December 2018

3D Shape Descriptor-Based Facial Landmark Detection: A Machine Learning Approach

Reihaneh Rostami
University of Wisconsin-Milwaukee

3D shape descriptor-based facial landmark detection: A machine learning approach

by

Reihaneh Rostami

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin–Milwaukee

December 2018

Abstract

3D shape descriptor-based facial landmark detection: A machine learning approach

by

Reihaneh Rostami

The University of Wisconsin–Milwaukee, 2018
Under the Supervision of Professor Zeyun Yu

Facial landmark detection on 3D human faces has had numerous applications in the literature such as establishing point-to-point correspondence between 3D face models which is itself a key step for a wide range of applications like 3D face detection and authentication, matching, reconstruction, and retrieval, to name a few.

Two groups of approaches, namely knowledge-driven and data-driven approaches, have been employed for facial landmarking in the literature. Knowledge-driven techniques are the traditional approaches that have been widely used to locate landmarks on human faces. In these approaches, a user with sufficient knowledge and experience usually defines features to be extracted as the landmarks. Data-driven techniques, on the other hand, take advantage of machine learning algorithms to detect prominent features on 3D face models. Besides the key advantages, each category of these techniques has limitations that prevent it from generating the most reliable results.

In this work we propose to combine the strengths of the two approaches to detect facial landmarks in a more efficient and precise way. The suggested approach consists of two phases. First, some salient features of the faces are extracted using expert systems. Afterwards, these points are used as the initial control points in the well-known Thin Plate Spline (TPS) technique to deform the input face towards a reference face model. Second, by exploring and utilizing multiple machine learning algorithms another group of landmarks are extracted.

The data-driven landmark detection step is performed in a supervised manner providing an information-rich set of training data in which a set of local descriptors are computed and used to train the algorithm. We then, use the detected landmarks for establishing point-to-point correspondence between the 3D human faces mainly using an improved version of Iterative Closest Point (ICP) algorithms. Furthermore, we propose to use the detected landmarks for 3D face matching applications.

To my Husband and Daughters

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my mentor and research advisor, Prof. Zeyun Yu. You have always been supporting me during my PhD studies. Your knowledge and constructive comments have always helped me to find a solutions for the problems I faced with. Thank you for giving me the opportunity of working under your supervision.

Ahmad and Fereshteh, I thank you for your friendship and the useful discussions we had during my graduate studies.

I would like to express my sincere gratitude to my thesis committee members: Professor Munson, Professor Suzuki, Professor Zhang, and Professor D'Souza for their insightful comments.

My sincere and deep gratitude to my dear parents who have always supported me during my life. I feel so honored and blessed to have you as my parents. You are my first teachers. You taught me perseverance, honesty, and strength. I cannot thank you enough for all your help and support during these years. I thank you Ramin, Behrouz, Elaheh, and Somayeh greatly for your constant support over the years and the positive energy I always receive from you.

Nothing could replace strong and deep support of my husband, Farid. I was never scared of any hardships during my studies because I had your wonderful and unending support. If were not your help and cooperation I was not able to overcome the difficulties. Thank you for everything.

Chapter 1

Overview

1.1 Problem Statement

3D facial landmark[1] detection (FLD) is the process of detecting keypoints on 3D face models. The keypoints correspond to salient features of a human face such as the tip of nose, mouth corners, inner and outer eye corners etc. [9, 10]. In many cases the located landmarks are used to establish the correspondence between the 3D face models, a task which plays a fundamental role in numerous applications such as 3D face reconstruction [11], face registration [12], 3D face morphing and animation [13], and various medical applications such as diagnosis of craniofacial disorders [14, 15], to name a few out of many.

3D facial landmarking has been an active area of research in the computer graphics and computer vision communities since more than a decade ago [1]. Large variations in different face structures make the exact landmark detection a difficult face analysis task. An inaccurate landmark could affect the process of finding other landmarks which are detected afterwards [16]. Furthermore, incorrect landmarks result in a weak correspondence which could be completely useless based on the application that the correspondence is used for. For instance, in video tracking applications, a sparse set of corresponding landmarks which are not sufficiently accurate would work. However, for applications such as 3D face reconstruction or morphing that require a dense set of corresponding points, small deviations from the ground-truth can generate improper results.

In this project, we suggest a FLD framework by using a combination of the knowledge-driven and data-driven approaches and locates the following 16 keypoints on 3D face models:

[1] We use landmark and keypoint terms interchangeably in this thesis.

1. Top of the forehead
2. Nasion (nose bridge)
3. Pronasal (tip of the nose)
4. Subnasal
5. Right cheilion (mouth right corner)
6. Left cheilion (mouth left corner)
7. Pogonion (chin)
8. Right cervical (neck)
9. Left cervical (neck)
10. Center cervical (neck)
11. Right exocanthion (right outer eye corner)
12. Right endocanthion (right inner eye corner)
13. Left endocanthion (left outer eye corner)
14. Left exocanthion (left inner eye corner)
15. Right nostril
16. Left nostril

For many 3D face analysis tasks (e. g., medical applications, 3D face reconstruction, morphing, face synthesis, animation, etc.) the availability of a dataset of 3D face models with dense correspondences is essential because these applications mostly require some statistics to be computed from a group of 3D faces [17]. For instance, Blanz and Vetter proposed a strong 3D face reconstruction and morphable model in which a 3D structure of a single

2D face image of an individual was reconstructed using a linear combination of the 3D face models [13]. Such approaches highly rely on the existence of 3D face databases in full and dense correspondence. However, these databases are rare in the literature. Building the point-to-point correspondences is quit challenging because of the variations in the scanned face model structures, acquisition noises and spikes, isolated points and all of such difficulties that arise while working on the real data. Therefore, for an application of our FLD system, we propose to construct a database of 3D face models with point-to-point correspondences using an available database of Chinese 3D scanned faces called BJUT.

Section 1.2 of this chapter reviews the recent 3D FLD systems proposed in the literature and explains how our work roots from the literature and proposes a new perspective in finding the facial landmarks on 3D data.

The remainder of this thesis is organized as follows. Chapter 2 summarizes the machine learning algorithms which are used in the literature to construct data-driven 3D shape descriptors. In addition to the survey of various descriptors, multiple comprehensive taxonomies are suggested that provide hierarchies for the data-driven 3D descriptors from application, learning algorithms, and input data type perspectives. We survey the literature for the shape descriptors and the learning algorithms because these topics make two main parts of our FLD system. While reviewing the literature, we learned that a survey on data-driven 3D shape descriptors was missing; therefore, we broadened our research and studied the field in greater depth. In Chapters 3 and 4 we elaborate technical aspects of the knowledge-driven and data-driven modules of our system. Finally, in Chapter 5 we propose an application of our FLD system in establishing point-to-point correspondence.

1.2 Related works

A first application of using machine learning algorithms for 3D facial landmarking goes back to 2013 when Creusot *et al.* proposed a data-driven system to detect 14 landmarks automatically [1]. They reviewed the FLD systems throughly and divided the previous works into two major groups of expert system-based and machine learning-based approaches. The former group, referred to as the knowledge-driven techniques, provided the apriori knowledge to the system by defining the ad-hoc definitions for the features. In these approaches, usually a user with sufficient knowledge and experience in the area defines the features to be extracted as the landmarks. The machine learning-based approaches, on the other hand, take advantage of machine learning algorithms to detect significant features on the 3D face models. In these approaches, referred to as the data-driven techniques, the system is provided with training data (rather than being provided with the features encoded into the knowledge) so that the system learns the feature models and finds the landmarks. The authors introduced a data-driven FLD system applied on the 3D data, for the first time. In their system they utilized the local shape descriptors which mostly consisted of the curvature related features such as Gaussian curvature, mean curvature, shapeIndex, etc. They trained their system by a set of N training faces annotated with L landmarks per mesh. The local descriptors of length $D = 10$ were constructed for all vertices of the training meshes. Then, the distribution of descriptors were estimated for the L landmarks considering the descriptors of landmarks over all N training samples. These distributions, were used to normalize the computed shape descriptors and converting them into $L \times D$ scores that are in [0,1] per vertex and per shape. Then, a set of training samples were selected from the input data as the landmark and non-landmark classes. The vertices close to a specific landmark and those far from it, were assigned to the landmark and non-landmark classes, respectively. Figure 1.1 illustrates how the training data were selected from the input 3D meshes.

The training dataset which included the scores computed for the samples along with the class labels were the input of a classifier that learned to discriminate between the landmark and non-landmark classes. In the testing phase, an unseen face with the computed scores was presented to the classifier to detect the landmarks. One of the best results of the landmark detection on sample faces is demonstrated in Figure 1.2.

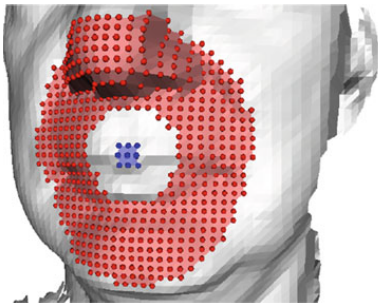


Figure 1.1: Training data generation process used by creusot *et al.* [1]. The blue and red points were annotated as the landmark and non-landmark class instances, respectively. The figure is taken from [1].

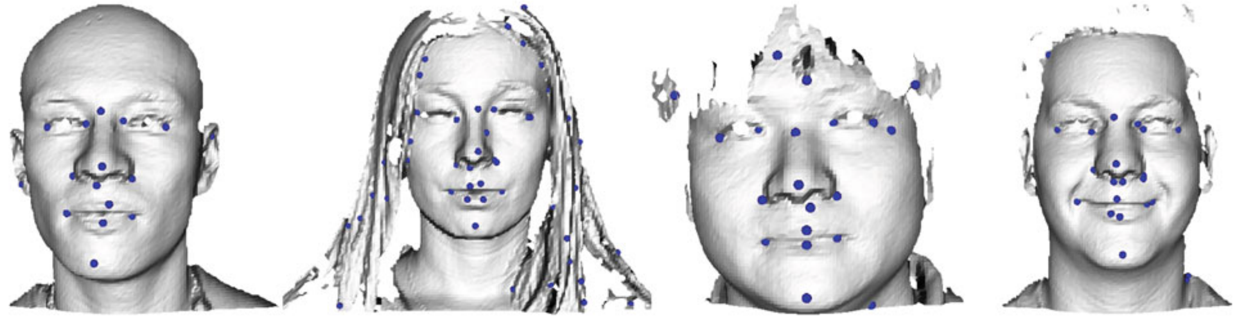


Figure 1.2: A sample result generated by the approach suggested by creusot *et al.* [1]. The figure is taken from [1].

Perakis *et al.* [9] introduced a 3D facial landmark detection system using the shape descriptors to detect eight keypoints. The main contribution of this work was its efficiency in faces with expression variations and large yaw rotations. They proposed to compute local descriptors per vertex of the input which were made up of two features; namely, shapeIndex and spinImages. At first, a set of candidate landmarks were selected using the shapeIndex and then these landmarks were reduced to a smaller set by considering the spinImages similarities. Furthermore, a set of manually selected landmarks on the training dataset were used to form a facial model by averaging. Therefore, when the landmark candidates were provided using the shape descriptors, they were used to find the best set of landmarks that minimizes the total distance between all of the landmarks and those of the average facial model.

In another application, Sukno *et al.* suggested a statistical approach that was able to capture the nonrigid transformation and handled the missing landmarks [16]. Their approach which detected 14 landmarks, used the shape regression with incomplete local features to detect the facial landmarks on 3D face models. The authors approached the problem from a global view by using a shape model that also extracts the prior knowledge out of the data. Their proposed technique consisted of three steps. At first, they computed a similarity score for each mesh vertex based on its local geometric descriptors in a training task. Then, using a shape model regression they estimated the positions of missing landmarks, and finally the detected landmarks from the last two steps were combined to form the final set of landmarks.

In contrary with the previous work which focused only on the geometric information, Fan *et al.* proposed a new framework that employed both geometry and texture information to detect facial landmarks [10]. They combined the geometry data from the 3D models with the texture information taken from the 2D images corresponding to the 3D data. They used the geometry images to obtain the 2D representation of the 3D face models. Eight landmarks were detected on the geometry images and then using one-to-one correspondences between the 2D and 3D data, the landmarks were transformed back to the 3D space.

With this introduction to the recent works on the 3D landmark detection, we propose our hybrid shape descriptor-based system to detect 16 landmarks on 3D face models which is mostly similar to the work introduced by creusot *et al.* [1]. Our system consists of a knowledge-driven module followed by a data-driven unit. The first module detects 10 landmarks on the input face model. These landmarks are already defined by an expert as the salient features of the face. The input to this module would be 3D face models that 10 landmarks will be detected on them. This step of the system uses both geometric and photometric (only for locating the mouth) information to detect the landmarks. Then, we computed local shape descriptors for every vertex of the meshes and applied a clustering algorithm to extract a cluster of interest for each of the remaining 6 landmarks. After that, each cluster is fed into a relevant classifier to detect the landmark of interest in the data-driven module.

The main advantage of our work over the existing works in the literature is combining the strengths of knowledge-driven and data-driven approaches in detecting facial landmarks. In fact, we use the knowledge-driven techniques to detect more salient features of the human face such as the nose tip, the top of forehead, chin, etc. For such landmarks, we do not need a complex set of rules and at the same time we do not generate the training data manually or train any classifiers. For the second set of landmarks that has much more variations over different faces and that we are not able to detect using simple ad-hoc feature definitions, we generate the training data and use the machine learning-based approaches to detect them. Another improvement is that we reduce the complexity of computation and guide our data-driven technique by limiting our search space into a local cluster for each landmark. To the best of our knowledge the existing approaches in the literature use the global search that can increase the chance of false positives (Figure 1.2).

We use BJUT dataset, a Chinese 3D face model database, for our work. We use our technique to establish a set of dense point-to-point correspondences between various 3D face models which could be finally used to form a database of faces in full correspondence. Availability of such a database of 3D face models (that are rare currently) is critical for 3D face analysis applications such as 3D face reconstruction and 3D morphable models. We applied our method on the BJUT dataset to build such 3D face benchmark and make it publicly available to the computer graphics community for various 3D face analysis research purposes. More detail regarding our approach is presented in Chapters 3 and 4.

Chapter 2

3D shape descriptors

2.1 Introduction

3D models have become an ubiquitous data type in various fields such as material and mechanical engineering [18], genetics [19], molecular biology [20], dentistry [21, 22], etc. Additionally, modern scanning devices such as Microsoft kinect [23] and laser scanners have made it possible to generate 3D models more efficiently and accurately. Many available domain-specific databases also provide a large number of 3D shapes publicly available (Section 2.3). A variety of applications like point matching, point-to-point correspondence, shape segmentation and labeling, object recognition, shape reconstruction, and shape retrieval use 3D models as input. Even though there exist successful techniques such as Iterative Closest Point (ICP) [24] and Thin Plate Spline (TPS) [25, 26] which use the geometric characteristics of the models (given as point clouds or meshes) for shape matching, many other analyses need their inputs to be represented in a more concise, yet informative, format. These techniques work with the informative and discriminating features, so-called shape descriptors, extracted from the 3D models rather than utilizing the models themselves.

A shape descriptor or signature is a d-dimensional vector that represents the shape in a compact space. 3D shape descriptors are obtained by defining features on 3D models, calculating, and concatenating them to form a compact vector representing the 3D shape [27]. Based on whether the features represent global or local attributes of the shape, their relative shape descriptors could be referred to as global or local descriptors, respectively. Global descriptors could be computed directly from the inputs or they could be preceded by calculating local descriptors (e. g. point signatures or descriptors which are computed for faces or different segments of a shape).

Many shape analysis tasks would be inconvenient if not impossible to conduct, without making use of the shape descriptors. As an example, for shape retrieval that is the process of querying a 3D model against a database of 3D models and finding the closest match, it is necessary to convert the shapes into feature descriptors to be able to store the large volume of 3D data models and to quickly query and find the closest match. Another example pertains to point matching applications where it is desired to describe point clouds by using their local descriptors (i. e. point signatures) and then the point similarities are determined with the distance between their descriptors. This is in contrast with the ICP method that computes the Euclidean distance between points in the 3D space to find the closest match. The significance of efficient shape descriptors have motivated groups of researchers to conduct extensive studies on 3D shape descriptors and to propose different taxonomies for them.

In 2004 Tangelder and Veltcamp summarized different methods of content based 3D shape retrieval [28]. While the main goal of this work was to review shape retrieval methods, 3D shape descriptors as a fundamental step in shape retrieval were also thoroughly discussed. The first survey on 3D shape descriptors was published in 2007 by Zhang et al. [29] following a taxonomy on 3D shape descriptors in a survey on feature-based 3D object similarity search [30]. These articles performed an extensive study and provided appropriate categorizations for the shape descriptors. However, these reviews were conducted about ten years ago, and did not include many recent state-of-the-art techniques. More recently, Heider et al. [31] and Tang and Godil [32] surveyed 3D shape descriptors but they limited their work to local methods only. In addition, the latter specifically evaluated the performance of 3D shape descriptors applied to the Bag-of-Features (BoF) method. Kazmi et al. [33] reviewed both 2D and 3D descriptors in 2013; however, their work did not include some of the recent effective spectral methods (e. g., Heat Kernel Signatures and Wave Kernel Signatures).

The approaches which were reviewed in the aforementioned surveys were mostly traditional techniques of constructing 3D shape descriptors. These approaches were mainly knowledge-driven systems in which researchers with sufficient knowledge and experience in the area, hand-crafted features to be extracted from shapes and be used as shape descriptors. On the other hand, the successful applications of machine learning have been increasingly attracting attentions in many research areas. Although machine learning algorithms have been used in developing 3D shape descriptors since 1987 [34], they have become more popular recently with new development for dimensionality reduction and especially the significantly accelerated learning of a deep structure which emerged in 2006 [35, 36].

Why data-driven approaches? In general, data-driven approaches are machine learning based techniques that analyze large volumes of data and extract meaningful knowledge out of them. The major advantage of using data-driven approaches rather than knowledge-driven techniques in constructing 3D shape descriptors is that they learn features and descriptors from the sample data (training data) automatically with no or little amount of knowledge provided by experts [37]. The ad-hoc feature definition in the knowledge-driven systems requires a well-informed and skillful expert who is not always available. Therefore, this could make potential barriers in constructing informative and reliable descriptors that captures the discriminative and salient features of the objects. In fact, providing a data-driven system with examples of a desired property and training it to learn the features automatically would be much easier and more intelligent than coming up with a formal definition of the property which most of the time is not easy to formulate and is subject-dependent [38]. Furthermore, due to being task-specific, the traditional approaches are less generic than the data-driven techniques and it is hard to apply them to different tasks. In fact, data-driven techniques build more generalizable shape descriptors because they learn constructing the descriptors through large training datasets and not pre-defined rules.

Contribution To our best understanding, the present survey is the first comprehensive study on data-driven 3D shape descriptors. The large body of work on data-driven 3D shape descriptors beside the lack of such an extensive study on state-of-the-art techniques, lead us to review the recent advances in this area. Therefore, in this study we particularly focus on the systems built based on machine learning algorithms to construct 3D shape descriptors. We provide an extensive taxonomy for the surveyed methods and elaborate on the comparisons of these approaches in various criteria such as input data types, applications, machine learning algorithms based on which the descriptors were computed, and the feature extraction techniques. We also categorize the proposed descriptors in the literature based on their input data type and the applications. In general, it is our hope to provide a complete and deep categorization and a broad insight into what has been accomplished in the area of data-driven 3D shape descriptors. We hope to help future researchers making an informed decision on algorithms for their available data type and specific applications they aim to work on.

Organization The remainder of this chapter is organized as follows. Section 2.2 provides the background for this survey. The general framework of data-driven systems which construct 3D shape descriptors besides the components of this framework are reviewed, particularly. Section 2.3 summarizes a list of many datasets used to construct shape descriptors, recently. Then, Section 2.4 reviews the data-driven 3D shape descriptors proposed in the literature both thematically and chronologically, along with proposing an algorithm-oriented taxonomy for them. The reviewed works are discussed in Section 2.5, alternative taxonomies for them are proposed, and advantages and limitations of each category are discussed in detail. Finally, Section 2.6 suggests several directions for future research.

2.2 Background

In this section we present a general pipeline of data-driven systems that build 3D shape descriptors. To construct such a descriptor, a set of training data needs to be presented to a machine learning algorithm. The input could be either the original data itself or a set of extracted features. The training set is presented to the system iteratively until some criteria is met, meaning the algorithm has converged to a potentially meaningful descriptive model. After the training phase, the system is ready to generate shape descriptors for unseen data samples. This pipeline which is illustrated in Figure 2.1 includes three major blocks, similar to the pipelines which were suggested in the literature for various data-driven systems (e. g., data-driven shape processing [39], model classification [40], and image-based object classification [41] as a few examples).

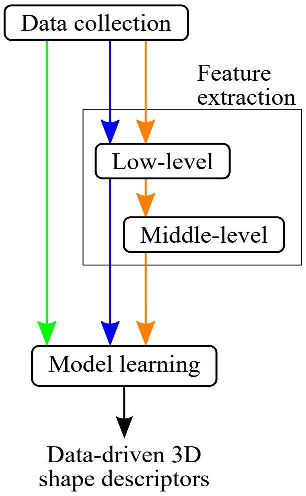


Figure 2.1: Three common components of the data-driven systems pipeline to generate 3D shape descriptors. The system could be feature-based or not. In the feature-based techniques the low-level features are extracted out of the input data and then could either be fed to the learning algorithms (blue arrows) or be processed one more step to generate middle-level features before being provided to the learning algorithms (orange arrows). The original data is fed to the learning algorithm directly in a non-feature-based technique (green arrow).

Different techniques and algorithms could be used to build components of the pipeline demonstrated in Figure 2.1. Any categorization for such techniques or algorithms will divide the obtained shape descriptors into categories, accordingly. For example, a categorization of different data types collected for the learning system (e. g., 2D images, 3D point clouds and meshes, voxel grid data, etc.) could divide the descriptors into relevant categories (e. g., the 3D descriptors which are built based on the 2D images, etc.). A categorization on the various learning algorithms used to generate the descriptors would also divide the obtained descriptors into the corresponding groups (e. g., 3D descriptors which are constructed employing the autoencoders, etc.). Therefore, we suggest three main categorizations for the data-driven 3D shape descriptors based on this pipeline:

- Algorithm-oriented
- Data-oriented
- Application-oriented

In Section 2.4 we propose a comprehensive and deep taxonomy that divides the reviewed descriptors into a fine hierarchical categorization. The taxonomy is proposed based on an algorithm-oriented perspective (i. e., it considers the learning architectures and algorithms used in the model learning component of the pipeline). We also propose two additional taxonomies based on input data types as well as applications, in Section 2.5. Three components of the pipeline, data collection, feature extraction, and model learning, are elaborated in subsections 2.2.1, 2.2.2, and 2.2.3, respectively.

2.2.1 Data collection

Input data makes an important component of the pipeline. As it was explained in the introduction, data-driven techniques highly rely on the provided input data as the training set. It does not matter how well-designed the architecture of the learning algorithm is or how precisely the feature extraction is defined to detect the salient features, no descriptor will be constructed if no appropriate dataset is available. The more information-rich and larger the training datasets are, the more accurate and precise the computed descriptors would be. Machine learning algorithms require large and versatile datasets so that all aspects of the objects with the fine details could be learned. There exist a lot of different datasets in the literature which most of them are publicly available on the web. A complete list of many datasets which are used in the literature to construct 3D shape descriptors is discussed in Section 2.3.

2.2.2 Feature Extraction

Groups of systems that exploit machine learning algorithms to build shape descriptors do not use the input data as they are; instead, they extract low-level features out of the data and feed them to the learning system for constructing the shape descriptors. For example, geometric features such as curvature (mean curvature, Gaussian curvature, etc.), average geodesic distance, and spin image have been used in the literature [42, 43, 38]. In addition, there have been recent spectral features such as Heat Kernel Signatures and Wave Kernel Signatures which were used broadly for local feature extraction (e. g. [43, 44, 37, 45, 46, 8], etc.).

On the other hand, a group of descriptors in the literature employed a technique called Bag-of-Features (BoF) to extract the middle-level features out of the low-level ones and these features were fed into the learning algorithms as the training data ([45, 47]). Experiments have shown providing the middle-level features to the deep learning algorithms leads to more successful descriptors compared with using the original 3D data or the low-level features as input [45]. The reason could be because the 3D data are not rich in features (as opposed to the 2D image data which could be directly fed into the machine learning algorithms) [45, 48]. On the other hand, providing the low-level features to the learner requires much higher volume of training data which is not the case for the middle-level features [45].

Due to the popularity of the spectral techniques including HKS and WKS for low-level feature extraction, we summarize them in more detail in this section. In addition, because of the importance of the BoF technique in extracting the middle-level features we explain its functionality at the end of this section, as well.

Spectral Shape Descriptors Generally speaking, spectral descriptors are constructed by computing the eigenvalues and eigenfunctions of mesh operators [49] such as the Laplace-Beltrami operator (LBO) [50]. Typically, the LBO which is defined as the divergence of gradient of the function f defined on the Riemannian manifold M, is approximated on a triangulated surface mesh using the cotangent scheme [51]. LBO is a widely used mesh operator and plays a key role in various spectral shape descriptors (e. g., [52, 53, 54, 5]).

Heat Kernel Signature (HKS) which takes a spectral approach for constructing shape descriptors was proposed in 2009 by Sun et al. [53]. HKS assigns a vector $\overrightarrow{P}(x) = (h_{t_1}(x,x), h_{t_2}(x,x), \cdots, h_{t_n}($ to each point x of the 3D mesh in which $t_1, \cdots, t_n$ represent a finite set of times and $h_t(x,x)$ is the amount of heat retained at point x after time t. These vectors which are called the local shape descriptors or heat kernel signatures represent the points of a 3D surface mesh. The heat diffusion on the manifolds can be described by the heat equation in which the LBO is a leading element [53]. With notable properties such as being intrinsic, isometric, and stable against the perturbations [53], HKS is a prominent spectral descriptor with many applications (e. g. [55, 56, 57] to name a few). Two famous extensions of the HKS namely Scale-Invariant HKS (SI-HKS) [58] and Volumetric HKS (V-HKS) [59] were also introduced in the literature.

Wave Kernel Signature (WKS) another spectral shape descriptor, was suggested in 2011 by Aubry et al. [54] being adopted from "Quantum Mechanics" to capture multi-scale details of 3D shapes. They used the Schroedinger equation to represent the movement of quantum mechanical particles on surfaces. Like HKS, each point of 3D meshes was associated with a vector $\overrightarrow{P}(x) = (p_{e_1}(x), p_{e_2}(x), \cdots, p_{e_n}(x))^T$ [5] where $p_e(x)$ is the probability of measuring a quantum particle at point x with an initial energy distribution function. The solution to the Schroedinger equation which is related to eigen-decomposition of the LBO, is used to compute $\overrightarrow{P}(x)$. Since WKS is parametrized using frequency rather than time it can control accessing high frequencies as well [54] by using a set of band-pass filters [5] as opposed to HKS which is made up of a set of low-pass filters. Therefore, WKS results in precise descriptors to capture local information of shapes [60]. WKS is a well-known descriptor and holds intrinsic, informative, and stability properties.

HKS and WKS have made the building blocks of many data-driven approaches in the literature; in addition, many researchers were inspired by these descriptors to construct data-driven shape descriptors in the spectral domain (e. g. [44, 48, 5]). Moreover, both descriptors have been employed as standards by researchers to evaluate the performance of their descriptors (e. g. [37, 4], etc.). We refer the readers to the work conducted by Litman an Bronstein for a profound comparison between the HKS and WKS descriptors [5]. Besides, additional information on spectral mesh processing could be found in a recent survey by Zhang et al. [49]. Also Masci et al. provided a nice review on various spectral shape descriptors [44].

Bag-of-Features Bag-of-Features (BoF) is one of the frequently used techniques in constructing global data-driven 3D shape descriptors or extracting middle-level features out of the provided low-level features. In this technique, at first the feature points are extracted out of the shape. Then, a clustering algorithm is used to segment the feature space. Finally, a codebook which is typically referred to as visual dictionary, is constructed representing each cluster seed (visual words) computed in the previous step. This codebook is used to generate the descriptors for any unseen input data. The feature points are computed for the new input, each feature point is assigned to the closest cluster seed and then the input is coded in a vector that represents the histogram of occurrences of the visual words [2]. While mostly the K-means algorithm is used in the BoF framework, some alternative algorithms could be used instead such as the sparse coding approach [61]. Figure 2.2 summarizes this technique applied on 3D data models.

2.2.3 Learning algorithms

This section reviews learning algorithms that were used in the literature to build 3D shape descriptors. These algorithms are trained to learn the desired shape descriptors. The learning process could be carried out in different types in terms of the amount of supervision provided to the algorithms. The learning type divides the algorithms into three categories namely supervised, unsupervised, and semi-supervised groups.

Supervised learning algorithms In this type of algorithms, the supervision is provided to the algorithms mostly through the class labels. For example, while training a system for

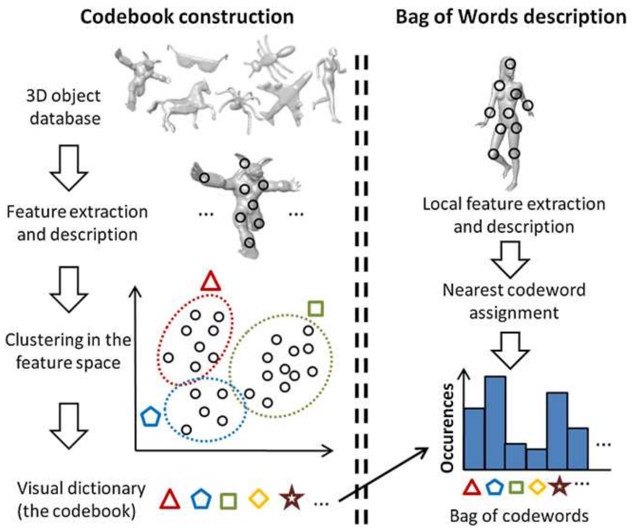

Codebook construction
Bag of Words description
3D object database
Feature extraction and description
...
...
Clustering in the feature space
Visual dictionary (the codebook)
...
Local feature extraction and description
Nearest codeword assignment
Occurences
...
Bag of codewords

Figure 2.2: A schema of the Bag-of-Feature approach [2]. The figure is taken from [2].

classification, the class label of each sample data is provided along with it. Supervision can also be provided in the form of a set of known correspondences such as the technique used in the distance metric learning in which a set of known correspondences are given to the system as the training data [62].

Unsupervised learning algorithms In this technique, no supervision is provided to the algorithms. All training data that the algorithm receives is the data itself. The algorithm will learn the underlying model of the data by observing a large volume of data from the domain of interest. Clustering is a famous example of unsupervised learning. The inadequacy of the labeled data and availability of loads of unlabeled data made the unsupervised learning a proper approach [63] for training data-driven systems.

Semi-supervised learning algorithms In this type of learning, a small amount of supervision is provided to the system beside the large amount of unlabeled training data [64]. For example, in a clustering algorithm we could provide the class labels for a small group of instances while for all the others the class labels remain unknown. This technique is useful for cases that only portion of our training data possesses the target class information.

Learning type is an important feature of a learning algorithm; however, it cannot provide a unique categorization for them. It is mainly because there are learning algorithms that, for example, could be trained in both supervised and unsupervised manners (we refer the reader to Section 2.4.2 for examples of such algorithms). Therefore, we consider another feature for an alternative categorization of the learning algorithms. The architecture of the algorithms could classify them into two broad deep and shallow groups [65]. We summarize the various learning algorithms used in the literature to construct the 3D shape descriptors in the following two subsections, accordingly.

Shallow learners

Shallow learners have very few levels of computation and require a huge set of training samples since they use a large number of computational units (e. g., functions that are computed in units of a learning architecture such as perceptron blocks in ANNs or kernel units in SVM, etc.) to capture and represent the underlying model of the data [66]. Linear models, decision trees [67], K-means clustering [68], support vector machines (SVM) [69], and one-hidden-layer neural networks [70] are famous examples of shallow learners.

Many data-driven shape descriptors in the literature were learned using the shallow structures. For example, descriptors that were computed using the BoF technique, mostly used unsupervised K-means clustering algorithm to transform local descriptors into global shape descriptors. These applications are listed in Section 2.4.1 of this chapter. Besides, some proposed descriptors in the literature formulated the descriptor learning as an optimization problem. They mostly used distance metric learning [62] to solve the problem. Metric learning which is a supervised learning algorithm provides the supervision in the form of known correspondences between similar candidates rather than providing class labels for the training examples. Other learning algorithm used in the literature [71] is the standard L-BFGS [72] which is the limited-memory version of BFGS algorithm for function minimization in large scale using the inverse Hessian matrix. The L-BFGS reduces the memory usage of the minimization algorithm.

Deep learners

In contrast to those of the shallow structures, the circuits of deep architectures which are made up of many layers are significantly simpler but they need to solve highly complex optimization problems [65]. Deep learning algorithms learn several levels of feature representations by making use of the supervised, semi-supervised, or unsupervised learning. These algorithms are typically implemented as a kind of Artificial Neural Networks (ANNs) which are consisting of many nodes/neurons that each of them is a simple computational block. The effectiveness of deep learning comes from avoiding predefinition of the features in order to capture hierarchical representations, automatically. In fact, the features are not engineered or hand-crafted by human programmers, instead they are automatically extracted from the input data by the deep structure [73, 74]. The key property of a deep network is being able to learn the high-level (abstract) features to deal with different types of invariances [65]. When an observation X is presented to a deep network, the training strategy for the lower layers is to extract the low-level (detailed) features while the upper levels learn the combinations of these features resulting in the abstract features defining X [35]. Even though the history of deep learning goes back to 1940s known as cybernetics at that time [75] it has become more popular since 2006 [35, 36].

The first deep structure that we introduce, is the Fully Connected Network (FCN). These networks are Multi-Layer Perceptrons (MLP)s with a large number of hidden layers [76]. As with MLPs, error back-propagation (BP) is the technique which is used to train FCNs and calculate the weights. However, this technique has some shortcomings such as back-propagated and summed error values in the network maybe faded or enlarged drastically as the training is continued [77]. In fact, this type of gradient based optimization that is initialized with randomly chosen weights may get trapped in a local solution [66]. The FCN formulates a highly non-convex problem; therefore, it needs a precise parameter initialization. Using an unsupervised pre-training technique that was first proposed in [35] could provide a better initialization for the parameters in terms of avoiding more local extrema [76]. We refer the reader to [77] for other methods that avoid the problems of BP.

