Int J Comput Vis
DOI 10.1007/s11263-012-0605-9

A Machine-Learning Approach to Keypoint Detection and Landmarking on 3D Meshes

Clement Creusot · Nick Pears · Jim Austin

Received: 14 October 2011 / Accepted: 17 December 2012

Abstract We address the problem of automatically detecting a sparse set of 3D mesh vertices, likely to be good candidates for determining correspondences, even on soft organic objects. We focus on 3D face scans, on which single local shape descriptor responses are known to be weak, sparse or noisy. Our machine-learning approach consists of computing feature vectors containing D different local surface descriptors. These vectors are normalized with respect to the learned distribution of those descriptors for some given target shape (landmark) of interest. Then, an optimal function of this vector is extracted that best separates this particular target shape from its surrounding region within the set of training data. We investigate two alternatives for this optimal function: a linear method, namely *Linear Discriminant Analysis*, and a non-linear method, namely *AdaBoost*. We evaluate our approach by landmarking 3D face scans in the FRGC v2 and Bosphorus 3D face datasets. Our system achieves state-of-the-art performance while being highly generic.

Keywords keypoint detection · landmarking · 3D face recognition · Machine learning · LDA · AdaBoost

C. Creusot (✉) · N. Pears · J. Austin
Department of Computer Science, University of York, York, UK
e-mail: clementcreusot@gmail.com

N. Pears
e-mail: nick.pears@york.ac.uk

J. Austin
e-mail: jim.austin@york.ac.uk

1 Introduction

Searching for corresponding points (correspondences) across a pair of shapes is an essential early process for a large number of applications involving 2D images, 3D meshes and multimodal systems that use a mixture of both. For example, in stereo vision, structure-from-motion and tracking objects in video, we need to establish pairs of image points that correspond to the same scene point. Establishment of 3D point correspondences is often required for 3D shape registration and matching, particularly in the case of partial views (the rear faces of 3D shapes are not captured in 2.5D scans). These processes are essential components of 3D object retrieval and 3D shape recognition systems.

Since the determination of correspondences is a search problem, with $\mathcal{O}(n^2)$ complexity, a reduction in the search space by detecting a sparse set of *keypoints*[1] is common practice. The use of corner detectors is the classic example of this in 2D images, while a locally extremal value of Gaussian curvature is commonly used on 3D meshes.

Note that a keypoint is an *unlabeled* point and a good keypoint detector generates outputs at pairs of points with locally similar shapes across a pair of similar 2D images or 3D scans. At the same time, the detector should generate as few as possible outputs in the overlapping region of the two images or scans that are not part of a valid correspondence.

Often keypoints are used directly to determine correspondences between a pair of 2D images or a pair of 3D scans of the same scene object. Keypoints can also be used to initialise the pose and shape parameters of a generic shape model, such as a 3D morphable face model. In this case, a sparse set of reliably detectable landmarks is selected (usually manually) on the generic shape model and these are the points that we

[1] Also called *interest points* or *feature detections* in other literature.

aim to match extracted keypoints to. We can view the sparse set of landmarks as a subset of a relatively dense generic shape model and we encapsulate this small subset of points and their relative positions in an entity called a *landmark model*, $\mathcal{L}$. This model is annotated with L landmarks, where landmarks have both a position and a label. Hence, in contrast to keypoints, they are *labeled* points.[2] When keypoints are the inputs of a labeling system, they are seen as *landmark candidates* and are usually associated with a list of possible candidate landmark labels. Often, the initial mapping between query scan keypoints and model landmarks is many-to-many. However, once a one-to-one correspondence is established between a keypoint on a query scan and a landmark on the model, $\mathcal{L}$, a keypoint may 'acquire' the associated model label and is then upgraded to a landmark.[3]

Thus the landmarking process usually contains two main parts: detecting keypoints on a query, which usually provides many landmark candidates, and eventually selecting n of these, up to a maximum of the total number of model landmarks, the actual number depending on the degree of occlusion on the query scan. Detecting the keypoints and computing the correspondences with the landmark model's labels are two different problems. The keypoint detection can be seen as a local-only (featural) problem while the correspondence search takes into account both local (featural) and global (configural) information. This paper is mainly focused on the *detection* of keypoints on 3D scans as probable landmark candidates. However, in Sect. 6, we present a method that uses our keypoint detector in a complete 3D face landmarking system and we compare it to the state-of-the-art in order to demonstrate the high performance and generality of our approach.

Captured 3D scans have several advantages over 2D images; for example, scale changes are often not so pronounced for some object class and it is easier to deal with pose and illumination variations. However, they also have a set of specific drawbacks. For example, if one compares a 2D image and 3D depth map of a soft, organic structure such as a human face, there appears to be a larger number of salient structures in the 2D image, which could be localized by standard 2D keypoint detectors. However, the application of a standard keypoint detector on a 3D scan, such as locally extremal Gaussian curvature, leads to a highly sparse set of keypoints. This is the main motivation for building our machine learning based keypoint detection system. An important aspect of our keypoint detector, that allows its general deployment, is that it can generate useful keypoints over large regions of soft, organic shapes, as there is no requirement for a keypoint to exist at the locally extremal value of any single, scalar local shape descriptor[4] (e.g. Gaussian curvature).

Rather, a set or *dictionary* of L local shapes is learned, where each member of the dictionary is associated with a particular landmark that is labeled across a set of training meshes. Subsequently, keypoints can be generated on query meshes, where the local shape is highly similar to one of the members of this dictionary.

Although many researchers have developed systems to detect facial features in 2D images, using well-defined machine learning techniques, researchers working with 3D faces have almost exclusively developed heuristic approaches that rely on restrictive assumptions, such as a near-frontal pose. These approaches are usually landmark dependent sequential recipes taking advantage of patterns detected by their designers. Such a recipe can be, for example, to take the tip of the nose as the most extreme point along one direction or as the maximal convex curvature in the query scan. These recipes give very good results for very salient points on most existing datasets, but are bound to fail on some scans of non-cooperative subjects; for example, when the assumption of a near-frontal pose is broken or when the nose tip is occluded.

In summary, the main contribution of this paper is a generic keypoint detection process for 3D meshes, based on machine learning, which is then employed in a 3D face landmarking system in order to improve upon existing heuristic approaches. In our system, detector functions are learned rather than being enforced by the designer of the system. This allows:

1. *generality*, as all of the landmarks are processed with the same general framework;
2. a more *reliable detection of landmarks* when the input data is not cropped (e.g. when there are non-facial elements in a face capture);
3. an *intrinsic robustness to missing data*, because we do not follow a sequential progression through the landmarks. Most other landmarking systems will fail if the nose or the inner eye corners are missing, as these strong features are often used to search for weaker, less salient features;
4. the *detection of less salient features* for which human designers struggle to create detection functions or rules (e.g. corners of the mouth and outer corners of the eyes on the human face).

[2] It is essential that the reader distinguishes carefully between *unlabeled* keypoints and *labeled* landmarks throughout this paper.

[3] To be more precise, a query scan point close to an extracted keypoint is sometimes designated to be the landmark, in order to minimize the least-squares error when fitting model $\mathcal{L}$. This is discussed in Sect. 6.

[4] We define a scalar local shape *descriptor* as a real number that describes the shape of the local neighborhood surrounding some mesh vertex. In some literature, this is termed a *feature* or *feature descriptor*. Here, the local neighborhood is Euclidean and enclosed by a sphere of predefined radius.

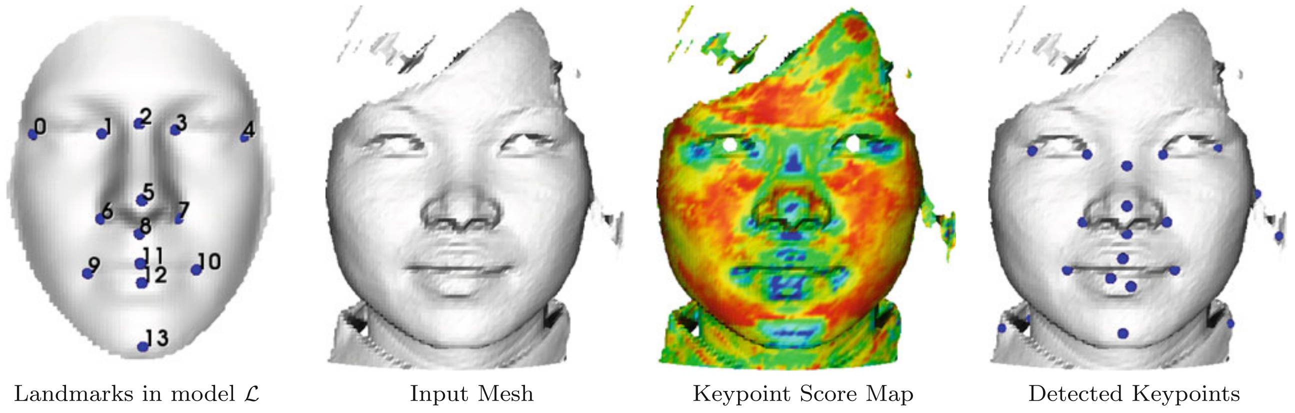
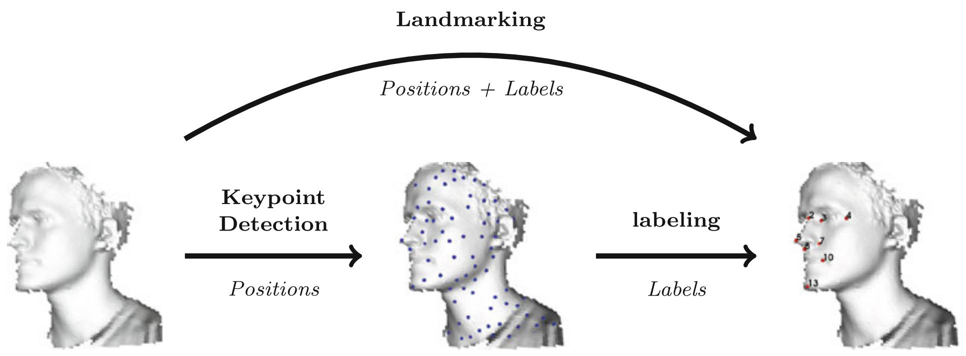


Fig. 1 Example of keypoint detection using our method on a 3D scan from the FRGC dataset

Fig. 2 Problem breakdown. The landmarking problem is split into two sub-problems that are solved independently: keypoint detection and labeling. Although this paper mainly focuses on solving the keypoint detection problem, we also apply our technique in a landmarking problem



Specific contributions on a practical level are:

1. a new method for keypoint detection on meshes, using a dictionary of L learned local shapes (see Fig. 1);
2. an evaluation of its performance using two different approaches to generate functional forms of local shape descriptors, linear (LDA) and non-linear (AdaBoost);
3. a new framework for 3D mesh landmarking based on automatically detected keypoints (see Fig. 2).

The work presented here is an extension of our work in Creusot et al. (2011) and is structured as follows. In the first section, previous work on keypoint detection and landmarking of 3D meshes is reviewed. In the following overview section, Sect. 3, we give our problem definition and we outline the offline training and online testing processes of our solution. Here, we also discuss our datasets and performance metrics used for evaluation. In the following section, we present the fine detail of our machine learning approach to keypoint detection on 3D meshes. In Sect. 5, we evaluate this keypoint detection system, while the following section describes its application to landmarking. A final section is used for conclusions.

2 Previous Work

Unlike tracking and stereo-vision applications, or matching generic salient points across 3D shapes, landmarking requires candidate positions that are close to the targeted human-defined landmarks required for a particular application. This is usually achieved through an expert system approach: the system designer (expert) will notice a correlation between the local shape of the mesh and the targeted landmark and use this observed correlation to define some heuristic rule to select candidate positions on new queries. For example, the expert will observe that the position of the nose tip is often correlated with the highest convex curvature points in the central facial area or that the nose is the closest point to the camera or the most extremal point in a particular direction. Unfortunately, rules extracted from such observations won't always work in the general case, as the input may contain artifacts other than the face, such as hair, hands and accessories (e.g. hats, glasses).

2.1 Landmark Candidate Detection on 3D Faces

A scalar descriptor of some specific type can be computed at every vertex of a mesh. Typically these values are color mapped and rendered over the mesh to give a visualization of how the descriptor responds to the local shapes within the mesh. Such a visualization is termed a *descriptor map* (or, in some literature, a feature map). It can be thought of as a scalar field over the scanned object's surface that is sampled at the vertices of the mesh.

The use of descriptor maps on 3D meshes, such as Gaussian curvature maps, is common practice to detect landmark candidates. A typical multiple landmark detection approach is presented in Colbry et al. (2005). First, the authors preprocess the face to remove spikes, before cropping the upper part of the scan as being the region of interest. Then the nose is localized as the closest vertex to the camera, or the most extreme point in a particular direction (left or right), or the one with the largest shape index. The inner eyes corner are detected as the points with the smallest shape index. This kind of approach has a lot of variants and is widely used in both academic and commercial systems.

Other authors have noticed that the sagittal slice of the face remains identical over orientation changes and therefore can be used to detect the nose. In Faltemier et al. (2008), contours of the mesh are extracted at varying angles until it matches a previously learned nose profile signature. The system achieved a 98.52 % accuracy on the NDOff2007 3D face dataset, for the nose tip with variations of angle up to 90°. (This dataset contains a total of 6,911 non-frontal images containing neutral expressions and a single frontal neutral image for each of 406 distinct subjects.) Some non pose-invariant techniques have also used transverse slices to detect the nose tip and the nose corners (Segundo et al. 2007; Mian et al. 2006).

To summarize, most papers on 3D face landmarking have their keypoint detection system grouped in one of the following categories:

- *curvature/volume extrema*: the candidates are defined as extrema over curvature and/or volume based descriptor maps (Chang et al. 2006; Colbry et al. 2005; D'Hose et al. 2007; Pears et al. 2010; Romero and Pears 2009; Segundo et al. 2007; Szeptycki et al. 2009).
- *directional extrema*: the candidates are defined as the extremal points in given directions (Chang et al. 2006; D'Hose et al. 2007). This is only used for the nose tip detection.
- *2D curve extrema*: by using profiles and slicing of the mesh, the detection of salient points is reduced to finding extremal points along a two-dimensional curve (Faltemier et al. 2008; Mian et al. 2006; Segundo et al. 2007).

Several previous studies have acknowledged the limitations imposed by heuristic approaches and employed machine learning techniques instead (Berretti et al. 2010; Zhao et al. 2011). However, they usually employed 2D descriptors on depth maps making their systems unusable in scenarios presenting a large rotation from the frontal view. To the best of our knowledge, it appears that no 3D machine learning method exists for facial landmark candidate (keypoint) detection (see Fig. 3). This is a gap in the literature that we aim to fill, to enable better landmarking on face scans of non-cooperative subjects. Clearly, this has great utility in high-throughput 3D face recognition systems that do not require subject cooperation. Our proposed system is sufficiently generic to be applied to meshes of other general classes of objects, in any application where landmarks of interest can be manually defined on a set of training scans.

2.2 Keypoint Detection on Faces

Keypoints on 3D faces are often not labeled and, typically, they are used for the purpose of face recognition. In these cases, the desired keypoints are repeatable for a given identity but differ from one individual to another. For example, Mian et al. (2008) use a coarse curvature-related descriptor to detect such keypoints, while in Berretti et al. (2010) and Mayo and Zhang (2009), they are computed using the Scale-Invariant Feature Transform (SIFT Lowe 2004) on 2D depth-maps.

In this paper, the term *keypoint* is justified as we try to detect unlabeled repeatable point of interest. However, our approach differs, as the scope for the targeted repeatability is larger. Our technique should be able to detect repeatable point of interest across the population and not only for several captures of the same individual. Therefore, our system is designed to extract macro-features (nose, eyes, mouth) common across a whole population of objects (faces), instead of discriminative micro-features (e.g. wrinkles) that are often specific to individuals.

2.3 Keypoint Detection and Landmarking on Other Objects

Computing keypoints in order to determine correspondences is useful for all kinds of object matching applications. Shape retrieval (e.g. in web search applications) is one such application and there is now a robust feature detection and description benchmark, the SHREC benchmark (Boyer et al. 2011), that tests the performance of feature detectors and descriptors under a variety of transformations, such as scaling, affine and isometry (bending) transformations, and a variety of noise conditions.

In Mian et al. (2010), keypoints are computed using a coarse curvature descriptor to localize objects in scenes with occlusions. In Zaharescu et al. (2009), an approach called Mesh DoG is presented. This is a multi-scale approach that makes use of Difference-of-Gaussians (DoG), and thus has similarities to the DoG approach applied to 2D images in

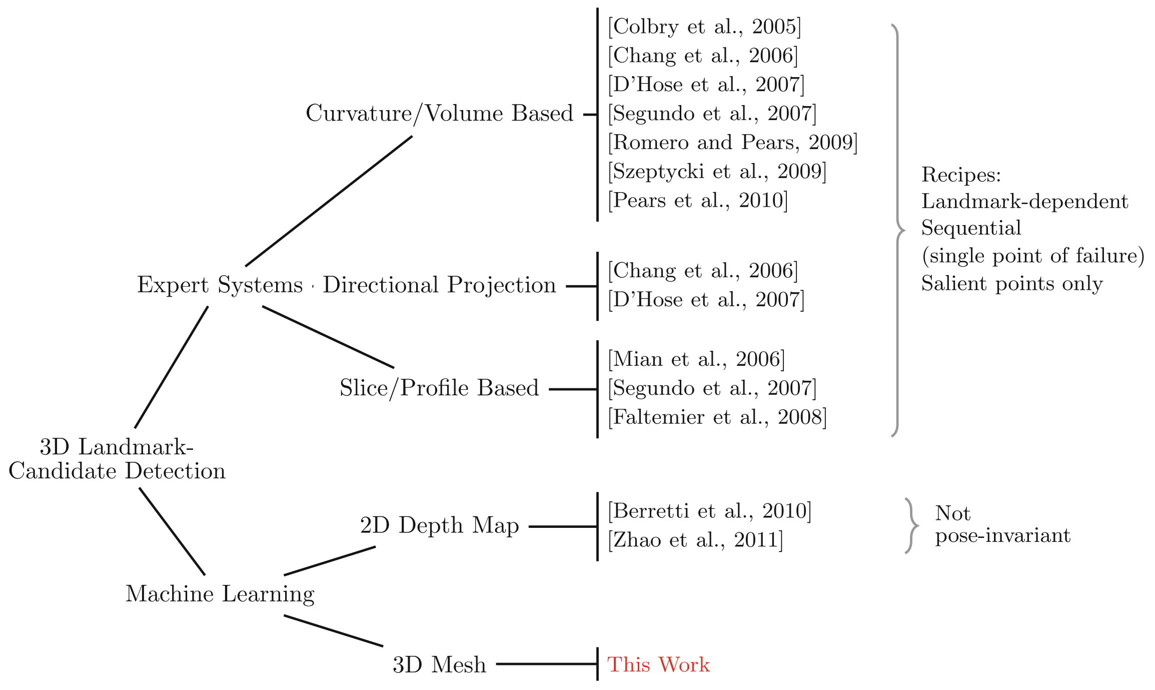


Fig. 3 Related work: facial landmark candidate detection usually appears as a single section in face landmarking papers. Most of them are sequential recipes. Some use machine learning techniques but in these cases, they are always with 2D representations (e.g. SIFT Lowe 2004 on depth maps)

the SIFT descriptor (Lowe 2004). In this approach, any surface descriptor map can be convolved with a set of Gaussian kernels of different scales (standard deviations). Subtracting convolutions across two adjacent scales gives the DoG operator response. Keypoints are then extracted as the local maxima across scale space, using non-maximal suppression in a one-ring neighborhood in the current and adjacent scales. A similar DoG-based approach is presented in Castellani et al. (2008). However, here the DoG operator is applied to the actual mesh over a range of scales. The amount that a vertex moves between Gaussian filtering at one scale and the next is projected along the vertex normal. Keypoints are extracted as points of maximal normal movement over local neighborhoods and local scales.

In Ben Azouz et al. (2006), landmarking of 3D human models is addressed. Here a graphical model is created, with landmarks at the graphical nodes being characterized by spin images (Johnson and Hebert 1999). Functions express the likelihood that a particular landmark corresponds to a given vertex on the query mesh (based on the distribution of learned spin images in the model). Other functions constrain a pair of landmarks to be consistent with their learned spatial relationship in the model. To perform the optimization of landmark assignment, the authors employ a belief propagation algorithm. An interesting aspect of this approach is that it does not use a keypoint extraction phase to reduce computation time. Encouraging results are obtained on a very small test set (30 scans).

