Techniques for Performance-based, real-time Facial Animation

Manuel Antonio Sánchez Lorenzo

University of Sheffield, February 2006

ABSTRACT

The purpose of this research project has been to construct a real-time Facial Animation system that reproduces a wide range of aspects of facial skin motion using information captured from actual performers. The large-scale deformation of facial skin is controlled by marker-based optical Motion Capture through the application of geometry-warping algorithms specially developed for this purpose. Smaller effects such as wrinkling and buckling are addressed through shading techniques applied onto the warped facial geometry, evaluating a model of fine-scale tissue behaviour that is also built using data retrieved from actual subjects. The synthetic vision framework required for capturing such data is also introduced in this thesis. Additional aspects regarding the analysis and reusability of Motion Capture data are also considered, enabling the system to apply the data collected from a specific individual to different physiognomies. The real-time nature of this animation system is reinforced by implementing some of its time-critical components in dedicated hardware (programmable graphics cards), and preprocessing the facial geometry to make better use of such hardware. Finally, recommendations are made for future work of this nature.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The short but intense story of Computer Graphics shows a vigorous endeavor to produce plausible computer-generated representations of our perceptual reality, and in that context human to human communication plays a very important role, as it is arguably one of the most familiar phenomena to our minds. Such is the importance of this particular aspect that even other fields of Computer Science, like for instance Artificial Intelligence, use communicative frameworks (the famous *Turing test* [Tur50]) to gauge their evolution.

From a graphics point of view, the implications of convincingly portraying a human face are not just limited to providing an accurate representation of the skin geometry and its lighting properties, but also include reproducing facial motion in a way that conveys the same compelling expressions as the real counterpart. While the modeling and shading aspects of human faces are widely covered by generic frameworks [MMA89, KA91, ASW93], animation on its own remains an open problem, giving way to experimentation with a wide variety of techniques (see for instance [Par74, Wat87, KMM92, LTW95, PSS99, PP01]).

Over the last decade, performance-based approaches have brought a revolution upon the animation of the human figure, mainly driven by the significant advances in optical Motion Capture (MoCap) techniques [WB97, Vic, Opt, Pol]. However, such transformation has not made such a great impact in the field of Facial Animation, as the biomechanical processes involved are far more complex, and the resulting soft tissue deformation is perceptually important (conversely to the case of full body animation, where skeletal posing plays the most representative role). Similarly, the anatomical complexity of the face prevents simple generalizations such as those possible with the human figure, therefore binding optically-captured data to the particular physiognomy of the performer.

In addition, there are some particular behaviors of facial skin, such as furrowing and wrinkling, whose fine geometric scale makes them extremely difficult to capture; yet they are extremely significant for the perception of facial expression. Some authors try to reproduce them through mechanical simulations of the tissue [LTW95, KGB98, KHS01]; however the elastic properties of the skin, together with the small scale at which it has to be sampled to represent such features, yield a highly stiff set of differential equations whose numerical solution is on its own a rather delicate problem (c.f. [Lan99, GW04]). Furthermore, numerical schemes able to guarantee convergence either require extremely fine time-stepping, or have computational complexities worse than quadratic on the density of the sampling, making them not efficiently implementable in real-time.

The aim of this thesis is to overcome the limitations of present frameworks by provid-

ing tools to enable extensive use of performance-based techniques for the animation of facial skin, thereby benefiting from the significant perceptual realism that these methods can provide. Such tools are integrated into a Facial Animation system whose synthetic part exhibits a linear degree of computational complexity (with respect to the sampling density of the facial representation), allowing its application in real-time, interactive contexts.

1.1 Anatomy of Facial Animation systems

From a functional point of view, the operation of a Facial Animation system can be divided into the following stages:

1. *Observation*: retrieving a body of information from actual facial motion,

2. *Abstraction*: extrapolating a set of representative characteristics from the observed data that can be later recombined in order to produce a synthetic representation of an expressive face, and

3. *Synthesis*: realising such representation as a visual model (e.g. an animated polygonal mesh).

Out of these three stages, abstraction turns out to be the most important for the system's design, as it carries along the assumption of a particular way in which to model facial expression. This encompasses all aspects of facial motion that are relevant to human perception, focusing mainly on those responsible for conveying speech and emotion, but including as well the ones that reflect the individuality of the performer (e.g. personality, age, etc). Such a model of expression will invariably determine the focus of observation, as well as the kind of control input governing the algorithms in charge of visual synthesis, hence its importance.

On the other hand, the choice of a model of facial expression is limited by the resources available to the observation and synthesis stages. In practice, this means that our process of abstraction has to be built upon elements of facial motion that can be easily quantified during observation (such as geometric or bio-mechanic measurements) instead of using their subjective representation in the human psyche (as figures of speech and emotion). Relating facial expression to such concrete elements will also facilitate the role of the synthesis stage, as the information they provide is closer to the actual appearance of the face.

However, this move towards concreteness also implies that the descriptions given by such models of expression will be bound, up to some extent, to the specific faces data is collected from (as it occurs, for instance, with marker-based MoCap). Should our animation system deal with different physiognomies (either during synthesis or observation), it would need to consider additional methods of generalization.

Realism, understood as the correspondence of the system's results with actual facial motion, can only originate from observed information, and gets invariably degraded by the successive steps of abstraction. Constructing a system that produces plausible results means that we need to be able to isolate the perceptually important factors

of facial expression, and provide the tools capable of reintegrating them into the synthetic representation. Such a process involves conscious design choices affecting not only the modelling of facial expression, but also further elaboration of the observation and synthesis stages.

The work presented in this thesis will provide a comprehensive solution to these requirements and design choices, combined with the real-time considerations described in the introduction.

1.2   Structure of this report

This thesis is organized into the following chapters:

- *Chapter 2: Models of Facial Expression* presents a classification of the various abstraction paradigms adopted by Facial Animation systems (such as appearance-based models, bio-mechanic descriptions, etc.), and discusses their implications for the rest of the system's design, given special attention to the implementation of the observation stage.

- *Chapter 3: Geometric Deformation of Facial Skin* studies how facial expression can be reproduced with a synthetic facial model, by translating the descriptions of facial motion given by the corresponding model of expression into control input for deformation algorithms. The performance aspects of these techniques are also taken into consideration.

- *Chapter 4: Adapting Facial Motion Capture* analyses the intrinsic problems of optically-based Facial MoCap, and introduces a framework which allows the generalization of this data across various physiognomies, different from those used in the capture process.

- *Chapter 5: Synthetic Wrinkling* is dedicated to a particular aspect of facial motion: the fine-scale behaviour of the skin. Several performance-based methods are considered in order to reproduce the furrowing and wrinkling that the large-scale deformation algorithms discussed in Chapter 3 overlook. As in this Chapter, we give special attention to performance-related aspects.

- *Chapter 6: Hardware-accelerated Facial Animation* explores the implementation of the methods described in Chapters 3 and 5 with dedicated hardware (programmable graphics cards), which plays a key role for the construction of a real-time system.

- *Chapter 7: Conclusions and Future Work* discusses the results achieved with a system using the techniques introduced in the preceding chapters, and proposes potential follow-on research.

1.3   Main contributions

This thesis introduces a series of novel contributions to Facial Animation, which are:

- Two surface-driven deformation paradigms, *Planar Bones* and *Bézier-Induced Deformation of Surfaces* (BIDS), which provide a suitable framework for translating facial expression onto polygonal models representing a static face, whenever the corresponding expression model is based on the spatial motion of a cloud of marker points. These two methods are specifically covered in Sections 3.4 and 3.5, where special attention is paid to the practical aspects of their implementation, providing semi-automatic solutions for all the manual processes involved. Additionally, the development of hardware-accelerated versions of these techniques is tackled in Sections 6.3 and 6.4. Planar Bones has been published in [SM03], while the formulation of BIDS appears as part of both [SEM04] and [ESM04b].

- A novel retargeting framework for the expressions retrieved with marker-based Motion Capture systems, which makes it possible to translate the results produced with a particular performer to a wider range of facial physiognomies. This operates by building a mapping between the embeddings of homotopic control structures fitted onto each of the faces, and evaluating it with the marker point displacements occurring on the captured data. By doing so, the anatomical differences between both models are represented in the scale and direction of the retargeted motion. Complete details are given in Section 4.3, and have been published in [SEK03].

- Two different approaches to the synthesis of expressive wrinkles and furrows, complementing the large scale deformation performed by Planar Bones and BIDS. Both these methods are related to measures of mechanical deformation of the tissue, and in the case corresponding to BIDS, this is treated as part of a performance-based framework integrating both marker-based Motion Capture and shape-from-shading techniques. Further details are given in Sections 5.2 and 5.3, and compiled in [SEM04].

2. MODELS OF FACIAL EXPRESSION

2.1 Introduction

Subconsciously, the human perception of facial motion is interpreted as elements of speech and emotion, together with the subjective connotations added by their particular realisation on the perceived subject (e.g. age, cunning, unease). These terms alone define a way in which to model facial expression, but they are far too abstract to be used by any practical implementation of a Facial Animation system. Instead, actual systems require more tangible elements to focus on, which can be later related to algorithms operating on a visual representation of the face. We will refer to the models of expression constructed in this manner as *low-level* or *concrete*, in contraposition to the *abstract* descriptions resulting from human perception.

The domain of information spanned by the observable aspects of facial motion is so wide that it becomes impossible to build a model of expression relating to all its elements (think, for example, of the movement of each bristle of facial hair). In practice, we have to apply some kind of reductionism in order to produce a more efficient and compact representation. Depending on the nature of the elements selected by this process, concrete models of expression can be categorised as:

- *Anatomically-based descriptions:* using a simplification of the anatomic system that produces facial motion (the muscles underlying facial tissue and driving the jaw), expressions are described as a series of muscular contractions.

- *Appearance-based descriptions:*
  - *Temporally discrete:* the full graphical representation of the face is known for a set of static poses representing particular gestures or vocal articulations. This information is used to describe facial expression as a weighted combination of the elements of such a set, with weights varying over time.
  - *Spatially discrete:* conversely, facial motion is known throughout the whole animation sequence, but only for specific features of facial geometry (e.g. contours or isolated points); the rest of the graphical representation is modelled upon them.

The boundaries set by this classification are not to be taken as strict barriers, for there are models that transgress them, either by illustrating muscular actions with their visual substantiation, or by using anatomic principles to provide additional structure to the raw appearance data collected during observation.

Later in this Chapter we will dissect the major contributions in the field according to this particular taxonomy; however, such classification alone would not provide sufficient criteria for a proper evaluation. Thereby, the next section will define a set of scales that we will refer to in each individual study.

2.2 Adequacy of expression models

As discussed in the introduction, choosing the most suitable model of expression for a particular Facial Animation system depends largely on the circumstances surrounding its application (e.g. method of visual representation, capture resources available, etc). This makes it rather difficult to compare different approaches outside of a common context; however, it is possible to define some general criteria that are applicable in any situation:

- *Data requirements:* the amount of information needed to construct a model of expression and use it in reproducing facial motion is a crucial factor for the animation system it supports. This is also extensive to the complexity of the observation processes involved and the accuracy of their results.

- *Completeness:* any model shall be able to describe the full range of facial expression that its corresponding animation system tries to convey (e.g. all possible phoneme utterances for a visual speech system, every representative eyebrow shape for a system portraying emotion, etc).

- *Adaptability:* another important factor to take into account is the ability of the model to translate a description of facial expression across different physiognomies. This becomes especially relevant whenever facial motion is explicitly encoded with respect to a reference subject (as it occurs with most appearance-based approaches).

- *Realism:* as a pivotal component of a Facial Animation system, its model of expression can make a significant contribution towards producing plausible results. Specifically:

  – by providing a description that can be accurately related to the visual appearance of the face, through the opportune synthesis algorithms (mostly geometric deformation techniques), and
  – by performing an abstraction that does not discard any significant feature of facial motion (e.g. expressive wrinkling, shaping of eye contours, etc).

While there is not a particular order in importance for these four aspects, on most occasions data requirements are considered first by the system design, which will later assign more or less importance to the remaining three. We will now proceed to illustrate this with specific examples.
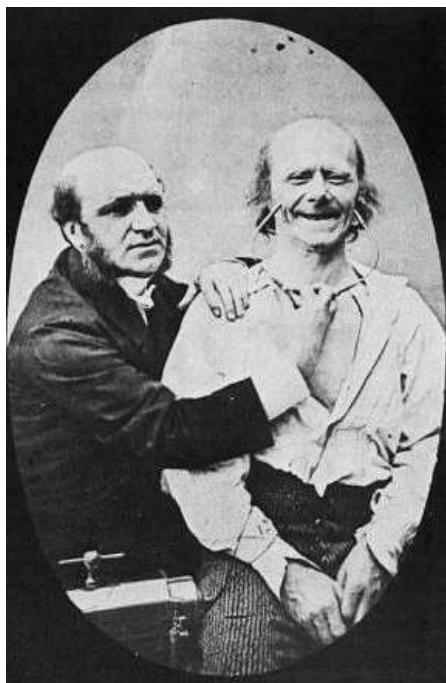
Fig. 2.1: One of the many pictures illustrating the work of Guillaume B. Duchenne MD, experimenting on a patient with facial paralysis. Photo courtesy of Eugenii Katz

2.3 Anatomically-based descriptions

A vast amount of work has been produced about the anatomic processes involved in facial motion, starting long before its application to computer-generated animation could be even imagined. The pioneering works of Duchenne [Duc62] in the late 19th century contributed an extensive functional description of the facial muscles, together with a study of their relation to human expression (particularly smiles). What was originally thought of as a rather eccentric piece of research came to the attention of the Psychology community, when in the 20th century perceptual implications of facial motion underwent intensive scrutiny. As a result a new study, whose aim was to categorise the full range of facial expression, was produced: the Facial Action Coding Scheme (FACS). This was to become a de-facto standard for many diverse applications, including Facial Animation.

2.3.1 Facial Action Coding Scheme

In 1969 the psychologists P. Ekman and W.V. Friesen conducted a series of experiments with the purpose of cataloguing the effects of muscular contraction on the appearance of the human face. By examining facial performances and relating them to the underlying anatomy, they identified 58 minimal actions (so-called Action Units - c.f. Table 2.1) whose combined effects are thought to produce the full range of facial motion which is

Tab. 2.1: Examples of Facial Action Units, relating perceptual elements of the facial motion to the muscular basis that produces them, thus providing an anatomically-based description.

Action Unit	FACS designation	Muscular basis
. . .		
AU10	Upper lip raiser	Levator labii superioris, caput infraorbitalis
AU11	Nasolabial furrow deepener	Zygomatic major
AU12	Lip corner puller	Zygomatic minor
AU13	Sharp lip puller	Caninus
AU14	Dimpler	Buccinator
AU15	Lip corner depressor	Triangularis
AU16	Lower lip depressor	Depressor labii
AU17	Chin raiser	Mentalis
AU18	Lip puckerer	Incisivii labii superioris, incisivii labii inferioris
AU19	Tongue out	-
. . .		

perceptually relevant. This collection was originally specified in the form of a printed manual documented with videos [EF78], and has since seen several updates, the latest in 2002.

Despite the fact that this scheme was originally meant as a reference guide for human scorers to categorise facial behaviour, it also proved to be a useful tool for Facial Animation systems, especially for those pursuing realism by reproducing the actual phenomena behind facial motion. The contribution of FACS to anatomically-based systems is an exhaustive decomposition of abstract figures of speech and emotion into a simpler set of 58 Action Units (AUs). While AU measurements are not exactly mechanical (let's say, $Newtons/m^2$), they indicate which muscles or muscle groups become active, and to some extent the degree of contraction involved. This helps narrowing the gap between *abstract* and *low-level* descriptions of facial motion, providing much of the data necessary to build a top-down model of expression.

However, the decomposition into AUs outlined above is both the main strength and major weakness of FACS. Combining atomic muscular actions may allow the representation of a wide range of motion, but the full intensity of muscular contraction does not always reflect on the appearance of the face. Whenever several units are active, the connectivity of facial tissue causes their effects to mutually interfere, resulting in a combination that is non-linear. Since FACS builds its classification on visual evaluation, it remains impervious to this particular phenomenon, and it is up to the system constructed atop to compensate for it. Otherwise the realism of the obtained results would be seriously compromised.

The adaptability of this scheme relies on being able to identify in the target face the muscular units that AUs refer to. This is exclusively an exercise of anatomy and, despite falling outside the usual skills of animators, perfectly solvable. Nevertheless other anatomical aspects that influence the expressive process, like the shape of the

skull or the elastic properties of the skin, are not explicitly considered, and are left for different stages of the system.

The early introduction of the Facial Action Coding Scheme made it a key reference for some of the initial works in computer animation. In fact the work by Platt and Badler in [PB81], a pioneer of anatomically-based systems, makes extensive use of it. In this case facial muscles are modelled as simple linear fibres, attached to isolated points on the skin surface. Waters [Wat87] also took into consideration complex muscular structures, such as *sphincters* or *planar sheets*, allowing for a more accurate anatomic model. An identical approach was also taken by Patel and Willis in [Pat92].

On the other hand, many muscle-based systems do not embrace the AU abstraction; however, if their anatomic bases are compatible with that of FACS, this coding scheme can be overlaid on top on these systems allowing for high-level parameterisations to be used as their control input. That is the case for the work of Lee et al. in [LTW95], where muscles are treated as non-linear Hookean fibres. This model is almost equivalent to that presented by Platt and Badler in [PB81], apart from introducing piecewise concatenation of muscular segments; therefore it can be controlled through AUs in an equivalent manner.

Further research continued to enrich the abstraction of muscles, and thereby increased the realism of the resulting models; Wu et al. [WMT94] originally considered using solids of revolution to represent their shape, and later in [WKM99] they adopted more generic B-Spline surfaces. This enabled their system to represent the interaction between the dermis and the underlying muscular surface, rather than limiting it to specific points of attachment. In addition, the approach taken by Kahler et al. in [KHS01] allowed the construction of any imaginable fibre topology by annealing elastic ellipsoids, and at the same time it enabled representing muscle dilation during contraction. This property is also reproduced by the use of parameterisable sweeping primitives in [SPC97], but such an approach imposes more limitations on the shaping of muscles.

2.3.2 Abstract Muscle Action procedures

As commented before, FACS was not conceived with computer animation in mind, and while in the academic world it has been thoroughly applied to demonstrate the completeness of Facial Animation systems, its formulation is somehow inadequate for some CG production contexts. Precisely, it was during the development of a production framework for character animation that researchers at MIRALab grew aware of this problem and designed an alternative model of facial expression: Abstract Muscle Action (AMA) procedures [MPT88]. This approach uses a coarser abstraction than AUs, considering as atomic actions that are produced by intersecting sets of muscles (AMAs - see Table 2.2). Such a choice breaks the parameter independence present in FACS, and consequently the order of application of AMAs becomes relevant to the result.

Despite the introduction of order dependence, this paradigm claims to be much more intuitive than FACS for the artists who manually produce control input for animation, thus easing data requirements. This criterion on its own is far too specific to define AMAs as a generic improvement over AUs, especially since there is no psychological study backing them. However it is not innovation which makes this paradigm important,

Tab. 2.2: Examples of Abstract Muscle Action (AMA) procedures, where the anatomic component of FACS is diluted into a more appearance-based framework.

AMA no.	Procedure name	Range
. . .		
10	Right_Lip_Raiser	0 to 1
11	Left_Zygomatic	0 to 1
12	Right_Zygomatic	0 to 1
13	Move_Right_Eye_Horizontal	-1 to 1
14	Move_Right_Eye_Vertical	-1 to 1
15	Move_Left_Eye_Horizontal	-1 to 1
16	Move_Left_Eye_Vertical	-1 to 1
17	Right_Risorius	0 to 1
18	Left_Risorius	0 to 1
19	Move_Right_Eyebrow	-1 to 1
. . .		

but the role it plays in the evolution of models of expression; AMAs have anatomic foundations, however some of them are defined not in terms of the muscles involved, but on the external appearance of specific regions of the face. In fact, this paradigm turns out to be a precursor of some of the appearance-based models of expression introduced in the next section.

2.4   Appearance-based descriptions

Systems modelling facial expression based on appearance try to reproduce realism by retrieving motion and visual clues directly from facial performances, rather than simulating the anatomic processes that produce them. This approach relies on the fact that no matter what paradigm is used to drive skin deformation, what ultimately conveys expression are its visual results. Furthermore, the closer the data we have is to the actual appearance of the face, the less realism will be lost in intermediate synthetic layers.

