map configuration where $z$-values represent distance to the acquisition device. Once this process is finished, the output can be used in the same manner as the original Bosphorus data. Examples of normalized faces are given in Figures 5.1 and 5.2.

We then have the feature/classifier ensemble. We chose Bayramoğlu et al.’s Center-Symmetric Local Binary Patterns (CS-3DLBP) and Ratio-Based Geometry (RBG) [BZP13]. The method was published very recently and as of this writing it is considered state-of-the-art. Moreover, it is simple to implement. As we are not developing a detection algorithm per se but rather a pre-processing step, we would like to show that newly normalized data run through the same algorithm causes more accurate classification. Thus we would like to mimic the setup of this method as closely as possible.

Thus our experimental setup consists of three main stages: data preparation, feature extraction and classification. First, the input Bosphorus depth maps are taken and downsampled from a width-by-height resolution of approximately $250 \times 280$ to $75 \times 100$ (the actual resolution of the input depth maps varies by several depth pixels). Median and Gaussian blur filters are then applied to suppress spikes and noise. Next, the depth map is treated as a point cloud and triangulated to create a mesh. Now that we have geometry, we calculate the normals at each face. With these surface normals we calculate at each depth pixel the CS-3DLBP feature as described by Equations (1.2), (1.3) and (1.4). This generates a $75 \times 100$-sized matrix of feature values.

This feature image is divided into a $5 \times 5$ grid, each cell of which contains $15 \times 20 = 300$ pixels. For each of these grid cells, we take a histogram of the feature values. Because this configuration of CS-3DLBP is 4-bit, each histogram contains sixteen bins. We concatenate all histograms to create a feature vector with 400 entries. Finally, in addition to CS-3DLBP, we also implement the Ratio-Based Geometry features. These values are concatenated on to our feature vector, increasing the size to 425 entries.

We run the previously described feature extraction on all input instances. The features are then run through the specified classifier. [BZP13] uses Random Forests for classification. We use this classifier with 2000 trees, 20 features per node, and a maximum tree depth of 5.

Most of the above code was written in MATLAB for convenient prototyping and analysis. Calculation of face normals, and both CS-3DLBP and RBG features were written in C++/Mex for speed. For the Random Forests classifier we used the Weka library [HFH+08]. Weka is vast library exposing many convenient machine learning tools, and is written in Java. We call the Weka functions through a MATLAB interface.
5.4 Cross Validation

With a multitude of data, we need an effective way of organizing it to ensure that we trust our classification scheme. In order to maximize the utility of the examples, machine learning analysis typically involves cross-validation. Cross validation is a method of dividing up the example data into training and test sets in different ways. This increases testability of the data by ensuring that all examples are at some point used in the test set, while also insuring lots of data on which to train the classifiers. In one scheme called one-vs-all, the training set includes all examples except one, and the test set includes the one left out. A more common scheme is known as $k$-fold, in which the data is split into $k$ partitions, and then throughout $k$ iterations, one partition is used as the test set after the classifier is trained with the data in the rest of the $k - 1$ partitions.

There are two other concerns that arise when setting up facial expression recognition testing. The first is that examples might not be completely independent from one another. Rather, examples from the same subject share certain geometric characteristics (e.g. physiognomy) with each other. For this reason, when building our cross validation partitions, we must ensure that in no cases do examples from the same subject appear both in the training set as well as in the test set. This means that a single fold may contain examples from multiple subjects, but no subject may appear in multiple folds. This practice prevents the classifier from ‘peeking’ into the test set while processing the training set.

Other concerns regard the data distribution itself. We would like as uniform and random a distribution of subjects across folds as possible. In addition, we would like to make sure that for any given partition, there are enough true positive examples of a given action unit to make calculating classification success interesting.

