Utrecht University
Department of Information and Computing Sciences
Master in Game and Media Technology

A template fitting, non-rigid registration algorithm applied to craniofacial depth scans

Author:
Zoilo Guillermo Ibáñez de Aldecoa Marín

Supervisors:
Zerrin Yumak
Frank van der Stappen

September 2018

Acknowledgements

First and foremost, I would like to thank my supervisors Zerrin Yumak and Frank van der Stappen for their constant support, patience and guidance thorough all the stages of this thesis. Secondly, I would like to thank the Utrecht University for offering me the possibility of undertaking this Game and Media Technology master and acknowledging the immense research interest contained in the animation world. Thirdly, I would like to thank my family and Luis for supporting my studies abroad even at the difficulties of living far from home.

Lastly, I would like to thank my colleagues Juan, Micaela and Rutger for accompanying me through these two years and offering me their advice, their company and their precious friendship.

Abstract

As faster and cheaper 3D acquisition systems emerge, so does the need to find methods that are able to deal with their inherent noise and lack of structure. In the particular case of facial and craniofacial scans, the available databases present abrupt changes on resolution, significant noise and holes. Aside from defects on the regularity of the meshes, there is not a vertex-to-vertex correspondence between scans of different subjects. Correspondence has traditionally been a crucial part for building morphable models and more recently, for building structured databases for machine learning applications. In this thesis, we present a template-fitting, non-rigid registration algorithm for morphing a clean and smooth template towards facial and craniofacials depth scans. Our approach employs an iterative closest point algorithm that minimizes the distance between correspondent points at the same time as imposing constraints on the deformations to preserve the smoothness and cleanness of the original mesh. We tested our framework with two modern datasets with positive results in capturing the user identity and dealing with missing parts, noise and low resolution meshes. Meshes processed with the same template also presented good correspondence between vertices of the same indexes. Finally, our algorithm shows comparable performance to state of the art approaches at minimizing the per-vertex difference between the morphed template and the scanned faces.

Contents

Chapter 1

Introduction

This thesis presents a template fitting, non-rigid registration algorithm applied to face depth scans. This brief chapter introduces the motivation and contributions of our work. Chapter 2 presents a literature review of non-rigid registration and describes the employed databases. Chapter 3 presents the differential geometry concepts applied in our method. Chapter 4 describes in detail our framework. Chapter 5 presents an evaluation of our results. Finally, Chapter 6 summarizes and concludes our work.

1.1 Motivation

Achieving a faithful reconstruction of a human face is arguably one of the most challenging and studied problems of modern Geometry Processing. Recent acquisition systems allow us to produce depth scans composed of vertices and faces by triangulating the images of multiple cameras (usually called *stereo cameras*) or even from a sequence of frames from the same camera. Although these systems are approaching face reconstruction to consumer-level applications because of their interactive rates and low cost, such speed and accessibility comes at the cost of producing meshes with low resolution, abundant noise and holes at the areas uncovered by the cameras or unsolved in the reconstruction process.

Apart from the mentioned issues, scans of different faces (or even scans of the same face at different time frames) do not have vertex-to-vertex correspondence, i.e., vertices of the same index number do not correspond to the same features across different scans. Vertex correspondence is a very desirable property because it allows the construction of consistent facial databases where all the samples are in direct relation with each other. Facial databases have been widely employed for the construction of morphable models [6], a statistical representation in a low dimensional space that can describe a wide variety of face shapes with a reduced set of parameters. Recent works have also leveraged from facial databases to obtain a compact representation of shapes that can be used to relate images of faces to 3D meshes trough convolutional networks [44, 40].

The process of obtaining clean and corresponding meshes from depth scans is typically carried out by registering a template mesh towards the scan. The template mesh serves as a strong prior for the registration as it will morph towards the scan preserving as much as possible its original topology. However, most rigid registration frameworks focus on obtaining good results for a specific face scan database and do not assess in detail what are the consequences of variations of face proportions, expressiveness, holes and scan resolution in the resulting meshes appearance and correspondence.

In this thesis, we present a method to morph a template mesh towards a face scan. This process is commonly called template fitting. The template is an artist-made, generic face mesh that adopts the subject identity (its features) while preserving its smoothness. The template deforms iteratively, adapting itself to the subject geometry. Some mathematical constraints are imposed on the vertex displacements so as to obtain smooth deformations that are unaffected by the scan's noise and holes. The result is a “clean version” of the scanned subject. We demonstrate that our method is robust to holes, noise and changes in resolution while also faithfully adapting to the user identity. The resulting meshes of different subjects are in good correspondence, meaning that the same features correspond to the same vertex indexes.

1.2 Research Questions

Considering that there are certain aspects that seem to not be well understood in the registration of facial depth scans, we aim to answer the following research questions in this thesis:

RQ1: *How can a non-rigid registration framework adapt to different face proportions?*

RQ2: *How can a non-rigid registration framework retrieve a smooth mesh from a scan in the presence of holes and noise?*

RQ3: *How can a non-rigid registration adapt to meshes of different resolutions?*

For answering RQ1, we designed an adaptative template method that employs corresponding landmarks on the template and the depth scan. By bringing the landmarks together and imposing smoothness constraints we obtain a better initialization for non-rigid registration. For answering RQ2 and RQ3, we investigated local ridigity contraints that minimize the relative changes of distances between neighboring vertices. This causes the vertices of the template to move coherently in group and ignore correspondences that would cause severe local deformations.

1.3 Evaluation

To validate our method, we test our framework with two face mesh databases constructed with 3dMD[1] stereo cameras, the D3DFACS [19] and Headspace [22]. We also compare Dai's performance measures of two modern morphable models: the Open Framework [35] and Large Scale Facial Model (LSFM) [7]. Our results demonstrate that our method's performance is comparable with previous and state-of-the art non-rigid registration frameworks at minimizing the difference between a depth scan and the morphed template.

Chapter 2

Literature review and materials

Overview

In the context of facial animation and facial geometry processing, non-rigid registration has been traditionally an intermediary step for other means. Constructing a morphable model, for example, requires a database of face meshes that are in full correspondence. Given a set of face scans acquired with any system, one can produce samples for the database by morphing a generic template towards them [6, 22].

In facial animation methods that require interactive rates, non-rigid registration is very often limited to fit a morphable model towards a scan [53] to obtain an approximation of the user geometry. This is because acquisition systems that work in real time (such as RGBD cameras) typically present too much noise to faithfully reconstruct more than a rough estimation of the shape features. The fitted model is animated afterwards commonly employing blendshapes [29, 13]. As a result, the user identity is not captured with fine detail, even if some works present refinement steps for capturing small scale features such as wrinkles or mouth corners [14].

Non-rigid, Iterative Closest Point algorithms are unsuited for real-time applications due to their higher complexity and non-deterministic convergence. Their reconstruction quality rely on a certain mesh quality, so for facial animation techniques these algorithms are employed offline on a higher quality mesh produced by, for example, accumulated consecutive depth scans of the user on a fixed posture and expression [11].

2.1 Morphable Models and Active Appearance Models

Parametric model fitting has been the main procedure for registering faces towards depth scans and monocular images [14, 26]. These models are constructed or synthesized from real scanned samples and therefore are able to produce a wide spectrum of realistic heads and faces by just tuning reduced set of parameters. This greatly simplifies the optimization problem of fitting a head model to a subject from solving thousands of linear equations for the 3-dimensional vertex positions to solving for dozens or hundreds of parameters.

One of the earliest and most influential works on constructing parametric facial models can be found in [6]. In their paper, Blanz and Vetter devise a technique capable of synthesizing 3D faces from photographs. The core of their method is the 3D Morphable Model (3DMM), a statistical representation on a low dimensional space that can separately describe the shape and the texture of a head with a coefficient vector. To obtain the model, the authors employed a database with hundreds of textured 3D scans of faces and performed Principal Component Analysis (PCA) on the data to obtain the directions of maximum variance (eigenvectors). The shape and texture of a new face can be then obtained with a sum of the average shape $\bar{S}$ and texture $\bar{T}$ with a weighted linear combination of the m shape eigenvectors s_i and the n texture eigenvectors t_i:

$$S_i = \bar{S} + \sum_{i=1}^{m} \alpha_i s_i, \quad T_i = \bar{T} + \sum_{i=1}^{n} \beta_i t_i \tag{2.1}$$

where α_i and β_i are coefficients that typically range between 1 and -1. The problem of synthesizing a new face from a photograph is simplified to find the eigenvector weights and illumination parameters that minimize per-pixel intensity differences between the photograph and the aligned, projected 3D face.

Numerous publications have leveraged from this work to obtain a static facial reconstruction of the user from images [55] or video [48, 53, 9, 29]. However, 3DMM struggle at synthesizing variations in expression without altering the overall shape of the reconstructed face.

The Active Appearance Model (AAM), developed by Cootes et al.[18] was one of the first facial models that captured dynamic information in the form of texture changes. First, a image dataset depicting faces performing different facial expressions with marked landmarks is collected. Every set of landmarks is aligned using Procrustes Analysis [41] to build a statistical shape model from the eigenvectors of the landmark locations. Then, all faces are warped towards a mean face shape to obtain a “shape free patch” and the eigenvectors of intensity and color values of the patch are retrieved (see Figure 2.1). Similar to the 3DMM, any face can be described as two weighted linear combinations of the shape x_i and texture g_i eigenvectors plus the mean shape $\bar{x}$ and texture $\bar{g}$:

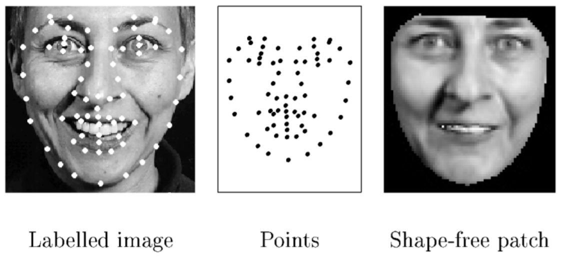


Figure 2.1: Active Appearance Model data representation. For every labelled face in the dataset (left), its distribution of landmark points (middle) and its color information warped towards a mean face (right) are collected to retrieve a shape and a texture sample. Image retrieved from [18].

$$x = \bar{x} + \sum_{i=1}^{m} c_i x_i, \quad g = \bar{g} + \sum_{i=1}^{n} d_i g_i \tag{2.2}$$

AAMs have been widely used in automatic speech animation to associate changes in texture and shape around the mouth area with auditory cues [27, 25, 50]. However, they have been struggling with animating faces unseen in the database since subtle changes in posture, face structure and illumination can produce wrong landmark estimation and significant texture changes [51].

2.1.1 The blendshape model

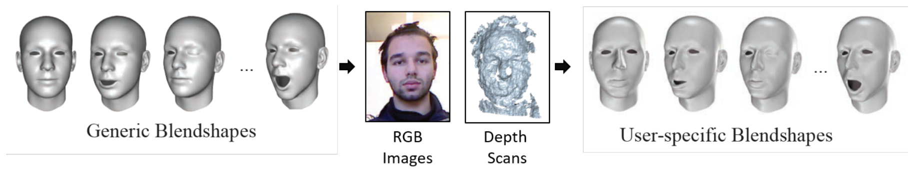


Figure 2.2: The standard processing pipeline of facial motion capture algorithms include generic blendshapes which are adapted to the user's facial geometry through RGB images and/or depth images. From there, the problem is simplified to finding at every instant the blendshape weights that reconstruct best the user's expression. Image composed from [53].

The blendshape model is currently the most common approach to obtain expressive models [32] and it is widely used in facial reconstruction algorithms (see Figure 2.2). Blendshapes are variations of the same facial face mesh showing different expressions or movements, e.g, smiling, blinking or frowning. New faces can be generated by adding a linear combination of blendshapes to a neutral face. A common algebraic way of representing new faces F from blendshapes is the following [11]:

$$F = b_0 + \sum_{i=1}^{n} w_i \Delta b_i$$

where n is the number of blendshapes, b_i is the stacked vertex coordinate vector of the blendshape i, b_0 is the face in resting pose, Δb_i is the difference $[b_0 - b_i]$, and w_i are weights between 0 and 1 that typically $\sum_i^n w_i = 1$. If a generic blendshape set is given, a common approach for facial motion tracking is to obtain a static model of the user and "copy and paste" the deformations of the generic set to the static model using the deformation transfer operator of Sumner and Popovic [47].

The blendshape model simplifies the problem of facial reconstruction to infer the rigid alignment of the head and the blendshape weights at every video frame or image. This also implies that in online applications the volume of data to be transferred to animate a character is reduced [29]. However, as discussed in [53], the number of blendshapes limits the expressiveness of the model and introducing many blendshapes could cause multiple combinations of weights to result in the same face, causing unstable tracking on noisy data. Furthermore, a blendshape model is best suited to coarse and medium detail reconstruction: fine-scale details such as emerging and disappearing wrinkles must be inferred in a post-processing step since they tend to get smoothed in the linear combination.

2.1.2 Non-rigid, Iterative Closest Point

Several non-rigid registration methods focused on craniofacial reconstruction [16, 45] have proposed a non-rigid version of the classic Iterative Closest Point (ICP) algorithm [5]. These methods have shown good performance at capturing the user identity but either they suffer from overfitting when points copy the position of their correspondences or they are too constrained to capture fine details of the user geometry. To close the gap between a too detailed mesh and a too rough one, we combined the ideas from the following three works:

- Amberg [2] poses non-registration as an energy minimization equation with a distance term, a *stiffness term* and a landmark term. The distance term minimizes the distance between corresponding points. The stiffness term, regulated by a stiffness parameter α, forces neighbouring vertices to follow similar transformations. The landmark term attempts to bring corresponding landmarks between the meshes together. The equation is not solved for the vertices positions but for affine 4×4 transformation matrices, one for each source vertex. The derivations of the formula result on a well-posed linear system that when solved reaches a global minimum. The user defines an initial and final value for the α. When convergence is reached, the stiffness parameter becomes lower to allow vertices to move towards their correspondences and the equation is solved again. This process is repeated as many times as the user desires.

- Weise et al. [52] builds a personalized template by deforming a generic template of a face towards the depth scan of a user. The problem of morphing is posed as an energy minimization equation with a distance term, a landmark term and a optical flow that employs image features. Its most important feature for this work is the inclusion of a membrane term that smooths the deformations on the template mesh by minimizing the Laplacian of the deformation vector between the original template and the deformed template. The membrane term is a strong regularizer that causes more natural, elastic like deformations.

- Very recently, Dai et al. [20] developed a framework for non-rigid shape registration using an adaptative template. The adaptative template is a deformed version of a generic template that roughly matches the user geometry and serves as a good prior for non-rigid registration. The method requires a depth scan of the user and a set of images of him/her at different perspectives. 2D facial landmarks are extracted from the images and triangulated with the extrinsic parameters of each camera to obtain their 3D position. The landmarks are projected onto the user mesh to obtain their corresponding vertices. After aligning the landmarks of a generic template and the scan, the template is smoothly deformed through Laplacian constraints to obtain an adapted template of the user. Lastly, the adapted template is deformed towards the scan by employing Coherent Point Drift [37] with iterative closest point correspondences.

2.2 Databases

We employ two modern databases for the evaluation of our results in Chapter 5, the D3DFACS and the Headspace dataset. Both datasets contain facial depth scans constructed from stereo cameras. In terms of reconstruction quality and system complexity, stereo cameras are placed at the middle of the spectrum compared with consumer level depth cameras (placed at the low end due to their abundant noise) or range scan lasers (placed at the high end for their fine resolution and sophisticated technology). The samples provided by these databases are able to capture the facial features of the subjects while at the same time they present noise on reflective and dark surfaces, holes on the areas uncovered by the cameras and abrupt changes of triangle resolution. This medium quality allows us to study the results on non-rigid registration directly to the raw scans with little-to-no scan pre-processing. Furthermore, the subjects are varied in identity, expressions and resolution which allow us to test our method in a wide range of situations.

A table summarizing their features can be seen below.

Property	D3DFACS	HeadSpace
# of subjects	10	1518
Textured	Yes	Yes
Resolution	16∼23k vertices	∼180k vertices
Capture area	Facial	Craniofacial
Capture system	Stereo Camera	Stereo Camera
Expressions	25∼40 per user	Only Neutral

2.2.1 D3DFACS database

The D3DFACS[19] is a mesh dataset of facial expressions containing 519 Action Unit sequences of 10 users. The 3D meshes are reconstructed from a Dynamic 3D Stereo system using a combination of six cameras. Since the reconstruction relies on the optical flow of the reconstructed texture[49], several irregularities frequently occur on the meshes (see Figure 2.3). They present isolated elements, discontinuous boundaries and holes on missing viewpoint areas. Furthermore, in using texture there is the implicit assumption that the human face is a Lambertanian surface (a perfect matte surface), making the reconstruction of reflective parts noisy. This issue is particularly noticeable around the eyes which often lack detail and present significant noise. Lastly, the meshes do not have a 1-to-1 vertex correspondence between users and consecutive frames.

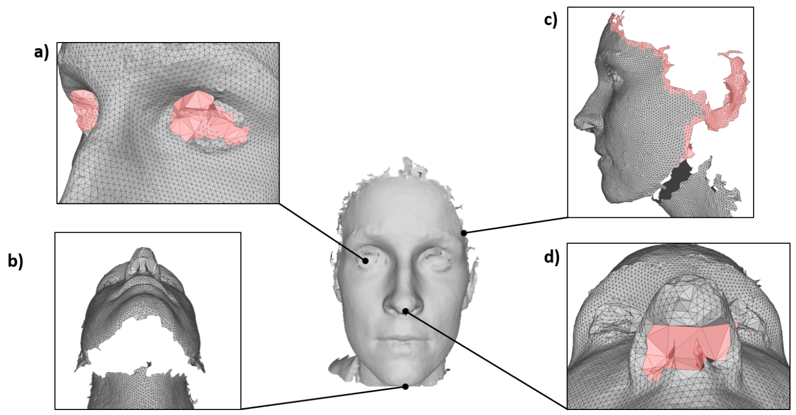


Figure 2.3: Common defects present on the D3DFACS face scans. a) Lack of detail and noise on the eyes b) Holes at the missing viewpoint areas c) Disjointed surfaces and discontinued boundaries. d) Stretches at nostrils.

2.2.2 Headspace database

Headspace [21] is a publicly available database that contains 1518 textured high resolution head meshes captured with a five-stereo camera system. Each subject wears a cap to diminish the effect of hairstyles in the reconstruction. The database comprises a wide range of ages and a balanced proportion of gender. The majority of subjects include a set of 68 facial landmarks extracted by employing the Zhu and Ramanan landmark detection algorithm [56]. The landmarks of the five images of the camera are triangulated and annotated as vertex indexes of the mesh. The scans present little to no noise at the key face features but some irregularities can be observed (see Figure 2.4). There are some abrupt changes of triangle densities around the nostrils and the back of the jaw, stretches and spurious triangles around the ears and small holes presumably produced at areas not covered well by the cameras. Lastly, the meshes present a different number of vertices and they are not in full correspondence.

The template fitting framework of Dai. et al [20] was designed for the samples of the Headspace dataset, thus we can establish a straightforward comparison between this and our method performance.

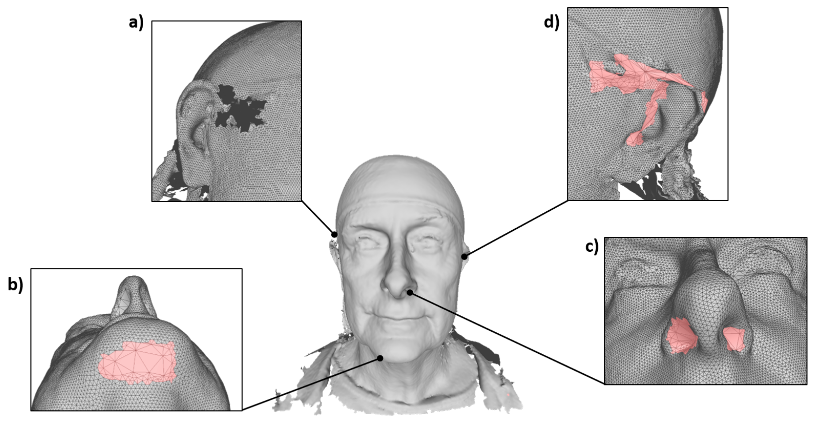


Figure 2.4: Common defects present on the Headspace face scans. a) Holes on missing viewpoint areas. b) Stretches at the back of the jaw c) Spurious faces around ears. d) Stretches at nostrils.