On the other hand, appearance-based models of expression bring in a series of problems unprecedented in anatomical approaches; adaptation has to operate on purely geometric principles, due to the lack of a functional description common to all faces (such as a muscular system). As a result, the realism of the results can be compromised whenever the system uses data captured from a physiognomy significantly different from that being animated (Chapter 4 elaborates further on this topic). Also, the limited temporal or spatial domain of motion data (if not both) requires the use of interpolative techniques to reconstruct the full deformation of facial skin, and that is difficult to achieve in a plausible way (a more detailed discussion can be found in Chapter 3).

In order to describe how particular approaches tackle these difficulties, we will continue with the categorisation started in section 2.1.

Tab. 2.3: Examples of MPEG-4 Facial Animation Parameters (FAPs), relating facial expression to the motion of FDPs (see Figure 2.2) in terms of FAP Units (see Figure 2.3).

FAP no.	FAP name	Units	Direction	FDP	min. range	max. range
...						
10	raise_b_lip_lm	MNS0	up	2.8	+-1860	+-600
11	raise_b_lip_rm	MNS0	up	2.9	+-1860	+-600
12	raise_l_cornerlip	MNS0	up	2.4	+-600	+-180
13	raise_r_cornerlip	MNS0	up	2.5	+-600	+-180
14	thrust_jaw	MNS0	forward	2.1	+600	+180
15	shift_jaw	MW0	right	2.1	+-1080	+-360
16	push_b_lip	MNS0	forward	2.3	+-1080	+-360
17	push_t_lip	MNS0	forward	2.2	+-1080	+-360
18	depress_chin	MNS0	up	2.10	+-420	+-180
19	close_t_l_eyelid	IRISD0	down	3.1	+-1080	+-600
...						

2.4.1   Temporally discrete variants

Parke's seminal work in Computer Facial Animation [Par74] modelled expression as the dominance of a set of static facial poses, varying over time. Whenever these poses are chosen to represent recognizable elements of speech and emotion, the model provides both a high and low-level description of facial expression, as it occurred with FACS. However, in order to match the completeness of Ekman and Friesen's approach the present system needs to capture a large amount of poses, thereby inflating its data requirements.

Further research [BCS97, PSS99] evolved this concept by using anatomic criteria to classify different facial regions according to the muscle groups active in each of them. This meant that facial motion could be expressed using a much smaller set of poses representing localized deformation. Recent work by Joshi et al. [JTD03] has turned manual region mark-up into an automatic process, by analysing the correlation of captured motion amidst neighbouring points of the facial geometry. The resulting Facial Animation framework is remarkably self-contained, making this technique the common practice in most production applications. However, there is not a standard choice of poses reassuringly capturing the complete range of facial expression; to address this need, the Motion Picture Experts Group (MPEG) included a series of recommendations in the fourth release of its popular standards, that we now proceed to describe.

MPEG-4 Facial Animation Tables

Following the introduction of AMA procedures (c.f. Section 2.3), and the successive description of facial motion in terms of the equivalent Minimal Perceptible Actions (MPAs - see [KMM92]), the MIRALab group, in cooperation with other European researchers, contributed the body of Facial Animation in the MPEG-4 standard [MPE01]. The main

purpose of this compendium is to define a common framework for compression and transmission of multiple kinds of visual data, and to that effect a high level parameterisation of facial expression is required. MPEG-4 Facial Animation Parameters (FAPs) include two layers of abstraction. The high-level description encompasses 14 visemes and 6 different emotions, tabulated over a lower-level model of expression. Such low-level description is strongly influenced by the heritage of AMA procedures and MPAs, as it is slightly based on anatomic aspects, but mainly ruled by appearance criteria (c.f. Table 2.3 for some examples).

Since this standard is only concerned with the encoding of data, it leaves open to implementators the actual design of a concrete model based on its choice of parameterisation. Focusing on this particular problem, Garchery and Thalmann [GM01] proposed the use of Facial Animation Tables (FATs), which relate the MPEG-4 FAPs to artist-designed poses, as well as encoding other information related to skin deformation. FATs are actually an adaptation of Parke's original work [Par74], and do not particularly benefit from any of the subsequent contributions described earlier. Consequently, the data requirements of FATs are rather exhaustive, and in addition, their realism is compromised by the non-strict orthogonality of FAPs (this prevents reproducing small scale deformation because of the mutual interference of different FAP contributions). To compensate for this, some high-level expressions have to be introduced disregarding the mapping between high and low-level parameters (inserting them as additional poses controlled through high-level FAPs alone). This increases even further the data demands. Fortunately MPEG-4 also allows for more flexible solutions, as the next section will point out.

2.4.2   Spatially discrete variants

Spatially discrete models of expression were first introduced by Williams in [Wil90], where he used marker points to track the continuous motion of specific facial features. This approach, that received the generic name of *Performance-driven Facial Animation*, is also common to [GGW98, TH98, JTD03], amongst many others. Additionally, some researchers have proposed the use of parametric curves for this purpose [LTB96, SMK97], tracing some representative contours of the facial anatomy (e.g. eye and mouth openings, brow arch).

In general, Performance-driven Facial Animation (PDFA) systems provide a straightforward paradigm for reproducing facial motion directly from capture, without having to consider the biomechanical aspects that support it. However, synthesis of new motion is far more complicated, and cannot be dealt with without a large database of captured segments (c.f. [SEK03]). Consequently the data requirements involved in modelling facial expression in this manner cannot be considered with disregard of its context of application. However, for most production purposes they turn out not be an issue, since marker-based Motion Capture systems are widely available and mostly self-contained.

As with temporally discrete models, there are many arbitrary choices to be made in PDFA systems, mainly concerning the placement and density of markers on the performer's face. Ideally, these have to be sparsely distributed in order to recollect the different kinds of motion originated by independent muscle groups, and at the same time
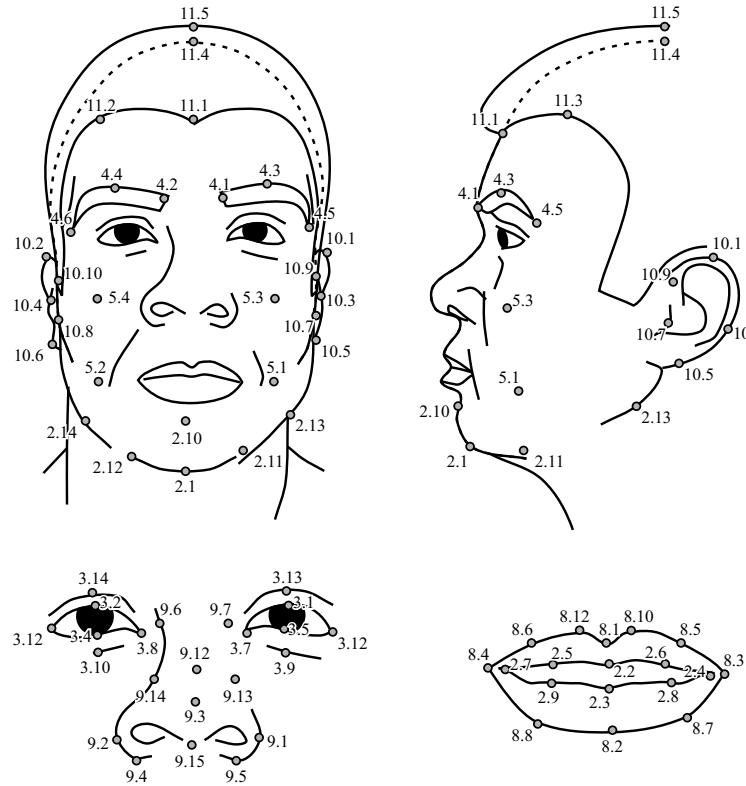
Fig. 2.2: MPEG-4 Facial Description Parameters (only those referred to facial skin).

concentrate on areas where the interaction with the overlaying tissue is more complex. Other restrictions, related to the vision approach taken for the capture, also apply (e.g. markers close to the eyelids may be obscured from certain points of view, complicating the 3D reconstruction of their movement). As a Facial Animation framework supporting PDFA, MPEG-4 dedicates part of its definition to marker-based MoCap; more details are given below.

MPEG-4 FDP Feature Points

In order to represent the animation of different facial physiognomies, MPEG-4 introduces the so-called Facial Description Parameters (FDPs), representing both the shape of the face and the scale of local motion. The first of these two aspects is described by the placement of a predefined set of Feature Points (FPs), which are also used as markers to track facial motion; Figure 2.2 illustrates their complete layout.

As for the scaling of motion, low-level FAPs relate to the displacement of FPs in terms of in terms of Facial Animation Parameter Units (FAPUs, see Figure 2.3), thereby performing an implicit adaptation to each particular physiognomy. While this process means a significant advantage over the use of FATs (as we don't have to recapture/model
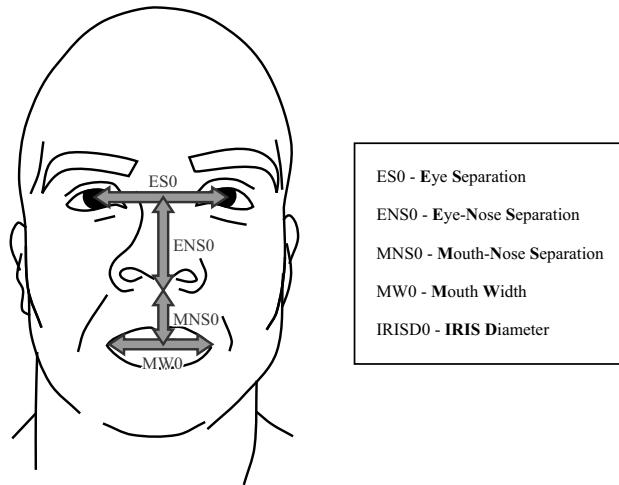
Fig. 2.3: MPEG-4 Facial Animation Parameter Units, used as basis for its approach to
facial motion retargeting.

every expression), it is actually a plain linear mapping, and the domain constructed with
FAPU measurements is far too simple to represent the anatomic differences between any
two faces. These limitations prevent MPEG-4 based systems from producing results as
plausible as other approaches to translating facial motion. Chapter 4 discusses this
particular topic in more detail.

Furthermore, the proposed mapping from FAPs to Feature Point movement is, apart
from high-level FAPs and jaw rotation, a one-to-one correspondence. This makes Facial
Animation Parameters somehow redundant, as they do not exploit the correlation of
motion amidst different sets of markers, and consequently do not bring in any additional
abstraction with respect to the bare tracking of the FPs. However, what they do add
is a set of numerical ranges useful for their quantization according to the usual MPEG
compression process, and data compression is precisely the main focus of this standard.

2.5   Summary and discussion

Recapitulating sections 2.4.1 and 2.4.2, and using the criteria from Section 2.2, we can
conclude that:

- Models of expression based on appearance prove to be rather demanding in terms of
  the data required to build them, needing to capture or manually produce significant
  amounts of information. On the other hand, systems based on anatomical models
  are more self-contained, since the behaviour of facial muscles is a well studied
  phenomenon.

- In an anatomic context, completeness implies having a full catalogue of all percep-
  tible muscular actions or, equivalently, an anatomical model complex enough to

represent them all. On the other hand, the range of expressions that can be modelled using appearance-based techniques depends on the diversity of data captured and/or manually produced.

- Systems using an anatomical approach achieve adaptability by identifying equivalent functional elements in different faces (muscles / muscular groups). However, such a solution is not extensible to appearance-based counterparts since the anatomic referent is absent, leading to more sensitive, purely geometric alternatives.

- While anatomically-based approaches claim that they attain the most realism by mimicking natural phenomena, expression models based on appearance give themselves equal credit for retaining the visual plausibility captured from facial performances. None of these claims can be verified without considering as well the deformation method used to substantiate a particular model of expression onto the graphical representation of the face.

This last paragraph serves as motivation for Chapter 3, precisily dedicated to the study of deformation techniques applied to Facial Animation.

3. GEOMETRIC DEFORMATION OF FACIAL SKIN

3.1 Introduction

Chapter 2 presented methods of modelling facial expression concrete enough to be used as control input for deformation techniques that change the representation of the face. Whenever such representation is of a three-dimensional kind (e.g. polygonal meshes), the applicable deformation techniques will necessarily consist of geometric operations. The nature of these operations varies significantly according to the deformation paradigm chosen, as well as to the expression model itself. This chapter is dedicated to exploring these relations, as well as proposing new techniques that contribute more realistic results to some of the existing frameworks.

Deformation paradigms applied to Facial Animation can be classified according to the abstraction of facial skin on which they base their operations:

- Physically-based approaches: facial tissue is considered as an elastic continuum, whose mechanical properties are simulated on a discrete representation of the face.

- Purely geometric approaches: the skin is regarded just as a geometric entity, where generic warping methods are applied, in general without any mechanical considerations.

There is a close relationship linking the use of models of expression based on anatomic criteria to the application of physically-based deformation techniques, as well as between appearance-based descriptions and purely geometric warps. However none of these connections are actually binding, since it is perfectly possible to translate any kind of expression model into control input for any other type of deformation, as later sections of this Chapter will illustrate with specific examples.

While in Section 2.2 concrete models of expression were judged upon a series of context-independent criteria, deformation algorithms in Facial Animation are just a supporting element for these expression models, and it is mainly in that role that they will be judged as adequate. However, there are additional factors (e.g. computational efficiency) which will definitely contribute to the value of the system, and therefore have to be taken into account by any evaluation. Additional details will be given alongside the description of the two deformation categories enumerated above, contained in Sections 3.2 and 3.3.

Furthermore, Sections 3.4 and 3.5 will present new additions to geometric deformation techniques (Planar Bones and Bézier-Induced Deformation of Surfaces, respectively), developed as part of the work presented in this thesis.

3.2 Physically-based deformation paradigms

The application of physically-based approaches to Facial Animation has closely followed the developments in simulation of deformable objects, on its own a rather popular trend in Computer Graphics. This relation has led to notable advances in the field, providing significant levels of realism by allowing approximating the behaviour of the human face with physical models. However this is also the reason for many of the weaknesses of this approach, as the generic models applied were originally conceived for highly regular, isotropically deformable materials (e.g. cloth, homogenous fluids), and that is not by far the case for facial tissue.

Similarly to the cloth case, the deformation sustained by facial skin is mainly planar, despite being connected to thick layers of muscular and fatty tissue. As is well known for computational models of cloth [VCM95, BW98], representing the planar incompress-ibility of the material against a larger tolerance for other types of deformation (bending and shearing) is on its own a numerically sensitive differential problem, presenting the distinguishable *stiff* condition [Lan99, GW04] that complicates the application of any solving scheme. Avoiding this numerical ill-conditioning requires operating at a rather coarse mesh resolution, and tweaking the elastic properties of the medium to account only for large-scale deformation, while ignoring the effects of additional compression inside the discrete regions.

This section distinguishes two kinds of physically-based approaches: those focused on simulating large-scale deformation, presenting the solution on a coarse discretisation of the face (polygonal mesh), and those using mechanical principles to extrapolate the fine-scale local deformation overseen by such discretisation.

3.2.1 Large-scale simulations

In 1981, Platt and Badler presented an early version of a fully physically-based Facial Animation framework [PB81], driven by an anatomic model controlled by FACS (c.f. Section 2.3.1). Motion is originated by muscle traction at predefined anchor points and transmitted along the connectivity of the tissue. Such transmission is simulated by itera-tive evaluation of a network of linear Hookean springs defined by the topology of the skin mesh, until a stable configuration is reached. On the other hand important anatomical issues, such as the influence of the bone structure or proper muscle transversal bulging, aren't handled by this early contribution. Neither does it define a numerical scheme for evaluating the dynamical system depicted. However it is undeniably an important foundation for successive research.

Subsequently, Terzopoulos et al. provided a far more complete simulation model for elastically deformable bodies [TPB87, TF88], taking as starting point the Lagrange equation of motion:

$$\frac{\partial}{\partial t}(\mu\frac{\partial r}{\partial t}) + \gamma\frac{\partial r}{\partial t} + \frac{\delta\varepsilon(r)}{\delta r} = f(r, t) \tag{3.1}$$

where $r(a, t)$ is the position of particle a at time t, $\mu(a)$ is the mass density of the body at a, $\gamma(a)$ is the damping density, and $f(r, t)$ represents the net externally applied

forces. $\varepsilon(r)$ is a functional (a function of functions) which measures the net instantaneous potential energy of the elastic deformation of the body.

The expression for potential energy is decomposed into terms of planar strain and bending, using the surface metric and the curvature tensors G and B, respectively, to describe such magnitudes:

$$\varepsilon(r) = \int_\Omega \|G - G_0\|_\alpha^2 + \|B - B_0\|_\beta^2 \; da_1 da_2 \qquad (3.2)$$

where $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are weighted matrix norms and G_0 and B_0 are respectively the rest state values for metric and curvature tensors. Their tendency to remain stationary is represented by the variational derivative $\frac{\delta\varepsilon}{\delta r}$, that can be computed through the second degree form of the Euler-Lagrange equation [AW00]:

$$
\begin{aligned}
\frac{\delta\varepsilon}{\delta r} &= \frac{\delta}{\delta r} \int_\Omega E \; da_1 da_2 \\
&= \sum_{j=\{1,2\}} \left(\frac{\partial E}{\partial r} - \frac{d}{da_i}\frac{\partial E}{\partial r_{,a_i}} + \frac{d^2}{da_i^2}\frac{\partial E}{\partial r_{,a_i,a_j}} \right) \\
&\cong \sum_{j=\{1,2\}} \left(-\frac{d}{da_i}\left(\alpha_{ij}\frac{\partial r}{\partial a_j}\right) + \frac{d^2}{da_i da_j}\left(\beta_{ij}\frac{\partial^2 r}{\partial a_i \partial a_j}\right) \right)
\end{aligned}
\qquad (3.3)
$$

Matrix norm weights α_{ij} and β_{ij} are taken to be constant along the deformed body. Therefore, by substituting Equation 3.3 into Equation 3.1 we obtain a set of quartic-degree linear differential equations. These are to be evaluated over a homogeneously distributed sampling of the surface, and solved numerically by an implicit scheme resulting from approximating the involved derivatives with finite differences. The implicit nature of this numerical scheme means that we have to solve a large sparse linear system compressing all nodes (geometry samples), which is computationally expensive for every step of simulation, but also guarantees good stability (in comparison with explicit approaches such as [PB81]).

We must note that this deformation model is given independently of discretization, and only relies on it to perform a numerical evaluation of the differential and variational equations involved. Indeed it applies to the continuum itself, contrary to the methods used in [PB81]. Celniker and Gossard make use of this fact in their *ShapeWright* modelling framework [CG91], which will be discussed later in this section.

Subsequent work by Platt and Barr in [PB88] considers as well the application of constraints to the dynamics of elastic models, enhancing the traditional Lagrange multipliers approach [Boa83] by combining it with the penalty method, resulting in the so-called *Augmented Lagrangian Constraints*. They also introduce simulation of plastic deformations modifying the rest state, by varying the correspondent terms in the energy functional. Bending deformation energy is not explicitly considered, however the solving schemes applied can be easily extended to include it.

Further developments in elastic deformation models can be found in the research presented in [BH94a], [BH94b] and subsequently used in [HDB96] and [Dev97]; in these
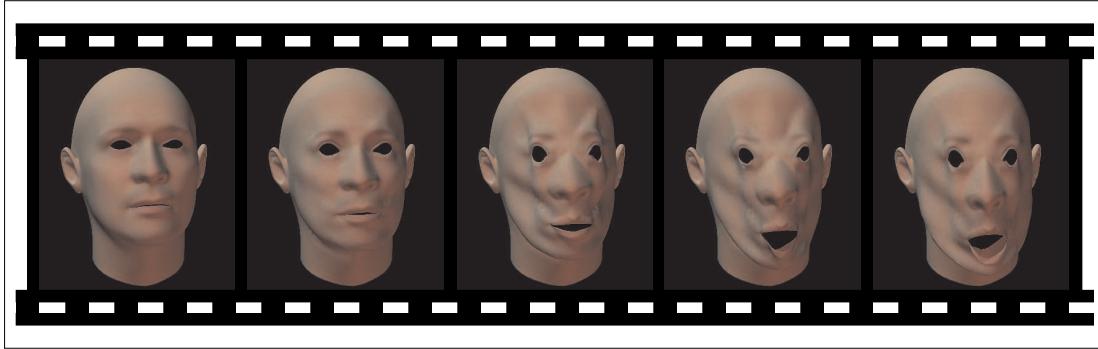
Fig. 3.1: Melting flesh: grotesque simulation of the skin coming loose on a human face. The elastic model in Equation 3.4 is discretized using the approximation described in Section 4.3.1, and integrated through a implicit Euler scheme (c.f. [BW98, BWK01, ES04]).

works the formulation of the strain and repulsive terms of the deformation energy introduced by Terzopoulos et al. is reworked, obtaining empirically better expressions. Also, a new term called *trellising* is added to the energy functional, resulting in a satisfactory representation of the resistance to surface twisting, which wasn't included in the bending term, defined only by surface curvature.