Marrying these constraints, we used subject-aware 10-fold cross-validation as specified in [BZP13]. To arrange the data into folds we used an available grouping that comes with the Bosphorus database. The file bospausDetCDE.mat in the bospcrvalsetups subdirectory supplies a grouping that obeys the above constrains for 25 different action units. For each action unit, a set of 10 lists of examples is provided. Each list represents a cross-validation partition, and includes at least some true positive examples containing the action units. Together, the lists for each action unit refer to most of the 2902 examples.

5.5 Results Comparison

For each of 25 action units we ran 10-fold subject-aware cross-validation, with and without Intrinsic Normalization as described above. Table 5.1 compares these two cases, in the middle and right-hand columns. The results of the original implementation, as published in [BZP13], have been provided in the left-hand column for reference. The results are shown graphically in Figures 5.3 and 5.4, and as AuC error ($100 - \text{AuC}$) in
Looking at the values, we see that for all but five AUs (AU16, AU22, AU23, AU27, AU43), classification with Intrinsic Normalization performed as well or better than without. Of these five, three performed only 0.1% worse, and one performed 0.2% worse. The only AU that performed significantly worse with normalization was AU22.

Of the action units that performed better with Intrinsic Normalization, some performed significantly better. Naturally, the closer the performance is to 100% the more difficult it is to improve. As the number of mistakes shrinks, there are simply fewer to correct, and so the percentage gains are smaller. Moreover, these last few are likely to be the more difficult examples. Still, even with results above the 90th percentile, improvements of 1.9% (AU20), 2.2% (AU17), 2.3% (AU15) and 3.0% (AU4) were achieved. The largest gain was clearly AU11, which improved by 8.8% from 75.7% to 84.5%.
Figure 5.1: Examples of non-normalized faces (left) and their corresponding normalized faces (right).
Figure 5.2: More examples of non-normalized faces (left) and their corresponding normalized faces (right).
Figure 5.3: Graph comparing AuC for normalized and non-normalized input data for first half of AU set.
Figure 5.4: Graph comparing AuC for normalized and non-normalized input data for second half of AU set.
<table>
<thead>
<tr>
<th>Action Unit</th>
<th>CS-3DLBP+RBG</th>
<th>Not normalized</th>
<th>Normalized</th>
<th>Positive example count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU1</td>
<td>96.5</td>
<td>94.4</td>
<td>96.1</td>
<td>215</td>
</tr>
<tr>
<td>AU2</td>
<td>98.3</td>
<td>96.2</td>
<td>98.0</td>
<td>147</td>
</tr>
<tr>
<td>AU4</td>
<td>94.9</td>
<td>95.3</td>
<td>98.3</td>
<td>176</td>
</tr>
<tr>
<td>AU5</td>
<td>-</td>
<td>88.3</td>
<td>91.0</td>
<td>133</td>
</tr>
<tr>
<td>AU6</td>
<td>-</td>
<td>94.4</td>
<td>95.1</td>
<td>100</td>
</tr>
<tr>
<td>AU7</td>
<td>-</td>
<td>94.4</td>
<td>95.7</td>
<td>477</td>
</tr>
<tr>
<td>AU9</td>
<td>98.4</td>
<td>98.3</td>
<td>99.1</td>
<td>107</td>
</tr>
<tr>
<td>AU10</td>
<td>98.1</td>
<td>95.1</td>
<td>95.8</td>
<td>72</td>
</tr>
<tr>
<td>AU11</td>
<td>-</td>
<td>75.7</td>
<td>84.5</td>
<td>14</td>
</tr>
<tr>
<td>AU12</td>
<td>98.1</td>
<td>98.9</td>
<td>98.9</td>
<td>295</td>
</tr>
<tr>
<td>AU14</td>
<td>97.5</td>
<td>94.0</td>
<td>94.3</td>
<td>76</td>
</tr>
<tr>
<td>AU15</td>
<td>96.6</td>
<td>95.8</td>
<td>98.1</td>
<td>68</td>
</tr>
<tr>
<td>AU16</td>
<td>95.6</td>
<td>94.2</td>
<td>94.1</td>
<td>102</td>
</tr>
<tr>
<td>AU17</td>
<td>98.6</td>
<td>95.1</td>
<td>97.3</td>
<td>134</td>
</tr>
<tr>
<td>AU18</td>
<td>98.9</td>
<td>98.1</td>
<td>98.4</td>
<td>185</td>
</tr>
<tr>
<td>AU20</td>
<td>95.8</td>
<td>97.1</td>
<td>99.0</td>
<td>51</td>
</tr>
<tr>
<td>AU22</td>
<td>98.9</td>
<td>98.0</td>
<td>97.3</td>
<td>41</td>
</tr>
<tr>
<td>AU23</td>
<td>98.3</td>
<td>93.3</td>
<td>93.2</td>
<td>63</td>
</tr>
<tr>
<td>AU24</td>
<td>95.2</td>
<td>96.2</td>
<td>97.0</td>
<td>153</td>
</tr>
<tr>
<td>AU25</td>
<td>96.8</td>
<td>99.2</td>
<td>99.3</td>
<td>568</td>
</tr>
<tr>
<td>AU26</td>
<td>96.4</td>
<td>89.9</td>
<td>91.6</td>
<td>156</td>
</tr>
<tr>
<td>AU27</td>
<td>99.7</td>
<td>99.1</td>
<td>98.9</td>
<td>133</td>
</tr>
<tr>
<td>AU28</td>
<td>99.5</td>
<td>99.6</td>
<td>99.8</td>
<td>43</td>
</tr>
<tr>
<td>AU34</td>
<td>99.4</td>
<td>99.6</td>
<td>99.6</td>
<td>74</td>
</tr>
<tr>
<td>AU43</td>
<td>99.7</td>
<td>97.7</td>
<td>97.6</td>
<td>137</td>
</tr>
</tbody>
</table>