Chapter 3

Differential Geometry

Overview

This chapter contains fundamental concepts of differential geometry necessary for understanding some elements of our template fitting algorithm. The concepts are first explained for continuous surfaces and they are later expanded for discrete meshes. Specifically, we are interested in finding a definition of mean normal curvature, differential area, vertex normal direction and Laplace-Beltrami operator for piecewise linear meshes. The reasons are the following:

- We employ the vertex normal direction for designing a distance-to-normal function to establish pairwise correspondences between two meshes. We also use the vertex normal direction to prune bad correspondences on Section 4.5.2.
- The Laplace-Beltrami operator, differential area and mean normal curvature are employed for imposing local smoothness constraints on the deformations of our template (see Section 4.4). We calculate the Laplace-Beltrami operator on the vertices of the original template and the deformed template to obtain shape descriptors based on local curvature. To preserve the original smoothness of the template, we attempt to minimize the distance between the two descriptors.

Most of this chapter is an adaptation of the Differential Geometry section (Chapter 3) of [8].

3.1 Curvature 1-manifolds

Let us consider a smooth, 1-manifold- planar curve. Such curves can be represented in a parametric form by a vector valued function $\mathbf{x} : [a, b] \to \mathbb{R}^2$. Any point on the curve can be obtained by selecting a parameter value $u \in [a, b]$ such that $\mathbf{x}(u) = (x(u), y(u))^T$. The *tangent vector* $\mathbf{x}'(u)$ of the curve is defined as the derivative of the vector valued function $\mathbf{x}'(u) = (x'(u), y'(u))$. Its *unit normal vector* $\mathbf{n}(u)$ can be obtained by rotating the tangent vector 90 degrees $\mathbf{n}(u) = \mathbf{x}'(u)/\|\mathbf{x}'(u)\|$.

To measure the length l between two points $c, d \in [a, b]$ on the curve, we can calculate the integral of the tangent vector across the segment:

$$l(c,d) = \int_c^d \|\mathbf{x}'(u)\|\, du \tag{3.1}$$

It is possible to reparametrize the curve with a length preserving mapping such that the range of values $[a, b]$ are mapped to $[0, L]$ where $L = \int_a^b \|\mathbf{x}'(u)\|$:

$$s = s(u) = \int_a^u \|\mathbf{x}'(t)\|\, dt \tag{3.2}$$

An intuitive definition of curvature in a planar curve can be a measure of how strongly does the curve deviate from a straight line. If we evaluate the tangent vector between two points, the curvature can be considered the ratio between the change in angle of the tangent divided by the traversed arc length (see Figure 3.1). When the traversed arc becomes infinitesimal, this ratio is actually the derivative of the length-preserving parametrized tangent vector $\mathbf{x}''(s)$.

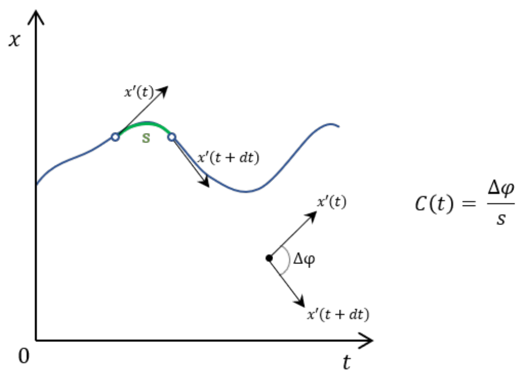


Figure 3.1: Curvature C in 1-manifold curves can be locally defined in t as the ratio of change of angle of the tangent vector $\Delta\varphi$ along the infinitesimal arc length s. Image adapted from [8].

3.2 Curvature in 2-manifolds

There exist multiple definitions of curvature in 2-manifold surfaces, each with its own interpretation. However, before discussion, it is easier to reapply the concepts explained previously with parametric surfaces $S \subset \mathbb{R}^3$. Following the example in [8], we can parametrize a sphere given its radius by extracting its polar coordinates functions. Two angles from two planes that intersect at its center are needed. If these planes are parallel to the ones spanned by the Cartesian coordinate system, one can then select any point in the parametric space to obtain a mapping to a 3D coordinate point lying on the sphere surface (see Figure 3.2). In this example, the mapping function $\mathbf{x}$ is defined in $\theta \in [-2\pi, 2\pi], \phi \in [-\pi, \pi]$ as follows:

$$\mathbf{x}(\theta, \phi) = \begin{pmatrix} x(\theta, \phi) \\ y(\theta, \phi) \\ z(\theta, \phi) \end{pmatrix} = \begin{pmatrix} R\cos(\phi)\cos(\theta) \\ R\sin(\phi)\sin(\theta) \\ R\sin(\phi) \end{pmatrix} \tag{3.3}$$

where x, y and z are *parametric functions* of the sphere. It can be noticed that fixing one of the parameters and drawing the image of the other one over its value range produces a planar curve in the surface. In this case, the sphere produces a grid pattern made of planar circles, the well-known *meridians* and *parallels*. For the rest of surfaces these curves are called *iso-parameter curves*.

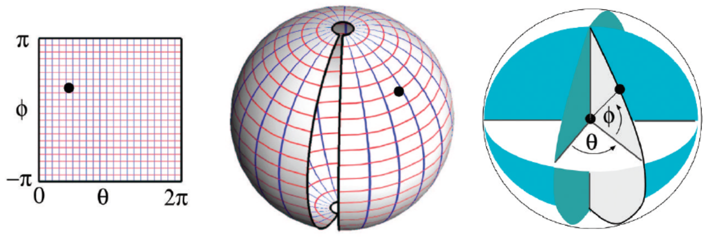


Figure 3.2: A sphere can be parametrized with two angles and its radius. Each point in the parametric space returns a point on the surface of the sphere. Image retrieved from [28].

Although it is out of the scope of this introduction, it can be proven that it is possible to parametrize any surface in a 2-manifold domain Ω in a similar fashion:

$$\mathbf{x}(\theta, \phi) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \quad (u, v) \in \Omega \subset \mathbb{R}^2, \tag{3.4}$$

where x, y and z are differentiable functions defined in Ω. As with planar curves, the metric of the iso-parameter curves can be obtained through their partial derivatives

$$\mathbf{x}_u(u_0, v_0) = \frac{\partial \mathbf{x}}{\partial u}(u_0, v_0), \quad \mathbf{x}_v(u_0, v_0) = \frac{\partial \mathbf{x}}{\partial v}(u_0, v_0) \tag{3.5}$$

These are, respectively, the tangent vectors of the two iso-parameter curves produced by fixing u_0 and v_0:

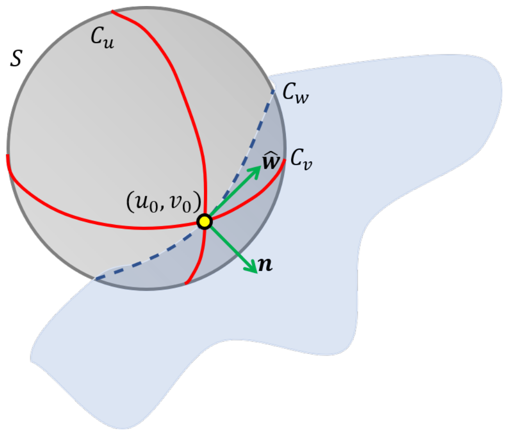


Figure 3.3: A directional derivative can be visualized as the rate of change of the curve formed by intersecting the surface S with the plane spanned by the chosen direction $\hat{w}$ and the normal of the surface n at the point (u_0, v_0) .

$$\mathbf{C_u} = \mathbf{x}(u_0 + t, v_0), \quad \mathbf{C_v} = \mathbf{x}(u_0, v_0 + t) \tag{3.6}$$

It can be observed that the planar curve is in fact a parametrized straight line in the parameter space. If the surface parametrization is *regular*, that is, if $\mathbf{x_u}$ and $\mathbf{x_v}$ are linearly independent [54], the plane spanned by $\mathbf{x_u}$ and $\mathbf{v_u}$ is tangent to the surface S at the evaluated point. We can then define the *surface normal direction* $\mathbf{n}$ as the vector perpendicular to both tangent vectors:

$$\mathbf{n} = \frac{\mathbf{x_u} \times \mathbf{x_v}}{\|\mathbf{x_u} \times \mathbf{x_v}\|} \tag{3.7}$$

We can define an arbitrary *directional derivative* $\mathbf{w}$ at the point $p = (u_0, v_0)$ given a direction $\hat{\mathbf{w}} = (\mathbf{u_w}, \mathbf{v_w})^\top$. Starting from p, following the direction $\hat{w}$ along the surface produces a planar curve. It is easier to visualize this curve as the result of intersecting the surface S with the plane spanned by its surface normal $\mathbf{n}$ and the vector $\hat{\mathbf{w}}$ (see Figure 3.3). Its formal expression is the following

$$\mathbf{C_w} = \mathbf{x}(u_0 + tu_\mathbf{w}, v_0 + tv_\mathbf{w}) \tag{3.8}$$

The directional derivative is defined to be the tangent of this curve at $t = 0$, given by $\partial\mathbf{w} = C_\mathbf{w}/\partial t$. By the chain rule it follows that:

$$\frac{\partial \mathbf{C_w}}{\partial t} = \frac{\partial \mathbf{C_w}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial t} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} (u, v)^\top = \mathbf{Jw} \tag{3.9}$$

Numerically, the directional derivative evaluates, starting at u_0, v_0, the rate of change of the planar curve $\mathbf{C_w}$ produced by following the direction $\hat{\mathbf{w}}$ on the parametric space.

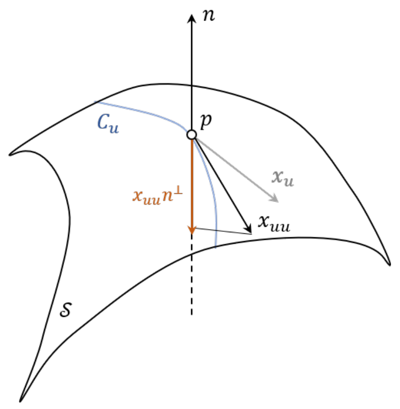


Figure 3.4: Geometric interpretation of the second fundamental form terms. As an example, the term $x_{uu}n^\top$ is the projection of the second derivative of the planar curve C_u on the surface $\mathcal{S}$ (named $C(t)$ in Figure 3.1) a point p. Image adapted from [23].

Geometrically, the Jacobian $\mathbf{J}$ can be interpreted as a linear mapping that transforms a vector on the parameter space to its tangent vector on the surface. Given a tangent vector, it is possible to calculate its squared length following the scalar product formula $\|\hat{\mathbf{w}}\| = (\mathbf{J}\hat{\mathbf{w}})^\top(\mathbf{J}\hat{\mathbf{w}}) = \hat{\mathbf{w}}\mathbf{J}^\top\mathbf{J}\hat{\mathbf{w}}$. The product $\mathbf{J}^\top\mathbf{J}$ is also known as the *first fundamental form* $\mathbf{I}$ in differential geometry. The elements of the matrix have a particular notation that will be referenced in the next sections:

$$\mathbf{I} = \begin{pmatrix} \mathbf{x_u}\cdot\mathbf{x_u} & \mathbf{x_u}\cdot\mathbf{x_v} \\ \mathbf{x_u}\cdot\mathbf{x_v} & \mathbf{x_v}\cdot\mathbf{x_v} \end{pmatrix} = \begin{pmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{F} & \mathbf{G} \end{pmatrix} \tag{3.10}$$

The *second fundamental form* $\mathbf{II}$, whose deduction will not be explained in this chapter, is formulated as:

$$\mathbf{II} = \begin{pmatrix} \mathbf{x_{uu}}\mathbf{n}^\top & \mathbf{x_{uv}}\mathbf{n}^\top \\ \mathbf{x_{uv}}\mathbf{n}^\top & \mathbf{x_{vv}}\mathbf{n}^\top \end{pmatrix} = \begin{pmatrix} \mathbf{e} & \mathbf{f} \\ \mathbf{f} & \mathbf{g} \end{pmatrix} \tag{3.11}$$

where $\mathbf{x_{ij}}$ denotes the second derivative $\partial^2\mathbf{x}/\partial ij$. Given a vector tangent to the surface, each term can be interpreted as the projection of the second derivative of the tangent vector onto the surface normal evaluated at a point p [23](see Figure 3.4).

Normal Curvature

One standard point-defined curvature measure of a surface is the *normal curvature κ_n*. Given a tangent vector $\hat{\mathbf{t}} = u_t x_u + v_t x_v$ and its normal $\mathbf{n}$ defined as in Equation 3.7, one can produce a curve $\mathbf{C}$ by intersecting the plan spanned by these two vectors on the surface. The curvature of said planar curve is:

$$\kappa_n(\hat{\mathbf{t}}) = \frac{\hat{\mathbf{t}}^\top \mathbf{II} \hat{\mathbf{t}}}{\hat{\mathbf{t}}^\top \mathbf{I} \hat{\mathbf{t}}} = \frac{eu_t^2 + 2fu_tv_t + gv_t}{Eu_t^2 + 2Fu_tv_t + Gv_t^2} \tag{3.12}$$

that is, the curvature of the planar curve divided by its *metric*. Notice that this measure is defined for a single tangent vector of the surface. Therefore, one intuitive curvature measure for a point in a surface would be to rotate the tangent vector and evaluate all possible curvatures κ_n. For the following formulation, it is convenient to take one unit vector of the tangent plane e_θ and consider the normal curvature as a function of the rotation angle θ of the unit direction e_θ around the axis formed by the surface normal $\mathbf{n}$:

$$\kappa_H = \frac{1}{2\pi}\int_0^{2\pi} \kappa_n(\theta)d\theta \tag{3.13}$$

It can be proven that there exist two extreme values κ_1 and κ_2 for κ_H called *principal curvatures*. These values correspond to the global maximum and minimum of Equation 3.4. By the Euler's theorem, it is possible to express the normal curvature in terms of its principal curvatures $\kappa(\theta) = \kappa_1 \cos(\theta)^2 + \kappa_2 \sin(\theta)^2$ giving rise to the simplified formula:

$$\kappa_H = \frac{\kappa_1 + \kappa_2}{2} \tag{3.14}$$

κ_H is known as the *mean normal curvature*.

The Laplace-Beltrami operator

In this work we will be making extensive use of the Laplace-Beltrami operator Δ_S. This operator is defined as the divergence of the gradient $\Delta_S = \text{div}\nabla$. For two-parameter vector-valued functions such as a parametric surface, the operator can also be seen as the sum of its second partial derivatives:

$$\Delta_S = \text{div}\nabla = f_{uu} + f_{vv} \tag{3.15}$$

Its most interesting property for the context of this project is its direct relationship to the Euler-Lagrange surface area minimization. This equation relates the mean normal curvature of a point P with its differential area [36]:

$$2\kappa_H \text{n(P)} = \lim_{\text{diam}(A)\to 0} \frac{\Delta_S A}{A} \tag{3.16}$$

where $diam(A)$ is the diameter around an infinitesimal area A around P. This last equation will become important in the next section to derive a discrete version of the Laplace-Beltrami operator.

Differential Area

Some point-defined differential properties of meshes are calculated in an infinitesimal area around the evaluated point. In the case of a manifold surface declared in a parametric form, one can approximate this infinitesimal area as the one of the tangent plane around the point. The tangent vectors $\mathbf{x_u}$ and $\mathbf{x_v}$ form a parallelogram whose area is projected onto the Cartesian plane using the Lagrange identity:

$$dA = \|\mathbf{X_u} \times \mathbf{X_v}\| = \sqrt{\mathbf{X_u^2 X_v^2} - (\mathbf{X_u X_v})^2} = \sqrt{\mathbf{EG} - \mathbf{F^2}} = \sqrt{\det(\mathbf{I})} \quad (3.17)$$

3.3 Differential properties of triangular meshes

The previous sections have described metrics and differential properties of parametric surfaces. However, these surfaces are not available for most real-world applications. In the case of range scans, the common procedure is to retrieve a cloud of points of the scanned object and connect them to their closest neighbors by edges performing a Delaunay triangulation. The result is a discrete mesh made of points and polygonal faces. We are particularly interested in triangular faces, for which an extensive body of literature has been produced.

To extend the previous metrics and properties to triangular meshes, it is assumed that they are piecewise-linear approximations of smooth surfaces. Differential properties are approximated as *spatial averages* around the vertices of the mesh and the approximations become more accurate with the sampling density (commonly called *mesh resolution*). These spatial averages are only contained in the 1-ring neighborhood of every vertex since it is the smallest region that lies around a vertex. Their calculations need a few assumptions about the mesh such as the absence of degenerate triangles and an average of 6 neighbors for every vertex.

Discrete Differential Area

Let us consider a vertex $\mathbf{x}$ and its 1-ring neigbors forming a fan of triangles (see Figure 3.18, left). The infinitesimal area around $\mathbf{x}$ can be approximated as a sum of the contributions of each of the triangles. For every triangle, we select a point interpolated between the three vertices of the face and form a patch around $\mathbf{x}$ and the interpolated point. The patches are connected and summed to obtain a finite version of the differential area. In this work, we will make use the Voronoi regions $\mathcal{A}_{\text{Voronoi}}$ that result from joining the circumcenter of a triangle to its adjacent edges. There exist three Voronoi cells for every triangle, yet we are only interested in the one that is connected to $\mathbf{x}$.

Given a non-obtuse triangle formed by its vertices P, Q and R and its circumcenter O, its Voronoi cell area $\mathcal{A}_{Voronoi}$ for P is calculated as follows. First, the circumcenter O is connected with each of the vertices to divide the

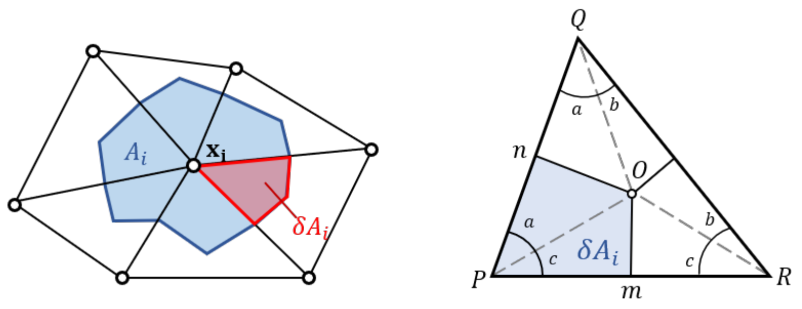


Figure 3.5: Left: Diferential area A_i around the point x_i is calculated as the sum of the individual Voronoi regions δA_i. Right: Voronoi region on a non-obtuse triangle. Images adapted from [8] and [36].

triangle by its bisectors, producing three pairs of adjacent angles a,b and c (Figure 3.18,right). By the properties of perpendicular bisectors, it follows that $a + b + c = \frac{\pi}{2}$ and therefore $a = \frac{\pi}{2} - \angle Q$ and $c = \frac{\pi}{2} - \angle R$ [36]. The Voronoi area δA_i is the sum of the area of triangles $\widehat{POm}$ and $\widehat{POn}$:

$$\begin{aligned} \delta A_i &= A_{\widehat{POm}} + A_{\widehat{POn}} \\ &= \frac{1}{4}\|PR\| \cdot (\frac{1}{2}\|PR\| \tan(\frac{\pi}{2} + \angle Q)) + \frac{1}{4}\|PQ\| \cdot (\frac{1}{2}\|PQ\| \tan(\frac{\pi}{2} + \angle R)) \\ &= \frac{1}{8}(\|PR\|^2 \cot \angle Q + \|PQ\|^2 \cot \angle R) \end{aligned} \tag{3.18}$$

Note that if the triangle is obtuse the circumcenter would lie outside of the triangle area. In this case, a rough estimation is used instead [23]. If the angle at P is obtuse, the area is approximated as $A_T/4$, where A_T is the area of the triangle given by

$$A_T = |PQ| \cdot |PR| \sin \angle(PR, PQ) \tag{3.19}$$

Otherwise, the area is approximated as $A_T/8$.

Finally, the differential area A_i around the vertex is calculated as the sum of each individual Voronoi region:

$$A_i = \sum_{j \in \mathcal{N}_1(i)} \delta A_j \tag{3.20}$$

We consider the the sum of Voronoi areas to be a discrete version of the differential area and thus we can relate it to the Euler-Lagrange area minimization of Equation 3.16.