Despite its popularity, simulations based on the elastic model proposed by Terzopoulos et al. present, under certain conditions, a rather unrealistic behaviour: the formulation of the damping term causes the simulation to interpret rigid motion at constant speed as an elastic energy gain, resulting in unrealistic stiffness for the simulated body. Damping itself is a key factor in the simulation, not only for representing the physical phenomenon of the same name, but also for stabilising the numerical solution near points of convergence. Therefore it has to be included in a more consistent way. Opportunely Carignan et al. [CYM92] propose a reformulation of Terzopoulos et al.'s dynamic equations that overcomes this issue:

$$
\mu \frac{\partial^2 r}{\partial t^2} + \frac{\delta}{\delta r} \int_\Omega \|G - G_0\|_\alpha^2 \, da_1 da_2 + \frac{\delta}{\delta v} \int_\Omega \|\frac{d(G - G_0)}{dt}\|_\alpha^2 \, da_1 da_2 +
$$
$$
\frac{\delta}{\delta r} \int_\Omega \|B - B_0\|_\beta^2 \, da_1 da_2 = f(r, t) \tag{3.4}
$$

The damping term described by the second variational derivative in Equation 3.4 involves just the fluctuation of the metric tensor along time, representing the speed of deformation instead of the instantaneous speed of the medium itself, thus modelling more consistently the physical fact. Figure 3.1 shows the result of a practical implementation using a face model.

Lee et al. [LTW93, LTW95] apply a particle-wise simplification of the continuum physical model proposed in [TPB87] to anatomically-based Facial Animation. This

approach defines a discretised elastic model for facial tissue organized in three layers, considering not only the skin surface but also the fatty layer that separates it from the underlying muscle and its attachment to the bone. In contrast with [PB81], the mass-spring network spreads not only over the skin surface, but also along the inter-layer connections, resulting in a much more capable and complex simulation model. On the other hand, the implicit scheme proposed in [TPB87] is replaced by simpler explicit Euler integration, with a fixed time interval. Its application in this context, combined with the arbitrary topology of the mass-spring network constitute a potential source of numerical instability, requiring time-dependent attenuation of the nodal forces in order to guarantee convergence. These factors cause the simulation to produce facial configurations that would not be stable according to the definition of the mass-spring system, but are forced to be so as a consequence of the damping terms. Therefore a progressive approach, by applying consecutive deformations to a previously modified face, is not applicable under this paradigm, since its behaviour is not order-independent in terms of the deformations put forth, nor reversible.

The weaknesses observed in the aforementioned work by Lee et al. do not cast aside the whole mass-spring iterative approach, as there are other explicit numerical integration schemes which can be adopted (e.g. Runge-Kutta's or Verlet's, c.f. [PTF92, GW04]) in order to produce more stable results. This is the path followed by Kahler et al. in [KHS01], where the authors first compute a coarse approximation of the deformed skin using a geometric warp based in the displacement of muscular elements, and then feed it to the iterative process minimising the elastic energy.

In general, explicit integration schemes are rather inadequate for the resolution of the differential equations governing the dynamics of facial skin, due to the characteristic *numerical stiffness* of the problem. This requires reducing the simulation step size substantially in order to guarantee stability; such a numerical problem is thoroughly described in [PTF92], and as the authors of this work point out it can be successfully tackled by switching to implicit or semi-implicit approaches. The lack of stability affects as well the simulation of other much simpler incompressible mediums, like cloth (c.f. [HDB96]). This underlines the importance of tackling this aspect as a key milestone in reproducing the complicated nature of the human face.

Opportunely, the Finite Element Method (FEM - c.f. [Zie89]) provides stable and accurate solutions for deformation models applied to the elastic continuum, where the medium is actually approximated by a finite set of shape functions defined over discrete piecewise domains. This also means that the number of Degrees Of Freedom (DOFs) involved in FEM numerical solutions is larger than that in discrete models, making computations more expensive (albeit some DOFs can be determined by piecewise continuity conditions). On the other hand, the larger number of DOFs relaxes the *stiffness* of the associated differential equations, enabling more stable solving and coarser time-stepping. This outlines a compromise between efficiency and stability that most engineering applications are willing to take, but has deeper consequences in Computer Graphics (performance and memory requirements make real-time uses infeasible).

When working with Finite Elements the applicable constraints, together with the remaining DOFs, are implicitly determined by the degree of the shape functions. If

these functions are linear and no continuity conditioning is considered, the method becomes equivalent to the discrete elastic systems commented as before (apart from enforcing implicit solutions). This is exactly the case of the first approach to Facial Animation using FEM, described by Koch et al. in [KGC96]. The real aim of this work was to simulate facial tissue adapting itself to a reconfigured skull through facial surgery, by using a two-layered skin model interconnecting bone and dermis; in addition, this approach provides a convenient method for specifying the constraints on the deformed medium by tagging those areas that stay still, thereby producing boundary conditions for the numerical integration.

Subsequent research by Koch et al. [KGB98] used an identical simulation approach for anatomical expression models, describing the *Emotion Editing* framework. Unfortunately, the acchievements of this work in terms of numerical accuracy are not matched by its take on Facial Animation, as it focuses mainly on static gestures. Rather than producing animation using FEM, the authors propose precalculating the nodal displacements for every Action Unit (under the FACS paradigm) and linearly interpolating them to achieve the final result corresponding to a combination of AUs. This basically means falling back to methods equivalent to those from [Par74], which are not the most suitable solution, as it will be discussed in Section 3.3.3. The same method is used in a more recent work by Choe et al. [CLK01], with the only difference being the use of a more extensive muscle model, *à la* [Wat87].

Most of the Facial Animation research discussed up to this point refers exclusively to anatomically-based models of expression, as could be expected from approaches that pursue representing as accurately as possible the actual facial phenomena. However, some of the work in the aforementioned [LTW93], together with that in [PM01, CLK01], also explores the use of inverse physical models to retrieve muscular activity information from Motion Captured data, allowing for appearance-driven models to control physically-based deformations. This is done by tracking specific nodes of the mass-spring network, and then inverting the iterative process that produced their corresponding displacements. While Lee et al. [LTW93] take a temporally-discrete approach, estimating the muscular activity that produced a static pose, Pitermann and Munhall [PM01] consider as well node velocities to aid in an incremental estimation process, operating frame by frame. This last procedure suits more closely with the nature of spatially-discrete models, thus completing the mosaic of all possible combinations between physically-based deformation algorithms and models of facial expression.

3.2.2   Fine-scale local deformations

The systems described in the preceding section are able to represent the large-scale deformation of facial skin at the cost of simplifying local phenomena like wrinkling and buckling to plain elastic contraction. The great importance of these fine detail aspects in perceiving human emotion has led to significant research into their representation, some of which actually refer to the physical foundations of the mechanisms that produce them. This is precisely the case of the work in [WMT94], where the authors attempt to build a physically-based model of facial tissue capable of representing wrinkling by introducing visco-plastic terms into the equations of motion seen before. The accumulated effect of

plastic deformation reflects on the rest state of the skin, ultimately causing wrinkles to appear when the tissue buckles around the areas in hysteresis. For a linear element with initial rest length l^0 and strain $\varepsilon(t)$ we have:

$$l^0(t) = l^0 + \int_T k_v \frac{\varepsilon(t)}{dt} + k_p \varepsilon(t) \ dt \tag{3.5}$$

where k_p and k_v are, respectively, the coefficients of plasticity and viscosity of the medium. By performing a simple linearization:

$$l^0(t + \Delta t) = k_e l^0(t) + k_v \varepsilon(t) \Delta t + k_p \varepsilon(t) \tag{3.6}$$

Analogously, rest conditions in the energy functionals of Equation 3.4 can by substituted by:

$$
\begin{aligned}
G_{ij}^0(t + \Delta t) &= \alpha_e G_{ij}^0(t) + \alpha_v (G_{ij} - G_{ij}^0)\Delta t + \alpha_p G_{ij} \\
B_{ij}^0(t + \Delta t) &= \beta_e B_{ij}^0(t) + \beta_v (B_{ij} - B_{ij}^0)\Delta t + \beta_p B_{ij}
\end{aligned}
\tag{3.7}
$$

In the final expression resulting this substitution, viscous damping is included again (third term of Equation 3.4) as an expression of the rate of change of the metric tensor, thereby accounting for twice as much contribution. Nevertheless this does not result in any problem since the weighting coefficients for the energy functionals are estimated arbitrarily.

The paradigm presented by Wu et al. might be an accurate physical model for skin wrinkling, however it is far from practical, since the low level at which these phenomena occur makes them very difficult to represent in Computer Graphics. Also, in order to produce realistic results, the amount of information describing the mechanics of facial tissue would have to be vastly larger. Some of the authors have followed this particular trend down to the level in which it is actually feasible [MKL02], but the aim of Facial Animation has lead most of the subsequent research into producing more specialized procedures, using simpler physical abstractions and more appearance-based knowledge.

The work in both [WKM97] and [WKM99] meets exactly the description given by this last paragraph, as it shifts from using fully physically-based models to synthetic generalizations of skin wrinkling, yet is based on mechanical measures of deformation. Specifically, the authors represent the bulging between furrow lines as a linear expression of the transversal strain, thus modelling tissue incompressibility. This approach has two significant inconveniences: one is the need to manually specify the positioning of such furrow lines, and another is that this simplification doesn't necessarily work for all wrinkling structures, some of which involve many more furrow lines than just two. However this is a significant improvement over the rather similar framework proposed by Viaud et al. in [VY92], where far more stringent restrictions apply to the facial model. Some recent contributions [VM99, LC04] also share as well much of this paradigm, albeit not neccerarily in Facial Animation. We will discuss this in more detail in Chapter 5, which focuses on this particular field.
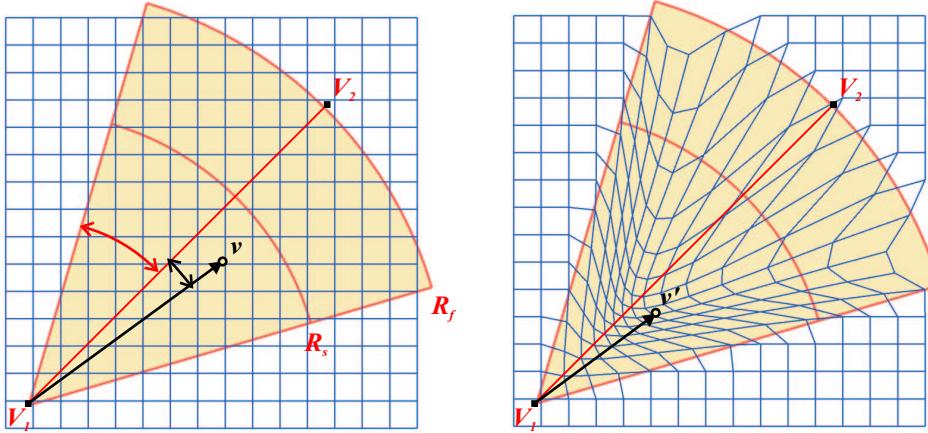
Fig. 3.2: Waters' linear pseudomuscle: point v on the skin surface is mapped to v' as the pseudomuscle contracts towards V_1.

3.3 Purely geometric deformation paradigms

The problems with physically-based methods for Facial Animation described in the preceding sections have bolstered the use of deformation techniques that exclusively exploit the geometric nature of facial tissue, rather than trying to model it as an elastic continuum. While physically-based approaches were mostly influenced by the anatomic paradigm of modelling facial expression, geometric techniques reflect the full variety of expression models, both anatomically and appearance-based. This also results in a wider range of deformation techniques put into practice while trying to better fit a particular kind of control input, making necessary a finer categorisation for their study. We will classify them in the following three groups:

3.3.1 Procedural warping

Procedural deformations, epitomized by the original work of Barr in [Bar84], have also been applied to Facial Animation as a straightforward way of simulating localized facial movements with specialized geometric procedures. The best-known example is the classic work by Waters [Wat87], where he uses a set of so-called *muscle functions* to mimic the effect of specific types of muscles on the skin surface.

Linear muscles (see Figure 3.2) are modelled by a function that pulls the vertices encompassed by its area of affection (a circular sector) towards its centre; for a given vertex $\mathbf{v}$:

$$\mathbf{v}' = \mathbf{v} + KR \frac{V_1 - \mathbf{v}}{\|V_1 - \mathbf{v}\|} \cos(a) \tag{3.8}$$
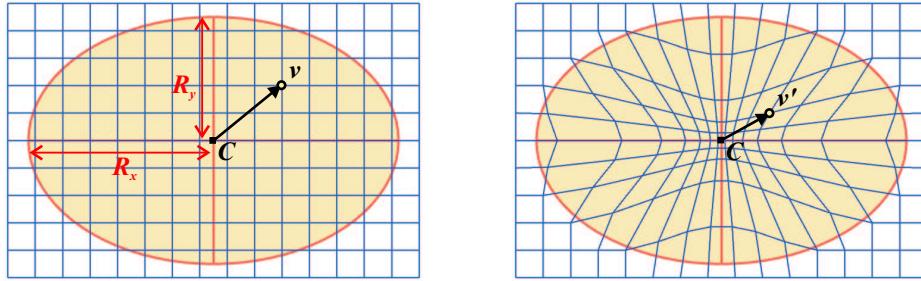
Fig. 3.3: Waters' elliptical pseudomuscle: point v on the skin surface is mapped to v' as the pseudomuscle contracts around C.

where

$$R = \begin{cases} \cos\left(\frac{\pi}{2}\frac{R_s - \|V_1 - \mathbf{v}\|}{R_s}\right) & \text{if } \|V_1 - \mathbf{v}\| \in [0, R_s) \\ \cos\left(\frac{\pi}{2}\frac{\|V_1 - \mathbf{v}\| - R_s}{R_f - R_s}\right) & \text{if } \|V_1 - \mathbf{v}\| \in [R_s, R_f) \end{cases}$$

In Equation 3.8 the constant K stands for the synthetic *elasticity* of the muscle, while a is the angular meassure given by the ratio $\frac{\pi}{2}(\mu/\Omega)$. Both μ and Ω angles are taken with respect to the muscle axis $\bar{V_1 V_2}$, and their use in the formulation guarantees that the displacement along the sector's boundary is identically null.

Conversely, synthetic sphincter muscles (Figure 3.3) define a full ellipse as the area of affection, and displace vertices in and out with respect to the centroid:

$$\mathbf{v}' = \mathbf{v} + K\cos(a)(C - \mathbf{v}) \text{ iff } a \in [0, 1) \tag{3.9}$$

where

$$a = \frac{\sqrt{(\mathbf{v}_x^2 R_y^2 + \mathbf{v}_y^2 R_x^2)}}{R_x R_y}$$

Both muscle models perform deformations along the plane; however formulations can be easily extended to operate on conical and ellipsoidal regions, respectively. Also, the effects of multiple pseudomuscles can be combined by a blending operation, where each vertex is weighted with respect to each deformer in a way that is proportional to its proximity to the pseudomuscle's point of attachment. This also enriches the three-dimensionality of the resulting warp, thus making it more realistic.

Equivalent approaches to that by Waters have been reported in [PW91] and more recently in [BBP01], both using the same planar muscle paradigm. Conversely, Wang [WL93] proposes muscle functions that operate following curved trajectories, thereby reproducing the behaviour of more complicated muscular structures in an otherwise rather limited anatomic framework. Combined with a parametric surface model, this method was also applied in the pioneering CGI short feature *Tin Toy*[1].

---

[1] 1988 Pixar `http://www.pixar.com/shorts/`
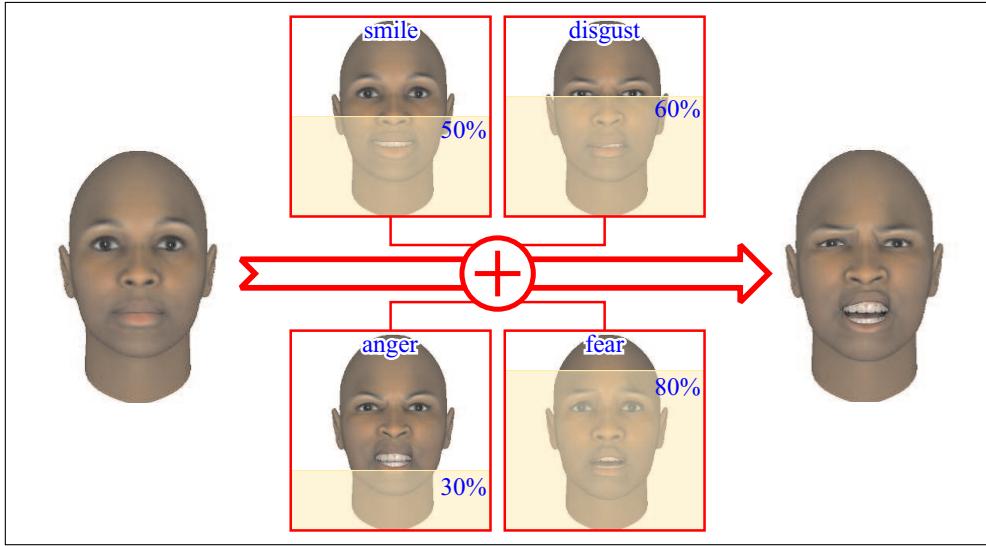
Fig. 3.4: Example of the application of blend shapes to contruct a combined expression. Morph targets have been produced with *SI FaceGen 3.0*.

3.3.2 Morphable models

The seminal work by Parke [Par74] expresses the deformation of a face model as the convex sum of weighted displacements measured for several static poses with respect to the neutral expression (see Figure 3.4). This approach is commonly known as *facial morphable models*, *blend shapes* or simply *morph targets*. For a set of facial poses $\{P_i\}$ mapping any vertex v_k to its posed image $P_j(v_k)$, the deformed image of v_k will be computed as:

$$v'_k = \sum_i \alpha_i P_i(v_k) + \left(1 - \sum_i \alpha_i\right) v_k \qquad (3.10)$$

The linearity of Equation 3.10 contrasts with the intricacy of facial motion, thus demanding a significant number of poses in order to be able to realistically represent the full range of facial expression. This is not only due to the limitations imposed by the implicit temporally-discrete, appearance-based model of expression, but also to the non-linearity of tissue motion, especially as skin slides over rigid structures such as bone and ocular globes. In part, these shortcomings can be avoided by also encoding the trajectories of vertices between poses, as Garchery and Thalmann propose in [GM01]. However this result is overwhelmingly costly in animation terms, and consequently impractical.

The control paradigm provided by temporally-discrete expression models particularly suits the traditional concept of animation, focused on emphasizing specific expressions at particular keyframes and blending in-between. This has made the marriage between this paradigm of facial expression and morphable models quite a popular combination in

the industrial field, being used in productions such as *Final Fantasy: the spirits within*[2] or *The Lord of the Rings* trilogy[3], as well as implemented in many of commercial Facial Animation solutions, for instance *BioVirtual's 3DMe Now!*[4] and *LIPSinc Ventriloquist*[5].

Further iterations of the technique proposed by Parke attempted to reduce the exceptionally large number of facial poses necessary to produce plausible Facial Animation, counted by the hundreds in cases such as the *Lord of the Rings' Gollum* character [For03]. Regional segmentation, as originally described by Pighin in [Pig99] and later extended in [JTD03], does so by using an anatomical basis to isolate the contribution of particular muscle groups and express complex poses as a combination of simpler, atomic ones. In [JTD03], the authors propose an automatic way of deriving this segmentation, by resorting to the Lamè formulation of linear elastic dynamics:

$$\rho \frac{\partial^2 r}{\partial t^2} = \lambda \Delta u - (\lambda + \mu) \nabla (\nabla \cdot u) \tag{3.11}$$

In this equation, u stands for the displacement function meassured at each vertex, and coefficients λ and μ are derived from the Young's modulus and the Poisson ratio of the material [Zie89]. Analogously to the equations in Section 3.2, the first (Laplacian) operand of the right hand term can be interpreted as the variation in linear deformation along the skin surface, while the second represents the force contribution by local area conservation. Joshi et al. consider the Laplacian term alone as a sufficient measure of skin warping, and by categorising facial regions according to the magnitude of this vector, averaged over the complete set of collected poses, they effectively isolate the most representative areas of deformation.

However, a study such as Joshi et al.'s still requires the existence of at least one whole domain of poses, and relies strictly on the ability of this set to accurately represent any characteristic facial expression. These requirements are overwhelming in terms of modelling, and demand vast amounts of artistic or capturing work to bring any of these systems into fruition.