Convolutional Neural Networks (CNNs), a category of ANNs, are based on convolving filters with multiple arrays input data [73] to extract features [78]. In fact, CNNs consist of layers of convolution, sub-sampling, and optional fully connected layers [73] which enables CNNs to learn the hierarchical features [37]. CNNs are much simpler than FCNs as they suppress the unnecessary complexity of FCNs. In FCNs, each node of the higher level gets input from all of the nodes in the lower level, generating $N \times M$ connections in a simple two-layer network with N and M neurons in the first and second layers, respectively. In contrast, CNN simplifies this structure such that the neurons in the higher layer only get input from local nodes of the lower layer. This structure drastically reduces the number of connections. As the connections represent the weights, there will be much lower number of weights to be calculated resulting in a much simpler optimization function [73]. In addition, CNNs have the weight sharing property which means smaller number of parameters need to be learned [43]. Besides, CNNs are translational invariant [78] which is rooted from the transition invariance property of the convolution operator [79]. CNNs have had success in handwritten text and document analysis since 1989 [80, 81]. Recent applications of CNNs in constructing 3D shape descriptors are summarized in Section 2.4.2. Also, for a succinct explanation and discussion of CNN we refer the readers to the "Deep Learning" article by LeCun et al. [73].

Autoencoders, are networks that solve the encoding problem with basic architecture of N input and N output units plus a total of $\log_2 N$ units in the hidden layer [82]. The network gets the input, encodes it into the data of smaller size and then decodes it back to the original data. Therefore, no class labels are provided for the autoencoders; instead, they use the input itself as the target output. However, some applications in the literature provided a single target class for a modified version of the autoencoder (called Many-to-one autoencoder) different from original input data (summarized in Section 2.4.2). Depending on the functions used for nodes of the hidden layer, the number of nodes, and the error criteria, the autoencoder can behave like or unlike the PCA [63]. More precisely, the autoencoder learns feature vector or the code for the input through minimizing the reconstruction error [83, 84]. To get more information on the general framework of autoencoders we refer the readers to the article by Rumelhart et al. [82] who first introduced the autoencoders. Also, different types of autoencoders are reviewed in [85] by Baldi.

Restricted Boltzmann Machines (RBMs) are relatively simple neural networks introduced by Hinton and Sejnowski in 1985 [86, 87]. RBMs consist of an input layer and a hidden layer which their nodes are connected in a way that represents a bipartite graph [35]. Each node in the hidden layer is connected to every node of the input layer. All of these undirected edges are assigned individual weights. When the input values arrive to the input layer they are multiplied by their corresponding weights and summed together plus a so-called bias value to make the input for nodes of the hidden layer. Each node of the hidden layer has an activation function that gets the input and generates some output. In the backward path, the outputs of the hidden layer are used as inputs for nodes of the first layer, their weighted sums are calculated, and used as a reconstruction for the original input values. RBM uses Kullback-Leibler Divergence [88] to diminish the difference between the distributions of the input and the reconstructed values [89]. This way the RBM learns the model of the input gradually in an unsupervised manner. Even though RBMs are relatively shallow learning architectures, we discuss them under deep learning algorithms mainly because RBMs are usually used as the basic units of Deep Belief Networks [35].

Deep Belief Networks (DBNs) are learning architectures made up of cascades of RBMs blocks. Therefore, a DBN that consists of a single block is equivalent to an RBM [66]. A famous property of DBNs is the greedy learning approach in which each block of RBM is trained individually [90]. After the training is done and the parameters are computed, the training of the next level starts and so on. In general, in a DBN weights of the last layer are determined through the RBM learning process. Then, all other weights are obtained through a top-down pass. This method of learning is called greedy layer-wise training [66].

In a final pass, the input vector is fed into the DBN. The values for all hidden units are calculated using the predicted weights [91].

Applications of deep learning in the 3D shape analysis became popular in the last decade. It has been broadly applied to various areas such as the natural language and text processing [92], image analysis [93], speech recognition [94], bioinformatics [95], transportation [96], and genetics [97, 98].

2.3 Datasets

This section introduces databases which have been used as the benchmarks for testing the data-driven 3D shape descriptors in the literature. The datasets cover a wide range of objects such as chairs, mugs, bags, vehicles, planes, buildings, human heads and bodies, etc. There are datasets that provide 3D models in unstructured data types such as point clouds and polygon meshes. Another data representations provided in the literature are depth and RGB images which unlike the previous group, have regular grid-like structures. At the end of this section, we summarize the information regarding each data source in table 2.1.

2.3.1 3D Head Database

The head database is provided by the Max-Planck Institute for Biological Cybernetics in Tuebingen, Germany. It contains images of 200 laser-scanned heads without hair from 7 different views [99]. Head structures are obtained by sampling at 512 equally-spaced angles and 512 equally-spaced vertical steps in RGB colors. Each structure contains 70000 vertices approximately. In order to protect the privacy of individuals, images are synthesized by morphing techniques and many are removed. In addition, 5 sets of full 3D head scans are available in the dataset [13].

2.3.2 Princeton Shape Benchmark (PSB)

The Princeton Shape Benchmark (PSB), has been introduced in order to overcome the lack of a general framework for comparing shape-based retrieval and analysis algorithms [100]. The benchmark contains 1814 3D polygonal models collected from the World Wide Web over two years, in addition to the multiple classifications and software tools for evaluating the results of shape matching experiments. The dataset has been split into halves to form the training and test sets. It also includes a set of hierarchical classifications supervised by humans according to functions and forms.

The training and test sets each contains 907 different models and are partitioned into 90 and 92 classes, respectively. From the bottom to the top of the hierarchy, the classes are merged to form coarser-grained classes. In the top most hierarchy, there are only two classes, natural and man-made. The Princeton Shape Retrieval and Analysis Group provides free source code for a variety of tasks including parsing and visualizing the Object File Format (.off) and the classification (.cla) files.

2.3.3 NORB Dataset

The NORB dataset is a large publicly available dataset for the problem of recognizing generic objects, from different lighting conditions and viewpoints purely based on the shape. It contains 97,200 stereo image pairs of 50 toys belonging to 5 categories consisting of the four-legged animals, human figures, airplanes, trucks, and cars. The images were captured under 6 lighting conditions, 9 elevations, and 18 azimuths. There are 10 instances in each category where half of them are used for training, and the other half for testing [101]. The dataset was last updated in October 2005.

2.3.4 SCAPE

The SCAPE dataset is part of the *Shape Completion and Animation of PEople* method, a data-driven method for building a human shape model. The models are generated by combining the shape and pose parameters in order to capture deformations due to changes in both human shapes and poses [102]. The surface data is acquired using a Cyberware whole-body scanner. Full-body meshes are constructed by merging four direction scans which are captured by the scanner simultaneously. Afterwards, each mesh is sub-sampled to about 50,000 triangles. The SCAPE dataset contains a scanned human figure of 37 different people, each in 70 different poses [103].

2.3.5 TOSCA

The TOSCA dataset [104] is part of the TOSCA (Tools for Surface Comparison and Analysis) project. It is a synthetic dataset which contains 3D non-rigid shapes in two categories named *TOSCA high-resolution* and *non-rigid world*. The high-resolution dataset provides 80 objects, including 11 cats, 9 dogs, 3 wolves, 8 horses, 6 centaurs, 4 gorillas, 12 female figures, and two different male figures in 7 and 20 poses. The number of vertices of each object is 50000 approximately. Meshes within the same class share the same triangulation and an equal number of vertices. This property leads to the groundtruth correspondence. As the meshes do not suffer from either noise or missing data, the dataset does not necessarily represent true models in real-world applications.

The non-rigid world dataset [105, 106] is composed of: 9 cats, 11 dogs, 3 wolves, 17 horses, 15 lions, 21 gorillas, 1 shark, 24 female figures, and two different male figures, containing 15 and 20 poses. It also contains a partial dataset of 6 centaurs and 6 seahorses. The number of vertices of each object is 3000, approximately. The datasets are available in MATLAB (.mat) and ASCII text file formats.

2.3.6 SHREC

The 3D SHape REtrieval Contest (SHREC) was initiated by the Network of Excellence AIM@SHAPE in 2006. It included single track of polygon soup model based on the Princeton Shape Benchmark. The next year, it was followed in conjunction with the SMI'07 conference (Shape Modeling International) involving multiple tracks including watertight models, CAD models, partial matching, protein models, and 3D face models. Since then, the contest is being held every year in multiple tracks [107, 108].

The purpose of the event is to provide an environment for researchers who work in the field of 3D object retrieval to present and evaluate state-of-the-art algorithms by using a common test collection. Since 2009, the event is being organized in conjunction with the Eurographics Workshop on 3D Object Retrieval (EG 3DOR) [109]. It has already provided many 3D object models in different categories.

2.3.7 NIST Shape Benchmark (NSB)

The SHape Analysis Research Project (SHARP) with the purpose of investigating 3D shape retrieval algorithms and advancing the state-of-the-art in this field, has been created by the National Institute of Standards and Technology (NIST). The NIST Shape Benchmark (NSB) contains 800 complete 3D models of daily life objects that are categorized into 40 classes, 20 models per class.

The NSB dataset, is collected from major 3D repositories on the Internet with the aim to overcome existing deficiencies of other available benchmarks, including domain dependency, having too few models per class, and having unequal number of 3D models in each class. This might lead to a bias in learning algorithms towards a specific class. Before providing groundtruth database, every model is normalized in terms of size, translation to the center of the mass, and rotation with respect to the principal axes. The latter process is very important to obtain a collection of models that are invariant to rotation, translation and scaling [110].

2.3.8 FAUST

The TOSCA dataset is unrealistic since artist-defined deformations and artificial noise can not reproduce what we find in the real world. The main causes of misaligned vertices are missing data (specially on hands and feet), skin stretching, and clothing. The FAUST dataset was created in 2014 containing 300 real, high resolution human body scans (10 subjects, 30 poses), with automatically computed groundtruth correspondences. The average number of vertices is 172000 for each subject [111]. The dataset is acquired with a full body high-accuracy 3D multi-stereo system.

To achieve the accurate registration of the meshes, subjects are painted with a high frequency texture pattern. And textured markers are placed on key anatomical locations. This explains why the dataset is called FAUST that stands for Fine Alignment Using Scan Texture. The groundtruth correspondences is assured to be accurate within 2mm.

The dataset is subdivided into a training set (100 scans, 10 per subject) and test set (200 scans). The groundtruth correspondences of the training set are defined by aligning each scan with a common template mesh and exploiting both 3D shapes and surface texture information. However, the alignment information of the test set is withheld for evaluation purposes through the provided website. The benchmark is partitioned into 60 scans requiring intra-subject matching, and 40 scans requiring inter-subject matching.

2.3.9 Generic Warehouse Shape Benchmark (GWSB)

The Generic Warehouse Shape Benchmark (GWSB) is a large benchmark provided by the SHARP group from National Institute of Standards and Technology as a special track under the SHREC'10 - 3D Shape Retrieval Contest 2010. It comprises 3168 generic models. All models are classified into 43 classes with different number of objects in each class [112]. The models were downloaded from Google 3D Warehouse. Duplicate and corrupted models are removed and the finalized models are accessible in the ASCII Object File Format (.off) including the models that had been originally created in the sketch-up format (.skp) [112].

2.3.10 Engineering Shape Benchmark (ESB)

Engineering Shape Benchmark (ESB) is a 3D shape benchmark which has been developed by Purdue University researchers. It provides 867 closed triangulated meshes of CAD parts in the mechanical engineering field. Models are available in (.stl) and (.obj) formats in addition to the thumbnail images (JPG). The classification of the dataset has two levels of hierarchies with three super-classes and 45 sub-classes [113]. The dataset has no training set or test set.

The need for ESB dataset despite existing PSB dataset arises from the fact that engineering shapes are characterized by the presence of features such as holes, tunnels, etc., unlike multimedia where the overall shapes are more important. Besides, models such as tables or chairs are considered as assemblies of individual parts in the engineering field rather than gross 3D shapes in multimedia field. Moreover, the PSB classifies models based on their functionality, while in the engineering field parts with different functions may have similar shape and vice versa. Therefore, models in the ESB are classified based on their shapes.

2.3.11 McGill 3D Shape Benchmark

The McGill 3D Shape Benchmark provides repository of 3D models with emphasis on articulating parts. The dataset is a collection of adopted models from PSB and some other repository websites in addition to a number of models created by CAD modeling tools [114]. The complete database contains 456 models, 255 of which are shapes with articulating parts. They are categorized into 10 classes with 20-30 models in each category. The rest of the models are shapes with moderate or no articulating parts [47]. The models are available in different forms including voxelized (.im) and mesh forms (.off or .ply).

2.3.12 RGB-D Object Dataset (RGB-D OD)

The large scale RGB-D Object Dataset is provided by researchers from University of Washington in collaboration with the Intel Labs Seattle. It is freely available to non-commercial research/educational use. There are images of 300 household objects organized into a hierarchical category structure with 51 leaves [115]. The dataset contains synchronized and aligned RGB and depth images of each object at a resolution of 640 × 480 pixel. While each object rotates on a turntable, images were captured using a Kinect style 3D camera from three different viewing heights (30, 45 and 60 degree above horizon). The images are already cropped and segmented [116].

2.3.13 ModelNet

The ModelNet, a comprehensive object dataset of 3D computer graphics CAD models, is collected by the researchers from the Computer Science Department of Princeton University. They intended to expand the variety of categories as well as the number of examples per category in comparison with previously available CAD datasets such as the Princeton Shape Benchmark. The dataset is collected by downloading common object categories from 261 CAD model websites and removing those with too few search results. After that, the miscategorized models were removed using Amazon Mechanical Turk followed by removing irrelevant, unrealistic, or duplicate models manually by the corresponding researchers [117]. Based on the latest information provided on the Princeton ModelNet website, the dataset consists of 127,915 CAD models belonging to 662 object categories.

2.3.14 NYU Depth Dataset

The NYU Depth Dataset is introduced by the researchers from the Computer Science Department of the New York University to provide a better understanding of the RGB-D scenes. It contains diverse and complex real world indoor scenes exhibiting a large number of objects which are recorded by both RGB and depth cameras from the Microsoft Kinect. The dataset is useful for various tasks including recognition, segmentation and inference of physical support relationships.

So far, there are two versions of the dataset available.The second version consists of

1449 RGB-D images, captured from commercial and residential buildings of three different US cities, featuring 464 indoor scenes. It is consisting of 35,064 distinct objects with a dense per-pixel labeling. Images are manually annotated using Amazon Mechanical Turk [118]. Both datasets come with a toolbox containing several useful functions in MATLAB for manipulating and handling the data [118, 119].

The dataset is provided in two versions, *normalized-uniform* and *jittered-cluttered*. In the normalized-uniform version objects are centered in the images with a uniform background [120]. In the jittered-cluttered version, objects are randomly perturbed in 5 ways, superimposed onto the complex background, and have been changed by adding distractor objects to the background toward the boundary of the image [101].

2.3.15 The Shape COSEG Dataset

The COSEG dataset provides 3D smooth manifolds of shapes with the co-segmentation groundtruth and labeling information. It consists of 11 sets of shapes with 7 of them collected from another research project conducted by Sidi et al. [121]. These object classes are the candelabra, chairs, four-legged animals, goblets, guitars, lamps, and vases. One small but challenging sets of the irons as well as three large sets of tele-aliens, vases, and chairs were created by researchers from Shandong University, China [122, 123].

2.3.16 Persistent Heat Signature (PHS) Dataset

The Persistent Heat Signature dataset was introduced by researchers from the Ohio State University after the need for an efficient shape retrieval algorithm in case for partial and incomplete models in presence of pose variations. The dataset was constructed by adopting models from available sources and adding partial and incomplete version of them [124]. It consists of queries of 50 models, in which 18 of them are complete and the rest are incomplete or partial models, and a database of 300 shapes from 21 classes such as dogs, horses, airplanes and etc. [46].

In order to generate the partial data, a model was rotated randomly and all vertices that were visible from the viewing point were kept. One or more portions of a complete model were removed to generate the incomplete data. Each shape is scaled to a unit box to eliminate the effect of the global scaling. Among complete, incomplete and partial data of every pose of each model, only one of them is kept [124].

2.3.17 RueMonge 2014

This dataset was provided by the ETH Zurich university [125] in 2014. The dataset includes 428 3D polygon meshes of textured Haussmanian style buildings with different segment labels. The data which is semantically segmented and annotated includes window, wall, door, street, road, etc labels. They also provide different sets of training and testing datasets in the txt file formats.

2.3.18 ShapeNet

ShapeNet is a huge database of 3D objects (e. g., planes, bicycles, bottles, chairs, tables, etc.) was proposed by a joint research group from the Princeton and Stanford universities and the TTIC institute [126, 127]. This continuing project includes the large-scale 3D models with complete annotations, parts description, and multiple images per 3D data model. The database includes two sub-categories, ShapeNetCore which includes 51,300 3D models divided into 55 classes of objects and ShapeNetSem which contains 270 classes including 12,000 models. The models are all included in a hierarchical taxonomy. Besides the data models, open-source frameworks are also available to view and rendering the ShapeNet models.

2.3.19 Part annotation dataset

A group of researchers from various universities and research institutes (including Stanford University, University of Texas-Austin, University of British Columbia, TTI Chicago, and

Adobe Research) generated a large-scale shape dataset [128]. The shapes of this dataset were obtained from the ShapeNetCore sub-category of the ShapeNet database.The new dataset, in which object parts are semantically annotated, consists of shapes from different categories such as bags, cars, lamps, airplanes, etc. They annotated more than 93,000 parts to generate a massive repository. The dataset is available in form of point clouds along with separate files provided for each object annotating the vertex labels.

2.3.20 Scannet

Scannet is the most recent dataset we list in this section. This database was introduced in 2017 by Stanford and Princeton Universities and the Technical University of Munich [129]. This dataset includes 2.5 million views in over 1500 scans of different places such as offices, apartments, etc. The data models are semantically segmented and fully annotated. A handheld capturing device was designed with a commodity RGB-D sensor similar to the Microsoft Kinect to collect the data. After the data was captured the 3D surface was reconstructed and the scenes were annotated by assigning instance-level semantic category labels and the object instance labels. the object labels are corresponding to the synonym sets of well-known datasets such as ModelNet and ShapeNet.

The data was captured during one month by 20 users in various countries. The project is continuing and included 1513 scans of different locations such as offices, apartments, libraries, etc. initially. The database is suggested to be useful for 3D scene understanding tasks such as labeling, classification, and retrieval.

2.4 Data-Driven 3D shape Descriptors

This section surveys data-driven 3D shape descriptors both thematically and chronologically and proposes an algorithm-oriented taxonomy for the reviewed works. Based on architectures of learning algorithms used to construct the shape descriptors, we divide the descriptors into two broad categories: (1) shallow and (2) deep shape descriptors summarized in Sections 2.4.1 and 2.4.2, respectively. Then, we narrow down this taxonomy, which is illustrated in Figure 2.3, into smaller sub-categories with respect to the specific learning algorithms used under each group. The related works within each sub-category are mostly organized based on their architectures, and the order they were published (sometimes if there were relevant approaches, we reviewed them next to each other disregarding their chronological orders). Learning type as another key feature of a learning algorithm, is also included in this taxonomy. Groups which are indicated with the same colors have the same learning types. The categorization which is illustrated in Figure 2.3 drives the organization of the data-driven shape descriptors being reviewed in this section. Afterwards, Section 2.5 will discuss alternative categorizations of the descriptors.

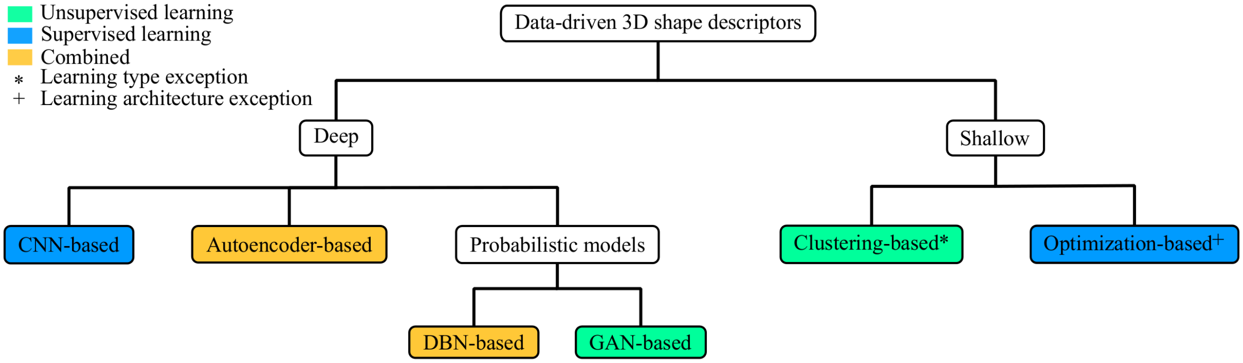


Figure 2.3: Algorithm-oriented taxonomy. The superscripted asterisk indicates a categorization exception, in terms of the learning type. In the framework suggested in [3], even though a supervised approach was used for dictionary construction phase of the BoF technique, it is discussed under the unsupervised clustering-based group. This is mainly because the method used in this work, was close to the works summarized under this group. Also, the superscripted plus sign, shows a categorization exceptions in the learning architecture. The descriptor suggested in [4] used a deep structure, however, because of the bind with the other similar works, it is listed under the optimization-based shallow group.

2.4.1 Shallow descriptors

This group of descriptors which are built using shallow learning architectures are divided into the optimization-based and the clustering-based sub-groups. The works under the optimization-based category were implemented in the supervised manner while the unsupervised learning was used for majority of the clustering-based descriptors.

Optimization-based

The approaches that fall into this sub-category formulate the construction of 3D shape descriptors as an optimization problem which is solved in a supervised manner. In these algorithms, the supervision is provided as sets of corresponding and non-corresponding training points on multiple 3D models. This is as opposed to the regular supervised learning in which the supervision is given in the form of class labels for each instance of the training dataset. A main disadvantage of these approaches is preparing the training data which is usually time consuming as it is often carried out manually. The optimization-based techniques are most useful for establishing point-to-point correspondence and shape matching.

One of the first optimization-based techniques was proposed by Steinke et al. [38] who introduced a machine learning approach to learn the correspondence between two objects from a same class rather than defining the correspondence using the ad-hoc criteria which was used in most of the related previous works. The main focus of this work is to find a mapping function between reference and target objects such that some properties of the reference are maintained under the deformation function. The invariant properties are learned by providing a set of correspondence and non-correspondence points to their learning system. To this end, they provide their learner with a dictionary of basic features. These potential features include a signed distance function and its first derivative (surface normal) and some curvature and texture related features. Their system learn the properties by minimizing the variations between the corresponding points while maximizing the variations between the features of non-corresponding pairs. The learned mapping for each object is proposed to be used as a representation for the entire shape.

In contrast to the works that define shape descriptors in spatial domain (such as the system suggested by Steinke et al. [38]), many recent frameworks introduced 3D shape descriptors in spectral domain. One of the pioneer works in this area, was proposed by Aflalo et al. [130] where the heat diffusion on surfaces is used to learn an optimal heat kernel. They use sets of corresponding pairs of vertices on different meshes along with collections of non-corresponding pairs as the training data. They solve an optimization problem to obtain a diffusion kernel $k(x, x)$ under the criteria of minimizing the ratio between diffusion distances of corresponding and non-corresponding pairs. All of the diagonal elements of k, are exploited as point descriptors (analogous to the HKS) and the histogram of k's values on the shape is used as the global shape descriptor. They use their proposed technique for shape retrieval applications.

The descriptor proposed by Aflalo et al. [130] had the limitation of using a single frequency response to characterize the diffusion metrics. Therefore, Litman and Bronstein [5] continued this work by approaching the problem of finding shape descriptors as a learning problem by considering a multiple frequency response function. They indicated that neither HKS nor WKS has a good level of both specificity and sensitivity. The frequency responses forming their elements are highly overlapped and they are invariant to exactly isometric deformations of the shape. Therefore, they proposed a family of spectral shape descriptors by generalizing HKS and WKS. The new descriptor is obtained by minimizing a distance function. The function finds a matrix of coefficients that minimize the distance between *geometry vectors* of some similar points and at the same time maximize the distance between those of dissimilar points. The corresponding similar and dissimilar points of the mesh are determined when training dataset is prepared. Given two radii r and R (with r ¡ R) for each point p of the mesh, the functions $B_r(x)$ and $B_R(x)$ denote the balls of radius r and R, respectively, centered at point x under the geodesic metric. All points that reside in $B_r(x)$ are marked as alike points to point x, while points that are not in $B_R(x)$ are considered as non-similar to x. The points that lie between r and R are removed from further consideration. These marked points are used as a training dataset to specify the coefficient matrix. They showed that their learnable descriptor outperforms the fixed ones (HKS and WKS). They discussed that

HKS is a 3D shape descriptor which performs poorly in the localization and is more *specific* than WKS while WKS is more *sensitive* than HKS. This makes HKS and WKS more useful for shape retrieval and shape matching, respectively [5]. Their proposed descriptor possesses the advantages of both HKS and WKS. Figure 2.4 shows the results of using all of the three descriptors on 3D human models. Different variations of the work by Litman and Bronstein were proposed by several researchers that are listed below.

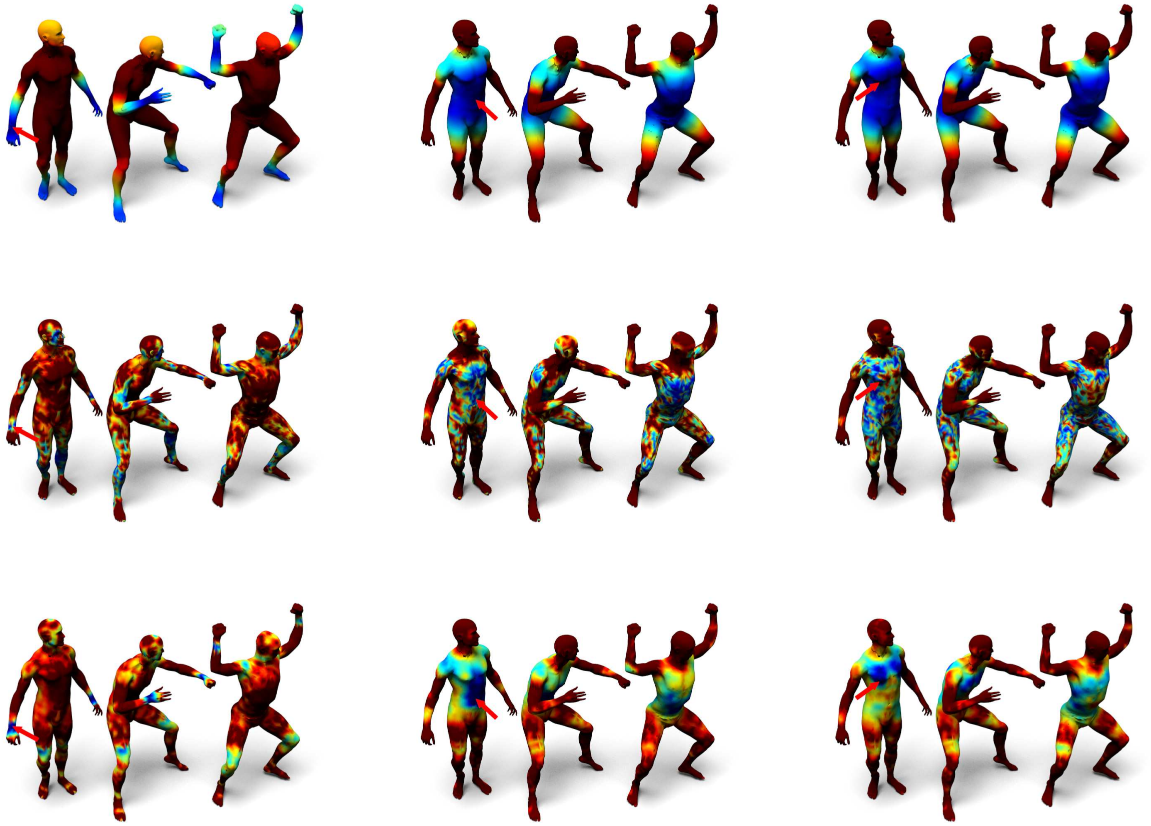


Figure 2.4: The comparison between HKS (first row), WKS (second row), and the data-driven spectral descriptors (third row) proposed in [5]. The Euclidean distances between the extracted descriptors for the white points (shown by red arrows) in each model were computed and visualized from the highest (red) to the lowest (blue). As it is observed, HKS finds the correspondence areas with high specificity, however, it is not sensitive enough. In fact, the areas illustrated with the blue color are connected but they are too large. On the other hand, the blue area in WKS are smaller, however, they are spread all over the model. The proposed descriptors in [5], enjoy both advantages to some extend. As it is observed, the blue areas in the third row, are smaller than the ones obtained using HKS. Besides, they are denser compared with those of WKS. This figure is taken from [5].

In a similar application to [5], Corman et al. [71] used the supervised learning technique for non-rigid shape matching. In their method, a correspondence model is learned given a collection of mappings between a set of training shapes using functional map representation [131]. They used the functional maps as the groundtruth to solve a non-linear optimization problem employing the standard L-BFGS algorithm. In this approach, a training set (consists of computed probe functions that represent local features such as HKS, WKS, etc., for corresponding like and dislike points) is used to learn a set of optimal descriptors (unlike the work proposed by Litman and Bronstein [5] in which point signatures are learned).

An improvement on Litman's work [5] was presented in [4]. This work is an exception under the optimization-based category as it uses the deep architecture. Indeed, we listed it here due to the strong bond it makes to previous works in terms of the way it approaches the shape descriptor learning problem by providing the supervision in the form of known correspondences to the learning algorithm. In this approach, rather than using the linear mapping between the geometry vectors (capturing the local geometric information surrounding the points employing B-spline functions and eigendecomposition of LB operator) and the feature space, Xie et al. employ a nonlinear mapping to deal with large deformations and to describe the shapes more precisely. To learn the binary spectral shape descriptors they solve an optimization problem using a deep Fully Connected Network. The problem formulates the minimization and maximization criteria for the Mahalanobis distance between similar and dissimilar pairs of points (comparable to the approach taken in [5]), respectively. The final descriptor would be the output of the neural network which is converted into a binary vector employing the sign function.

Clustering-based

The descriptors which are reviewed in this section, are mostly built using the BoF technique. In brief, this technique uses a large volume of shapes to construct a codebook or dictionary of visual words (which could be any local features computed for shapes such as HKS, etc.)

employing an unsupervised algorithm. In fact, the local features of shapes are sampled to generate a subset of features (visual words) which will be used to construct such a dictionary. The sampling strategy of feature points plays a key role in selecting the informative and influential visual words to construct a proper dictionary [2]. Various sampling methods were proposed in the literature. Based on how dense or sparse the features are sampled, the result will be referred to as dense [48] or sparse [132] feature point subset, respectively. These sampling methods and their applications are discussed later in this section. After the dictionary is constructed, histogram of the visual words are computed for each input 3D shape, as a global shape descriptor. Different vector quantization algorithms such as k-means clustering or sparse coding have been used in the literature for this step. The major advantage of using BoF technique is that it does not require any class labels for training datasets. On the other hand, its limitation is that it usually ignores the joint occurrences of visual words resulting in the lack of spatial information in computed shape descriptors. Some novel techniques however, included the expression information into the descriptors by considering joint occurrences of the visual words to improve the performance. Also, other applications suggested to provide some levels of supervision at the time of building the dictionary to increase the efficiency of shape descriptors. These techniques are discussed later in this section.

One of the first applications of machine learning algorithms in 3D shape analysis in which a clustering algorithm is used for shape descriptor construction was proposed in 1987 by Hoffman et al. [34]. In this application, input is provided in the form of range images and feature vectors (such as coordinates of the pixel in the image, the depth information, etc.) are computed for each pixel of input. Then, a clustering algorithm is employed to partition the input image into different segments (or surface patches). They suggested using a decision tree algorithm for coarse classification of surface patches and assigning concave, convex, or planar class labels to each patch of the image. Two suggested applications of the proposed classification method are object recognition and shape representation. They recommended to use a combination of the patch class labels for shape description. Other researchers in the literature, used the concavity and convexity information for shape segmentation tasks, later on.

After the above background on an early clustering-based approach, we review a set of more recent applications of these techniques in shape description. Using the BoF technique, Toldo et al. [132] segmented 3D shapes into their constructing parts, computed local descriptors for each part, and then built a global shape descriptor. In detail, the concavity and convexity information extracted from principal curvatures is employed for shape segmentation (similar to the pioneer approach proposed by Hoffman et al. [34]) in their work. Then, four different local descriptors, namely Shape Index (SI), Radial Geodesic Distance (RGD), Normal Direction (ND), and Geodesic Context Histogram (GCH) are computed to construct a single descriptor for each segment. The K-means clustering algorithm is exploited to generate a visual codebook using the region descriptors. Finally, histogram of the visual words included in the dictionary is used to construct global shape descriptors which are applied to retrieval and partial matching tasks. The authors in [133] proposed a similar schema for shape categorization applications. However, they used a different set of local descriptors for shape segments.

The techniques proposed in the works suggested by Toldo et al. (e.g. [132, 133]) improve the performance of BoF technique by including salient visual words (i. e., the segment features) into the dictionary (these application are examples of sparse feature sampling algorithms). Another improvement in BoF technique in shape representation is suggested by Bronstein et al. [48] who introduced spatially sensitive BoF techniques. More specifically, they use both dense (computed at each point of the shape) and sparse (computed at salient features of the shape) spectral descriptors to construct a vocabulary of geometric words using the K-means algorithm. Each shape is represented by its distribution over the words of this vocabulary. Furthermore, they suggested to compute the histograms using a combination of words (resulted in spatially sensitive descriptors) that they referred to as expressions. In the end, metric learning techniques [134] are employed to encode input shapes into binary representations so that they could be indexed and compared easily. Their technique outperformed the approaches which consider single geometric words, in the presence of noise or in partial shape retrieval applications.

The previous works that sampled feature points using either sparse [132] or dense [48] approaches, are confined by some limitations [2]. For example, the quality of mesh topology can affect the sparse sampling. Also, the dense sampling requires to be spread equally on meshes which could be more of a problem in low quality mesh tessellations [2]. To address these issues, Lavoue used the BoF technique to establish shape descriptors out of randomly selected feature points on meshes [2]. To this end, he randomly selects a set of seeds on the mesh and then applies the Lioyd's relaxation algorithm iteratively multiple times so that the seeds are spread on the surface uniformly. He includes the spatial information in the descriptor considering the histogram of pairs of words rather than merely single visual words that ignores this information. In this work, a spectral descriptor is computed for each feature point over its local neighborhood, using the Fourier spectrum of its surrounding patch. The K-means clustering algorithm is employed to construct the codebook of visual features. Then each shape is illustrated as a histogram of the visual words as in the standard BoF framework. In this spatial BoF technique, the final histogram includes the information regarding pairs of visual words that are close in the spatial domain. Furthermore, the standard and spatial versions of BoF are combined to construct a global shape descriptor that is applied to both complete and partial shape retrieval.