Vertex Normal

In the absence of a continuous domain around vertices, the normal $\mathbf{n}(\mathbf{v})$ of the vertex v is calculated as a weighted average of the normals $\mathbf{n_T}$ of the incident triangles :

$$\mathbf{n}(\mathbf{v}) = \frac{\sum_{T\in\mathcal{N}_1(v)} \alpha_T \mathbf{n_T}}{\left\| \sum_{T\in\mathcal{N}_1(v)} \alpha_T \mathbf{n_T} \right\|} \tag{3.21}$$

the normals $\mathbf{n_T}$ are calculated by computing the normal direction of the plane containing the triangle $\triangle$ABC:

$$\mathbf{n_T} = ||AB| \times |AC|| \tag{3.22}$$

Note that the length of this vector is proportional to the triangle area. This is in fact one way of determining the weights α_T of Equation 3.21 since bigger triangles will contribute more to the direction of the normal. There exist more alternatives to weight the triangle contributions but for the purpose of this project we will simply normalize the vectors $\mathbf{n_T}$ and apply $\alpha_T = A_T / \sum_{T\in\mathcal{N}_1(v)} A(T)$. Equation 3.22 is employed on the distance-to-normal function of Section 4.5.1 and on the normal pruning criteria of Section 4.5.2.

Gradients

To obtain a discrete version of the Laplace-Beltrami operator, we must find first a suitable definition of gradient for piecewise linear functions such as our mesh. Indeed, we can consider our mesh as a discrete version of a vector-valued function such that for every vertex $\mathbf{x}$ it follows that $f(\mathbf{x}) = x$.

Suppose a triangular face on the mesh of vertices i, j and k and their respective function values f_i, f_j and f_k. The triangle can be parametrized by its barycentric coordinates $\mathbf{u} = (u, v, w)$ and by its unity of partition condition it follows that $u + v + w = 0$ and $\nabla u + \nabla v + \nabla w = 0$. We can access the value of the function at any point on the triangle by linearly interpolating between the three function values:

$$f(\mathbf{u}) = f_i u + f_j v + f_k w \tag{3.23}$$

The gradient of f is given by:

$$\nabla f(\mathbf{u}) = f_i \nabla u + f_j \nabla v + f_k \nabla w \tag{3.24}$$

which by the unity of partition property can be rearranged as:

$$\nabla f(\mathbf{u}) = (f_j - f_i)\nabla v + (f_k - f_i)\nabla w \tag{3.25}$$

The steepest ascent direction of ∇v and ∇w is the one perpendicular to the line that connects their opposing vertices (See Figure), therefore:

$$\nabla f(\mathbf{u}) = (f_j - f_i)\frac{(x_j - x_i)^\perp}{2A_T} + (f_k - f_i)\frac{(x_k - x_i)^\perp}{2A_T} \tag{3.26}$$

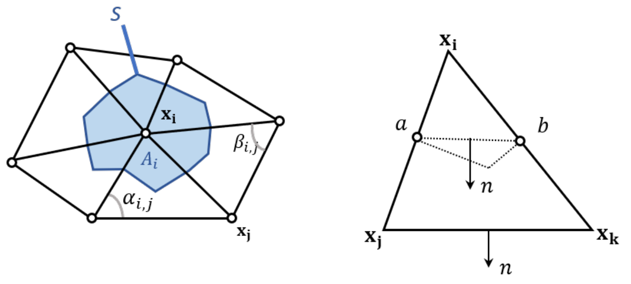


Figure 3.6: Angles and measures employed in the Laplace-Beltrami derivation.

Cotangent Discretization of Laplace-Beltrami operator

Meyer et al. [36] proposes an accurate discretization of the Laplace-Beltrami operator applied over a finite region A_i of a triangular piecewise linear function. First, employing the Gauss divergence theorem, the integral of the Laplace-Beltrami operator over the area A_i transforms into the integral over the boundary of the area s:

$$\begin{aligned}\int_{A_i} \Delta f(\mathbf{u}) dA &= \int_{A_i} \text{div} \nabla f(\mathbf{u}) dA \\ &= \int_s \nabla f(\mathbf{u}) n(\mathbf{u}) ds\end{aligned} \tag{3.27}$$

We can split the integral over all the adjacent triangles T to $\mathbf{x}$. The integral starts and ends at the midpoints a and b of the triangle T (see Figure 3.6, right). Since the normal of the line connecting a and b is the same as the line connecting the vertices opposite to $\mathbf{x}$ and the gradient $\nabla f(\mathbf{u})$ is constant along the line, the integral can be reformulated as:

$$\begin{aligned}\int_{s\cap T} \nabla f(\mathbf{u}) n(\mathbf{u}) &= \nabla f(\mathbf{u})(\mathbf{b}-\mathbf{a})^{\perp} \\ &= \frac{1}{2} \nabla f(\mathbf{u})(\mathbf{x}_k - \mathbf{x}_j)^{\perp}\end{aligned} \tag{3.28}$$

Note that because a and b are midpoints $\|a-b\| = \frac{1}{2}\left\|\mathbf{x_k} - \mathbf{x_j}\right\|$. Combining Equations 3.28 and 3.26 we obtain:

$$\begin{aligned}\int_{s\cap T} \nabla f(\mathbf{u}) n(\mathbf{u}) = \quad & (f_j - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^{\perp} \cdot (\mathbf{x}_k - \mathbf{x}_j)^{\perp}}{4A_T} \\ & + (f_k - f_i) \frac{(\mathbf{x}_k - \mathbf{x}_i)^{\perp} \cdot (\mathbf{x}_k - \mathbf{x}_j)^{\perp}}{4A_T}\end{aligned} \tag{3.29}$$

Let γ_j and γ_k be the inner angles at $\mathbf{x_j}$ and $\mathbf{x_k}$ respectively. The triangle area A_T is related to these angles by:

$$A_T = \frac{1}{2}\sin\gamma_j \|\mathbf{x_j}-\mathbf{x_k}\|\|\mathbf{x_j}-\mathbf{x_i}\| = \frac{1}{2}\sin\gamma_k \|\mathbf{x_k}-\mathbf{x_j}\|\|\mathbf{x_k}-\mathbf{x_i}\| \tag{3.30}$$

It also follows that:

$$\begin{aligned}\cos\gamma_j &= \frac{(\mathbf{x_j}-\mathbf{x_i})\cdot(\mathbf{x_j}-\mathbf{x_k})}{\|x_j-x_i\|\|x_j-x_k\|}\\ \cos\gamma_k &= \frac{(\mathbf{x_i}-\mathbf{x_k})\cdot(\mathbf{x_j}-\mathbf{x_k})}{\|x_i-x_k\|\|x_j-x_k\|}\end{aligned} \tag{3.31}$$

Plugging Equations 3.30 and 3.31 into 3.29 we obtain:

$$\int_{s\cap T} \nabla f(\mathbf{u})n(\mathbf{u}) = \frac{1}{2}(\cot\gamma_k(f_j-f_i)+\cot\gamma_j(f_k-f_i)) \tag{3.32}$$

By summing the contributions of all edges we obtain back the integral over the area A_i (Equation 3.27):

$$\iint_{A_i} \Delta f(\mathbf{u})dA = \frac{1}{2}\sum_{x_j\in\mathcal{N}_i}(\cot\alpha_{i,j}+\cot\beta_{i,j})(f_j-f_i) \tag{3.33}$$

we plug Equation 3.16 to finally obtain a relationship between the mean normal curvature and the discrete Laplace-Beltrami operator:

$$2\kappa_H\mathbf{n} = \frac{1}{2A_i}\sum_{x_j\in\mathcal{N}_i}(\cot\alpha_{i,j}+\cot\beta_{i,j})(f_j-f_i) \tag{3.34}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the angles opposite to x_j as indicated in Figure 3.6, right. The area A_i can be computed by summing all the mixed Voronoi areas of the adjacent faces to the vertex with Equation 3.18. This widely utilized version of the Laplace-Beltrami operator allows us to calculate the mean normal curvature at any point of the mesh as a sum of the Voronoi areas and angles of its neighboring faces. We employ Equation 3.34 for designing a smoothness prior that constraint the changes in local curvature of the original template (see Section 4.4 for a detailed explanation).

Chapter 4

Methodology

Introduction

This chapter contains a detailed description of our template fitting, non-rigid registration framework for registering a template mesh towards a depth scan. First, we introduce the concept of non-rigid registration and pairwise correspondences. Second, we explain a pre-alignment and alignment scheme to obtain a proper initialization for the non-rigid registration. Third, we employ the mathematical concepts of vertex normal, infinitesimal area and mean normal curvature obtained in Chapter 3 to design an energy minimization problem for morphing the template mesh to the scan under constrained deformations. Lastly, we discuss methods for obtaining pairwise correspondences and a scheme for discarding of lowering the importance of bad correspondences.

4.1 Method overview

In the context of this thesis, registration refers to the process of finding an alignment between two meshes through a transformation. Usually one of the meshes (called *target*) remains fixed and the other (called *source*) is adapted to match its shape. While rigid registration algorithms are limited to affine transformations, i.e, rotation and translation, non-rigid registration algorithms allow relative distances between points to be altered. In this work, we focus on a branch of registration algorithms called *pairwise iterative closest point algorithms* or ICP, where each point of the source mesh gets assigned with a *corresponding point* on the target mesh. Corresponding points belong to semantically or topologically similar parts between the two meshes and are typically assigned by a distance function. By bringing corresponding pairs together, the source mesh morphs iteratively towards the target until a certain error threshold is achieved.

The final purpose of our non-rigid registration method is to register a clean and smooth template mesh of a face towards a 3D scan mesh so the template

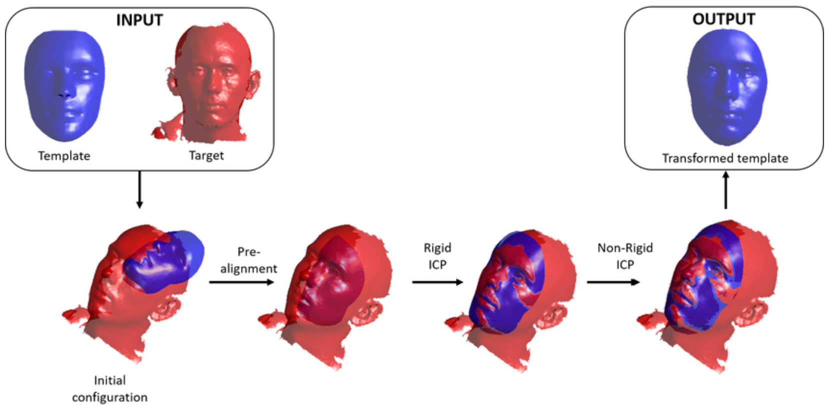


Figure 4.1: Proposed method pipeline. Initially, a target and template mesh begin in an arbitrary position and orientation. The template is aligned and iteratively adapted until it matches the target mesh geometry.

adopts the shape of the scan while preserving its *cleanness* (mesh triangles are regular and of similar size around a small local area) and *smoothness* (mesh resembles a continuous surface). This process is commonly called *template fitting*. For the remainder of this work, we will call the template *source mesh* $\mathcal{X}$ and the 3D scan *target mesh* $\mathcal{Y}$. There are no assumptions on the meshes other than $\mathcal{X}$ is a continuous surface embedded in $\mathbb{R}^3$ composed of a set of points $\mathrm{X} = \{\mathbf{x}_i \in \mathrm{X}, i = 1, ..., n\}$.

Our template fitting method consists of three distinctive phases (see Figure 4.1):

- A **pre-alignment phase** in which the source and target mesh are roughly oriented to avoid local minima on the rigid registration.
- A **rigid registration phase** in which the target and source get as close as possible without altering the relative distance between their points.
- A **non-rigid registration phase** in which the source mesh deforms consistently to match the target mesh while preserving its quality.

Lastly, we will discuss the distance functions used by our method to find correspondences and a scheme to discard or downweight bad correspondences.

4.2 Pre-alignment phase

Initially, the target and the source mesh begin in arbitrary orientations and locations. In this state, there is a risk of failure in the rigid registration phase. It has been shown thoroughly that ICP algorithms get stuck in local minima if the objects to register are not initialized properly. The purpose of the pre-alignment phase is then to initialize the source mesh in such a way that the rigid alignment succeeds.

To prealign the source mesh, we can leverage the fact that human faces have a characteristic shape that can be roughly approximated as the half of an ellipsoid. Its most prominent dimension would be the length of the face, followed by its width and depth. The reason why depth is the least prominent dimension is because ranged scans usually only cover the front of the face, leaving holes on the back of the head.

The three dimensions of the face can be interpreted as the directions of a coordinate system centered on the centroid of the mesh (see Figure 4.2). We can perform a rough alignment by making the coordinate systems of two face meshes coincide.

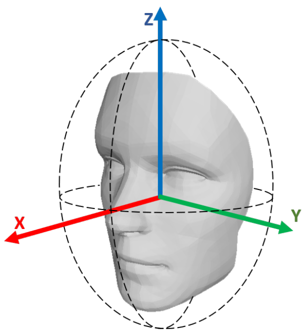


Figure 4.2: Imaginary face axes when approximated as an ellipsoid.

By treating the mesh vertices as a cloud of points, we can perform Principal Component Analysis to obtain the directions of maximum variance, i.e., the directions in which data is more spread. These directions are given by the eigenvectors of the covariance matrix of the $n \times 3$ vertex positions, where n is the number of vertices of the mesh. Since the covariance matrix is symmetric and of dimensions 3×3, it has 3 eigenvectors perpendicular to each other. We subtract these three eigenvectors ordered by their eigenvalues to obtain the three directions of maximum variance and form a right-hand basis centered on the centroid of the mesh. The process is applied on both the target mesh and the source mesh. Let M_x and M_y be the matrix containing the principal directions of the systems of $\mathcal{X}$ and $\mathcal{Y}$:

$$M_x = \begin{pmatrix} e_{1,x} & e_{1,y} & e_{1,z} \\ e_{2,x} & e_{2,y} & e_{2,z} \\ e_{3,x} & e_{3,y} & e_{3,z} \end{pmatrix}, \quad M_y = \begin{pmatrix} p_{1,x} & p_{1,y} & p_{1,z} \\ p_{2,x} & p_{2,y} & p_{2,z} \\ p_{3,x} & p_{3,y} & p_{3,z} \end{pmatrix}, \tag{4.1}$$

where $\|e\| = \|p\| = 1$. We want to find a rotation R and translation t that brings $\mathcal{X}$ in alignment with $\mathcal{Y}$ such that $M_x R + t = M_y$. The translation can be obtained by simply subtracting the centroid coordinates:

$$t = c_y - c_x \tag{4.2}$$

If we suppose that both frames are located at the point $(0, 0, 0)$, the rotation matrix R can be obtained by a simple inverse operation:

$$M_x R = M_y \rightarrow R = M_x^{-1} M_y \tag{4.3}$$

It should be noted that the principal directions of the mesh become skewed in the presence of holes or points that belong to the neck and shoulders. More sophisticated approaches for pre-alignment employ shape descriptors to find a transformation between a subset of salient points on the shape [30]. However, for the purpose of this thesis, this simple alignment scheme succeeds at bringing the source mesh in an appropriate position for the rigid alignment phase.

4.3 Rigid registration phase

The objective of the rigid registration phase is to find a rigid transformation composed of scaling, rotation and translation that brings the source mesh $\mathcal{X}$ as close as possible to $\mathcal{Y}$. The closeness is measured as the sum of Euclidean distances between corresponding points:

$$E_{\text{rigid}} = \sum_{i=1}^{n} \|y_i - \tilde{x}_i\|_2^2 \tag{4.4}$$

where $\tilde{x}$ is a rigidly transformed source point through a scaling factor s, a rotation matrix R and a translational offset t:

$$\tilde{x} = sR(x) + t \tag{4.5}$$

The correspondences between $\mathcal{X}$ and $\mathcal{Y}$ are obtained with the least Euclidean distance. The use of the Euclidean distance may cause the algorithm to get trapped in local minima but our results show that the pre-alignment phase of Section 4.2 returns a sufficiently good initial estimation as to obtain a good final result.

There are numerous ways of obtaining the transformation parameters. We opted to choose the quaternion optimization implementation based on the material of [24] because of its simplicity and speed. The algorithm returns the rigid transformation parameters and once applied to the source mesh vertices, we can assume that the target mesh and the source mesh are at their closest possible distance.

4.4 Non-Rigid registration phase

We formulate the pairwise non-rigid registration problem as the minimization of an energy expression. As it is common in literature, we divide the expression into two terms: a matching energy E_{match} and a prior energy E_{prior} [9, 12].

$$E_{\text{reg}} = E_{\text{match}} + E_{\text{prior}} \tag{4.6}$$

The matching energy E_{match} measures how close the source mesh is to the target mesh in terms of the sum of distances between corresponding points. The prior energy E_{prior} imposes constraints over the vertex displacements so as to produce an effect on the deformations of the source mesh. If the constraints are not respected, the prior energy rises. This creates a trade-off between moving the vertex towards its corresponding points on the target mesh and preserving the desired type of deformations.

Our formulation consists of one matching term and three prior terms:

$$E_{\text{reg}} = \underbrace{\omega_1 E_{\text{match}}}_{\text{matching term}} + \underbrace{\omega_2 E_{\text{global}} + \omega_3 E_{\text{local}} + \omega_4 E_{\text{smooth}}}_{\text{prior terms}} \tag{4.7}$$

The total registration energy E_{reg} is a weighted sum of the energy terms. The equation is solved iteratively producing a *transformed source* $\mathcal{Z}$ until convergence is found. Initially, the weights ω_i of the prior energy terms are high with respect to the matching energy term so as to preserve the topology of the template. When the algorithm reaches convergence, prior terms are downweighted to favor the matching energy. The desired effect is that the source mesh deforms slowly and conservatively until good correspondences can be found. When convergence has been reached a certain number of times (defined by the user), the algorithm stops and returns the resulting mesh. This weight scheduling is inspired by the methods proposed in [2, 52, 33].

Matching energy E_{match}

The matching energy term attempts to minimize the distance between corresponding points of $\mathcal{Z}$ and $\mathcal{Y}$.

$$E_{\text{match}} = \sum_{i=1}^{n} \left\| z_i^{t+1} + \mathcal{P}_{\mathcal{Y}}(z^t) \right\|_2^2 \tag{4.8}$$

The superindex t represents the iteration index and in $t = 0$ it follows that $\mathcal{Z} = \mathcal{X}$. The function $\mathcal{P}_{\mathcal{Y}}$ returns the closest point of $\mathcal{Y}$ according to a distance function (see Section 4.5.1). Since $\mathcal{P}_{\mathcal{Y}}$ is usually highly non-linear, it is preferable to use the previous iteration as estimate.

To speed up the convergence, it is possible to linearize $\left\| z_i^{t+1} + \mathcal{P}_{\mathcal{Y}}(z^t) \right\|_2$ at $\mathcal{P}_{\mathcal{Y}}(z^t)$ which gives the *point-to-plane* error $\mathbf{n_i}^\top (z_i^{y+1} + \mathcal{P}_{\mathcal{Y}}(z^t))$ [15]. $\mathbf{n_i}$ is the normal of the corresponding point of the surface $\mathcal{Y}$. The points on $\mathcal{Z}$ are then not moving directly towards their corresponding points on $\mathcal{Y}$ but to their projections on the tangent planes formed by the vertex normals of the target mesh (see Figure 4.3).

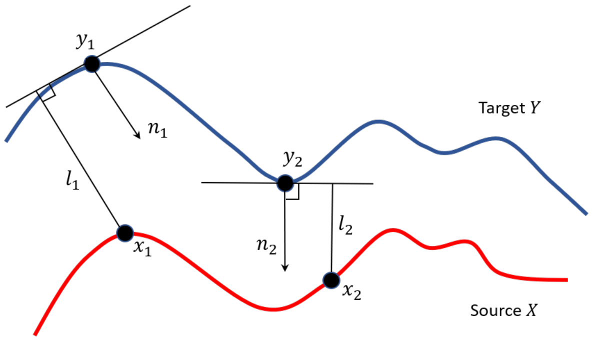


Figure 4.3: Point-to-plane distance visualization. The points of X attempt to minimize the length of projections on the tangent plane formed by the normals of Y. Image adapted from [34].

This linearization gave unexpected results which are discussed in the next chapter. Unless specified, we will use of the formula in Equation 4.8 for the matching energy instead.