In Itskovich and Tal (2011), two kind of curvature-related descriptor (shape index and Willmore energy) are combined to detect the keypoints on archaeological objects in order to detect regions matching a given pattern. This paper is one of the rare case where more than one local shape descriptor is used for the keypoint candidate selection. An other example is Dibeklioglu et al. (2008) where shape index, difference map, and image gradient are combined for landmark localization. Besides, when several such scalar descriptors are used, combining them is usually done using fixed coefficients.

In this paper, a framework is presented to determine automatically how descriptors should be combined for each landmark in landmark model $\mathcal{L}$, designed for the specific problem of 3D face landmarking.

3 Overview

In this section, we give the problem statement, outline our solution and the methodology for its evaluation.

3.1 Problem Statement

The main problem that we address is *the detection and subsequent labeling of points on input 3D meshes that are locally similar to at least one member of a set of L predefined landmarks in a landmark model* (see Fig. 1).

This statement implies that the landmarked points should be a subset of the input mesh's vertices. This can be justified in our case by the fact that the resolution of the input mesh is good enough compared to the acceptable error in positioning. 'Sub-vertex' localization of landmarks (i.e. to a higher resolution than the input mesh) is an interesting problem for landmark refinement techniques, but is not discussed in this paper. Rather, our aim is to enable the system to automatically learn functions of a set of D local shape descriptors, extracted over a set of training meshes, that enable robust keypoint detection on unseen input meshes.

3.2 Outline of the Keypoint Detection System

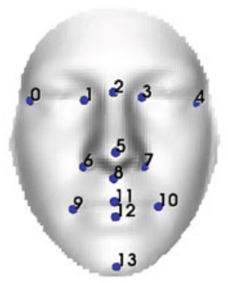
We define $L = 14$ landmarks in a landmark model $\mathcal{L}$, as shown in Fig. 4. In each of a set of N training meshes, these L landmarks are manually localized with a point-and-click interface.

To implement our system, we need to define what kind of local shape descriptors to use. The use of a single *scalar* 3D local descriptor on its own is usually not very discriminative. Therefore, we use a set of D (10–48) scalar shape descriptors including, for example, Gaussian curvature, mean curvature and a volumetric descriptor (details given in Sect. 4.1). After normalization to a set of scores (which range from 0 to 1), these form a feature vector that describes local shape at some mesh vertex. (It can be thought of as a D-dimensional vectorial descriptor, which is composed of a collection of scalar descriptors.)

Our framework is composed of an offline training process and an online keypoint detection process, as illustrated in Figs. 5 and 6 respectively, and we now describe each of these processes in turn.

3.2.1 The Offline Training Process

Figure 5 shows the offline training process, which is used to teach the system what is considered to be a shape of interest. The inputs to this training system are a set of N training meshes and an associated set of L landmarks per



0 - Left exocanthion (Outer Eye Corner)
1 - Left endocanthion (Inner Eye Corner)
2 - Nasion (Nose Bridge)
3 - Right endocanthion (Inner Eye Corner)
4 - Right exocanthion (Outer Eye Corner)
5 - Pronasale (Nose Tip)
6 - Left Alar Crest Point (Nose Corner)
7 - Right Alar Crest Point (Nose Corner)
8 - Subnasale
9 - Left Cheilion (Mouth Corner)
10 - Right Cheilion (Mouth Corner)
11 - Labiale Superius (Upper lip)
12 - Labiale Inferius (Lower lip)
13 - Pogonion (Chin)

Fig. 4 Position of the 14 landmarks, centers of our local shapes of interest for the training part of the system

training mesh, where each specific landmark, λ (e.g. nose tip), has a vertex index, i, within a training mesh. Within our system, D local shape descriptors are defined and we can process an arbitrary number of such descriptors, up to some limit imposed by memory and computation time limitations. The following four broad steps outline our training process.

1. For every training mesh, D descriptor maps are generated, where a descriptor map is defined as the raw descriptor values, x_d, $d \in \{1 \ldots D\}$, computed over all vertices of a training mesh.
2. Then, for each landmark, $\lambda \in \{1 \ldots L\}$, we collect the N raw descriptor values over the N training meshes and estimate the parameters of their distributions (we primarily use Gaussian distributions, see Sect. 4.2.1).
3. These learned statistical distributions allow us to map raw *descriptor values* into normalized *descriptor-landmark scores* or DL-scores. (In terms of visualizations, DL-score maps are generated from descriptor maps.) Such scores are specific to a descriptor-landmark, (d, λ), pair and hence there are $D \times L$ of these per vertex per training mesh (see Fig. 5). To generate these scores, over all vertices of a training mesh, the raw values, x_d, of descriptor d are projected against the distribution of x_d at landmark λ and normalized by the probability density function (pdf) maximum of the distribution. Scores close to 1 are obtained if the descriptor value is close to the modal value of the distribution, and scores close to zero are obtained if the descriptor is far from that modal value.
4. Effectively, these D scores, computed with respect to the distributions at some landmark, $\lambda \in \{1 \ldots L\}$, form a D-dimensional feature vector at every vertex across all training meshes. We can form two classes of such feature vectors: those from vertices that are close to landmark λ across all of the training meshes and those from surrounding vertices that are more remote from the same landmark. An example of the two classes employed is shown in Fig. 7, where neighboring vertices of the upper lip landmark are shown in blue and the non-neighboring class of vertices is shown in red. (Note that this is only shown for one face scan, but these classes are formed by the union of all such vertices across the full training set.) We then learn the function operating on these feature vectors that best discriminates between these two classes. In effect, this is a detector function, f_λ, and there is one for each landmark, λ. In the case of LDA, this detector function is a linear combination of the elements of the feature vector, whereas a non-linear function is generated using AdaBoost.

To summarize the training process, L landmarks labeled over each mesh in a training set are used to define a dictionary of local shapes where, for each shape, both the statistical distributions of shape descriptors and the rules for combining such descriptors are learned. The concept of this dictionary is illustrated by the red boxes in Figs. 5 and 6

It is important to note that the normalization of DL-scores in step 3 is not *essential* to apply our machine learning processes in step 4. Learning of the detector function could be applied directly to the raw descriptor values, as one might expect when using a powerful classifier such as AdaBoost. In fact in Creusot (2011)[p. 150–152], we compare detector functions learned from scores against those learned from raw descriptor values. We found that the performance was very marginally better for scores, but the difference was so small as to not be significant. (In more detail: for 6 landmarks, scores gave marginally better performance, for 3 landmarks, raw descriptors gave marginally better performance and for 5 landmarks the performance was almost identical.) However, improved detector functions was never the intention of using

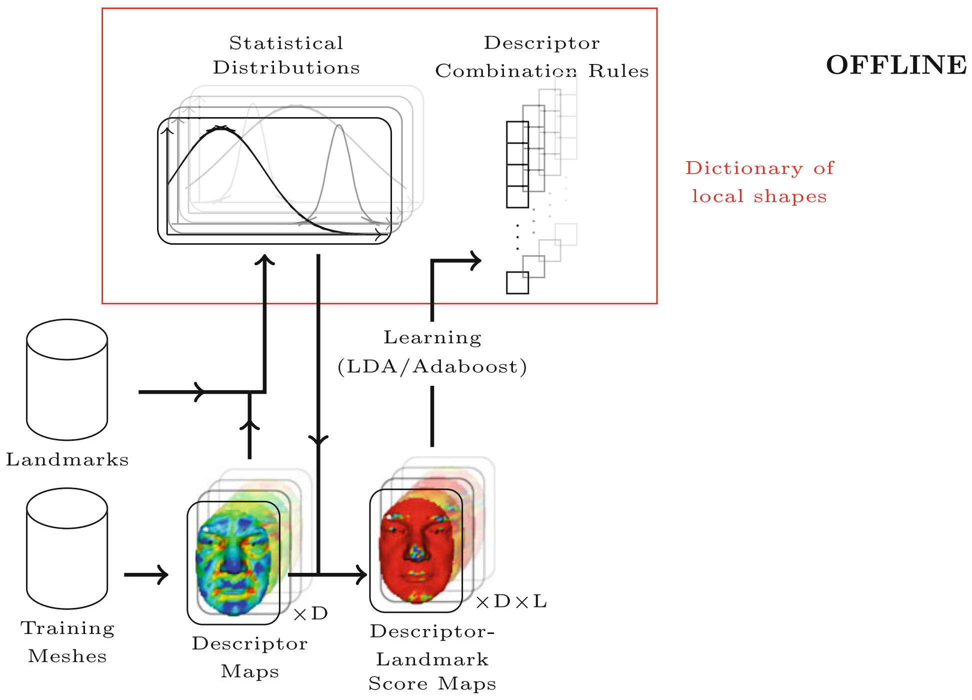

Statistical Distributions
Descriptor Combination Rules
OFFLINE
Dictionary of local shapes
Learning (LDA/Adaboost)
Landmarks
Training Meshes
×D
Descriptor Maps
×D×L
Descriptor-Landmark Score Maps

Fig. 5 Offline process: known landmark positions on the training set are used to learn idealized distributions (parameterized class-conditional probability density functions) of the descriptor values for each shape of interest. The matching scores computed using those distributions are used to train the descriptor combination system for every landmark as explained in Sect. 4.3

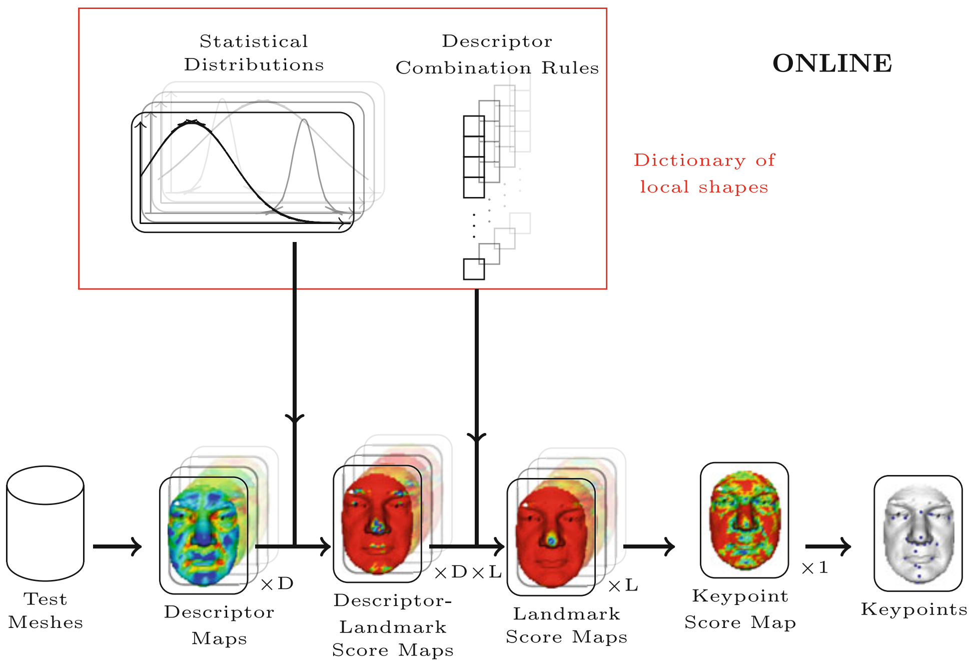

Statistical Distributions
Descriptor Combination Rules
ONLINE
Dictionary of local shapes
Test Meshes
×D
Descriptor Maps
×D×L
Descriptor-Landmark Score Maps
×L
Landmark Score Maps
×1
Keypoint Score Map
Keypoints

Fig. 6 Online process: D descriptor maps are computed from the input mesh, each value is matched against the L learned descriptor distributions (14) to get score maps with values between 0 and 1. For each landmark, the D descriptor score maps are combined using the learned combination rules and renormalized to span the range (0–1). The L normalized landmark score maps are combined into a single final keypoint score map, using the maximal value (of L values) at each vertex. The output keypoints are the strongest local maxima detected on this keypoint score map that are above some given threshold T

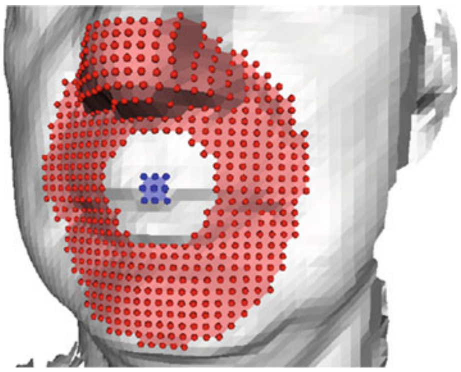


Fig. 7 Example of vertex class generation, showing neighboring (*blue*) and non-neighboring (*red*) vertices for the upper-lip landmark on one face of the training set

scores when we developed our framework. Rather, we claim the following advantages:

– Scores normalize descriptor values to compatible ranges, thus giving a great deal of insight to the researcher about the inner workings of the classification technique allowing us to rapidly know which descriptors are good to use and for what landmarks they should be employed (Creusot et al. 2011).
– Scores make visualization of results much easier, as witnessed by the colormap graphics in this paper.
– Scores allow histogram descriptors and scalar descriptors to be used in the same framework at the same time. All types of descriptor are converted to the same space (a unit hypercube), which make it possible to use very heterogeneous types within the same framework. This is a real strength of our framework in terms of its extensibility.

3.2.2 The Online Keypoint Detection Process

The online part of our system takes a previously unseen face mesh as its input. The process is composed of the following stages, see Fig. 6.

1. D local shape descriptors (scalars, e.g. Gaussian curvature) are computed for all V vertices in the input mesh.
2. For each of the L learned local shapes:
   – D Descriptor-Landmark score (DL-score) maps are computed by projecting the descriptor values of each vertex against the associated learned distributions of the target landmark. The scores generated are between 0 and 1. A value of 1 is generated when the value of the descriptor is equal to the maximum of the learned distribution (modal value). These normalized scores are collected together into a D-dimensional descriptor, or feature vector, at each vertex of the input mesh.
   – Using the learned detector function, f_λ, for landmark λ in the training phase, a (scalar) response is generated, which we call a *landmark score*. The landmark score map is then normalized over the mesh to ensure the matching scores associated with different landmark shapes have the same impact on the final keypoint extraction stage. This normalization ensures that the landmark score map has values ranging from 0 to 1.
3. All of the landmark score maps are combined into one final keypoint score map, by using the maximum value (over all L landmark dictionary shapes) for each vertex (see Fig. 14).
4. The keypoints are defined as the strong local maxima on this final keypoint score map. A threshold T is used to discard weak candidates. The maximum number of keypoint retained is limited to 1 % of the total number of vertices.

This completes the outline of our system. A more detailed description of our keypoint detection system is presented in Sect. 4 and describes the descriptors employed and implementation details. The remainder of this section describes the datasets (Sect. 3.3) and performance metrics (Sect. 3.4) that we use for evaluation.

3.3 Datasets

Our keypoint detection system, described above, is tested on the Face Recognition Grand Challenge version 2 (FRGC v2) dataset (Phillips et al. 2005) containing 4,950 faces of 557 individuals. This data contains both males and females with some variation in ethnicity, age and expression as well as small variations in pose (under 10°). The dataset is fairly uneven in terms of capture per identity with some individuals appearing only once while others appear thirty times. This dataset is widely used in the research community and, over the years, has become a standard benchmark for 3D face processing systems.

The input data for our system is lower resolution than the original structured point cloud (640 $\times$ 480 vertices). The point density is reduced by replacing each block of 4×4 raw 3D data points with its average. A mesh is then created by defining two triangular faces for every group of four adjacent vertices (∷ $\Rightarrow$ ⊠) .

For the landmarking experiments, both the FRGC and the Bosphorus dataset Savran et al. (2008) are used. The Bosphorus dataset contains 4,666 captures of 105 people. Unlike the FRGC, it contains large variations in pose and occlusions (hands, hair and glasses partially covering the face). In the landmarking experiments, for which time performance is measured, the point density of the original input data in both datasets is reduced by binning the points into square pixels of fixed size 3.5 mm (⊞ ⇒ ⁘) . Compared with the averaging method, this allows us to reduce even further the point density (and therefore the computation time), while keeping a fixed resolution whether the face is captured from a close or remote position.

For the FRGC dataset, 200 neutral expression scans of different individuals are selected randomly as our training set and all remaining scans are used as a test set (4,750 faces). The manually-derived 'ground truth' landmarks are a mixture of contributions from Romero-Huertas et al. (2008) and Szeptycki et al. (2009) with additional manually located landmarks and refinements.

For the Bosphorus dataset, only 99 neutral expression frontal scans are used as training, leaving 200 faces in the neutral-frontal test set and 4339 scans in total in the global test set.[5] The ground-truth landmarks are the ones provided with the dataset (Savran et al. 2008) with some manual additions (subnasale, nasion) and refinements.

3.4 Performance Metrics

We define three performance metrics to evaluate the performance of our systems: the *landmark retrieval rate*, the *landmark positioning error* and the *global registration error*. Each of these is described in the following three subsections.

3.4.1 Landmark Retrieval Rate

We define the *landmark retrieval rate* for a particular landmark as the percentage of test scans in which the system correctly retrieved its position given an error acceptance radius. For example, if the test set contains 1,000 models, among which 990 have a ground truth landmark for the nose tip, and if we use an error acceptance radius of 10 mm, the landmark retrieval rate will be the percentage of the 990 models which present a detected *nose tip* landmark within 10 mm of the known nose tip position.

Usually it is not clear what radius should be used for such evaluation. As a good practice and to facilitate results comparison with other researchers, the retrieval rates are provided for a varying acceptance radius (usually increasing from 2.5 mm to 25 mm in steps of 2.5 mm).

3.4.2 Landmark Positioning Error

Computing the distance from every localized landmark to the ground truth position of the corresponding label is an interesting measure for the global landmark localization framework. However this continuous measure is more meaningful for landmark positioning refinement than for coarse landmark localization. This measure doesn't allow notions of discrete failure and will only be used in the last section where a complete landmarking system is presented.

3.4.3 Global Registration Error

We can register the detected landmarks with the ground truth landmarks in the same scan. If a scale-adapted rigid registration is used, we expect to see zero rotation, zero translation and a unity scale in the ideal case. Any deviation from the ideal values gives a global measure of the landmarking performance. This give us an interesting performance metric for the overall matching.

4 A Machine Learning Approach to Keypoint Detection

In this section, the implementation details of our keypoint detector are presented. When reading this section, it is useful to keep the system overview presented in Sect. 3.2 and associated figures, Figs. 5 and 6, in mind.

4.1 Computation of Descriptors

Our system makes extensive use of local shape descriptors; in this section, details about their computation are presented. In our experiments, two kinds of descriptors are used, scalar-valued descriptors (e.g. Gaussian curvature) and vector-valued descriptors. This latter form of descriptor is a local shape histogram, such as a spin image (Johnson and Hebert 1999).

4.1.1 Normals

Several of the descriptors require a normal defined at every vertex. To compute the normals, a simple method using the adjacent triangle faces is used. When the mesh has been built from the 2D depth map, all the triangle faces have been defined anti-clockwise with regard to the camera position. Therefore all the normals of the faces are pointing outward. When setting the normal at a vertex i, a weighted sum of the normals of the neighboring faces is computed. The weights given for each of the adjacent faces are computed using the Nelson Max technique (Max 1999):

$$c\mathbf{n}_i = \sum_{j=1}^{n_i^e} \frac{\mathbf{e}_j \times \mathbf{e}_{j+1}}{|\mathbf{e}_j|^2 |\mathbf{e}_{j+1}|^2} \qquad (1)$$

[5] Subsets R_90, L_90 and IGN of the Bosphorus dataset are not used in this paper.

where $\mathbf{n}_i$ is the normal vector at vertex i, c is a constant that disappears after unit-length normalization, n_i^e is the number of edges adjacent to vertex i and $\mathbf{e}_j$ the vector corresponding to the jth neighboring edge. As the normal is computed at every vertex, the efficiency of the process is improved by looping over the triangle faces and accumulating the face normals with the corresponding weights on the three vertices of the face. This guarantees that the face normals are computed only once (instead of three times with a naive algorithm looping on vertices).

4.1.2 Neighborhoods

A point by itself contains very little information: only its position in the space. To extract more information one needs to know how this point is positioned with regard to the other points within its local neighborhood. In other words, a quantitative description of *local shape* needs to be extracted in a way which is invariant to the orientation (pose) of the scanned object (face) with respect to the 3D camera. This is a function operating on a set of vertices contained within a local neighborhood.