BoF technique has been applied to various input data types in the literature. In an application proposed by Blum et al. [116], RGB-D images are used to extract global shape descriptors. In this work, at first salient feature points are detected in RGB-D images using the SURF algorithm [135]. Then, a region (by the size 16×16) surrounding each feature point is determined in all four channels (color and depth) of the input. Multiple random patches are selected in such region, to generate the training set X. This training set is clustered using the k-means algorithm to obtain k cluster centroids which are all added to the dictionary. After the dictionary is built, it is used to generate global shape descriptors for unseen images. Therefore, for any new image at first, the SURF algorithm is used to find the feature points in a similar approach to that of the training phase and then patches are sampled randomly in the surrounding region of the salient points. The distance between the patch and all of the k cluster seeds are computed and averaged to be used for generating the salient points histogram vectors. If the distance to a centroid, is less than the average, the corresponding entry for that centroid in the histogram vector is set to 1 and it is set to 0 otherwise. Finally, these histogram vectors are sum-pooled to form the overall descriptor. The authors, compared the classification performance of their unsupervised method with supervised algorithms such as the SVM [69] and Random Forests [136] showing a considerable superiority.

In another view-based technique, Lian et al. [137] proposed to construct 3D shape descriptors for retrieval applications using depth images. After a preprocessing step (such as the pose normalization), they represent the input shape with a set of depth images from which salient SIFT [138] features are extracted. Rather than using the K-means clustering to find the cluster centroids, they simply use a randomly selected set of local features to create the codebook. Afterwards, each generated view is represented by a descriptor which is actually the histogram of visual words. Finally, the Clock Matching technique is applied to calculate the dissimilarity between two shapes for shape retrieval using the computed shape descriptors. Their suggested descriptor, expose superior performance in both rigid and nonrigid shape retrieval in comparison to a set of state-of-the-art methods.

In contrast to the previous methods that used unsupervised learning for dictionary construction, Litman et al. [3] proposed a supervised framework for BoF technique to improve its performance. This supervised application is an exception listed under the clustering-based category which includes unsupervised techniques. It is mainly because this application uses a BoF framework and is tightly coupled with all the previous techniques we mentioned in this section. In the previous works, no shape class labels are provided at the time of constructing the dictionary; however, Litman et al. suggested a supervised learning approach by incorporating the class labels in the codebook generation phase. In fact, in unsupervised approaches local shape descriptors are computed for a large set of training shapes and then such descriptors are partitioned into k clusters which their centroids are added to the dictionary. Finally, histograms of the visual words are computed for unseen shapes as global shape descriptors. Typically, the k-means clustering algorithm is used for vector quantization and computing the histograms in the standard BoF technique. In contrary, in the approach suggested in [3], which is tailored to retrieval task, class labels are available in the training phase. They also suggested sparse coding as a substitute for vector quantization in the standard BoF technique. In brief, the output of vector quantization procedure (typically implemented using k-means algorithm) are binary vectors with a single non-zero element. However, sparse codes could have multiple nonzero elements with arbitrary scalars. To include the supervision into the dictionary construction phase, they created a set of similar (objects from the same class) and dissimilar (objects from dissimilar classes) training instances. The criterion for adding a visual word to the dictionary is that it must minimize the distance between the sparse codes of similar instances and at the same time must maximize the distance between the codes of dis-similar samples. This way rather than using a fixed dictionary as in standard BoF, the dictionary gets updated by each training entry knowing its class label. At last, the dictionary is used to build the global descriptors in the BoF framework. The performance metric comparisons between this approach and other recent techniques such as ShapeGoogle [48], proves the excellence of the proposed framework.

In a recent approach, Wan et al. [46] applied the BoF framework for incomplete 3D shape retrieval. With this application the researchers showed that a dictionary of local features which was built using complete shapes could be used to extract local descriptors for incomplete shapes, if the words of the dictionary are sparsely coded. In this approach, at first sparse dictionary learning is used to construct dictionaries of HKS basis descriptors for each individual complete shape (e.g., complete human body, complete hand, etc.). Then, HKSs are computed for all vertices of incomplete input shape which its actual class should be retrieved. For such an incomplete shape S_i the researchers reconstruct each signal f_i (where f_i represents a local shape descriptor computed for vertex i of S_i) as a linear combination of basis descriptors of a dictionary D_c which is built exclusively for a specific shape S_c. These reconstructed vectors are local shape descriptors for the incomplete shape. In the end, the reconstruction error of all $f_i s$ are summed to compute the distance between S_i and S_c. As each shape has its exclusive dictionary, the distances between S_i and all those shapes are computed using local shape descriptors and relevant dictionaries. The smallest distance among others determines the actual class for the input. With this application the researchers measured the distance between complete and incomplete shapes using the constructed data-driven local shape descriptors. This technique avoids the approach taken by most traditional applications in which feature points are extracted and matched between the shapes for incomplete shape retrieval. A disadvantage of the traditional approaches which is addressed in this work is that missing parts of incomplete shapes can affect the feature points detection, easily [46].

2.4.2 Deep descriptors

A wide range of deep learning algorithms have been used in the literature to construct the 3D shape descriptors. We divide this group into descriptors that are built based on probabilistic models, autoencoders, or CNNs. The probabilistic group is further divided into two sub-categories named DBN-based and GAN-based descriptors.

The CNN-based approaches are implemented in the supervised manner while the majority of the other techniques use variety of learning types (Figure 2.3).

Probabilistic models

A group of state-of-the-art applications in the literature use probabilistic models to learn 3D shape descriptors. The works reviewed in this category are mostly implemented using the DBNs; however, some state-of-the-art works use Generative Adversarial Networks (GANs) to construct the shape descriptors. Descriptors of this category are divided into the DBN-based and GAN-based sub-groups, accordingly.

DBN-based The satisfactory performance of DBNs on unlabeled data [120] makes them good candidates for the unsupervised learning tasks (however, the works that utilized DBN were mostly using a combination of supervised and unsupervised learning types). Furthermore, because of their greedy training scheme in which each layer of the architecture is trained locally they overcome the training complexity of other deep architectures [47]. This section reviews proposed methods that employ DBNs as major components in their pipeline to construct shape descriptors.

In one of the first applications of deep learning in constructing 3D shape descriptors, Nair and Hinton [120] proposed to exploit DBNs in a semi-supervised manner. In this work, the efficiency of DBNs on unlabeled data is coupled with a level of supervision provided to the learner through a small set of labeled data, resulting in a semi-supervised algorithm for DBNs. Originally, in DBNs which are stacked RBMs each layer is trained individually in an unsupervised manner. Through this training process a DBN learns levels of non-linear features from the provided data. Nair and Hinton discussed two different approaches could be taken to use DBNs for classification purposes. In the first approach, class labels of each training object could be provided to a pre-trained DBN to fine-tune the learned hierarchies of features. In an alternative approach, the highest level of unsupervised features (learned by the pre-trained DBN for each training object) along with the object's class label could be provided to the last layer of the DBN (an RBM) to train the system for classification applications. In this work, Nair and Hinton suggested a new top-level model for DBNs in which both hidden and visible (last layer) units along with the class labels contribute in training the DBN for object classification tasks. Their results shows that using deep learning outperforms shallow models particularly when their novel top-level model is used for DBNs. The proposed DBN could be trained by providing either a set of labeled feature vectors or labeled images to learn a 3D tensor of parameters. These parameters code inputs into global shape descriptors.

A more recent application of DBNs was proposed by Liu et al. [47] in which a global descriptor is built based on the rendering multiple 2D views for 3D shapes. In this approach 200 depth images are taken from the input 3D model, then low-level local features are extracted out of the images using the SIFT [138] algorithm. After that, a visual vocabulary is built and the BoF technique is used to extract histogram of the words for each model as the descriptor (middle-level features). These vectors are fed into a DBN to learn higher level features. The new extracted features are used as shape descriptors to represent the models for classification and retrieval purposes.

Another application of applying DBNs on 3D data, was proposed by Wu et al. [117]. Rather than using a view-based approach (approaches that use 2D/2.5D images as input) they transformed the unstructured input data into a volumetric representation to be used as input in their approach. This novel application, utilizes a convolutional DBN (CDBN) to represent a 3D model in the form of probability distribution of binary variables on a 3D voxel grid. Their system, called 3DShapeNets, which is the first application of deep learning on 3D data, receives a 2.5D depth image as input and generates a volumetric structure, in which the shape distribution is learned. In more detail, they represent the shape in the form of a 3D matrix of size $30 \times 30 \times 30$. Each element of this matrix is set to be either 1 or 0 if its corresponding voxel resided inside or outside of the mesh, respectively. Multiple levels of filters are convolved with the input subsequently, resulted in computing a 1D vector. In their proposed architecture, they use the weight sharing property of convolution to reduce the parameters to be learned; however, this structure does not contain pooling levels to prevent the extra uncertainty in the shape reconstruction. Finally, the 1D vector along with a class label are presented to the top layer of a learning architecture which is built as an RBM with 4000 hidden units. This layer made an associative memory that learns the distribution of the binary values mentioned above. Such a distribution is used as a global descriptor for the input shape. The proposed descriptor was evaluated in classification and retrieval applications.

The *extrinsic* works such as the view-based and volumetric approaches, suffered from the problem of not being able to capture the shape deformations; therefore, the researchers moved towards the approaches which could analyze the shape deformations. An example is the work suggested by Bu et al. [45] who employ a DBN to build 3D shape descriptors from local features (i. e. HKS and the Average Geodesic Distance (AGD)). They establish middle-level features from the low-level features using the BoF technique. Then, by stacking a number of RBMs, they make a DBN to learn the high-level features out of the middle-level ones. Such high-level representations are used to classify and retrieve 3D objects. In another experiment, they compare the retrieval performance of the middle-level vs the high-level features and show the latter is superior to the former. The retrieval performance of this approach, demonstrated substantial success on some benchmark datasets while the similar work proposed in [47] for 2D views, showed much less proficiency.

Another state-of-the-art example was proposed by Han et al. [6] by combining benefits of the DBN in unsupervised learning and advantages of convolutional networks, to propose a local shape descriptor called circle convolutional RBM (CCRBM) [6]. The work proposed by Bu et al. [45] was novel in terms of being applied on 3D meshes; however, it was limited by the requirement of computing the local features and using them as the input to the learning algorithm. In contrast, the innovation proposed by Han et al. is to use 3D meshes directly as input for a DBN algorithm. Applying the convolution on 3D shapes is not as straightforward as applying it on 2D images due to the irregular structures of 3D models. To tackle the problem, Han et al. suggested a circular convolution in which a sector window is located on a vertex and is rotated around the normal vector of the vertex by some stride angle in a specific direction, to compute the convolution. While calculating the convolution, the local area is projected onto the tangent plane perpendicular to the normal vector of the center vertex (Figure 2.5). The Projection Distance Distribution (PDD) is used to generate a histogram that shows number of the points residing within a sector window. CCRBM used three virtual, detection, and max-pooling layers to learn the final point signature from the PDD.

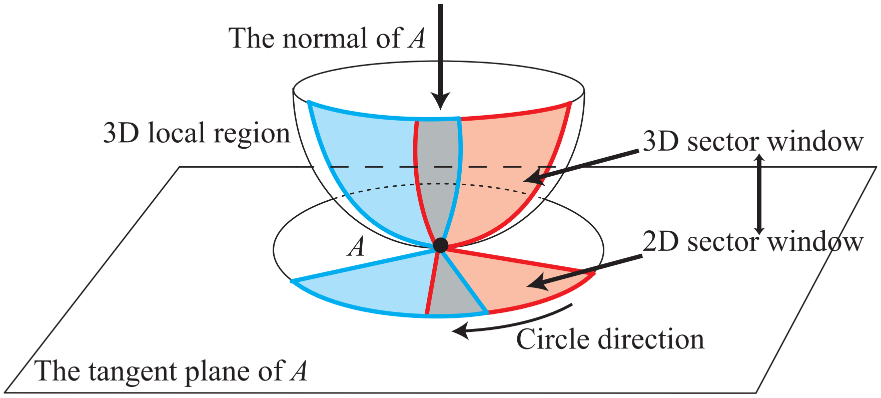


Figure 2.5: Circle convolution on a 3D local area, taken from [6].

GAN-based Generative probabilistic models are algorithms that receive a set of samples taken from a specific distribution as the training data and learn an estimation of the data distribution. There are two types of generative models. A group which are able to generate new samples based on the distribution that they learned, and a group that estimates the probability distribution of data, explicitly [64]. The latter is usually accomplished through learning compact descriptors for groups of shapes. GANs are special types of generative models that generate sample data; however, they are used to provide distribution of the data, as well [64]. An advantage of using GANs is that they could be easily used for unsupervised learning as well as being trained with the missing data (e. g., semi-supervised learning) [64]. To review the most recent works on GANs at first we list the similar works in the literature which introduce a history of GAN-based shape descriptors.

Kalogerakis [139] proposed a generative model which learns the underlying model of the training data and generates samples of that model. Learning the data models require a dataset of the shapes segmented into their constructing components along with their geometric features (i. e. a detailed feature vector for each component including 3D scale vector, curvature information, etc.) as the training data. In fact, for each training segmented shape, a feature vector including the discrete and continuous feature vectors as well as the number of components of each category (observed variables) are provided to the system. After the system is trained, it is able to combine various components, to generate new synthesized shapes by capturing the probabilistic relationship between the components of a class of shapes. Besides, the latent variables of the probabilistic model (learned through Expectation-Maximization (EM) algorithm) could result in a compact representation of the shape segments. However, no measurements of the shape descriptor applications are reported in this work.

Following the above idea, in another application Huang et al. [140] proposed a structure based on Deep Boltzman Machines (DBMs) to learn the probabilistic model of the data and the compact shape descriptors. They improved the preceding wok [139] by providing much less amount of supervision to their algorithm. In fact, they did not rely on the training data of segmented shapes. Their proposed system which learns a hierarchical part template consists of multiple phases. First, the input data (of each particular group for example, chairs, tables, etc.) are clustered into the shapes with similar compositions. Then, within each cluster the different parts of the shape are learned, and finally, the high-level part models are learned. Such a high-level representation is used to establish the correspondence between the shapes from different classes. In the next phase, a probabilistic deformation model is learned by estimating the joint probability of point correspondences and the part template assignments. Such a model is the basis of learning surface variability within a category of shapes which is hierarchically captured in the latent variables of the model. Top layers of the latent variables produce the compact shape descriptors that are applied to shape classification tasks.

To improve the previous work suggested by Huang et al. [140] through avoiding the supply of a training dataset of parts, Wu et al. [141] proposed using GANs to learn representations of 3D shapes from the volumetric data. In this approach, the researchers proposed a GAN in which an adversarial discriminator was added to the generative modeling. In fact, in the GAN (called 3D-GAN) a generator and a discriminator work together. While the generator synthesizes the new samples (trying to challenge the discriminator), the discriminator is trained to classify real and generated samples, accurately. The discriminator generates a score in the range [0,1] for each sample, determining how likely the sample belongs to real or synthesized classes. Five volumetric convolutional modules plus some layers in between (e. g., normalization layers, ReLU layers, etc.) make the building blocks of the generator. On the other hand, the architecture of the generator is inverted to build the discriminator module. In this framework, both generator and the discriminator are trained, simultaneously. After the system is trained, the discriminator learns the representations for the shapes. The accuracy of the learned descriptors is comparable with a list of supervised learning methods, and in some cases (e. g., the 3DShapeNets [117]) it is better. In comparison with the other unsupervised techniques, 3D-GAN is superior. For an in depth article on GANs, we refer the readers to the comprehensive tutorial prepared by Goodfellow [64].

CNN-based

This group of broadly used descriptors, are constructed using the convolutional neural networks. The benefits (see Section 2.2.3 for details) of CNNs such as the weight sharing property which simplifies the network substantially, have made CNN-based shape descriptors popular in many shape analysis tasks. This attribute significantly reduces the number of parameters to be learned which is a desired feature in deep networks. On the other hand, irregular structure of meshes and lack of the shift-invariance property [44, 142] in unstructured nature of 3D data, makes it complicated to apply the convolution on 3D shapes (as opposed to the straightforwardness of convolving filters with 2D images). Therefore, many researchers suggested novel ideas to bridge this gap by either suggesting an equivalent grid-like representations for 3D data or proposing a novel implementation of the convolution operation for 3D data. This section reviews the convolutional approaches, beginning with systems that are pioneers in extending the convolution operation to 3D domain (being motivated by successful applications of CNNs in 2D images). We review a diverse list of CNN-based approaches in which the descriptors are built using various input data types.

One of the first applications of CNN in constructing 3D shape descriptors was proposed by Socher et al. [78]. In this application, a 3D object classifier is developed using a combination of CNN and Recurrent Neural Networks (RNNs). In detail, at first RGB-D images (representing 3D data) are given to a CNN to extract low-level features such as edges. Then, these features are given to an RNN to learn combinational higher level features and their relations. This is carried out by mapping the input into a lower dimension space, resulting in the shape descriptors used for object classification. They reported that their fast combinatorial model worked more efficiently in comparison with the other methods such as SVM [69] and Random Forests [136].

In a state-of-the-art approach Su et al. [143] designed a two-layer CNN system to build a 3D shape descriptor using 2D images as input. The proposed descriptor is computed on multiple 2D images rendered from a 3D model [143]. The collection of 2D views, are presented to a CNN hierarchy to generate a compact 3D shape descriptor. Each CNN of the first level extracts the information from its single view input image, resulting in a descriptor for the view. Then, the higher level CNN learns a combination of this information to create a single shape descriptor. A contribution of this work is that rather than simple averaging or concatenating the multiple descriptors, they are combined using a CNN architecture. The researchers used multiple 2D views of a single 3D model to train their system for classification and retrieval applications. Their proposed system, proves to be superior compared with the ShapeNets [117] in retrieval application. Also, testing the system for classification application by providing a single view, resulted in an enhancement with respect to the ShapeNets.

The previous works that used 2D/2.5D views of a 3D shape as input (we refer to them as view-based techniques), used the *extrinsic* approaches to represent the unstructured data.

These approaches are unable to analyze the isometric deformation of the shapes. To deal with such a limitation, different grid-like representations are suggested by researchers that capture more detail (deformations) of 3D shapes. For example, Sinha et al. [43] introduced a new implementation of the geometry images for 3D shapes to fill the gap of applying the CNN on 3D data avoiding view-based techniques. To extract geometry images, they transform the surface of a mesh into a sphere which is later mapped into an octahedron. Then, the octahedron is cut and padded into a grid-like geometry image. The intrinsic properties of the input mesh such as curvature information and the HKS local descriptors are coded into the suggested geometry images. These images are used as the input for CNN to compute shape descriptors that represent the surface of input shapes. They demonstrated this approach outperformed the previous volumetric ([117]) and view-based ([48]) approaches in both classification and retrieval applications.

In another application, Guo et al. [42] used CNN to build local 3D shape descriptors which were learned from low-level geometry features of input mesh and were applied to a labeling task. In this work, the researchers introduced a new grid-like representation for each triangle of irregular mesh data. Such a representation generates vectors of 600 components for each face of input mesh. These vectors are computed from seven geometry features (e. g., curvature, average geodesic distance, spin images, etc.) of each face and are reshaped as 30×20 patches to be used as the input to the proposed CNN architecture. A sequence of convolution, down-sampling, and pooling combines the 600 values non-linearly resulting in a compressed feature vector V containing 192 components. In the last step, a mapping function is used to convert V into a vector P of length N_c which determines the probability of input face belonging to each of the N_c components of the shape. The parameters of proposed CNN and mapping function are learned in a supervised manner given a vector of class labels for each input. Such groundtruth information is provided in the form of binary vectors with a single non-zero element representing the part of object that input face belongs to. The generated local shape descriptors (Ps) in this work are used for shape labeling application.

The work suggested by Guo et al. [42] which uses feature vectors computed for faces of the input mesh is intrinsic and deals with the unstructured-ness of meshes; however, this approach ignores the spatial correlation [144]. Therefore, Zeng et al. [144] proposed a CNN-based system that addresses this limitation by considering local neighborhood of points in the input mesh. In detail, a patch surrounding an arbitrary point (randomly selected) on input is mapped into a local descriptor. In fact, a set of known corresponding and non-corresponding patches are used as groundtruth matches to learn a nonlinear mapping function. The function finds local descriptors for similar points in such a way that the l_2 distance between their descriptors is smallest. The patches are provided to the CNN in the form of $30 \times 30 \times 30$ voxel-grid representations and 512-dimensional local descriptors are computed for patches. The network architecture consists of eight layers of convolution plus a single pooling layer. They use a siamese network architecture and provide pairs of inputs simultaneously, to train the network. In siamese architecture two identical networks are used concurrently to compare two input patterns. These networks share the same parameters and weights and compute a single output that demonstrates the similarity between the input data [145]. The constructed local descriptors are employed for applications such as matching and registration. In this work, the proposed network is trained using self-supervised learning in which the target values do not come from human annotations; instead, they are taken from an existing source in the world [146]. For example, correspondence labels for this network is adopted from various RGB-D reconstruction datasets.

The work suggested by Zeng et al. [144] was constrained by limited training data and the low resolution of the volume generated around points of interest [147]; therefore, Huang et al. proposed constructing local shape descriptors in a view-based technique to exploit numerous image datasets for training their system [147]. This work is a novel technique in which a view-based approach builds local descriptors as opposed to all of the previous reviewed works in the literature that construct local descriptors from the mesh, volume, or point cloud data. Following the idea suggested by Su at al. [143], the authors train a

CNN-based architecture to construct local descriptors. They render multi-scale 2D images for each point of the input shape (could be polygon meshes or point clouds) and use pairs of similar and dissimilar points as training data for a siamese network. Using a network consisting of a set of convolution, pooling, and nonlinear transformation a 4096-dimensional vector is computed for each view. These vectors are combined in the max view pooling layer to generate a single vector of the same dimension for each point. Finally, a dimensionality reduction technique yields a vector of a lower dimension $K = 128$ by experimenting a wide range of values for K. The vectors computed for the pairs of training data are computed such that the similar points are mapped to closer descriptors and the distance between the dissimilar pairs are maximized. A part-aware nonrigid alignment method is employed to generate a massive amount of correspondence training data for their system.

Masci et al. [44] also proposed a novel approach by an architecture that expands the CNN to non-Euclidean manifolds and is able to capture shape deformations and anisotropic structures. They suggested a geodesic CNN (GCNN) to learn features for shape description by generalization of CNN to manifolds and establishing a system with multiple linear, geodesic convolution, angular max-pooling, Fourier transform magnitude, and covariance layers. In this framework, a local geodesic patch is defined to apply the convolution on triangular meshes in a point-wise manner. In more detail, once being placed on each vertex x_i of input mesh, the geodesic patch divides 1-ring neighborhood of the vertex into equal N_θ angular bins. Then, the bins are propagated to the neighboring triangles of the 1-ring area resulted in a more expanded region. Also, this selected region is divided into N_ρ radial bins being equally expanded over the region. Such a patch is slid over the triangular mesh to convolve the filter with functions (e. g., HKS, WKS, etc.) defined on vertices of the input mesh similar to the approach which is carried out in 2D domain. This proposed GCNN is employed to build local shape descriptors in a supervised manner. Using a set of known corresponding and non-corresponding points as the training set, parameters of the network are obtained in a way that minimize the distances between the alike points and at the same time maximize the dissimilarities between the uncorrelated points.

The work suggested by Masci et al. [44] was continued by Boscaini et al. [37]. They generalized the defined convolution to the spectral domain for learning local shape descriptors. The main advantage of this work over the proposed approach by Masci et al. [44] is its flexibility in being applied to various 3D structures. In fact, their proposed convolution operation is applicable to frequency domain and hence it is not limited to polygon meshes only (i. e., it is applicable to other 3D data structures too, such as point clouds). In more detail, the patch operator that was applied on meshes in [44], is replaced with a patch which works in the frequency domain. In this application, Windowed Fourier Transform (WFT) is generalized to the spectral domain to extract local patches. WFT is applied on a function f defined on input mesh (f could be a point descriptor such as HKS, WKS, etc.) and by taking the first K frequencies of WFT into account, a "meta-descriptor" (i. e., a K-dimensional vector) is obtained. A group of windows with different sizes are used for this step and for each window size, a 300-dimensional vector is computed as point signatures. The vectors summarize function f surrounding a certain point x. A two-layer CNN with a function f as the input is used to produce a Q-dimensional descriptor at the point x. The similar set of training data which was used by the previous work is employed in this system as well to learn descriptors.

In a similar application, Boscaini et al. [7] proposed local descriptors that are declared in the spectral domain; however, they utilized anisotropic diffusion to take advantage of the curvature direction in learning local descriptors using CNN. To form the anisotropic descriptors, they introduced the definition of *anisotropic Laplacian*, in which a thermal conductivity tensor is incorporated in the diffusion equation

$$f_t(x,t) = \mathrm{div}_X(\mathbf{D}(x)\nabla_X f(x,t)) \tag{2.1}$$

Using this equation the direction information can be included in the diffusion resulting in the descriptors that are unambiguous to symmetry (Figure 2.6). For more information on applying CNN to spectral domain we refer the readers to a recent article by Bronstein et al. [148]. They present an in-depth discussion on the considerations for CNN applications in shape analysis tasks for spatial and spectral domains. In summary, defining the convolution in the spectral domain resolves the problem of expanding convolution to the graphs; however, such formulation of the convolution would be domain dependent.

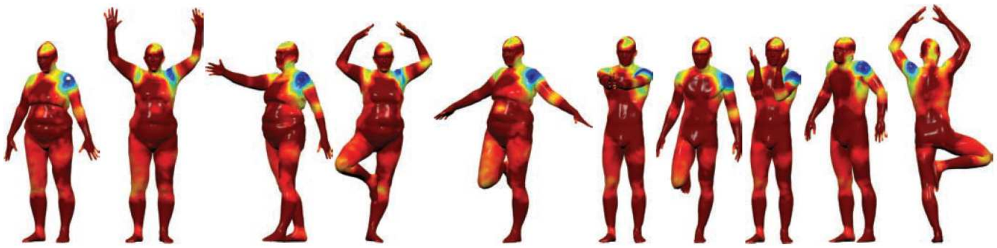


Figure 2.6: The Euclidean distance between the descriptor of the white point on the left shoulder obtained in [7] and those of the other points are represented. Red color corresponds to the higher distances. As it is observed, the point is detected locally with high level of sensitivity and specificity. The figure is taken from [7].

In a recent work proposed by Qi et al. [149], the performance of CNNs applied on volumetric data and multiple 2D views are compared and it is shown that the former performed worse than the latter. The researchers propose three techniques to improve the performance. First, they adopt the architecture of multi-view CNNs to achieve a volumetric CNN in which an anisotropic elongated kernel is employed to convert the input data from a volume to 2D representation. In fact, the kernel is convolved with the volumetric data followed by an aggregation step to form a 2D plane representation of the input to be fed into the image-based CNN. This novel approach enables users to take advantage of rich volume data as well as the efficiency of multi-view CNNs. Second, in a separate improvement they enhance the performance of the volumetric CNN, through augmenting the data by both azimuth and elevation rotations rather than a single orientation used in 3D ShapeNets by Wu et al. [117]. Finally, they add a multi-orientation pooling stage to the system to capture various orientations in the input. Similar to other CNN architectures, a sequence of convolutions and pooling operations (mlpconv was used in this work to be more discriminative [149]) extracts high-level features out of the input data being used as shape descriptors.

In another state-of-the-art study, Qi et al. suggested a deep learning algorithm called ”PointNet” to form global and local shape descriptors on raw point cloud data for the first time [150]. They discussed that methods which generate structured representations (e.g., grid-like or voxelized) for point clouds or meshes to feed them to CNNs, transform the data into large volumes which leads in computation complexities. Moreover, they found these methods result in fine details loss due to the quantization process. Therefore, they suggested an approach that consumes the raw data directly to avoid such limitations. Their system, takes a $n \times 3$ matrix, containing coordinates of point clouds in 3D space as input (plus some additional features such as normals) and generates a vector representing k class assignment scores for a classification task. The concept of order is a challenge of working with unstructured data (which does not apply to the regular data such as 2D images). Qi et al. addressed this problem by using a symmetry function. More specifically, the input data goes under a transformation (to be canonicalized) and then is fed to a MLP that generates a $n \times 64$ matrix as point features. These features, are used as the input to a symmetric function (implemented in form of a max pooling, an average pooling, and a weighted sum layer) to make the system invariant to the input order. This symmetry function works as a mapping that transforms the irregular input point cloud into an ordered vector. The proposed system was also tested for a segmentation task, generating a $n \times m$ matrix that illustrates segment assignment scores for n points and m different segments of the input. By concatenating the shape descriptor and point features extracted in the classification network, a new set of local features are extracted in the segmentation module. The proposed architecture in this work is pretty similar to the CNN, because of extracting a set of local features from the data within the deep architecture (similar to the application of convolution in a CNN). Also, because pooling could be think of as a special convolution operation [151] we categorized this work under the CNN-based group.

Qi et al. [152] proposed an improvement on the above work by introducing a hierarchical neural network, called PointNet++. This enhancement generalizes the previous design by applying the PointNet recursively on overlapped subsets of the input data in a multi-resolution manner to learn local hierarchical features in the metric space.

Another work inspired by the PointNet was introduced by Deng et al., recently [153]. They discussed limitations of the PointNet such as constructing task-specific local descriptors. Therefore, they suggested the PPFNet to improve the performance of PointNet by extracting more powerful global-aware local descriptors. In this system, the supervision is provided in the form of sets of N-combination of input points (as opposed to pairs of corresponding and non-corresponding points utilized by optimization-based systems). In more detail, a set of N local patches are selected uniformly from two input point clouds. The similarities and dissimilarities between the pairs of input patches are given to the networks for the training data. The patches are given to separate weight-sharing PointNets to extract local features which are then fed into a max-pooling layer to construct the global features and attaching them back to the local features. In the last step, a group of MLPs are used to combine the global and local features, furthermore. Finally, a distance matrix of all patch descriptors are computed and given the groundtruth and using the back-propagation the architecture updates PPFNet's weights such that the optimal descriptors are constructed.

A recent and novel convolutional approach for shape labeling and keypoint detection was proposed by Yi et al. [142], in which a kernel is convolved with the function defined on input mesh in the spectral domain. To this end, the graph Laplacian of input mesh is formed and its eigenfunctions are computed to be used as basis functions. Function f defined on the mesh is represented as the linear combination of such basis functions. The collection of magnitudes associated with each basis function is used to represent the function f. Analogous to the Fourier analysis, the convolution in the spatial domain is transformed to a point-wise multiplication between a kernel and the function in the spectral domain. The input to their neural network, are functions which represent a set of hand-crafted local features (such as geometry and curvature related features) computed on each vertex as well as the graph Laplacian of 3D objects. Their system generates a segment label or a landmark detector function for each vertex of the mesh by computing high-level features for the vertices. This is carried out through a convolutional network including multiple steps of spectral-spatial transformations which are the key blocks of their deep architecture. Because of the suggestion of a synchronization over spectral representations of input shapes, the researchers introduced their approach being generalizable to shapes with various topology and geometry.

The last two works which are reviewed under this group, are approaches that use the octtree idea to voxelize the input 3D data. Riegler at al. [154] proposed a technique to deal with the 3D data in a novel way to capture the shape descriptors by expanding the idea of octtrees to the 3D meshes. Due to the sparsity of 3D data, such as point clouds and meshes, converting them into the volumetric data, as many previous works did, is memory demanding. Besides, processing such volume of data is time consuming. This motivated the researchers to propose a new method that only focuses on the boundaries of the objects rather than the entire volume. With this technique, the convolutional networks are easily applicable to 3D data and significant amount of memory and computational time are saved. In this approach, called OctNet, a voxel grid is constructed for the input data through dividing the entire shape into octants, iteratively. Partitioning the space into smaller cells is done adaptively by considering a criterion. For example, the cells which overlap the boundary of the shape, are divided further into another level of smaller cells. Therefore, the cells are denser on the boundary and more sparse anywhere else. Such a volumetric structure (some local geometry and texture features are extracted for each vertex) is fed into a deep convolutional network consisting of convolution, pooling, and unpooling layers. In fact, the architecture of this network simulates a set of an encoder and a decoder such that the convolution and the pooling layers form the encoder and the unpooling (e. g., an interpolation implemented using the nearest neighbor) layer forms the decoder function. In such a structure, the results of the encoder step are concatenated to form shape descriptors which are later on used in a shape classification application.

In a similar approach to the method proposed by Riegler at al. [154], Wang et al. [151] suggested to apply convolutional networks on high-resolution octtree-based data. The data includes features such as averaged normal vectors computed at leaf octants for 3D unstructured data. Their approach, called O-CNN, focuses on the boundaries of 3D data and generates high-resolution representation being able to capture fine details of shapes. In their proposed technique, convolution operation is applied on the octtree data, followed by a max-pooling layer, iteratively. Finally, a softmax layer is used to apply the suggested O-CNN in a classification task. They suggested to use the result of the classification step, as a shape representation to query the dataset for shape retrieval applications. In fact, the O-CNN generates a vector for each input shape which represents the probability of the shape being a member of a particular category. Such vectors serve as global shape descriptors for inputs.

Autoencoder-based

Autoencoders are one of the frequently used deep learning algorithms because of featured characteristics such as being able to capture non-linear feature spaces and stability against the missing values [155]. Autoencoders are categorized under the unsupervised learning algorithms [85] because no class label is provided to them for training. However, most of the techniques discussed under this sub-category modified the target output of the autoencoders such that a single target (instead of the input itself) is provided to the autoencoder, making it rather a discriminative model. This type of encoder is called many-to-one encoder [8].

Fang et al. [8] used a many-to-one autoencoder to construct a deep shape descriptor for shape retrieval. In their method, two shape features namely HKS and Heat shape descriptors (HeatSD) are extracted. The HeatSD, a global descriptor, is developed using a multi-scale HKS by computing the distribution of HKS values at all vertices and over all scales. HeatSD features are used as inputs for two encoder modules. Eigen-shape and Fisher-shape descriptors are target outputs for the two modules. The networks are trained using Eigen-shape and Fisher-shape descriptors to minimize the intra-class variance and maximize the inter-class distance. Target Eigen-shapes and Fisher-shapes are obtained from a collection of the HeatSDs using Principal Component Analysis (PCA) [156] and Linear Discriminant Analysis (LDA) [157] techniques. In the end, hidden layer nodes of the neural networks are concatenated to each other forming a deep shape descriptor. The general schema of their proposed system is demonstrated in Figure 2.7. The comparison between the proposed deep shape descriptor and the ShapeGoogle [48] using the precision-recall curve, shows that the former outperforms the latter. Furthermore, it works well for the partial shape retrieval and is robust against some levels of noise.