Global Rigidity Prior E_{global}

The global rigidity prior forces the surface $\mathcal{Z}$ to follow a rigid transformation of its original shape $\mathcal{X}$:

$$E_{\text{global}}(\tilde{\mathbf{R}}, \tilde{\mathbf{t}}, \mathcal{Z}) = \sum_{i=1}^{n} \left\| z_i^{t+1} - \tilde{\mathbf{R}}(\mathbf{R}\mathbf{x_i} + \mathbf{t}) + \tilde{\mathbf{t}} \right\|_2^2 \tag{4.9}$$

R and **t** are the rotation matrix and translation vector obtained in Section 4.3. The expression attempts to find a position for every vertex z_i that resembles a rigid transformation of the original source mesh (see Figure 4.4).

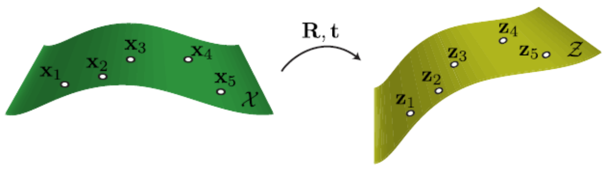


Figure 4.4: Global rigidity term visualization. Image retrieved from [9].

To prevent the non-linearity of the trigonometry functions we can linearize the rotation matrix $\tilde{\mathbf{R}}$ by the small angle approximation from the Taylor's expansion $\cos(\theta) = \theta, \sin(\theta) = 1$ [42]:

$$\tilde{\mathbf{R}} \approx \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \tag{4.10}$$

Since the approximation assumes that α, β and γ are small, we group all the unknown angles in the vector $\lambda = \begin{bmatrix}\alpha & \beta & \gamma\end{bmatrix}^\top$ and add a L_2 regularization term [38] to the energy with an associated weight ω_t:

$$E_{\text{global}}(\tilde{\mathbf{R}}, \tilde{\mathbf{t}}, \mathcal{Z}) = \sum_{i=1}^{n} \left\| z_i^{t+1} - \tilde{\mathbf{R}}(\mathbf{R}\mathbf{x_i} + \mathbf{t}) + \tilde{\mathbf{t}} \right\|_2^2 + \omega_t \|\lambda\|_2^2 \tag{4.11}$$

In our experiments, we fixed a L_2 norm weight of $\omega_l = 100$. In preliminar experiments, this weight value produced rotational coefficients $\alpha, \beta, \gamma \leq 0.01$ and translational vector changes of $t < 1$ (in mm in the D3DFACS database). We consider that these values do not break the small displacement assumptions.

Local Rigidity Prior E_{local}

The local rigidity prior reinforces the vertices to follow an averaged transformation of its 1-ring neighbors:

$$E_{\text{local}} = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(j)} \left\| (z_j^{t+1} - z_j^t) - (z_i^{t+1} - z_i^t) \right\| \tag{4.12}$$

The objective of this term is two-fold. First, it emulates an elastic force that punishes vertices that separate from each other or get too close to each other. This preserves the regularity of the mesh and prevents vertices from sticking to corresponding points and cause bumps and noisy patches. Secondly, it yields a plausible deformation for points which have not found a good correspondence as they will follow an average of the transformation of their neighbors. Both effects can be visualized in Figure 4.5.

It should be noted that the local rigidity term constraints all deformations. This is an undesirable effect because vertices should be able to follow good correspondences to adopt the topology of the target mesh. Otherwise, we would only obtain a slightly deformed version of the template mesh. This problem is addressed in Section 4.5.3, where the distance d of a vertex to its corresponding point influences negatively the weight ω_1 and positively the weight ω_3 of Equation 4.7.

Smoothness Prior E_{smooth}

The smoothness prior forces the deformation on $\mathcal{Z}$ to preserve the local curvature of the original mesh $\mathcal{X}$ [52]. This term is defined as the l-2 norm of the Laplace-Beltrami operator of the deformation vector $d_i = z^{t+1} - x_i$:

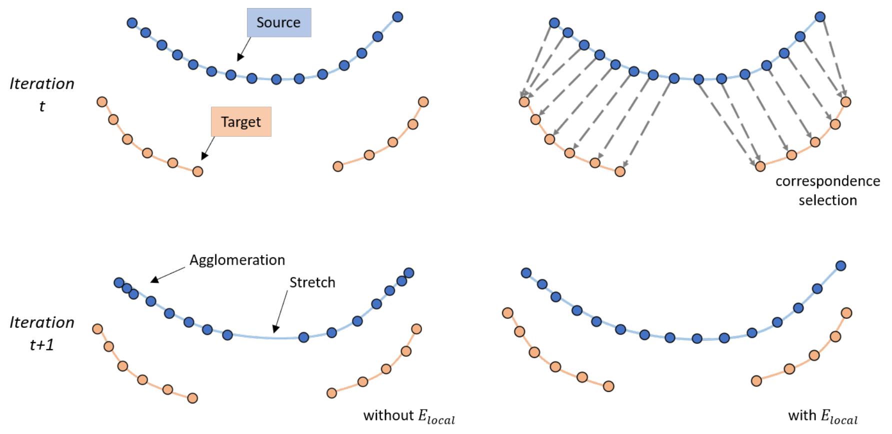


Figure 4.5: Visualization of the local rigidity term effect. *Upper left*: Initial situation of a target mesh and a source mesh. *Upper right*: correspondence selection. *Bottom left*: without the local rigidity term E_{local}, vertices move freely towards their correspondences causing holes and agglomerations. *Bottom right*: with the local rigidity term, vertices are forced to move similarly to their neighbours yielding a more plausible deformation.

$$E_{\text{smooth}} = \sum_{i=1}^{n} \|\Delta_{\mathcal{S}} d_i\|_2^2 = \sum_{i=1}^{n} \left\|\Delta_{\mathcal{S}} z_i^{t+1}\right\|_2^2 - \sum_{i=1}^{n} \|\Delta_{\mathcal{S}} x_i\|_2^2 \tag{4.13}$$

Following Equation 3.33 this term can be rewritten as

$$\begin{aligned} &\sum_{i=1}^{n} \left\| \frac{1}{\tilde{A}_i} \sum_{j \in \mathcal{N}(j)} (\cot \alpha_{ij} + \cot \beta_{ij}))(z_j^{t+1} - z_i^{t+1}) \right\|_2^2 \\ &- \sum_{i=1}^{n} \left\| \frac{1}{A_i} \sum_{j \in \mathcal{N}(j)} (\cot \alpha_{ij} + \cot \beta_{ij}))(x_j - x_i) \right\|_2^2 \end{aligned} \tag{4.14}$$

Notice that the tilded variables $\tilde{A}_i, \tilde{\alpha_{ij}}$ and $\tilde{\beta}_{ij}$ are dependent on the unknown variables $\mathbf{Z}$. The presence of roots and trigonometric functions would severely complicate the optimization of this formula. For this reason, we opted to fix the cotangent weights and Voronoi areas as the ones in the original mesh $\mathcal{X}$. By fixing these parameters, the formula becomes:

$$\sum_{i=1}^{n} \left\| \frac{1}{A_i} \sum_{j \in \mathcal{N}(j)} (\cot \alpha_{ij} + \cot \beta_{ij}))((z_j^{t+1} - z_i^{t+1}) - (x_j - x_i)) \right\|_2^2 \tag{4.15}$$

Note that we punish the deformations of points with high local curvature more harshly. Looking at Equation 3.33, we can interpret the smoothness term as a curvature regularizer. This is because the error will increase with either changes in the magnitude of curvature or in the direction of the normal.

Optimization Expressions 4.8, 4.12 and 4.15 form a set of quadratic equations and thus they can be solved by setting their partial derivatives to 0 and solving the linear system. The derivation of most terms is straightforward, except for the smoothness prior energy. We included the derivations of its expression in Appendix B. For our experiments, we employ the standard Matlab linear solver.

4.5 Correspondences

Reliable correspondence establishment is critical for achieving meshes of good quality and in good correspondence. In this section, we will discuss three distance functions for selecting pairwise correspondences and a scheme for pruning or downweighting correspondences that possibly lead to visible residual errors on the resulting mesh.

4.5.1 Distance functions

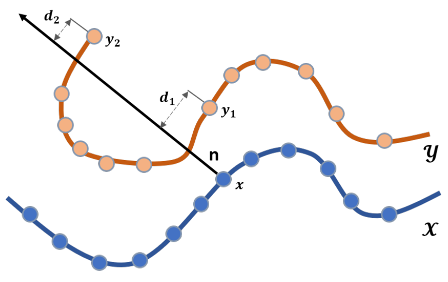
Every source vertex gets assigned to the target vertex with the least distance. The distance function is therefore critical to ensure that we are selecting pairs that belong to the same parts of the same features. In this work, we will study three distance functions:

Euclidean distance The Euclidean distance is the optimal way of searching for correspondences when the distribution of the points is Gaussian [9]. This is not usually the case on scanned objects because they present regions with uneven densities of points and holes on missing viewpoint areas.

Distance-to-normal direction The distance-to-normal direction measures the distance between the line formed by a source vertex normal (Equation 3.21) and the target vertices. Since the normal line prolongs indefinitely, it could pass near vertices that are far from the source point. To prevent these bad correspondences from being assigned, we draw a spherical region around the source vertex and consider only the target vertices that lie inside of it (see figure below).

Given a source point x, its unit normal n and a set of candidates $\mathcal{C}$ inside the bounding sphere, the corresponding point $c_x \in \mathcal{C}$ is the one that minimizes the distance to the line formed by $n - x$:

$$c_x = \underset{c_i \in \mathcal{C}}{\arg\min} \left\| (c_i - x) \times (c_i - (n - x)) \right\| \tag{4.16}$$



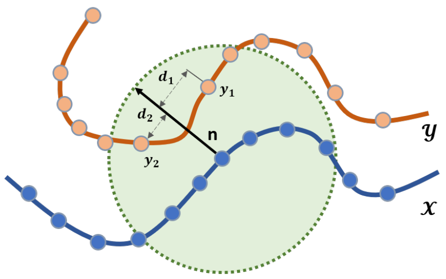


Figure 4.6: Distance-to-normal visualization. *Left:* A source point finds a target point y_2 with a small distance d_2 even if intuitively y_1 seems a better correspondence. *Right:* A bounding sphere prevents correspondences far from the source vertex to be assigned.

Laplace-Beltrami distance The normal mean curvature is defined as a vector with the direction of the vertex normal and a length proportional to its local curvature. Therefore, it can be considered a locally defined shape descriptor. We can establish a distance measure d_{LB} between two normal mean curvature vectors n_T and n_S with their Euclidean distance:

$$d_{\mathrm{LB}} = \|n_T - n_S\|_2^2 \tag{4.17}$$

It is very likely that in a high-density mesh many vertices from different regions have a similar mean normal curvature. Thus, when calculating the distance d_{LB}, we will only consider the subset of target vertices that lie on the bounding sphere described in the previous paragraph.

4.5.2 Trimming correspondences

In the context of our work, trimming a correspondence means setting the matching weight ω_1 to 0 while leaving the rest of the prior weights unaltered. This way, if a point gets assigned a bad correspondence, it will ignore it and attempt to preserve the original template shape.

Directly discarding correspondences is a very common way of removing unreliable matches [9]. Several authors [33, 9, 2, 43] have proposed different criteria to prune bad correspondences. In preliminary tests, we experimented with the trimming schemes proposed by these authors. We picked three criteria that yielded good results at discarding bad correspondences and were easy to implement:

- Remove correspondences of boundary points.
- Remove correspondences points that are further than a threshold distance d_t. This threshold distance can be measured with any of the distance functions of Section 4.5.1.

- Remove correspondences whose angle between normals exceeds a threshold α_t.

The last two methods introduce two parameters d_t and α_t that are difficult to optimize. For α_t, a fixed threshold angle of 60 degrees is chosen. The value of d_t is automatically adjusted at every iteration with the fourth spread extreme outlier criteria [17]. First, the N Euclidean distances between corresponding points are arranged from lowest to highest. The fourth spread f_s is the median of the $N/2$ lowest distances. Distances that are farther than $3 \cdot f_s$ are considered *extreme outliers* (see Figure 4.7). In our work, we set $d_t = 3 \cdot f_s$. We also set d_t as the bounding sphere radius of Section 4.5.1.

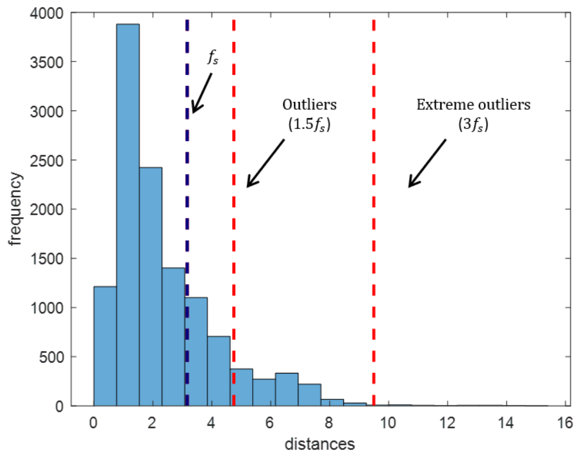


Figure 4.7: Histogram of Euclidean distances between source and target mesh. The value $3f_s$ is used as the cutoff distance d_t to prune outliers and as the radius of the bounding sphere of section 4.5.1.

4.5.3 Downweighting correspondences

Apart from directly pruning correspondences, we can penalize bad matches by downweighting their matching energy weight ω_1 of Equation 4.7 according to a penalty function $\phi(p)$. The penalty function should increase with the match distance so as to prevent points from stretching by moving towards vertices near holes.

However, we do not want points with bad correspondences to just become static. Ideally, we would prefer that they present a plausible deformation. Therefore, the penalty function $\phi(p)$ also increases the weight of ω_3. That is, points with unreliable matches follow an average of the deformation of their neighbours.

The function $\phi(p)$ chosen is a slightly varied version of the l-p norm as in [10]:

$$\phi(p) = 1 - \|d\|^p , \quad 0 < p < 1 \tag{4.18}$$

We can normalize the set of distances obtained in the correspondence search and use every distance as an input to the function. The value ω_{lp} returned by the function will be used as a downweighting factor to ω_1, e.g., $\tilde{\omega_1} = \omega_{lp} \cdot \omega_1$. Likewise, the weight ω_4 will be updated as $\tilde{\omega_4} = (1 - \omega_{lp}) \cdot \omega_4$.

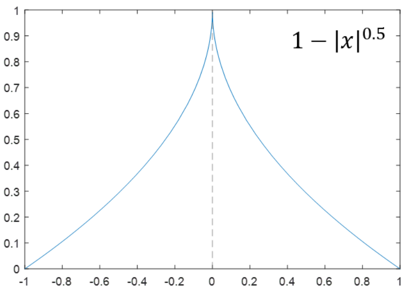


Figure 4.8: Function l-p when $p = 0.5$

Equation 4.18 decays its value quickly from $x = 0$ to $x = 1$, strongly penalizing large distances (see Figure 4.8. In this work, we opted to fix $p = 0.5$ as if it remained as a variable the Equation 4.7 would stop being quadratic.

4.6 Summary

Algorithm 1 summarizes the non-rigid registration process after the pre-alignment and the rigid alignment has been performed. As inputs, we have the source and target meshes described by their vertices and faces and the set of scheduled weights described in Section 5.2.2. First, the Laplace-Beltrami matrices (Equations 3.19 and 3.33) and derivative coefficients of the original source mesh (Equation 3.15 of Appendix A) are computed. After the correspondences are found between the Target mesh and the Source mesh, they can be plugged in the derivative of Equation 4.8. The rest of the derivatives only need the source vertices. Once the linear system is solved, the source vertices are updated and plugged onto the total error formula (Equation 4.7). If the error changes less than a user-defined threshold ϵ, the process is repeated for the new scheduled weights.

```
input  : Source vertices and faces $sV$, $sF$;
         Target vertices and faces $tV$, $tF$;
         Energy weights $\omega$ (5×4 matrix for each scheduled weight set)
output: Transformed Source vertices $new\_sV$

$new\_sV \leftarrow sV$;
$\epsilon \leftarrow 1e-6$ (User-defined error threshold);
$\mathcal{A}, \mathcal{M} \leftarrow$ getLaplaceBeltramiMatrices($sV$,$sF$) (See Appendix A);
$L \leftarrow \mathcal{M}^{-1}\mathcal{A}$;
$L_d \leftarrow 2 \cdot (L)^\top (L)$
foreach $\omega_i \in \omega$ do
    $new\_error = 0$ ;
    $old\_error =$ Inf.;
    while $(\|old\_error - new\_error\|) \geq \epsilon$ do
        $U, \omega^* \leftarrow$ getTargetCorrespondences($new\_sV, sF, tV, tF$);
        /* $U$ is a vector with indexes of corresponding points
           */
        [A,b] $\leftarrow$ getDerivativeCoeffMatrix($new\_sV, tV, U, L_d, \omega^*$);
        $new\_sV \leftarrow A^{-1}b$ ;
        $old\_error = new\_error$;
        $new\_error =$ errorFunction($new\_sV, tV, U, \omega^*$)
    end
end
```

Algorithm 1: Non-rigid registration algorithm pseudocode.

Our correspondence procedure is described in Algorithm 2. The bounding sphere radius r is obtained with the fourth spread outlier threshold of the Euclidean distances. For every vertex of the source mesh v_i, we collect the target points that lie inside the sphere centered at v_i. We check that the candidate points do not lie on a boundary and that have compatible normals. If they fulfill these two requirements, their distance to the source point is stored. The candidate with the lowest distance is selected and the weights are updated according to Equation 4.18.

```
input  : Source vertices and faces $sV$, $sF$;
         Target vertices and faces $tV$, $tF$;
         Energy weights $\omega$
output: Correspondence indexes $U$ ;
         Updated weights $\omega^*$
$U \leftarrow$ knnSearch($sV$,$tV$);
$d_{knn} \leftarrow \left\| sV - tV(U) \right\|_2^2$ (Euclidean distances between correspondences);
$r \leftarrow$ calculateFourthSpreadThreshold($d_{knn}$);
foreach $v_i \in sV$ do
    $C =$ vertices of $tV$ inside a sphere of radius $r$ centered in $v_i$ ;
    $d_{min} = r$;
    foreach $c_j \in C$ do
        if (isBoundary($c_j$) $\|$ angleBetweenNormals($c_j, v_i$) $\geq 60$) then
            $C \leftarrow C \backslash c_j$. (Candidate $c_j$ is eliminated from $C$)
        else
            d = calcDistance($c_j, v_i$) (Section 4.5.1) ;
            if $d < d_{min}$ then
                $d_{min} = d$;
                $U(i) = j$;
            end
        end
    end
    $\omega_1^* = \omega_1 (1 - \left\| \frac{d_{min}}{r} \right\|^{0.5})$ (Equation 4.18);
    $\omega_3^* = \omega_3 \cdot \left\| \frac{d_min}{r} \right\|^{0.5}$;
end
```

Algorithm 2: Correspondence matching pseudocode.

Chapter 5

Results

Overview

This chapter describes the results of our framework when we register a sample template mesh towards the depth scans of the D3DFACS dataset and the HeadSpace dataset. First, we will analyze the influence of each term on the final result to select an appropriate set of weights. Then, we will propose an extra step for adapting the template to the user geometry before the non-rigid registration. Lastly, we will test our work against the datasets to evaluate how well the user identity is captured, its robustness against holes and changes in resolution and its per-vertex distance between source and target points. The obtained results are compared to the generic non-rigid registration algorithm of Amberg et al. [2] and the contemporary framework of Dai et al. [20].

We chose to compare our results with Amberg et al. because our approach and theirs are based on the minimization of an energy equation with local constraints and Iterative Closest Point elements. The primary difference between their and our energy equation is the inclusion of smoothness and global rigidity constraints as well as a different formulation of local rigidity constraints.

There are several reasons for comparing our work with the non-rigid registration framework of Dai et al. First, they employ an adaptative template step by bringing corresponding landmarks together on the source and target mesh. We added a similar template adaptation step on Section 5.3. Secondly, one of the objectives of their research was to produce meshes in full correspondence for the HeadSpace database, thus making the comparison between their and our framework straightforward. Lastly, this work is one of the most recent publications on non-rigid registration of depth scans and we consider it to be a good representation of the state-of-the-art of the facial reconstruction field.