To determine this local neighborhood,the only thing that is needed is a metric and a threshold. Here we use a simple Euclidean metric to determine the neighborhood (see Fig. 8) i.e. vertices need to be within a bounding sphere of some specified radius that defines neighborhood size. The choice of neighborhood size is a compromise. A neighborhood that is too small with respect to the raw data resolution will lead to local surface properties being noisy or undetermined. On the other hand, if the local region is too large, the notion of locality is broken and the descriptor values become vulnerable to occlusions. Since the neighborhoods depend on a fixed distance, every descriptor using these neighborhoods cannot be invariant to the scale of the scanned object.

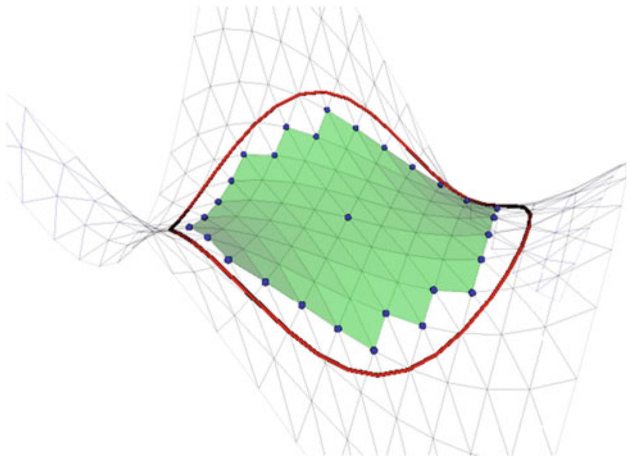


Fig. 8 Neighborhood computed using Euclidean distance. The red line is the intersection of the sphere of radius R with the surface. Every point inside the sphere is part of the local neighborhood. The *blue* vertices represent the perimeter (Color figure online)

4.1.3 Principal Curvatures

The curvature is a simple notion of 2D geometry that measures the bending of a curve at a particular point. It is defined as the inverse radius of the osculating circle at that location. This 2D notion can be used with points on a 3D surface by extracting 2D plane curves at those points using an intersecting plane. This intersecting plane always contains the normal, leaving one degree of freedom, the angle around the normal, and therefore an infinite number of possible 2D curves and curvature values. To provide a simple measure of the 3D surface curvature, only two angles are selected: the ones giving the maximal and minimal curvature values known as first (k_1) and second (k_2) principal curvatures (see Fig. 9 for these descriptor maps).

Computing these two values for a discrete surface is not trivial. The curvature computation approach employed is the *Adjacent-Normal Cubic Approximation* method proposed in Goldfeather and Interrante (2004).

4.1.4 Descriptors Derived from Principal Curvatures

The two principal curvatures k_1 and k_2 are rarely used directly. Most of the time they are used to compute other descriptors. A list of the most commonly used ones is given below:

– Gaussian Curvature (K):

$$K = k_1 k_2 \tag{2}$$

– Mean Curvature (H):

$$H = \frac{k_1 + k_2}{2} \tag{3}$$

– Shape Index (SI): two variants

$$\mathrm{SI}_{0,1} = \frac{1}{2} - \frac{1}{\pi} \arctan \frac{k_1 + k_2}{k_1 - k_2}, \quad 0 \leq \mathrm{SI}_{0,1} \leq 1 \tag{4}$$

or

$$\mathrm{SI}_{-1,1} = \frac{2}{\pi} \arctan \frac{k_1 + k_2}{k_1 - k_2}, \quad -1 \leq \mathrm{SI}_{-1,1} \leq 1 \tag{5}$$

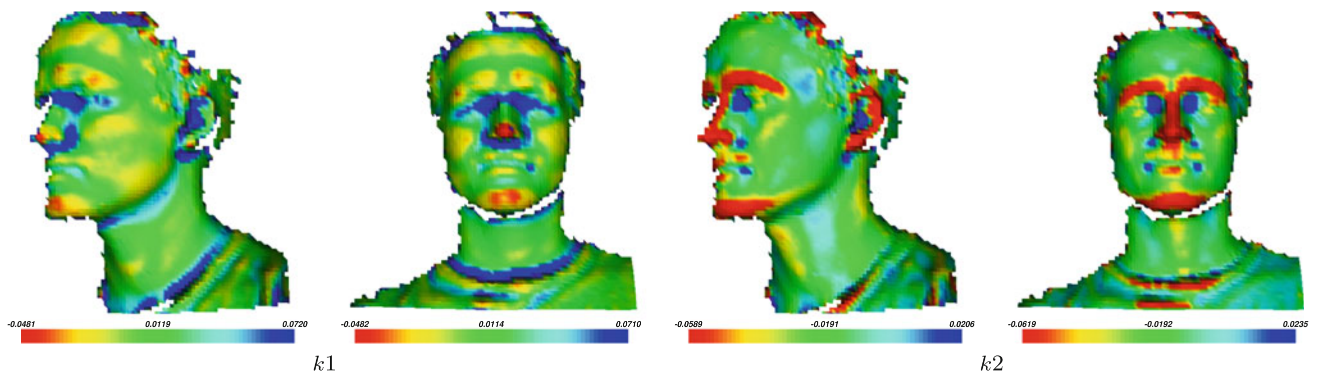


Fig. 9 Examples of descriptor maps for the first (k_1) and second (k_2) principal curvatures ($R^{Eucl.}_{scalar} = 15$ mm) (Color figure online)

– Curvedness (C):

$$C = \sqrt{\frac{k_1^2 + k_2^2}{2}} \tag{6}$$

– Log-Curvedness (LC):

$$\text{LC} = \frac{2}{\pi} \log \sqrt{\frac{k_1^2 + k_2^2}{2}} \tag{7}$$

– Willmore Energy (W):

$$W = H^2 - K = \frac{(k_1 - k_2)^2}{4} \tag{8}$$

– SC (Kim et al. 2009):

$$\text{SC} = \text{SI}_{-1,1} \cdot LC = \frac{4}{\pi^2} \log \sqrt{\frac{k_1^2 + k_2^2}{2}} \arctan \frac{k_1 + k_2}{k_1 - k_2} \tag{9}$$

– Log Difference map (Dibeklioglu et al. 2008):

$$\text{LD} = \ln(K - H + 1) = \ln(k_1 k_2 - \frac{k_1 + k_2}{2} + 1) \tag{10}$$

Figure 10 shows some examples of scalar descriptor maps on frontal and profile 3D faces.

4.1.5 Local Volume (VOL)

Other kinds of measures that can be made from the local neighborhood are the ones that use volume. First, the barycenter (point $\mathbf{p}_c(v_i)$) of the perimeter of the neighborhood of vertex v_i is determined. Then the volumes of the tetrahedra computed from this point and all the faces of the neighborhood are summed. Figure 11 shows how the tetrahedra are computed. As the faces are oriented the volume can be positive (concave shape) or negative (convex shape).

4.1.6 Distance to Local Plane (DLP)

The distance to local plane is a coarse measure of the convexity/concavity at a point (Pears et al. 2010). It is defined as the Euclidean distance between vertex v_i and the plane fitting its neighboring points. This corresponds to projecting the vector between the centroid of the neighborhood and the vertex v_i onto the normal direction of the plane. In general, the neighborhood used to compute the normal and the neighborhood used to compute the target centroid can be different. However, it is usually simpler to take them as equal.

4.1.7 Histogram-based Local Shape Descriptors

Simple scalar descriptors are sometimes limited in terms of their ability to describe local surface shape. When dealing with complex surface shapes, more information is needed than a simple scalar value. It is possible to build local shape descriptors using histograms i.e. any array of fixed dimensions containing information about the neighboring surface at some mesh vertex. These are inherently feature vectors in themselves, but can be combined, in full or part, with the scalar descriptors described above to form larger and potentially more discriminating feature vectors.

Such histogram-based descriptors are often computationally expensive. For this reason they are usually only used as descriptors for matching a sparse set of keypoints, rather than used for detection of keypoints on a relatively dense set of raw mesh vertices. However, it is important that our framework is versatile and can use any kind of descriptor. Therefore, we have implemented two histogram-based descriptors and employed them within our keypoint detection framework, as follows.

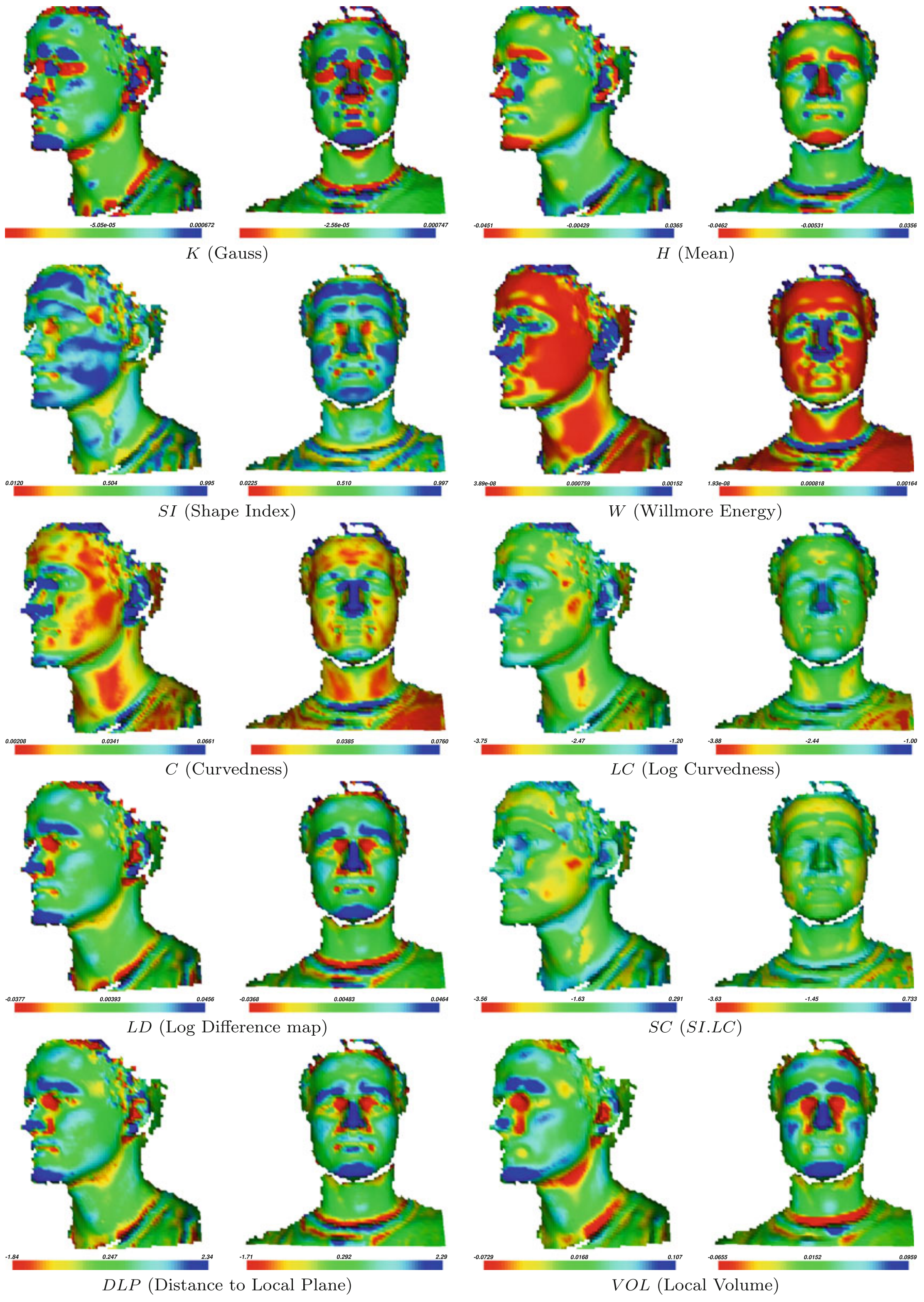


Fig. 10 Examples of scalar fields computed on two models of the same individual with different orientations ($R^{Eucl.}_{scalar} = 15$ mm). In an standard expert system approach, the researchers would look for extremal (*red* or *blue*) blobs on this maps that repeat under different pose and identities at known landmark position. Those human-extracted patterns would then be used to extract landmarks on new unseen query meshes

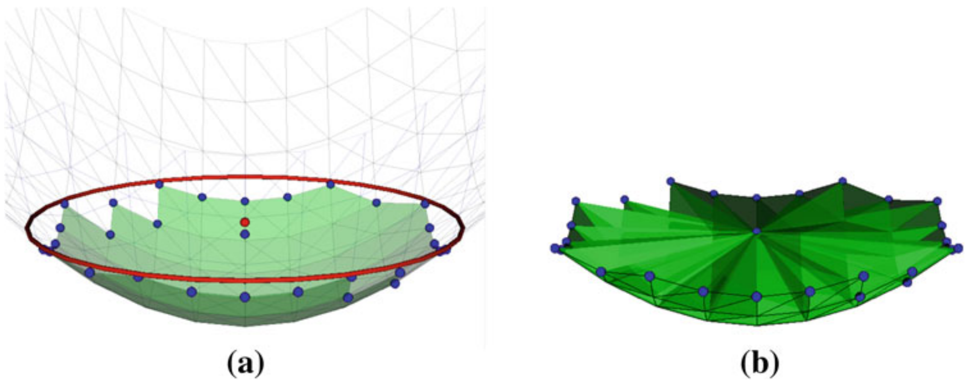


Fig. 11 Example of local volume (*VOL*) computed at the extreme vertex of a hyperboloid surface. (**a**) The neighborhood border points are used to compute the centroid point (*blue*) which is not far from the ideal center of the intersection curve (*red*). (**b**) The signed volumes of all tetrahedron are summed (Color figure online)

– *Spin images* The spin-image descriptor introduced in Johnson and Hebert (1999), encodes local shape relative to a mesh vertex and its normal. In particular, it is a histogram of radius and height values, where the radius is the orthogonal distance to the normal, and the height is a signed distance, relative to the vertex, in the direction of the normal. The name *spin image* is used because we can visualize a gridded half-plane being rotated around the vertex normal and neighboring vertices being accumulated in cells (bins) to form the shape histogram. In this sense, the cells of the histogram are analogous to the pixels of an image. The cells are not required to be square and the cell size can vary from cell to cell; for example, by following a log function. Here, only fixed sized cells are considered. The parameters for this descriptor are the number of radial cells, the number of vertical cells and the radial and vertical cell sizes.
– *Spherical images* The spherical image is a local shape descriptor that is simpler than the spin image and consists of a one dimensional vector of bins. Each cell represents the number of vertices present between two consecutive spheres centered on some mesh vertex, v_i.

4.2 Generating Descriptor-Landmark Scores

While correlations sometimes exist between descriptors' extremal values and the presence of landmarks (e.g. at the nose tip and near the inner eye corners), this can not be extended to less well-defined local shapes. By learning a distribution of the values of a descriptor for a particular shape of interest, one can easily determine, for any new point, a score for matching to a particular landmark.

If the value of one descriptor, at a particular vertex, is close to the maximum of the probability density function of a known shape, it has a good chance of corresponding to this shape, at least in the context of that descriptor.

4.2.1 Distributions of Local Shapes

For each landmark in the training set, the distribution of the values for one descriptor can be observed and approximated with a parameterized class-conditional probability density function where, in this context, the class is the landmark instance.

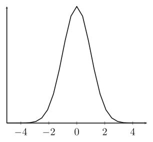
A lot of possible density functions and their mixtures can be used to approximate the descriptor distribution collected from the training set. Examples of commonly used functions are the Heaviside step function, the top-hat function, the Gaussian, inverse Gaussian, Von Mises, and so on.

In this paper only two are used: the inverse Gaussian, for the shape index descriptor, and the Gaussian distribution for all the others. Examples of these functions superimposed on the training data distributions of a descriptor value at a landmark position can be seen in Fig. 12.

The framework that we have implemented is generic regarding distribution modeling. The distribution of a particular descriptor is manually provided as a parameter to the system. The reason that we only used unimodal pdfs is that the observed distributions over the training data look unimodal. An enhancement to our system would be to select the appropriate distribution model from a pool of such models automatically.

Gaussian (2 parameters: mean, μ, and standard deviation, σ)

$$\mathrm{pdf}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Inverse Gaussian (4 parameters: μ, σ, x_0, $direction$, where μ and σ are mean and standard deviation, as before, and x_0 is the origin of descriptor values, x.)

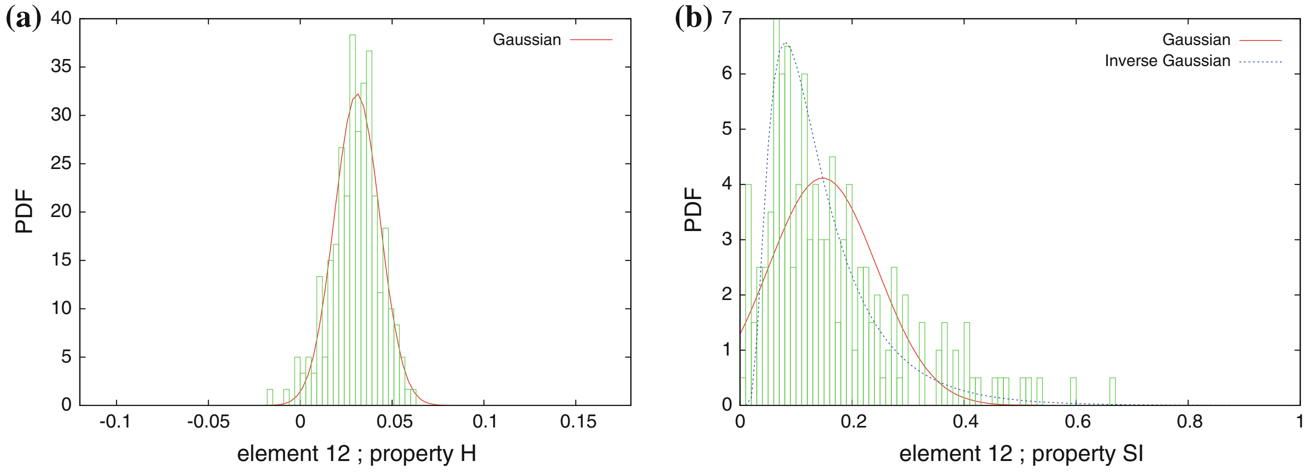
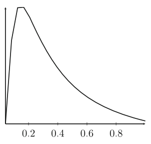


Fig. 12 Examples of parameterized probability density functions (computed from the mean and variance of training data) superimposed on the observed distribution for the lower-lip landmark. On the left, the mean curvature descriptor (H) is shown. On the right, the shape index descriptor ($SI_{0,1}$) is shown

$$\mathrm{pdf}(x) = \sqrt{\frac{\xi}{2\pi x'^3}} \exp\left(-\xi \frac{(x'-\mu)^2}{2\mu^2 x'}\right)$$



where $x' = \mathrm{sign}(direction).(x - x_0)$

The variance of this probability density function is given as $\sigma^2 = \frac{\mu^3}{\xi}$. If the observed mean and standard deviation of the training data is (μ, σ), we compute the inverse Gaussian parameter, ξ, as $\xi = \frac{\mu^3}{\sigma^2}$.

4.2.2 Converting Descriptor Maps to Descriptor-Landmark Score Maps

Computing descriptor maps is useful, especially if the positions of the targeted landmarks are the same as the positions of the local extrema over those maps. In general, this is not the case, and the problem becomes the mapping of the raw descriptor values via some function, where the function has extrema that are coincident with the targeted landmark positions.

For any vertex, v_i, and any descriptor-landmark pair, (d, λ), a score can be computed that can be thought of as the *relative likelihood* that the landmark λ is at vertex v_i given the scalar raw descriptor value $x_d(i)$ computed at that vertex. We use the term 'relative' because such scores are relative to the maximum likelihood of the given landmark for the given descriptor. These descriptor-landmark (DL) scores are in the range (0–1), where the maximum score of 1 is attained only if $x_d(i)$ is at the modal value of the modeled distribution for landmark λ.

Thus we define DL-scores as:

$$s_\lambda^d(i) = \frac{\mathrm{pdf}_\lambda^d(x_d(i))}{\max\limits_x(\mathrm{pdf}_\lambda^d(x))} \tag{11}$$

where pdf_λ^d models the distribution of descriptor d for the landmark λ, $x_d(i)$ is the raw value of descriptor d at vertex v_i, and the maximum probability density is taken over the descriptor value variable, x.