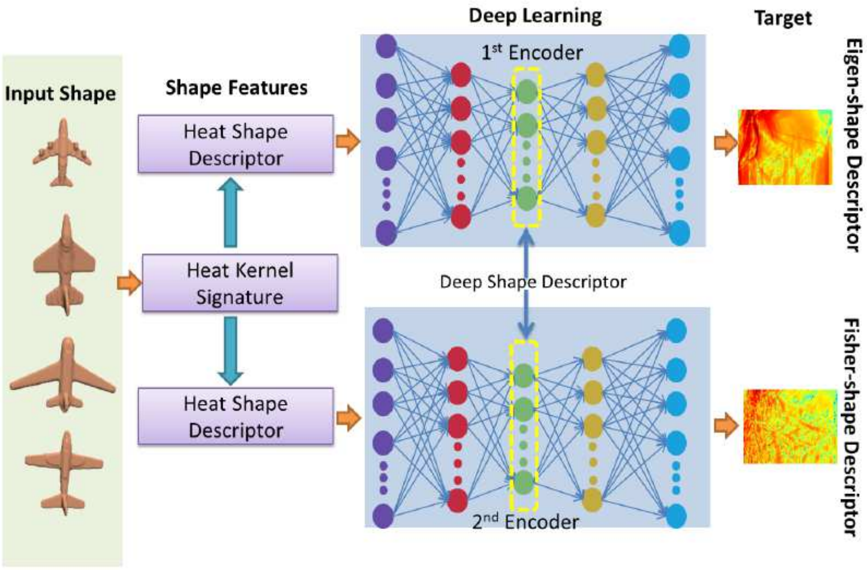


Figure 2.7: Framework of the autoencoder based descriptor proposed by Fang et al. [8]. The figure is taken from [8].

In a similar approach, Xie et al. [158] used a discriminative autoencoder to form the 3D shape descriptors. In their system, many multi-scale shape distributions at different scales are created for each shape in classes. The HKS are obtained for each shape at each time step. Then, the shape distribution at time t is formed by calculating the histogram of the HKSs on N vertices of the shape at time t (known as scale t). A discriminative autoencoder is established for each scale using the Fischer discrimination criterion. The nodes of the hidden layers of the autoencoders are concatenated to form the 3D shape descriptor.

As opposed to the previous works that spectral features were extracted out of input 3D shapes, Zhu et al. [74] proposed a global view-based descriptor for 3D shapes using an autoencoder whose weights are initialized by DBN. They generated multiple 2D depth images for the 3D models. The images are directly fed to an autoencoder-based learner without forming any feature vectors. Similar to previous approaches, the autoencoder generates a compact representation for the model in the hidden layer. To improve the performance, the researchers used DBNs to initialize the weights of the autoencoder and then used BP for fine-tuning. The autoencoder, constructs shape descriptors by encoding each depth image into a lower-dimension vector. Furthermore, in another experiment they constructed different feature descriptors using the SIFT [138] algorithm and BoF technique, to be used along with their global shape descriptor. They emphasized, more local information is captured by the second descriptor because of using the SIFT algorithm. They suggested to combine the two descriptors by computing their linear combination, to improve the retrieval performance.

Many-to-one autoencoders are also applied on 3D data in a recent feature-based approach. Dai et al. extracted global shape descriptors computed using the SI-HKS features of each vertex for different 3D shapes [159]. The global descriptors computed for shapes of each class are presented to the separate autoencoders. The deformation-invariant shape descriptors are learned in hidden layer of the auto-encoder. Comparing the mean average precision of the proposed approach on different datasets proved it to be superior to some of the existing supervised and unsupervised algorithms.

2.5 Discussion

In this chapter we reviewed the data-driven 3D shape descriptors. In Section 2.4 the descriptors were surveyed in accordance with the proposed taxonomy in Figure 2.3. This taxonomy is discussed under Section 2.5.1. We propose two other possible classifications for the reviewed papers in Section 2.5.2. Also, Section 2.5.3 lists limitations of using data-driven techniques in constructing 3D shape descriptors.

2.5.1 Algorithm-Oriented Categorization

The learning type and architecture are key features that provide a good overview of any learning algorithm. In Figure 2.3, we categorized the existing data-driven 3D shape descriptors based on the learning architectures of the algorithms which was used to construct the descriptors. We also used learning types to group the works by color. Therefore, we divided all of the descriptors into two major categories (deep and shallow), accordingly. In the next level, the subcategories were suggested based on the learning algorithms which were commonly used in each broad category.

Although the above mentioned categorization rules seem straightforward, assigning the proper category was challenging for some descriptors, mostly because they were calculated using a combination of different techniques. For several descriptors, we studied how the learning algorithms were combined to form the descriptors. In more detail, to assign descriptors to most relevant categories, we considered which part of a proposed framework played a key role in constructing the 3D shape descriptor or which one of the used algorithms made the most prominent part of the pipeline. For example, Zhu et al. suggested a combination of DBN and autoencoder to form a global shape descriptor [74]. They also proposed another shape descriptor in BoF framework and used a linear combination of the two to form a final shape descriptors. This work has been categorized under autoencoder-based descriptors (disregarding the usage of BoF technique and the DBN) as this deep algorithm was the focus of their proposed method. Another example, is the work proposed in [6]. Even though BoF has been used along with the CCRBM, we listed the technique under the DBN-based category (rather than clustering-based group which is inferred by usage of the BoF technique), since BoF was merely employed to evaluate the proposed local descriptor in shape retrieval applications. The above examples show that there is not a solid boundary between different branches of the proposed taxonomy.

Table 2.2 summarizes the reviewed data-driven 3D shape descriptors in terms of different criteria. The following items discuss the different columns of the table in more detail:

1. *Feature-based*: If the algorithms extracted the low-level or middle-level features out of the original input data and fed these features to the learning algorithms rather than the original data itself, we marked them as the feature-based techniques.

2. *Feature level (Low-level/ Middle-level)*: If a technique was indicated as the feature-based technique, this column is filled out specifying which type of features were extracted from the input data to be used as the input of the learning algorithm.

3. *Learning algorithm*: Various machine learning algorithms such as deep learning algorithms, BoF, clustering, etc. were employed to construct data-driven 3D shape descriptors in the literature. The learning algorithm used by each work to construct the shape descriptor, is listed under this column in Table 2.2.

4. *Domain (spectral/spatial)*: Spectral methods such as HKS and WKS have had successful applications in the literature to compute both local and global descriptors (Section 2.2.2). If a proposed work used spectral methods to compute shape descriptors, we marked it as the spectral otherwise it was considered as a spatial method. Besides, there were techniques that rather than extracting the local spectral features and using them as the input to the learning algorithm, transformed the input data from spatial to spectral domain and then fed this data into their learning algorithms (e. g., [37, 7, 142] etc.). These approaches are also marked as spectral descriptors.

5. *Application*: Shape descriptors have been applied to a wide variety of shape analysis tasks such as establishing point-to-point correspondence, shape matching, mesh labeling and segmentation, shape retrieval, shape detection and recognition, and shape classification. We listed the applications for each work under this column in Table 2.2.

6. *Dataset*: The datasets used for each algorithm are listed under this column. The data types and the objects included in each dataset were reviewed in Section 2.3.

7. *Data type*: This column lists the data type used by the algorithms to construct the descriptors.

8. *Type*: The type of descriptors would be either local or global. If a method captures local features of input shape it is considered as local. The methods that consider the shape as a whole and ignore the local features, would generate global descriptors.

9. *Learning Type (LT)*: This column lists the type of learning algorithm used for constructing the descriptors. The learning type is determined as supervised or unsupervised. For the semi-supervised applications or the descriptors that were built using both supervised and unsupervised algorithms, we marked both columns under the learning type.

Moreover, advantages and limitations of all the reviewed articles are discussed in Table 2.3 to Table 2.7, where each table belongs to a subcategory of descriptors.

2.5.2 Alternative Taxonomies of 3D Shape Descriptors

Some aspects of the systems that build data-driven shape descriptors, such as input data type and application, are not included in the algorithm-oriented taxonomy because it only considers the learning algorithm used to build the descriptors. Learning algorithm, input data type, and application are three important attributes of such systems. Using any of these fundamental attributes as a criterion for categorization potentially provides a great perspective for the researchers who want to establish a new data-driven system to build 3D shape descriptors. Therefore, we provide two alternative taxonomies based on the input data types of the algorithms and the applications the descriptors were employed for.

Input-oriented taxonomy

The input type has a profound influence on the success of different algorithms for extraction of 3D shape descriptors. For example, applying some of the approaches on 3D meshes is not as straightforward as applying them on 2D images because of the unstructured nature of the polygon meshes. Therefore, it is important to present a taxonomy of the data-driven shape descriptors based on their input data type.

Figure 2.8 demonstrates the input-oriented categorization. We divide all of the existing data-driven shape descriptors into three broad categories, namely: view-based, geometry images, volumetric, and model-based groups.

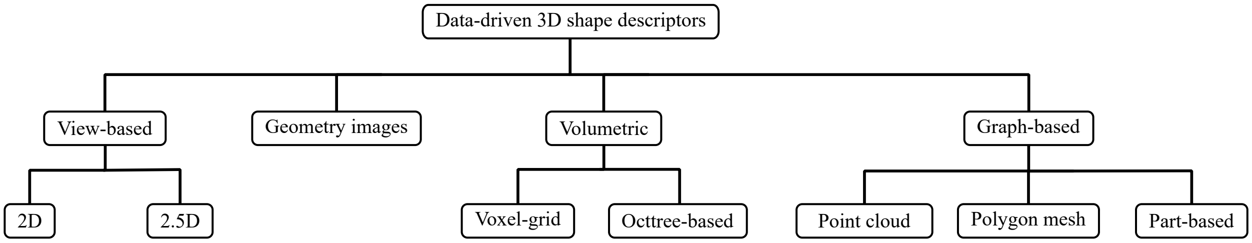


Figure 2.8: An alternative taxonomy for data-driven 3D shape descriptors by considering the input data types.

View-based In these techniques usually a geodesic sphere is considered surrounding the shape. Then RGB/gray or depth images, or a combination of both are taken from a given set of angles. In fact, the 3D shape is represented by a sequence of 2D images. In most applications the descriptors were computed for each image individually and then they were combined to form a global shape descriptor.

Using view-based techniques to built 3D shape descriptors have been a prominent approach. It is because progressive 2D image descriptors could be invoked easily [143] and machine learning and more specifically deep learning algorithms could be successfully applied on 2D images. Also, the shape descriptors which are made using 2D images are fairly compact and persistent against the perturbations (e.g., noise in the surfaces or tessellations [143]). Besides, a large volume of 2D datasets (much higher than the existing 3D datasets) are available, that makes the model training more accurate and precise [178, 143]. Due to the unstructured nature of the 3D shape models, applying deep learning techniques directly on them is inconvenient [6]. Therefore, several effective applications suggested projecting 3D shapes on 2D images to convert the irregular format of 3D shapes into regular grid-like structures [6]. This helps the researchers to use the descriptors which are introduced for 2D images in the literature rather than applying the learning algorithms directly on 3D data.

On the other hand, the researchers would face with some challenges while using the view-based approaches. For example, the number of 2D views rendered from a 3D model must be large enough so that the entire shape surface is captured as accurate as possible [178]. Usually, these approaches (and the volumetric representation which will come later) miss some levels of details during the process of transforming 3D shapes into 2D views [148]. Besides, different parts of a shape may be appeared in more than one view, therefore, an effective technique needs to be taken to unify the parts specifically [178]. Also, as view-based and volumetric techniques handle the geometric properties in the Euclidean representation which is not intrinsic, their analysis miss significant level of details of 3D models. This issue has been elaborated clearly by Bronstein et al. in [148].

The view-based approaches typically generate global shape descriptors that are used for shape retrieval, classification, and recognition. These methods mainly take average over all of the descriptors obtained for each view or concatenate them to form a single 3D shape descriptor. However, some techniques were proposed in the literature to combine the descriptors more elegantly such as the approach proposed in [143].

Geometry images Geometry images are another structured 2D representations suggested in the literature for 3D shapes to convert their unstructured formats to grid-like representations. Using geometry images enables analyzing isometric deformations of 3D shapes which is not possible with view-based techniques. On the other hand, geometry images are confined with some limitations. For example, computing geometry images for shapes of non genus-0 (shapes with holes) is not easy, and for surfaces of high genus can constitute difficulties. In addition, they are not applicable to non-manifold geometry [179].

Volumetric Volumetric techniques take the input data in form of voxelized (voxel-grid) models. Using the volumetric data to construct shape descriptors, is suggested by researchers in the literature as another approach of transforming the unstructured 3D data into structured representation. However, the main disadvantage of this approach is that the voxel-grid data needs much larger storage with respect to the original data [150]. Besides, this extra volume implies excess required processing resources [43]. Some novel approaches in the literature suggested using octtrees for generating the volumetric representation which reduces the storage requirement significantly by focusing on boundaries of 3D shapes (e.g., [154, 151]).

Graph-based The graph-based techniques used the irregular 3D models to compute data-driven shape descriptors. Majority of the available 3D databases (summarized in Table 2.1), provide the data in the form of point cloud or polygon mesh. Therefore, the main advantage of the graph-based techniques is that no data conversion is required and the raw irregular 3D data could be fed to learning algorithms as they are. On the other hand, applying some of the techniques such as convolutional approaches on these data types is tricky and requires either to modify and customize the definition of convolution operation or to convert 3D data into grid-like structures. This is mainly because, the concept of order is unclear in 3D data as opposed to the grid-like data types. Also, the limited number of available training 3D data for deep learning [74], makes these algorithms less feasible for constructing data-driven 3D shape descriptors.

To the best of our knowledge, none of the proposed graph-based 3D shape descriptors (except the work proposed in [38]) considered both photometric and geometric information to form the shape descriptors.

In addition to the point cloud and polygon mesh data types, we also considered a part-based data representation under the graph-based group. The part-based representation, characterizes a shape in an abstract format focusing on its hierarchical structure of parts.

Even though the reviewed data-driven shape descriptors did not use this type of representation, some recent applications such as [180, 181] used deep learning algorithms to learn such representations for different objects. The part-based representation provides a succinct explanation for 3D objects to understand different components of a shape which could be used for object decomposition, shape understanding, shape interpolation, component matching, etc. [180, 181].

Application-oriented taxonomy

In a different point of view, we classify the 3D shape descriptors into two main categories, namely: global and local descriptors which have particular applications in shape analysis tasks. This categorization is demonstrated in Figure 2.9.

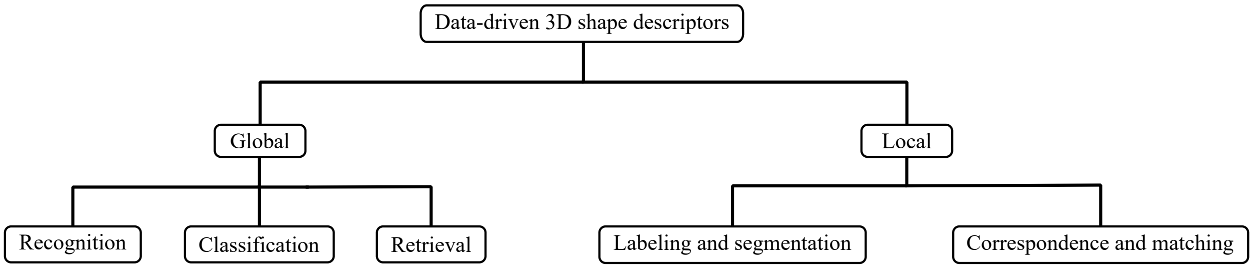

Data-driven 3D shape descriptors
Global
Local
Recognition
Classification
Retrieval
Labeling and segmentation
Correspondence and matching

Figure 2.9: An alternative taxonomy for data-driven 3D shape descriptors by considering the application.

The application-oriented categorization would be useful for future researchers to know which applications the different data-driven shape descriptors could be efficient in.

Global shape descriptors were mainly used for shape recognition, classification, and retrieval purposes, while the local ones were mostly applied to shape segmentation and labeling, point-to-point correspondence, and shape matching tasks. To our best understanding, all of the local descriptors from literature (that are reviewed in this work) used the 3D data representation (including point clouds, meshes, and volumes), and were mostly implemented using deep learning algorithms. On the other hand, the global descriptors were built using all types of input data. In addition, variety of learning algorithms were used to implement the global descriptors.

Based on the application, global or local features or a combination of both could be used. For shape retrieval, the global characteristics of shape models would be able to provide us with a concise shape descriptor, no matter what class the input data belongs to (e.g., animals, objects, planes, cars). On the other hand, if the classes of the input models are the same such as looking in a dataset for human heads or human body, exploiting both global and local features would be beneficial to distinguish between the different objects, more efficiently.

Even though the proposed algorithms were designed to provide the descriptors with a high level of discriminating property to distinguish between fine details of input data, we did not find any shape processing applications on human faces which includes a lot of details (except the work proposed by [38] for learning correspondence between human heads).

2.5.3 Limitations

This subsection discusses some limitations of using data-driven techniques for constructing 3D shape descriptors. These limitations mostly pertains to any data-driven application in general; however, we discuss them particularly with respect to the applications of constructing 3D shape descriptors.

Since the availability of large datasets makes an essential contribution in success of data-driven techniques, large and information-rich training data should be available so that learning algorithms capture 3D shape descriptors from the examples, accurately and precisely. Moreover, supervised learning algorithms in general, require precisely selected class labels. The systems that use this type of learning algorithms to construct 3D shape descriptors should provide their systems with proper set of class labels so that a compact and effective shape descriptor can be learned. Such a shape descriptor must be capable of capturing fine details of the shape and should possess the desired characteristics of a shape descriptor such as being efficient [5], intrinsic and invariant to shape deformations [182], etc. In some applications (for instance in constructing local shape descriptors for mesh labeling, segmentation, and point-to-point correspondence [42]) providing the class labels is not straightforward and sometimes entails large volume of manual work [38]. However, in many other applications training data can be generated automatically (e. g., [5]). Overfitting is another problem that could arise while working with learning algorithms, however, it could be prevented by, for example, providing large number of training data [42].

In spite of the above challenges, the potentials of data-driven approaches have proved them to be beneficial and extremely reliable specifically with the recent advances in the deep learning approaches. This encourages researchers in the computer graphic community to exploit these approaches in constructing 3D shape descriptors. We refer the readers to a recent comprehensive survey by Xu et al. [39] for data-driven methods employed for various shape analysis applications such as classification, segmentation, reconstruction, etc.

2.6 Future Directions

This chapter aimed to include the majority of the relevant and state-of-the-art systems for constructing data-driven 3D shape descriptors. We found a large body of intelligent and fully automated descriptors suggested in the literature. We included the exemplary works from a wide variety of applications and discussed samples from each novel group of ideas proposed for constructing data-driven shape descriptors. However, we would like to acknowledge the works we missed in this survey.

There have been numerous techniques suggested in the literature for feature extraction out of 3D data. As the automatic feature extraction is advancing rapidly and has become more popular than hand-crafted techniques, various feature learning methods that use machine learning algorithms could be of great interest and a good future research topic.

Another future direction could be a comparative study that quantitatively evaluates various approaches that combine local descriptors to compute global descriptors that preserve as much local information as possible and are both specific and sensitive.

Another interesting future research area, is to explore the type of features (e. g. low-level versus middle-level features) fed to deep learning algorithms to compute shape descriptors. A comparative study on low-level and high-level features used for learning algorithms in the literature, would be beneficial. Each of the above approaches have some advantages and disadvantages. For instance, using low-level features needs a larger set of training data than utilizing the middle-level features.

Besides, a comprehensive study could be conducted on various shape descriptors to evaluate and compare their robustnesses against noise, incomplete data, and resolution. Furthermore, some recent approaches (e. g. [144]) suggested in the literature moved towards using self-supervised learning approaches. A study on how this learning type improves learning various shape descriptors could be an interesting future research direction.

Table 2.1: Outline of the 3D databases.

Dataset	Data Type	Provided by	date	Data format	total	Annotation	Availability
3D Head database	Images of laser scanned heads	Max-Plank Inst.	1996	.png, .bmp, .jpg, & .obj (for head scans)	200	Synthetic laser-scanned heads without hair from 7 different views plus 5 full 3D head scans.	http://faces.kyb.tuebingen.mpg.de/index.php
PSB	Polygon meshes	Princeton University	2003	.off & .cla	1814	907 for training and 907 for testing	http://shape.cs.princeton.edu/benchmark
NORB	Images	Courant Inst. & New York University	2004	.mat	97200	Stereo image pairs of 50 toys belonging to 5 categories	http://www.cs.nyu.edu/~ylclab/data/norb-v1.0
SCAPE	Polygon meshes	Stanford University & University of California, Santa Cruz	2005		2590	Human models, scans of different people	http://ai.stanford.edu/~drago/Projects/scape/scape.html
TOSCA	Polygon meshes	Part of TOSCA project	2006	.mat & .off	216	High resolution synthetic 3D non-rigid shapes	tosca.cs.technion.ac.il/book/resources_data.html
SHREC	3D object models	Eurographics workshop	2006			3D object retrieval using a common test collection	http://www.shrec.net/
NIST	Polygon meshes	National Institute of Standards and Technology (NIST)	2008	.off	800	Contains of complete 3D models in 40 classes and 20 models in each one	http://www.itl.nist.gov/iad/vug/sharp/index.html
McGill 3D shape benchmark	Polygon meshes	McGill University	2008	.im & .off or .ply	456	10 classes. 20-30 models in each class	http://www.cim.mcgill.ca/~shape/benchMark/.
ESB	Polygon meshes	Purdue University	2008	.stl, .obj, & jpg	867	Meshes of CAD parts classified into a groundtruth classification with two levels of hierarchy. There are three super-classes with sub-classes under them.	https://engineering.purdue.edu/PRECISE/shrec08.
PHS	Polygon meshes	Ohio State University	2010		300	Consists of 300 shapes that are divided into 21 classes of objects such as dogs, horses, airplanes, humans, etc.	
GWSB	Polygon meshes	Sharp group	2010	.off & .skp	3168	Generic models from google 3D warehouse	http://www.itl.nist.gov/iad/vug/sharp/contest/2010/Generic3DWarehouse
RGB-D OD	RGB-D images	University of Washington & Intel	2011		300	Synch & aligned RGB and depth images	https://rgbd-dataset.cs.washington.edu
COSEG	3D shapes	SIAT, hina TAU, Isreal SFU, Canada	2012	.seg	1090	Consists of 11 sets of 3D smooth manifolds of shapes	http://web.siat.ac.cn/~yunhai/ssl/ssd.htm
NYU	RGB & Depth images	New York University	2012	.mat	1449	Introduced to provide a better understanding of the RGB-D scenes, consisting of 35064 distint objects	http://cs.nyu.edu/~silberman/datasets/
FAUST	Polygon meshes	Max Planck Inst.	2014	.ply & .png	300	Real high-resolution non-watertight meshes of human body scans. 10 different subjects from 30 different poses. Divided into train and test categories with 100 and 200 shapes, respectively.	http://faust.is.tue.mpg.de
RueMonge2014	Polygon meshes & 2D images	ETHZ	2014		428	Textured 3D point clouds of Haussmanian style buildings with seven different segment labels	http://www.vision.ee.ethz.ch/~rhayko/paper/eccv2014_riemenschneider_multiviewsemseg/
ModelNet	Polygon meshes	Princeton University	2015	.off	127915	Includes 3D computer graphics CAD models from 662 object categories	http://modelnet.cs.princeton.edu/.
ShapeNet	Polygon mesh	Princeton University, Stanford university, and TTIC	2015	OBJ & MTL	63300	Contains 3D models along with the complete annotations for each object and multiple views	https://shapenet.org/
Part annotation dataset	point cloud	Various universities and institutes	2016	.off	∼16K models of 16 shape categories	Contains a subset of ShapeNet models with semantic annotations added for their parts	http://web.stanford.edu/~ericyi/project_page/part_annotation
Scannet	RGB-D images	Stanford University, Princeton University, and Technical University of Munich	2017	.ply&.sens	2.5M views of ∼1500 scans	The dataset includes the RGB-d images and reconstructed indoor scenes from various offices and apartments which are fully annotated (semantically labeled).	http://www.scan-net.org/

Table 2.2: Summary of data-driven 3D shape descriptors.

Ref	Feature level		Learning algorithm/framework	Domain		Application	Database	Input data type	Type		LT		Feature-based
	Low-level	Middle-level		Spectral	Spatial				Local	Global	Unsupervised	Supervised	
[34]	✓		clustering, decision tree		✓	segmentation, recognition	synthetic data and the range images	2.5D		✓	✓	✓	✓
[38]	✓		SVM		✓	correspondence	heads	mesh		✓		✓	✓
[120]	✓		RBM, DBN		✓	recognition	NORB	2D		✓	✓	✓	✓
[132]	✓		BoF		✓	retrieval, partial matching, segmentation	SHREC'07	mesh		✓	✓		✓
[133]	✓		BoF		✓	retrieval, classification, segmentation	SHREC'07	mesh		✓	✓		✓
[48]	✓		BoF	✓		retrieval	ShapeGoogle	mesh		✓	✓		✓
[130]	✓		metric learning	✓		retrieval	ShapeGoogle	mesh		✓		✓	✓
[78]	-	-	CNN, RNN		✓	classification	RGB-D OD	2.5D		✓		✓	-
[116]	✓		BoF		✓	classification	RGB-D OD	2.5D		✓	✓		✓
[2]	✓		BoF	✓		retrieval	McGill, SHREC'07	mesh		✓	✓		✓
[137]	✓		BoF		✓	retireval	PSB, NSB, GWSB, McGill, SHREC'11	2.5D		✓	✓		✓
[47]		✓	BoF , DBN		✓	classification, retrieval	SHREC'07, McGill	2.5D		✓	✓		✓
[71]	✓		L-BFGS	✓		shape matching	TOSCA	mesh	✓			✓	✓
[5]	✓		metric learning	✓		correspondence	TOSCA, SCAPE	mesh	✓			✓	✓
[45]		✓	BoF, DBN	✓		classification, retrieval	PSB, SHREC'07, SHREC'11	mesh		✓	✓		✓
[3]	✓		BoF	✓		retrieval	ShapeGoogle, SHREC'14	mesh		✓		✓	✓
[117]	-	-	CDBN		✓	retrieval	ModelNet, PSB, NYU	2.5D		✓		✓	-
[42]	✓		CNN		✓	labeling	COSEG	mesh	✓			✓	✓
[44]	-	-	CNN	✓		correspondence, retrieval	FAUST, TOSCA	mesh	✓			✓	-
[37]	-	-	CNN	✓		correspondence	SCAPE, FAUST	mesh	✓			✓	-
[143]	-	-	CNN		✓	recognition	ModelNet	2D		✓		✓	-
[8]	✓		auto-encoder	✓		shape retrieval	ShapeGoogle, McGill	mesh		✓		✓	✓
[158]	✓		auto-encoder	✓		matching, retrieval	McGill, ShapeGoogle	mesh		✓		✓	✓
[46]	✓		Sparse coding	✓		incomplete shape retrieval	PHS, SHREC'15	mesh	✓		✓		✓
[74]	-	-	Autoencoder, BoF		✓	retrieval	PSB, ESB, NTU	2.5D		✓	✓		-
[?]	✓		FCN	✓		correspondence	SCAPE, TOSCA	mesh		✓		✓	✓
[6]	-	-	CCRBM, BoF		✓	correspondence, retrieval	SCAPE, McGill, SHREC'07, SHREC'15	mesh	✓		✓		-
[7]	-	-	CNN	✓		correspondence	SCAPE, FAUST	mesh	✓			✓	-
[149]	-	-	CNN		✓	classification	ModelNet	volume		✓		✓	-
[43]	✓		CNN	✓		classification, retrieval	McGill, SHREC'11, ModelNet	geometry images		✓		✓	✓
[141]	-	-	GAN		✓	classification	ModelNet	volume		✓	✓		-
[159]	✓		Auto-encoder	✓		retrieval	McGill, SHREC'10	volume		✓		✓	✓
[154]	✓		convolutional encoder-decoder		✓	segmentatiotn, classification	ModelNet, Rue-Monge2014	volume	✓			✓	✓
[150]	✓		FCN		✓	classification, segmentation	Part annotation dataset, ModelNet	point cloud	✓	✓		✓	✓
[152]	✓		FCN	✓		classification, segmentation	ModelNet, MNIST, SHREC'15, Scannet	point cloud	✓	✓		✓	✓
[142]	✓		CNN	✓		segmentation, key-point detection	SHREC'14	mesh	✓			✓	✓
[151]	✓		CNN		✓	classification, retrieval, object part segmentation	Part annotation dataset, ModelNet	volume		✓		✓	✓
[144]	-	-	CNN		✓	matching, reconstruction	RGB-D scene images taken from a variety of reconstruction datasets	volume	✓			✓	-
[147]	-	-	CNN		✓	correspondence, segmentation, matching	correspondences generated automatically from part annotation dataset	2D	✓			✓	-
[153]	-	-	CNN		✓	registration	the same as [144]	point cloud	✓			✓	-

Table 2.3: Clustering-based descriptors.

Ref.	Summary	Advantages	Limitations
[34]	A shape representation technique is proposed based on the surface primitives classification of range images.	• The intrinsic shape properties are used in the learning algorithm.	• As the success of the method depends on the segmentation of the object, occlusion could affect the recognition [160]. • Selecting a good patch size is a trade off [34].
[132, 133]	Local descriptors are computed for each segment of input shape, out of which global shape descriptors are learned using the BoF technique.	• Fast. • Efficient for partial matching as the segments are used as words. • Successful for composed objects retrieval.	• The segmentation is unstable with respect to the topology changes [2].
[48]	A spatial BoF (bag of expressions) followed by metric learning to represent the shapes as the binary codes is introduced.	• Invariant to isometric deformation and many types of transformation.	• Not considering photometric information [161].
[2]	The BoF framework is used on 3D meshes for constructing global shape descriptors.	• Fast. • Effective for partial shape retrieval.	• Not a precise matching between the corresponding segments of a partial query [2].
[116]	RGB-D images are used to extract global shape descriptors using the BoF technique. SURF features are used as local features.	• Encodes both texture and depth information into the descriptor.	• The time length of finding the interest points and convolution computations is likely to make the algorithm slow [162].
[137]	Depth images are used for shape retrieval employing MDS embedding, Clock Matching, and BoF. A dictionary is built with randomly selected local feature vectors and SIFT features are extracted out of depth images as local descriptors.	• Good non-rigid shape retrieval.	• Feature descriptors are all put inside a shared bag rather than using separate bags for features of each view [163].
[3]	A supervised BoF is proposed for dictionary learning.	• Invariant to any kind of transformations.	• It could be time consuming [3]. • A simple mean pooling filter is used in the Vector quantization phase [3]. • HKS which is not sensitive enough, is used as the local descriptor [3].
[46]	Sparse dictionary learning is used to describe incomplete shapes.	• Efficient descriptor for incomplete non-rigid shape retrieval.	• The incomplete query shape needs to be connected [46]. • There are some restrictions on boundary region detection [46].

Table 2.4: Optimization-based descriptors.

Ref.	Summary	Advantages	Limitations
[38]	A human head correspondence is computed through a deformation function that maps source points to the target points. Such mapping is suggested to be used as shape descriptors.	• Both geometry and texture information is used.	• A large number of 3D head scans are required with the texture which is not easy to obtain [164, 165].
[130]	A supervised learning of an optimal diffusion kernel is proposed. The diagonal elements of this kernel, are exploited as point descriptors and the histogram of their values is used as a global shape descriptor.	• Scale invariant. • Noise resistant.	• A single frequency response is employed to characterize the diffusion metrics [166].
[5]	A generic local shape descriptor is learned using metric learning.	• Benefits from the advantages of both HKS and WKS.	• The similar and dissimilar training point sets are selected from the same shape which does not allow the invariance across various objects [37].
[71]	A correspondence model is used to learn a set of optimal descriptors using functional maps.	• Generates informative descriptors for non-rigid shape matching. • Obtains good functional subspaces useful for establishing point-to-point correspondence [167].	• Training cost is comparatively high [71].
[4]	A binary global 3D shape descriptor is proposed based on local spectral descriptors.	• The binary descriptors require less memory and are fast for retrieval purposes.	

Table 2.5: Probabilistic models.

Ref.	Summary	Advantages	Limitations
[120]	A global view-based shape descriptor is proposed using DBN formed by stacking RBMs.	• The effective initialization of the DBN helps to avoid local optima [168].	• The initialization step increases the cost of building the DBN model [168].
[47]	A combination of BoF and DBN methods are used on depth images to form a global shape descriptor. Feature points are extracted by SIFT.	• Middle-level features extracted by BoF are used for DBN resulting in less required training data than using low-level features for training the DBN.	• A large number of depth images (i. e., 200) are required for each mesh to train the system.
[117]	Convolutional DBN is used to form global shape descriptors for voxelized models generated from 2.5D images.	• Requires an arbitrary single-view image as input.	• Has a comparatively massive architecture [169]. • Inappropriate for deformable models [44, 6].
[45]	DBN is used to form the global shape descriptors using HKS and AGD as local descriptors.	• Middle-level features are used as the input for deep learning to deal with the unstructured nature of the input mesh. • A small number of training data are required.	• Weak generalization because of using middle-level features (information loss) [45].
[6]	A circle convolutional RBM is proposed for local shape descriptor.	• Automatic feature extraction. • Applicable on irregular data.	• High level of noise or non-manifold surfaces affect the performance of the algorithm [6]. • Selecting the appropriate stride angle is a trade off [6].
[141]	A combination of convolutional networks and GANs are used to learn a representation for volumetric data.	• This technique does not require any segmented shapes or parts given as the training data and the discriminator is trained completely unsupervised.	• The GAN architecture is confined with the resolution of the input data and not able to capture the fine details [170]. • The generator and the discriminator can be easily instable during the training phase which can cause the training not to converge [171].

Table 2.6: CNN-based descriptors.

Ref.	Summary	Advantages	Limitations
[78]	A combination of CNN and RNN is used to form a global descriptor using RGB-D input images. An unsupervised pre-training is used for CNN.	• No supervision is required for feature extraction step. • Fast.	• The depth modality has not been treated as a separate channel. Hence, the method is not modeling in full 3D [117, 172].
[143]	High-level features are learned out of low-level ones to generate a global shape descriptor using CNNs.	• uses the benefit of regular structure of 2D images for unstructured 3D data.	• Depending on the appearance of objects in rendered 2D images, this technique would not work well for sparse and incomplete models [173].
[43]	A novel implementation of geometry images is suggested to convert the irregular 3D shapes models into the grid-like 2D representation to be used as the input for a CNN.	• Constructed descriptors include intrinsic properties of shapes. • Being an extension of deep learning to manifolds operated in the Euclidean domain.	• Transforming 3D objects into the geometry images is challenging as there is no unique way for the transformation [79].
[42]	A local shape descriptor is computed for each triangle of the input mesh. The descriptors are a label vector that include probability of the triangle belonging to each object part.	• Applicable for meshes of various classes. • Simple network structure.	• To work well, the training data must have a small number of label categories [42]. • Ignores the spatial correlation [144]. • Uses synthetic and complete data, only [144].
[144]	A CNN based system is proposed to learn local shape descriptors in a supervised manner. The network is trained in the siamese fashion.	• Uses the massive amount of available 2D datasets to learn 3D shape descriptors.	• Limited training data [147]. • Low resolution of volumes generated at the points of interest [147].
[147]	A siamese architecture is trained to construct local shape descriptors. The system is trained by 2D images taken at different scales and views for each certain point of 3D meshes.	• Point descriptors are learned using a view-based approach that benefits from the large volume of available 2D image datasets [147].	• Viewing configuration is selected heuristically [147].
[44]	Geodesic CNN is applied on Non-Euclidean manifolds to compute the local shape descriptors.	• Generalizable. • Anisotropic. • Could be used for any function defined on the surface (e. g., texture).	• The method works on triangular meshes and may fail on low quality triangulations [37]. • Deciding the topology of the patches is important in succeed of the algorithm [37].
[37]	A generalization of CNN to spectral domain is introduced to form local descriptors.	• Robust to noise. • High level of both specificity and sensitivity	• The proposed class-specific descriptor could only be used for objects of the same or close-to-same classes [37].
[7]	An Anisotropic local spectral shape descriptor is proposed using CNNs.	• Anisotropic and being able to distinguish between symmetric features. • Works for both meshes and point clouds.	• Requires a rich training dataset with given correspondences [7].
[149]	An anisotropic elongated kernel is suggested to apply CNN on the volumetric data and computing global shape descriptors.	• Combines the benefits of	• Computing the convolution

Table 2.7: Autoencoder-based descriptors.