In their publication, Dai et al. compare their performance on the Headspace database with two state-of-the-art morphable models: the Open Framework (OF)[35] and the Large Scale Facial Model [7]. Open Framework is a morphable model constructed from variations of a single template synthesized through Gaussian processing. Large Scale Facial is a standard morphable model constructed from a large pool of $\approx$ 9000 subjects. These statistical models are fitted towards Headspace subjects and their vertex error is compared with Dai's framework. We leverage from the performance figures obtained by Dai et al. to establish a rough comparison of these models to our framework. Since our non-rigid registration is based on a tailored pairwise correspondence, we expect to achieve better results than these two other methods.

5.1 Scan preprocessing

The 3D scans were preprocessed with the software MeshLab [17]. The following operations were performed on the scans before they were registered:

- Removed duplicate vertices.
- Removed unreferenced vertices.
- Removed duplicate faces.
- Removed non-manifold edges.
- Flipped T-edges on degenerate faces.

These standard filters are aimed to avoid computational errors as well as to preserve the two-manifold surface assumption needed to compute the Laplace-Beltrami operator.

5.2 Energy Weights influence

In this section, we will discuss the effects of the weights of the energy terms on the resulting mesh. For discussion purposes, we display the complete formulation of our energy equation:

$$\begin{aligned} E_{\text{total}} = \\ \omega_1 \sum_{i=1}^{n} \left\| z_i^{t+1} + \mathcal{P}_{\mathcal{Y}}(z^t) \right\|_2^2 \\ + \omega_2 \sum_{i=1}^{n} \left\| z_i^{t+1} - \tilde{\mathbf{R}}(\mathbf{R}\mathbf{x_i} + \mathbf{t}) + \tilde{\mathbf{t}} \right\|_2^2 + \omega_t \|\lambda\|_2^2 \\ + \omega_3 \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(j)} \left\| (z_j^{t+1} - z_j^t) - (z_i^{t+1} - z_i^t) \right\| \\ + \omega_4 \sum_{i=1}^{n} \left\| \Delta_{\mathcal{S}} z_i^{t+1} \right\|_2^2 - \sum_{i=1}^{n} \|\Delta_{\mathcal{S}} x_i\|_2^2 \end{aligned} \tag{5.1}$$

5.2.1 Error Magnitude

To set appropriate weights, we must analyze the magnitude of each energy term of Equation 4.7. For that, we will identify the case at which a vertex yields the maximum error according to each term. The magnitude of the term is bounded between 0 and its maximum value multiplied by the number of vertices n (in the very unlikely case that the worst case happens n times).

Note that all terms are expressed as Euclidean distances. Let us suppose that the rigid alignment phase already moved the source mesh and the target mesh close together and that they are well aligned (i.e., not trapped in local minima). The maximum error that E_{match} can have is at the point with the maximum distance d_{MAX} to its corresponding point on the target mesh. The maximum error that E_{global} can have (assuming that $R \approx I$) is also d_{MAX} because it is the maximum distance that a point has to deviate from its original position on the source mesh. Therefore, we conclude that E_{match} and E_{global} must have the same order of magnitude.

The term E_{local} is the summed difference of relative translations between neighbouring vertices (see Equation 4.12). The maximum possible error occurs when a vertex moves in the opposite direction with respect to all its neighbours. The maximum that a vertex can move from its original position is d_{MAX}. Considering that there can be more than one neighbour moving d_{MAX} on the iteration and that the average vertex has 6 neighbours, the local rigidity error will be at most $6 \times 2 \times d_{\text{MAX}}$. We can then conclude that E_{local} is at most one order of magnitude over E_{match} and E_{global}. However, our experiments show that if the target and source mesh are well aligned and have similar resolutions, E_{local} will stay within the same order of magnitude of E_{match} and E_{global}.

Lastly, the term E_{smooth} cannot be easily related to the other terms because it measures distances between normal vectors instead of corresponding pairs. Additionally, its value cannot be well estimated because it is a sum of cotangent weights that depend heavily on the initial configuration of the mesh. Our experiments show that for the D3DFACS dataset and the template employed E_{smooth} is approximately one third of E_{match} and E_{global}.

5.2.2 Weight setting

In this section, we will assess the effect of different weight combinations on the resulting mesh. We established a pairwise comparison of all weights under different values in Appendix A. Specifically, we executed the algorithm for one iteration, we fixed two weights to $\omega = 10$, varied the other two and arranged the results in a table. We will refer to these tables for the following observations:

Matching weight ω_1 Initially, we set $\omega_1 = 100$, ten times the regularization terms. As a result, the algorithm prioritizes minimizing the distance between correspondences over preserving the original topology of the source. Since source vertices can share the same corresponding point on the target mesh, we obtain collapsed vertices, self-intersections and a noisy patches (Figure A, middle columns). This effect is only reduced when $\omega_1 < 50$, the point at which the matching error becomes smaller than the regularization terms. Finally, at $\omega_1 = 0$ the algorithm no longer follows correspondences and the source mesh is not altered. If ω_3 is sufficiently large with respect to ω_1 (green cells of Figure A), the small movements produced by the global rigidity term cause vertices to average their positions with their neighbours, producing a characteristic surface flattening.

Global rigidity weight ω_2 When we set $\omega_2 = 100$, the algorithm prioritizes following a rigid transformation of the original source mesh over minimizing the distance between correspondences (Figure A, bottom rows). As a result, the source mesh does not noticeably deform. When we set $\omega_2 < 50$, the source mesh starts following correspondences. If both ω_1 and ω_2 are low, the local rigidity error becomes dominant and causes the mesh to average the displacements towards correspondences.

Local rigidity weight ω_3 The local rigidity weight have strong effects on the mesh deformation due to its error magnitude. When $\omega_3 > 50$, the algorithm prioritizes that vertices follow the transformation of their neighbours. As a result, vertices tend to form a flat surface and the template original topology is broken. This effect is diminished when ω_2 is also high, since vertices are also constrained to follow a rigid registration of the template. At low levels $\omega_3 = 10$, $\omega_3 = 5$, the local rigidity preserves the regularity on a small scale (nostrils, lip corners, eye corners) without altering the template topology.

Smoothness weight ω_4 The smoothness term has a similar effect to the local rigidity term. When the local rigidity weight ω_3 is set to 0, ω_4 preserves the topology of the template without causing artifacts even at very high values (Figure A).

Considering these observations, for the rest of our experiments we initially set $\omega_1 = 10$, $\omega_2 = 50$, $\omega_3 = 20$ and $\omega_4 = 50$. We execute 5 iterations in which we keep the matching weight ω_1 constant. The rest of the weights linearly decrease to $\omega_2 = 10$, $\omega_3 = 5$ and $\omega_4 = 10$. The local rigidity weight ω_2 is always smaller than the global rigidity weight ω_2 to prevent the observed vertex averaging effect. During the first iterations, the regularization terms are dominant. The source mesh is deformed conservatively, preserving its topology while roughly capturing the user identity. As the regularization weights decay, the matching energy error becomes more important and source mesh starts prioritizing following correspondences. The main purpose of the weight scheduling is to find good correspondences on the final iteration. Increasing the number of iterations from 5 does not improve the results significantly.

Note that the weight values were selected upon a mix of objective and subjective criteria. Weight combinations that presented self-intersections and collapsed vertices were discarded as well as combinations that were considered to

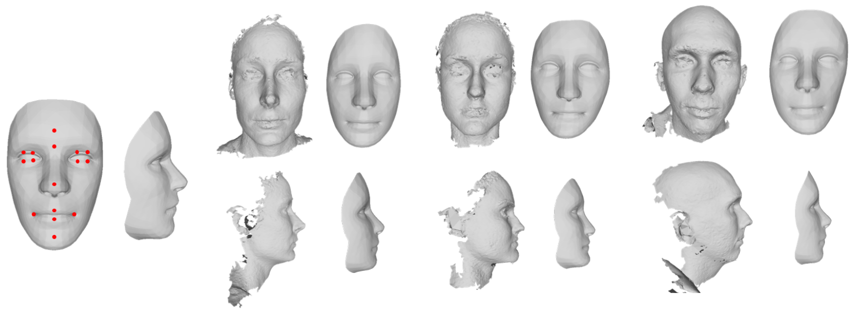


Figure 5.1: Template meshes transformed with the adaptative template step of Section 5.3. Left: Original template with 14 landmark locations. Right: the transformed templates match better the geometry of the user and give a stronger prior for the non-rigid registration algorithm.

not capture the user identity well. It is common in literature ([20, 33]) to aim to minimize the per-vertex nearest point error. However, this is not a good quality measure on scans that present noisy patches since the optimal solution would be to copy the scan noise vertex-to-vertex. As a conclusion, it is up to the user to decide what is an acceptable result.

5.3 Adaptative template

In our preliminary experiments, we used the same template as our source mesh. However, human faces present a wide spatial distribution in its constituent parts [20]. Our algorithm showed limitations when the template differed too much from the scan. The problem resides in that our correspondence matching uses a point-to-point and point-to-plane distance, and therefore, it assumes proximity of semantically similar regions (e.g., the left eye on the source mesh is closer to the left eye on the target mesh than to the nose). If, for example, the eyes of the scan and the template are placed at different heights, the source mesh may look visually similar to the target but the correspondences will not be correct.

We propose an extra step for roughly adapting the proportions of the source mesh to the target mesh. This step requires corresponding landmarks on the source mesh and the target mesh. The landmarks can be introduced manually or can be generated through shape descriptor matching such as spin images [30] or spherical harmonics [31]. In the particular case of textured meshes, one can also apply a landmark detector on the texture image and trace back its corresponding point [56]. Ideally, the landmarks should be placed at regions that are considered important in capturing the identity of the user such as lips, nose, eyes, forehead and chin.

We pose the problem of adapting the source mesh to the target mesh as an energy minimization equation of two terms:

$$E_{\text{template}} = E_{\text{land}} + \omega E_{\text{smooth}} \tag{5.2}$$

Equation 5.2 has a matching and a regularizer term. E_{land} measures the distance between the m landmarks:

$$E_{\text{land}} = \sum_{i=1}^{m} \left\| l_{\mathcal{Y},i} - l_{\mathcal{X},i} \right\|_2^2 \tag{5.3}$$

Where $l_{\mathcal{Y},i}$ and $l_{\mathcal{X},i}$ are corresponding landmarks on $\mathcal{Y}$ and $\mathcal{X}$. The regularization term E_{smooth} is exactly the same as in Equation 4.13. Its purpose is to smooth the deformations caused by the displacements of the landmarks. The factor ω weights the importance between the two terms (in our experiments $\omega = 100$). Equations 5.3 and 4.13 form a quadratic system whose global minimum can be computed by setting its partial derivatives to 0. Since the source mesh vertices are not updated, the equation is only solved once.

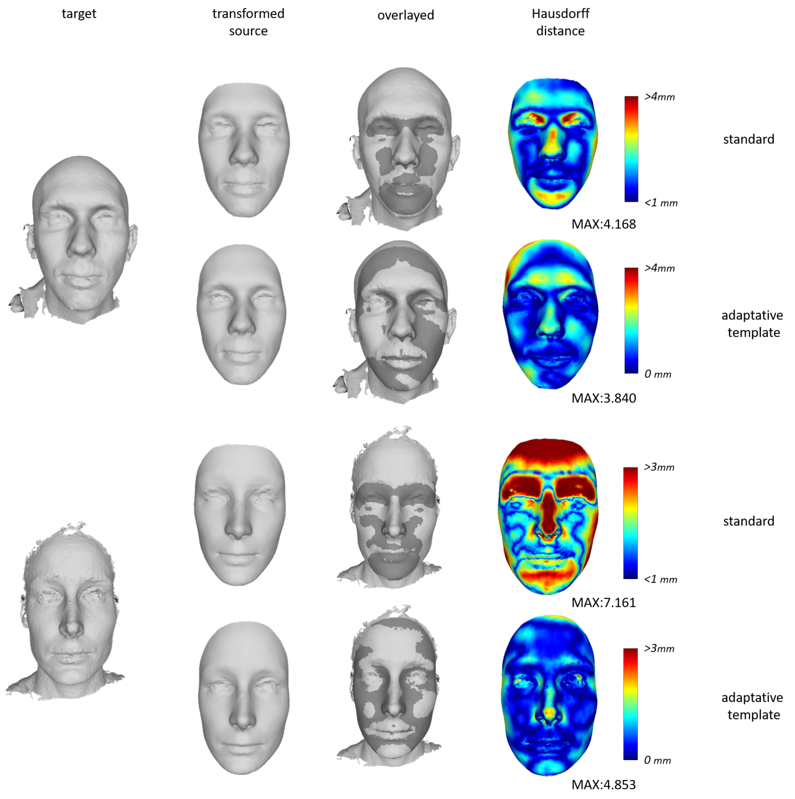


Figure 5.2: Non-rigid registration results with and without an adaptative template prior. The heatmaps present the minimum vertex-to-vertex distance.

Note that in this non-rigid registration formulation only the landmarks of the source mesh are pulled towards the target mesh. The rest of the vertices just preserve the smoothness of the template by following an elastic-like deformation. Therefore, it is critical that this step is performed after the rigid alignment, when the Target mesh and the Source mesh are close to each other.

Some examples of adapted templates can be seen in Figure 5.1, where 14 landmark locations were manually placed between the original template and the scans. Even if the user identity is not captured, the proportions of the deformed template match the user geometry better than the original template. A comparison of the non-rigid registration output with and without an adapted template is shown in Figure 5.2. Generally, when the adaptative template is employed, the error is reduced with respect to the original template on the areas around the landmarks. The output mesh also presents a more detailed representation of the user identity, specially at the eyes, lips and nose.

5.4 Correspondences

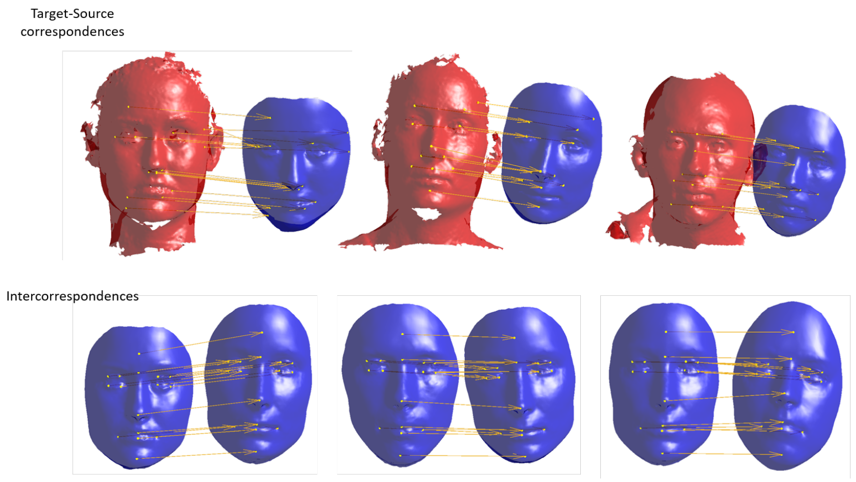


Figure 5.3: Target-source correspodences and intercorrespondences between deformed templates of different users. The arrows connect corresponding points. Our method shows good final correspondences between target and source meshes and between deformed templates.

A sample result of the same mesh registered with the different distances can be seen in Figure 5.4. The Euclidean distance favors points that are close even if they do not belong to the same features. This is specially noticeable on the nose close-up (third column of Figure 4.5.1): the source mesh stretches the tip of its nose towards the lateral cartilages simply because it is closer than the target's nose tip. The Laplace-Beltrami distance produces a jarred mesh with numerous self intersections, meaning that the mean normal curvature vector is not a good descriptor for pairwise correspondence matching. The normal distance function achieved a more natural deformation, closely capturing the shape of the nose, lips and eyes. For the rest of this work, we employ this the normal distance as our distance function $\mathcal{P}$ (see Equation 4.8).

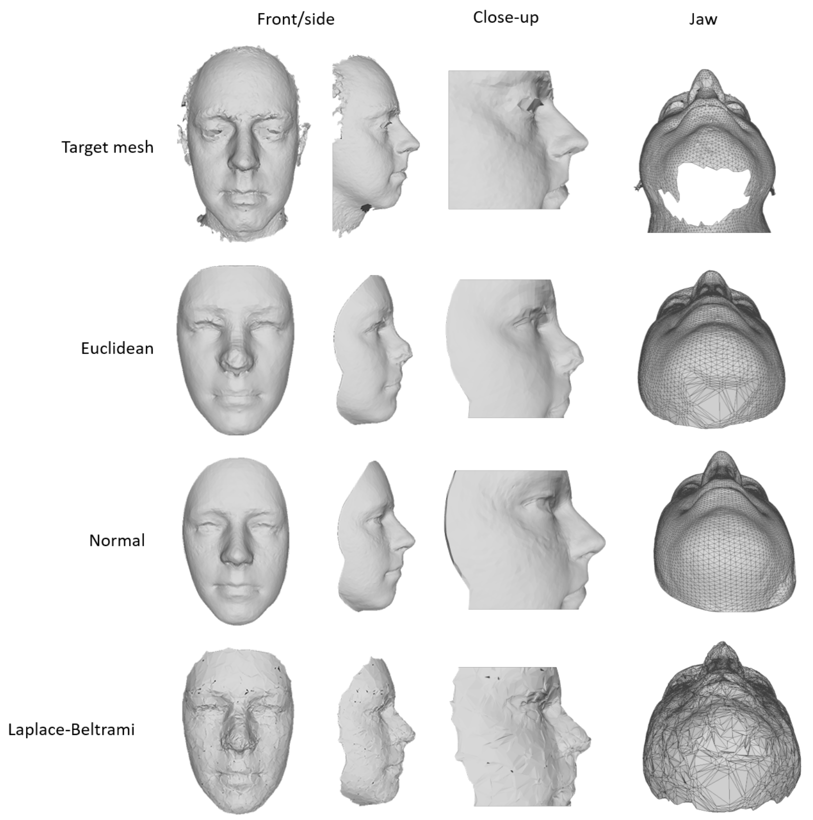


Figure 5.4: Comparison of Euclidean, normal and Laplace-Beltrami distance functions. The normal distance clearly produces superior results at adapting to the user geometry and dealing with large holes.

Apart from the visually appealing results of the normal distance, we are also interested in evaluating *target-source* correspondences and *intercorrespondences*. Target-source correspondences refer to how well are the scan points identified with the template points. Intercorrespondences refer to correspondences between points of morphed templates. Our method seems to perform well at both aspects as sampled feature points are located at the intuitive vertices (see Figure 5.3).

5.5 Resolution effect

We are interested in evaluating the robustness of our framework at morphing our template towards meshes of different resolutions. We extracted samples from the D3DFACS database and applied the Quadratic Edge Collapse Decimation algorithm (see [4] for a detailed explanation) implemented in Blender [39] to reduce the mesh resolution at 75%, 50% and 25%. We define here resolution as the ratio between the number of faces of the decimated mesh and the original mesh. Figure 5.5 shows a sample result. Naturally, the deformed source mesh loses detail as the mesh resolution is reduced but our framework is able to provide a plausible solution at all cases. The regularity of the mesh triangulation is also preserved even if at 25% the faces of the source mesh start forming discrete bands same-sized triangles.

5.6 Comparison with related work

We evaluated our non-rigid registration framework with samples of the D3DFACS and the *Headspace*[20] datasets. We compare our results with the generic optimal step algorithm of Amberg et al. [2][1] and the adaptative template framework of Dai et al. [20]. As the latter work, we employ two metrics for the quantitative evaluation, the *nearest vertex error* and the *landmark error*. The landmark error measures the average distance between landmarks. The nearest vertex error measures the distance between each point of the deformed source and its closest point on the target. The distances are summed and averaged over the number of vertices. Vertices whose closest point lies on a border are discarded as they may belong to a hole and they can artificially inflate the mean error. For the qualitative evaluation we looked at how well is the user identity preserved and how well is the original mesh topology preserved.

5.6.1 Comparison with Amberg et al.[2]

For the experiments with Amberg's optimal step non-rigid registration algorithm, no landmarks were employed as they did not have a significant effect on the result. Amberg's algorithm was set to 10 weight schedules where the stiffness parameter α linearly decayed from 100 to 20. Since the algorithm assumes proximity between the target and the source mesh, they were previously aligned with the methods described in Sections 4.2 and 4.3. The results were compared to our framework set with the parameters described on Section 5.2.2 and with the adaptative template step with the 14 landmarks of Figure 5.1. Neutral meshes and expressive meshes of the D3DFACS dataset were employed.

Neutral Meshes We extracted one mesh with a neutral expression for each of the 10 users of the D3DFACS dataset. The results are shown in Figures 5.6

[1]Available Matlab implementation at `https://nl.mathworks.com/matlabcentral/fileexchange/54077-optimal-step-nonrigid-icp`. Last accessed: 07-09-2018.