In the case of a Gaussian distribution of mean $\mu_{d,\lambda}$ and deviation $\sigma_{d,\lambda}$, $\max(\mathrm{pdf}_\lambda^d)$ is reached at $\mu_{d,\lambda}$ and we have:

$$s_\lambda^d(i) = \exp\left(-\frac{(x_d(i) - \mu_{d,\lambda})^2}{2\sigma_{d,\lambda}^2}\right) \tag{12}$$

Note that, in the case of the Gaussian distribution, which we use most often, a DL-score is a function of the Mahalanobis distance (number of standard deviations) from the mean of the modeled distribution. For example, Mahalanobis distances of (0, 1, 2, 3) gives scores of (1, 0.607, 0.135, 0.011).

In Fig. 13 examples of DL-score maps are presented.

4.2.3 Dealing with Shape Histograms

Compared to scalar descriptors, shape descriptors based on histograms are more difficult to deal with. Using the value within each cell (histogram bin) as a scalar descriptor within our framework is indeed computationally too expensive. Therefore, a single scalar descriptor is computed from each histogram.

To do this, the difference to the mean histogram of the target landmark is computed and projected onto a single scalar value along a direction that is defined by a two-class LDA problem. Here, one class is a set of neighboring vertices and the other class is a set of non-neighboring vertices. For each vertex, v_i, the scalar descriptor generated from a local shape histogram is computed as follows:

$$x_d(i) = \boldsymbol{\omega}_\lambda^T(\mathbf{x}'_d(i) - \bar{\mathbf{x}}'_{d,\lambda}), \tag{13}$$

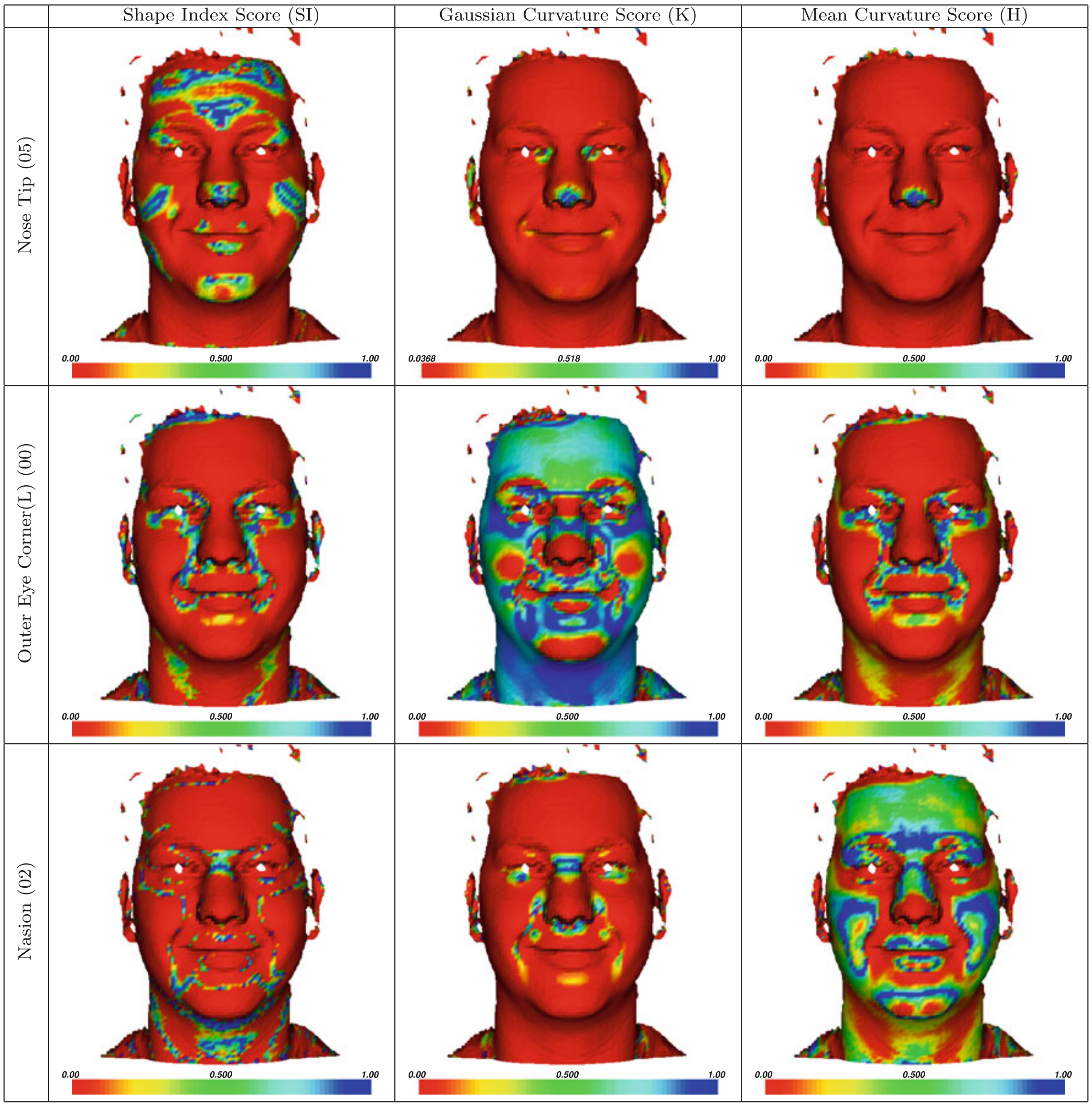


Fig. 13 Examples of descriptor-landmark (*DL*) score maps for three descriptors, whose scores are computed relative to modeled distributions at three landmarks

where $\mathbf{x}'_d(i)$ is an M-dimensional feature vector of the histogram's cell values at vertex v_i, $\bar{\boldsymbol{x}}'_{d,\lambda}$ is the mean feature vector learned at landmark λ and $\boldsymbol{\omega}_\lambda$ is the vector of weights constituting the direction in M-dimensional feature vector space that best separates the two classes in the above LDA problem. The scalar $x_d(i)$ is then treated in the same way as any other scalar descriptor and is mapped to a score in the range (0–1), as described in the previous section.

While the use of more complex histogram comparison techniques could be justified in terms of improved class separation (for example, the earth mover's distance Rubner et al. 2000), they are too expensive in terms of computation time for our application, as the number of operations is linear in the number of mesh vertices and the number of targeted landmarks.

4.3 Combining DL-Score Maps into a Single Landmark Score Map

Most of our DL-score maps are highly correlated. Indeed, most of them are based on descriptors derived from the principal curvatures (k_1 and k_2). It is important for our machine-learning approach to take this correlation into account when determining a function or set of rules to combine our D DL-score maps per landmark into a single score map, which we

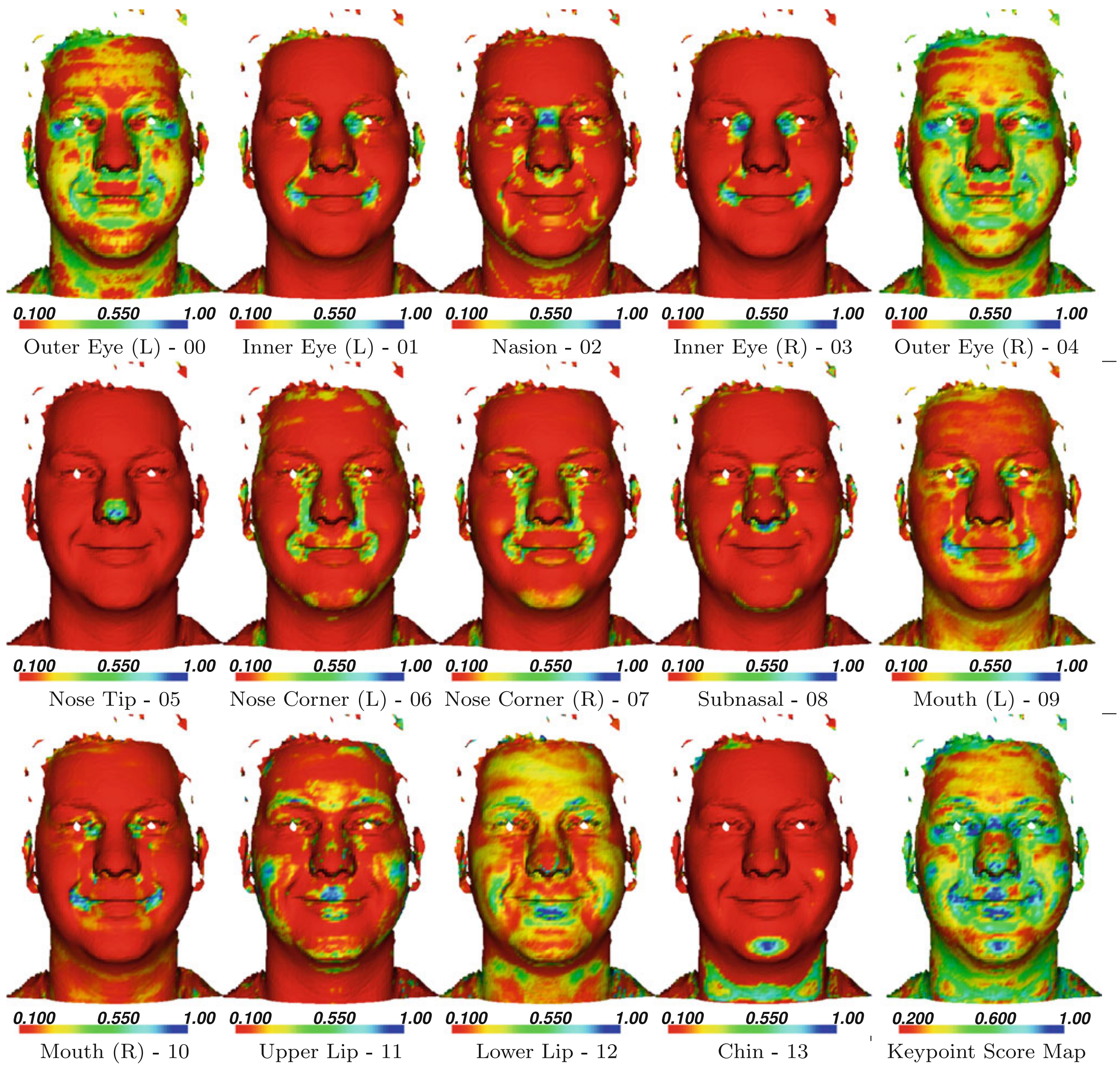


Fig. 14 Examples of normalized landmark score maps for the L shapes of interest. The final keypoint score map, computed for the same subject, is shown on the *bottom right cell*

call a *landmark score map* or L-score map. The following subsections detail two approaches, LDA and AdaBoost, to learn how to do this combination. Note that most applications of LDA and AdaBoost in the literature use their outputs directly to assign class labels. In contrast, we use these algorithms to generate a landmark score that is likely to be high for a vertex that is in the vicinity of something that looks like a landmark and low otherwise. This allows us to defer the classification to a discrete landmark label until later, when global, structural information is introduced.

4.3.1 Linear Discriminant Analysis (LDA)

A first simple idea is to use linear combinations of DL-scores. For a particular landmark, each of the D corresponding DL-score maps are multiplied by a learned weight and, added together, they form the L-score map.

Another way of viewing this is that the D DL-scores at some vertex, v_i, constitute a D-dimensional feature vector which is projected down to a scalar, by forming a dot product with a D-dimensional unit vector. Thus, if $\mathbf{s}_\lambda$ is a V-dimensional vector of such projected DL-scores that

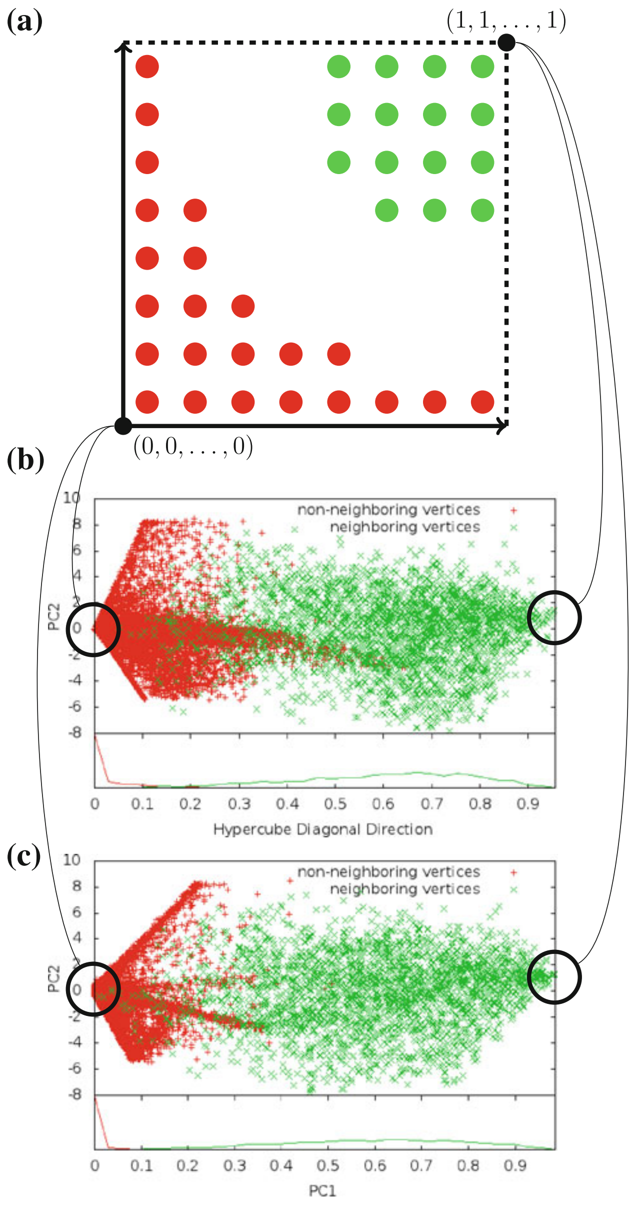


Fig. 15 (**a**) Schematic representation of the two class distributions inside the DL-score unit-hypercube for an ideal situation where the classes are separable by a single continuous class boundary. (**b**) and (**c**) Real capture of the two classes inside the DL-score hypercube for the nose tip landmark projected along (**b**) the hypercube diagonal direction, vector from $(0, 0, \dots, 0)$ to $(1, 1, \dots, 1)$ (for visualization), and (**c**) the extracted LDA direction

represents an L-score map, we have:

$$\mathbf{s}_\lambda = \mathsf{S}_\lambda \mathbf{u}_\lambda \quad (\lambda = 1 \dots L), \tag{14}$$

where S_λ is a $V \times D$ matrix of DL-scores (each column contains the values of a DL-score map) and $\mathbf{u}_\lambda$ is the unit vector of projection for landmark λ.

Note that L-score maps are normalized by shifting and scaling so that they span the standard range (0–1).

The weights (in unit vector $\mathbf{u}_\lambda$) used to combine DL-score maps are defined using LDA over a population of neighboring and non-neighboring vertices, relative to the relevant landmark. The population of neighboring vertices is defined as those at a distance <5 mm from the specified landmark on all facial meshes in the training set. The population of non-neighboring vertices is constituted of those between 15 and 45 mm from the same landmark (see Fig. 7 for the upper-lip landmark). These empirical radii have been selected to get two vertex populations of manageable size on faces.

LDA applied to these two classes (neighboring and non-neighboring vertices) returns the direction in D-dimensional DL-score space that best separates the two sets.

As each DL-score is between 0 and 1, the feature vector for a given vertex is a single point in a D-dimensional unit-hypercube. Non-neighboring vertices are expected to have many scalar DL-scores close to zero. Figure 15 shows an ideal representation of the two classes in the unit-hypercube as well as real projections observed in the training process. Examples of the resulting landmark score maps (L-score maps) are shown in Fig. 14.

It is interesting to compare DL-score maps and L-score maps. For example, the *Gaussian-curvature, nasion* DL-score map in Fig. 13 (bottom row, central map), reveals a ring of high (near 1.0) values around the ground truth nasion position. This is a consequence of the subject having a higher than average Gaussian curvature at his nasion and therefore lower DL-scores in the center of the nasion area. However, when multiple DL-score maps at the nasion are linearly combined into a single L-score map, we have a single blob of high values at the nasion, as seen in Fig. 14 (top row, central map).

We note in passing that if multiple landmarks have similar local structure, such as occurs with facial symmetry, we may wish to exclude repeated local shapes from the set of non-neighboring vertices. In effect, the training process would need to be made aware of such repeated local structures (e.g. the inner eye corners form a pair). Although such class-formation processes are simple to implement, we think that the improvements achieved will be small, and they are not implemented in the work presented here.

4.3.2 AdaBoost

One limitation of the LDA method is that it assumes that the two classes (neighboring and non-neighboring) can roughly be separated by a hyperplane. If the classes are well separated for some particular landmark, then this might be the case, but Fig. 15 suggests that some class boundaries might be better modeled by a hypersphere rather than a hyperplane. If this is the case, then a non-linear technique may allow us to extract and use even more discriminative information from the set of DL-score maps. We selected the AdaBoost technique[6] because the online computation is fast enough and because it has shown excellent performance over a wide range of applications in Computer Vision and Pattern Recognition over recent years, quite often giving state-of-the-art results, a prime example being the 2D face detection system of Viola and Jones (2004).

This method differs from the LDA approach in both the offline training process and online keypoint detection process, as follows.

- When training the system using the two classes (neighboring and non-neighboring vertices), a boosting technique is used to learn weak classifiers. This is described in Sect. 4.3.4.
- In the online part, these weak classifiers are used instead of the LDA weights, to obtain a landmark score (L-score) for each vertex. This is described in Sect. 4.3.5.

4.3.3 Boosting Technique

The boosting technique used here is simple. Each scalar DL-score is treated independently and each weak classifier is composed of the following:

- the index, d, of the descriptor, i.e. the dth element of the DL-score feature vector;
- a threshold T splitting the DL-score's 1D scalar space into two;
- a direction dir -1 ($t \leq T$) or +1 ($t > T$) stating which side of the space corresponds to the 'match' class, and
- a scalar, α, describing the weight associated with this single weak classifier.

Such weak classifiers are often termed *decision stumps* and they partition the DL-score vector space (a feature vector space) with a set of orthogonal hyperplanes.

4.3.4 Offline AdaBoost Training

For the training of the weak classifiers, a simple Adaptive boosting technique (or AdaBoost Freund and Schapire 1997) is used. At each additional iteration, the weights are assigned to the training data according to the classification error using the existing weak classifiers. Each dimension of the D-dimensional DL-score space is divided in small steps. The triplet $(d/T/dir)$ that best classifies the training set with the new weights is selected as the new classifier. To search the threshold along one DL-score dimension, a simple two-level coarse-to-fine approach is used. The range of observed values $[min, max]$ is divided into 200 steps. Once the best step k is found using this discretization, the range $[k-1, k+1]$ is divided again in 50 steps. The new best step k' that is found is used as the candidate threshold for this dimension.

The weight of a current best classifier is defined relative to the current error, e, in classification:

$$\alpha = \frac{1}{2} \log \frac{1-e}{\max(e, \epsilon)}. \tag{15}$$

where ϵ is a small constant. The influence of each input point is updated at each iteration by reducing the weights of the well classified points and retaining the weights of badly classified points, such that:

$$w_{ij} = \begin{cases} w_{ij}.\exp(-\alpha) & \text{if good classification} \\ w_{ij} & \text{otherwise.} \end{cases} \tag{16}$$

4.3.5 Online Landmark Score Generation using AdaBoost

For the online part of the process, each input vertex has a vector of DL-scores, each DL-score corresponding to a descriptor-landmark pair. The scalar *landmark score* (L-score) for each vertex is computed from the DL-score vector as shown in Algorithm 1.

Algorithm 1: Boosting L-score Computation.

```
Data: Vector of DL scores, $\mathbf{s}_\lambda(i)$; weak classifiers, $W$
Result: scalar $Lscore$
$Lscore = 0.0$
foreach weak classifier $(d, T, dir, \alpha)$ in $W$ do
    if $(dir \cdot s_\lambda^d(i) > dir \cdot T)$ then
        $Lscore += \alpha$
    else
        $Lscore += -\alpha$
return $\frac{1+Lscore}{2}$
```

Instead of using the sign of the result as a binary classification (−1 or +1), the output score is retained as a continuous value and remapped into the interval (0–1).