Ref.	Summary	Advantages	Limitations
[8, 158]	Autoencoders are used to construct 3D deep shape descriptors employing HKS local descriptors applied on mesh data.	• Robust to the inconsistencies and large variations within the input meshes.	• Does not assure to generate the maximum margin between the classes of shapes [176].
[74]	DBN is used to initialize the weights of an autoencoder to form global descriptors for 2D depth images. A linear combination of the global descriptors with engineered local descriptors are used to construct final descriptors.	• No supervision is used to extract the global descriptors.	• 3D information is lost due to considering each 2D view individually [177].
[159]	SIHKS local features are extracted for the 3D shapes and are fed to a many-to-one autoencoder to form global shape descriptors.	• Deformation invariant.	

Chapter 3

Facial landmark detection: Knowledge-driven approach

3.1 Introduction

In this chapter we describe the technical aspects of our system which detects the 10 landmarks on the 3D facial models using the expert systems. In fact, the ad-hoc features have been defined (partly based on the anthropometric properties) by the experienced and knowledgeable experts beforehand. A general framework is illustrated in Figure 3.1.

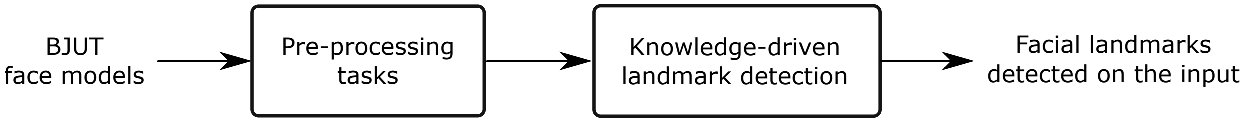


Figure 3.1: The framework of the expert system designed for facial landmark detection. The expert system module consists of a knowledge-driven landmark detection algorithm.

The advantage of using this approach for landmark detection is that we avoid the labor intensive process of the training data generation. We benefited from the simple feature definitions and also there were no need to going through the computationally complex process of training a machine learning algorithm to detect the landmarks. However, we experimented various definitions to encode the features of the mesh salient points into a set of rules to be used by our system.

We believe it is a trade off between where to move to the intelligent landmarking. In more detail, if we choose to continue with expert systems we will save a huge amount of time because of avoiding manual work and time that was spent on gathering a precise training data; but on the other hand we are missing the accuracy.

The organization of this chapter is as follows. Section 3.2 introduces the database which was used to implement the proposed idea. In addition, this section specifies our system in detail. We elaborate the results in Sections 3.3. In the end, Section 3.4 discusses the results.

3.2 Materials and methods

This section begins with an introduction to the BJUT database (Section 3.2.1) which we utilized to experiment our proposed approach and continues with the elaboration of the approach. However, before we describe the main module of the system we demonstrate the pre-processing tasks performed on the data to prepare it as the input for the system in Section 3.2.2. The main module of our system explained in Section 3.2.3 contains of using the knowledge-driven techniques to detect the landmarks on the face models. A set of local and global properties of the faces such as geometric and photometric features, local shape descriptors computed on each vertex of the model, the average length, height, and width of the human faces, etc. have been used to detect the landmarks.

3.2.1 Database

The BJUT-3D Large-Scale Chinese Face Database [183] constructed by The Multimedia and Intelligent Software Technology Beijing Municipal Key Laboratory, Beijing University of Technology, was used for our experiments. This database consists of 500 high-resolution Chinese 3D faces, 250 females and 250 males with frontal views and neutral poses without accessories (e. g., glasses, hats, etc.). The object models consist of both shape and texture data with 65,000 vertices and 130,000 triangles, in average. A set of sample female and male faces are illustrated in Figure 3.2. While working with the dataset we noticed various artifacts such as low-quality tessellation, spikes, and isolated vertices in the meshes particularly in the high-frequency areas (e. g., ears, noses, eyelids, lips). The most noisy part of the faces were the eyes because of the light reflections at the scanning time. On the boundaries also (particularly, close to the neck corners), there have been numerous noisy samples. Figure 3.3 shows samples of these artifacts.

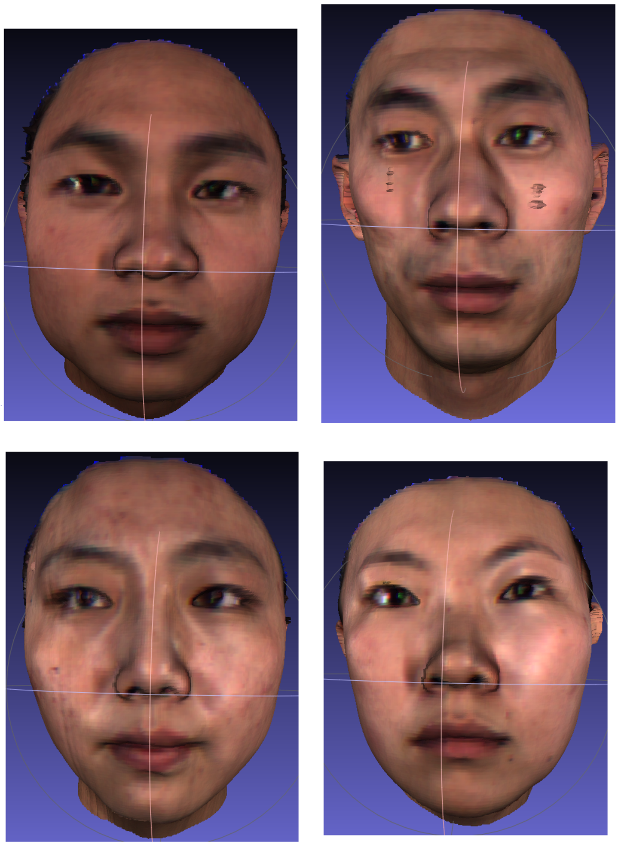


Figure 3.2: Examples of male (first row) and female (second row) 3D face models provided in the BJUT database.

The BJUT database is popular and have been used in various applications in the literature (e. g., [10, 184, 185, 186]) for different 3D face analysis tasks.

3.2.2 Pre-processing and data preparation

Multiple preprocessing algorithms for data preparation are required before proceeding to the main modules of the system. The pre-processing phase includes the following techniques to prepare the input data for the system:

- File type conversion: The 3D models provided in the BJUT dataset were not in text or an standard format (such as (Object File Format) OFF files) readable by most 3D

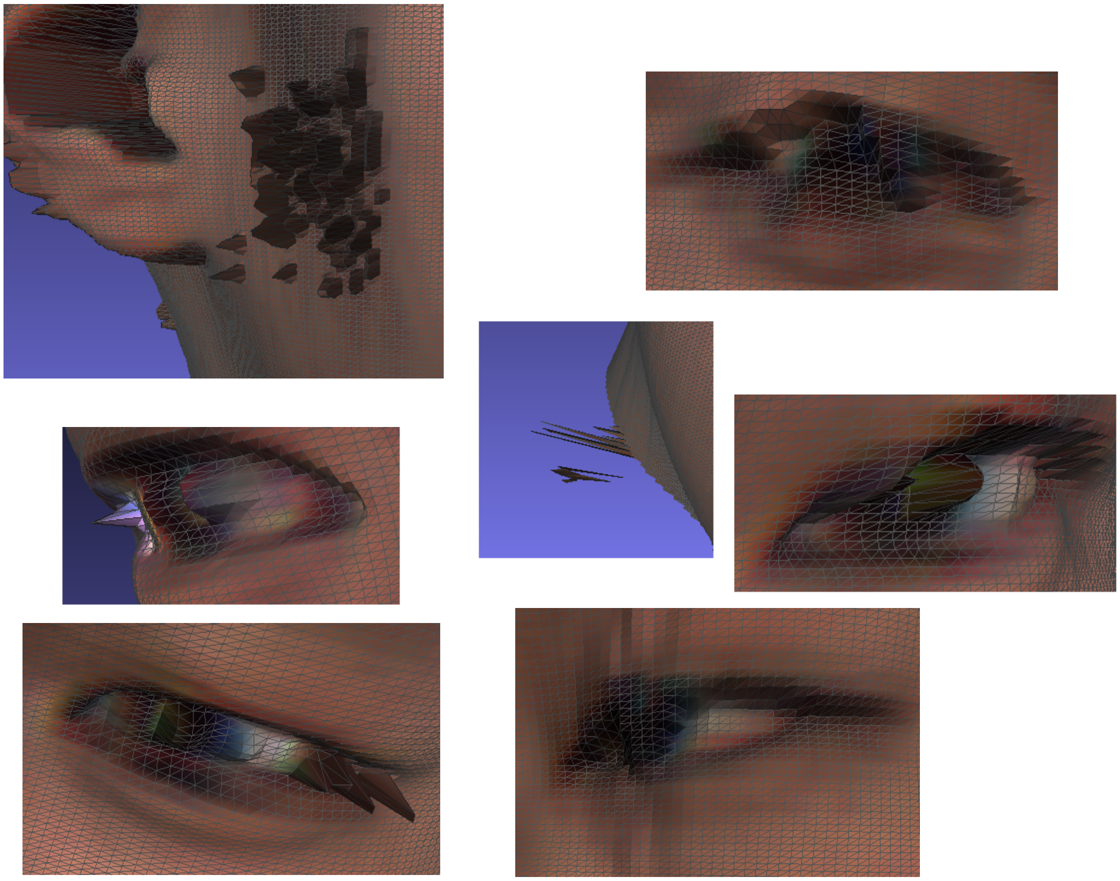


Figure 3.3: Examples of noisy shapes.

visualization software packages. Therefore, the ”FDViewer” a tool provided along with the BJUT database was used to visualize and transfer the data models to readable text files, manually. The files generated by this software were in a predefined text format including a header file that contained information about the institute that conducted the scanning project plus the indicators for vertices, faces, and texture information. We implemented a MATLAB program to read all the text files and convert them into the standard OFF format.

- Scaling: To unify the height of each face, the height, width, and depth were all divided by the height. Therefore, the height of each face is 1 cm. As in most faces the height was the largest value among the other dimensions, the face resides within a bounding cube with the unit side length.

- Smoothing: To alleviate the effect of noise, Taubin mesh smoothing algorithm [187] was applied to the input meshes. We iterated the smoothing algorithm 10 times with the shrinkage factor, λ, and the inflation factor, μ, equal 0.50 and -0.51, respectively.

- Curvature estimation: We performed the curvature analysis for every point of the mesh. Our curvature calculation was based on the well-known curvature estimation algorithm proposed by Rusinkiewicz [188] and the source code was taken from [189]. More precisely, we used the Gaussian curvature which is the multiplication of the two principle curvatures in our work.

- Computing HKS: HKS point signatures [190] were computed and stored for every single point of the 3D models. The program was written in c++ and called from our MATLAB function to compute the HKS for all input shapes. The HKS descriptors were computed using the first smallest 100 eigenvalues and their corresponding eigenfunctions of the LB operator discretized on the 3D face model. The heat diffusion was sampled at 100 time steps. The temporal domain is sampled logarithmically over the time interval $[t_{min} t_{max}]$ while $t_{min} = 4ln10/\lambda_{100}$ and $t_{max} = 4ln10/\lambda_2$ [190].

3.2.3 Landmark detection

This section, explains the expert system module in which a knowledge-driven approach was taken to find the facial landmarks. This module, locates 10 different landmarks on each input face automatically using the predefined rules by an expert. These landmarks are showed in Figure 3.4. Various geometry related metrics and features were exploited for different landmarks. We elaborate the approach used to extract the landmarks in the order they were detected.

1. **Pronasal (tip of the nose):** The tip of the nose is the first salient feature we detected on the faces. As the faces in our database were all taken in frontal view, we

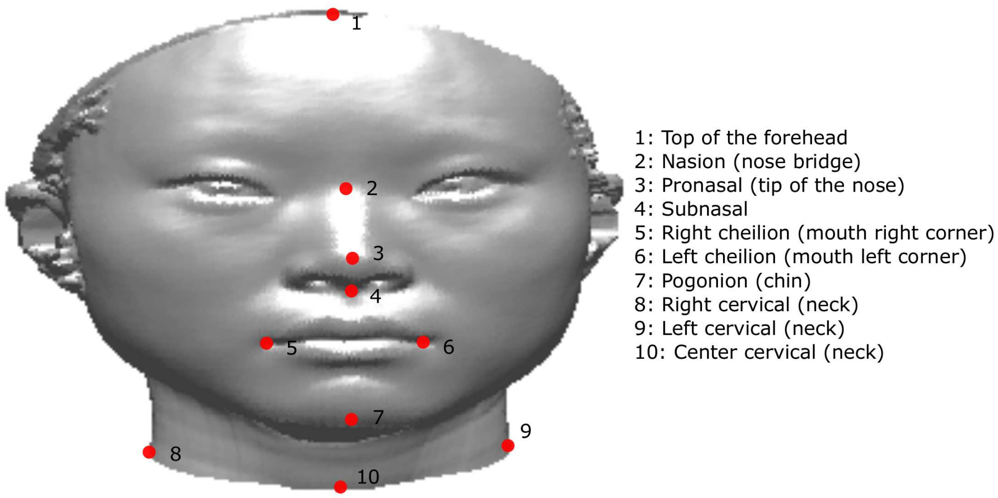


Figure 3.4: The 10 landmarks which are found in the knowledge-driven module.

selected the most common approach used in the literature (e. g. [191]) to detect the nose which is the closest point to the camera location. To improve the result and avoid likely noises in the nose area we selected the closest 20 vertices to the camera and then averaged them to obtain a single point. Then, the closest point on the mesh (out of the selected 20 faces) was selected as the tip of the nose.

2. **Nasion (nose bridge):** There is a concave area above the nose and with almost the same distance from the eyes. To detect this point, we limited our search to the points resided above the detected tip of the nose and their x coordinates were in a distance 0.05 from it. Then, the HKSs were computed for all these candidates. The point which has the smallest HKS at the first time step among the others was selected as the point of interest.

3. **Top of the forehead:** To detect this point, we employed the other two landmarks detected so far. We experimented two approaches to find this landmark. First, to have an estimation of the x coordinate of the top of the forehead we averaged the x values of tip of the nose and the middle of the eyes. A set of candidate points were the ones who resided higher than the nose bridge and within a distance 0.005 of the desired x

value for the top of the forehead. Out of these possible candidates, the point which had the highest z coordinate was selected as the landmark.

The previous approach was not accurate enough particularly for the faces that had noisy boundary around the forehead; therefore, we used an alternative approach which was based on averaging. After finding the estimation of x coordinate the same as the previous approach, a set of candidate points whose x coordinates resided within a distance 0.05 of the estimated x were found. These candidates were sorted descendingly. The top $N = 20$ points were averaged and then a point of the surface which had the closest Euclidean distance to this mean point was selected as the top of the forehead landmark.

4. **Subnasal:** For this point, we restricted our search to an area below the tip of the nose and looked for a point with the least difference between the HKS values at first and second time steps. In more detail, the heat values at the second time step was subtracted from the heat values captured at the first time step and named $heatDiff$. A candidate set of points were obtained by an intersection of the following criteria. The points whose:

   - z coordinates (heights) were below the tip of the nose within a distance 0.07.
   - x coordinates were within a distance 0.005 from the x coordinate of the nose tip.
   - $heatDiff$ values were less than that of the nose tip.

   The candidate with the lowest $heatDiff$ was selected as the bottom of the nose. However, as this criteria did not work very well, we tried another idea in which a mean value is used rather than taking a min values as the landmark to handle the noisy data better. Also, based on the experiments we ran on the 79 faces we decided to widen the local search area to avoid missing the true positive samples. Furthermore, we also tested the Gaussian curvature to be included in our experiment rather than using the heat differences and found it detecting the bottom of the nose landmarks much better;

hence, we disregarded the HKS differences from further consideration and continued with the curvature estimation.

Therefore, the improved criteria was to find a set of candidates whose x coordinates resided withing the distance 0.01 of that of the nose tip and they were located below the tip of the nose in a distance = 0.1 from it. The candidates were narrowed down to get a new subset of points by getting the original candidates sorted ascendingly based on their Gaussian curvature estimations and selecting the points corresponding to the first $N = 10$ smallest estimated curvatures. A point of the surface which had the closest Euclidean distance to the center of gravity of the candidates, was selected as the bottom of the nose.

5. **Cheilions (mouth right and left corners):** To detect these landmarks, we used the texture along with the geometry information. At first, we limited our search to all the points that resided in the front 20% of the face (along y axis which is towards the viewer), between the $d_1 = 0.03$ and $d_2 = 0.2$ distance below the nose tip, and within the $2 \times width/5$ band centered at tip of the nose (please see Figure 3.5).

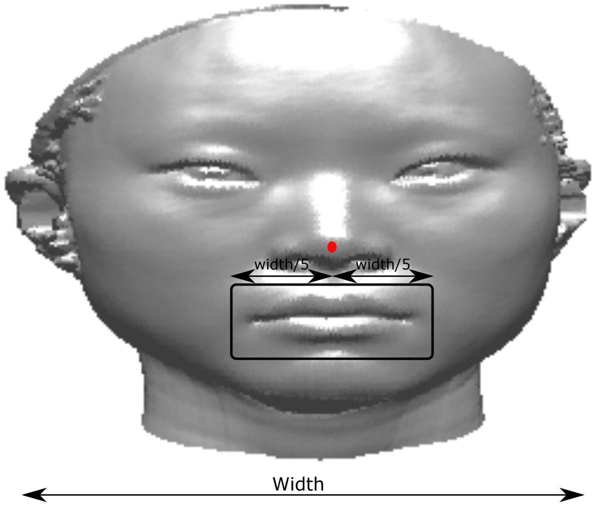


Figure 3.5: The local search area for the lip.

After limiting our search from the geometry point of view, we need to find the vertices that belong to lips considering the texture information. We transformed the color from RGB to HSV space to disregard the brightness and lighting effects and only considering the color. The vertices which their hue values met the following criterion:

$$hue < 0.03 \text{ or } hue > 0.12 \tag{3.1}$$

were considered as the lip candidates. Figure 3.6 illustrates a sample detection of the lips based on the texture. Limiting the search area prior to using the texture information is required. For example, if we use the texture without the restriction, some areas of the skin on the cheeks can result in the false positives because of scars or acnes on the face that have changed the skin color towards the lip color on the area.

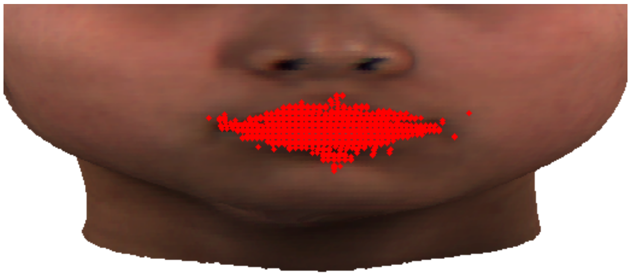


Figure 3.6: A sample detection of the lips based on the texture information.

To find the mouth right and left corners, we limited the search to the most right and most left part of the bounding box. The selection of the amount of the bounding box that should be considered for each corner required a lot of experiments to find the optimum criteria that matched every face with different genders and face proportions. Among the candidates points for each corner, the ones which had the highest Gaussian curvature were detected as the corners.

6. **Cervical (right and left):** To find these landmarks the search is limited to all the points whose z coordinates were below the lower side of the lip bounding cube. For the right and left sides, we limited the search to all the points with an x coordinate greater than or smaller than the 0, respectively. Out of each group of candidates, the

vertex that had the highest value of HKS point signature at the first time step were selected as the neck landmarks.

7. **Pogonion (chin):** To detect the chin landmark, at first we computed a desired x coordinate by taking an average over the x coordinates of top of the forehead, tip of the nose, middle of the eyes, and left and right corners of the lips. We selected a set of candidates whose heights were below the lower side of the lips bounding cube by $dist = 0.1$, whose x coordinates were within a $dist = 0.01$ of the estimated x coordinate, and their depths were within the front half of the face. After constructing the bounding box to limit the search space, we implemented a first approach in which a set of top $N = 20$ vertices with the highest Gaussian curvature among all the candidates were extracted, they were averaged to get a mean point, and then a closest vertex to the computed mean was detected on the surface as the chin landmark. However, our experiments showed that this criterion did not work well for all of the faces.

   The large variations in the chin structures of different faces was the major reason of failing the technique. Besides, we noticed that our criteria missed the normal vector information that could be a distinguished indicator for the chin landmark. Therefore, we revised the above technique by defining a score for the point that resided within the bounding cube of interest. We also defined a normal vector $\mathbf{v} = (0, 0, -1)$ that represents the desired normal vector for the chin landmark. The score s was defined as follows:

$$s = \alpha \mathbf{p} + \beta \mathbf{y} \tag{3.2}$$

   in which p is a vector that holds the dot products of vector $\mathbf{v}$ with the normal vectors of the vertices in the bounding cube, $\mathbf{y}$ represents the y coordinates of the candidate vertices, and α and β represent the contribution coefficients of the normal vector and the y coordinates. In fact, with such a criteria we are looking for the point which its y coordinate is the closest to the camera and at the same time its normal vector is the closest to the vector $\mathbf{v}$. In our experiments, α and β were set to 0.4 and 0.6, respectively. The candidate vertices (resided in the bounding cube of interest) were sorted descendingly based on their s scores, the vertices corresponding to the top $N = 2$ scores were averaged and a mean value was obtained. A vertex on the surface mesh with the closest distance to the mean was selected as the chin landmark.

8. **Center cervical:** An estimation of the desired x coordinate for the middle of the neck was computed by taking average over the x coordinates of top of the forehead, middle of the eyes, tip of the nose, right and left corners of the lips, and the chin landmarks which were detected so far. A set of candidate vertices were formed by the points which their z coordinates were below the chin landmark, their x coordinate were within a distance $d = 0.01$ of the estimated x, and their y coordinates were less that that of the chin landmark. These points were sorted according to their z coordinates, ascendingly. The top $N = 20$ vertices were selected, averages, and the closest point to such mean value was detected as the landmark of interest on the surface mesh.

Rotation invariant nose detection:

Various rotation invariant nose tip detection approaches were presented in the literature. For example, Anuar *et al.* used the morphological approaches being applied on the candidate regions for nose tip that were selected by considering the convexity of the areas. Then, using a trained system with the point signatures computed for the vertices in the areas of interest, they selected a finer region for the nose tip [192]. However, the shortcoming of this approach is that it generates the nose region rather that the certain nose tip. Another application pertains to the work proposed by Guo *et al.* [193]. In this pose invariant approach, a sphere is fitted to every vertex of the face and a statistic value was computed for the sphere. Using a set of training data a proper f values that correspond to tip of the nose were obtained. Therefore, out of all vertices, a vertex which had the closest value to that of the target, was selected as the nose tip. In this heuristic approach, a set of training set was used to find the correct definition of the feature (the tip of nose) by the researchers. Therefore, it has a disadvantage of requiring the expert to make decision for two parameters for the used sphere.

In this section, we propose an improved algorithm for nose tip detection. This algorithm, is invariant with respect to yaw, pitch, and roll rotations and requires two simple thresholds to be selected by an expert. To this end, we used HKS local shape descriptors. We have studied how the heat is diffused over 3D face models for many faces and figured out a specific pattern that governs this diffusion. To study such a diffusion pattern, we created videos of 100 steps that heat is diffused on the face over the 100 sampling steps. As it is shown in Figure 3.7, the initial heat is higher on the boundary of the face.

As the time passes towards the last steps, the heat is diffused from high temperature to low temperature areas such that all points of the face hold an almost equal amount of heat at the end. Somewhere around halfway of the 100 steps, there is a good chance of separating the wide band of boundary from the interior of the surface. This will result in a portion of the face that includes nose and mouth. Please refer to step 50 illustrated in Figure 3.7. By configuring the threshold which is used for this segmentation a wider or smaller area of the center of face could be included in the result. We selected the low 8% temperature from step 50 of faces.

3.3 Experimental results

150 face models (75 males and 75 females) out of the 500 were randomly selected to evaluate our proposed approach. In this section, we report on the software packages and the frameworks we used to implement our proposed approach (Section 3.3.1). We also, represent the pre-processing results in Section 3.3.2. The 10 landmark defined in Section 3.2.3 were detected on all of the 150 faces. In addition, a set of groundtruth landmarks were created on the face models, manually. The Euclidean distance between the detected landmarks and

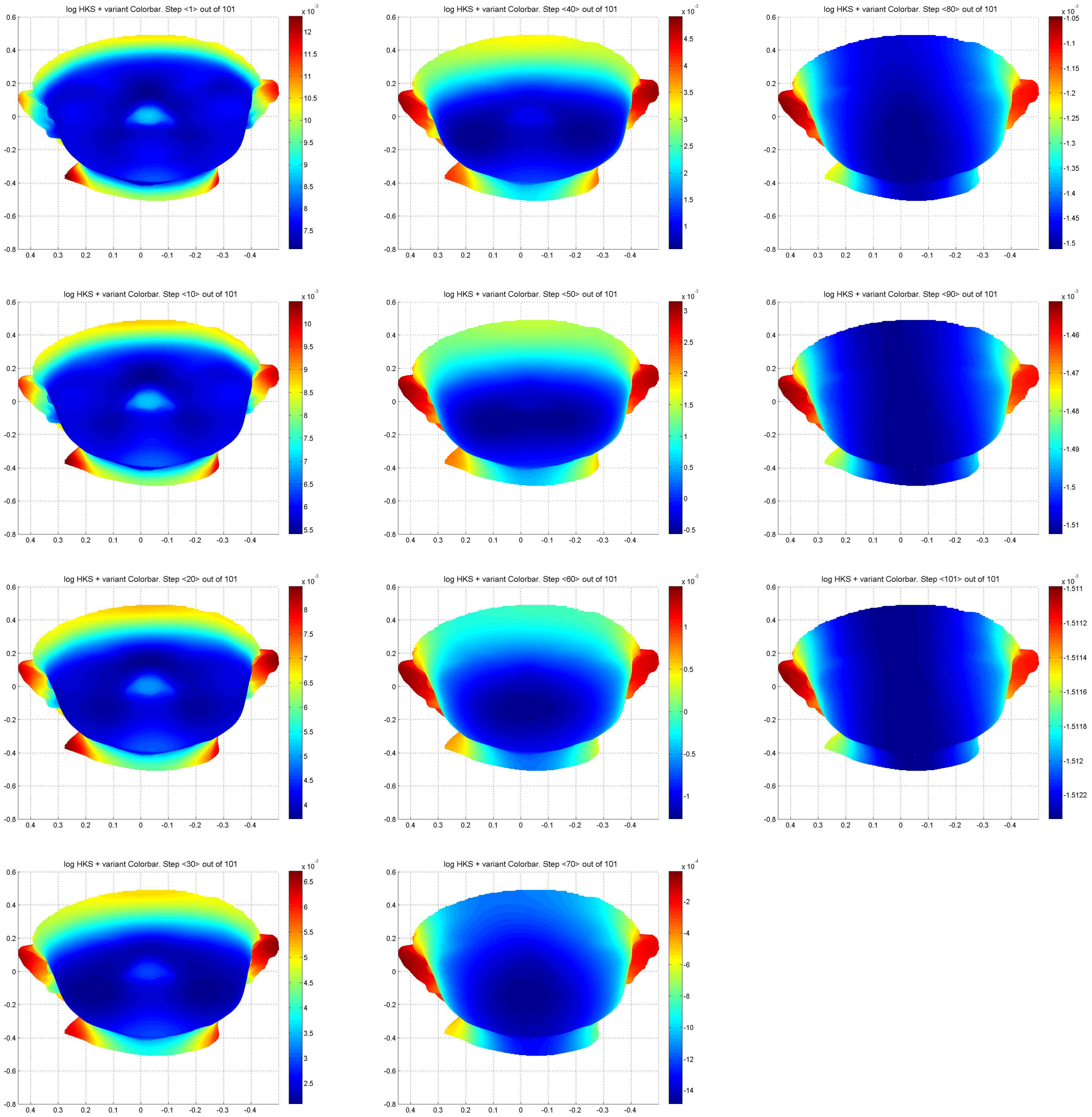


Figure 3.7: Heat diffusion over face models within 100 steps sampled every 10 steps on an example face.

their corresponding groundtruth points are reported in Section 3.3.3.

3.3.1 Experimental setup

Major parts of the pre-processing step such as computing HKS point signatures and the Taubin Smoothing were implemented in C++ programing language. However, the main part of our system is implemented in MATLAB environment. The executable files were built for the above algorithms which were called from within our MATLAB program. We ran all of the experiments on a PC with 64-bit Windows 10 Home Edition and Inter(R) Core(TM)i3 CPU 3.4 GHz, which was supplied with a 8.00 GB of RAM.

3.3.2 Preprocessing results

As it was mentioned earlier, a mesh smoothing technique namely Taubin algorithm was used to alleviate the effect of noise on the data. Figure 3.8 shows an example of how this method improves the mesh quality by enhancing the poor triangulations.

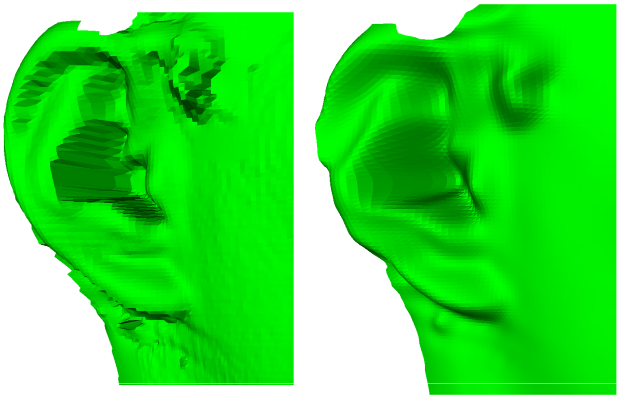


Figure 3.8: A sample result of applying Taubin method. The poor triangulations were improved, significantly. Even though the algorithm is famous in avoiding undesired changes (e. g., blurring, geometry changes, etc.), or blurring in the model, iterating the algorithm for too many times results in missing the salient features of the mesh.

Figure 3.9 also, shows a Gaussian curvature values estimated on a sample face model. The warmer colors show the more concave areas such as the corners of the eyes and lips.

A visualization of the first value of HKS is also provided in Figure 3.10.

3.3.3 Landmark detection results

This section summarizes the experimental results for the facial landmarking to detect the 10 salient feature points introduced in Section 3.2.3. We applied our knowledge-driven technique on 150 faces (75 male and 75 female) drawn from the BJUT database. It is necessary to

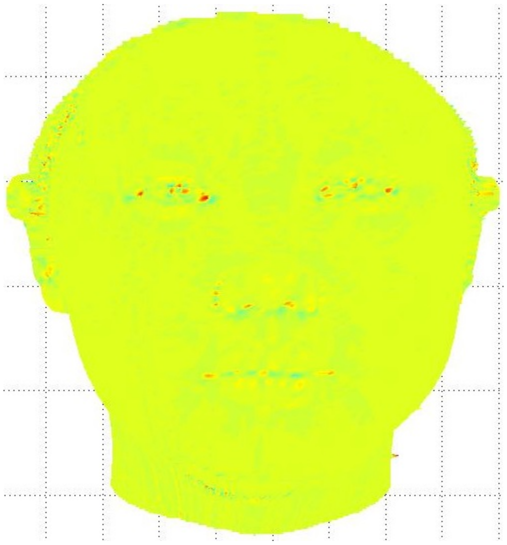


Figure 3.9: Example of the Gaussian curvature visualization on a sample face.

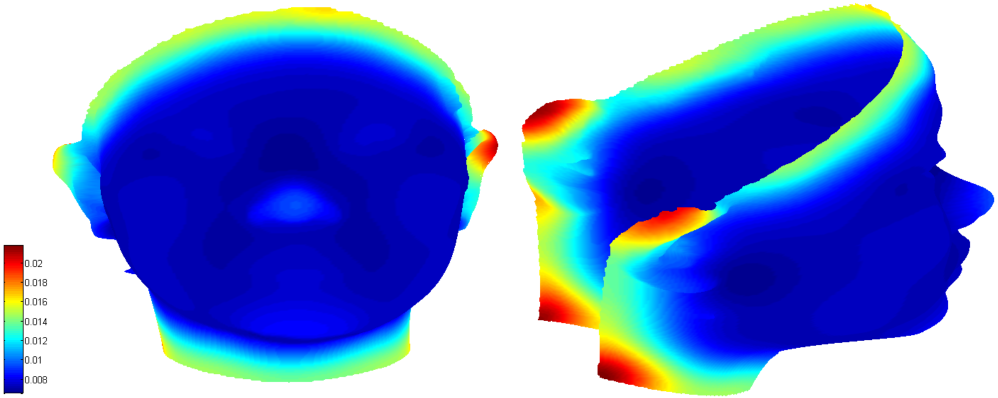


Figure 3.10: Visualization of the first value of the HKS point signature of a sample face model (frontal and side views). This figure shows that how HKS values are salient in some of the landmarks such as the two sides of the neck (colored in dark red) and tip of the nose.

say that for some points of interests such as tip of the nose or chin, the detected landmark illustrates a better point than the ground-truth. This is because selecting the ground-truth particularly for nose tip was not an easy task to do. Therefore, the average distance for those points are worse than actual value. Average distances are shown in Table 3.1.

Sample results obtained using the knowledge-driven approach are demonstrated in Figures 3.11.

Table 3.1: Average distance (localization error) between the detected landmarks and the ground-truth using the knowledge-driven technique on 150 faces of BJUT database.

localization error (mm)	top of the forehead	nose bridge	nose tip	subnasal	mouth right corner	mouth left corner	chin	right cervical	left cervical	center cervical
average	2.60	2.52	2.46	1.46	0.80	0.74	3.72	3.90	3.90	3.43
standard deviation	1.74	1.72	1.50	1.12	2.50	1.94	3.81	12.75	13.14	7.90
min	0	0	0	0	0	0	0	0.40	0.29	0
max	9.10	9.79	7.60	7.89	20.64	17.46	39.95	133.27	136.58	97.02
<= 10 mm (%)	100	100	100	100	97.33	98.67	98.00	90.67	92.00	98.67
<= 20 mm (%)	100	100	100	100	99.33	100	99.33	98.00	96.00	99.33

3.3.4 Rotation invariant nose tip detection results

Sample results are demonstrated in Figure 3.12 for faces undergone random rotations.