Table 5.1: ROC AuC (%) Column 1: Published results from [BZP13]. Column 2: Our implementation of [BZP13] run on non-normalized input. Column 3: Our implementation of [BZP13] run on normalized input. Boldface indicates where normalization improves the prediction results. Underlined values indicate that using normalized input gives the best result across all three columns. Last column: number of example faces that express the current AU.
Figure 5.5: Graph comparing AuC error for normalized and non-normalized input data for selected AUs.
Figure 5.6: Graph comparing AuC error for normalized and non-normalized input data for the rest of the AU set.
Chapter 6

Conclusions and Discussion

6.1 Conclusion

We have demonstrated the effectiveness of our Intrinsic Normalization algorithm. With this method, data that spans many different subjects can be naturally made subject independent by transfer of only the facial expression to a common face, eliminating physiognomy as a source of variability. Using an algorithm considered state-of-the-art, we demonstrated the difference between using the original data and normalized data, specifically that in general intrinsically normalizing the data improves classification results across 25 facial action units.

6.2 Limitations

Like any method, Intrinsic Normalization comes with its limitations.

6.2.1 Topological Differences

In order for the deformation transfer algorithm to execute properly, all input meshes must be in correspondence with one another. Using the method we suggest, one can deform a single mesh onto all target meshes, creating new meshes identical in shape to the targets, and identical in triangulation to the source. However, in the case of significant topological differences such as differences in genus number, achieving this is impossible. A difference in genus number implies that there are a different number of boundaries associated with the surface. This could be attributed to an opening of the mouth, or a case where the target mesh has holes for eyes while the source does not. Another significant topological difference is the difference between shapes that are water-tight, and those that have a boundary. In these cases there is no way to find a proper correspondence, unless we were to confine our analysis to some subset of each surface with the same genus and boundary configuration. This especially impacts our research when attempting to involve examples with open mouths. Even if the
topological difference is handled by stretching a surface over it, the matching will cause serious distortions in the mesh around the mouth.