Traditionally, the animation process via this specific paradigm has been an equally laborious task, involving manually keyframing the variation of pose weights while trying to fit the expression and speech of the character, in a manner appallingly reminiscent of classical animation approaches. Fortunately, further applications of morphable models were to ease these cumbersome routines, as they started being used for the analysis and recognition of faces from photographs, noticeably in the works by Cootes [CET98] and Blantz and Vetter [BV99]. This was later extended to video-recorded performances by Blantz and Vetter's work in [BV03], as well as used in conjunction with marker-based Motion Capture (MoCap) in the aforementioned [JTD03]. This last contribution actually builds a bridge between spatially-discrete and temporally-discrete models of expression, allowing to control blend-shape systems with much more tractable Facial MoCap. While highly non-automatic processes are still involved, such as the generation

---

[2] 2001 Square/Sony Pictures http://www.sonypictures.com/movies/finalfantasy/
[3] 2001-03 New Line Cinema http://www.lordoftherings.net/
[4] 2003 Biovirtual http://www.biovirtual.com/
[5] 1999 LIPSinc http://www.lipsinc.com/

of equivalent sets of poses for both performer and synthetic character, the improvements to the animation workflow brought forward by these contributions are outstanding.

Another problem intrinsic to the use of morphable models is how to represent the temporal correlation of facial gesture as a consequence of the mechanical properties of the supporting physiognomy. Representative phenomena such as vocal coarticulation [KM77, RPF97] tend to be ignored by the usual keyframing approaches, as this deformation technique forces vertices to linearly interpolate the keyed poses at constant intermediate speed, without regard for the previous history of deformation. This severely affects the perceptual realism of the resulting facial motion, with the skin moving at abruptly changing speeds without smooth acceleration or dampening. Recently, additional research in this field [EM04] has addressed this particular problem through a global optimization technique, constraining the general speed of the tissue and maintaining mechanical consistency during the animation.

Summarizing, morphable models are the longest lasting approach to skin deformation in the history of Facial Animation, and despite their many inconveniences, they are extremely well settled in the industry. Consequently, this technique gets the most academic research attention, as well as a significant dose of creative skill, coming from specific artist training favouring this paradigm of modelling facial expression. In a context where sheer artistic manpower has traditionally been a readily available commodity, with the reassurance that everything can be ultimately tweaked, it is unlikely that morphable models will soon be replaced in production workflows.

3.3.3   Motion interpolation

In contrast with the extensive modelling demands of morphable models, alternative geometric approaches try to reconstruct skin deformation in a more self-contained way, by extrapolating the motion of easily tractable facial features, either inferred from natural traits or artificially placed over the skin. This clearly corresponds to the expression modelling paradigm posed by spatially-discrete models based on appearance, but these are not subject to this specific family of deformation techniques, as commented in Section 3.3.2. In the case of motion interpolation, such extrapolation is performed by applying some kind of interpolative scheme which translates the displacements of local features onto the whole extent of the skin.

The interpolative schemes mentioned in the previous paragraph can be classified according to the type of features they derive motion from. In the existing literature, a large variety of generic deformation approaches have been applied where these features range from isolated points to more complex structures like parametric curves, volumetric lattices and even other polygonal meshes. However, the generality of these techniques does not necessarily suit the nature of facial tissue deformation. Comparing their adequacy on just visual results is rather inconclusive, as many of them involve a series of manual steps (e.g. weight masking, point colocation, partitioning) that significantly affect the output quality; instead, we shall identify the intrinsic qualities which these methods may offer disregarding human intervention, in order to be able to provide a more objective evaluation.

Despite geometric warps not being based on the mechanical properties of the skin,

there are certain aspects of the skin's behaviour that have to be taken into consideration for the interpolated motion to attain an acceptable level of realism, provided that the distribution of reference features is dense enough to represent the variety of deformation exerted by independent functional units. Such aspects are:

- *Motion propagation:* the deformation applied at the points of attachment of facial muscles is transmitted along the connectivity of the tissue, generally coincident with that of the observable skin surface. Consequently the interpolation of feature displacements has to be performed in a way that is consistent with the invaginations and discontinuities of the skin (e.g. ocular depressions, mouth and eye openings, etc.)

- *Tissue smoothness:* excluding areas where elasticity is compromised, such as furrows and creases, the observable skin surface preserves its overall smoothness throughout the deformation associated with facial expression. This translates into continuity requirements for the formulation of the corresponding interpolative paradigm.

- *Locality of the deformation:* beyond ordinary tissue connectivity, facial motion is dampened by the underlying fat and muscle layers, restricting it to the local environment of the connection with the muscle. Such dampening also has to reflect in the locality of the deformation extrapolated from each of the features, ideally by allowing its extents to be constrained to an arbitrarily small neighbourhood.

Furthermore, the degree of automation and other practical aspects of these methods should be given special prominence, as they constitute one of the main advantages of Motion Interpolation with respect to Morphable Models. The following sections undertake this task on an individual basis.

Point-based deformations

Section 2.4.1 introduced the seminal work by Williams [Wil90] taking a temporally discrete approach to modelling facial expression based on appearance; its author makes use of reflective marker points on the face of the performer, and subsequently defines a deformation paradigm that operates by simply interpolating marker displacements across the Euclidean space where the skin surface is embedded.

For a given vertex $\mathbf{v}_i$ on a face labelled with points $\{\mathbf{m}_k\}$, the deformed configuration $\mathbf{v}'_i$ corresponding to a displaced marker layout $\{\mathbf{m}'_k\}$ is given by the expression:

$$\mathbf{v}'_i = \sum_j w^i_j (\mathbf{m}'_j - \mathbf{m}_j) + \mathbf{v}_i \qquad (3.12)$$

In Equation 3.12, factor w^i_j stands for the weighting of the displacement of the jth marker referred to the ith vertex representing the face. Whilst Williams does not go into details on how to compute such factors, apart from giving the abstract idea of "airbrushing" them over the model, similar works [PLG91, KGM01, PP01] effectively use cosine drop-off functions of the Euclidean distance for this purpose.

Let $R(\mathbf{m}_j)$ be the radius of the spherical volume of affection of the jth marker point, which enables locality control by means of delimiting which vertices are influenced by this marker's motion. We can now define a family of C^1-continuous weighting functions as:

$$w_j^i(S) = \begin{cases} \frac{1}{2}\left(1 + \cos\left(\frac{\|\mathbf{v}_i - \mathbf{m}_j\|}{R(\mathbf{m}_j)}\pi\right)\right)^n & \text{if } R(\mathbf{m}_j) > \|\mathbf{v}_i - \mathbf{m}_j\| \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

However, the locality restrictions enforced by the non-null domain of weighting functions is generally not sufficient to prevent motion propagation across facial discontinuities. Also, motion being interpolated across the space embedding the skin surface rather than along the surface itself allows for unrealistic displacement propagations to take place, specially in regions of high curvature. For instance, in the neighbourhood of facial concavities, the interpolated motion emanating from one of the markers can affect areas otherwise connected through a much longer skin path than the corresponding radius of affection.

In [Wil90], Williams did not give any special considerations to the above-mentioned particulars, as his implementation was mainly experimental and aimed solely to showcase the fundamental idea of temporally-discrete expression modelling. Successive iterations, more concerned with the final results, did pay more attention to these aspects; Kshisagar et al. [KGM01] adopted an edge-based metric along the model's polygonal mesh in order to discriminate discontinuities, as well as approximating the surface metrics that should weight the interpolation. This approach is heavily dependent on the mesh topology, and consequently it is rather difficult to translate across diverse facial models. Additionally it has its own singularities following the choice of local triangulation.

In [PP01], Pasquiarello et al. adopt an alternative approach for static partitioning, delimiting in the model's mesh areas to be influenced just by the motion of a particular set of markers. This is more model-dependent than the previous case, and also introduces additional discontinuities in the formulation of the deformation, allowing for it to drop below the C^1 degree of the weighting functions themselves. Nevertheless such shortcomings do not represent a problem for Pasquiarello et al., as their work targets a single model for which the regionalization is tweaked in order to avoid such effects. Should this approach be used in the design of a more generic framework, tackling these problems will effectively require a vast amount of manual adjustment, partially defeating the purpose of using motion interpolation instead of morphable models.

A more exotic solution is given in a commercial product called *Famous3D*[6], where in a similar fashion to Williams' original idea, animators are required to "paint" the weighting of every marker over the model's mesh. This as laborious as the method in [PP01], demanding extensive tweaking for each animated model.

Radial Basis Functions

In contrast with the simplicity of point-based deformations, Radial Basis Functions (RBFs) offer a more sophisticated interpolative scheme that, when used to construct

---

[6] Famous3D `http://www.famous3d.com/`
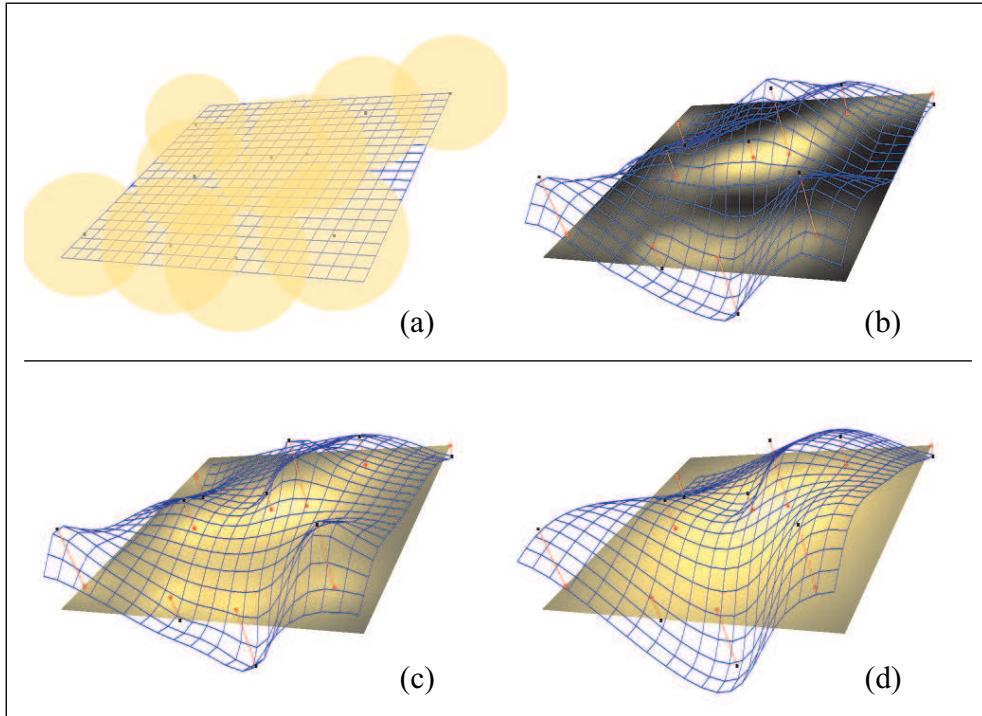
Fig. 3.5: Examples of Point-Based Deformations (PBDs) and Radial Basis Functions (RBFs) operating on a planar sheet. Ordinary PBDs with the radii of affection exhibited in (a) produce the results in (b), while quadric RBF kernels yield (c), and the application of Hardy's multiquadrics results in (c). Weighting functions are represented as yellow shading on the xz plane.

a deformation algorithm, allows further control of locality and continuity aspects (see Figure 3.5). The expression of the deformed image of the ith vertex is similar to Equation 3.12, but referred instead to each independent coordinate α of $\mathbf{v}_i$:

$$\mathbf{v}'_i[\alpha] = (P\ \mathbf{v}_i)[\alpha] + \sum_j w_j^\alpha\ \phi_j\left(\|\mathbf{v}_i - \mathbf{m}_j\|\right) \tag{3.14}$$

Despite formulaic similarities with point-based deformations, the RBF warp computes weighting factors w_j^α in a purely deterministic way, while the arbitrary term that enables locality is in fact the family of functions ϕ_j, theso-called RBF kernels. These kernels are also chosen as expressions of the Euclidean distance between vertices and markers, with arbitrary degree of continuity and domain of definition; Hardy's multiquadrics [HG75] are a good choice for extrapolating deformation from an irregular set of samples. This specific set of RBF kernels takes a distinct form for each of the markers;

in particular, for the quadratic case:

$$\phi_j(d) = \frac{1}{\sqrt{d^2 + r_j^2}} \tag{3.15}$$

where the radius r_j is computed as:

$$r_j = min_{i \neq j} \|\mathbf{m}_i - \mathbf{m}_j\|$$

Whenever a global domain of definition is considered, multiquadric kernels behave as C^∞-continuous functions. By computing their values at the known sample points (markers) and evaluating Equation 3.14 accordingly we can construct a series of equations, linear on the different w_l^β, which for each marker coordinate u yield the following matrix system:

$$\begin{bmatrix} \phi_0^0 & \dots & \phi_n^0 & (\mathbf{m}_0)_u \\ \vdots & \ddots & \vdots & \vdots \\ \phi_0^n & \dots & \phi_n^n & (\mathbf{m}_n)_u \\ (\mathbf{m}_0)_u^T & \dots & (\mathbf{m}_n)_u^T & 0 \end{bmatrix} \begin{bmatrix} w_0^\alpha \\ \vdots \\ w_n^\alpha \\ P[\alpha] \end{bmatrix} = \begin{bmatrix} (\mathbf{m}_0'[\alpha])_u \\ \vdots \\ (\mathbf{m}_n'[\alpha])_u \\ 0 \end{bmatrix} \tag{3.16}$$

where ϕ_j^i stands for $\phi_j\left(\|\mathbf{m}_i - \mathbf{m}_j\|\right)$

Assuming that this linear system is well conditioned, coefficients w_j^α are uniquely determined and can be trivially computed for every coordinate α making up the space of the markers. At the same time, the rigid body transformation that complements local deformation is expressed by means of P, otherwise known as the polynomial term of the RBF, and it is also determined by the relations expressed in Equation 3.16. This finally fulfils the requirements for the computation of v_i' according to Equation 3.14.

This procedure has been used in all sorts of animation contexts, such as morphing photographs [ADR94, EM03]. The first work in applying this approach to three-dimensional facial animation was Fidaleo et al.'s [FNK99], although this is debatable since Pighin [Pig99] used an equivalent method to deform reference facial poses to fit alternative physiognomies. Such deformation has become a common practice, documented in successive works [SCS03, TC03] and even used in commercial products, like *BioVirtual's 3DMe Now!*[7].

While continuity is no longer an issue, due to the smoothness of multiquadrics, motion propagation is as much of a problem as it was in the case of point-based deformations. Fidaleo et al. [FNK99] propose using a edge-based cut-off similar to the mesh metrics used in [KGM01]; this yields the same problems described in Section 3.3.3:*Point-based deformations*, now even more manifest due to the numerical sensitivity of RBFs. However the facial models used by Fidaleo et al. do not exhibit a discontinuous mouth that could be affected by such issues.

Curve-driven warps

Contours constitute a feature of key importance in the perception of facial expression, and therefore it is sensible to introduce deformation techniques specifically conceived to

---

[7] 2003 Biovirtual `http://www.biovirtual.com/`
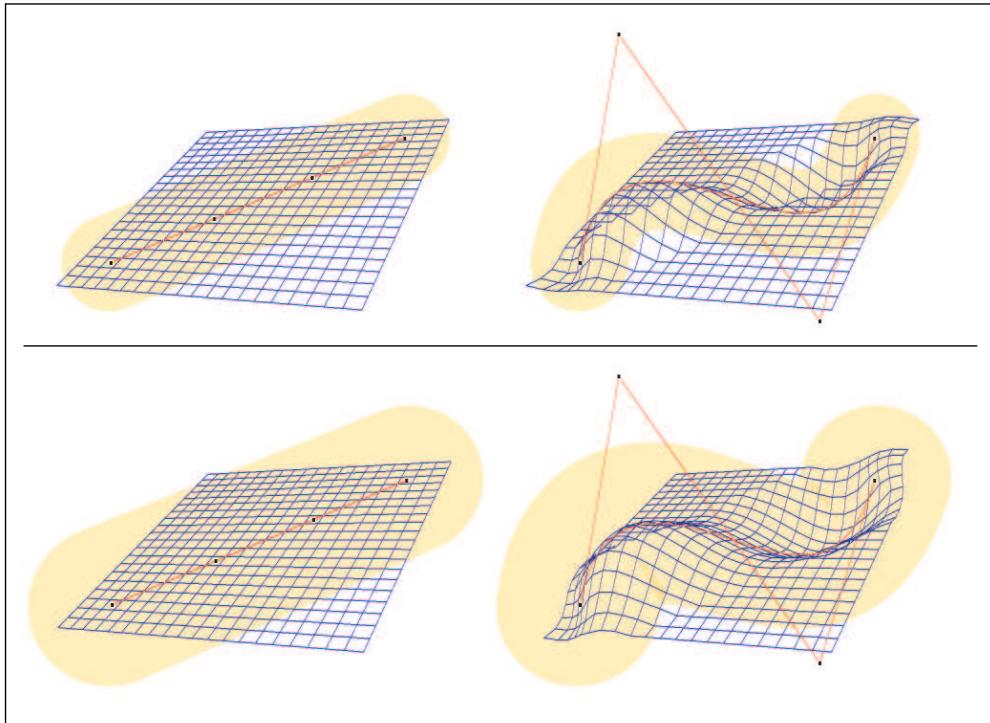
Fig. 3.6: Examples of Axial Deformations operating on a planar sheet. The figures on
the left show the rest configurations of the curve, that when deformed perpen-
dicularly to the plane produce effects such as those to their right. Bottom and
top rows exhibit results with different lozenge sizes.

represent them with the maximum accuracy. Parametric curves are a well known field
of Computational Geometry (see, for instance [Far95]), and from their study it is easy
to derive interpolative schemes translating their deformation to a local neighbourhood
in any surface where they could be embedded. In Computer Graphics literature, two
alternative solutions for this problem have been proposed, the first by Lazarus et al. in
[LCJ94], and a later and more popular one by Singh and Fiume in [SF98].

Lazarus's technique, Axial Deformations (AxDefo), proposes deriving a full param-
eterisation of the surface with respect to a well-defined tangent frame field of the curve
[Klo86], and then computing the deformed image of any given vertex by re-evaluating
such parameterisation over the displaced curve (see Figure 3.6). Denoting as $\alpha(\mathbf{v})$ the
parametric coordinate of the projection of the vertex $\mathbf{v}$ over the curve $c(s)$, and taking
$F(s)$ as the corresponding expression of Klok's frame field (a 3×3 matrix in $\mathbb{R}^3$), the
algorithm can be given as:

$$\mathbf{v}'_i = \mathbf{c}'(t) + (\mathbf{v}_i - \mathbf{c}(t))\, F(t)^{-1}\, R(\theta(t))\, F'(t)\big|_{t=\alpha(\mathbf{v}_i)} \tag{3.17}$$

where $R(\theta(s))$ represents an arbitrary rotation (row-major) around the unitary vector
mapped by $F(t)$ to the tangent of the curve. The effects of such curves can be lo-

cally constrained by applying the usual weighting drop-off functions, also allowing the combination of several of these control elements.

Conversely, Singh and Fiume's approach (Wires) is reminiscent of physically-based techniques, as it "pulls" vertices towards the curve rather than parameterising them along a tangent frame field. By including a scaling term s, and denoting as $w_c(\mathbf{v})$ the weighting function for a vertex $\mathbf{v}$ with respect to the curve $\mathbf{c}(t)$, we can formulate this warp as the result of the following series of operations:

$$
\begin{aligned}
t &= \alpha(\mathbf{v}_i) \\
\mathbf{v}_i^s &= \mathbf{v}_i + (1 + (s-1)\, w_c(\mathbf{v}_i))\, (\mathbf{v}_i - \mathbf{c}(t)) \\
\mathbf{v}_i^r &= (\mathbf{v}_i^s - \mathbf{c}(t))\, \theta(t)) + \mathbf{c}(t)\, R_c(w_c(\mathbf{v}_i) \\
\mathbf{v}_i' &= \mathbf{v}_i^r + w_c(\mathbf{v}_i)\, (\mathbf{c}'(t) - \mathbf{c}(t))
\end{aligned}
\tag{3.18}
$$

Disregarding the change of approach, the method expressed in Equation 3.18 shares the projective spirit of Equation 3.17, making the two techniques effectively equivalent in practice. Only the work in [SF98] actually refers to Facial Animation, and not in an extensive way, but its integration in commercial modelling tools (*Alias/Wavefront Maya*[8]) has given this deformation approach a special relevance. Also, the simplest linear form of AxDefos is frequently used for interactive content, in the shape of *bone ridges* (see *project: messiah*[9]) controlled through specialized software originally intended for skeletal animation (e.g. *Kinetix's Character Studio*[10]).