Table 3.2 summarizes the results obtained using the rotation invariant technique on all 150 faces in comparison to the original nose tip detection approach.

Table 3.2: Average distance between (localization error) the detected nose tip landmarks and the ground-truth using the rotation invariant and the closest point to the camera approaches on all 150 faces.

localization error(mm)	nose tip detection	
	original	rotation invariant
average	2.46	2.62
standard deviation	1.50	4.92
min	0	0
max	7.60	47.00
<= 10 mm (%)	100	98.67
<= 20 mm (%)	100	98.67

3.4 Discussion

In this chapter we used knowledge-driven techniques to detect 10 different landmarks on 3D human faces. We summarized our results for the proposed method. We also, proposed a

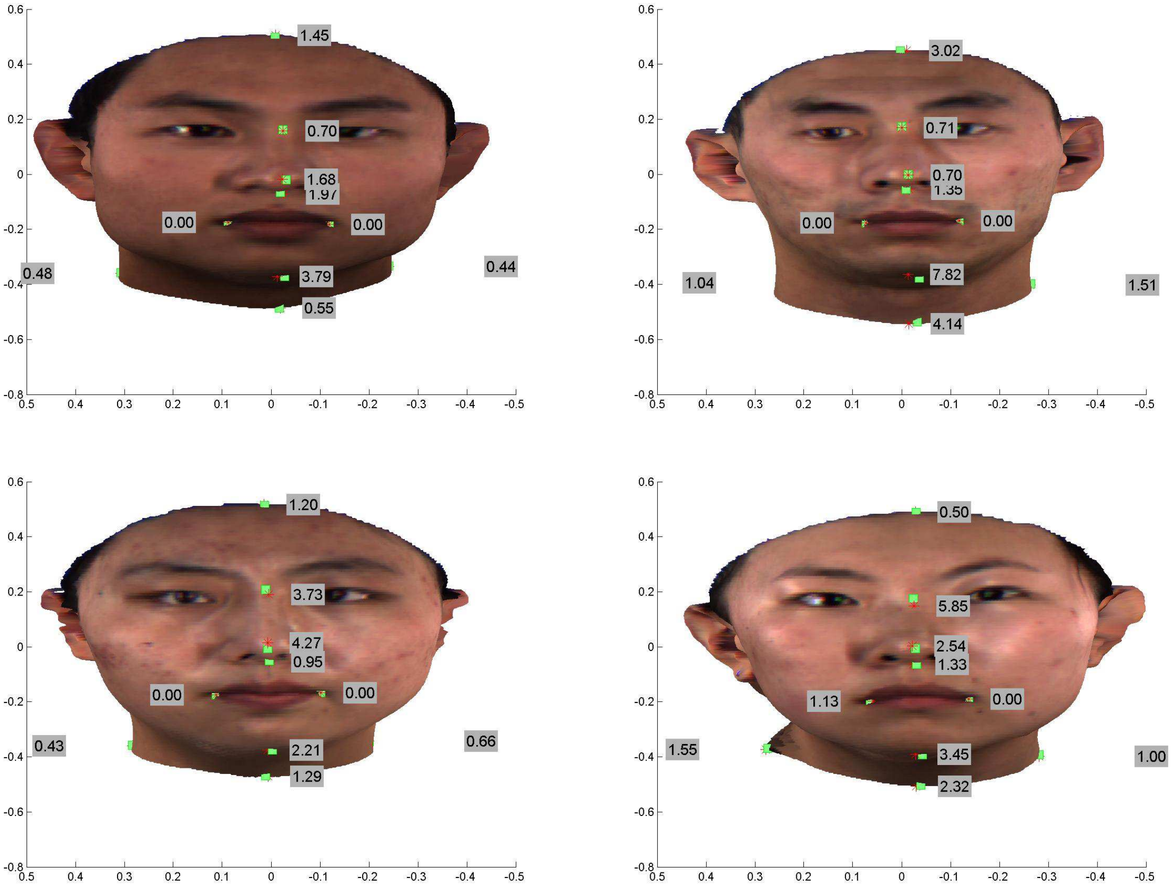


Figure 3.11: Automatically detected landmarks are shown with red asterisks and the ground-truth are shown with green squares on male (first row) and female (second row) sample faces. Numbers in gray boxes show the distance between the detected and ground-truth points in millimeter.

rotation invariant technique to detect the nose tip for faces with arbitrary yaw, pitch, or roll rotations. We compared the results of this technique with the original one that used the closest point to camera as the criterion for nose tip detection. This comparison (Table 3.2) shows that the rotation invariant techniques works very close to the original approach. In fact, we obtained only 2 wrong nose tip detected out of 150 faces. Figure 3.13 shows the two false positive results.

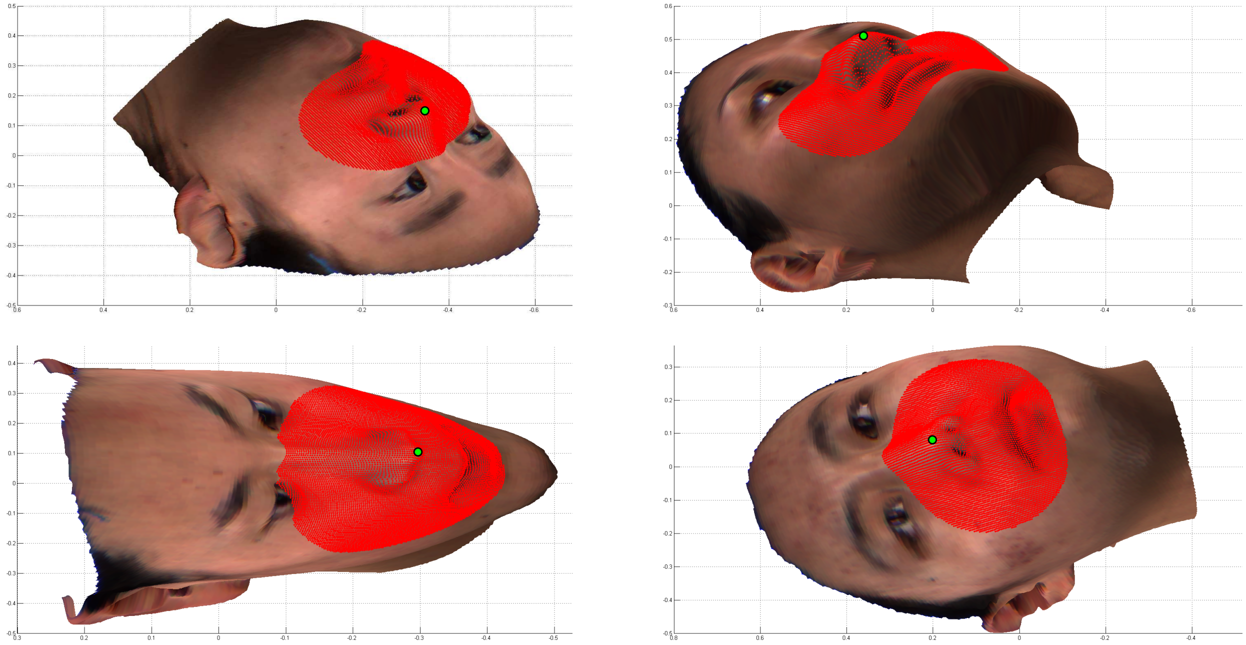


Figure 3.12: Rotation invariant nose tip detection on four sample faces. Red asterisks show the candidates points selected from step 50 of HKSs and the green points show the nose landmark detected by our algorithm. Selection criterion for this point is the maximum HKS value in the first step among all red candidates.

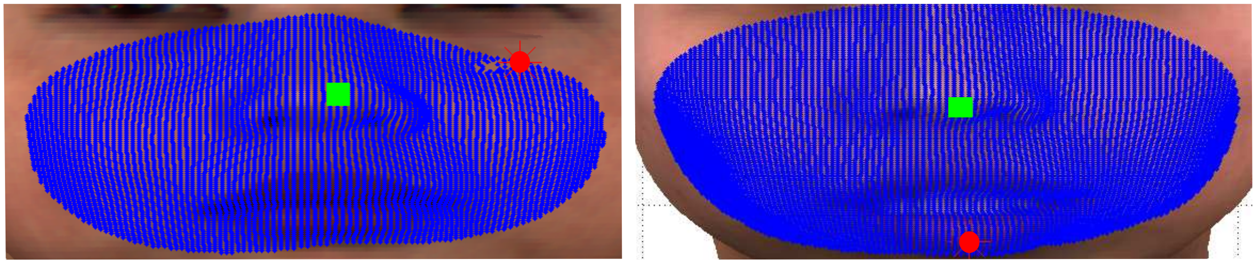


Figure 3.13: Two wrong nose tips detected using the rotation invariant technique. Green squares and red dots indicate the ground-truth and detected nose tips, respectively.

Chapter 4

Facial landmark detection: Data-driven approach

4.1 Introduction

We evaluated the knowledge-driven approach proposed in the previous chapter to detect the 10 landmarks and presented the results. However, we were not able to continue with the traditional approaches to detect more complicated landmarks such as the corners of the eyes. In fact, the major difficulty with the traditional approaches is defining good features which is not always an easy task to accomplish. For example, for tip of the nose it is pretty easy (if the faces have the frontal or near frontal orientations) to detect it by finding the closest point to the camera. But it is not always the case, most times feature definition is not straightforward and several definitions need to be designed, implemented, and evaluated to get the accurate results. The chin and top of the forehead keypoints are examples of this issue in our experiments. Therefore, we chose to find the rest of the landmarks using the machine learning algorithms. In this chapter we express the data-driven approach of detecting 6 landmarks (showed in blue in Figure 4.1).

Data-driven approaches, were proposed in the literature to learn and locate the features automatically without any need to define the rules beforehand. In these approaches, that utilize machine learning algorithms, the examples that own the features of interest are presented to the learning algorithm such that the algorithm learns the features. This is a great improvement that eliminates the need for encoding the features into rules which some times makes a barriers for feature extraction because of the lack of the knowledge. However, this improvement is achieved at the cost of preparing a large set of training instances to the algorithm to learn the rules accurately and precisely. In addition, if the supervised learning type is used, a class label is required for each data sample as well which is not always available and often requires a lot of manual work.

Therefore, selection between the two knowledge-driven and data-driven approaches is always a trade-off. This led us to the idea of combining them together such that for the landmarks which were easy to find we used the traditional approaches to avoid the burden of training data preparation. When it was no longer worth to continue with the knowledge-driven approaches, we switched to the data-driven techniques. This point was where, even with several advanced feature definitions and running experiments repeatedly, we were not able to detect the landmarks of interest accurate enough. Our proposed methodology which detects 6 additional landmarks on the 3D facial model (Figure 4.1), is elaborated in Section 4.2.

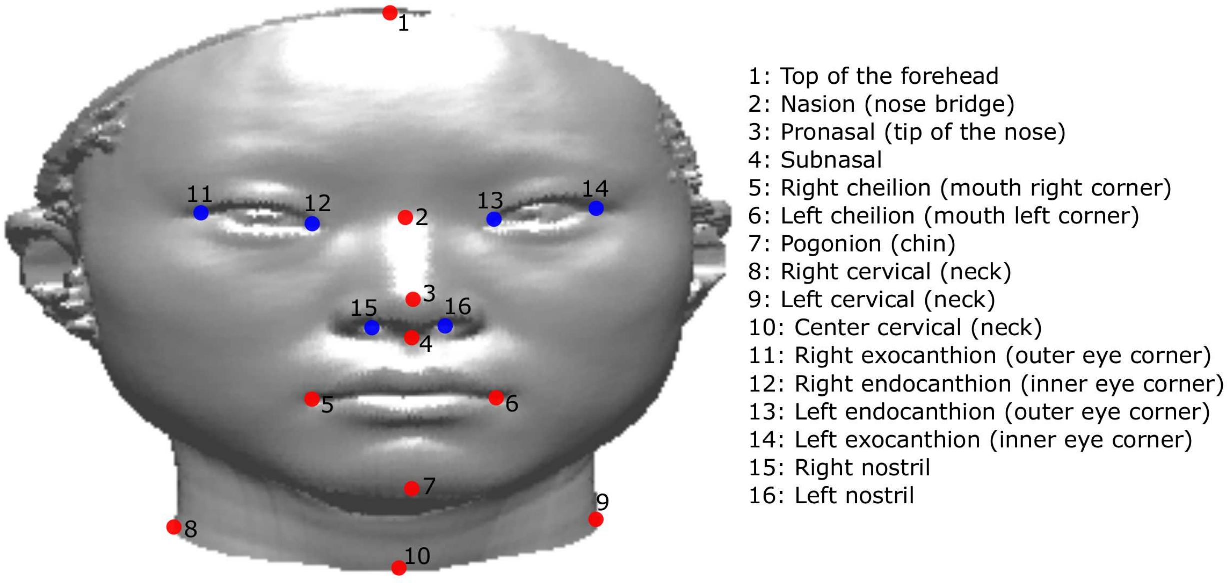

1
11
12
2
13
14
3
15
16
4
5
6
7
9
8
10
1: Top of the forehead
2: Nasion (nose bridge)
3: Pronasal (tip of the nose)
4: Subnasal
5: Right cheilion (mouth right corner)
6: Left cheilion (mouth left corner)
7: Pogonion (chin)
8: Right cervical (neck)
9: Left cervical (neck)
10: Center cervical (neck)
11: Right exocanthion (outer eye corner)
12: Right endocanthion (inner eye corner)
13: Left endocanthion (outer eye corner)
14: Left exocanthion (inner eye corner)
15: Right nostril
16: Left nostril

Figure 4.1: The 16 landmarks being detected on the 3D facial models using the proposed hybrid methodology. The red and blue landmarks are detected using the knowledge-driven (expert system module) and data-driven (machine learning module) parts of the system, respectively.

4.2 Landmark detection

Our proposed hybrid system which detects the landmarks on 3D face models consists of (1) off-line and (2) on-line phases described in Sections 4.2.1 and 4.2.2, respectively. Figure 4.2 shows pipeline of the system.

4.2.1 Off-line phase

The off-line phase includes the following four modules:

1. Ground-truth data generation
2. Semi-supervised clustering setup
3. Training data and local shape descriptor construction
4. Supervised learning

Ground-truth data generation In this step a user selects 16 ground-truth landmarks (demonstrated in Figure 4.1) on the 100 training faces, manually. We iterated this operation twice. Once with the textured faces and once without the texture and only by visualizing the Gaussian curvature. Visualizing the faces with the texture was misleading for the user to select the accurate ground-truth. Therefore, in the second round of capturing the ground-truth keypoints, we guided user by visualizing the curvature. This, resulted in a much accurate ground-truth landmarks (except for the nose tip). Furthermore, we select a reference face from within the database. In future, the training and testing faces will be deformed towards this face for cluster analysis.

Semi-supervised clustering setup This step which itself consists of three sub-steps, configures the semi-supervised clustering algorithm. At first, all the 100 training faces were deformed towards the candidate reference face. We used the manually selected landmarks from the previous step as the control point for the Thin Plate Spline (TPS) algorithm [25, 26] to implement the deformation. This algorithm, received a set of corresponding points on the

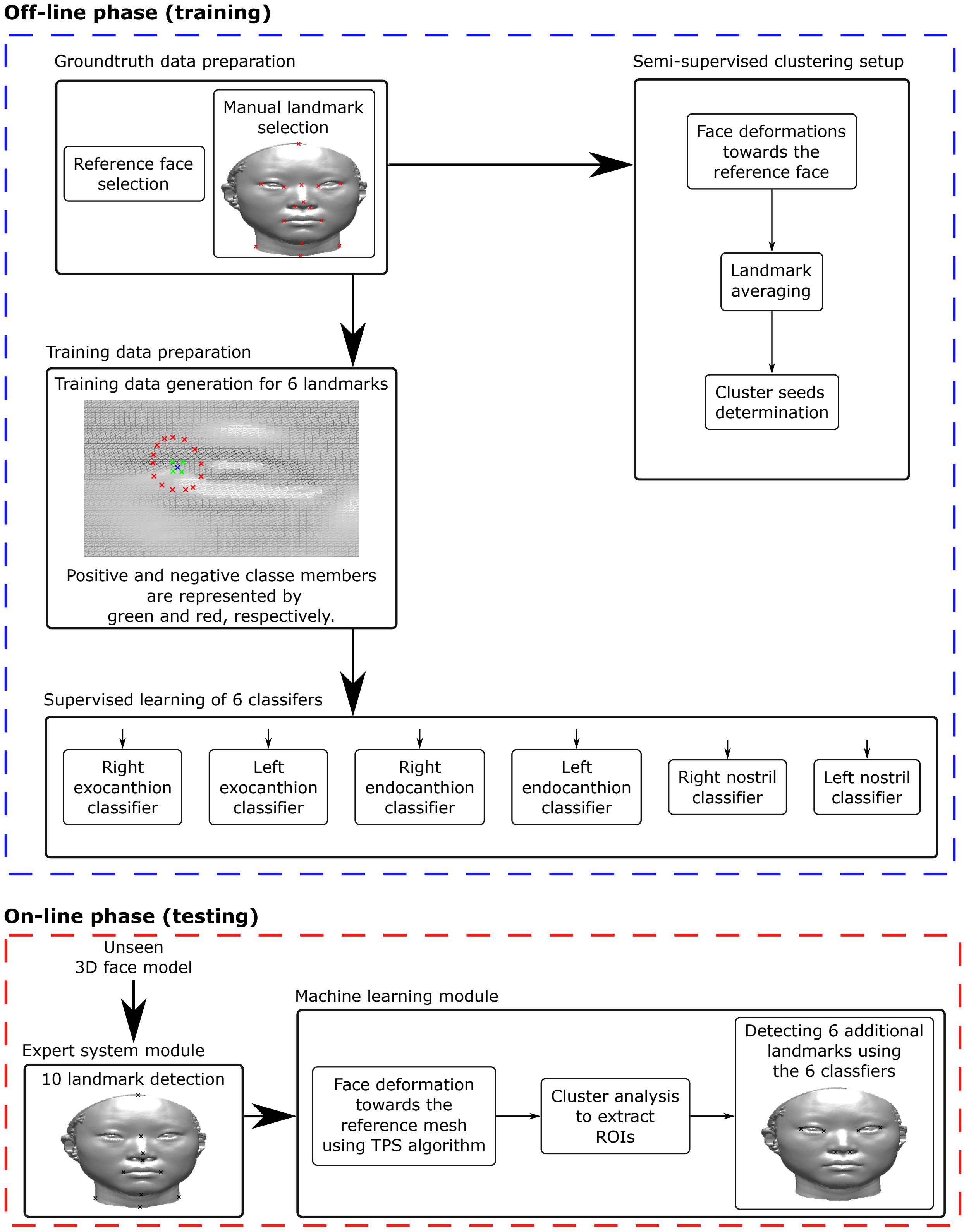

Off-line phase (training)
Groundtruth data preparation
Reference face selection
Manual landmark selection
Semi-supervised clustering setup
Face deformations towards the reference face
Landmark averaging
Cluster seeds determination
Training data preparation
Training data generation for 6 landmarks
Positive and negative classe members are represented by green and red, respectively.
Supervised learning of 6 classifers
Right exocanthion classifier
Left exocanthion classifier
Right endocanthion classifier
Left endocanthion classifier
Right nostril classifier
Left nostril classifier
On-line phase (testing)
Unseen 3D face model
Expert system module
10 landmark detection
Machine learning module
Face deformation towards the reference mesh using TPS algorithm
Cluster analysis to extract ROIs
Detecting 6 additional landmarks using the 6 classfiers

Figure 4.2: The general pipeline of our proposed approach consisting of two main phases.

two source (training meshes) and target (reference mesh) models as input and generated a deformation matrix. This algorithm which is summarized in Equation 4.1, is able to capture both affine and non-affine transformations for every single point of the source mesh.

$$f(P) = Pd + Kw \tag{4.1}$$

In this equation, P is a matrix of the homogeneous coordinates of the points on source mesh, d and w which are obtained through solving the following equation, represent the affine and non-affine transformations, respectively.

$$\begin{bmatrix} K & M_1 \\ M_1^T & 0 \end{bmatrix} \begin{bmatrix} w \\ d \end{bmatrix} = \begin{bmatrix} M_2 \\ 0 \end{bmatrix}$$

In which K is an $M \times M$ matrix where M is the number of control points and M_1 and M_2 represent the coefficients of the control points on source and target meshes, respectively. TPS has been used widely in the literature for face registration [191].

After all faces were transformed towards the target mesh, we took the averages for every landmark over all 100 faces in the deformed space obtaining 6 vectors of length three holding the geometry information of the average points. These mean values, were used in the online phase of the system as the seeds to provide the clustering algorithm with the already computed seeds.

Training data construction In this step we collected the data to train our classifiers following the ideas proposed in [5, 1] on how to prepare the training dataset. In the next substep, we will train six different classifiers for the six landmarks colored in blue in Figure 4.1. In fact, each one of these classifiers will receive a set of relevant training data and will learn a landmark of interest.

The training data is a set of features computed for each vertex in a group of vertices. We refer to this feature vector as the local shape descriptor or point signature. Each point signature was coupled with either a positive or a negative class label indicating whether its associated vertex belonged to the *landmark* or *non-landmark* classes, respectively. By *landmark* class we mean the ground-truth landmark itself and its 4 closest neighbors in terms of Euclidean distance and by *non-landmark* we mean the vertices whose Euclidean distance from the landmark was less that $\alpha \times diameter$ and $\beta \times diameter$ while *diameter* was the longer diameter of the shape bounding box. We selected α and β to be 0.025 and 0.015 for the eye landmarks, respectively. For nostril landmarks we selected a greater set of α and β that is 0.04 and 0.03, respectively. The reason was the particular nose structure in the nostril areas. An example of the training data preparation for the outer corner of the right eye is illustrated in Figure 4.3.

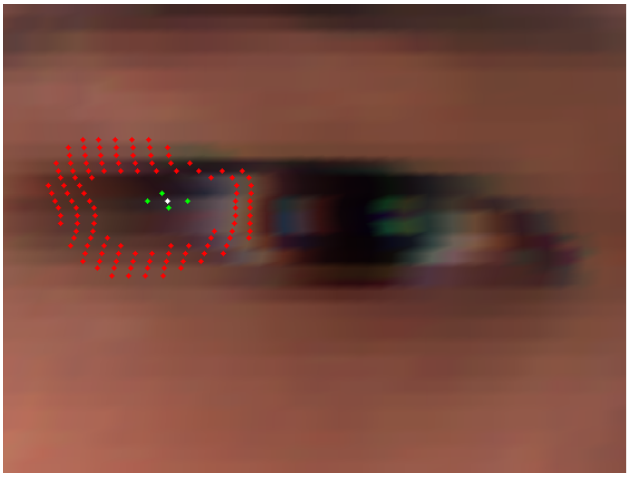


Figure 4.3: An example of selecting the positive (green and white (ground-truth)) and negative (red) class samples for the training dataset.

The training data was collected from every single 100 training face models. For example, to prepare the training set for the classifier which was selected to learn the right eye outer corner landmark we collected the positive and negative training data from all training faces for the right eye corner landmark. This procedure was iterated for all 6 landmarks colored in blue in Figure 4.1.

The point descriptor that we made for each vertex included the following features:

- Hue
- Saturation

- k_1 (Principle Curvature)
- k_2 (Principle Curvature)
- Gaussian curvature
- ShapeIndex

The curvature estimated at a point represents how much the surface has been bended in the area. The k_1 and k_2 represent two principle curvatures estimated at each point. These measures usually are not used alone [1]. Often a new measure which is computed using the two is used as a feature. Gaussian curvature (K) is one of the famous features computed based on the principle curvature as $K = k_1 k_2$. Another feature that we used, was the ShapeIndex which was computed at each vertex as follows:

$$SI = \frac{1}{2} + \frac{1}{\pi}\, arctan\, \frac{k_1 + k_2}{k_1 - k_2} \tag{4.2}$$

We were inspired by the work proposed by Creusot et al. [1] to include the curvature related features into the point descriptors. ShapeIndex was also suggested by other work in the literature to be used in constructing the shape descriptor for 3D faces [9]. We combined the texture information in form of the hue and saturation to our descriptor, as well. To our best understanding, few number of landmarking techniques in the literature were based on the shape descriptors and no descriptors for the 3D face models in the literature were built using a combination of photometric and geometric features of the shapes. They mostly included only the local geometric information of the shape.

A final step in the data preparation was dealing with the unbalanced data. As the number of positive instances we collected in the previous step were much less than the number of negative ones, our dataset suffered from the class imbalance problem. Various techniques have been proposed in the literature to overcome such a problem [194]. One of the frequently used approaches was the data oversampling technique. Oversampling means to generate more samples from the rare class to build a dataset which has almost identical number of samples in all (in our case two) classes. This could be done in different ways such as simply duplicating the instances of the smaller class or synthesizing new samples from the rare class. In this application, we oversampled the positive class by repeating the shape descriptors of the rare class to tackle the problem of unbalanced data.

Supervised learning After the training data was prepared, we needed to setup the classifiers which learn each of the six landmarks, individually. We used the Multi-Layer Perceptron (MLP) and the Adaboost algorithms for the classification problem. Considering the number of available features per vertex and the total number of instances, the two methods were promising options that we expected to generate accurate results.

4.2.2 On-line phase

The on-line phase consists of an expert system and a machine learning module.

1. Expert system module: This module locates the 10 landmarks on each input unseen face automatically using various geometry concepts and computed features predefined by an expert (this module was explained in Chapter 3). The located landmarks were corresponding to the red keypoints illustrated in Figure 4.1.

2. Machine learning module: The result of the expert system module which were the 3D face models with 10 detected keypoints, are fed into this module which itself consists of three sub-modules: TPS deformation, cluster analysis, and classification. In fact, the clustering model and the classifier which were learned in the off-line phase, are tested in this module.

   Face deformation In the off-line phase we selected a reference face which is used in the machine learning module. The faces that were presented to the face deformation step had 10 detected landmarks. The corresponding landmarks were also detected on the reference mesh. Therefore, we used this set of corresponding points as the control points to initialize the TPS algorithm and deformed the input face towards the reference face.

Classification The clusters extracted in the previous step were presented to their relevant classifiers. For example, the 100 points that belonged to the right nostril cluster were used by the classifier which was train to detect the right nostril landmark and so on.

4.3 Experimental results

In this section we provide the results for the data-driven approach of our system. Table 4.1 shows the result of classifier training for all 12 classifiers (6 MLPs and 6 Adaboosts) trained for detecting six landmarks.

Table 4.1: Performance of MLP and Adaboost classifiers on 100 training faces (class 0: non-landmark, class 1: landmarks) using 10-fold cross validation.

metric (%)		right eye right corner		right eye left corner		left eye right corner		left eye left corner		right nostril		left nostril	
		MLP	Adaboost	MLP	Adaboost	MLP	Adaboost	MLP	Adaboost	MLP	Adaboost	MLP	Adaboost
TP rate	class 0	97	99.3	99.7	99.8	99.6	99.9	96.8	99.1	95.9	99.7	95.3	99.7
	class 1	96.8	100	99.6	100	99.2	100	90.8	100	94.8	100	92.4	100
	average	96.9	99.6	99.6	99.9	99.4	100	93.8	99.5	95.3	99.9	93.9	99.8

Table 4.2 shows the results of applying the MLP classifiers on 100 taring samples including 50 female and 50 male face models. Computing the distance between the manually selected landmarks and the ones detected by the algorithm is suggested as an evaluation metric in the literature [9]. Therefore, based on this idea we propose to evaluate our landmark detection system as such.

Sample faces from Figure 3.11 are shown in Figure 4.4 with 6 landmarks detected on them using the data-driven techniques.

Table 4.2: Average distance (localization error) between the detected landmarks and the ground-truth using the data-driven technique on 100 training and 50 testing faces of BJUT database.

localization error (mm)	right eye right corner		right eye left corner		left eye right corner		left eye left corner		right nostril		left nostril	
	train	test	train	test	train	test	train	test	train	test	train	test
average	2.07	3.15	0.65	1.05	0.97	0.82	2.15	2.43	1.04	1.47	0.67	0.97
standard deviation	2.81	4.38	1.54	1.94	1.86	1.52	3.33	3.95	0.93	2.22	0.76	2.24
min	0	0	0	0	0	0	0	0	0	0	0	0
max	19.30	24.63	14.46	11.95	14.42	8.56	22.75	22.31	6.98	15.88	3.20	15.88
<= 10mm (%)	98.00	92.00	99.00	98.00	99.00	100	97.00	96.00	100	98.00	100	98.00
<= 20mm (%)	100	92.00	100	100	100	100	97.00	96.00	100	100	100	100

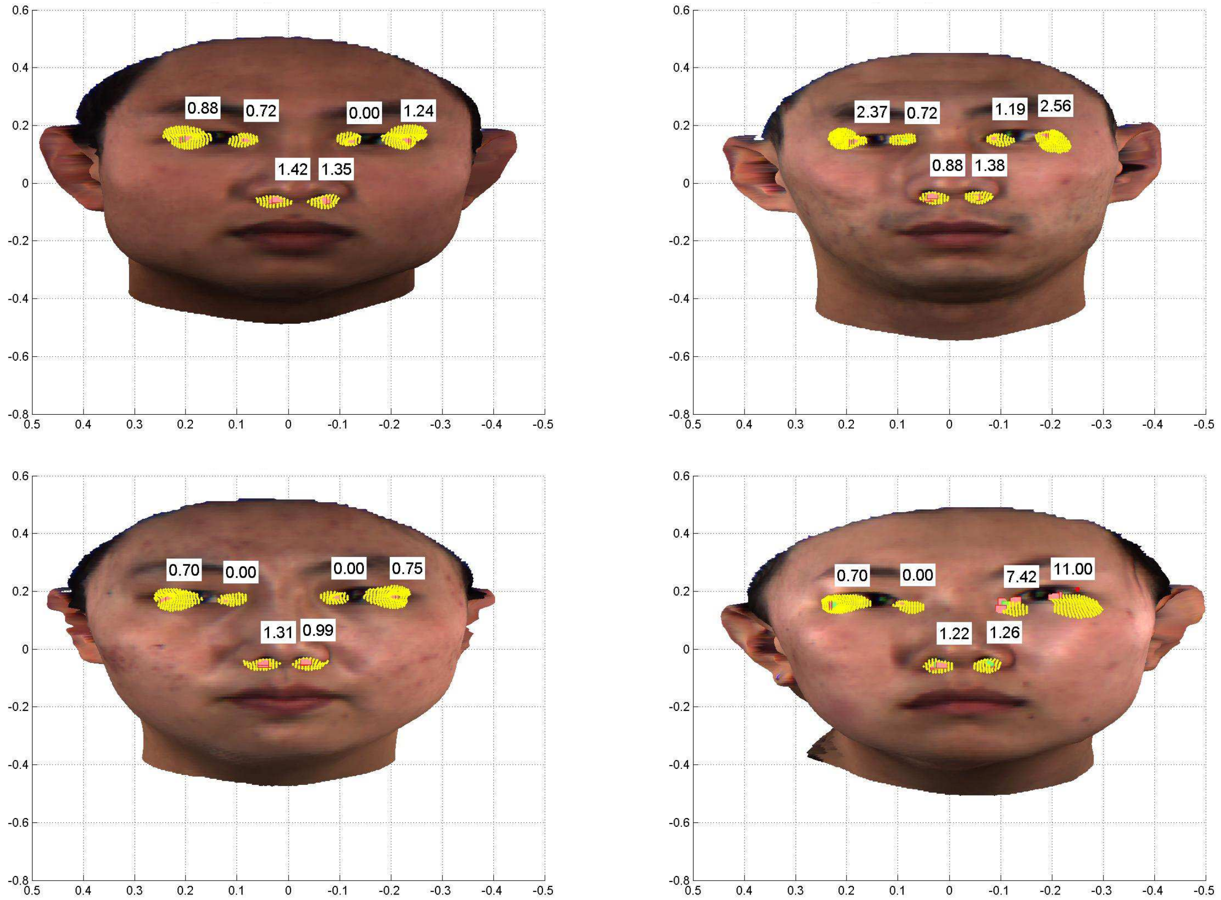


Figure 4.4: Six automatically detected landmarks and their corresponding distances from the ground-truth in millimeters.

4.4 Discussion

In this chapter we proposed a shape descriptor-based system to detect 6 landmarks on 3D face models automatically. We trained two different classifiers, Adaboost and MLP, to detect 6 landmarks on the 3D human faces. We used 100 faces (66% of the data) selected randomly from the BJUT database as the training data and 50 face models were used to test our classifiers.

In our experiments, Adaboost classifier generated better results with respect to MLP in terms of accuracy (Table 4.1). However, visualizing the classifiers results showed the MLP is superior to Adaboost. In fact, the number of false positives detected by Adaboost was less than that of the MLP; however, the localization error for those points were worse. In other words, the MLP detects more points as the false positives but they are close to the ground-truth. This, lead us to only consider the results of MLP classifier.

Limitations There were some limitations that confined our proposed approach:

- Since our technique is implemented on a Chinese dataset, it is very likely that our measurements of the faces which are mostly used in the expert system module of our framework may need some configuration to apply it on other datasets which include non-Asian faces.
- If a keypoint is detected inaccurately in the expert system module, it is very likely that the rest of the keypoints which were computed based on which are false localized[16].
- Our proposed approach works perfectly for the faces with neutral expression and frontal view without pose variations.

Chapter 5

Application and conclusion

5.1 Introduction

In this chapter we use our FLD system for establishing point-to-point correspondence between 3D face models. In fact, we utilize the automatically detected landmarks for establishing an accurate and precise point-to-point correspondence between the polygon meshes taken from the BJUIT database.

5.2 Point-to-point correspondence

Being inspired by the approach suggested in [165], this section elaborates an application of our proposed framework in establishing point-to-point correspondence between 3D face models. For this application, we select a reference face model which will be deformed towards a target model to find dense correspondences. Using this approach, we establish the correspondence by generating the target mesh using the reference mesh. The geometry and topology of the new face (the estimated face for the target) is identical to the geometry of the target mesh and the topology of the reference mesh, respectively.

The landmarks on the reference were already selected manually. Also, the corresponding landmarks are detected for the target model using our proposed system automatically. Then, the two sets of landmarks (i. e. those of reference and target meshes) will be used to initialize a TPS deforming module that transforms the reference mesh to the deformed space. Afterwards, an improved version of the ICP algorithm [24] is applied to move the deformed mesh toward the target mesh iteratively. In each iteration, the distance between the vertices of the reference mesh and a point of interest on the target mesh is computed. The point of interest would be the intersection of the normal vector of the vertex on the reference mesh and the surface of the target mesh. Then, the points of the reference mesh are moved towards their corresponding points on target mesh by a coefficient α (while $0 < \alpha < 1$) of the distance. We chose α to be $\frac{1}{5}$ of the distance in each iteration. We iterated the ICP algorithm five times for each face.