We emphasize that the optimal combination of descriptors for keypoint detection is not the same as the optimal combination of descriptors for locally discriminating between *different* landmark shapes: each of these learning processes requires a different set of vertices in the training data.

4.4 Keypoints from L-score Maps

In the L-score map, vertices with higher values (colored blue) are more likely to be good keypoints than the ones with lower values (colored red). However if a simple threshold was applied to the map, the system would detect areas (blobs) of vertices instead of a sparse set of points. Here, local

[6] Note, however, that our framework allows us to 'plug in' and use any classifier as an L-score generator. Switching to another technique is straightforward, if there is some advantage to this, in terms of the class of meshes that are being processed.

maxima are computed using a neighborhood of fixed radius (15 mm). Vertices that have a maximum value within this neighborhood are selected as keypoints. A threshold T_k can help eliminate local maxima that are too weak. The choice of T_k mainly influences the number of keypoints detected and therefore the computational cost of the matching process. In these experiments T_k is fixed to 0.85.

Given that we have a set of L landmark score maps, there are a number of different ways in which we can extract keypoints. The approach employed depends on how keypoints are to be processed to generate landmarks, in particular, how configural information is to be used. Here we describe two approaches that we have investigated and Fig. 16 compares keypoints using these two techniques.

4.4.1 *Keypoints from the Keypoint Score Map*

A first approach is to combine all of the $L = 14$ L-score maps into a single map, which we call an *keypoint score map*, by taking, at every vertex, the maximum normalized landmark score over all L L-score maps. In this technique, we completely ignore which L-score map the maximum came from. It may seem counter-intuitive to throw this landmark information away, but the L-score maps are the result of learning how to distinguish vertices from their neighbors for a set of targeted local shapes, not how to optimally distinguish one targeted local shape (landmark shape) from another.

An important advantage is that the resulting keypoint set is very sparse, because it does not allow two keypoints, generated by different landmark target shapes, to be too close together. Keypoints generated in this way are the ones used to evaluate our framework in Sect. 5, using the metrics of landmark retrieval rate and repeatability. An example of a keypoint score map can be seen in the last panel of Fig. 14 and and the keypoints generated from such map are shown in Fig. 16(top).

4.4.2 *Keypoints from Landmark Score Maps*

An alternative approach is to employ the information regarding which L-score map a particular keypoint came from. The simplest way to do this is to detect local maxima on the individual L-score maps for every landmark. Thus we can immediately generate a landmark candidate, as each keypoint is associated with the L-score map that it has been extracted from and thus can be associated with a single candidate label. It is interesting to ask whether such keypoint information can be used within a simple model-fitting approach in the assignment of final landmark labels. In Sect. 6, we address this problem and we find that we are able to get good landmarking performance.

Note that a problem with this approach is that the number of landmark candidates generated will increase linearly with L, the size of the dictionary of targeted landmarks. This is not usable for other configural matching approaches, such as graph and hypergraph matching techniques, where the number of edges and hyperedges become intractable to process. An example of keypoints generated directly from the L-score maps is given in Fig. 16 (bottom).

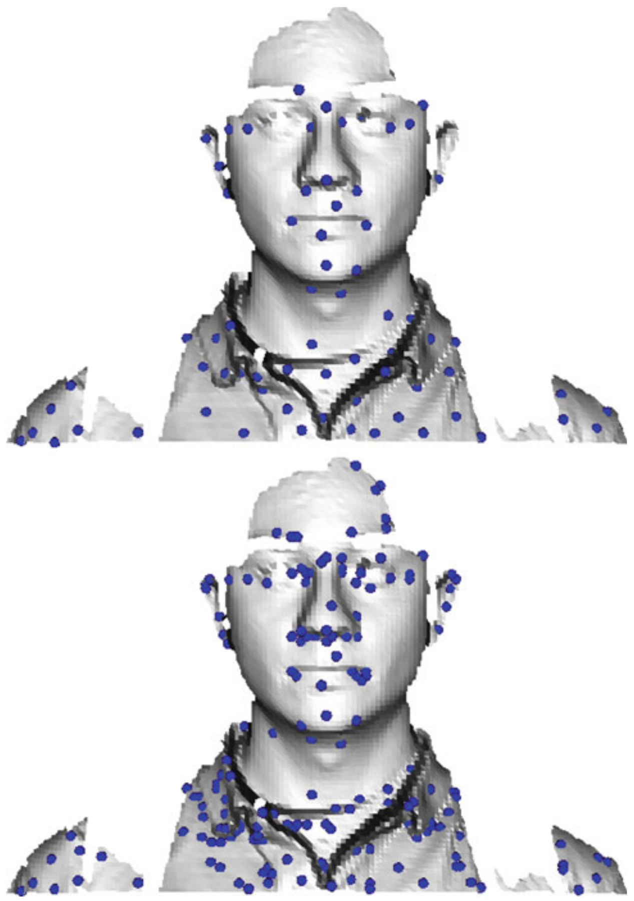


Fig. 16 Examples of extracted keypoints: (*top*) from the final keypoint score map (70 keypoints) and (*bottom*) directly from the landmark score maps (198 keypoints)

5 Keypoint Detection Results

In this section, we first describe two configurations that our system was tested with: a multi-scale configuration and a single-scale configuration. We then describe keypoint detection results associated with our LDA-based learning system and our AdaBoost-based learning system.

5.1 System Configurations

For all results presented in this paper, 10 descriptors were selected including two histogram descriptors:

- First principal curvature (k1)
- Second principal curvature (k2)
- Gaussian curvature (K)
- Mean curvature (H)
- Shape Index (SI)
- Log Curvedness (LC)
- Distance to Local Plane (DLP)
- Local Volume (VOL)
- Spin Image Histogram (SIH)
- Spherical Histogram (SH)

Using these descriptors, two different configurations are set up:

- Configuration 1: The histogram descriptors are defined with 4 different bin sizes (from 2.5 to 10 mm). The others are defined with 5 different neighborhood sizes (5, 15, 30, 45 and 60 mm).
- Configuration 2: All descriptors are computed at only one scale. The neighborhood size is set at 15 mm for all scalar descriptors, and the bin size at 5 mm for the histogram descriptors.

In total, configuration 2 uses 10 descriptor maps leading to 140 descriptor-landmark (DL) score maps which is far less than configuration 1 that requires 672 DL-score maps. The neighborhood and bin sizes have been set to values informed by previous research results (Creusot et al. 2011).

5.2 LDA Results

Figure 17 shows examples of final keypoint score maps computed for configuration 1 and 2 using LDA-based linear combination of DL-scores and the corresponding detected keypoints. A visual check of these results gives a lot of indications about the system and its drawbacks. The scan in the second column, for example, contains lots of false positive keypoints in the hair areas. However, in order to evaluate the results for the whole dataset, quantitative cost functions have to be used. We now present results for landmark retrieval rates and keypoint repeatability, using keypoints generated from keypoint score maps.

5.2.1 Landmark Retrieval

To evaluate the rate at which keypoints are localized near defined landmarks, the percentage of face meshes in which a keypoint is present in a sphere of radius R from the manually labeled landmark is computed. As there is no clear definition about what distance error should be considered for a match, this percentage is computed for an increasing acceptance radius ranging from $R = 2.5$ mm to $R = 25$ mm. Results for configuration 1 and 2 are given in Fig. 18. With configuration 2, at 10 mm, the nose tip is present in the detected keypoints 99.47 % of the time, and the left and right inner eye corners in 90.50 and 92.56 % of the cases. The figure does indicate that some landmarks (e.g. nose tip) are much stronger than others (e.g. mouth corners) in terms of their retrieval rate. That is not to say weak landmarks should be avoided altogether, because in some query meshes, only weak landmarks will be present.

In other words, our method will not succeed in detecting all potential landmarks in all facial meshes. However, it aims to provide an initialization for further face processing that doesn't rely on a small, specific set (e.g. triplet) of target landmarks. In Fig. 19, it can be seen that the mean number of correctly selected landmark candidates is around 12 (of a possible 14) for a radius of 10 mm. This illustrates a significant benefit our approach: by detecting more landmarks with no sequential dependencies between those landmarks, we decrease the single-point-of-failure risk that many candidate landmark selection systems often have.

5.2.2 Repeatability

The intra-class (same subject identity) repeatability is measured on the FRGC v2 dataset for which registration of the faces to a common pose has been computed using the Iterative Closest Point method (ICP Besl and McKay 1992) on the cropped meshes. The transformation matrices describing these registrations are available on the first author's webpage.[7] For each pair of faces of the same subject, the two sets of keypoints are cropped and registered to a common frame of reference. The proportion of points in the smallest set that have a counterpart in the second set at a distance R is computed. The repeatability using configuration 1 and 2 is given in Fig. 20 and compared with the repeatability of the hand-placed landmarks. It can be seen that at 10 mm the proportion of repeatable points is around 85 % (configuration 1) and 75 % (configuration 2) on average, whereas for hand-placed landmarks (the best performance that we could expect) is around 96 %. As expected, our system works better with more descriptors, because it can effectively ignore correlations. However, as more and more descriptors are added, we get to a situation where there are diminishing returns for the additional computation involved. Therefore, configuration 2, which has fewer descriptors, is used in the following section, to make our AdaBoost training computable in a reasonable amount of time and the overall system faster.

5.3 AdaBoost Results

In this section, we compare the LDA-based DL-score combination approach to the non-linear DL-score com-

[7] http://www.cs.york.ac.uk/~creusot.

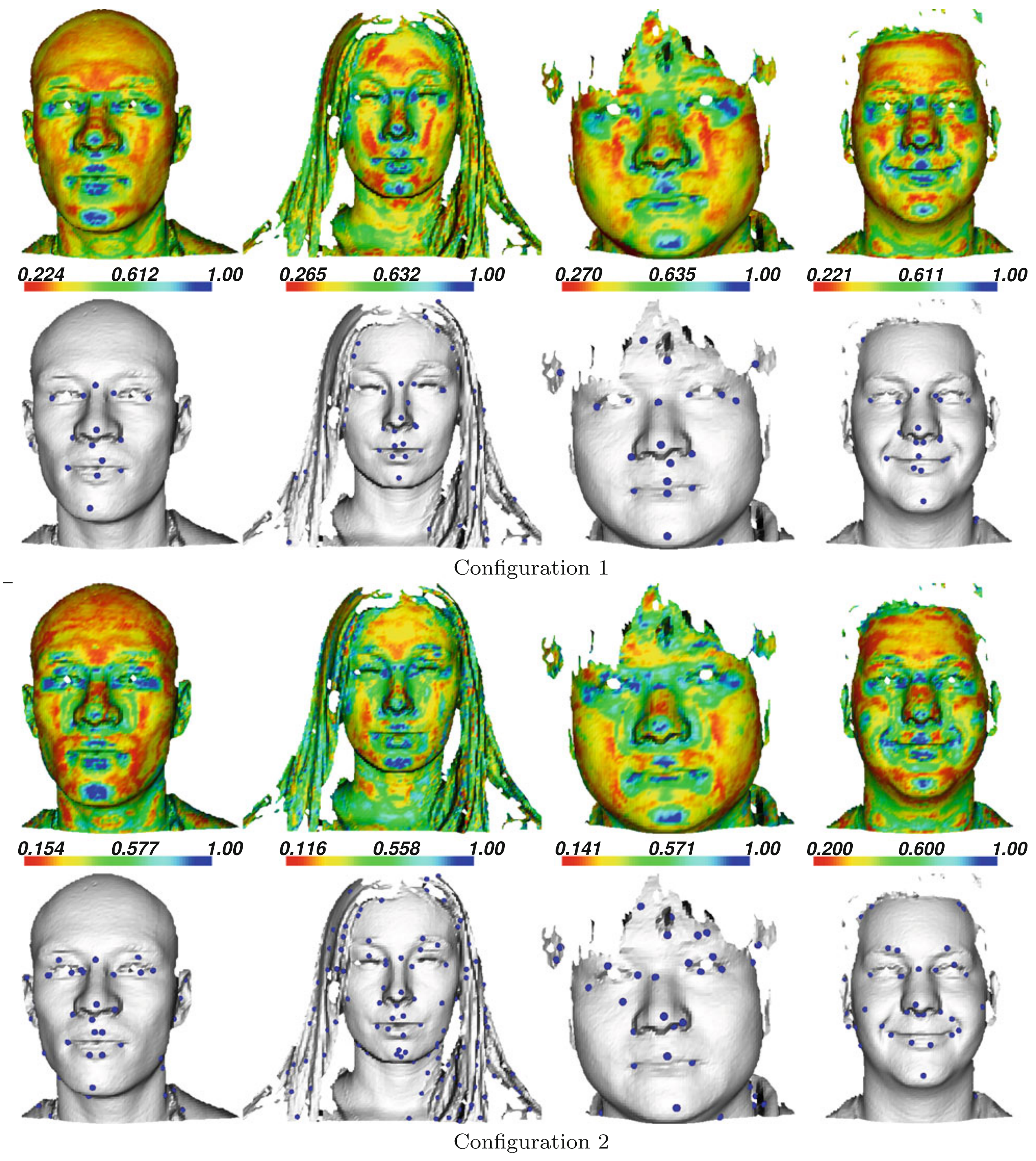


Fig. 17 Examples of extracted keypoints on faces from the FRGC v2 dataset using our multi-scale system (configuration 1, *upper two rows*) and our single-scale system (configuration 2, *lower two rows*). For each configuration, the *first row* shows the final keypoint score map where vertices *colored blue* represent the highest scores while, in the *second row*, the detected keypoints are shown

bination technique using AdaBoost. All results here are using the DL-score maps computed with configuration 2 (i.e. single local neighborhood scale and histogram bin size).

5.3.1 Number of Classifiers

A study of the variation of the number of weak classifiers on the training set shows that a plateau is reached relatively

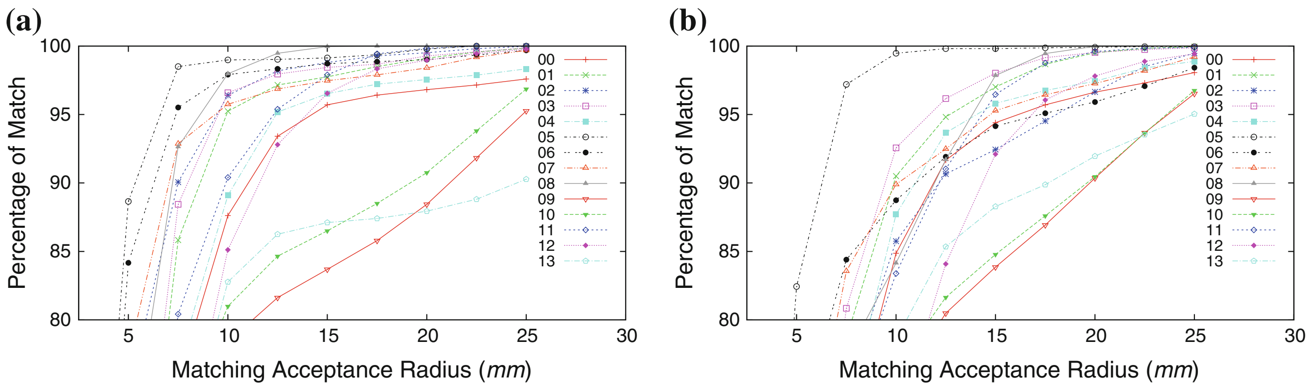


Fig. 18 Matching percentage per landmark (0–13) with an increasing matching acceptance radius on the FRGC v2 test set. (**a**) using all descriptors (Configuration 1), (**b**) using a subset of descriptors (Configuration 2)

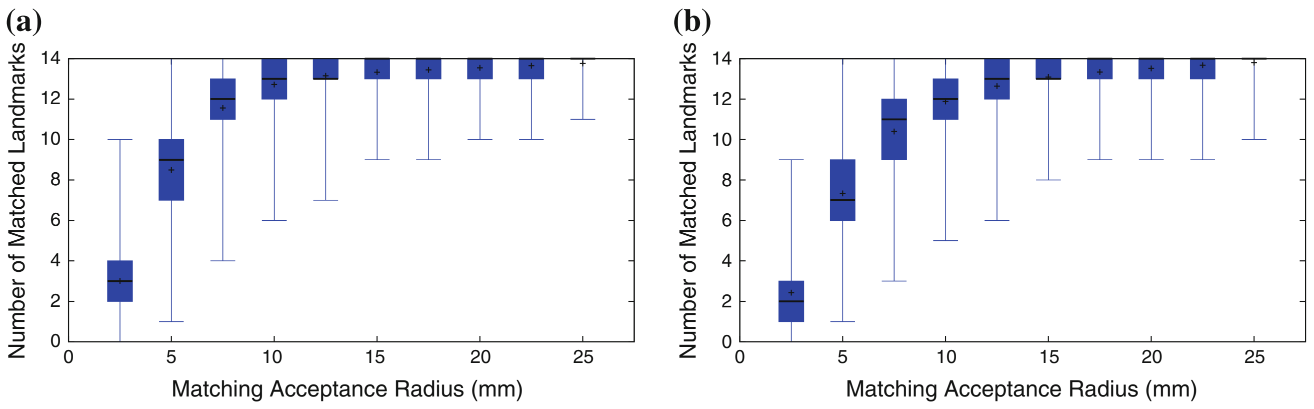


Fig. 19 Number of matching landmarks per file on the test subset of the FRGC v2 database. (**a**) LDA-based learning using all descriptors (Configuration 1). (**b**) LDA-based learning using a single-scale subset of descriptors (Configuration 2)

quickly (see Fig. 21). A cross validation on the training set showed that an upper limit on the number of classifiers is not really important: the system doesn't seem to over-fit the training data even with 160 classifiers (compared to the 10 descriptors used). This can be explained by the fact that the classifiers can be very similar to each other within the AdaBoost technique. In our experiment we usually used 10 or 20 classifiers for each landmark shape of interest.

5.3.2 *Computing the Separation Between L-Score Distributions*

Both the LDA and AdaBoost methods produce L-scores closer to zero for non-neighboring vertices and closer to one for neighboring vertices. An ideal output when plotting the distributions of those L-scores would be to have a sharp peak at zero for non-neighboring and a single spike at one for neighboring vertices. Of course, this is not the case and often the two distributions will overlap. In order to compare quantitatively the two classification techniques, a cost function to measure how well it separates the two classes is needed. Given two distributions densities D_0 and D_1 (respectively non-neighboring and neighboring) over a domain (0–1) how can the separation between the two distributions be quantified? There are many possible solutions to this simple problem. In our particular case, the distribution is not necessarily smooth or continuous, therefore looking at the problem at one threshold is not meaningful.

For every threshold t set between (0–1), the following can be computed:

$$\begin{array}{ll} \text{True Negative Rate:} & \mathrm{TNR}(t) = \int_0^t D_0(x)\,dx \\ \text{False Positive Rate:} & \mathrm{FPR}(t) = \int_t^1 D_0(x)\,dx \\ \text{False Negative Rate:} & \mathrm{FNR}(t) = \int_0^t D_1(x)\,dx \\ \text{True Positive Rate:} & \mathrm{TPR}(t) = \int_t^1 D_1(x)\,dx \end{array} \tag{17}$$

Obviously our aim is to be able to determine which methods are able to give low FNR and FPR values. By integrating over all possible t, a global notion of the intersection between the two distributions is defined:

$$\begin{aligned} I(D_0, D_1) &= \int_0^1 \mathrm{FNR}(t)\cdot\mathrm{FPR}(t)\,dt \\ &= \int_0^1 \left(\int_0^t D_1(x)\,dx\right).\left(\int_t^1 D_0(x)\,dx\right)dt \\ &= \int_0^1 \int_0^t D_1(x).(1 - D_0(x))\,dx\,dt. \end{aligned} \tag{18}$$

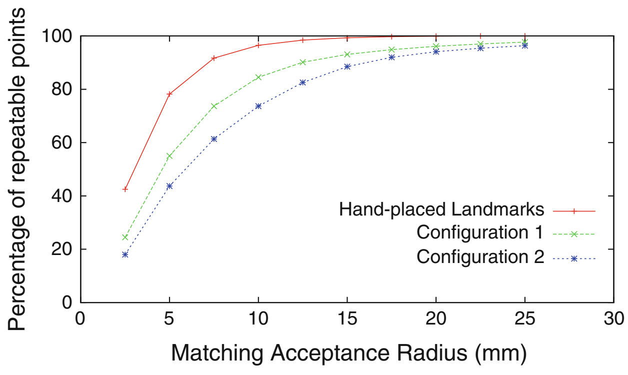


Fig. 20 Percentage of points repeatable after registration at an increasing matching acceptance radius. The measure of the human hand-placed landmarks is used as a reference

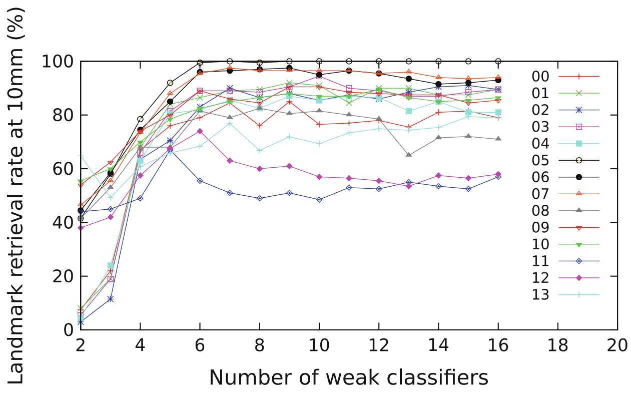


Fig. 21 Variation of the retrieval rate with a 10 mm error acceptance radius for different numbers of weak classifiers on the training set

This metric can be used to compare LDA and AdaBoost approaches and it is very explicitly related to how well some landmark can be separated from its surrounding area.