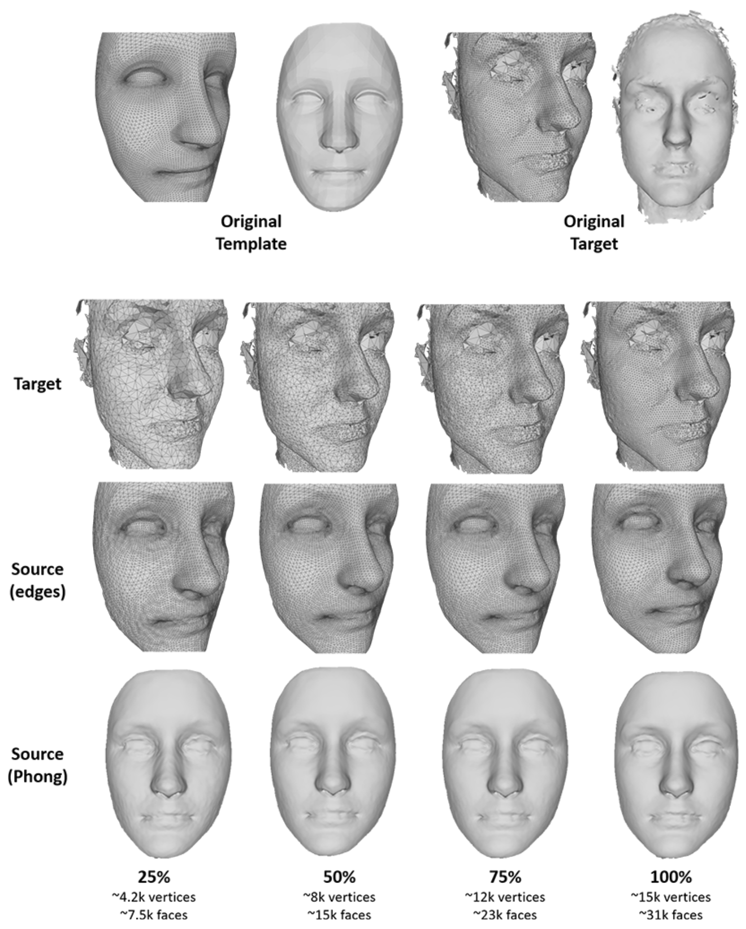


Figure 5.5: Resulting source meshes of the same target mesh at different resolutions. Our template has 12,073 vertices and 24,014 faces. The second and third row show the edges of the target and deformed source mesh respectively. The bottom row shows the deformed source under the standard Phong reflection model.

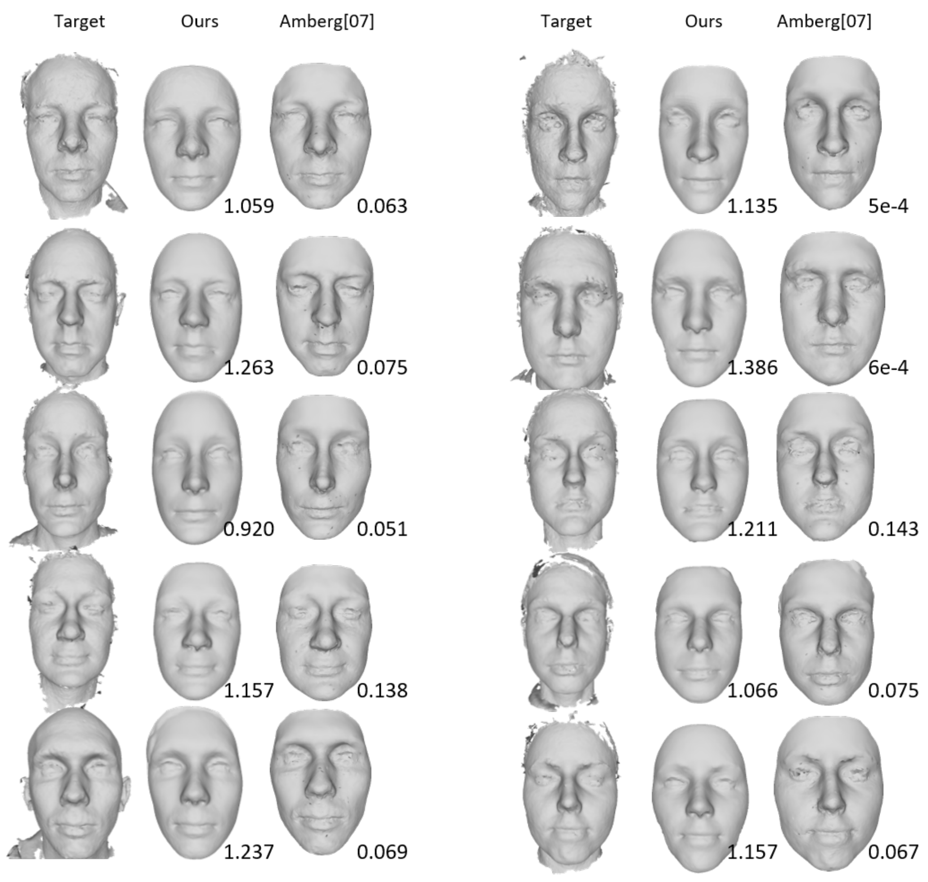


Figure 5.6: Comparison with Amberg's optimal step algorithm on neutral expression meshes. The number at the lower-right corner indicates the mean per-vertex error (in mm.).

and 5.8. Amberg's algorithm achieves a lower error and captures better the user identity. However, correspondences are not constrained enough to produce smooth deformations and several artifacts appear. Specifically, in Figure 5.8, we observe how the triangle faces deform abruptly around noisy areas of the target mesh. Holes are correctly assessed by both algorithms but Amberg's results present collapsed vertices and stretches around the borders of the holes. Lastly, some of our results presented displaced patches at the boundaries of the mesh. The patches are caused by the difference in proportions between the template and the target. When the target and source mesh differ too much, the distance between corresponding points at the boundaries is higher than the rest of the vertices and our bounding sphere cannot reach them.

Expressive Meshes We selected 5 facial movements of the D3DFACS database to assess the results of our algorithm when performed on expressive meshes (see Figure 5.7). For every facial movement, we extracted from the sequence two meshes at the middle and at most intense point of the expression. This way, we captured two different levels of deformation with respect to the neutral expression.

The facial movements chosen were: eyebrows raise, lips part, lip depressor, lip sucking and nose wrinkler. These movements comprise a wide range of the possible facial movements at different parts of the face. In total, we collected 89 meshes as not all users included the specific actions.

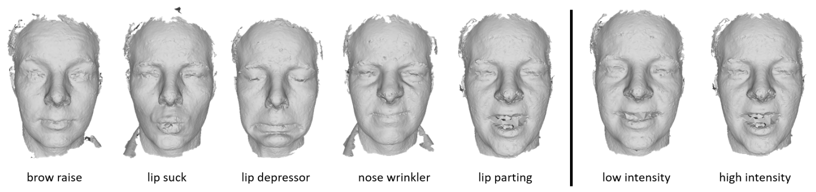


Figure 5.7: Left: Selected facial movements from the D3DFACS database. Right: Two meshes of different expression intensities are collected for each action.

Some result samples are shown in Figure 5.9. Amberg's algorithm captures better the user identity and its expressiveness while our algorithm has a more smoothed output. As expected, intense expressions generally yield a higher mean-vertex error. Amberg's algorithm is able to achieve an error of less than 1mm in all cases at the cost of losing the template regularity. In our method, the smoothness prior causes that intense expressions deform the proportions of the template, producing a pronounced head narrowing at the lip suck action.
Both methods struggle at finding a plausible deformation for the lip parting movement. The are two reasons for this. First, the target meshes already present a disorganized structure at the interior of the mouth with uneven densities of triangles. Secondly, the template meshes do not have faces for the interior of the mouth, so they have to employ faces of the parts at their vicinity, stretching their lips. We can conclude that both methods are unsuitable for strong expressive meshes.

Correspondences Corresponding points of the original and deformed template mesh present a subtle misalignment on Amberg's algorithm (Figure 5.8, bottom). The misalignment becomes stronger with expressive meshes yet our algorithm is able to preserve in most cases a good feature correspondence between different results.

Robustness We evaluated the robustness of the methods by their ability to fill an artificially created empty region on a mesh (see Figure 5.10). Our results

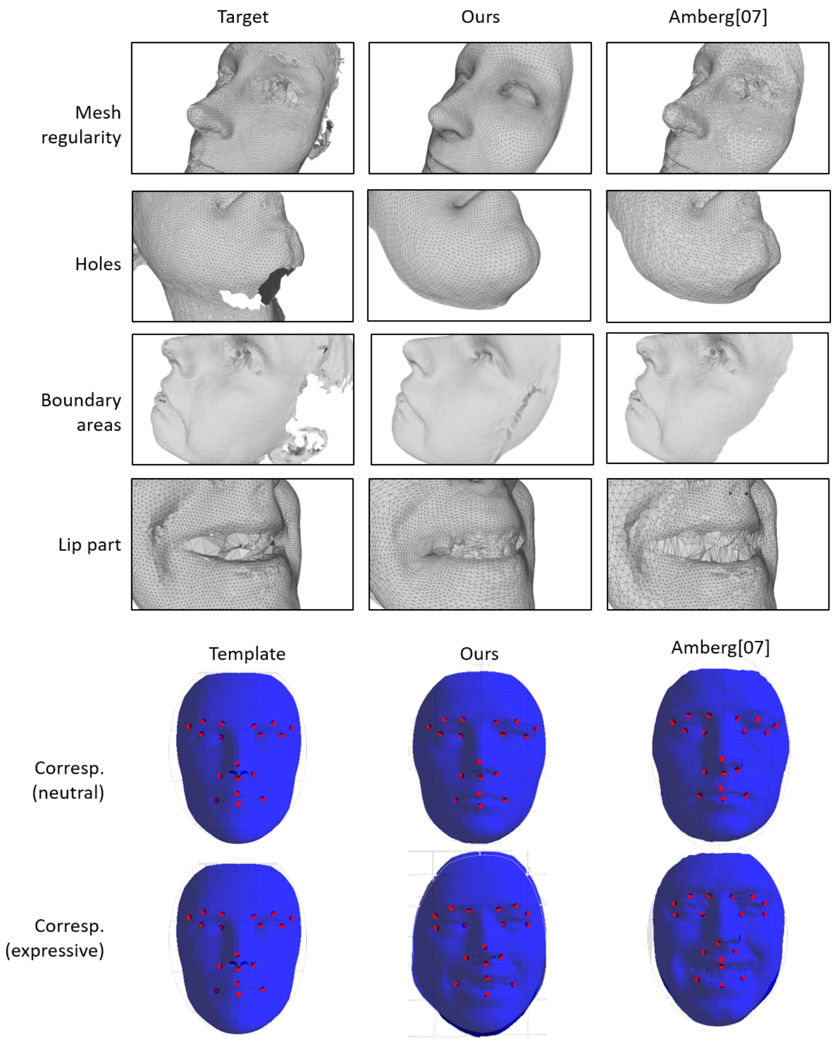


Figure 5.8: Detail comparison between our method and Amberg's optimal step algorithm. Bottom rows display red dots on locations of feature points on the template before and after the non-rigid registration. The desired result is that the marks are placed at the semantically same places.

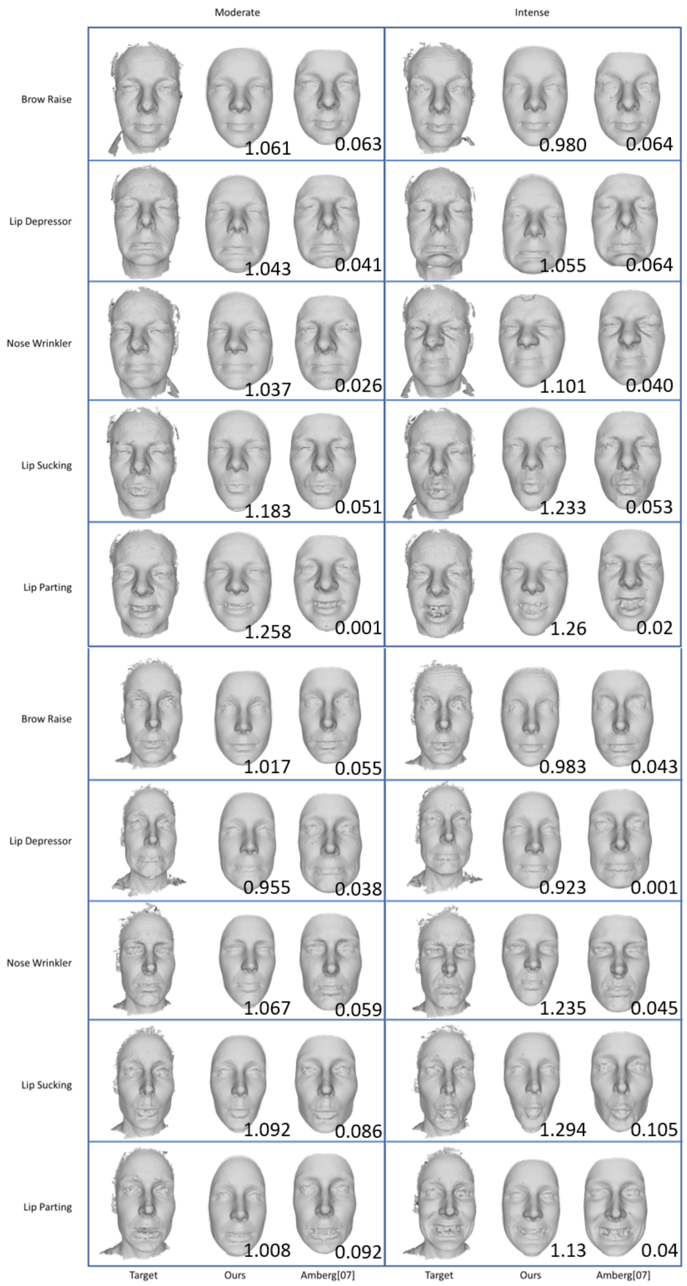


Figure 5.9: Comparison of expressive meshes with Amberg's optimal step algorithm. The number at the lower right corner indicates the mean per-vertex error (in mm).

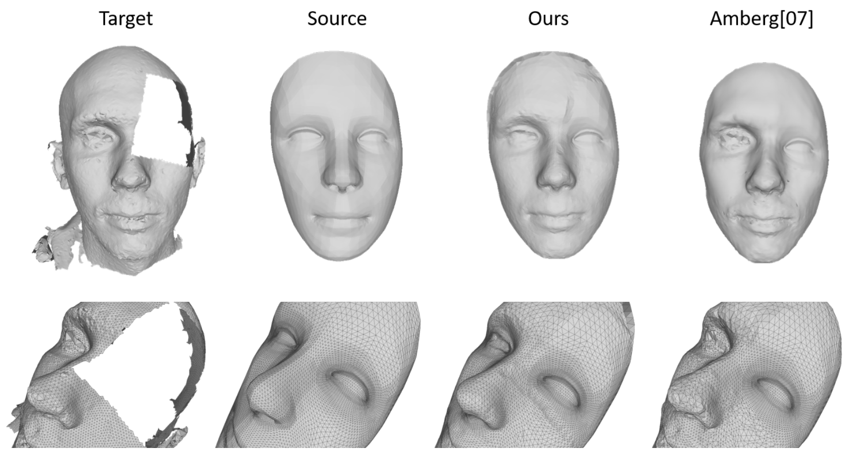


Figure 5.10: Robustness comparison of our algorithm at filling an artificially created hole. The bottom row shows the edges of the resulting mesh.

show that both methods succeed at preserving the template geometry at the empty areas, even if our method presents some agglomerations of vertices at the hole boundary.

5.6.2 Comparison with Dai et al.[20]

The adaptative template framework of Dai et al. was tested with 1212 scans of the Headspace dataset. We also extracted samples from the Headspace dataset and compared our results against their performance measures. However, there are some differences between Dai et al. experiments and ours.

The framework of Dai et al. adapts a template to the user geometry prior to the non-rigid registration through landmark correspondences. The landmarks are extracted automatically by the texture channel of the scans. We do not extract the landmarks automatically but we can leverage on the 68 annotated 3D landmarks included on most of the users of the Headspace dataset. We also annotated the same 68 landmarks for our template. We executed the pre-alignment and rigid alignment phases using only the landmarks and applied the resulting transformation to the rest of the mesh.

Due to time limitations, we reduced the number of scans for our experiments to 100. Since our template resembles an adult face and our method does not handle well strong deformations, subjects younger than 20 and older than 60 years were discarded. Subjects flagged with mesh artifacts, non-neutral expression and hair bulges were also discarded. The selected subjects were evenly divided between female and male.

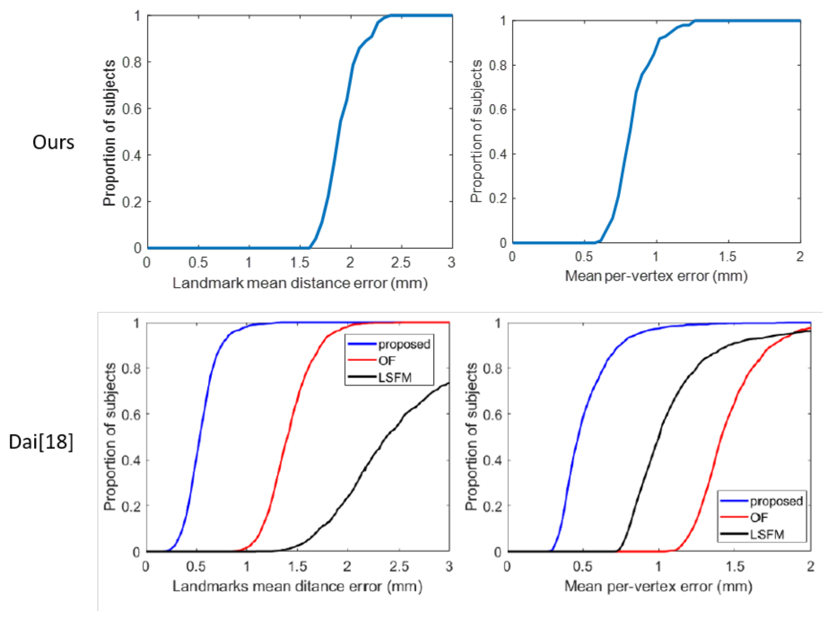


Figure 5.11: Preliminary comparison with Dai et al. The red and black line describes results obtained with the Open Framework [35] and LSFM [7] morphing respectively. Bottom graphs retrieved from Dai[20].

The results are shown in Figure 5.11, where we count the proportion of subjects under a certain mean error. Our method does not outperform Dai's framework but it has a similar range of error as other contemporary works. For %90 of our meshes we achieve a mean vertex error of 1mm with respect to the $\sim$ 0.6mm of Dai and a mean landmark error of $\sim$ 2.2mm. Lastly, we computed the per-vertex nearest point error of the resulting meshes and compared the results with the figures shown in [20] (see Figure 5.12). The deformed meshes produced by our framework are able to capture the user identity but they present a larger vertex error, especially around the forehead and cheeks.

5.7 Generability

The original purpose of our algorithm is to register face scans. However, in none of the steps of our method we leveraged on the fact that the objects to be registered are human faces except on the pre-alignment step. Even in the pre-alignment, the PCA analysis still holds valid for any object that has characteristic dimensions. It is interesting then to see if we can register other objects with an adequate template.