5.2.1 Spike removal

High levels of noise (particularly spikes) in the 3D models harden the shape analysis task. It was mentioned in the literature that inner eye corners are easy to detect in human 3D face models by metrics such as shapeIndex; however, spikes in the eyes (Figure 3.3) made it impossible to detect the eye corners using the knowledge-driven approaches, accurately. In fact, the algorithm was trapped by the spikes in many face models and selected eye corner points with high localization errors. Furthermore, finding the intersection points on the target meshes in the ICP algorithm is degraded when spikes are present. Therefore, the aforementioned problems encouraged us to implement a spike removal algorithm before establishing the correspondence.

It is impossible to remove spikes without degrading the mesh structure. This makes the landmark detection more challenging than it is for both knowledge-driven and data-driven approaches. However, we found that dealing with such problem using the machine learning algorithms is much easier than employing the heuristic approaches; that is why we detect the eye corners using the data-driven techniques.

Even though Taubin smoothing algorithm has been used in the pre-processing step to improve the triangulation quality of the meshes (Figure 3.8), it was not successful in spike removal. We did not find many spike removal algorithm proposed in the literature. In implementing our algorithm for spike removal we are inspired by the techniques suggested in ??. We limited our algorithm to removing the spikes from eyes as these areas include the most spikes and the the spikes in the eye affect our shape processing algorithm the most.

Spikes are areas with high curvatures. Therefore, we are interested in such points on the mesh. Also, since the eye spikes are the most problematic ones we focus on them. To narrow down our search area to finding the eye spikes, we look for the spikes whose deviations are towards the viewer (i. e., towards y axis). This way we can filter out the spikes which are located on the sides of the face models (we have many of them around the neck corners and the areas with hair). With liming our work to the spikes towards y axis in fact, we have limited our search to the eye areas. Because the majority of such spikes are located in the eye areas and we observed few spikes towards y axis which are not in the eyes.

Spike detection and removal algorithm

The spike detection and removal algorithm is implemented in an iterative manner. Following steps summarize the algorithm:

1. All points whose curvatures are beyond *mean curvature* $\pm 2\times$*standard deviation* are considered as spike candidates.
2. y coordinates of the candidates are updated with the median of y coordinates of their 1-ring neighbors (the x and z coordinates remain unchanged).
3. Curvature is re-computed.
4. Steps 1 to 3 are iterated until the cardinality of the candidate set is less than a threshold (we considered 100 elements) or for n number of times[1].

5.3 Experimental results

In this section we represent the results of applying our algorithm on 150 3D face models (75 males and 75 females) from BJUT database. 100 (50 males and 50 female) out of the 150

[1] We ran experiments for both and realized that there are not much smoothness after five iteration.

faces were used as the training data in the data-driven module of our FLD system the rest 50 faces (25 males and 25 females) were used at testing phase.

The average face of our database of 150 face models is illustrated in Figure 5.1. This face has 30,572 vertices and 60,983 triangles the same as all 150 face models of the database.

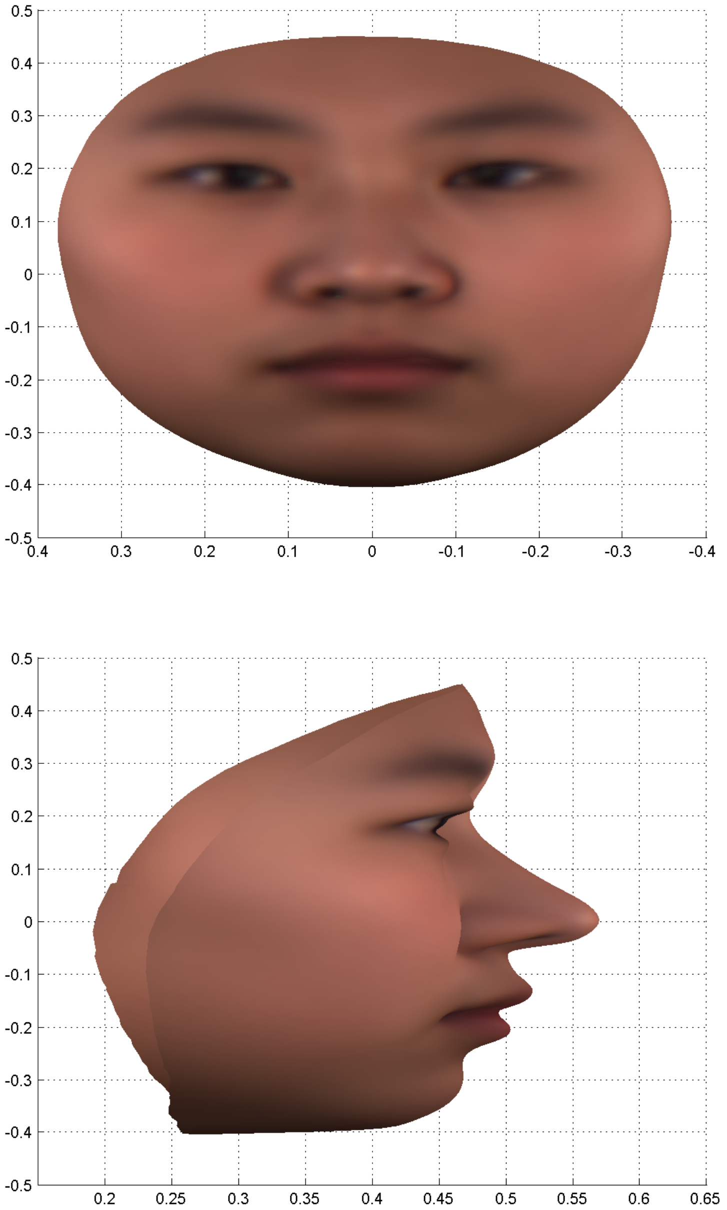


Figure 5.1: An average face obtained from the database of 150 human faces in point-to-point correspondence (frontal and side views).

Results of spike removal algorithm is also illustrated in Figure 5.2 for a sample face. Also, Figures 5.6 to 5.9 demonstrate the generated in full correspondence face models for the four samples from Figure 4.4.

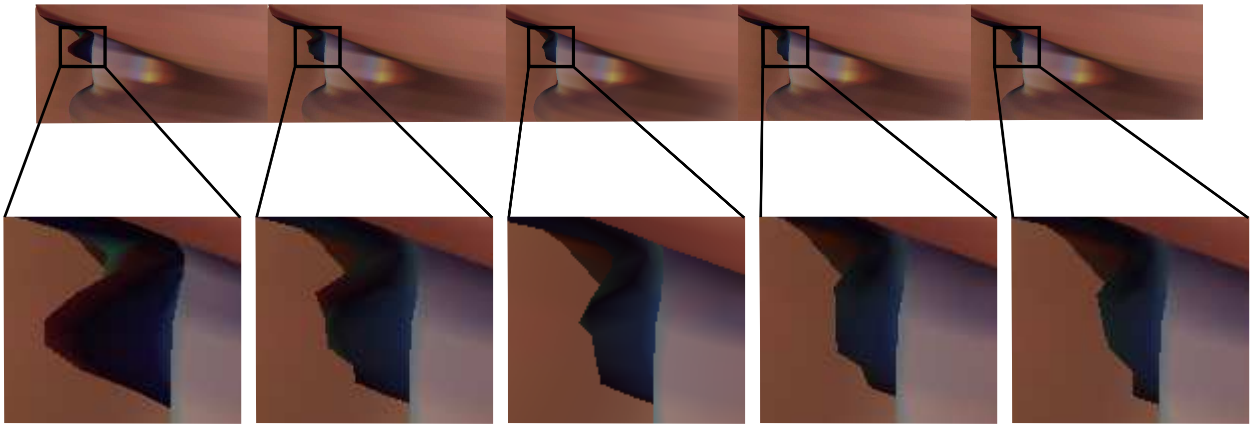


Figure 5.2: Top row: from left to right, the original mesh (eye) and first to last iterations of spike removal algorithm is demonstrated. Bottom row magnifies the particular spike areas.

5.4 Discussion

We used the 16 landmarks for establishing a set of dense point-to-point correspondences between various 3D face models. Applications of such correspondences would be face authentication and authorization, 3D face analysis, 3D face reconstruction and morphing, medical applications such as diagnosis of the abnormality in the skull structure, etc. Moreover, availability of a database of 3D face models which are in full correspondence is critical for most of the above applications. Many existing 2D and 3D face datasets are listed here `http://www.face-rec.org/databases/`; however, databases of 3D face models that are in point-to-point correspondence are rare in the literature. We applied our method on the BJUT dataset to build such 3D face benchmark and provide it for various 3D face analysis tasks.

Constructing the aforementioned dataset was a challenging task. It was mainly because a set of landmarks were necessary to initialize establishing the correspondence between the 3D face models. Such landmarks could be selected manually by user and then the TPS and

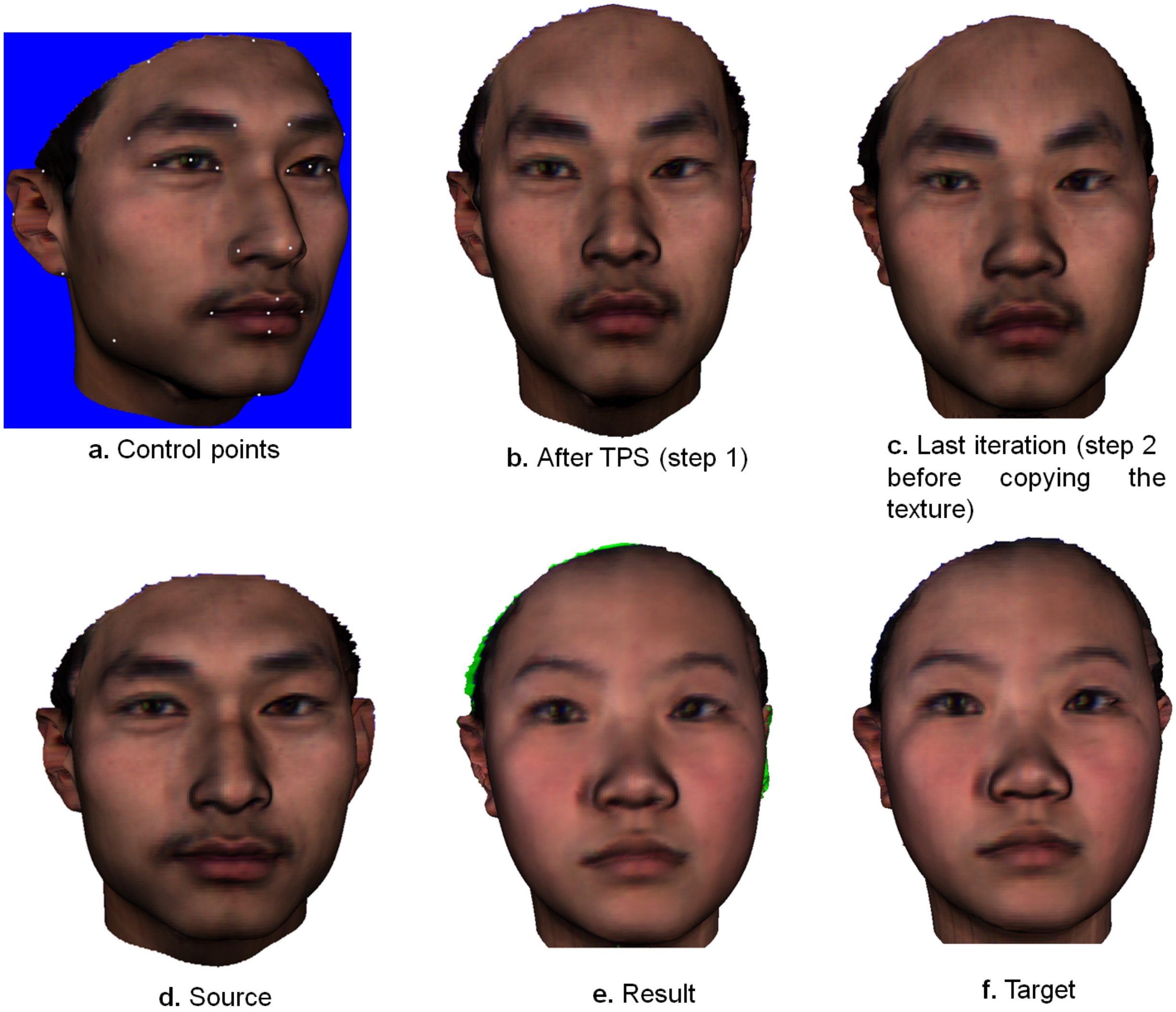
ICP algorithms can be used to establish the correspondence. Figure 5.3 shows a sample result of establishing the correspondence using 26 manually selected landmarks. However, manually selection of the landmarks for every 150 faces of dataset was time consuming, error prone, and labor intensive. Therefore, we designed our hybrid FLD system to initialize the correspondence module with automatically selected landmarks. That saves a great deal of time and speeds up constructing the database. In addition any new face model can be added to the dataset easily without need of manual landmark selection. On the other hand, using a FLD system (rather than asking a human to select the landmarks) add more errors to the system. Also, our FLD detects 10 less landmarks (Figure 5.3) that degrades the final results. In fact, initializing the TPS and ICP algorithm with more landmarks generates more accurate and precise results.

5.4.1 Performance metric

To measure how close the generated faces are to the original face models, we used the Euclidean distance between the surfaces of the faces as a performance metric. Figure 5.4 illustrates the Euclidean distance between the 150 faces and the reference face in each of the five iterations of ICP algorithm. Figure 5.5 demonstrates the timing for 150 faces for establishing correspondence using ICP algorithm. Other similar performance metrics [1] were also proposed in the literature.

5.5 Conclusion and future work

In this thesis we implemented a hybrid facial landmark detection system. The system consists of two knowledge-driven and data-driven modules. Ten and six landmarks were detected in the knowledge-driven and data-driven modules, respectively. This system detects one of the highest number of landmarks among the similar approaches in the literature [195]. The detected landmarks were used to establish point-to-point correspondence for 150 3D



a. Control points

b. After TPS (step 1)

c. Last iteration (step 2 before copying the texture)

d. Source

e. Result

f. Target

Figure 5.3: Face correspondence step-by-step. This implementation works with a large number of manually selected landmarks.

face models resulting in a database in full and dense correspondence that will be publicly available to the computer graphics community for various face analysis tasks.

Future work A future direction to continue this work would be to improve the accuracy of the landmark detection by either making more efficient rules in the knowledge-driven module or exploring more powerful classifiers in the data-driven landmark detection. Furthermore, recently introduced shape descriptors (reviewed in Chapter 2) that are applied directly on 3D meshes or point clouds could be used in the data-driven module. Such methods that are mostly based on deep learning should be more effective in capturing the local

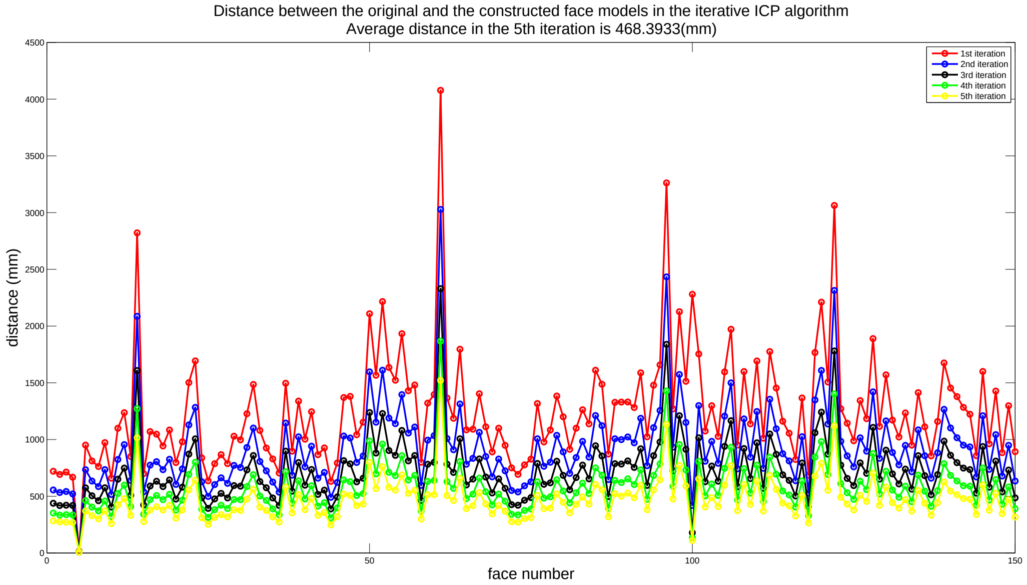


Figure 5.4: Distance statistics for establishing point-to-point correspondence.

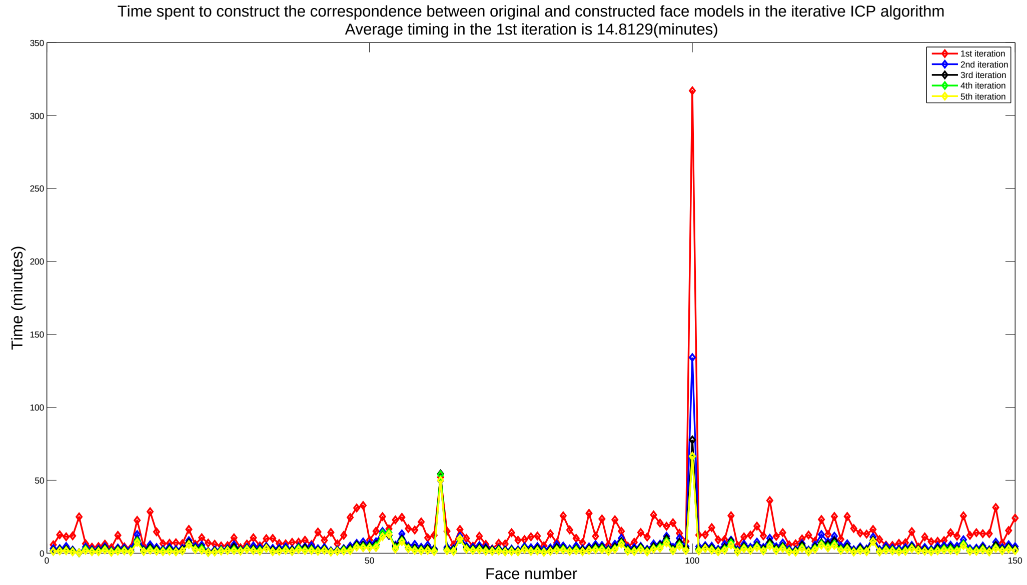


Figure 5.5: Time statistics for establishing point-to-point correspondence.

structure of the landmarks.

Another future work could be improving the spike removal algorithm. Our current approach performs well on the spikes and enhances mesh quality; however, a more proficient method could be suggested to take care of small spikes which are ignored by our algorithm.

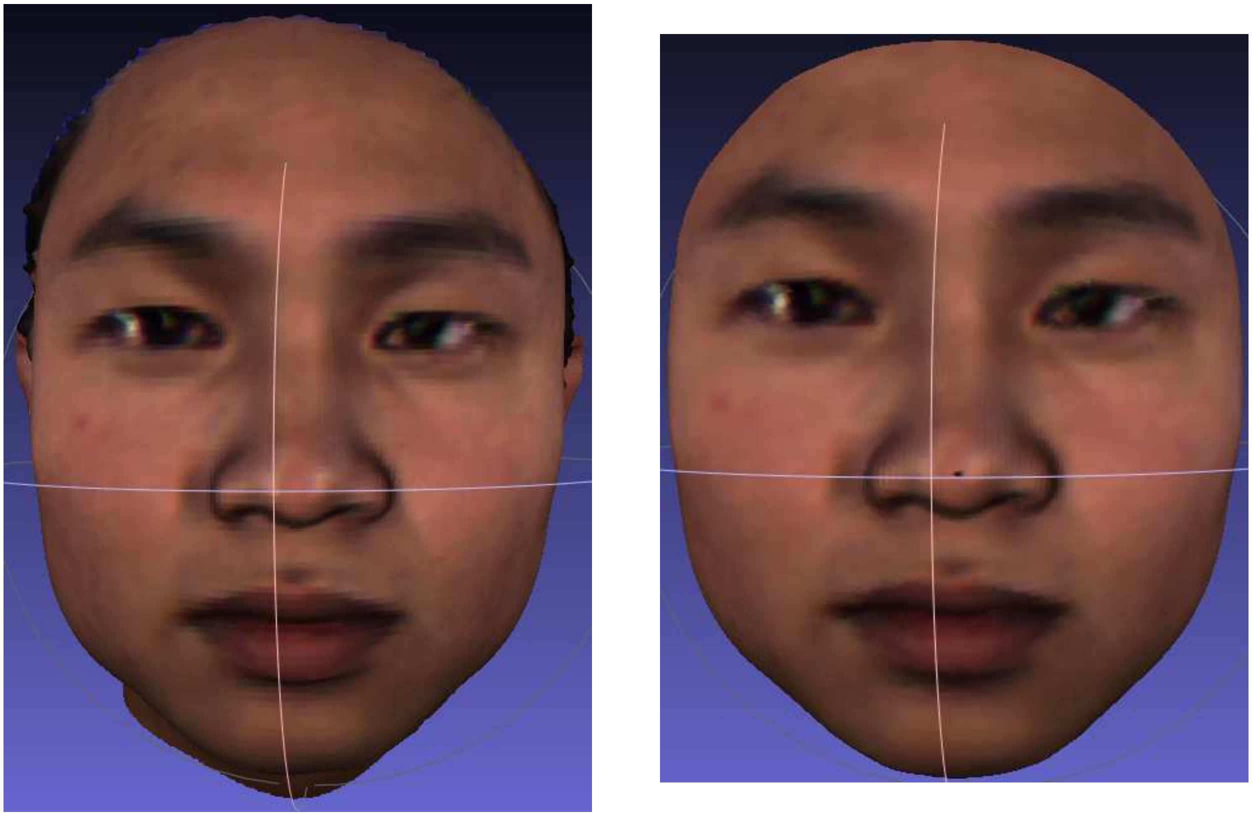


Figure 5.6: Original (left) and created (right) models. This face model is corresponding with the face on the top left of Figure 4.4.

Finally, more diverse set of 3D face models could be included to the generated database. The current database that we created includes only Chinese people. Adding more faces can make a more versatile and strong database which is useful in variety of face analysis tasks.

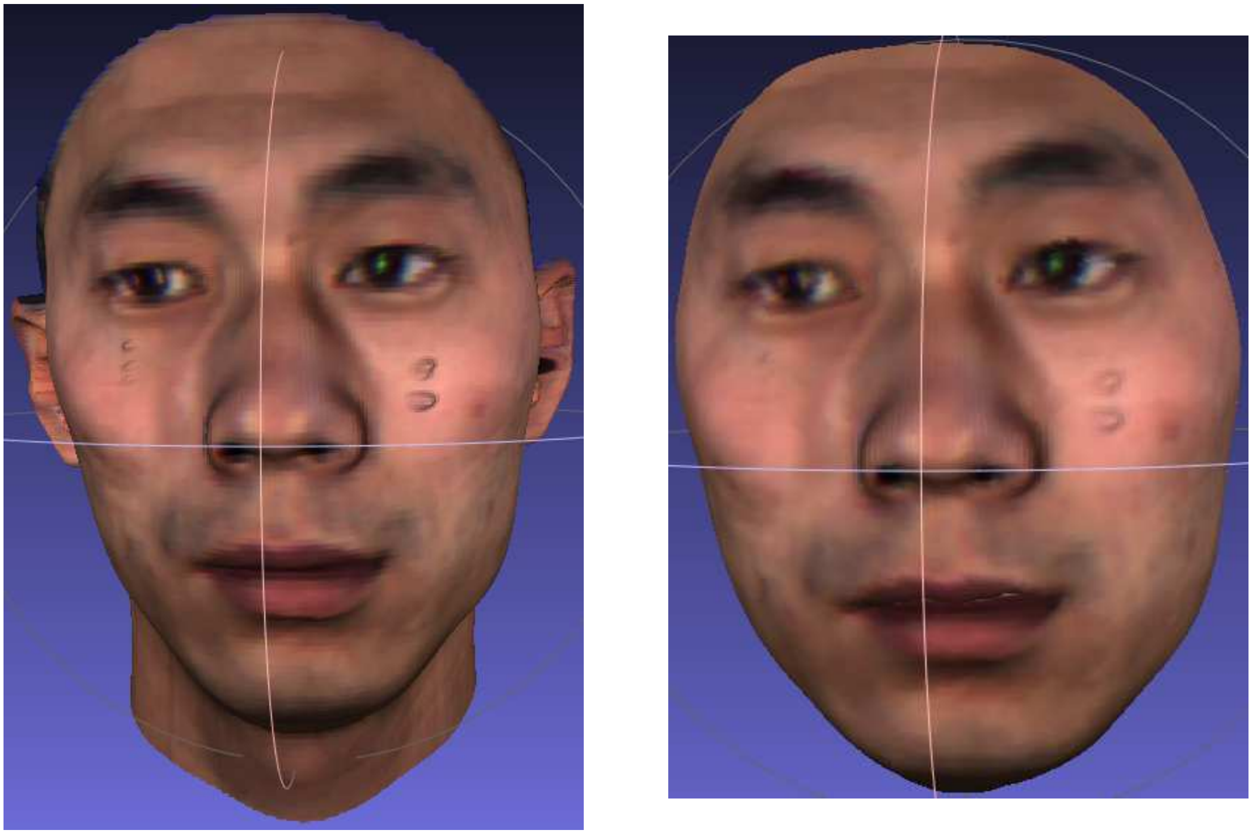


Figure 5.7: Original (left) and created (right) models. This face model is corresponding with the face on the top right of Figure 4.4.

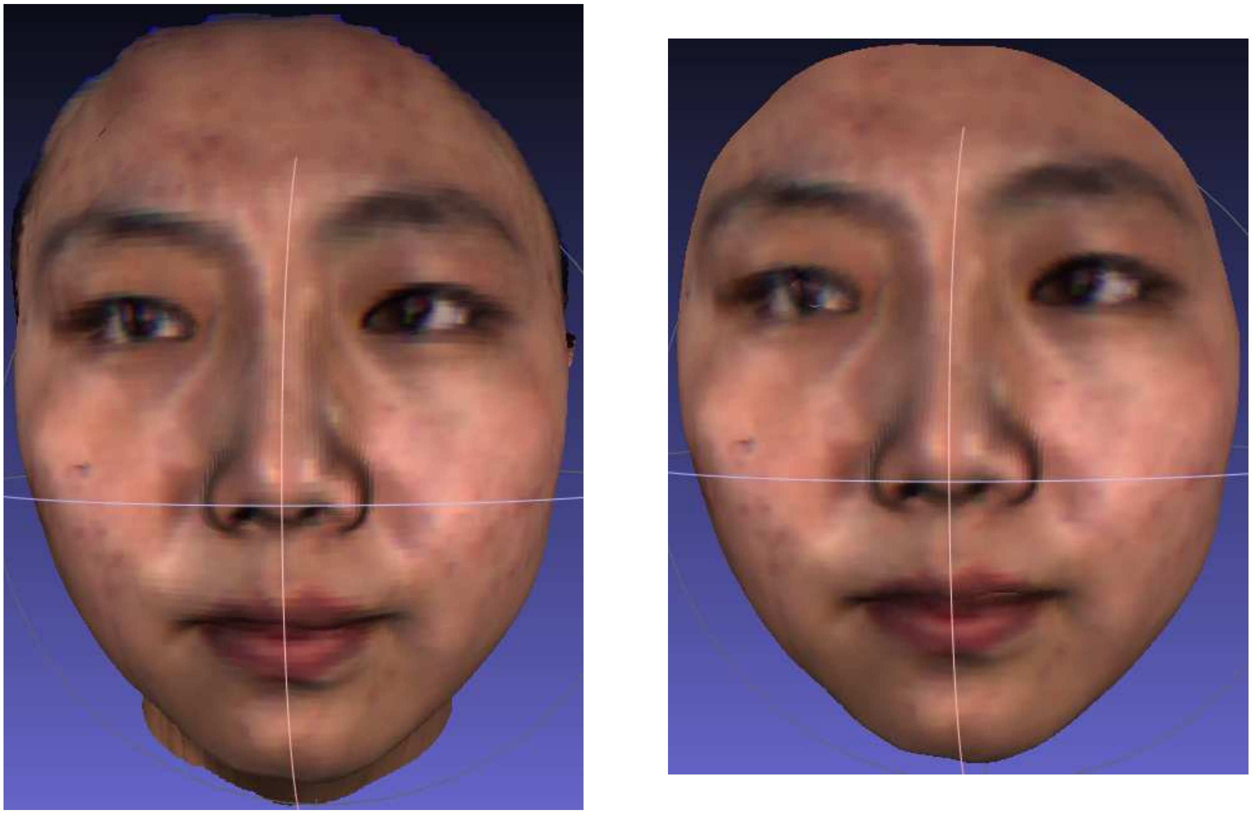


Figure 5.8: Original (left) and created (right) models. This face model is corresponding with the face on the bottom left of Figure 4.4.

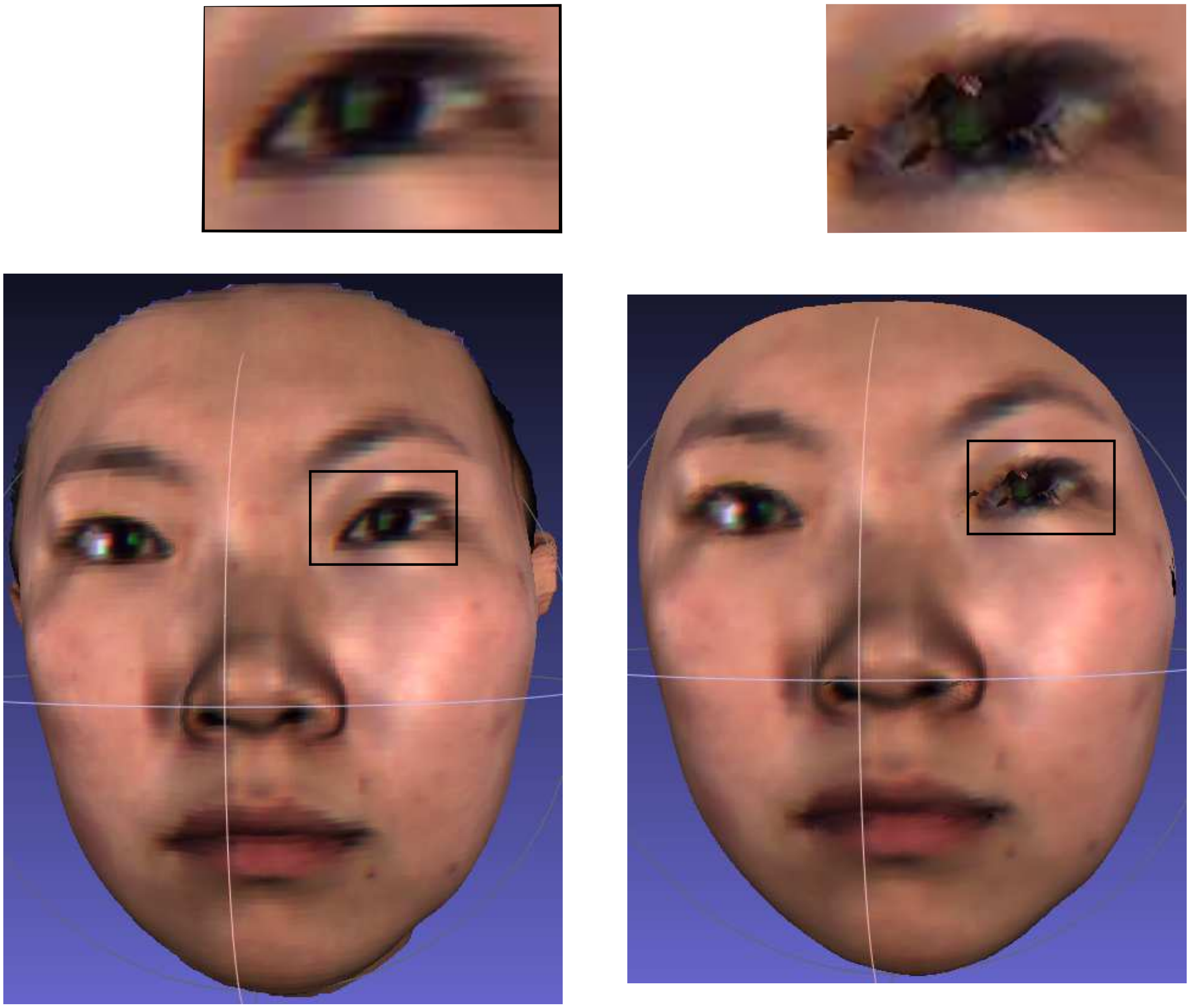


Figure 5.9: Original (left) and created (right) models. Left eye is noisy. As it is shown in the face model located on the bottom right of Figure 4.4, right and left corners of the left eye are detected with a large localization errors 7.42 and 11.00, respectively. This has degraded establishing point-to-point correspondence and quality of the generated mesh.

Bibliography

[1] C. Creusot, N. Pears, and J. Austin, “A machine-learning approach to keypoint detection and landmarking on 3d meshes,” International journal of computer vision **102**, 146–179 (2013).

[2] G. Lavoué, “Combination of bag-of-words descriptors for robust partial shape retrieval,” The Visual Computer **28**, 931–942 (2012).

[3] R. Litman, A. Bronstein, M. Bronstein, and U. Castellani, “Supervised learning of bag-of-features shape descriptors using sparse coding,” in “Computer Graphics Forum,” , vol. 33 (Wiley Online Library, 2014), vol. 33, pp. 127–136.

[4] J. Xie, M. Wang, and Y. Fang, “Learned binary spectral shape descriptor for 3d shape correspondence,” in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,” (2016), pp. 3309–3317.

[5] R. Litman and A. M. Bronstein, “Learning spectral descriptors for deformable shape correspondence,” IEEE transactions on pattern analysis and machine intelligence **36**, 171–180 (2014).

[6] Z. Han, Z. Liu, J. Han, C.-M. Vong, S. Bu, and X. Li, “Unsupervised 3d local feature learning by circle convolutional restricted boltzmann machine,” IEEE Transactions on Image Processing **25**, 5331–5344 (2016).

[7] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers, “Anisotropic diffusion descriptors,” in “Computer Graphics Forum,” , vol. 35 (Wiley Online Library, 2016), vol. 35, pp. 431–441.

[8] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3d deep shape descriptor," in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition," (2015), pp. 2319–2328.

[9] P. Perakis, G. Passalis, T. Theoharis, and I. A. Kakadiaris, "3d facial landmark detection under large yaw and expression variations," IEEE transactions on pattern analysis and machine intelligence **35**, 1552–1564 (2013).