It is possible to use a simpler measure of distribution intersection in place of this metric, for example the Bhattacharyya distance (Choi and Lee 2003), which is the integral of the square root of the product of the two distributions and lies in the range (0–1).

5.3.3 Comparisons of AdaBoost with LDA

A first way of comparing the our LDA-based system with our AdaBoost-based system is to see how well each of them separate the two classes used in training. Figure 22 shows the different scoring results of both LDA and AdaBoost methods when trying to differentiate neighboring and non-neighboring vertices. It can be seen that the neighboring class is much more scattered with the LDA scoring than with AdaBoost. In Fig. 23 the intersections of the density distributions are compared for each of the L landmark shapes of interest. In all cases, AdaBoost performs better than the LDA with this intersection metric.

The two methods can also be compared when looking at the performance of keypoint detection. In terms of repeatability and number of retrieved landmark positions both methods give similar results (see Fig. 24a, b). However, when looking at single landmark retrieval rates (see Fig. 25) the AdaBoost method seems to perform better for the same set of descriptors on some landmarks and less well on others. In Fig. 26 we see that AdaBoost seems to work better for the nasion, nose and mouth corners, while LDA works a lot better for the subnasal and upper lip landmarks.

Of course, AdaBoost is more likely to gain in accuracy with a bigger training set than the LDA methods. But these results indicate that improvement of the system is still more likely to come from the use of more and better descriptors than from the use of a more sophisticated DL-score combination mechanism. Indeed, when comparing Figs. 25 and 18, we observe that the gain in performance linked to the DL-score combination method is minor compare to the gain of using more descriptors at multiple scales.

If more descriptors are used, the price in terms of computation can grow rapidly. We think that a good way to deal with this would be to look at dynamic scoring of the vertices using decision trees, as used in Shotton et al. (2008). At each node in the tree, only the most discriminative descriptor for this sub-tree would be computed.

6 Application to Landmarking

In this section, a landmarking system based on our keypoint detection framework is presented. Results are compared with state-of-the-art 3D face landmarking systems on the FRGC v2 dataset. We also show results on a more challenging dataset, the Bosphorus dataset (Savran et al. 2008).

6.1 Workflow

The landmarking framework is based on the keypoint detection system. The first steps are exactly the same (see Fig. 27). However, this time, the L-score maps are not combined into a final keypoint score map. Rather, the keypoints are detected on each L-score map separately. This leads to more keypoints, but with the advantage that only one label is associated with each landmark candidate (see Sect. 4.4). The final labels are selected by fitting a *scale adapted rigid model* of the targeted landmarks to the query using the RANdom SAmple Consensus (RANSAC) approach. The output landmarks are defined as the projection of the registered landmark model's points onto the face. The model fitting is not only adding the label but also is adjusting the position of the points. This has the advantage of providing landmarks even in face regions that have missing or spurious data.

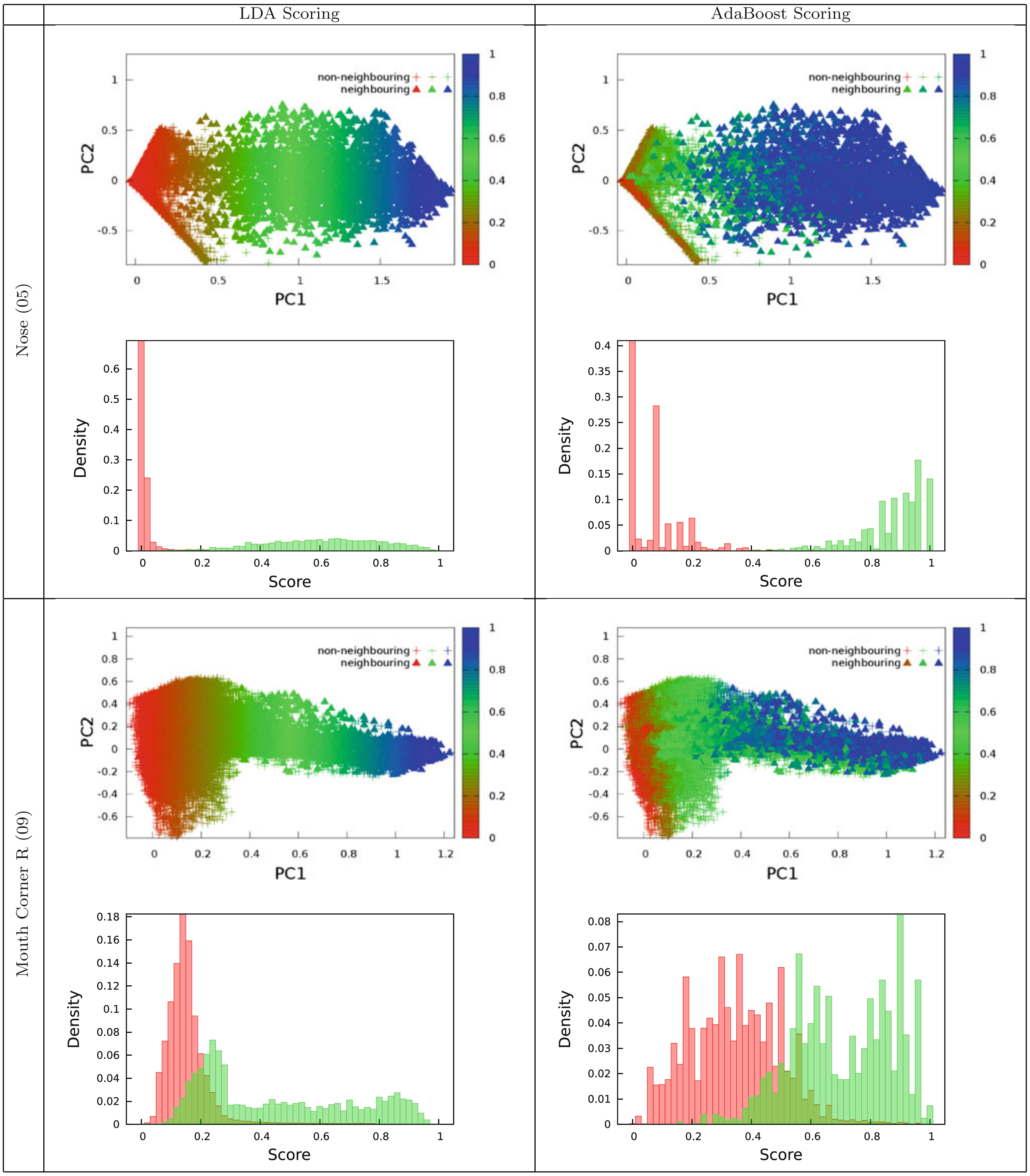


Fig. 22 Examples of the differences in L-scores using the LDA method (*left column*) and the AdaBoost method (20 weak classifiers, *right column*) for two landmarks: nose (*upper four* figures) and right mouth corner (*lower four* figures). The *upper row* of figures for each landmark show the L-score features of the two classes in a basis composed of the LDA-extracted direction and an arbitrary orthogonal direction with the color mapping showing the L-scores normalized to (0–1). The lower figures for each landmark represent the density of the neighboring and non-neighboring classes through the (0–1) scoring spectrum. The distributions show that the nose-tip is much easier to separate from its neighbors than the mouth corner, as expected

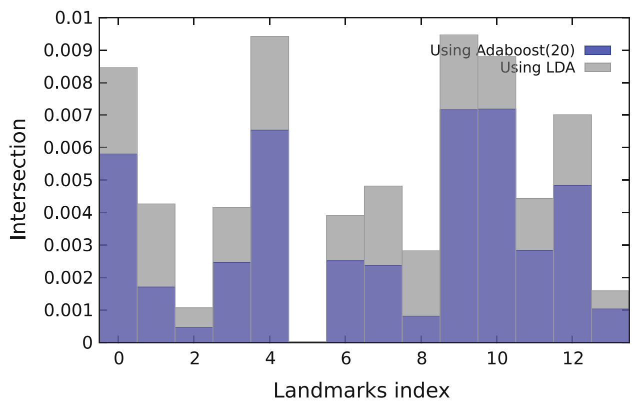


Fig. 23 Comparison by landmark of the intersection between neighboring and non-neighboring L-score distributions. AdaBoost is better at separating the two classes than LDA (the intersection is closer to zero)

The threshold for keypoint detection on the landmark score maps is set to 0.75 to reduce the number of false negative landmark candidates and therefore speed up the matching process.

6.2 Model-fitting

6.2.1 Scale-adapted Rigid Registration

In order to establish a one-to-one correspondence between the query landmark candidates and the model landmarks, we need to use configural information related to the global geometry of the face. Here the simplest possible approach is chosen: to register a scale-adapted rigid model of the face onto the query keypoints. A registration between two sets of labeled points is a geometric rigid transformation that can be represented as a 4×4 transformation matrix $\mathbf{T}$, with 7 degrees of freedom (3 for rotation, 3 for translation and one for scale). In our case, the query keypoints only have one candidate label each. Therefore finding the best registration is equivalent to finding the best subset of keypoints in the query, i.e. those that best support a specific transformation. To do so a RANdom SAmple Consensus (RANSAC) approach is used. RANSAC is a model fitting meta-algorithm introduced in Fischler and Bolles (1981) that consists of constructing a set of points

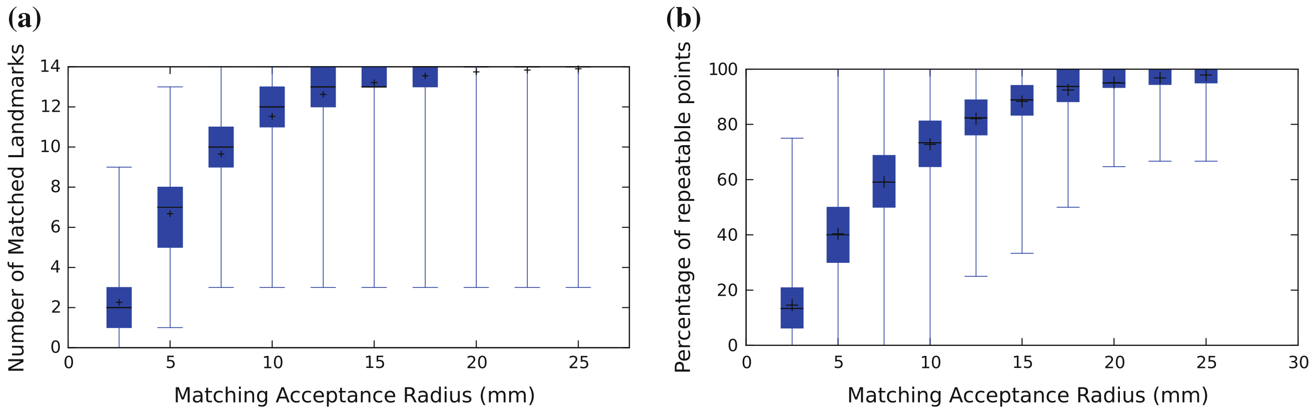


Fig. 24 AdaBoost-based learning system in configuration 2 (single-scale): **a** number of retrieved landmarks, **b** repeatability for an increasing matching acceptance radius on the FRGC v2 test subset

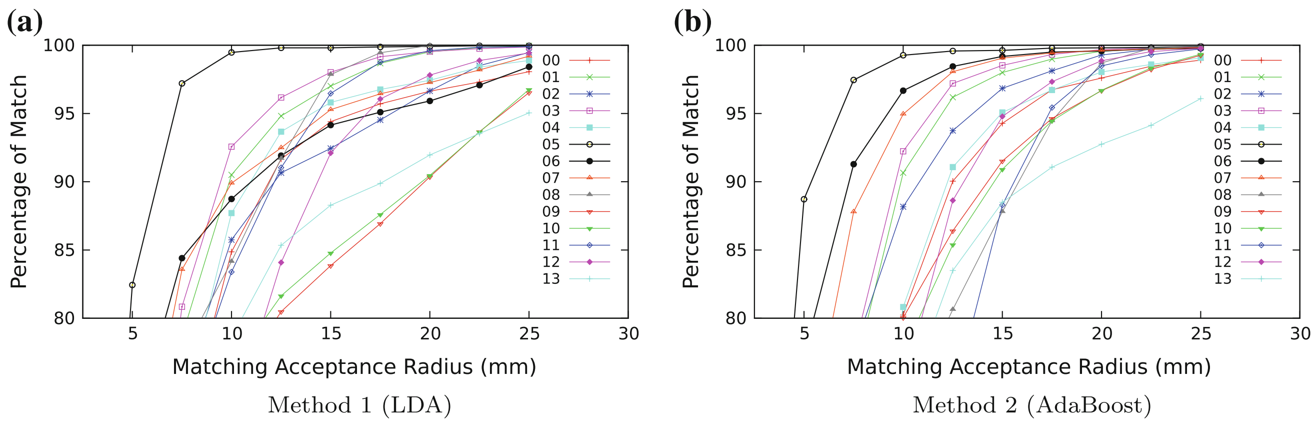


Fig. 25 Matching percentage per landmark (0–13) with an increasing matching acceptance radius on the FRGC v2 test set. AdaBoost gives better results than LDA for the same configuration (configuration 2)

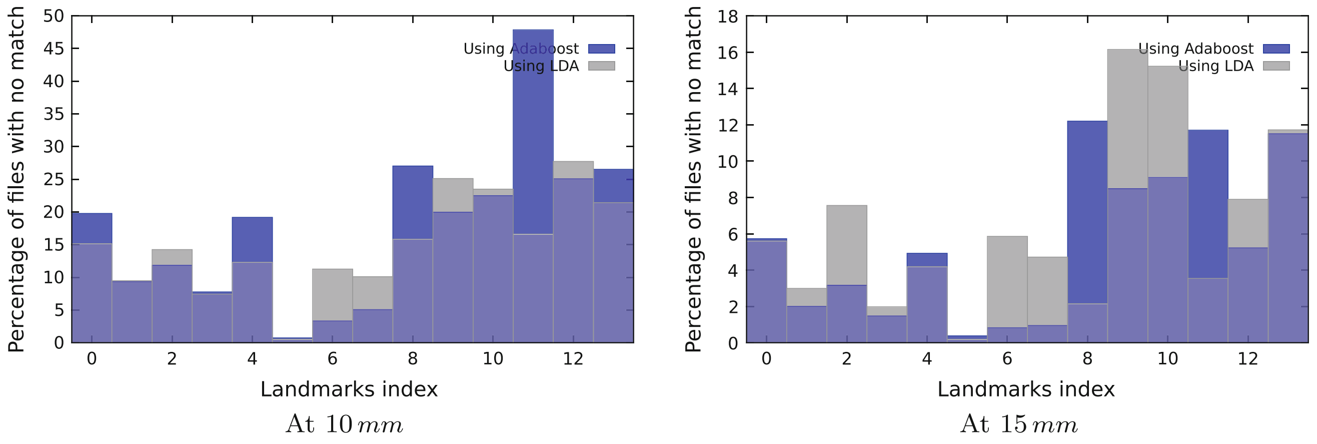


Fig. 26 Retrieval error rate for the LDA and AdaBoost DL-score combination methods for the 14 landmarks

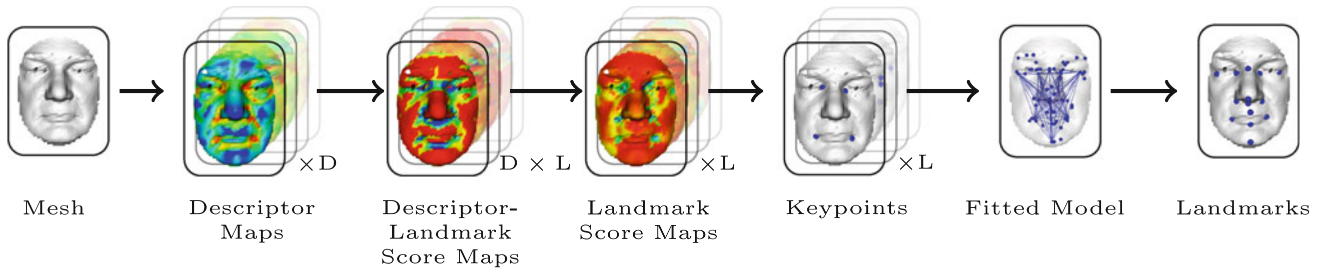


Fig. 27 Workflow of the landmarking system.

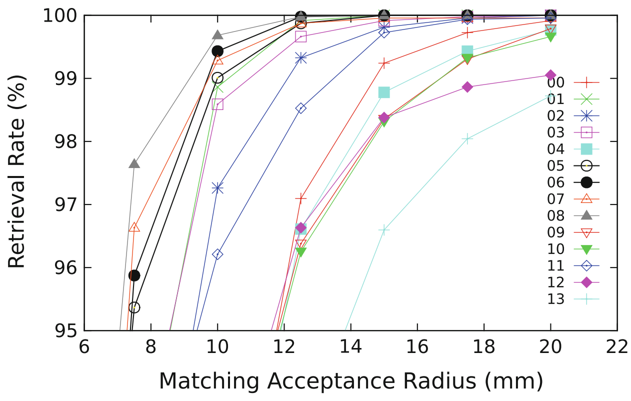


Fig. 28 Landmark retrieval rate for the 14 landmarks

that agree on fitting the same parameterization of a model. To find the parameterization with the largest number of supporting keypoints, N_s sets of keypoint triplets are randomly sampled to instantiate the model. In our case the model is a scale-adapted rigid model of the target landmarks and its parameterization is the transformation matrix T. The 'consensus' function of the algorithm decides whether a new point agrees with the current solution by enforcing two conditions:

– the distance between the query point and its corresponding transformed model landmark should be under a given threshold
– the dot product between the normals of the query and model points should be above a given threshold.

If both conditions are true, the correspondence is added to the consensus and a new transformation T is computed by using a linear least-squares method.