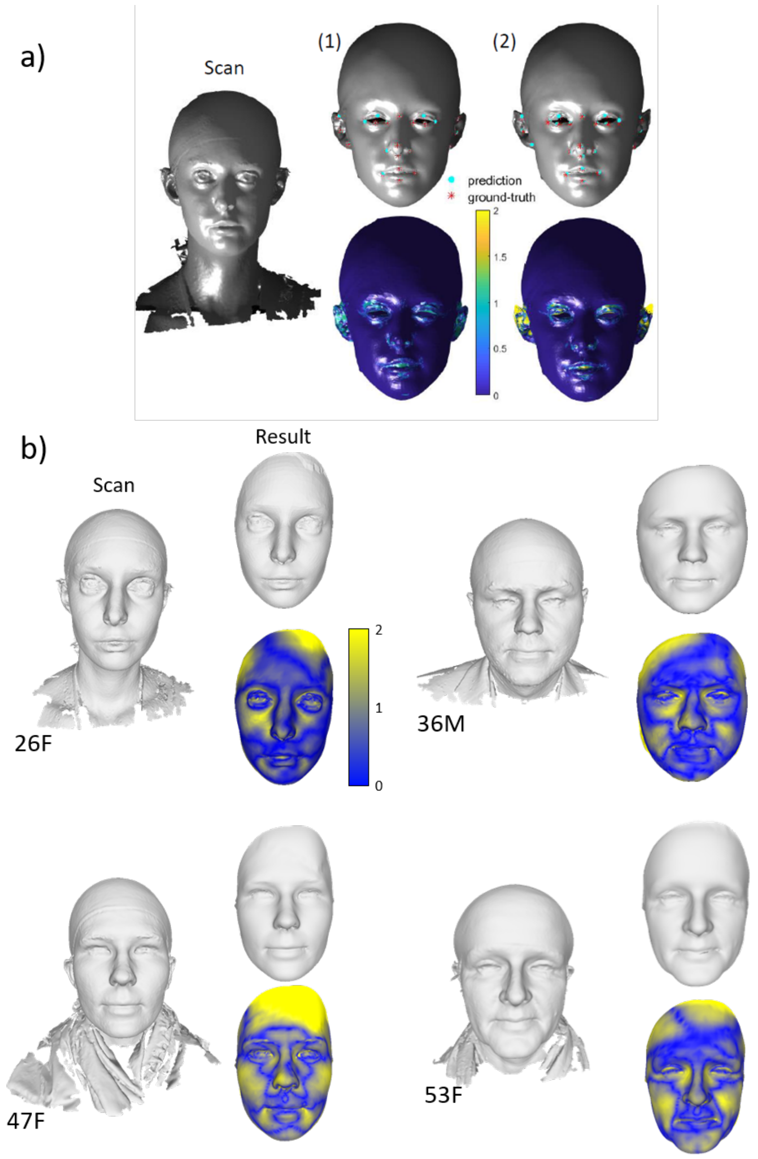


Figure 5.12: Preliminary result comparison with Dai et al. a) Per-vertex nearest point error map of the adaptative template framework of Dai (image retrieved from [20]). b) Sample results of our algorithm with users of different ages and gender. Error is measured in mm for figures a) and b).

In Figure 5.13, we show two fragments of a pelvic bone, one complete and another split with a rectangle-shaped hole. These models were employed on the non-rigid registration framework work of Audenart [3][2]. Both target and source mesh have known correspondences, and therefore, we can evaluate the quality of the algorithms by measuring the per-vertex distances. We ran the algorithm of [3] with 15 iterations and our algorithm and compared results. Overall, our algorithm presents a more plausible solution on the complete version and a smaller RMS error on the split version of the model.

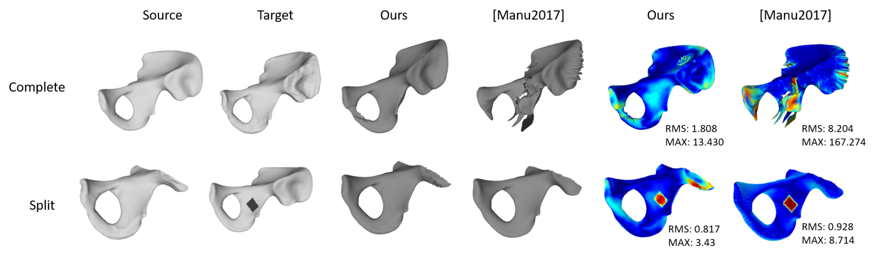


Figure 5.13: Non-rigid registration result comparison on a complete pelvic bone and a split pelvic bone.

It should be noted that our algorithm presents some limitations compared to [3]. Specifically, in Figure 5.14 we observe that a flat surface on the pelvic bone is full of holes. This error is caused by the thinness of the bone area, also called *ilium wing*. When calculating correspondences, the bounding sphere around the vertices in the ilium is big enough to collect points at both sides of the bone. Some of these points get assigned correspondences at the other side and the prior terms are not restrictive enough to balance the matching term.

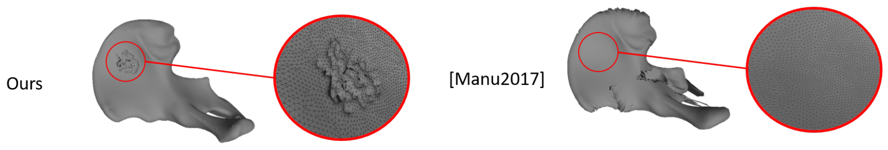


Figure 5.14: Comparison of results on a surface patch on the bone (Ilium). Left: Our algorithm presents holes caused by bad correspondences. Right: The method proposed by [3] presents a smooth surface on the highlighted patch.

Despite the issues mentioned above, we believe that this case study demonstrates that our algorithm can be potentially extended to other areas of study.

[2]Still in development (last visit on 17-08-2018).

Chapter 6

Conclusion

In this thesis, we presented a template fitting, non-rigid registration framework for morphing a clean and smooth template towards facial and craniofacials depth scans. We call the depth scan *target mesh* and our template *source mesh*. Our proposed method has four phases:

- A prealignment phase brings the source and target mesh into rough alignment and avoids getting trapped in local minima for the following steps.
- An iterative closest point algorithm that brings the source and target mesh in close alignment.
- An optional template adaptation phase in which the template is deformed to roughly match the users geometry by bringing together landmarks. This process has proven to improve the results on the final source mesh.
- An iterative closest point non-rigid registration phase that deforms the source mesh towards the target mesh by bringing together correspondent points. We pose the problem as an energy minimization equation with one matching term and three regularization terms.

The framework was tested with the datasets of D3DFACS and Headspace and they were compared to the results of Amberg et al. [2] and Dai et al. [20] We analyzed how well was the subject identity captured, the regularity of the meshes, the nearest per-vertex error and the correspondence between deformed meshes.

Amberg's optimal step algorithm captured better the user identity and yielded a lower mean per-vertex error at the cost of overfitting on the target mesh. Additionally, the produced meshes did not have good 1-to-1 correspondences between them at key feature points. Compared to Amberg, our method produces smoother and cleaner meshes that resemble the subject identity and are in good correspondence with the original template features. Both methods deal well with holes but our algorithm present subtle agglomerations at hole boundaries and indentations when the template proportions differ too much from the subject. This issue was caused by the bounding sphere used to collect candidates since it was not big enough to reach points that are too far. Lastly, both methods were tested against expressive meshes, producing mixed results on target meshes that presented strong expressions.

Dai's framework presented a lower mean per-vertex error, where 90% of meshes were under 0.6mm versus 1mm in our case. Dai also achieves a lower landmark error than our method on the Headspace dataset (90% under 1mm and 2mm in our case) but our performance measures are within the range of this and the Open Framework and Large Scale Face Model morphable models. These results should be interpreted with caution as we are only comparing error measures and figures and we have not implemented the framework ourselves.

Lastly, we proved how our method was able to produce a plausible result for a medical range scan, showing that our algorithms can possibly be extended to other areas of study.

6.1 Answering research questions

RQ1: *How can a non-rigid registration framework adapt to different face proportions?*

We designed an adaptative template step on Section 5.3 that roughly matches the template shape proportions to the scan by bringing together corresponding landmarks under Laplace-Beltrami constraints. We demonstrate that performing this step yields a considerable lower vertex error than employing the original template. The adaptative template provides a better initialization for the non-rigid registration algorithm as it is able to close the distance between the template features to the scan features. This naturally leads to better pairwise correspondences and consequently a better end result.

RQ2: *How can a non-rigid registration framework retrieve a smooth mesh from a scan in the presence of holes and noise?*

In Section 4.5.2, we designed a trimming and downweighting scheme that aims to mitigate the effect of bad correspondences by lowering or eliminating the matching term. That is, we favoured following the original geometry of the mesh and producing a smooth deformation over following correspondences. To take into account holes, we take out the correspondences to target boundary points. Furthermore, in Section 4.5.1, we also draw a bounding sphere around every point of the template so only the target points that lie inside the sphere are eligible to be the corresponding point of the source vertex. The consequence is that source vertices located in hole areas just follow the original template geometry and deformation constraints. Figure 5.10 shows that the resulting mesh has a plausible result.

As for the noise, the local rigidity prior that we designed in Section 4.4 forces neighbouring vertices to follow similar transformations. If a source vertex gets a noisy target vertex assigned, its neighbours will "calm it down", by forcing it to follow an average of their translations. In our comparison with Amberg in Section 5.6.1, we demonstrate that our meshes are unaffected by the noise of the D3DFACS database.

RQ3: *How can a non-rigid registration adapt to meshes of different resolutions?*

The conjunction of local and global rigidity constraints, summed to downweighting correspondences according to a distance function allow us to produce a plausible solution for meshes that have at least 4 times less faces than our template. We demonstrate that our framework produces good results in Section 5.5.

6.2 Contributions

Our contributions can be summarize in the following points:

1. A template fitting, non-rigid iterative closest point algorithm capable of capturing the user identity and preserving the regularity of the template for facial scans generated with stereo cameras.
2. A method to roughly adapt the proportions of a template mesh towards a target mesh by employing corresponding landmarks on important features.
3. A simple pre-alignment scheme for bringing two face meshes that avoids Iterative Closest Point local minima.
4. An automatic downweighting scheme for pruning bad correspondences.
5. A detailed assessment of the influences of each weighted term in non-rigid registration.

6.3 Applications

The most straightforward application of our non-rigid registration algorithm is the construction of a clean database of facial meshes. These databases are extremely useful as they present less noise, no holes and a fixed number of vertices and faces in good correspondence. They have been widely employed in the construction of morphable models but recent work in facial animations have also leveraged from structured meshes for deep learning frameworks [51].

If we also desired to animate the obtained face, we can leverage from the good correspondence between the vertices of the original source and the morphed source. If we have an expressive version of the original source, one can "copy" the deformations between the neutral and expressive mesh and "paste" it to the morphed source employing a deformation transfer operation [47].

In this thesis, we presented a small medical example for which a template pelvis bone is fitted towards a scanned one with acceptable results. We believe that any 3D reconstruction application for which a rough template is provided could potentially benefit from our work.

6.4 Limitations and Future Work

We consider that our algorithm is suitable for neutral and slightly expressive meshes but not for strongly expressive meshes (at least with our neutral expression template). Since in the non-rigid registration process correspondences are based on distance, when the source mesh and an expressive mesh are aligned, corresponding features such as lip corners, brows and cheeks are located at different points. A possible solution to this issue could be to employ a blendshape model instead of a fixed template. After the rigid alignment of a neutral mesh, a blendshape weight term such as the one in [11] could be included in Equation 5.2 to obtain template that roughly matches the expression of the target mesh and guarantee a better final result.

The importance of the template is not only limited to its expressiveness. A template that roughly matches the user proportions is recommended, otherwise indentations may appear near the most different areas. A possible solution to this problem could be to increment the bounding sphere radius of Section 4.5 on points that have a low geodesic distance to the source mesh boundary. Another option would be to fit a morphable model towards the target mesh to obtain a stronger prior.

Our method does not capture the user identity in fine detail as the smoothness term and local rigidity term constrain small displacements from the local neighbour of a vertex. These details are rarely included on the meshes and they can be easily mistaken as noise as they are at the same value scale. The texture channel of the target mesh could be employed after the nonrigid registration is completed to reconstruct small scale features such as wrinkles, lip shape and mouth corners.

Lastly, our pre-alignment scheme will not succeed for scans that present big holes or extra triangles since the axes of Figure 4.2 will be skewed. Recent frameworks have pre-aligned 3D objects by applying an Iterative Closest Point algorithm on points that share a similar shape descriptor. In the course of this work, we attempted to perform pre-alignment and landmark detection with spin images [30] yet our template seemed to differ to much from the target as to obtain a good result.

We look forward to investigate these and other related issues in our future work.

Appendix A

Term Weight comparison

This appendix contains a set of tables comprising results of our algorithm when employed on the neutral expression mesh of one user of the D3DFACS. The algorithm is only executed with one iteration, two weights are set to $\omega = 10$ and the other two (indicated on the upper-left corner) are varied. The tables are color coded to signal undesired deformations of the source meshes:

- The dark gray meshes are considered acceptable results.
- The orange meshes are considered to have copied the imperfections of the target mesh (present collapsed vertices, stretches, self-intersections...).
- The blue meshes are considered to not have captured the user identity.
- The green meshes present undesirable deformations caused by the local rigidity term (Equation 4.12).
- The light gray meshes present numerical errors due to the Laplace-Beltrami calculations.

The figure below shows a reference of the target mesh and the source mesh.

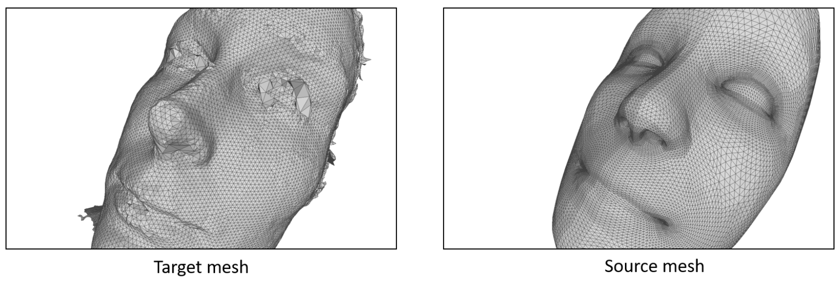


Figure A.1: Reference meshes for result comparison.

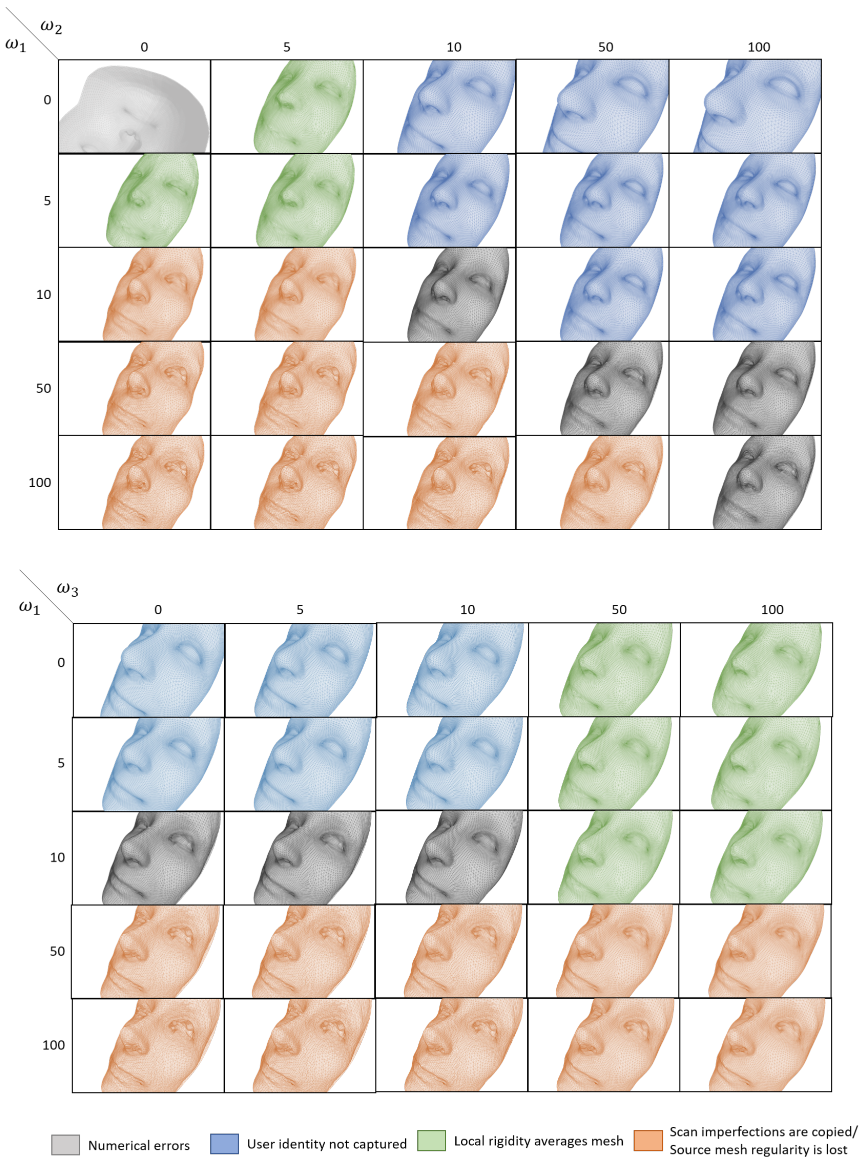

ω_2
ω_1
0
5
10
50
100
ω_3
ω_1
Numerical errors
User identity not captured
Local rigidity averages mesh
Scan imperfections are copied/ Source mesh regularity is lost

Figure A.2: Top: Matching weight ω_1 vs. global rigidity prior weight ω_2. When ω_1 is high wrt. ω_2, our method strongly favors following correspondences and the resulting meshes present stretches and self-intersections. When ω_2 is too high, the template remains in its original form. If both are low, the resulting mesh presents an undesired flattening effect. Bottom: Matching weight ω_1 vs. local rigidity prior weight ω_3. If ω_3 is high wrt. ω_1, the flattening effect appears. If both are low, the global rigidity dominates and the user identity is not captured.

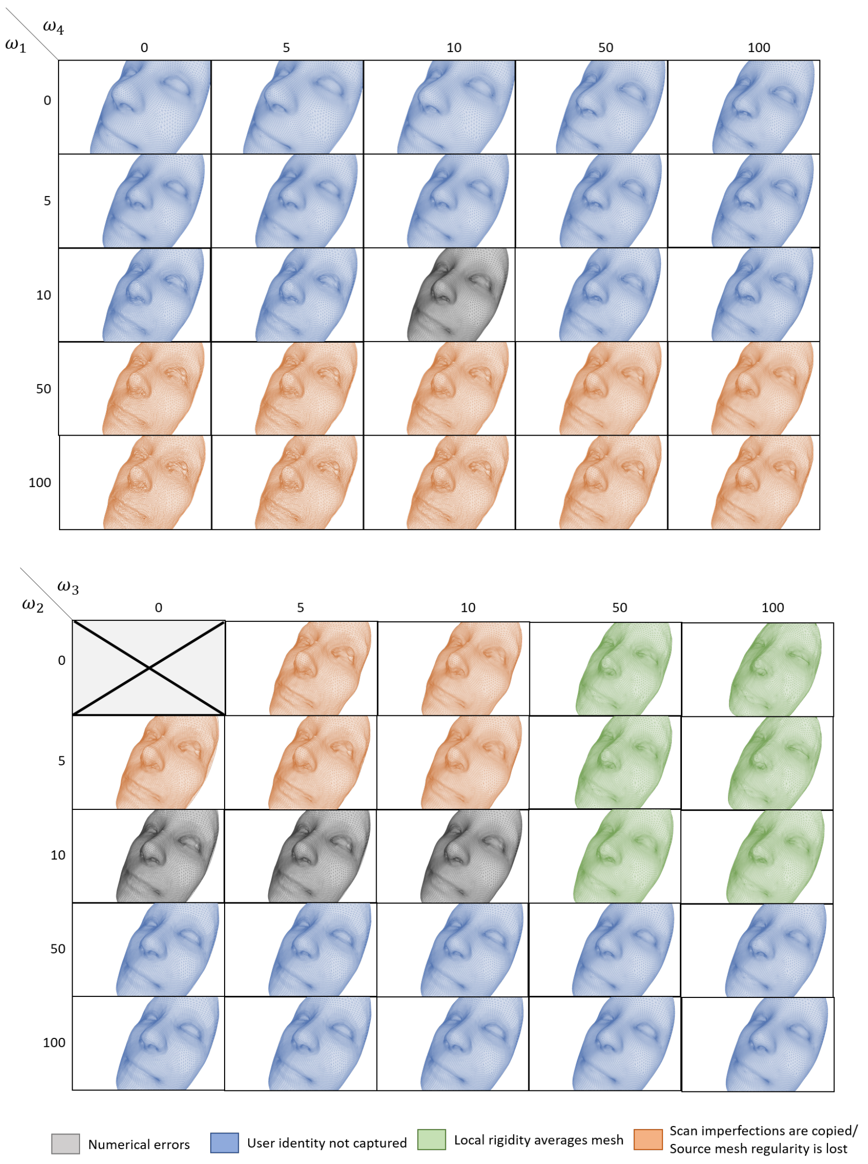
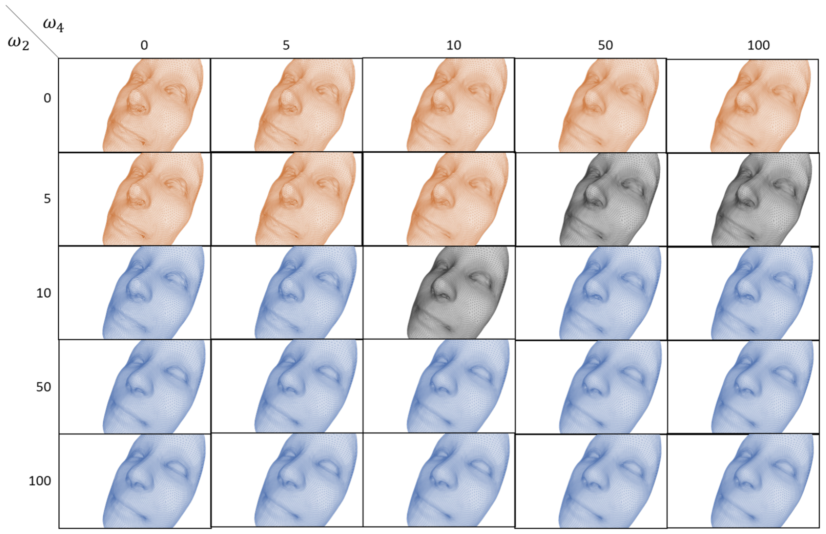

ω_4
ω_1
0
5
10
50
100
ω_3
ω_2
Numerical errors
User identity not captured
Local rigidity averages mesh
Scan imperfections are copied/ Source mesh regularity is lost

Figure A.3: Top: Matching weight ω_1 versus smoothness prior weight ω_4. If ω_1 is low, the user identity is not well captured and if it is too high, stretches and triangle self-intersections appear. The smoothness weights produces a subtle smoothing effect (noticeable at the eyelids). Acceptable results are obtained when both weights are balanced. Bottom: Global rigidity prior weight ω_2 versus local rigidity prior weight ω_3. When both priors are low, the matching term dominates and the meshes follow correspondences without respecting the original template geometry. If both are high, the constraints impede the mesh to deform towards correspondences and the user identity is not captured. Acceptable results are obtained when both weights are balanced.