6.2.2 Processing Time

Another limitation brought by the Intrinsic Normalization process is the extra time needed to perform. Currently, the non-rigid registration algorithm takes about 70 seconds to complete, per face. In addition, deformation transfer takes roughly 6 seconds. Finally, eight iterations of Laplace Beltrami smoothing takes 6.2 seconds. However, these values correspond to a MATLAB implementation of these algorithms. We look to Weise et al.’s implementation [WLVP09], which involves non-rigid registration as well as deformation transfer, implemented on the GPU and C++. Their implementation runs at 10 seconds per frame unoptimized, and 67 ms per frame (i.e. 15 fps) optimized. Between this and an efficient GPU implementation of the Laplace-Beltrami smoothing, which involves a fairly trivial set of operations, we are confident that Intrinsic Normalization could be made to perform in real time, if not very close to this.

6.2.3 Manual Constraints

Finally, a vexing constraint comes from the fact that the non-rigid registration algorithm requires input of user-selected facial landmarks. These landmarks are supplied by the Bosphorus database, but would not be available in the wild.

There are three potential solutions. The first is to rely on recent research such as [PPTK13] that automatically discovers facial landmarks. Successful execution of algorithms like these could help circumvent the need for manual landmark picking. A second option is to track locations on the subject’s face. This may involve an initial picking process for a single instance of the subject’s face, but a good tracking algorithm would properly find these landmarks in the rest of the frames in a video. Finally, newer research demonstrates matching success without the use of sparse initial reference correspondence points, although at the expense of more processing time [SK14]. If automation is prioritized over interactivity, this could be a viable option.

6.3 Future Directions

Here we describe what we believe is the first use of intrinsic normalization in facial action unit detection. There are many improvements that can be made and several directions in which this technique could be taken. This section maps out potentially fruitful possibilities of future work.

6.3.1 Topological Adjustment

As we mentioned in section 6.2.1, topological differences between surfaces pose a problem for non-rigid registration, and therefore deformation transfer. Fortunately, we could
work around this by fixing our template mesh. By carving a seam between the lips, the mesh could be deformed to properly fit both open and closed mouths.

6.3.2 Ideal Target Manifold

In our implementation of Intrinsic Normalization we manually constructed a template face to supply the target physiognomy that the normalized faces would assume. However, this target physiognomy was somewhat arbitrary as the original formulation imposed no constraint on this parameter. This choice therefore amounts to somewhat randomly sampling the space of template faces. The question then arises whether there is a better or even theoretically ideal template face to use as a target physiognomy.

Future work could explore space of target physiognomies for the ideal template face. A viable first guess would be some kind of average of all example faces, or some facial model that represents minimum stress from all other examples examples.

6.3.3 Generalization

We demonstrated the effectiveness of Intrinsic Normalization on the space of facial expressions, however, there are no explicit constraints confining us to this domain. Since deformation transfer and all other supporting tools work on any set of meshes with similar topology, we can generalize the applicability to any kind of smooth mesh (for which deformation transfer could perform and behave well). In this vein, the normalization procedure could be used for generalized mesh deformation learning and be applied to problems such as gesture and motion type recognition.

Further, while the implementation is constrained to 2D meshes, we could imagine generalizing the algorithm to higher dimensional meshes (e.g. tet-meshes and higher) to improve the performance of algorithms that would detect higher-dimensional deformations. This would obviously involve more sophisticated versions each component in our process.

6.3.4 Localization

Though this work focuses on the applicability of Intrinsic Normalization, we can also improve the CS-3DLBP detection algorithm itself. [SS09] claims that localizing the classifier improves performance. Since we do have a classifier for each action unit we can specify a region of interest on the face for each. This translates to choosing a specific subset of the $5 \times 5$ grid. Or, since all faces are in correspondence after normalization, we can predefined regions over which to lay the grid. This would give us higher resolution in the areas of interest, while ignoring areas of the face not relevant to that action unit. On the other hand, if there does exist any useful correlation between presence of an action unit and lack of activation on another part of the face, localizing the region of interest would miss this.
Bibliography


59


Deformation is a concept used to describe changes in the shape of objects. In the context of computer vision, deformation is achieved using geometric tools. The implementation of this technique is focused on an algorithm that analyzes the deformation given on a mesh surface. The algorithm then transfers it to a new manifold, and an additional algorithm is used to match the non-rigid deformation mapping between the surfaces.