6.2.2 Dealing with Symmetry

A known problem with our approach is that the local shapes in our dictionary can be correlated, especially when associated with symmetric regions. For example, a keypoint detected near the outer right corner of the eye has big chance of being a landmark candidate for its left counterpart. This can lead to 'upside-down' face detection, or simply to imprecision in the global registration as some points might be discarded in the fitting process. The RANSAC solution depends on a list of pairs $(point, label)$ given as inputs. As the quality of the labels is uncertain at this point, running the RANSAC algorithm with different inputs can help cover a bigger search area. Selecting the best RANSAC solution among several

Table 1 3D face landmarking systems that are tested on more than 4000 models from the FRGC v2 dataset

Authors	[Chang et al., 2006]	[Mian et al., 2006]	[Segundo et al., 2007]	[Romero and Pears, 2009]	[Alyuz et al., 2010]			[Segundo et al., 2010]		This Work, 2011			
#Landmarks	3	1	6	3	5			5		14			
Acceptance Radius	$<?$	$<?$	$<?$	< 12	< 10	< 12	< 20	< 10	< 15	< 10	< 12	< 15	< 20
Nose (05)	99.40	98.3	99.95	99.77	99.62	99.80	99.87	**99.95**	99.95	99.01	**99.81**	**100.0**	**100.0**
Eye Inner Corners (01,03)	–	–	99.83	96.82	96.59	98.54	99.54	**99.02**	99.64	98.73	**99.71**	**99.96**	**100.0**
Nose Corners (06,07)	–	–	99.76	–	98.60	99.29	99.87	99.35	99.95	**99.36**	**99.87**	**99.98**	**99.98**
Subnasale (08)	–	–	99.98	–	–	–	–	–	–	99.68	99.98	100.0	100.0
Mouth Corners (09,10)	–	–	–	–	–	–	–	–	–	91.33	95.63	98.34	99.73
Eye Outer Corners (00,04)	–	–	–	–	–	–	–	–	–	89.84	95.92	99.01	99.84
Nasion (02)	–	–	–	–	–	–	–	–	–	97.26	99.07	99.81	100.0
Upper Lip (11)	–	–	–	–	–	–	–	–	–	96.21	98.21	99.73	99.96
Lower Lip (12)	–	–	–	–	–	–	–	–	–	92.04	96.00	98.38	99.05
Chin (13)	–	–	–	–	–	–	–	–	–	84.94	91.96	96.60	98.72
Candidate Selection	ES	ES	ES	ES	ES			ES		ML			
Independence	no	n/a	no	yes	no			no		yes			
Test Size	4,485	4,950	4,007	4,013	4,007			4,007		4,750			
Train Size	–	–	–	–	–			–		200			
Pre-processing	S,C[1]	∅	H,C	S,H	S,H,C			S,H,C		∅			
Pre-processing Time	–	–	1.1s	–	–			1.0s		0s			
Processing Time	–	–	0.4s	–	–			0.3s		1.18s			

ES: Expert System, ML: Machine Learning, C: Cropped/Segmented, H: Hole Filling, S: Spike Removal
[1] In [Chang et al., 2006] the mesh were cropped using 2D texture (skin colour).

Results using the same metric are colored the same. When a comparison is possible, results in bold font highlight the best system score for the given metric

can easily be done by looking at mean projection distance to the mesh.

Running RANSAC several times with different seedings, can only improve the chances of finding the best match. However, designing meaningful seedings is not always easy. In this experiment, RANSAC is run twice with two different starting sets of correspondences. One where the keypoints are associated with the one label derived from the score maps, and a second where the keypoints are associated to one or two labels depending on whether the initial label belongs to a symmetric pair. One of the two transformations is selected as the best if its corresponding projection distance to the surface mesh is minimal.

On the 4,750 scans from the test set on the FRGC, 2,564 obtain better fitting with the first seeding, while 2,186 prefer the second. In most cases, the difference between the mean projection distances is small. In 99.31 % of the cases, the difference between the two solutions is under 2 mm.

6.2.3 Projection of Landmarks onto the Query Mesh

Once the transformation, T, has been defined, all model landmarks are associated to the closest vertices on the query mesh. Those positions and associated labels are defined as detected landmarks and are the outputs of our system.

6.3 Landmarking Results

The following tests have been executed using configuration 2 (single-scale descriptors) of our keypoint detection system. The landmarks obtained using our technique can be downloaded on the first author's webpage[8] to help future results comparison. Figure 28 shows the landmark retrieval rate for an increasing acceptance radius.

[8] http://www.cs.york.ac.uk/~creusot.

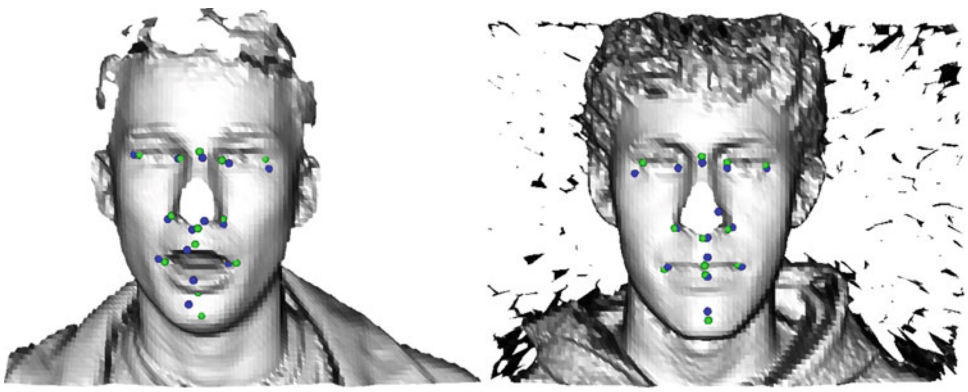


Fig. 29 Examples of landmarking in cases with missing nose where expert systems usually fail (model 04814d22 and 04505d222 of the FRGC). Our system doesn't need the nose tip to be correctly detected in order to find the other landmarks (landmark independence). *Blue* points represents our results. *Green* points represent the ground truth (Color figure online)

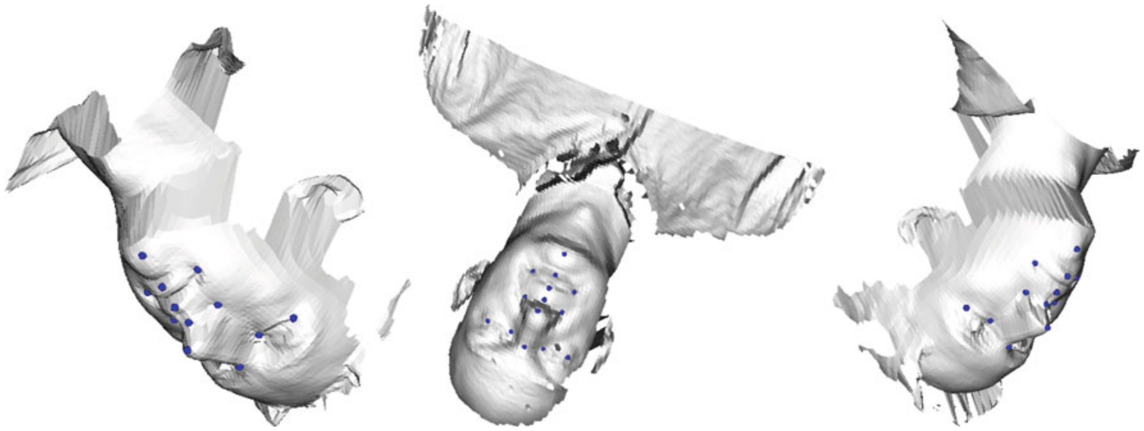


Fig. 30 Examples of localizations on rotated meshes. Our system only uses relative vertex positions and normals and is therefore translation and rotation invariant (pose invariant)

Landmarking results are often difficult to compare due to the variety of datasets, preprocessing and performance metrics. Obviously, the bigger the dataset the more meaningful the results are. Here our method is compared with previous studies that give results on at least 4000 models from the FRGC v2 dataset. In Table 1, comparison to state-of-the-art methods is presented at the most commonly used acceptance radii for human face landmarking (10, 12, 15 and 20 mm).

Our system does not outperform all recent expert system techniques in terms of precision (at 10 mm), but does so in terms of robustness, while also presenting some unique capabilities. Firstly, the number of discrete failures is zero. The system always succeeds to coarsely register the face and find some correct landmarks. Indeed 100 % of the models have at least three points retrieved at 20 mm. Secondly, our system is highly generic, allowing us to detect L (14) local shapes corresponding to targeted landmarks using exactly the same method for each of them. Thirdly, due to the landmark independence of the system and the non-sequentiality of the search, our system has no trouble dealing with face scans with missing facial features, for example when the nose-tip is missing (see Fig. 29). Most existing techniques will fail in such cases, as this landmark presence is required in the query scan. This confers to our system a great advantage when dealing with occlusions, as observed in real life scenarios (for example, those that include pose variation, occlusions by hands, cell phone, glasses, and so on). In addition, given that we avoided any pose-dependent assumptions, our system is invariant to rotation of the 3D surface, as seen in Fig. 30. Figure 31 shows distance errors relative to the manual mark up for the FRGC v2 test scans, while examples of detected landmarks on this dataset are shown in Fig. 32.

The landmarking experiment has also be run on the Bosphorus dataset (Savran et al. 2008). This dataset hasn't been used very often in the literature and is therefore less convenient to compare results with existing techniques. However it is far more challenging in terms of non-standard pose capture with scans presenting large pose variation as well as occlusions. Results are given on this dataset for two main reasons:

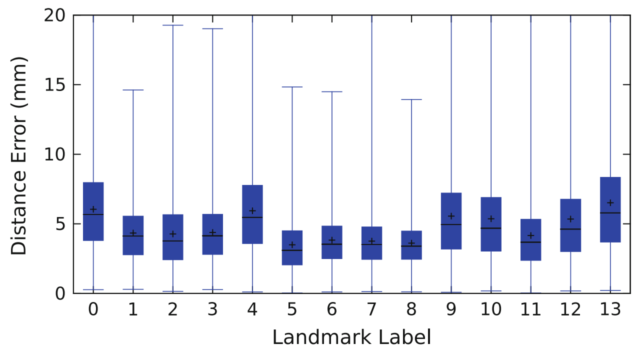


Fig. 31 Distance error for the 14 landmarks. The candle stick represents the min/Q1/median/Q3/max values. The plus sign represents the mean

– to highlight some limitations of our technique that could not have been detected with the FRGC dataset alone;
– to provide enough data to allow results comparison with this dataset in future research.

Table 2 contains the landmark retrieval rates for the different parts of the dataset, as well as for the whole set. Scans of yaw rotation marked as 90° were not used due to their poor associated descriptor maps (higher resolution might be necessary to treat those cases). Moreover, scans marked as IGN (ignored) in the dataset have also been discarded. In total we tested 4,339 face scans from the Bosphorus dataset. Figure 33 shows examples of landmark localization on some of these scans.

6.3.1 Limitations

A limitation of our system is that it relies heavily on local descriptors. Therefore, the robustness of the system strongly depends on the robustness of the local descriptors used. If those are not robust to occlusion the local DL-scores might be spurious and detection quality will suffer. For example, in a profile view, when a vertex is near the border of the mesh, its local neighborhood is incomplete and the descriptors computed on this neighborhood are likely to be noisy. It might be interesting to develop different specialized descriptors that can detect keypoints near the border of the mesh using, for example, extracted 2D curves along

Table 2 Results on the more challenging Bosphorus dataset using our landmarking method

Test Sets (size)	Acceptance Radius	Eye Outer Corners (L) (00)	Eye Inner Corners (L) (01)	Nasion (02)	Eye Inner Corners (R) (03)	Eye Outer Corners (R) (04)	Nose (05)	Nose Corners (L) (06)	Nose Corners (R) (07)	Subnasale (08)	Mouth Corners (L) (09)	Mouth Corners (R) (10)	Upper Lip (11)	Lower Lip (12)	Chin (13)
Neutral (200)	< 10	93.00%	100.0%	96.50%	99.50%	90.50%	98.50%	99.00%	99.50%	100.0%	97.00%	96.50%	99.00%	96.00%	58.50%
	< 12	98.00%	100.0%	99.00%	100.0%	94.00%	99.50%	99.50%	99.50%	100.0%	99.00%	98.50%	100.0%	98.50%	69.00%
	< 15	99.50%	100.0%	99.50%	100.0%	99.50%	100.0%	100.0%	100.0%	100.0%	100.0%	99.50%	100.0%	100.0%	84.00%
	< 20	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	95.50%
Emotions (453)	< 10	85.43%	99.34%	93.60%	98.68%	85.21%	94.26%	97.57%	96.25%	99.12%	65.34%	67.33%	90.51%	71.08%	26.39%
	< 12	92.27%	99.56%	97.35%	99.34%	93.38%	98.68%	99.12%	99.78%	99.56%	75.50%	76.60%	95.36%	79.47%	34.59%
	< 15	97.79%	99.78%	99.78%	99.56%	98.23%	99.34%	99.78%	99.78%	99.78%	84.11%	85.21%	97.57%	83.66%	47.45%
	< 20	99.56%	99.78%	99.78%	99.56%	99.56%	100.0%	99.78%	99.78%	99.78%	95.81%	94.92%	99.56%	86.09%	66.52%
Action Units (2150)	< 10	88.93%	98.60%	93.81%	98.37%	86.41%	95.44%	97.95%	98.79%	98.79%	68.30%	70.50%	89.48%	72.59%	35.24%
	< 12	95.21%	99.63%	97.49%	99.40%	93.30%	97.91%	99.16%	99.72%	99.67%	78.03%	80.97%	95.30%	81.06%	44.62%
	< 15	99.16%	99.95%	99.49%	99.77%	98.28%	98.79%	99.81%	99.86%	99.91%	88.97%	91.07%	98.37%	87.72%	58.18%
	< 20	99.86%	99.95%	99.86%	99.91%	99.77%	99.12%	99.91%	99.91%	99.91%	97.30%	97.77%	99.53%	90.88%	75.62%
Occlusions (381)	< 10	77.31%	95.00%	91.67%	97.02%	83.95%	94.71%	94.57%	94.57%	97.52%	92.98%	96.23%	96.58%	92.45%	56.03%
	< 12	86.15%	96.67%	96.39%	98.10%	88.25%	96.83%	97.43%	97.01%	98.45%	97.54%	97.60%	97.95%	95.68%	67.38%
	< 15	95.00%	97.33%	98.06%	98.64%	95.70%	98.41%	98.29%	98.10%	98.76%	98.60%	98.97%	98.97%	98.20%	79.08%
	< 20	98.08%	98.33%	99.17%	98.92%	97.71%	98.68%	98.57%	98.91%	100.0%	98.60%	99.32%	99.32%	99.28%	92.20%
Yaw Rotations (up to 45°) (525)	< 10	80.05%	89.19%	85.52%	98.33%	91.19%	89.89%	95.41%	99.05%	94.65%	95.15%	96.90%	94.29%	88.19%	49.04%
	< 12	90.29%	91.08%	91.81%	99.05%	94.76%	95.23%	97.25%	99.52%	96.75%	96.94%	97.86%	96.19%	93.90%	61.73%
	< 15	95.28%	95.68%	97.14%	99.52%	98.10%	98.66%	98.17%	99.52%	98.09%	98.21%	99.29%	98.10%	98.10%	75.38%
	< 20	97.90%	97.84%	97.90%	99.52%	99.29%	99.05%	98.62%	99.52%	99.04%	99.49%	99.52%	99.05%	98.29%	91.15%
Pitch Rotation (419)	< 10	89.98%	98.33%	94.27%	99.28%	84.25%	93.32%	98.33%	98.81%	98.79%	94.51%	96.90%	97.37%	94.98%	55.56%
	< 12	95.70%	99.28%	98.09%	99.76%	91.65%	95.47%	99.52%	99.52%	99.52%	97.37%	97.85%	98.57%	97.13%	66.67%
	< 15	98.57%	99.76%	99.05%	99.76%	97.37%	97.85%	99.76%	99.76%	99.76%	99.28%	98.81%	99.76%	98.56%	80.43%
	< 20	99.76%	99.76%	99.52%	99.76%	99.28%	98.33%	100.0%	100.0%	100.0%	99.52%	99.52%	99.76%	99.04%	92.03%
Cross Rotation (Yaw and Pitch) (211)	< 10	73.17%	83.33%	85.78%	96.67%	87.20%	90.00%	81.82%	98.10%	95.73%	85.61%	93.36%	94.31%	90.05%	49.28%
	< 12	84.15%	85.29%	92.89%	100.0%	91.94%	94.76%	90.91%	99.52%	98.58%	91.67%	95.73%	97.16%	94.79%	62.32%
	< 15	91.46%	93.14%	98.10%	100.0%	98.10%	98.57%	90.91%	100.0%	100.0%	97.73%	100.0%	100.0%	98.58%	78.26%
	< 20	98.78%	99.02%	99.05%	100.0%	100.0%	100.0%	93.94%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	91.30%
All the above (4339)	< 10	86.87%	97.17%	92.40%	98.34%	86.65%	94.28%	97.44%	98.20%	98.17%	77.04%	79.80%	92.16%	79.74%	41.08%
	< 12	93.74%	98.22%	96.59%	99.34%	92.86%	97.23%	98.87%	99.41%	99.20%	84.54%	86.70%	96.28%	86.54%	51.34%
	< 15	98.15%	99.15%	99.03%	99.64%	98.02%	98.75%	99.50%	99.67%	99.58%	91.93%	93.48%	98.63%	91.47%	64.99%
	< 20	99.49%	99.57%	99.49%	99.74%	99.50%	99.17%	99.66%	99.79%	99.81%	97.89%	98.07%	99.55%	93.58%	80.92%

The training set is composed of 99 faces from the neutral subset of the dataset. The results are provided for the four most commonly used acceptance radii in landmarking to facilitate future results comparison

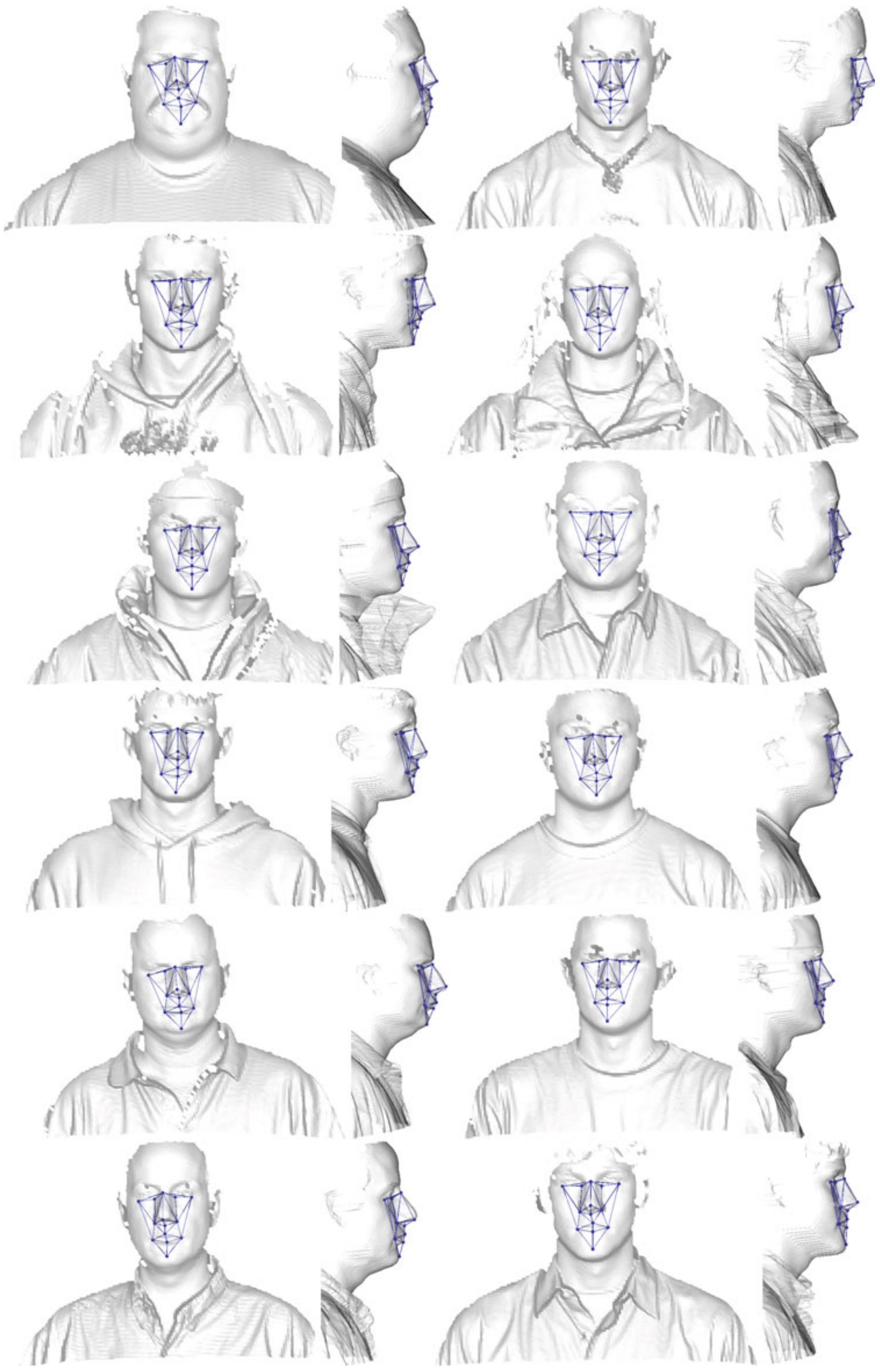


Fig. 32 Examples of landmarks detected by our system. Landmarks are defined by the positions of a scale-adapted landmark model, $\mathcal{L}$, which is a set of L (position, label pairs) for an average face mesh. The model $\mathcal{L}$ is equipped with a triangulation (i.e. it is displayed as a graph) purely to aid visualization. The input face meshes are taken from the FRGC v2 dataset

the direction of occlusion. Combining the DL-scores using these new descriptors can easily be done within our framework.