The continuity of the deformation obtained through these algorithms depends directly on that of the operator projecting vertices onto the parametric domain of the curve, which in turn relates to the smoothness of the curve itself (for a deeper justification refer to Section 3.5.2). Once more, the use of Euclidean metrics for weight computation complicates the enforcement of discontinuities and proper propagation of motion, although the higher dimensionality of the curves' embedding in the face model attenuates this problem. However, the gains over previous methods are diminished by the inadequacy of curve controllers for driving the deformation of internal regions of the model where no contour is present (e.g. cheeks). This usually requires combining them with some other warping method, something that both these frameworks allow for.

Free Form Deformations

Free Form Deformations (FFDs) are an extremely popular deformation technique that uses the de Casteljau algorithm [Far95] to increase the degree of distance-weighted interpolation schemes operating on cubic topologies, enabling higher degrees of continuity in the resulting formulation (see Figure 3.7 for several examples). This also comes at the expense of introducing additional displacement samples, in the shape of a regular lattice formed by the so-called control points. The algorithm only guarantees an exact interpolation of the samples at the corners of each topologic cell, therefore the interme-

---

[8] Alias/Wavefront http://www.alias.com/eng/products-services/maya/

[9] 2001 pmG Worlwide LLC http://www.projectmessiah.com/

[10] 1997 Kinetix Inc. http://ktx.com/3dsmaxr2/
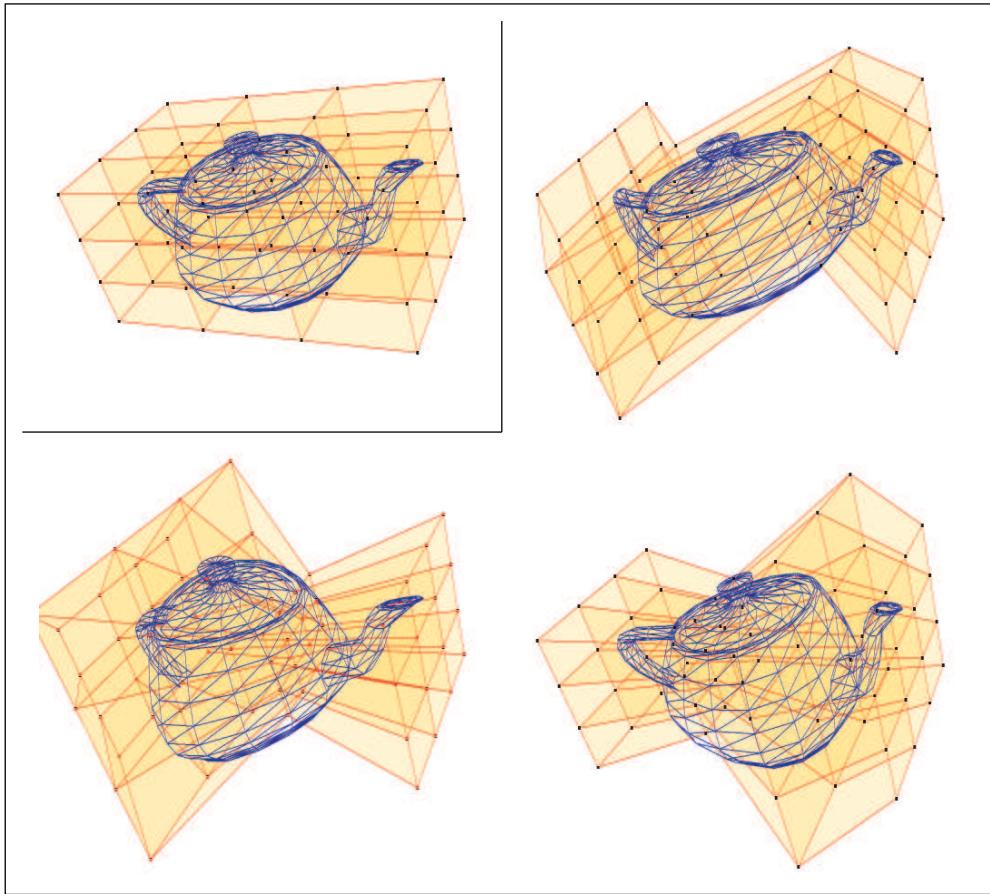
Fig. 3.7: Examples of Free Form Deformations operating on the Utah teapot model. The initial configuration of the cubic lattice (top left) is transformed onto its surrouding images, transmiting the corresponding deformation onto the model it encompasses.

diate control points cannot be directly taken from the set of facial markers, requiring for them to be specified in some other manner.

The expansion of de Casteljau's polynomial expressions in a cubic grid's cell yields the trivariate version of the well-known Bernstein polynomials of degree n [Far95], denoted as B_{ijk}^n; using Bézier's scheme for labelling the control points as $\mathbf{c}_{ijk}$ [B78], we can formulate the expression of the deformed image of the ith vertex in the model as follows:

$$\mathbf{v}_i' = \sum_{p,q,r}^{n} \mathbf{c}_{pqr}' \, B_{pqr}^n(\alpha(\mathbf{v}_i), \beta(\mathbf{v}_i), \gamma(\mathbf{v}_i)) \tag{3.19}$$

where Bernstein polynomials can be computed as:

$$B_{ijk}^n(u,v,w) \equiv B_{ij}^n(u,v,w) = \begin{pmatrix} n \\ i,j \end{pmatrix} u^i v^j w^{(n-(i+j))} \tag{3.20}$$

In Equation 3.19 terms α, β and γ represent the coordinate functions of the original (undeformed) cell in which $\mathbf{v}_i$ is embedded. For a regular arrangement distributed over a cube or more generally any paralleliped, these functions can be equated to the expressions below:

$$\alpha(\mathbf{v}) = \frac{< \mathbf{v} - \mathbf{o}, e_\beta, e_\gamma >}{< e_\alpha, e_\beta, e_\gamma >}; \ \beta(\mathbf{v}) = \frac{< e_\alpha, \mathbf{v} - \mathbf{o}, e_\gamma >}{< e_\alpha, e_\beta, e_\gamma >}; \ \gamma(\mathbf{v}) = \frac{< e_\alpha, e_\beta, \mathbf{v} - \mathbf{o} >}{< e_\alpha, e_\beta, e_\gamma >}$$

where axial vectors e_α, e_β and e_γ stand respectively for $\mathbf{c}_{003} - \mathbf{c}_{000}$, $\mathbf{c}_{030} - \mathbf{c}_{000}$ and $\mathbf{c}_{300} - \mathbf{c}_{000}$. The point $\mathbf{o}$ is identically $\mathbf{c}_{000}$, while ternary vector operator $< u,v,w >$ denotes the mixed product, otherwise expressed as $(u \wedge v) \cdot w$.

The introduction of FFDs to Computer Graphics is due to the work by Sederberg and Parry in [SP86], although it was originally outlined long before in the original documents by Bézier [B78], without its potential application being much noticed until this comeback. In fact, the plain paralleliped formulation presented in the equations above is rather limited for what modelling needs could require. Additionally, weights can be given to every control point in order to accentuate the speed of local convergence, in a similar way to what rational Bézier curves do (see [Far95]). The resulting expression for Equation 3.19 is now:

$$\mathbf{v}_i' = \frac{\sum_{p,q,r}^n w_{pqr} \, \mathbf{c}_{pqr}' \, B_{pqr}^n(\alpha(\mathbf{v}_i), \beta(\mathbf{v}_i), \gamma(\mathbf{v}_i))}{\sum_{p,q,r}^n w_{pqr} \, B_{pqr}^n(\alpha(\mathbf{v}_i), \beta(\mathbf{v}_i), \gamma(\mathbf{v}_i))} \tag{3.21}$$

Such Rational Free Form Deformations are the particular form of FFD used for Facial Animation in the earliest contribution of this kind, by Kalra et al. [KMM92]. In this work isolated lattice blocks are laid over the face model, encompassing the regions affected by specific sets of FACS AUs (c.f. Section 2.3.1); then a manually produced mapping relates AU variations to distortions of the control point lattices, ultimately representing the desired expression. This anatomically-based approach results evidently more cumbersome than the frameworks previously discussed which model expression on appearance, however it also allows a higher degree of control thanks to the additional degrees of freedom introduced (intermediate control points and weights).

Nevertheless the additional parameters added by Kalra's work make up a non-intuitive and difficult way of manipulating the skin surface, as this deformation technique operates on a regular volumetric partition and the dermis itself is an unruly 2-manifold embedded in it. Therefore motion propagation does not happen along the skin surface like it should, but across an encompassing volume, as if the embedding of the face was somehow solid. Also, while smoothness is invariably preserved inside the lattices, continuity can be compromised along their corresponding boundaries, as this paradigm does not make provisions for overlapping lattices, neither does it consider a cell network that could cover the whole face. Discontinuities can be enforced as long as they can be

separated in disjoint cells, however contour control is not as precise as with curve-driven deformations.

The approach taken in the commercial tool *JetaReyes* by *REM Infografica*[11] tries to ease the animation workflow outlined by the technique commented above, by providing a heuristic that controls the lattice configurations through a simple 2D curve-based interface; while this approach does not elude the limitations of Kalra's method (specially precise lip control), it significantly facilitates the production of artist-made animation.

Successive iterations of the FFD algorithm [Coq90, MJ96] extended the technique by enabling the use of lattices of arbitrary shape and topology. Tao et al. [TH98] made use of these advances to propose an alternative approach to spatially-discrete Facial Animation systems, by constructing a piecewise embedding of the face model connecting offset markers with contiguous grid cells. This approach, although applied to a different model of expression from that in [KMM92], solves much of its problems as a deformation framework. Arbitrary lattice configuration enables a better fitting of contours, and the piecewise topology of irregular cells provides a far better fitting of the surface to deform than the cubic grids described before. Consequently, the resulting propagation of motion is far more consistent with the actual phenomenon, as the introduction to this section discussed in more detail. Deformation smoothness however is not so easily achievable. Tao et al. do not consider this aspect, as the facial models they use are slightly more complex than the control topology itself, but it is in fact attainable by properly conditioning the internal control points of contiguous cells [Far95]. However this is quite an arduous task, given the significant amount of unregulated degrees of freedom implicit in the cells, and specially the arbitrary topology and degree of each of them. Also, a high degree of continuity compromises locality through the conditioning, as can be demonstrated in a similar way to the reasoning in Section 3.5.2.

Dirichlet Free Form Deformations

An alternative warping algorithm that operates over volumetric partitions is the so-called Dirichlet Free Form Deformation (DFFD). Despite naming similarities, the procedure outlined by this approach is rather different from that followed by ordinary FFDs; for a start, the lattice's topology is implicitly generated as part of the algorithm, rather than as a prerequisite. This is so because the parameterisation used in this case is Sibson's coordinates [Sib80], constructed over the Voronoi/Dirichlet graph of the marker cloud [Ber00]. Such a parameterisation specifies the influential markers for the deformation of a given vertex $\mathbf{v}_i$ as its set of *natural neighbours* $N(\mathbf{v}_i)$, consisting of the nodes of the graph whose Voronoi cells are adjacent to that of $\mathbf{v}_i$ after its insertion. Upon displacement of the markers, the deformed image of $\mathbf{v}_i$ is computed by re-evaluating its corresponding parametric expression:

$$\mathbf{v}'_i = \sum_{\mathbf{m}_j \in N(\mathbf{v}_i)} w^i_j \, \mathbf{m}_j \qquad (3.22)$$

Sibson coordinates are proportional to the area of the intersection between the cells

---

[11] 1996 REM Infografica `http://www.reyes-infografica.com/`

removed in the insertion (corresponding to each of the natural neighbours) and that of the new cell added. These intersection areas (or volumes, in the n-dimensional case) are then normalized with respect to the magnitude of the vertex cell, resulting in the proper w_j^i. As usually occurs with Voronoi diagrams, there is a simplified local algorithm that allows computing this coefficient without performing the actual insertion, using a static solution for the dual problem (Delaunay) constructed upon the $\mathbf{m}_i$s alone (see Figure 3.8). In the planar case, where $R(\mathbf{v}_i)$ is the set of Delaunay triangles invalidated by the insertion of $\mathbf{v}_i$, we can compute Sibson coordinates through the following procedure:

- for each of the triangles t in $R(\mathbf{v}_i)$:

$$
\begin{aligned}
o_1(\mathbf{v}_i, t) &= O(p_2(t), p_3(t), \mathbf{v}_i) \\
o_2(\mathbf{v}_i, t) &= O(p_3(t), p_1(t), \mathbf{v}_i) \\
o_3(\mathbf{v}_i, t) &= O(p_1(t), p_2(t), \mathbf{v}_i) \\
o(t) &= O(p_1(t), p_2(t), p_3(t))
\end{aligned}
$$
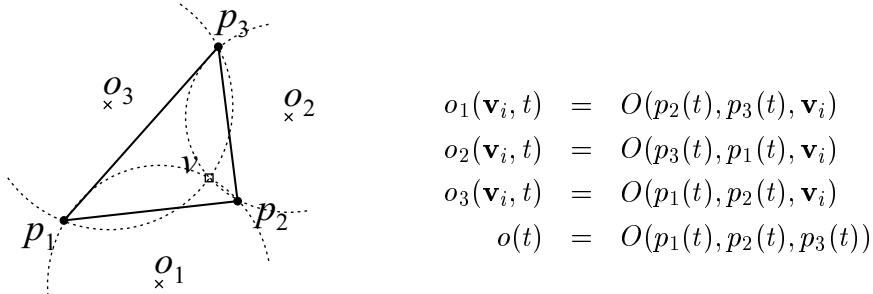
- then for each tuple $\{m, n, l\} \in perm(\{1, 2, 3\})$:

$$
\alpha_l(\mathbf{v}_i, t) = \frac{1}{2} \|(o_m(\mathbf{v}_i, t) - o(t)) \wedge (o_n(\mathbf{v}_i, t) - o(t))\| \tag{3.23}
$$

- and finally, taking $I(\mathbf{m}_j)$ as the set of Voronoi cells incident to a given marker $\mathbf{m}_j$:

$$
w_j^i = \sum_s \sum_{l=1}^{3} \alpha_l(s)\ \delta_{idx(s,l)=\mathbf{m}_j} \ \bigg|\ s \in R(\mathbf{v}_i) \cap I(\mathbf{m}_j) \tag{3.24}
$$

Extending the previous formulation to the three-dimensional case can be achieved by substituting triangular cells with Delaunay tetrahedrons and making the corresponding changes [Bow81, Wat81]. This particular approach is applied by Moccozet and Magnenat-Thalmann [MT97] to hand animation, interpolating the displacement of a lattice of control points embedding the skin. The deformation paradigm is similar to that of ordinary FFDs, but as commented before no topology is necessary. On the other hand, this method has the severe inconvenience of requiring the marker points to be offset over the skin surface, as they need to span a volume encompassing the whole deformed object. This complicates the application of raw Motion Capture, since the artefacts needed to reproduce such a structure will surely interfere with natural hand motion (they will have to protude significantily over the skin).

Conveniently, the planar nature of the human face allows for alternative control constructions in the case of Facial Animation. In [EPM98] Escher et al. define a similar deformation paradigm to that in [MT97] by using a cylindrical projection of the face. Apart from the slight metric distortion incurred by the projection, motion propagation defined in this way is consistent with the skin surface, as long as boundary enforcement
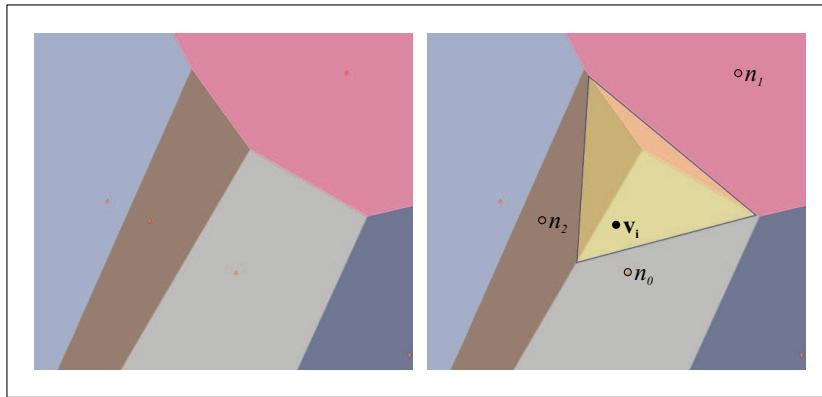
Fig. 3.8: Voronoi diagram of a cloud of control points before (left) and after (right) the addition of $\mathbf{v}_i$; $\{\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2\}$ are the natural neighbours of this new point, and their corresponding Sibson coordinates w_j^i are proportional to the intersecting area of their previous Voronoi cells with that of $\mathbf{v}_i$ (shaded in yellow, right).

methods, such as those introduced by Lawson in [Law77a], are used for mouth and eye discontinuities. Continuity under this framework is generally C^1, except in the neighbourhood of sampled points, where it actually drops to C^0, as Sambridge et al. [SBM95] demonstrate. This can produce a rather undesiderable *pinching* effect in the skin around the markers, although it would only become noticeable at higher mesh resolutions than those employed by Escher et al. Locality is also clearly delimited, as the topologic conditions of the Voronoi graph guarantee under a sensible distribution of marker points.

Surface-Oriented Free Form Deformations

Alternatively to the control structures used by the deformation paradigms previously discussed in this Chapter, Surface Oriented Free Form Deformations (SoFFDs, [SK00]) take up coarse polygonal meshes to drive the warp of much more complex surfaces. The mapping between coarse and fine meshes is such that both can be arbitrarily defined, without the need for the crudest representation to be in any way coincident with the other (e.g. a simplification). This allows for a flexible deformation paradigm where the animation of complex models can be controlled by MoCap or manual keyframing directly applied to a loosely-related coarse mesh. Such a tool is a useful resource for character animation, as well as for many other purposes; in fact, SoFFDs have became part of the *Maya* modelling package by *Alias/Wavefront*[12], where they take the less sophisticated name of *meshwrap*.

But the real contribution of SoFFDs is not their reductionistic nature, shared by all the other paradigms described in this section, but the fact that the control elements share the same structure as the object to be deformed: discrete 2-manifolds. This alleviates

---

[12] Alias/Wavefront `http://www.alias.com/eng/products-services/maya/`

the inconsistency of many of the other approaches, which try to relate the skin surface to distinct counterparts such as volumes or scattered points. While simplification is not a requirement, it is a possibility in order to automatically derive the control structure from the model, while preserving vertices identified with marker points (see for instance [Hop96]). This eases significantly the manual fitting of control elements, especially in comparison with the other alternatives discussed in this Section.

The SoFFD algorithm operates by relating the deformation of each of the target object's vertices to the control mesh facets. For a given vertex $\mathbf{v}_i$, the contribution to its deformation by the jth facet of the control mesh, connecting marker points $\{\mathbf{m}_l, \mathbf{m}_m, \mathbf{m}_n\}$, is computed as follows:
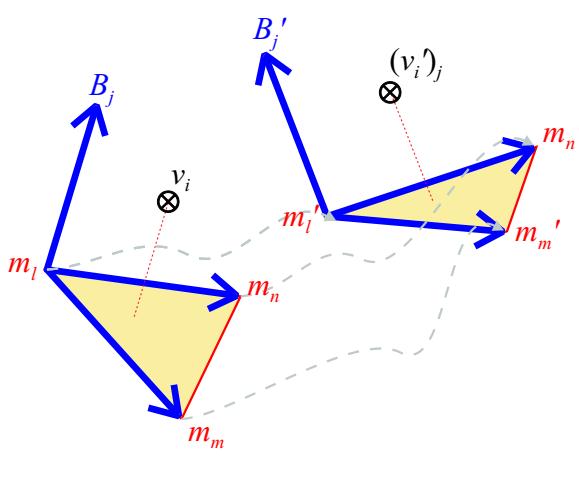
$$(\mathbf{v}_i')_j = (\mathbf{v}_i - \mathbf{m}_l)\, B_j^{-1}\, B_j' + \mathbf{m}_l' \tag{3.25}$$

where B_j is a $3{\times}3$ matrix that takes the form:

$$B_j = \begin{bmatrix} \mathbf{m}_m - \mathbf{m}_l \\ \mathbf{m}_n - \mathbf{m}_l \\ \frac{(\mathbf{m}_m - \mathbf{m}_l) \wedge (\mathbf{m}_n - \mathbf{m}_l)}{\|(\mathbf{m}_m - \mathbf{m}_l) \wedge (\mathbf{m}_n - \mathbf{m}_l)\|} \end{bmatrix}$$

and equivalently

$$B_j' = \begin{bmatrix} \mathbf{m}_m' - \mathbf{m}_l' \\ \mathbf{m}_n' - \mathbf{m}_l' \\ \frac{(\mathbf{m}_m' - \mathbf{m}_l') \wedge (\mathbf{m}_n' - \mathbf{m}_l')}{\|(\mathbf{m}_m' - \mathbf{m}_l') \wedge (\mathbf{m}_n' - \mathbf{m}_l')\|} \end{bmatrix}$$

Ultimately, the definition of $(\mathbf{v}_i')_j$ in Equation 3.25 is used as a term the expression $\mathbf{v}_i' = \sum_j w_j(\mathbf{v}_i)\, (\mathbf{v}_i')_j$ in order to compute the final deformation. Weight functions w_j control locality in an identical way to the cosine drop-off used by previously described algorithms (Equation 3.13).