In addition, we use additional matching algorithms to match the deformation. Finally, we perform a filtering of the mesh surface to reduce the noise present in the data.

Using this method, we develop a semi-automatic system for the identification of unique features. Based on the method, we can derive results that are more robust and accurate. Specifically, we use a Center-Symmetric 3D method to perform a semi-automatic identification of unique features. We also use the Bosphorus Database for validation of the method. Local Binary Patterns are used in the evaluation of the method, and we compare the accuracy of our method with other existing methods.

For the final evaluation, we use the Local Binary Patterns for validation of the method. Finally, we compare the accuracy of our method with other existing methods.
תקציר

הבעות הפנים היא רכיב מ🧩intellectualität הב颋කושר בין בני אדם. הבעות הפנים הן חלק בלתי נפרד מהתקשורת האנושית והחברתית.

הבעות הפנים הן יכולות להעביר משמעות לאין מילים ואף לה动生成 הבתיםמשלבינות על שלושה פעילויות בין אדם אך לחם. לא כל ההבעות לשוינה כלולות בפנטזיה הפנימית באף שישה פעילויות, או SMS, או פספסות בדיבור ספורטיביደת צניחה חוטנת צ'אט או אוניברסיטום.

ה(mutual) פעילויות הפנימיות היא בין החשובים בשתי התפקידים של האנושיות ושל האנושיות. פורשים של דיבור יש בין הלשון הלשון בברוך, בצק, בו פספסות הפנימיות

למרות הדרישה לעובדים בממלכת שיווק, ולמרות השישיות של התפקידים האוניברסיטום, הנחיתות החיצונית של הפנטזיה, בבש כ־2, ומגשנת את המיון הווה היתודה שיזמן AU15 במגשפת של זהותveral 듣, AUs, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנה

למרות המ✏️המקס ובאומתים של תהליך קידום העדשה, למחרת שיש תהליך הפרדהTASK שContours של חכו fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של זהותveral יסתנות תחית fACSn, במגשפת של הזו
המחקר בוצע בחנויות של פרופסור רון קימל, בפקולטה למדעי המחשב.

תודות

ראשת בורגי להדודות לכל הנכים של, וו, קימל, עבורה הנחיה הפוסרה במחלקה המחקר של, הרבה.

תודות למתן שלל, אחרים, בורח, סטיה, וברדינוג, והאר, מוענקת הה"ה GIP" עבורה התמכה התכנית.

הتهمירויות והמדברים, הלומדים, בונה משכון עברה במרחיב הלברתי.

רב תודות להודעה על랩ה השותף אחרים, במעבדה לארובני, ג'ימי, יאני, יאני, משתח היר, משתח

케ほן, משתח מיאורג, ולא האחים, שיעור כל להוריהם בביית הפ בואר ישראלי.

אחרים, אחרון, חיבר, זמנים, אשר אסירי תודות Inserts ולא במעבדה כל התמוך של בלעל.

לא חיותו כל עשה כל אט על האחבה. התמכה, העידון והסבלנות שלום.

הכרת תודות מסורה לשבינה על מימוק מחקר זה.
אנליזה של ההעוצת פניס
בעזרת נירמול של משטחים

הייבור על מחקר

לשם مليולי חלקי של הדרישות לקליבת התואר
מניסו למדעי בציבור המناقش

אריק רוברט ויdni

הנה בלנסק הסכנין - מוכן הסכנונים לישראל
ארד התשע"ה בחיפה מברואר 2015
אנליזה של הבעות פנים
בעזרת נירמול של משטחים

אריק רוברט יודין