In Fig. 34, the worst case landmark localizations in terms of the three global registration metrics are presented for the FRGC test set. When registering the ground-truth and the localized landmarks, the transformation is decomposed by steps into a mean translation (aligning the centroids), a scale change (by scaling the mean edge length) and a final rotation, when translation has been canceled and scale equalized.

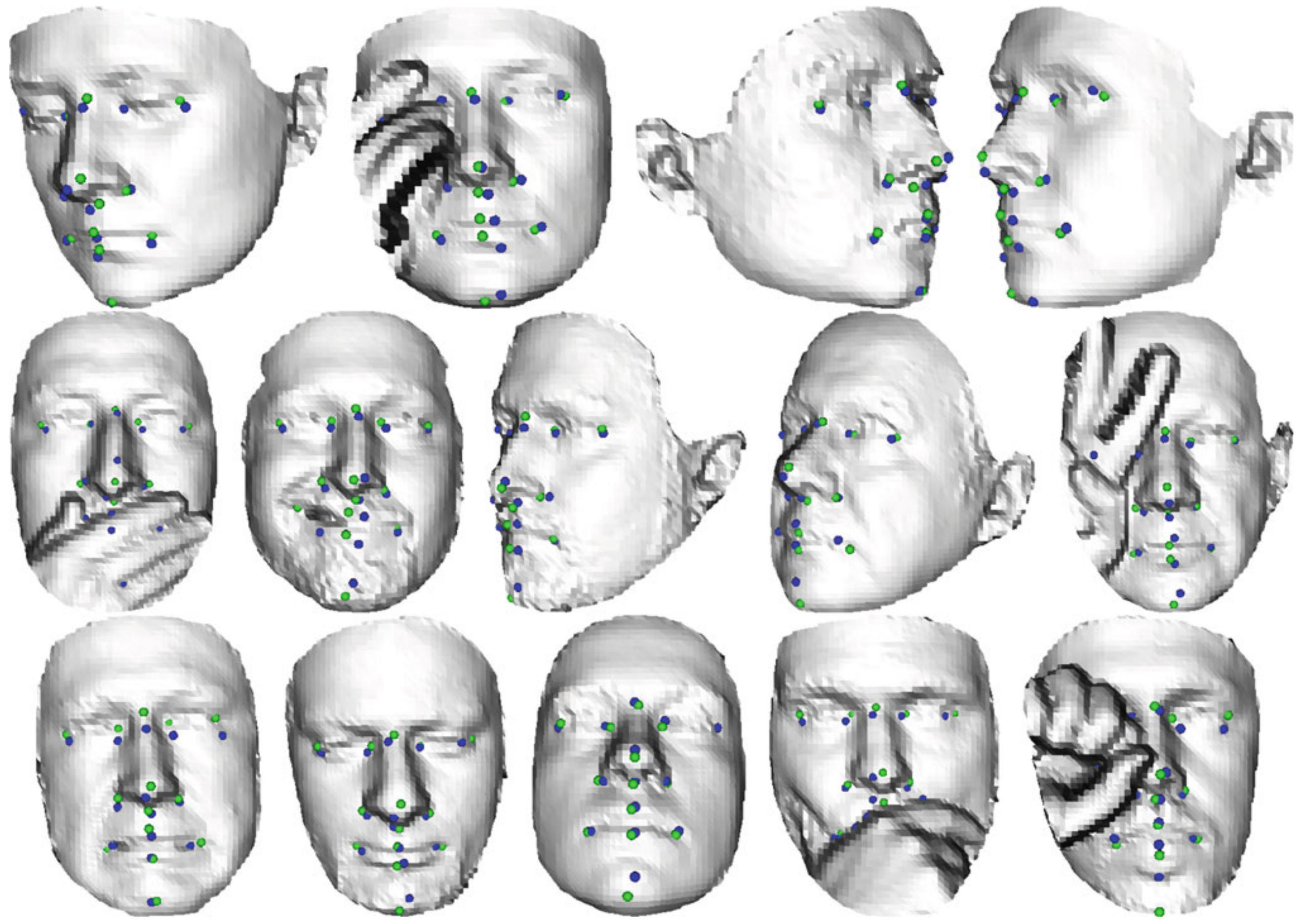


Fig. 33 Examples of landmark localization on the Bosphorus dataset. *Blue* points are our results, *green* points are the ground truth (Color figure online)

The mean error in translation is represented by τ in millimeters. The scale difference is represented by the ratio ρ of the ground truth to the detected mean length. The final rotation error is defined as the angle θ between the unit-quaternion representing the computed rotation and the one representing the identity. Figure 35 shows the distribution of those errors for the FRGC test set.

A limitation of our framework for landmarking is that the matching technique used is very naive and employs a rigid registration of the face to a common model. It is straightforward to build a more shape-adapted model using principal component analysis (PCA) and this may give us an improved landmarking system in terms of the precision of the landmark localizations. However, it is likely that detection of the chin landmark with the mouth open will still be difficult (our current system always fail to detect the chin landmark when the mouth is open). Future work could look at graph and hypergraph matching techniques to find a softer assignment between the keypoint and the target landmark labels (e.g. Creusot et al. 2010). Figure 36 shows examples of discrete failure on the Bosphorus dataset, where a correct coarse registration is not found by the system. A discrete failure is declared if the rotation error θ is above 10° (~ 0.17 rad) or if the translation error τ is above 20 mm.

6.4 Computation Time Performance

Timing data was gathered on a standard dual-core (Intel Core 2 Duo Processor) PC running at 3 GHz and with 4 GB of memory. The total computation time per query scan on the FRGC dataset is 1.18 s for meshes having 3,232 vertices on average. Most of the time is spent on the keypoint detection (0.97 s). The most computationally expensive part of this is the histograms computation 0.70 s. The neighborhood computation costs 0.11 s while the principal curvatures computation takes 0.06 s on average. Big improvements in terms of computational speed have been achieved by modifying the curvature computation and using a Normal solver instead of SVD for the cubic surface fitting. However some parts of the framework remain computationally expensive. The histogram computation for every vertex takes more time than all the rest put together. Indeed, the complexity of the

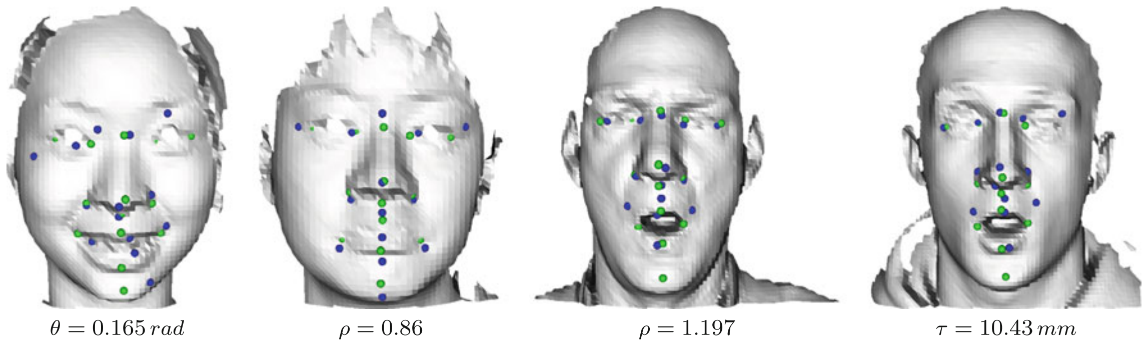


Fig. 34 Four worst cases in the FRGC test set by global registration metric: largest angle, smallest scale ratio, largest scale ratio, largest mean translation

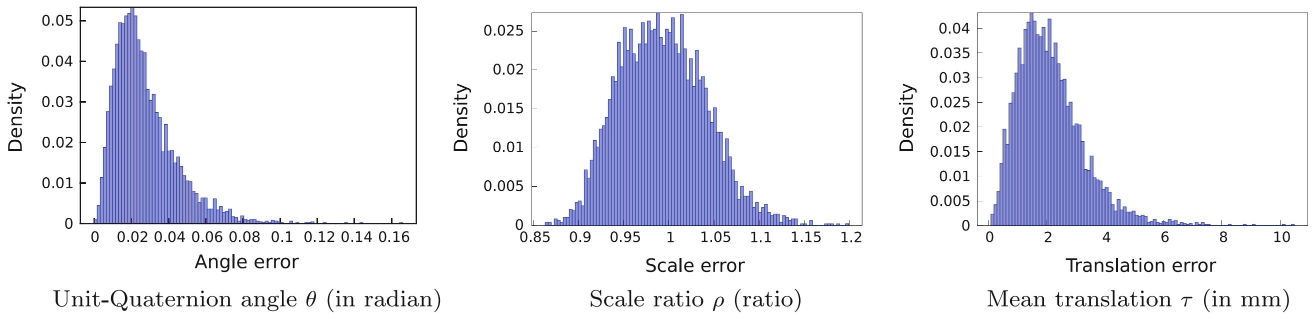


Fig. 35 Distribution of the global registration errors on the FRGC test set

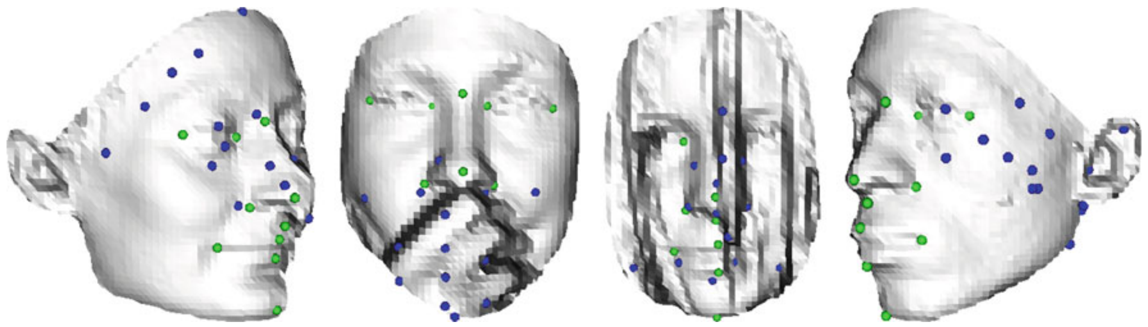


Fig. 36 Examples of discrete failures on the Bosphorus dataset. The 17 failures detected on the 4,339 scans (0.39 %) are mainly due to occlusion (7 scans) and rotation (six 'yaw' and one 'pitch' scans). The three remaining failures are on scans that show exaggerated facial expressions

naive algorithm used to get the histograms is quadratic in the number of vertices. The speed can be improved by using better structures for locality retrieval, for example an octree structure. The computation of DL-score and landmark score maps is performed under 0.03 s. The final matching using RANSAC takes 0.18 s with 130 landmark candidates per face on average.

The total computation time per input scan on the Bosphorus dataset was 0.55 s for meshes having 1,879 vertices on average.

7 Conclusions

A simple method has been proposed to deal with keypoint detection and landmarking of learned local shapes. This method requires landmarks on training meshes to be manually marked up with a set of landmarks defined in a landmark model. One can think of this landmark model as being augmented by learned keypoint detector functions, f_λ, (one per landmark), and this can then be used to automatically landmark unseen query meshes.

A flexible aspect of our method is that it doesn't assume that detected points on the mesh should have an extremal value over a descriptor map. Instead, it assumes that the matching score of this descriptor against a learned distribution should be maximal.

While other techniques are landmark-dependent, our can be be applied to any shape of interest as long as training is provided. The same method is used to detect the nose, an eye's corner or the chin. We detected 14 facial features, while expert system methods are usually limited to a few salient features (see Table 1).

While being more fuzzy (many-valued) compared to expert system methods, we believe that this kind of approach is necessary to deal with uncontrolled input data. In particular, it is more likely to be successful for non-cooperative face preprocessing where there are great uncertainties about what is present in the query scan.

In our opinion, the main gain in performance in the future will come from adding new local descriptors that better deal with occlusions and profile views, leading to less ambiguous keypoints. Local refinement of coarse landmark localization will also be essential to gain precision and high retrieval rates at low acceptance radii.

References

Alyuz, N., Gokberk, B., & Akarun, L. (2010). Regional registration for expression resistant 3-d face recognition. *IEEE Transactions on Information Forensics and Security, 5*(3), 425–440. doi:10.1109/TIFS.2010.2054081.

Ben Azouz, Z., Shu, C., & Mantel, A. (2006). Automatic locating of anthropometric landmarks on 3d human models. In *Third international symposium on 3D data processing, visualization, and transmission* (pp. 750–757, 14–16). doi:10.1109/3DPVT.2006.34.

Berretti, S., Bimbo, A. D., & Pala, P. (2010). Recognition of 3d faces with missing parts based on profile networks. *1st ACM workshop on 3D object. Retrieval (ACM 3DOR'10)* (pp. 81–86). doi:10.1145/1877808.1877825.

Besl, P., & McKay, N. (1992). A method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(2), 239–256. doi:10.1109/34.121791.

Boyer, E., Bronstein, A. M., Bronstein, M. M., Bustos4, B., Darom, T., Horaud, R., et al.. (2011). Shrec 2011: robust feature detection and description benchmark. Eurographics workshop on 3D object. Retrieved. doi:10.2312/3DOR/3DOR11/071-078.

Castellani, U., Cristani, M., Fantoni, S., & Murino, V. (2008). Sparse points matching by combining 3d mesh saliency with statistical descriptors. *Computer Graphics Forum, 27*(2), 643–652. doi:10.1111/j.1467-8659.2008.01162.x.

Chang, K. I., Bowyer, K., & Flynn, P. (2006). Multiple nose region matching for 3d face recognition under varying facial expression. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(10), 1695–1700. doi:10.1109/TPAMI.2006.210.

Choi, E., & Lee, C. (2003). Feature extraction based on the bhattacharyya distance. *Pattern Recognition, 36*(8), 1703–1709. doi:10.1016/S0031-3203(03)00035-9.

Colbry, D., Stockman, G., & Jain, A. (2005). Detection of anchor points for 3d face verification. In *IEEE computer society conference on computer vision and pattern recognition—workshops, 2005. CVPR workshops* (pp. 118–118, 25–25). doi:10.1109/CVPR.2005.441.

Creusot, C. (2011). *Automatic landmarking for non-cooperative 3D face recognition*. Ph.D. thesis, University of York. http://etheses.whiterose.ac.uk/2274/.

Creusot, C., Pears, N., & Austin, J. (2010). 3D face landmark labelling. In *Proceedings of the ACM workshop on 3D object retrieval. ACM, 3DOR '10* (pp. 27–32). doi:10.1145/1877808.1877815.

Creusot, C., Pears, N., & Austin, J. (2011). Automatic keypoint detection on 3d faces using a dictionary of local shapes. In *2011 International conference on 3D imaging, modeling, processing, visualization and transmission (3DIMPVT)* (pp. 204–211). doi:10.1109/3DIMPVT.2011.33.

D'Hose, J., Colineau, J., Bichon, C., & Dorizzi, B. (2007). Precise localization of landmarks on 3d faces using gabor wavelets. In *First IEEE international conference on biometrics: theory, applications, and systems, 2007. BTAS 2007* (pp. 1–6). doi:10.1109/BTAS.2007.4401927.

Dibeklioglu, H., Salah, A., & Akarun, L. (2008). 3d facial landmarking under expression, pose, and occlusion variations. In *BTAS08* (pp. 1–6). doi:10.1109/BTAS.2008.4699324.

Faltemier, T., Bowyer, K., & Flynn, P. (2008). Rotated profile signatures for robust 3d feature detection. In *8th IEEE international conference on automatic face gesture Recognition, 2008. FG '08* (pp. 1–7, 17–19). doi:10.1109/AFGR.2008.4813413.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395. doi:10.1145/358669.358692.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*, 119–139. doi:10.1006/jcss.1997.1504.

Goldfeather, J., & Interrante, V. (2004) A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics, 23*(1), 45–63. doi:10.1145/966131.966134.

Itskovich, A., & Tal, A. (2011). Surface partial matching and application to archaeology. *Computers & Graphics, 35*(2), 334–341. doi:10.1016/j.cag.2010.11.010.

Johnson, A., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(1), 433–449.

Kim, J.-S., & Choi, S.-M. (2009). Symmetric deformation of 3d face scans using facial features and curvatures. *Computer Animation and Virtual Worlds, 20*, 289–300. doi:10.1002/cav.v20:2/3.

Lowe, D. G., (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110. doi:10.1023/B:VISI.0000029664.99615.94.

Max, N. (1999). Weights for computing vertex normals from facet normals. *Journal of Graphics Tools, 4*, 1–6. http://portal.acm.org/citation.cfm?id=334709.334710.

Mayo, M., & Zhang, E. (2009). 3D face recognition using multiview keypoint matching. *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009. AVSS '09* (pp. 290–295). doi:10.1109/AVSS.2009.11.

Mian, A., Bennamoun, M., & Owens, R. (2010). On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision, 89*(2), 348–361. doi:10.1007/s11263-009-0296-z.

Mian, A. S., Bennamoun, M., & Owens, R. (2008). Keypoint detection and local feature matching for textured 3d face recognition. *International Journal of Computer Vision, 79*(1), 1–12. doi:10.1007/s11263-007-0085-5.

Mian, A. S., Bennamoun, M., & Owens, R. A. (2006). Automatic 3d face detection, normalization and recognition. In *3DPVT* (pp. 735–742). doi:10.1109/3DPVT.2006.32.

Pears, N., Heseltine, T., & Romero, M., (2010). From 3D point clouds to pose-normalised depth maps. *International Journal of Computer Vision, 89*(2), 152–176. doi:10.1007/s11263-009-0297-y.

Phillips, P., Flynn, P., Scruggs, T., Bowyer, K., Chang, J., Hoffman, K., et al. (2005). Overview of the face recognition grand challenge. In *IEEE computer society conference on computer vision and pattern recognition, 2005. CVPR 2005* (Vol. 1, pp. 947–954). doi:10.1109/CVPR.2005.268.

Romero, M., & Pears, N., (2009). Landmark localisation in 3d face data. In *Sixth IEEE international conference on advanced video and signal based surveillance, 2009. AVSS '09* (pp. 73–78). doi:10.1109/AVSS.2009.90.

Romero-Huertas, M., & Pears, N. (2008). 3D facial landmark localisation by matching simple descriptors. In *2nd IEEE international conference on biometrics: theory, applications and systems, BTAS 2008* (pp. 1–6). doi:10.1109/BTAS.2008.4699390.

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision, 40*, 99–121. doi:10.1023/A:1026543900054.

Savran, A., Alyüz, N., Dibeklioğlu, H., Çeliktutan, O., Gökberk, B., Sankur, B., et al. (2008). Bosphorus database for 3d face analysis. In *Biometrics and identity management: first European workshop, BIOID 2008* (pp. 47–56). Springer: Roskilde, Denmark. doi:10.1007/978-3-540-89991-4_6.

Segundo, M., Queirolo, C., Bellon, O., & Silva, L., (2007). Automatic 3D facial segmentation and landmark detection. In *14th International conference on image analysis and processing, 2007 (ICIAP 2007)* (pp. 431–436). doi:10.1109/ICIAP.2007.4362816.

Segundo M., Silva L., Bellon O. R. P., & Queirolo C. C., (2010). Automatic face segmentation and facial landmark detection in range images. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 40*(5), 1319–1330. doi:10.1109/TSMCB.2009.2038233.

Shotton, J., Johnson, M., & Cipolla, R., (2008). Semantic texton forests for image categorization and segmentation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1–8). doi:10.1109/CVPR.2008.4587503.

Szeptycki, P., Ardabilian, M., & Chen, L., (2009). A coarse-to-fine curvature analysis-based rotation invariant 3D face landmarking. *International conference on biometrics: theory, applications and systems* (pp. 32–37). doi:10.1109/BTAS.2009.5339052.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*, 137–154. doi:10.1023/B:VISI.0000013087.49260.fb.

Zaharescu, A., Boyer, E., Varanasi, K., & Horaud, R., (2009). Surface feature detection and description with applications to mesh matching. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco* (pp. 373–380). doi:10.1109/CVPR.2009.5206748.

Zhao X., Dellandr anda E., Chen L., & Kakadiaris I. A. (2011). Accurate landmarking of three-dimensional facial data in the presence of facial expressions and occlusions using a three-dimensional statistical facial feature model. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(5), 1417–1428. doi:10.1109/TSMCB.2011.2148711.