Motion propagation as prescribed by this algorithm is far more consistent with the nature of the tissue than the other interpolative approaches commented on before. This is so because the control structure over which discrete displacements are interpolated is itself an approximation of the skin surface. Skin discontinuities still need to be explicitly specified, as the weighting metrics are identically Euclidean; nevertheless, this constitutes a less sensitive problem than in previous cases, again due to the approximating property of the control mesh.

The degree of continuity that can be attained with SoFFDs matches that of the weighting functions used, as long as the combined area of affection encompasses the whole surface to be deformed. Choosing cosine drop-off functions as commented before means that such degree wo be at least C^1. Locality of deformation is equally contained in the weighting, ultimately relating the performance of this algorithm to the correspondence between control and target meshes. Fortunately, in the case of the relatively simple geometry of the human face, adequate control mesh configurations can be easily
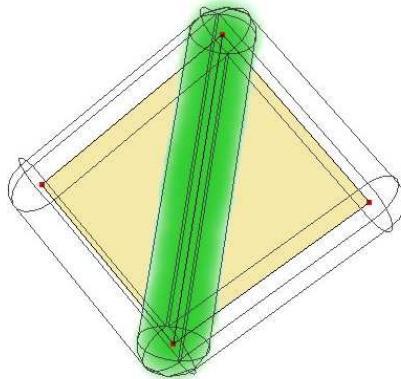
Fig. 3.9: Overlapping areas of affection of two triangular facets in the SoFFD control mesh.

found for sensible marker layouts.

Extending the area of deformation of each facet in the control mesh to a its neighbourhood in $\mathbb{R}^3$ may prove a valuable tool for enforcing continuity, but it also compromises the proper propagation of motion whenever shearing or non-uniform scaling of the facets occurs. This is specially ostensible in areas under the effect of two or more of these control elements, like the volume shaded in green in Figure 3.9.

While shearing and local scaling of the control mesh is not so frequent in the context of full-body animation where SoFFDs are mainly applied, it represents the kind of non-rigid motion that would be experienced by a triangulation of marker points tracking the deformation of facial skin. Figure 3.10 demonstrates how these flaws become evident under relatively small deformations of the control mesh. In addition, deformation defined in this manner is not reversible: e.g. after reattaching a displaced control mesh to a warped configuration, identical to that which produced the warp, reverting the control structure to the neutral pose does not produce the original target mesh. The negative consequences for Facial Animation of these two aspects serve as motivation for the reformulation of SoFFDs as Planar Bones, which is described in the next section.

3.4   Planar Bones

Planar Bones are a new deformation technique derived from SoFFDs; as with the technique by Singh and Kokkevis, they interpolate the displacements of marker points over a connecting polygonal mesh. This defines their area of application in Facial Animation as systems driven by a spatially-discrete model of expression. Another deformation approach that bases mesh warps on the displacement of marker points are the popular skeletal deformations, also known as Linear Bones or simply Bones [MLT88, CHP89, SCP00, Web00]. While in this case the topology of connections is 1-dimensional, Linear Bones still exhibit the same kind of distortions caused by SoFFDs.

a) Confluence of four affection volumes

b) Confluence of three affection
volumes on the boundary
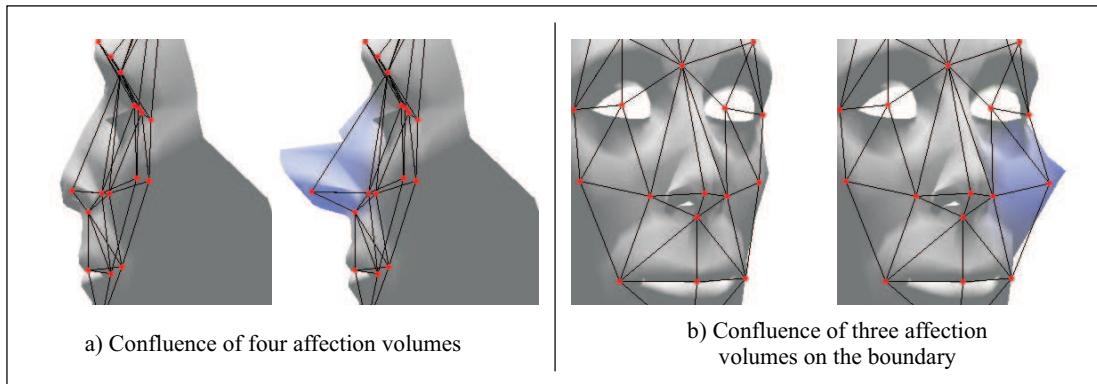
Fig. 3.10: Distortions in the application of SoFFDs

a) Joint between two bones

b) Isolated extremities
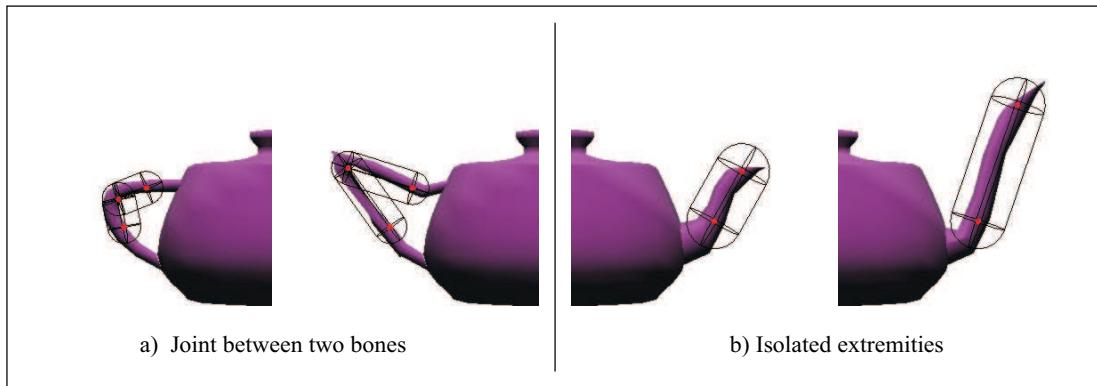
Fig. 3.11: Distortions in the application of Linear Bones

This is shown in the examples in Figure 3.11 where, under local scaling of the control structure, parts of the teapot's spout and handle escape the *lozenge* (capsule) delimiting the volume of affection of the bone segments.

The similarities between the flaws present in these two algorithms (basically deformation overshooting) hints towards the common nature of the problems that cause them. Therefore this section will proceed by analyzing in detail the simpler Linear Bones case, providing a solution for the observed errors, and then trying to extend it to SoFFDs by relating the formulation of both deformation techniques. In doing so, the Planar Bones approach will be presented.

3.4.1   Correct application of Linear Bones

Bones are a simplification of Lazarus' Axial Deformations (see Section 7), substituting control curves by piecewise linear segments (hence the *Linear* denomination). While these segments do not define unequivocally a non-singular tangent frame as curves do,

they can be used in constructing a geometric warp that reproduces the translational and rotational aspects of skeletal deformation.

For a given vertex $\mathbf{v}_i$ on the target surface, the contribution to its deformation by the jth Bone associated with the segment $\overline{\mathbf{m}_l\mathbf{m}_m}$ is computed as:
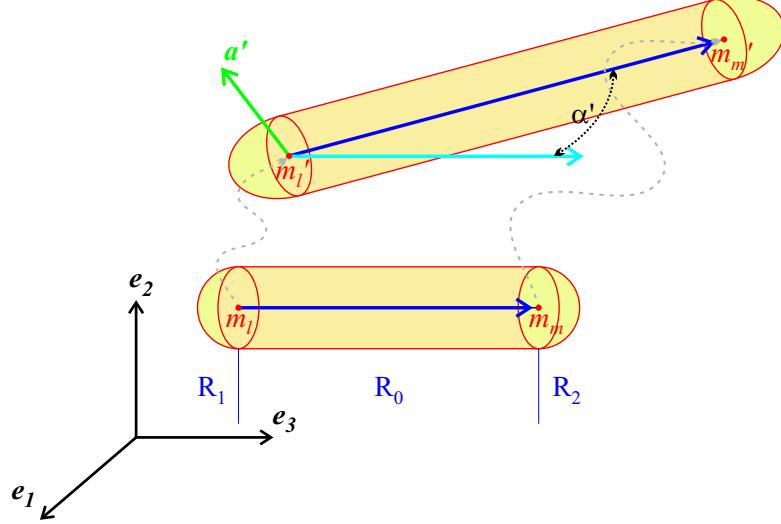
$$(\mathbf{v}'_i)_j = (\mathbf{v}_i - \mathbf{m}_l)\, R_a(-\alpha)\, S_{e_3}\left(\frac{\|\mathbf{m}'_m - \mathbf{m}'_l\|}{\|\mathbf{m}_m - \mathbf{m}_l\|}\right)\, R_{a'}(\alpha') + \mathbf{m}'_l \qquad (3.26)$$

In Equation 3.26 $R_v(\theta)$ represents the 3×3 rotation matrix of θ radians around the arbitrary axis v, and $S_{e_k}(s)$ stands for the equivalent non-uniform scaling of magnitude s along the kth unitary vector. As the diagram illustrates, axes a and a' are computed as the normalized cross product between segments $\overline{\mathbf{m}_l\mathbf{m}_m}$, $\overline{\mathbf{m}'_l\mathbf{m}'_m}$ and the vector e_3, respectively. Also, angles α and α' are given by:

$$\alpha = \angle(e_3, \overline{\mathbf{m}_l\mathbf{m}_m}); \ \ \alpha' = \angle(e_3, \overline{\mathbf{m}'_l\mathbf{m}'_m}).$$

Should any of these expressions result in an indetermination, there is always an alternative choice of e_k that would disambiguate it, unless one of the two segments is null.

In a similar manner to SoFFDs, the final deformation is computed by a weighted sum of the contributions of all segments in the control topology. The use of weighting functions results in finite affection volumes like the ones depicted by elongated capsules in Figure 3.11 and Equation 3.26. In these volumes we can distinguish three different regions, labelled in Equation 3.26 as $\mathbf{R_0}$, $\mathbf{R_1}$ and $\mathbf{R_2}$, where the deformation behaves differently. In the longitudinal region $\mathbf{R_0}$, the deformation exerted by the bone preserves the distance of the warped geometry with respect to the segment. However, in extremes $\mathbf{R_1}$ and $\mathbf{R_2}$, the scaling term of the formulation causes vertices to arbitrarily approximate or distance themselves from the segment's apices, breaking the isometric invariant of $\mathbf{R_0}$.

The outcome of this anisometric behaviour is the unexpected stretching distortion showcased in 3.11.b. Whenever the influence of several control segments is combined

| a) Original | b) Results with Linear Bones | c) Results using the alternative formulation |
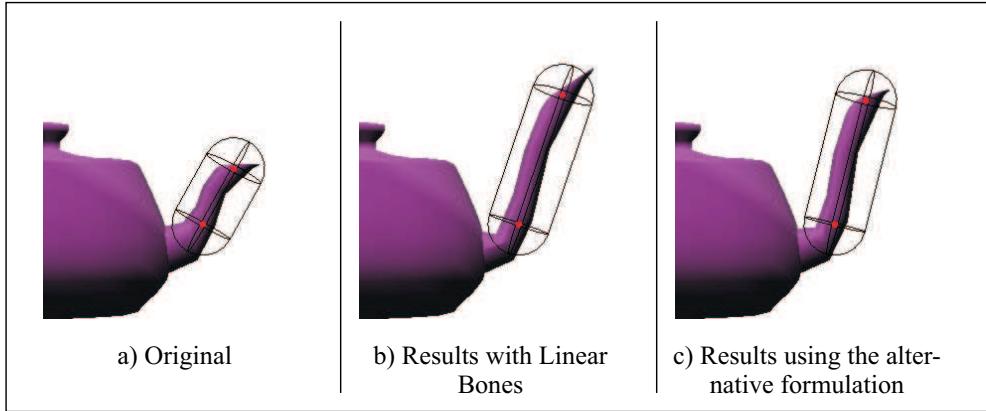
Fig. 3.12: Application of the isometric formulation of Linear Bones.

at one point, like for instance a skeletal joint, the distortive effects simply build up, as Figure 3.11.a illustrates. These flaws in skeletal animation systems using Linear Bones are well known, and have even been given specific names, like for instance the *twisted elbow* issue reported by Lewis et al. in [LCF00]. Correcting such behaviour at a per-segment level can be easily attained by deriving an alternative formulation of Equation 3.26 that treats each distinct region differently:

$$\forall \mathbf{v}_i \in \mathbf{R_k}, (\mathbf{v}'_i)_j = (\mathbf{v}_i - \mathbf{m}_l) \ (B_k(\mathbf{m}_l, \mathbf{m}_m))^{-1} \ B_k(\mathbf{m}'_l, \mathbf{m}'_m) + \mathbf{m}'_l \qquad (3.27)$$

where B_k stands for the following composite transformation:

$$B_k(\mathbf{o}, \mathbf{p}) = \begin{cases} S_{e_3}(\|\mathbf{p} - \mathbf{o}\|) \ R_{a(\mathbf{o}, \mathbf{p})}(\alpha(\mathbf{o}, \mathbf{p})) & k = 0 \\ R_{a(\mathbf{o}, \mathbf{p})}(\alpha(\mathbf{o}, \mathbf{p})) & k \in \{1, 2\} \end{cases}$$

and consistently with the notation employed in Equation 3.26

$$a(\mathbf{o}, \mathbf{p}) \ = \ \frac{(\mathbf{p} - \mathbf{o}) \wedge e_3}{\|(\mathbf{p} - \mathbf{o}) \wedge e_3\|}$$

$$\alpha(\mathbf{o}, \mathbf{p}) \ = \ \begin{cases} \arccos\left(\frac{(\mathbf{p} - \mathbf{o}) \cdot e_3}{\|\mathbf{p} - \mathbf{o}\|}\right) & (a(\mathbf{o}, \mathbf{p}) \wedge e_3) \cdot e_3 > 0 \\ \arccos\left(\frac{(\mathbf{o} - \mathbf{p}) \cdot e_3}{\|\mathbf{p} - \mathbf{o}\|}\right) & \text{otherwise} \end{cases}$$

In Equation 3.27, the arccos function is assumed to follow the IEEE 1754 conventions ($\forall \theta, \arccos(\theta) \in [0, \pi]$). Also, the same premises given above for handling singularities (e.g. $\overline{\mathbf{m}_m \mathbf{m}_l} \parallel e_k$) apply analogously to these equations.

Effectively removing the scaling component in the deformation for regions $\mathbf{R_1}$ and $\mathbf{R_2}$ guarantees isometry and solves the issues experienced in the application of Linear Bones, as Figure 3.12 illustrates. Alternative solutions require considering a pose-space approach [LCF00] or including additional segments to attenuate the problem, resulting in far less efficiency in comparison with the simplicity of Equation 3.27.
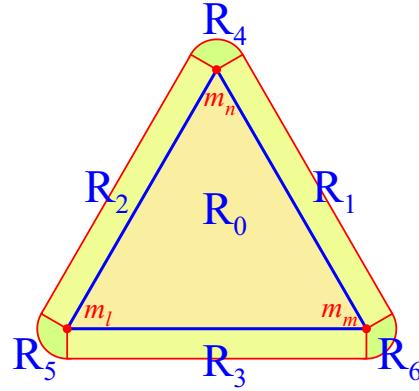
Fig. 3.13: Partitioning of the affection volume for the Planar Bones control facet spanned by points $\{\mathbf{m}_l, \mathbf{m}_m, \mathbf{m}_n\}$.
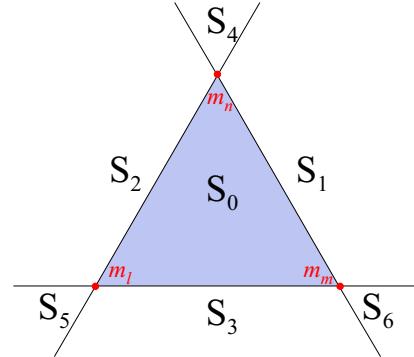
Fig. 3.14: Initial partitioning of the affection volume for a Planar Bones control facet into regions $\{\mathbf{S_k}\}$.

3.4.2 Extending the solution to SoFFDs

We can exploit the similarity between SoFFDs and Linear Bones in order to solve non-rigid deformation problems, resulting in a new formulation that we will refer to as Planar Bones. Partitioning the affection volume for a control facet into regions according to the relevant element (vertex, edge, or the facet itself) yields a rather different topology from the Linear Bone case (see Figure 3.13), however, we can derive a conditional formulation that guarantees isometry in a similar way.

While delimiting $\mathbf{R_0}$ is as trivial as for its Linear Bone equivalent, specifying the conditions distinguishing $\mathbf{R_1}$, $\mathbf{R_2}$ and $\mathbf{R_3}$ from their adjacent regions is a much more complicated task. We start by considering a simpler partitioning; for the jth facet made up of marker points $\{\mathbf{m}_l, \mathbf{m}_m, \mathbf{m}_n\}$ (as depicted in the diagram below) we use barycentric coordinates to define the family of regions $\{\mathbf{S_k}\}$ as follows:

$$\forall \mathbf{v}_i, (r, s, t) = (\mathbf{v}_i - \mathbf{m}_l) \, (B_j)^{-1} \tag{3.28}$$

$$
\begin{aligned}
(r \geq 0) \wedge (s \geq 0) \wedge (t \leq 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_0} \\
(r \geq 0) \wedge (s < 0) \wedge (t \leq 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_1} \\
(r \geq 0) \wedge (s \geq 0) \wedge (t > 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_2} \\
(r < 0) \wedge (s \geq 0) \wedge (t \leq 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_3} \\
(s < 0) \wedge (t > 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_4} \\
(r < 0) \wedge (t > 1) &\Rightarrow \mathbf{v}_i \in \mathbf{S_5} \\
(r < 0) \wedge (s < 0) &\Rightarrow \mathbf{v}_i \in \mathbf{S_6}
\end{aligned}
$$

resulting in the partitioning depicted by Figure 3.14. Using it, we can now refine Equation 3.28 as:

$$
\begin{aligned}
\forall \mathbf{v}_i \in \mathbf{S_1} \wedge
\begin{cases}
(\mathbf{v}_i - \mathbf{m}_l) \cdot (\mathbf{m}_m - \mathbf{m}_l) < 0 & \Rightarrow \mathbf{v}_i \in \mathbf{R_6} \\
0 \leq (\mathbf{v}_i - \mathbf{m}_l) \cdot (\mathbf{m}_m - \mathbf{m}_l) \leq \|\mathbf{m}_m - \mathbf{m}_l\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_1} \\
(\mathbf{v}_i - \mathbf{m}_l) \cdot (\mathbf{m}_m - \mathbf{m}_l) > \|\mathbf{m}_m - \mathbf{m}_l\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_4}
\end{cases} \\[2mm]
\forall \mathbf{v}_i \in \mathbf{S_2} \wedge
\begin{cases}
(\mathbf{v}_i - \mathbf{m}_m) \cdot (\mathbf{m}_n - \mathbf{m}_m) < 0 & \Rightarrow \mathbf{v}_i \in \mathbf{R_4} \\
0 \leq (\mathbf{v}_i - \mathbf{m}_m) \cdot (\mathbf{m}_n - \mathbf{m}_m) \leq \|\mathbf{m}_n - \mathbf{m}_m\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_2} \\
(\mathbf{v}_i - \mathbf{m}_m) \cdot (\mathbf{m}_n - \mathbf{m}_m) > \|\mathbf{m}_n - \mathbf{m}_m\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_5}
\end{cases} \\[2mm]
\forall \mathbf{v}_i \in \mathbf{S_3} \wedge
\begin{cases}
(\mathbf{v}_i - \mathbf{m}_n) \cdot (\mathbf{m}_l - \mathbf{m}_n) < 0 & \Rightarrow \mathbf{v}_i \in \mathbf{R_5} \\
0 \leq (\mathbf{v}_i - \mathbf{m}_n) \cdot (\mathbf{m}_l - \mathbf{m}_n) \leq \|\mathbf{m}_l - \mathbf{m}_n\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_3} \\
(\mathbf{v}_i - \mathbf{m}_n) \cdot (\mathbf{m}_l - \mathbf{m}_n) > \|\mathbf{m}_l - \mathbf{m}_n\|^2 & \Rightarrow \mathbf{v}_i \in \mathbf{R_6}
\end{cases}
\end{aligned}
\tag{3.29}
$$