[10] X. Fan, Q. Jia, K. Huyan, X. Gu, and Z. Luo, "3d facial landmark localization using texture regression via conformal mapping," Pattern Recognition Letters **83**, 395–402 (2016).

[11] F. Liu, D. Zeng, Q. Zhao, and X. Liu, "Joint face alignment and 3d face reconstruction," in "European Conference on Computer Vision," (Springer, 2016), pp. 545–560.

[12] C. Conde, R. Cipolla, L. J. Rodríguez-Aragón, Á. Serrano, and E. Cabello, "3d facial feature location with spin images," .

[13] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in "Proceedings of the 26th annual conference on Computer graphics and interactive techniques," (ACM Press/Addison-Wesley Publishing Co., 1999), SIGGRAPH '99, pp. 187–194.

[14] S. Liang, J. Wu, S. M. Weinberg, and L. G. Shapiro, "Improved detection of landmarks on 3d human face data," in "Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE," (IEEE, 2013), pp. 6482–6485.

[15] K. Wilamowska, *Shape-based quantification and classification of three dimensional face data for craniofacial research* (University of Washington, 2009).

[16] F. M. Sukno, J. L. Waddington, and P. F. Whelan, "3-d facial landmark localization with asymmetry patterns and shape regression from incomplete local features," IEEE transactions on cybernetics **45**, 1717–1730 (2015).

[17] T. Bolkart and S. Wuhrer, “A groupwise multilinear correspondence optimization for 3d faces,” in “The IEEE International Conference on Computer Vision (ICCV),” (2015).

[18] E. Omrani, A. P. Tafti, M. F. Fathi, A. D. Moghadam, P. Rohatgi, R. M. D’Souza, and Z. Yu, “Tribological study in microscale using 3d sem surface reconstruction,” Tribology International **103**, 309–315 (2016).

[19] L. Ng, S. Pathak, C. Kuan, C. Lau, H.-w. Dong, A. Sodt, C. Dang, B. Avants, P. Yushkevich, J. Gee *et al.*, “Neuroinformatics for genome-wide 3-d gene expression mapping in the mouse brain,” IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) **4**, 382–393 (2007).

[20] Z. Gao, R. Rostami, X. Pang, Z. Fu, and Z. Yu, “Mesh generation and flexible shape comparisons for bio-molecules,” Molecular Based Mathematical Biology **4**, 1–13 (2016).

[21] S. Riehemann, M. Palme, P. Kuehmstedt, C. Grossmann, G. Notni, and J. Hintersehr, “Microdisplay-based intraoral 3d scanner for dentistry,” Journal of Display Technology **7**, 151–155 (2011).

[22] C. Wu, D. Bradley, P. Garrido, M. Zollhöfer, C. Theobalt, M. Gross, and T. Beeler, “Model-based teeth reconstruction,” ACM Transactions on Graphics (TOG) **35**, 220 (2016).

[23] J. Han, L. Shao, D. Xu, and J. Shotton, “Enhanced computer vision with microsoft kinect sensor: A review,” IEEE transactions on cybernetics **43**, 1318–1334 (2013).

[24] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” IEEE Transactions on Pattern Analysis and Machine Intelligence **14**, 239–256 (1992).

[25] F. L. Bookstein, “Principal warps: Thin-plate splines and the decomposition of deformations,” IEEE Transactions on pattern analysis and machine intelligence **11**, 567–585 (1989).

[26] G. Wahba, *Spline models for observational data* (SIAM, 1990).

[27] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, "Three-dimensional shape searching: state-of-the-art review and future trends," Computer-Aided Design **37**, 509–530 (2005).

[28] J. W. Tangelder and R. C. Veltkamp, "A survey of content based 3d shape retrieval methods," in "Shape Modeling Applications, 2004. Proceedings," (IEEE, 2004), pp. 145–156.

[29] L. Zhang, M. J. da Fonseca, A. Ferreira, and C. R. A. e Recuperação, "Survey on 3d shape descriptors," Tech. rep., Technical Report, DecorAR (FCT POSC/EIA/59938/2004) (2007).

[30] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranić, "Feature-based similarity search in 3d object databases," ACM Computing Surveys (CSUR) **37**, 345–387 (2005).

[31] P. Heider, A. Pierre-Pierre, R. Li, and C. Grimm, "Local shape descriptors, a survey and evaluation," in "Proceedings of the 4th Eurographics Conference on 3D Object Retrieval," (Eurographics Association, 2011), 3DOR '11, pp. 49–56.

[32] S. Tang and A. Godil, "An evaluation of local shape descriptors for 3d shape retrieval," in "IS&T/SPIE Electronic Imaging," (International Society for Optics and Photonics, 2012), pp. 82900N–82900N.

[33] I. K. Kazmi, L. You, and J. J. Zhang, "A survey of 2d and 3d shape descriptors," in "Computer Graphics, Imaging and Visualization (CGIV), 2013 10th International Conference," (IEEE, 2013), pp. 1–10.

[34] R. Hoffman and A. K. Jain, "Segmentation and classification of range images," IEEE transactions on pattern analysis and machine intelligence **PAMI-9**, 608–620 (1987).

[35] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural computation **18**, 1527–1554 (2006).

[36] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," science **313**, 504–507 (2006).

[37] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," in "Computer Graphics Forum," , vol. 34 (Wiley Online Library, 2015), vol. 34, pp. 13–23.

[38] F. Steinke, B. Schölkopf, and V. Blanz, "Learning dense 3d correspondence," Advances in Neural Information Processing Systems **19**, 1313 (2007).

[39] K. Xu, V. G. Kim, Q. Huang, N. Mitra, and E. Kalogerakis, "Data-driven shape analysis and processing," in "SIGGRAPH ASIA 2016 Courses," (ACM, 2016), p. 4.

[40] C. Y. Ip, W. C. Regli, L. Sieger, and A. Shokoufandeh, "Automated learning of model classifications," in "Proceedings of the eighth ACM symposium on Solid modeling and applications," (ACM, 2003), pp. 322–327.

[41] F. Bürger, C. Buck, J. Pauli, and W. Luther, "Image-based object classification of defects in steel using data-driven machine learning optimization," in "Computer Vision Theory and Applications (VISAPP), 2014 International Conference on," , vol. 2 (IEEE, 2014), vol. 2, pp. 143–152.

[42] K. Guo, D. Zou, and X. Chen, "3d mesh labeling via deep convolutional neural networks," ACM Transactions on Graphics (TOG) **35**, 3 (2015).

[43] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3d shape surfaces using geometry images," in "European Conference on Computer Vision," (Springer, 2016), pp. 223–240.

[44] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, “Geodesic convolutional neural networks on riemannian manifolds,” in “Proceedings of the IEEE international conference on computer vision workshops,” (2015), pp. 37–45.

[45] S. Bu, Z. Liu, J. Han, J. Wu, and R. Ji, “Learning high-level feature by deep belief networks for 3-d model retrieval and recognition,” IEEE Transactions on Multimedia **16**, 2154–2167 (2014).

[46] L. Wan, J. Jiang, and H. Zhang, “Incomplete 3d shape retrieval via sparse dictionary learning,” Pacific Graphics (2015).

[47] Z. Liu, S. Chen, S. Bu, and K. Li, “High-level semantic feature for 3d shape based on deep belief networks,” in “Multimedia and Expo (ICME), 2014 IEEE International Conference on,” (IEEE, 2014), pp. 1–6.

[48] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, “Shape google: Geometric words and expressions for invariant shape retrieval,” ACM Transactions on Graphics (TOG) **30**, 1 (2011).

[49] H. Zhang, O. Van Kaick, and R. Dyer, “Spectral mesh processing,” in “Computer graphics forum,” , vol. 29 (Wiley Online Library, 2010), vol. 29, pp. 1865–1894.

[50] B. Levy, “Laplace-beltrami eigenfunctions towards an algorithm that ”understands” geometry,” in “IEEE International Conference on Shape Modeling and Applications 2006 (SMI’06),” (2006), pp. 13–13.

[51] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo, “Discrete laplace–beltrami operators for shape analysis and segmentation,” Computers & Graphics **33**, 381–390 (2009).

[52] R. M. Rustamov, “Laplace-beltrami eigenfunctions for deformation invariant shape representation,” in “Proceedings of the fifth Eurographics symposium on Geometry processing,” (Eurographics Association, 2007), pp. 225–233.

[53] J. Sun, M. Ovsjanikov, and L. Guibas, “A concise and provably informative multi-scale signature based on heat diffusion,” in “Computer graphics forum,” , vol. 28 (Wiley Online Library, 2009), vol. 28, pp. 1383–1392.

[54] M. Aubry, U. Schlickewei, and D. Cremers, “The wave kernel signature: A quantum mechanical approach to shape analysis,” in “Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on,” (IEEE, 2011), pp. 1626–1633.

[55] D. Raviv, R. Kimmel, and A. M. Bruckstein, “Graph isomorphisms and automorphisms via spectral signatures,” IEEE transactions on pattern analysis and machine intelligence **35**, 1985–1993 (2013).

[56] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas, “Persistence-based segmentation of deformable shapes,” in “Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on,” (IEEE, 2010), pp. 45–52.

[57] D. De Youngster, E. Paquet, H. Viktor, and E. Petriu, “An isometry-invariant spectral approach for protein-protein docking,” in “Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on,” (IEEE, 2013), pp. 1–6.

[58] M. M. Bronstein and I. Kokkinos, “Scale-invariant heat kernel signatures for non-rigid shape recognition,” in “Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on,” (IEEE, 2010), pp. 1704–1711.

[59] D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel, “Volumetric heat kernel signatures,” in “Proceedings of the ACM workshop on 3D object retrieval,” (ACM, 2010), pp. 39–44.

[60] M. Aubry, U. Schlickewei, and D. Cremers, “Pose-consistent 3d shape segmentation based on a quantum mechanical feature descriptor,” in “Joint Pattern Recognition Symposium,” (Springer, 2011), pp. 122–131.

[61] H. Lee, A. Battle, R. Raina, and A. Y. Ng, “Efficient sparse coding algorithms,” in “Advances in neural information processing systems,” (2007), pp. 801–808.

[62] L. Yang, “Distance metric learning: A comprehensive survey,” (2006).

[63] Y. Bengio *et al.*, “Learning deep architectures for ai,” Foundations and trends® in Machine Learning **2**, 1–127 (2009).

[64] I. Goodfellow, “Nips 2016 tutorial: Generative adversarial networks,” arXiv preprint arXiv:1701.00160 (2016).

[65] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, “An empirical evaluation of deep architectures on problems with many factors of variation,” in “Proceedings of the 24th international conference on Machine learning,” (ACM, 2007), pp. 473–480.

[66] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, “Greedy layer-wise training of deep networks,” Advances in neural information processing systems **19**, 153–160 (2007).

[67] J. R. Quinlan, “Induction of decision trees,” Machine learning **1**, 81–106 (1986).

[68] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” (1967).

[69] C. Cortes and V. Vapnik, “Support-vector networks,” Machine learning **20**, 273–297 (1995).

[70] R. Hecht-Nielsen, *Neurocomputing* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989).

[71] É. Corman, M. Ovsjanikov, and A. Chambolle, "Supervised descriptor learning for non-rigid shape matching," in "European Conference on Computer Vision," (Springer, 2014), pp. 283–298.

[72] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," Mathematical Programming **45**, 503–528 (1989).

[73] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).

[74] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, "Deep learning representation using autoencoder for 3d shape retrieval," Neurocomputing **204**, 41–50 (2016).

[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT Press, 2016). `http://www.deeplearningbook.org`.

[76] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," Foundations and Trends® in Signal Processing **7**, 197–387 (2014).

[77] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks **61**, 85–117 (2015).

[78] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification." in "NIPS," , vol. 3 (2012), vol. 3, p. 8.

[79] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. G. KIM, and Y. Lipman, "Convolutional neural networks on surfaces via seamless toric covers," (2017).

[80] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural computation **1**, 541–551 (1989).

[81] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE **86**, 2278–2324 (1998).

[82] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel distributed processing* (MIT Press, Cambridge, MA, USA, 1986), vol. 1: Foundations, chap. Learning internal representations by error propagation, pp. 318–362.

[83] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in "Proceedings of the 25th international conference on Machine learning," (ACM, 2008), pp. 1096–1103.

[84] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in "2007 IEEE Conference on Computer Vision and Pattern Recognition," (2007), pp. 1–8.

[85] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures." ICML unsupervised and transfer learning **27**, 37–50 (2012).

[86] G. E. Hinton and T. J. Sejnowski, *Parallel distributed processing* (MIT Press, Cambridge, MA, USA, 1986), vol. 1: Foundations, chap. Learning and relearning in Boltzmann machines, pp. 282–317.

[87] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," Cognitive science **9**, 147–169 (1985).

[88] S. Kullback and R. A. Leibler, "On information and sufficiency," The annals of mathematical statistics **22**, 79–86 (1951).

[89] "Deeplearning4j: Open-source distributed deep learning for the jvm," `https://deeplearning4j.org/`. Apache Software Foundation License 2.0. [Online; accessed 25-August-2016].

[90] "The neural network zoo," `http://www.asimovinstitute.org/neural-network-zoo/,`. [Online; accessed 13-October-2017].

[91] G. E. Hinton, "Deep belief networks," Scholarpedia **4**, 5947 (2009).

[92] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," IEEE Transactions on Audio, Speech, and Language Processing **20**, 30–42 (2012).

[93] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," IEEE transactions on pattern analysis and machine intelligence **35**, 1915–1929 (2013).

[94] A.-r. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in "NIPS workshop on deep learning for speech recognition and related applications," , vol. 1 (2009), vol. 1, p. 39.

[95] A. M. Abdel-Zaher and A. M. Eldeib, "Breast cancer classification using deep belief networks," Expert Systems with Applications **46**, 139–144 (2016).

[96] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," IEEE Transactions on Intelligent Transportation Systems **16**, 865–873 (2015).

[97] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," Nature methods **12**, 931–934 (2015).

[98] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of dna-and rna-binding proteins by deep learning," Nature biotechnology **33**, 831–838 (2015).

[99] N. F. Troje and H. H. Bülthoff, "Face recognition under varying poses: The role of texture and shape," Vision research **36**, 1761–1771 (1996).

[100] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in "Shape modeling applications, 2004. Proceedings," (IEEE, 2004), pp. 167–178.

[101] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in "Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.", , vol. 2 (IEEE, 2004), vol. 2, pp. 97–104.

[102] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in "ACM Transactions on Graphics (TOG)," , vol. 24 (ACM, 2005), vol. 24, pp. 408–416.

[103] D. Anguelov, "Learning models of shape from 3d range data," Ph.D. thesis, Stanford University (2005).

[104] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes* (Springer Science & Business Media, 2008).

[105] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Calculus of nonrigid surfaces for geometry and texture manipulation," IEEE Transactions on Visualization and Computer Graphics **13**, 902–913 (2007).

[106] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Efficient computation of isometry-invariant distances between surfaces," SIAM Journal on Scientific Computing **28**, 1812–1836 (2006).

[107] R. C. Veltkamp, R. Ruijsenaars, M. Spagnuolo, R. Van Zwol, and F. ter Haar, "Shrec2006: 3d shape retrieval contest," Tech. Rep. UU-CS-2006-030, Department of Information and Computing Sciences,Utrecht University (2006).

[108] R. C. Veltkamp and F. B. ter Haar, “Shrec2007: 3d shape retrieval contest,” Tech. Rep. UU-CS-2007-015, Department of Information and Computing Sciences,Utrecht University (2007).

[109] R. C. Veltkamp and F. B. Ter Haar, “Shrec 2009 - shape retrieval contest,” in “Proceedings of the 2nd Eurographics conference on 3D Object Retrieval,” (Eurographics Association, 2009), 3DOR ’09, pp. 57–59.

[110] R. Fang, A. Godil, X. Li, and A. Wagan, “A new shape benchmark for 3d object retrieval,” in “International Symposium on Visual Computing,” (Springer, 2008), pp. 381–392.

[111] F. Bogo, J. Romero, M. Loper, and M. J. Black, “Faust: Dataset and evaluation for 3d mesh registration,” in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,” (2014), pp. 3794–3801.

[112] T. Vanamali, A. Godil, H. Dutagaci, T. Furuya, Z. Lian, and R. Ohbuchi, “Shrec’10 track: generic 3d warehouse,” in “Proceedings of the 3rd Eurographics conference on 3D Object Retrieval,” (Eurographics Association, 2010), pp. 93–100.

[113] R. Muthuganapathy and K. Ramani, “Shape retrieval contest 2008: Cad models.” in “Shape Modeling International,” (2008), pp. 221–222.

[114] K. Siddiqi, J. Zhang, D. Macrini, A. Shokoufandeh, S. Bouix, and S. Dickinson, “Retrieving articulated 3-d models using medial surfaces,” Machine vision and applications **19**, 261–275 (2008).

[115] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgb-d object dataset,” in “Robotics and Automation (ICRA), 2011 IEEE International Conference on,” (IEEE, 2011), pp. 1817–1824.

[116] M. Blum, J. T. Springenberg, J. Wülfing, and M. Riedmiller, "A learned feature descriptor for object recognition in rgb-d data," in "Robotics and Automation (ICRA), 2012 IEEE International Conference on," (IEEE, 2012), pp. 1298–1303.

[117] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition," (2015), pp. 1912–1920.

[118] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in "European Conference on Computer Vision," (Springer, 2012), pp. 746–760.

[119] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in "Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on," (IEEE, 2011), pp. 601–608.

[120] V. Nair and G. E. Hinton, "3d object recognition with deep belief nets," in "Advances in Neural Information Processing Systems," (2009), pp. 1339–1347.

[121] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, *Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering*, vol. 30 (ACM, 2011).

[122] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," ACM Transactions on Graphics (TOG) **31**, 165:1–165:10 (2012).

[123] O. Van Kaick, A. Tagliasacchi, O. Sidi, H. Zhang, D. Cohen-Or, L. Wolf, and G. Hamarneh, "Prior knowledge for part correspondence," in "Computer Graphics Forum," , vol. 30Wiley Online Library (Blackwell Publishing Ltd, 2011), vol. 30, pp. 553–562.

[124] T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang, “Persistent heat signature for pose-oblivious matching of incomplete models,” in “Computer Graphics Forum,” , vol. 29 (Wiley Online Library, 2010), vol. 29, pp. 1545–1554.

[125] H. Riemenschneider, A. Bódis-Szomorú, J. Weissenberg, and L. Van Gool, “Learning where to classify in multi-view semantic segmentation,” in “European Conference on Computer Vision,” (Springer, 2014), pp. 516–532.

[126] M. Savva, A. X. Chang, and P. Hanrahan, “Semantically-Enriched 3D Models for Common-sense Knowledge,” CVPR 2015 Workshop on Functionality, Physics, Intentionality and Causality (2015).

[127] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “ShapeNet: An Information-Rich 3D Model Repository,” Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015).

[128] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, “A scalable active framework for region annotation in 3d shape collections,” SIGGRAPH Asia (2016).

[129] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3d reconstructions of indoor scenes,” in “Proc. Computer Vision and Pattern Recognition (CVPR), IEEE,” (2017).

[130] Y. Aflalo, A. M. Bronstein, M. M. Bronstein, and R. Kimmel, “Deformable shape retrieval by learning diffusion kernels,” in “International Conference on Scale Space and Variational Methods in Computer Vision,” (Springer, 2011), pp. 689–700.

[131] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, “Functional maps: a flexible representation of maps between shapes,” ACM Transactions on Graphics (TOG) **31**, 30 (2012).

[132] R. Toldo, U. Castellani, and A. Fusiello, “Visual vocabulary signature for 3d object retrieval and partial matching,” in “Proceedings of the 2nd Eurographics conference on 3D Object Retrieval,” (Eurographics Association, 2009), pp. 21–28.

[133] R. Toldo, U. Castellani, and A. Fusiello, “The bag of words approach for retrieval and categorization of 3d objects,” The Visual Computer **26**, 1257–1268 (2010).

[134] P. Jain, B. Kulis, and K. Grauman, “Fast image search for learned metrics,” in “Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on,” (IEEE, 2008), pp. 1–8.

[135] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in “European conference on computer vision,” (Springer, 2006), pp. 404–417.

[136] T. K. Ho, “Random decision forests,” in “Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on,” , vol. 1 (IEEE, 1995), vol. 1, pp. 278–282.

[137] Z. Lian, A. Godil, X. Sun, and J. Xiao, “Cm-bof: visual similarity-based 3d shape retrieval using clock matching and bag-of-features,” Machine Vision and Applications **24**, 1685–1704 (2013).

[138] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International journal of computer vision **60**, 91–110 (2004).

[139] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun, “A probabilistic model for component-based shape synthesis,” ACM Transactions on Graphics (TOG) **31**, 55 (2012).

[140] H. Huang, E. Kalogerakis, and B. Marlin, “Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces,” in “Computer Graphics Forum,” , vol. 34 (Wiley Online Library, 2015), vol. 34, pp. 25–38.

[141] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in "Advances in Neural Information Processing Systems," (2016), pp. 82–90.

[142] L. Yi, H. Su, X. Guo, and L. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," (2017).

[143] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in "Proceedings of the IEEE international conference on computer vision," (2015), pp. 945–953.

[144] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in "Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on," (IEEE, 2017), pp. 199–208.

[145] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in "Advances in Neural Information Processing Systems," (1994), pp. 737–744.

[146] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Ambient sound provides supervision for visual learning," in "European Conference on Computer Vision," (Springer, 2016), pp. 801–816.

[147] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer, "Learning local shape descriptors from part correspondences with multiview convolutional networks," ACM Trans. Graph. **37**, 6:1–6:14 (2017).

[148] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," IEEE Signal Processing Magazine **34**, 18–42 (2017).

[149] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, “Volumetric and multi-view cnns for object classification on 3d data,” in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,” (2016), pp. 5648–5656.

[150] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” (2017).

[151] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, “O-cnn: Octree-based convolutional neural networks for 3d shape analysis,” ACM Trans. Graph. **36**, 72:1–72:11 (2017).

[152] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” arXiv preprint arXiv:1706.02413 (2017).

[153] H. Deng, T. Birdal, and S. Ilic, “Ppfnet: Global context aware local features for robust 3d point matching,” arXiv preprint arXiv:1802.02669 (2018).

[154] G. Riegler, O. Ulusoy, and A. Geiger, “Octnet: Learning deep 3d representations at high resolutions,” in “IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),” (2017).

[155] G. Alain and Y. Bengio, “What regularized auto-encoders learn from the data-generating distribution,” The Journal of Machine Learning Research **15**, 3563–3593 (2014).

[156] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559–572 (1901).

[157] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” Annals of eugenics **7**, 179–188 (1936).

[158] J. Xie, Y. Fang, F. Zhu, and E. Wong, “Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval,” in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,” (2015), pp. 1275–1283.

[159] G. Dai, J. Xie, F. Zhu, and Y. Fang, “Learning a discriminative deformation-invariant 3d shape descriptor via many-to-one encoder,” Pattern Recognition Letters **83**, 330–338 (2016).

[160] G. Hetzel, B. Leibe, P. Levi, and B. Schiele, “3d object recognition from range images using local feature histograms,” in “Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on,” , vol. 2 (IEEE, 2001), vol. 2, pp. II–II.

[161] A. Zaharescu, E. Boyer, and R. Horaud, “Keypoints and local descriptors of scalar functions on 2d manifolds,” International Journal of Computer Vision **100**, 78–98 (2012).

[162] R. Socher, “Recursive deep learning for natural language processing and computer vision,” Ph.D. thesis, Citeseer (2014).

[163] X. Bai, S. Bai, Z. Zhu, and L. J. Latecki, “3d shape matching via two layer coding,” IEEE transactions on pattern analysis and machine intelligence **37**, 2361–2373 (2015).

[164] Q. Deng, M. Zhou, and Z. Wu, “An automatic non-rigid registration method for dense surface models,” in “Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on,” , vol. 1 (IEEE, 2010), vol. 1, pp. 888–892.

[165] Y. Hu, M. Zhou, and Z. Wu, “A dense point-to-point alignment method for realistic 3d face morphing and animation,” International Journal of Computer Games Technology **2009**, 3 (2009).

[166] A. M. Bronstein, “Spectral descriptors for deformable shapes,” arXiv preprint arXiv:1110.5015 (2011).

[167] E. Corman, M. Ovsjanikov, and A. Chambolle, “Continuous matching via vector field flow,” in “Computer Graphics Forum,” , vol. 34 (Wiley Online Library, 2015), vol. 34, pp. 129–139.

[168] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” Neurocomputing **187**, 27–48 (2016).

[169] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in “Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on,” (IEEE, 2015), pp. 922–928.

[170] A. Dai, C. R. Qi, and M. Nießner, “Shape completion using 3d-encoder-predictor cnns and shape synthesis,” arXiv preprint arXiv:1612.00101 (2016).

[171] C. Li, D. Alvarez-Melis, K. Xu, S. Jegelka, and S. Sra, “Distributional adversarial networks,” arXiv preprint arXiv:1706.09549 (2017).

[172] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” The International Journal of Robotics Research **34**, 705–724 (2015).

[173] N. Sedaghat, M. Zolfaghari, and T. Brox, “Orientation-boosted voxel nets for 3d object recognition,” arXiv preprint arXiv:1604.03351 (2016).

[174] R. F. H. ORMESHER, “Towards unsupervised object classification and prediction in 3d through semantic sampling.” (2017).

[175] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger, “Octnetfusion: Learning depth fusion from data,” arXiv preprint arXiv:1704.01047 (2017).

[176] M. Wang, J. Xie, F. Zhu, and Y. Fang, “Linear discrimination dictionary learning for shape descriptors,” Pattern Recognition Letters **83**, 349–356 (2016).

[177] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang, “Projective feature learning for 3d shapes with multi-view depth images,” Computer Graphics Forum **34**, 1–11 (2015).

[178] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, “3D shape segmentation with projective convolutional networks,” in “Proc. IEEE Computer Vision and Pattern Recognition (CVPR),” (2017).

[179] X. Gu, S. J. Gortler, and H. Hoppe, “Geometry images,” ACM Transactions on Graphics (TOG) **21**, 355–361 (2002).

[180] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, “Grass: Generative recursive autoencoders for shape structures,” ACM Transactions on Graphics (TOG) **36**, 52 (2017).

[181] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, “Learning shape abstractions by assembling volumetric primitives,” in “Proc. CVPR,” , vol. 2 (2017), vol. 2.

[182] M. Reuter, F.-E. Wolter, and N. Peinecke, “Laplace–beltrami spectra as ’shape-dna’of surfaces and solids,” Computer-Aided Design **38**, 342–366 (2006).

[183] Y. Baocai, S. Yanfeng, W. Chengzhang, and G. Yun, “Bjut-3d large scale 3d face database and information processing,” Journal of Computer Research and Development **6**, 020 (2009).

[184] X. Hu, Y. Wang, F. Zhu, and C. Pan, “Learning-based fully 3d face reconstruction from a single image,” in “Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on,” (IEEE, 2016), pp. 1651–1655.

[185] S. Li, X. Liu, X. Chai, H. Zhang, S. Lao, and S. Shan, “Morphable displacement field based image matching for face recognition across pose,” Computer Vision–ECCV 2012 pp. 102–115 (2012).

[186] S. Li, X. Liu, X. Chai, H. Zhang, S. Lao, and S. Shan, "Maximal likelihood correspondence estimation for face recognition across pose," IEEE Transactions on Image Processing **23**, 4587–4600 (2014).

[187] G. Taubin, "Curve and surface smoothing without shrinkage," in "Computer Vision, 1995. Proceedings., Fifth International Conference on," (IEEE, 1995), pp. 852–857.

[188] S. Rusinkiewicz, "Estimating curvatures and their derivatives on triangle meshes," in "3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on," (IEEE, 2004), pp. 486–493.

[189] Y. B. Shabat and A. Fischer, "Design of porous micro-structures using curvature analysis for additive-manufacturing," Procedia CIRP **36**, 279–284 (2015).

[190] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," Computer Graphics Forum **28**, 1383–1392 (2009).

[191] A. A. Salah, N. Alyüz, and L. Akarun, "Registration of three-dimensional face scans with average face models," Journal of Electronic Imaging **17**, 011006–011006 (2008).

[192] L. Anuar, S. Mashohor, M. Mokhtar, and W. Wan Adnan, "Nose tip region detection in 3d facial model across large pose variation and facial expression," IJCSI (2010).

[193] J. Guo, X. Mei, and K. Tang, "Automatic landmark annotation and dense correspondence registration for 3d human facial images," BMC bioinformatics **14**, 232 (2013).

[194] H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Transactions on knowledge and data engineering **21**, 1263–1284 (2009).

[195] F. M. Sukno, J. L. Waddington, and P. F. Whelan, "3-d facial landmark localization with asymmetry patterns and shape regression from incomplete local features," IEEE Transactions on Cybernetics **45**, 1717–1730 (2015).

Curriculum Vitae

Reihaneh Rostami

EDUCATION

Doctor of Philosophy, Computer Science

University of Wisconsin - Milwaukee, Milwaukee, WI — Fall 2018

Master of Science, Computer Engineering

Amirkabir University of Technology, Tehran, Iran — Feb. 2010

Bachelor of Science, Scientific-Applied Computer Engineering

University of Birjand, Birjand, Iran — Sep. 2005

Associate Degree, Scientific-Applied Computer Engineering

University of Zanjan, Zanjan, Iran — Jan. 2003

INTERNSHIP EXPERIENCE

SAS Technician (summer intern) — Jun-Aug. 2017

Capacity Management Group, **Thomson Reuters**, Brookfield, WI, USA

Project: Design and implementation of a dashboarding system using SAS Visual Analytics in order to analyze the performance of servers through dynamic data visualization

Informatics Specialist (summer intern) — May-July 2016

Department of Health Sciences Research, **Mayo Clinic**, Rochester, MN, USA

Project: Deployment of a web application to make the Pharmacogenomic test results publicly available via the QR-code

Research Assistant (summer intern) — May-Aug. 2015

Biomedical Informatics Research Center, **Marshfield Clinic Research Institute**, Marshfield, WI, USA
Project: Implementation of an Oral cancer risk assessment tool using data mining techniques and design and deployment of a prototypical web application for the tool

INDUSTRIAL EXPERIENCE

IT Expert 2007-2011
Ports and Maritime Organization, Tehran, Iran.
Job description: Supervising the development and maintenance of software systems (developed through outsourcing)

RESEARCH EXPERIENCE

Research Assistant June 2013- present
Department of Computer Science, University of Wisconsin-Milwaukee, WI, USA.
Project: 3D shape descriptor-based facial landmark detection- A machine learning approach

Research Assistant Sept. 2007 - Feb. 2010
Department of Computer Science and Information Technology, Amirkabir University of Technology, Tehran, Iran.
Project: Data mining to enhance the throughput of container ports.

TEACHING EXPERIENCE

Graduate Teaching Assistant Fall 2016, 2017, and 2018
Department of Computer Science, University of Wisconsin-Milwaukee, WI, USA.
Fundamentals of Computer Graphics (C++, OpenGL)

Graduate Teaching Assistant Spring 2014 - Spring 2018

Department of Computer Science, University of Wisconsin-Milwaukee, WI, USA.

Intermediate Computer Programming (Java)

Graduate Teaching Assistant Fall 2013

Department of Computer Science, University of Wisconsin-Milwaukee, WI, USA.

Introductory Computer Programming (Java)

JOURNALPUBLICATIONS

- R. Rostami, F. S. Bashiri, B. Rostami, Z. Yu, "A Survey on Data-Driven 3D Shape Descriptors", COMPUTER GRAPHICS *forum* (Published).
- Z. Gao, R. Rostami, X. Pang , Z. Fu, Z. Yu, "Mesh Generation and Flexible Shape Comparisons for Bio-Molecules", Molecular Based Mathematical Biology, 4:1-13, 2016.

CONFERENCE PUBLICATIONS

- R. Rostami, G. Bartowitz, X. Lin, D. Edge, "An Advanced Dynamic Analysis of Server Performance Using SAS Visual Analytics", WIILSU 2018, Milwaukee, USA, June 2018.
- H Chen, Y Guo, R. Rostami, S Fan, K Tang, Z Yu , "Porous Structure Design Using Parameterized Hexahedral Meshes and Triply Periodic Minimal Surfaces", Proceedings of Computer Graphics International 2018, Bintan Island, Indonesia, June 2018.
- R. Rostami, S. Hofer, K. Blagec, H. Xu, R. Kiefer, M. Samwald, R. Freimuth, "Sharing Clinical Pharmacogenomic Test Results via Mobile Devices". AMIA Joint Summits on Translational Medicine 2017, San Francisco, USA, May 2017.
- R. Rostami, H. Hegde, N. Shimpi, G. Pack, A. Acharya, "Oral Cancer Risk Assessment

Using Machine Learning Algorithms”. In: Proceedings of the 45th Annual Meeting of the American Association for Dental Research (AADR), Los Angeles, USA, March 16-19, 2016.

- R. Rostami, M. Matash Boroujerdi, ”Proposed Algorithm to Discover the Influential Factors in Enhancement of throughput of Queue-based Servicing systems”, 18th Iranian Conference on Electrical Engineering , Isfahan, Iran, May 2010.
- R. Rostami, M. Matash Boroujerdi,”Data Mining to Enhance the Throughput of Container Ports”, 9th International Conference on Computer Applications and Information Technology in the Maritime Industries, Gubbio, Italy, April 2010.
- R. Rostami, F. FathNezhad, F. Aminzade, B. Rostami, R. Rostami, ”An Artificial Decision Support System Based on Neural Networks to predict flood in a Special geographic Area”, 1st National Conference on Software Engineering Applications, Iran, March 2009.

POSTER PRESENTATIONS

- R. Rostami, Z. Yu, ”3D shape descriptor based facial landmark detection: A machine learning approach”, Poster Competition 2018, University of Wisconsin - Milwaukee, USA, April 2018.
- H. Hegde, N. Shimpi, G. Pack, R. Rostami, A. Acharya, “Smoking Status Classification Of Clinical Notes Using Natural Language Processing”. In: Proceedings of the 45th Annual Meeting of the American Association for Dental Research (AADR), Los Angeles, California, March 16-19, 2016.
- R. Rostami, Z. Yu, ”3D Point Matching in Face Modeling”, Poster Competition 2015, University of Wisconsin - Milwaukee, USA, April 2015.

HONORS AND AWARDS

Graduate Honorable Mention Poster April 2018

University of Wisconsin-Milwaukee poster competition, WI, USA.

Graduate Student Excellence Fellowship award (GSEF) Fall 2017-Spring 2018

University of Wisconsin-Milwaukee, WI, USA.

Grace Hopper Celebration travel award (BRAID) Fall 2014, Fall 2015

Department of Computer Science, University of Wisconsin-Milwaukee, WI, USA.

Ranked as top 2% students 2007

Nationwide universities entrance exam for M.S., Iran.

Ranked as top 1% students 2002

Nationwide universities entrance exam for non-continuous B.S., Iran.

Ranked 2^{nd} ***among students of Computer Science Department*** 2002

University of Zanjan, Zanjan, Iran.