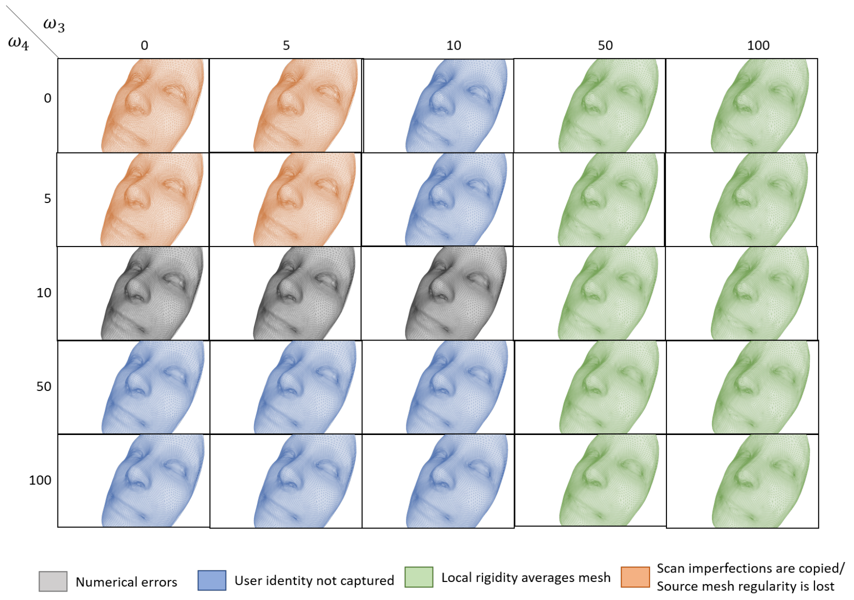

ω_3
ω_4
0
5
10
50
100
0
5
10
50
100
Numerical errors
User identity not captured
Local rigidity averages mesh
Scan imperfections are copied/ Source mesh regularity is lost

Figure A.4: Top: Global rigidity weight ω_2 versus smoothness prior weight ω_4. If both weights are high, the constraints impede the mesh to deform towards correspondences and the user identity is not captured. A low global prior weight and a high smoothness prior weight (grey meshes of second row) produces a smooth mesh that adopts in greater detail the user geometry without causing stretching. Bottom: Local rigidity weight ω_3 versus smoothness prior weight ω_4. Regardless of the smoothness prior weight, if the local rigidity prior weight is high it will dominate over the other terms producing a flattening effect. If both are low, the smoothness of the template is partially lost. Good results are obtained when both weights are balanced.

Appendix B

Smoothness term derivative coefficients

Let $\tilde{v}_i$ be the vertex i of the deformed mesh at a certain iteration and v_i be the same vertex on the original mesh composed of n vertices. The cotangent discretization of the Laplace-Beltrami operator applied on a coordinate function is:

$$l_i = \frac{1}{2\mathcal{A}_i} \sum_{j \in \mathcal{N}_i} (\cot\alpha + \cot\beta)(v_j - v_i) \tag{B.1}$$

We can decompose the previous formula for all vertices into a mass matrix $\mathcal{M}$ and a cotangent weight matrix C [46]. $\mathcal{M}$ is a diagonal matrix of elements $\mathcal{A}_i$, where $\mathcal{A}_i$ is the mixed Voronoi area as defined in Equations 3.18 or 3.19:

$$\mathcal{M} = \begin{bmatrix} \mathcal{A}_1 & & \\ & \ddots & \\ & & \mathcal{A}_n \end{bmatrix} \tag{B.2}$$

Notice that since $\mathcal{M}$ is diagonal it follows that:

$$\mathcal{M}^{-1} = \begin{bmatrix} \frac{1}{\mathcal{A}_1} & & \\ & \ddots & \\ & & \frac{1}{\mathcal{A}_n} \end{bmatrix} \tag{B.3}$$

C is a $n \times n$ sparse matrix with non-zero entries on its main diagonal and on the elements ij where $j \in \mathcal{N}_i$:

$$C_{ii} = -\sum_{j \in \mathcal{M}} C_{ij}, \qquad C_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}) \tag{B.4}$$

The Laplace operator L is equal to the product of the two matrices $L = \mathcal{M}^{-1}C$. We denote L entries with l_{ij}. As explained in Section 4.4, we apply the Laplace operator on the deformation vector $d_i = \tilde{v_i} - v_i$. Let $\mathbf{d}$ be the stacked vector of length n with the deformation vectors of all vertices. The residual smoothness term is defined as:

$$E_{\text{smoothness}} = \|L \cdot d\|_2^2 = \left\| \begin{bmatrix} l_{11} & \dots & l_{in} \\ & \ddots & \\ l_{n1} & \dots & l_{nn} \end{bmatrix} \cdot \begin{bmatrix} \tilde{v_1} - v_1 \\ \vdots \\ \tilde{v_n} - v_n \end{bmatrix} \right\|_2^2 \tag{B.5}$$

Expanding Equation B.5 we obtain a large sum of $n \times n$ terms:

$$\begin{aligned} l_{11}^2(\tilde{v_1} - v_1)^2 + l_{12}^2(\tilde{v_2} - v_2)^2 + \cdots + l_{1n}^2(\tilde{v_n} - v_n)^2 + \\ l_{21}^2(\tilde{v_1} - v_1)^2 + l_{22}^2(\tilde{v_2} - v_2)^2 + \cdots + l_{2n}^2(\tilde{v_n} - v_n)^2 \\ + \cdots + \\ l_{n1}^2(\tilde{v_1} - v_1)^2 + l_{n2}^2(\tilde{v_2} - v_2)^2 + \cdots + l_{nn}^2(\tilde{v_n} - v_n)^2 + \\ 2l_{11}l_{12}(\tilde{v}_1 - v_1)(\tilde{v}_2 - v_2) + 2l_{11}l_{13}(\tilde{v}_1 - v_1)(\tilde{v}_2 - v_2) + \cdots + 2l_{11}l_{1n}(\tilde{v}_1 - v_1)(\tilde{v}_n - v_n) + \\ 2l_{21}l_{22}(\tilde{v}_1 - v_1)(\tilde{v}_2 - v_2) + 2l_{21}l_{23}(\tilde{v}_1 - v_1)(\tilde{v}_3 - v_3) + \cdots + 2l_{21}l_{1n}(\tilde{v}_1 - v_1)(\tilde{v}_n - v_n) \\ + \cdots + \\ 2l_{n1}l_{n2}(\tilde{v}_1 - v_1)(\tilde{v}_2 - v_2) + 2l_{n1}l_{n3}(\tilde{v}_1 - v_1)(\tilde{v}_3 - v_3) + \cdots + 2l_{n1}l_{nn}(\tilde{v}_1 - v_1)(\tilde{v}_n - v_n) \end{aligned} \tag{B.6}$$

The previous sum forms a system of quadratic equations that can be solved by equating its partial derivatives to 0. We calculate the partial derivatives with respect to every point $\tilde{v}_i$ and gather the terms into a coefficient matrix A and an independent term vector b to solve for $A\tilde{\mathbf{v}} = b$. The partial derivative of any vertex $\tilde{v_i}$ follows the expression:

$$\begin{aligned} \frac{\partial E_{\text{smooth}}}{\partial \tilde{v_i}} &= 2l_{1i}^2(\tilde{v}_i - v_i) + 2l_{1i}l_{12}(\tilde{v}_2 - v_2) + \cdots + 2l_{1i}l_{1n}(\tilde{v_n} - v_n) \\ &+ 2l_{2i}^2(\tilde{v}_i - v_i) + 2l_{2i}l_{22}(\tilde{v}_2 - v_2) + \cdots + 2l_{2i}l_{2n}(\tilde{v_n} - v_n) \\ &+ \cdots + \\ &2l_{ni}^2(\tilde{v}_1 - v_1) + 2l_{ni}l_{n2}(\tilde{v}_2 - v_2) + \cdots + 2l_{ni}l_{nn}(\tilde{v_n} - v_n) \end{aligned} \tag{B.7}$$

We can group the terms into a coefficient matrix:

$$\begin{bmatrix} \frac{\partial E}{\tilde{v}_1} \\ \frac{\partial E}{\tilde{v}_2} \\ \vdots \\ \frac{\partial E}{\tilde{v}_n} \end{bmatrix} = 2 \begin{bmatrix} l_{11}^2 + l_{21}^2 + \cdots + l_{n1}^2 & \cdots & l_{11}l_{1n} + l_{21}l_{2n} + \cdots + l_{n1}l_{nn} \\ & \vdots & \\ l_{11}l_{12} + l_{21}l_{22} + \cdots + l_{n1}l_{n2} & \cdots & l_{12}l_{1n} + l_{22}l_{2n} + \cdots + l_{n2}l_{nn} \\ l_{11}l_{1n} + l_{21}l_{2n} + \cdots + l_{n1}l_{nn} & \cdots & l_{n1}^2 + l_{n2}^2 + \cdots + l_{nn}^2 \end{bmatrix} \mathbf{d} \tag{B.8}$$

The coefficient matrix can be simplified as:

$$\frac{dE}{d\tilde{\mathbf{v}}} = \begin{bmatrix} \sum_{i=1}^{n} l_{i1}^2 & \sum_{i=1}^{n} l_{i1}l_{i2} & \cdots & \sum_{i=1}^{n} l_{i1}l_{in} \\ \sum_{i=1}^{n} l_{i1}l_{i2} & \sum_{i=1}^{n} l_{i2}^2 & \cdots & \sum_{i=1}^{n} l_{i2}l_{in} \\ \sum_{i=1}^{n} l_{in}l_{i1} & \sum_{i=1}^{n} l_{in}l_{i2} & \cdots & \sum_{i=1}^{n} l_{in}^2 \end{bmatrix} \mathbf{d} \tag{B.9}$$

This way, it is easy to see that the coefficient matrix is actually $L^{\top}L$. The independent term vector is simply the result of multiplying $2L^{\top}L \cdot \mathbf{v}$.

As an additional note, if the mesh is regular, every vertex will have on average 6 neighbours except those which lie on boundaries (for which the Laplace operator is not defined). Therefore, for a mesh of 1000 vertices only $\approx 0.07\%$ of entries in l will be non-zero, producing a very sparse matrix.

Bibliography

[1] *3dMD*. `http://www.3dmd.com/`. 2018.

[2] Brian Amberg, Sami Romdhani, and Thomas Vetter. “Optimal step non-rigid ICP algorithms for surface registration”. In: *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[3] A. Audenaert Emmanuel. “Bootstrapped statistical shape model based segmentation of the full lower limb in CT”. In: 2017.

[4] Mirela Ben Shen and Andy Lai Lin. “Geometry Processing Algorithm - Mesh Simplification (Lecture Slides)”. In: University of Stanford. 2010.

[5] Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In: *Sensor Fusion IV: Control Paradigms and Data Structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–607.

[6] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1999, pp. 187–194.

[7] James Booth et al. “Large scale 3d morphable models”. In: *International Journal of Computer Vision* 126.2-4 (2018), pp. 233–254.

[8] Mario Botsch et al. *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[9] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. “Dynamic 2D/3D Registration.” In: *Eurographics (Tutorials)*. 2014, p. 7.

[10] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. “Sparse iterative closest point”. In: *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*. Eurographics Association. 2013, pp. 113–123.

[11] Sofien Bouaziz, Yangang Wang, and Mark Pauly. “Online modeling for realtime facial animation”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 40.

[12] Sofien Bouaziz et al. “Modern techniques and applications for real-time non-rigid registration”. In: *SIGGRAPH ASIA 2016 Courses*. ACM. 2016, p. 11.

[13] Chen Cao et al. “3D shape regression for real-time facial animation”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 41.

[14] Chen Cao et al. “Real-time high-fidelity facial performance capture”. In: *ACM Transactions on Graphics (ToG)* 34.4 (2015), p. 46.

[15] Yang Chen and Gérard Medioni. “Object modelling by registration of multiple range images”. In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[16] Shiyang Cheng et al. “Active nonrigid ICP algorithm”. In: *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*. Vol. 1. IEEE. 2015, pp. 1–8.

[17] Paolo Cignoni et al. “MeshLab: an Open-Source Mesh Processing Tool”. In: *Eurographics Italian Chapter Conference*. Ed. by Vittorio Scarano, Rosario De Chiara, and Ugo Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: `10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136`.

[18] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. “Active appearance models”. In: *IEEE Transactions on pattern analysis and machine intelligence* 23.6 (2001), pp. 681–685.

[19] Darren Cosker, Eva Krumhuber, and Adrian Hilton. “A FACS valid 3D dynamic action unit database with applications to 3D dynamic morphable facial modeling”. In: (2011).

[20] Hang Dai, Nick Pears, and William Smith. “Non-rigid 3D Shape Registration using an Adaptive Template”. In: *arXiv preprint arXiv:1803.07973* (2018).

[21] Hang Dai et al. “A 3D Morphable Model of Craniofacial Shape and Texture Variation”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[22] Hang Dai et al. “A 3d morphable model of craniofacial shape and texture variation”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 3104–3112.

[23] Manfredo P Do Carmo. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Courier Dover Publications, 2016.

[24] Shireen Elhabian, Amal Farag, and Aly Farag. “Iterative Closed Point: A tutorial on Rigid Registration”. In: University of Louisville, CVIP Lab. 2009.

[25] Bo Fan et al. “Photo-real talking head with deep bidirectional LSTM”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4884–4888.

[26] Pablo Garrido et al. “Reconstructing detailed dynamic face geometry from monocular video.” In: *ACM Trans. Graph.* 32.6 (2013), pp. 158–1.

[27] Pengyu Hong, Zhen Wen, and Thomas S Huang. “Real-time speech-driven face animation with expressions using neural networks”. In: *IEEE Transactions on neural networks* 13.4 (2002), pp. 916–927.

[28] Kai Hormann, Bruno Lévy, and Alla Sheffer. “Mesh parameterization: Theory and practice”. In: (2007).

[29] Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. “Dynamic 3D avatar creation from hand-held video input”. In: *ACM Transactions on Graphics (ToG)* 34.4 (2015), p. 45.

[30] Andrew E Johnson. “Spin-images: a representation for 3-D surface matching”. In: (1997).

[31] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. “Rotation invariant spherical harmonic representation of 3 d shape descriptors”. In: *Symposium on geometry processing.* Vol. 6. 2003, pp. 156–164.

[32] John P Lewis et al. “Practice and Theory of Blendshape Facial Models.” In: *Eurographics (State of the Art Reports)* 1.8 (2014).

[33] Hao Li, Robert W Sumner, and Mark Pauly. “Global correspondence optimization for non-rigid registration of depth scans”. In: *Computer graphics forum.* Vol. 27. 5. Wiley Online Library. 2008, pp. 1421–1430.

[34] Kok-Lim Low. “Linear least-squares optimization for point-to-plane icp surface registration”. In: *Chapel Hill, University of North Carolina* 4.10 (2004).

[35] Marcel Lüthi et al. “Gaussian process morphable models”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.8 (2018), pp. 1860–1873.

[36] Mark Meyer et al. “Discrete differential-geometry operators for triangulated 2-manifolds”. In: *Visualization and mathematics III.* Springer, 2003, pp. 35–57.

[37] Andriy Myronenko and Xubo Song. “Point set registration: Coherent point drift”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.

[38] Andrew Y Ng. “Feature selection, L 1 vs. L 2 regularization, and rotational invariance”. In: *Proceedings of the twenty-first international conference on Machine learning.* ACM. 2004, p. 78.

[39] Stephane Popinet. *Blender v2.79.* `https://www.blender.org/`. 2018.

[40] Elad Richardson et al. “Learning detailed face reconstruction from a single image”. In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on.* IEEE. 2017, pp. 5553–5562.

[41] Amy Ross. “Procrustes analysis”. In: *Course report, Department of Computer Science and Engineering, University of South Carolina* (2004).

[42] S. Rusinkiewicz. “Derivation of point-to-plane minimization”. In: 2013.

[43] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on.* IEEE. 2001, pp. 145–152.

[44] Shunsuke Saito, Tianye Li, and Hao Li. "Real-time facial segmentation and performance capture from rgb input". In: *European Conference on Computer Vision*. Springer. 2016, pp. 244–261.

[45] David C Schneider and Peter Eisert. "Fast nonrigid mesh registration with a data-driven deformation prior". In: *ICCV Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment.* Citeseer. 2009.

[46] Justin Solomon, Keegan Crane, and Etienne Vouga. *Laplace-Beltrami: The Swiss Army Knife of Geometry Processing.* Standford University. 2014.

[47] Robert W Sumner and Jovan Popović. "Deformation transfer for triangle meshes". In: *ACM Transactions on Graphics (TOG).* Vol. 23. 3. ACM. 2004, pp. 399–405.

[48] Supasorn Suwajanakorn, Ira Kemelmacher-Shlizerman, and Steven M Seitz. "Total moving face reconstruction". In: *European Conference on Computer Vision.* Springer. 2014, pp. 796–812.

[49] Rui Tang, Darren Cosker, and Wenbin Li. "Global alignment for dynamic 3d morphable model construction". In: *Workshop on Vision and Language (V&LW'12).* 2012, pp. 1–2.

[50] Sarah Taylor et al. "A deep learning approach for generalized speech animation". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 93.

[51] Ayush Tewari et al. "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction". In: *The IEEE International Conference on Computer Vision (ICCV).* Vol. 2. 3. 2017, p. 5.

[52] Thibaut Weise et al. "Face/off: Live facial puppetry". In: *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer animation.* ACM. 2009, pp. 7–16.

[53] Thibaut Weise et al. "Realtime performance-based facial animation". In: *ACM transactions on graphics (TOG).* Vol. 30. 4. ACM. 2011, p. 77.

[54] Eric W. Weisstein. *Regular Parametrization. From MathWorld—A Wolfram Web Resource.* Last visited on 14/07/2018. URL: `%5Curl%7Bhttp://mathworld.wolfram.com/RegularParameterization.html%7D`.

[55] Zhengyou Zhang et al. "Robust and rapid generation of animated faces from video images: A model-based modeling approach". In: *International journal of computer vision* 58.2 (2004), pp. 93–119.

[56] Xiangxin Zhu and Deva Ramanan. "Face detection, pose estimation, and landmark localization in the wild". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE. 2012, pp. 2879–2886.