Additionally, $\mathbf{S_4} \subseteq \mathbf{R_4}$, $\mathbf{S_5} \subseteq \mathbf{R_5}$ and $\mathbf{S_6} \subseteq \mathbf{R_6}$. These indications, together with $\mathbf{S_0} \equiv \mathbf{R_0}$, complete the $\mathbf{R_k}$-partitioning outlined in Figure 3.13. Once done, we can proceed by deriving the regional expressions for the deformation of any given $\mathbf{v}_i$; let's take the original derivation in Equation 3.25, and add a regionalizing condition to it:

$$\forall \mathbf{v}_i \in \mathbf{R_k}, (\mathbf{v}_i')_l = (\mathbf{v}_i - \mathbf{m}_l) \left(B_j^k\right)^{-1} B_j^{k'} + \mathbf{m}_l' \tag{3.30}$$

For $k = 0$, B_j^k takes the same expression as in Equation 3.25,

$$B_j^0 = \left[\begin{array}{c} \mathbf{m}_m - \mathbf{m}_l \\ \mathbf{m}_n - \mathbf{m}_l \\ n_j \end{array} \right], \text{ with } n_j = \frac{(\mathbf{m}_m - \mathbf{m}_l) \wedge (\mathbf{m}_n - \mathbf{m}_l)}{\|(\mathbf{m}_m - \mathbf{m}_l) \wedge (\mathbf{m}_n - \mathbf{m}_l)\|}$$

Furthermore, $B_j^{k'}$ can be analogously computed using the $\mathbf{m}_o'$s. In the case of $k \in \{1, 2, 3\}$, matrices take the alternative forms:

$$B_j^1 = \left[\begin{array}{c} \mathbf{m}_m - \mathbf{m}_l \\ J(\mathbf{m}_m - \mathbf{m}_l) \\ n_j \end{array} \right] ; \; B_j^2 = \left[\begin{array}{c} \mathbf{m}_n - \mathbf{m}_m \\ J(\mathbf{m}_n - \mathbf{m}_m) \\ n_j \end{array} \right] ; \; B_j^3 = \left[\begin{array}{c} \mathbf{m}_l - \mathbf{m}_n \\ J(\mathbf{m}_l - \mathbf{m}_n) \\ n_j \end{array} \right]$$

In the above expressions, conjugate vectors $J(v)$ are computed with respect to n_j:

$$J(v) = \frac{v \wedge n_j}{\|v \wedge n_j\|}$$

Isometry in the transformation embodied by Equation 3.30 for these three regions is guaranteed by congruence with region $\mathbf{R_0}$ in Equation 3.27: matrices B_j^k with $k \in \{1, 2, 3\}$ are always reducible to a composition of the scaling and rotation transformations of their corresponding facet edges, thus their concatenation is distance-preserving with respect to these segments.

Similarly to the approach taken with $\mathbf{R_i} | i \in \{0, 1, 2, 3\}$, we can associate coordinate frames to the remaining partitions:

$$C_j^4 = \left[\begin{array}{c} J(\mathbf{m}_m - \mathbf{m}_l) \\ J(\mathbf{m}_n - \mathbf{m}_m) \\ n_j \end{array} \right] \; ; \; C_j^5 = \left[\begin{array}{c} J(\mathbf{m}_m - \mathbf{m}_n) \\ J(\mathbf{m}_l - \mathbf{m}_n) \\ n_j \end{array} \right] \; ; \; C_j^6 = \left[\begin{array}{c} J(\mathbf{m}_l - \mathbf{m}_n) \\ J(\mathbf{m}_m - \mathbf{m}_l) \\ n_j \end{array} \right]$$

However, if these C_j^k matrices replaced B_j^k in Equation 3.30, the distance with respect to the relevant apex wouldn't be preserved, as it is not possible to define an orthogonal frame that maintains that property during non-uniform scaling and/or shearing of the control facet. It is therefore necessary to introduce additional terms in the equation to compensate for any possible metric distortion:

$$\forall k \in \{4, 5, 6\}, B_j^k = \left(G_j^k \right)^{-1} C_j^k$$

where G_j^k is the metric tensor associated with the C_j^k coordinate system, computed as:

$$G_j^k = C_j^k \left(C_j^k \right)^T \tag{3.31}$$

Given these considerations, all free vectors referred to the C_j^k basis have their 2-norm preserved throughout the deformation. The same does not necessarily apply to any other vector subspace, as the transformation itself is not affine. In addition, Equation 3.31 admits a simpler expression, given the orthogonality of n_j with respect to the other elements of the basis. For instance, in the case of $k = 4$:

$$G_j^4 = \left[\begin{array}{ccc} J(\mathbf{m}_m - \mathbf{m}_l) \cdot J(\mathbf{m}_m - \mathbf{m}_l) & J(\mathbf{m}_n - \mathbf{m}_m) \cdot J(\mathbf{m}_m - \mathbf{m}_l) & 0 \\ J(\mathbf{m}_m - \mathbf{m}_l) \cdot J(\mathbf{m}_n - \mathbf{m}_m) & J(\mathbf{m}_n - \mathbf{m}_m) \cdot J(\mathbf{m}_n - \mathbf{m}_m) & 0 \\ 0 & 0 & 1 \end{array} \right]$$

More simplifications can be achieved by applying the identity $J(v) \cdot J(w) = v \cdot w$. By virtue of the properties of n_j outlined before, all matrix computations included in the expressions resulting from expanding Equation 3.30 can be reduced to operations on the top-leftmost 2×2 minor. This simplifies products and especially inversions, yielding an efficient framework that adds very little computational cost to ordinary SoFFDs.

Furthermore, in order to combine the effects of several control facets, we need to evaluate weighting functions dependent on the distance to each control element. This

a) Originals   b) Results with SoFFDs   c) Results with Planar Bones
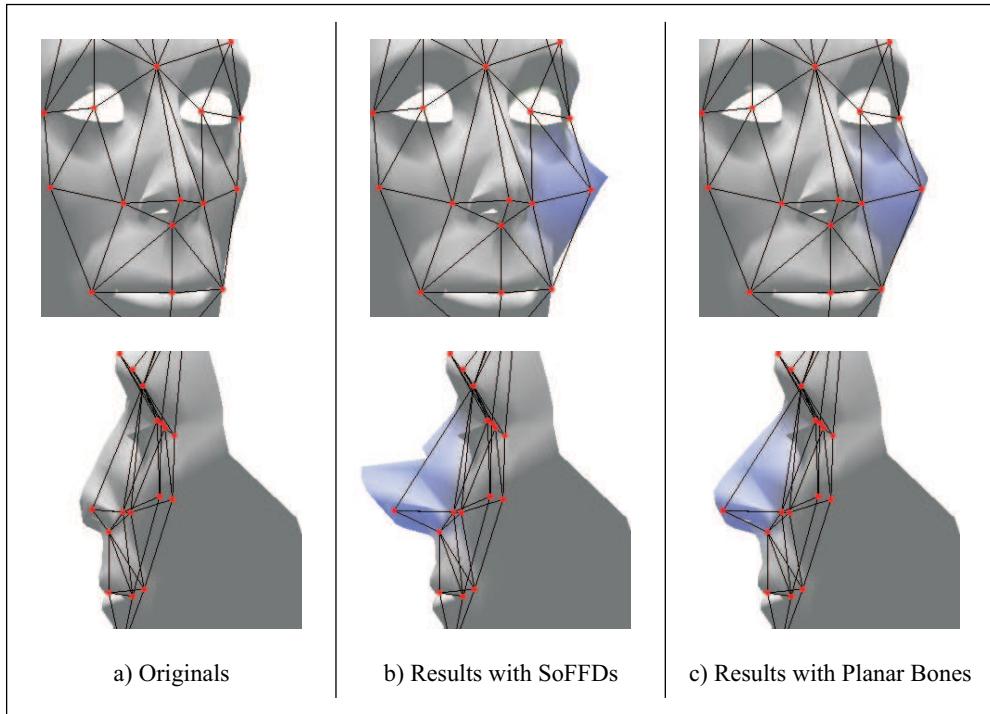
Fig. 3.15: Planar Bones versus Surface-oriented Free Form Deformations: using the same
examples presented before, we can see how Planar Bones (c) correct the de-
formation overshoots present in the application of SoFFDs (b).

requires knowing the projected image $\perp_j (\mathbf{v}_i)$ of any vertex $\mathbf{v}_i$ to be deformed by the
j^{th} facet of the control mesh. Similarly to the deformation itself, this computation is
done on a per-region basis. For apical regions $\mathbf{R_4}$, $\mathbf{R_5}$ and $\mathbf{R_6}$, such projections are
respectively $\mathbf{m}_l$, $\mathbf{m}_m$ and $\mathbf{m}_n$. Alternatively, in the case of edge boundary $\mathbf{R_1}$:

$$\forall \mathbf{v}_i \in \mathbf{R_1}, \perp_j (\mathbf{v}_i) = \frac{(\mathbf{v}_i - \mathbf{m}_l) \cdot (\mathbf{m}_m - \mathbf{m}_l)}{\|\mathbf{m}_m - \mathbf{m}_l\|^2} (\mathbf{m}_m - \mathbf{m}_l) + \mathbf{m}_l \qquad (3.32)$$

Regions $\mathbf{R_2}$ and $\mathbf{R_3}$ receive identical treatment after substituting the relevant mark-
ers in Equation 3.32. Finally, $\mathbf{R_0}$ recurs to the definition of facet normal n_j to compute
the projection:

$$\forall \mathbf{v}_i \in \mathbf{R_0}, \perp_j (\mathbf{v}_i) = \mathbf{v}_i - ((\mathbf{v}_i - \mathbf{m}_l) \cdot n_j) \, n_j \qquad (3.33)$$

For each vertex $\mathbf{v}_i$ and control facet f_j we compute the Euclidean distance with
respect to the corresponding $\perp_j (\mathbf{v}_i)$; if this is lesser than a specified radius, the contri-
bution of the control facet will be added to the final deformation $\mathbf{v}'_i$, modulated by the
relevant weighting function.

Similarly to Linear Bones, enforcing isometry at the level of individual control ele-
ments enables Planar Bones to overcome the limitations of SoFFDs, and makes them
suitable to be applied for the purposes of Facial Animation. This is demonstrated with

a) r = 0.25   b) r = 0.50   c) r = 2.50
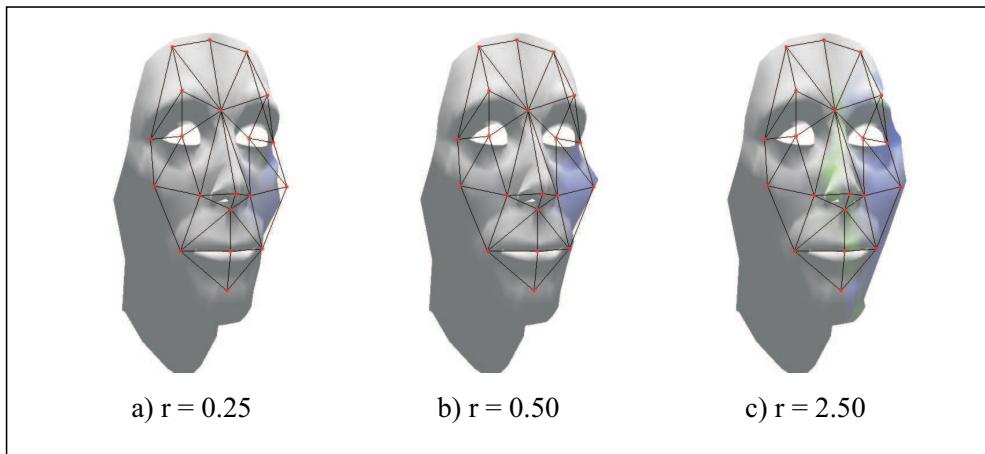
Fig. 3.16: With a very small radius of affection (a), the deformation by Planar Bones of the left cheek fails to capture most of the underlying geometry; choosing a more appropriate value (b) produces a smooth result, but overshooting it (c) causes the deformation to propagate across areas not related to the displaced feature.

the examples in Figure 3.15, where we can see how the correspondence between control mesh and warped surface is preserved throughout the deformation, enabling proper animation control, either from Motion Capture or simply manual movement of control points.

3.4.3   Application to Facial Animation

Apart from the correct performance of the deformation, there are some additional aspects to consider in order to make Planar Bones applicable to Facial Animation. Prominently, one of these aspects is deriving a proper definition of the affection volume of the control elements, which should necessarily encompass the whole facial mask to be animated.

In order to prevent interference between the affection regions of different control facets, it is sensible to introduce a unique common radius for the whole mesh. Deciding the correct value is a critical process, as Figure 3.16 illustrates. Small values may lead to parts of the surface not being encompassed by the deformation, causing a breach of continuity; conversely, excessively large radii revoke locality and cause the effects of several of the control facets to mutually interfere along large sections of the skin. While there is not a deterministic way to go through this process, a blind optimization technique such as midpoint [PTF92, GW04] can be used to iteratively approximate the minimum radius that guarantees full mask coverage.

Another aspect to bear in mind is an appropriate handling of facial discontinuities, specifically eye and mouth openings. As commented in preceding sections, many of the automatic approaches adopted by previous research in this topic (e.g. [KGM01, PP01]) are intrinsically flawed by their dependence on the target mesh's topology. In practical

terms, the boundary separation can be more easily achieved in texture space, given an appropriate texture projection. The approach relies on painting a texture mask marking grossly the geometry affected by the control facets on each side of the boundary (see Figure 3.17); this mask is later used to restrict the weighting computation for vertices affected by these facets, effectively tagging out those not in the relevant side of the opening.

This texturing approach turns out to be a light requirement, as defining and manipulating texture maps is a process intrinsic to the modelling, even when the sources are range scans or alternative geometry capture methods. Further masking like that performed by Pasquiarello and Pelachaud in [PP01] is not necessary, as the motion propagation scheme defined by interpolating motion along an approximating control mesh solves all the other issues that could have required this special treatment (e.g. separating the nose tip from the top lip).

a) Separation of upper and
lower eyelids

b) Separation of upper and
lower lips and jaw
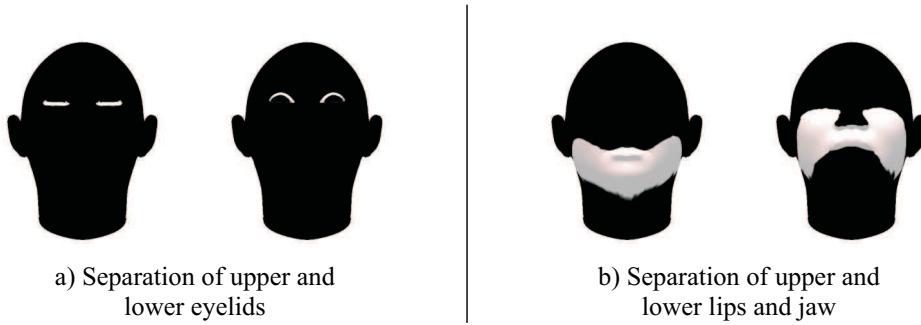
Fig. 3.17: Planar Bones: handling of discontinuities through texture mapping.

After tackling these two specific aspects, applying Planar Bones to Facial Animation is a straightforward process; the only requirement is translating the relevant marker set to the facial model to be deformed, and defining a control mesh topology spanning these points. Figure 3.18 illustrates how deformation is conducted with a control mesh connecting the marker set of MPEG-4 (c.f. Section 2.4.1), constructed as part of a animation system compliant with this standard (see [SM03]).

3.5 Bézier-Induced Deformation of Surfaces

In defining the Planar Bones approach, we have accomplished a suitable warping method for animation systems modelling facial expression through the position of marker points. This is in part due to the convenience of a surface-to-surface deformation paradigm, that suits especially well the problem of animating skin tissue. However, this reformulation comes at the expense of losing the continuity properties of the SoFFD approach, as the regionalized treatment of the affection volume only guarantees C_0 continuity across region boundaries. Fortunately, this does not undermine significantly the application of Planar Bones, since blending the combined effects of several control facets has a smoothing effect at these boundaries.
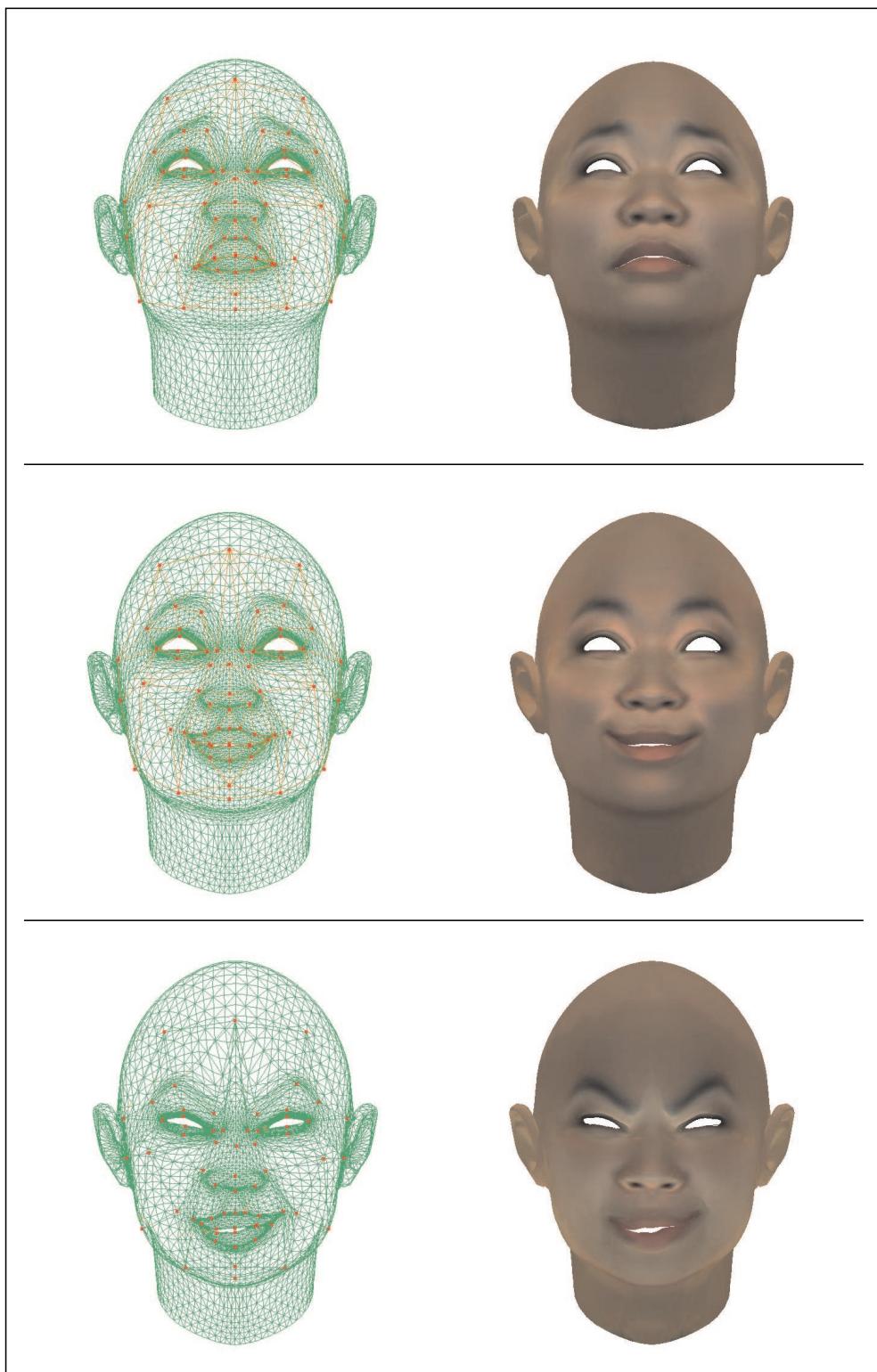
Fig. 3.18: Expressions created with Planar Bones using the MPEG-4 marker set. On the accompanying CD, see **Planar Bones - Static poses.avi** and **Planar Bones - MoCap.avi**

Nevertheless, the lack of a higher continuity may arise as a problem in very detailed face models (high polygonal count), therefore it would be desirable to have the ability of prescribing its degree arbitrarily. This is precisely the motivation behind developing Bézier-Induced Deformation of Surfaces (BIDS): enabling a surface-to-surface mapping of adjustable continuity that preserves the consistent correspondence that Planar Bones exhibited.

Such a mapping is constructed not over a control mesh, but over a piecewise parametric surface with triangular domains spanning the set of marker points. The existence of a smooth normal field defined over this control surface allows constructing a proper projective relation between it and the geometry to be deformed, instead of the discrete approach taken by SoFFDs and Planar Bones. By doing so, we enable a better approximation of the face model by the control structure that translates into a more adequate propagation of motion.

In order to present BIDS, we will start by defining the deformation at the level of a single control patch (Section 3.5.1); once given such a formulation, Section 3.5.2 will analyse its degree of continuity. Constructing a piecewise control structure will be dealt with in Section 3.5.3, while enforcing continuity across the piecewise boundaries will be the focus of Section 3.5.4. In addition, Section 3.5.5 will recapitulate on how continuity conditions affect the locality of the deformation, and Section 3.5.6 will consider other practical aspects in the application of BIDS.

3.5.1   Surface-to-surface mapping

BIDS operates through a projective relation that is initially evaluated between the polygonal mesh $\mathcal{M}$ we want to deform and a control surface $\mathcal{S}$, constructed as a piecewise arrangement of Bézier Triangles. Once the position of the markers change, spanning a new control surface $\mathcal{S}'$, the deformed image $\mathcal{M}'$ of the polygonal mesh is computed by reversing the initial mapping and evaluating it over $\mathcal{S}'$.

In order to describe the process of BIDS in more detail, let's first consider the application r, relating tuples of points of $\mathcal{S}$ and elements of the real line to a subset of $\mathbb{E}^3$ we will refer to as the *normal coverage* of $\mathcal{S}$:

$$r : \mathcal{S} \times \mathbb{R} \rightarrow C(\mathcal{S}) \subseteq \mathbb{E}^3$$
$$r(\mathbf{p}, h) = \mathbf{p} + h\, U(\mathbf{p}) \qquad (3.34)$$

taking U as the *unitary normal vector field* of $\mathcal{S}$.
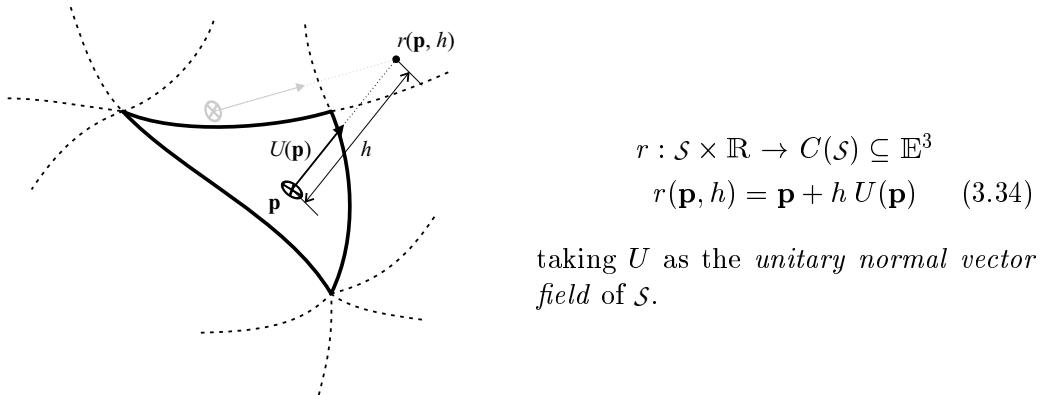
Depending on the shape of the control surface, there could be points of $\mathbb{E}^3$ onto

which no scaled normal vector of $\mathcal{S}$ is incident, causing its normal coverage $\mathscr{C}(\mathcal{S})$ not to encompass the whole Euclidean space. Also, it must be noted that r is not necessarily injective, since, as the diagram in Equation 3.34 illustrates (light grey), there can be different points of the control surface mapped onto the same element of $\mathscr{C}(\mathcal{S})$.

Bearing this in mind, we can define the inverse mapping of r, denoted by $p :$ $\mathscr{C}(\mathcal{S}) \to \mathcal{S} \times \mathbb{R}$, as the result of projecting the points of $\mathscr{C}(\mathcal{S})$ over the control surface along its normal field. Given the non-injective nature of r, the projection will be ill-defined at those points where its inverse mapping is not one-to-one.

Constraining $p(\mathbf{q})$ with $\mathbf{q} \in \mathscr{C}(\mathcal{S})$ to be the tuple with minimum magnitude for h partially solves this problem, disambiguating the projection for all points but those equidistant to different locii on $\mathcal{S}$. Yet we are really only concerned about the restriction of p to $\mathcal{M} \cap \mathscr{C}(\mathcal{S})$ (the portion of $\mathcal{M}$ mapped onto the control surface); therefore, this limitation will be relevant only if the equidistance condition is met on the polygonal mesh itself. Testing this by brute force would result in a rather intensive computation, but fortunately it can be done much more efficiently by just checking for intersections between $\mathcal{M}$ and approximations of the *medial surface* of $\mathcal{S}$ (see [Blo02] and [YBS03], as well as Figure 3.20).

Provided that the polygonal mesh is exempt of these singularities with respect to the control surface, the projection p thus defined describes a non-degenerate mapping of $\mathcal{M} \cap \mathscr{C}(\mathcal{S})$ onto $\mathcal{S}$. Also, its inverse r (henceforth referred to as r' when evaluated on $\mathcal{S}'$) allows reconstruction of the deformed image of $\mathcal{M}$ over different configurations of the control mesh it was originally projected onto. Thereby, the deformation of a vertex $\mathbf{v}_i$ into its image $\mathbf{v}_i'$ can be summarised in the following procedure:
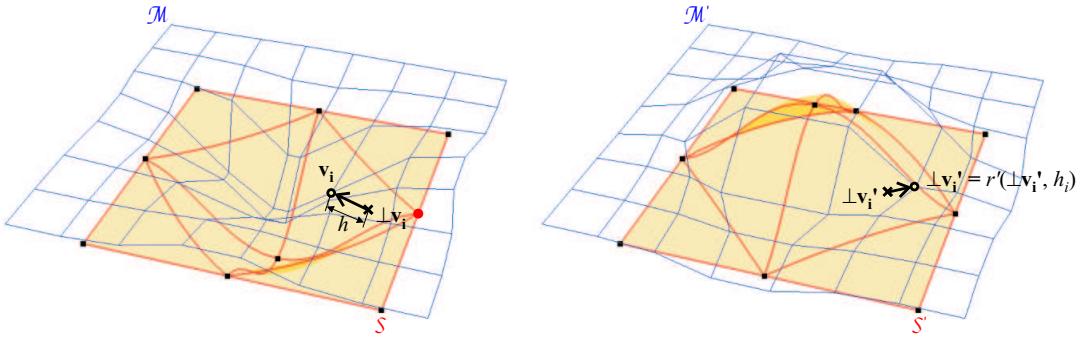
$$
\begin{aligned}
\{\perp \mathbf{v_i}, h_i\} &= p\left(\mathbf{v_i}\right) \\
\perp \mathbf{v_i'} &= \mathbf{s}_j'\left(\mathbf{s}_j^{-1}\left(\perp \mathbf{v_i}\right)\right) \\
\mathbf{v_i'} &= r'\left(\perp \mathbf{v_i'}, h\right) = \perp \mathbf{v_i'} + h\, U'(\perp \mathbf{v_i'})
\end{aligned}
\tag{3.35}
$$

Where $\mathbf{s}_j$ and $\mathbf{s}_j'$ are the parametric expressions for the neutral and deformed configurations, respectively, of the Bézier Triangle in $\mathcal{S}$ where the initial projection of $\mathbf{v_i}$ lies on. These expressions result from the recursive de Casteljau's algorithm (introduced along
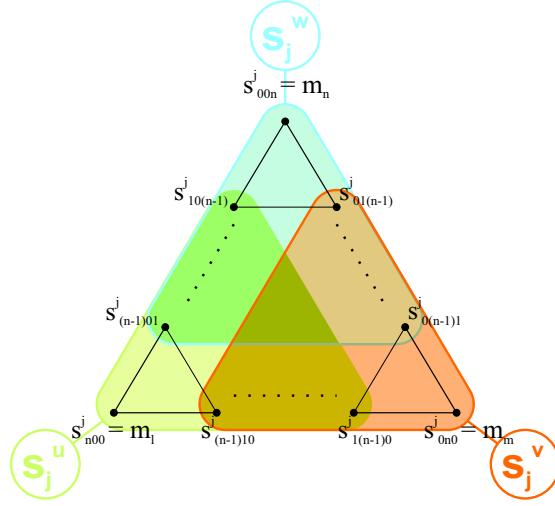
Fig. 3.19: Labelling of Bézier points and subpatch classification.

with FFDs in Section 3.3.3); borrowing the definition of trivariate Bernstein polynomials from Equation 3.20, we define the biparametric patch of degree n as:

$$\mathbf{s}_j(u,v) = \sum_{p,q,r}^{n} \mathbf{s}_{pqr}^{j} \; B_{pqr}^{n}(u,v,1-(u+v)) \tag{3.36}$$

Here, u and v stand for the two first barycentric coordinates on the triangular domain of the jth control patch, spanning markers $\{\mathbf{m}_l, \mathbf{m}_m, \mathbf{m}_n\}$. The control point lattice follows a completely different arrangement from the FFD case. Taking the notation used by Farin in [Far95], Bézier points $\mathbf{s}_{pqr}^{j}$ are labelled as in Figure 3.19. This diagram also shows the layout of the control nets for subpatches $\mathbf{s}_j^u$, $\mathbf{s}_j^v$ and $\mathbf{s}_j^w$, corresponding to de Casteljau's recursive step previous to the computation of $\mathbf{s}_j$. Given the recursive nature of Bézier Triangles, patch derivatives themselves are a linear expression on these $n-1$ degree subpatches, yielding the following parametric expression for the unitary normal map U_j:

$$U_j(u,v) = \frac{\frac{d\mathbf{s}_j}{du}(u,v) \wedge \frac{d\mathbf{s}_j}{dv}(u,v)}{\left|\frac{d\mathbf{s}_j}{du}(u,v) \wedge \frac{d\mathbf{s}_j}{dv}(u,v)\right|} \text{ where } \begin{cases} \frac{d\mathbf{s}_j}{du} = n\left(\mathbf{s}_j^u - \mathbf{s}_j^w\right) \\ \frac{d\mathbf{s}_j}{dv} = n\left(\mathbf{s}_j^v - \mathbf{s}_j^w\right) \end{cases} \tag{3.37}$$

The deformed counterpart U_j' of Equation 3.37 is trivially computed by substituting the corresponding markers in the expressions of patches and subpatches.

Assuming that U_j is at least C^0 continuous over the patch domain, we can compute the parametric coordinates of the initial projection of the vertex $\mathbf{v_i}$ onto $\mathbf{s}_j$ (noted as $\mathbf{s}_j^{-1}(\perp \mathbf{v_i})$ in Equation 3.35) by simply finding $\{u,v\} = \arg\min(\|\mathbf{v_i} - \mathbf{s}_j(u,v)\|)$. For non-degenerate patches, this is equivalent to finding a root for the set of non-linear equations given by:

$$[\mathbf{v_i} - \mathbf{s}_j(u,v)] \; J\left(\mathbf{v_i} - \mathbf{s}_j(u,v)\right) = 0 \tag{3.38}$$
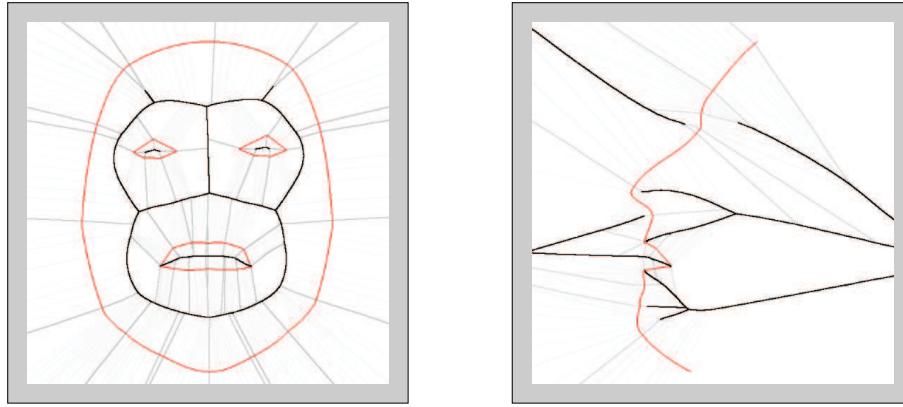
Fig. 3.20: Approximation of medial curves (in black) by concatenation of contiguous Voronoi cells. Given the solution for the Voronoi problem in $\mathbb{E}^3$, the construction of medial surfaces is analogous.

where $J(f)$ denotes the 3×2 Jacobian matrix of f. Given the regularity of Bézier Triangles for low patch degrees, relatively simple numeric schemes can rapidly converge to an accurate solution provided a sensible initialization (e.g. derived from a coarse sampling of the patch). Appendix A gives a more detailed account of the methods adopted for this particular problem.

We have now described the computational aspects of Equation 3.35. However, it is yet to be seen how this formulation can produce a deformation of arbitrary continuity. Also, the control surface needs to be defined in terms of both its patch topology and internal Bézier points. These two aspects, warp continuity and the construction of the control surface, are intrinsically related, as the following sections describe in further detail.

3.5.2 Continuity of the deformation

By construction, the parametric expression $\mathbf{s}_i$ for any Bézier Triangle in $\mathcal{S}$ is differentiable over the interior of its (barycentric) domain of definition $\mathcal{D}_i$, provided that its control net is non-degenerate. Therefore r is also differentiable over $\mathcal{D}_i^o \times \mathbb{R}$. Under these conditions, it can also be proved that r is regular for all those points in its domain whose image lies out of the medial surface of $\mathbf{s}_i$ (henceforth $\mathscr{E}(\mathcal{S}_i)$, with $\mathcal{S}_i = \mathbf{s}_i(\mathcal{D}_i^o)$). Consequently, by application of the inverse mapping theorem, p is a C^∞ diffeomorphism over $\mathscr{C}(\mathcal{S}_i) \setminus \mathscr{E}(\mathcal{S}_i)$. Hence we can conclude that the composition implicit in Equation 3.35 is ultimately differentiable over this same domain.

Whenever several triangular patches exert a combined deformation, p restricts the region actually affected by each of them by ruling out intersections amongst different $\mathscr{C}(\mathcal{S}_j)$. With the polygonal mesh guaranteed not to intersect $\mathscr{E}(\mathcal{S})$, we still have to verify whether the deformation is continuous across the boundaries shared between regions corresponding to adjacent Bézier Triangles. Assuming a uniformly varying normal over $\mathcal{S}$, this verification becomes equivalent to assessing continuity for those points being
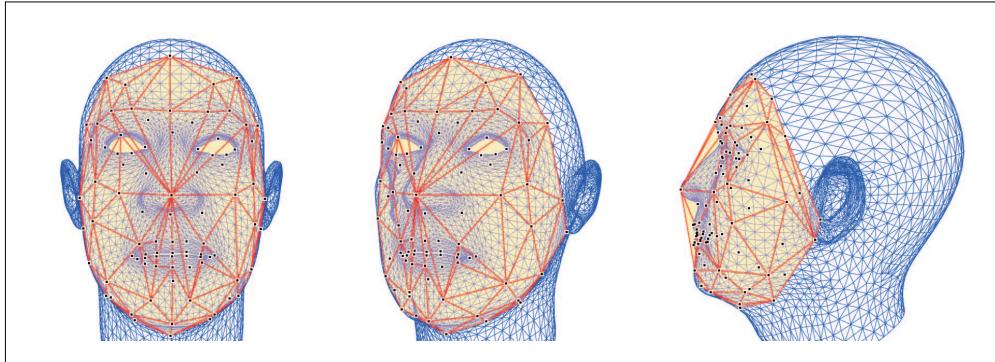
Fig. 3.21: Patch topology of the control surface derived from Voronoi tetrahedralization of the marker points (in black). Domains spanning both points on the tip of the nose and chin are evidence that the result of this method is not necessarily compatible with the connectivity of the skin surface.

projected onto boundaries between adjacent patches.

Since these boundaries are themselves images of the boundary of their respective parametric domains, we need a common parameterisation of the incident patches to study the degree of continuity for the components of Equation 3.35. Enforcing geometric continuity of degree k (G^k) means that adjacent patches can be reparameterised so that the resulting expression is C^k continuous (see [Boe88] and [Pet02]). This would also lead to a C^{k-1} definition of the normal vector field, implying that p is diffeomorphic of class C^{k-1} over the combined domain corresponding to the incident patches. Following identical reasoning to that used for a single patch, we can finally ascertain that the whole deformation attains this same degree of continuity.

Summarizing, for a control surface where adjacent Bézier Triangles meet with G^k continuity, the deformation induced over $\mathcal{M} \cap \mathcal{C}(\mathcal{S})$ will be at least C^{k-1} continuous, provided that $\mathcal{M} \cap \mathcal{E}(\mathcal{S}) = \emptyset$. This will determine the way Bézier control nets will be constructed over an arrangement of triangular patches whose topology is yet to be defined. The procedure followed for this definition is described in the next section.

3.5.3 Constructing the topology of triangular domains

For a given set of markers points on $\mathcal{M}$, a triangulation has to be found so that the resulting polyhedron best approximates this polygonal mesh, while preserving the connectivity implicit in it. Each of the triangles will then serve as parametric domains for the corresponding $\mathbf{s}_j$ patches that make up the control surface $\mathcal{S}$. In the case of Planar Bones, this problem was given a manual solution, mostly because the majority of the work carried out with this system uses the relatively coarse MPEG-4 marker set (see Figure 2.2). However, in our new deformation approach, it is sensible to aim for an automatic method since the approximating nature of the control surface plays a much more important role. Also, the density of markers used may be much higher in order to
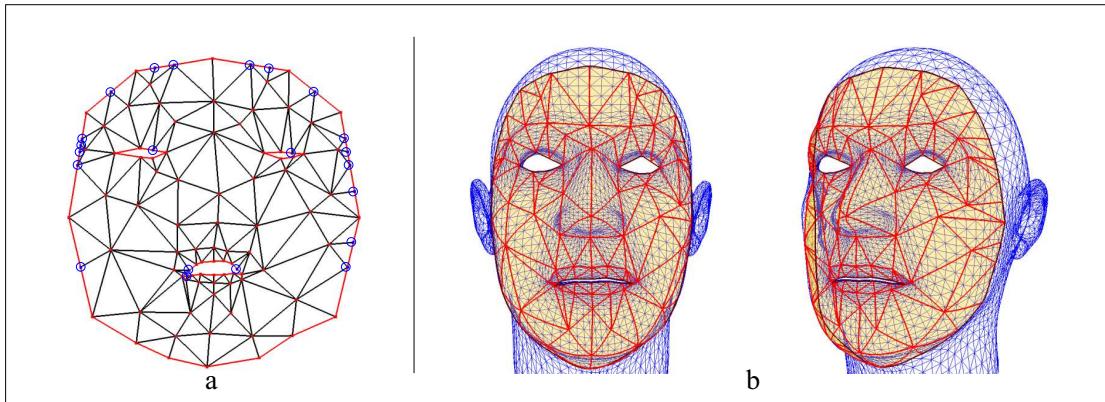
Fig. 3.22: a) Delaunay triangulation of the projected markers, highlighting the vertices inserted while enforcing countours (mouth and eye openings, along with the control mask boundary). b) Resulting control surface with the same topology, overlayed onto the model whose deformation it drives.

provide more accurate animation.

A general solution to the triangulation of any unstructured point cloud in $\mathbb{E}^3$ could be found by applying Voronoi tetrahedralization [Bow81, Wat81] and then selecting the facets which form the exterior hull of the resulting polyhedron. However, as Figure 3.2 illustrates, this would only produce the boundary of the convex hull of the cloud, which is not necessarily consistent with the connectivity of $\mathcal{M}$. Specifying constraints in the tetrahedralization would not be any less laborious than giving the topology by hand, thus this whole approach does not result in any practical automation. Fortunately, there are particular properties of the context we are exploring that allow a less generic and more effective solution.

In the particular case of a human face, cylindrical or ellipsoidal projections of $\mathcal{M}$ onto $\mathbb{E}^2$ generally carry little or no metric distortion along most of the skin surface. This means that the triangulation problem can be solved by simply applying Delaunay triangulation on the Euclidean plane. Whenever suitable projections cannot be found (e.g. when dealing with cartoon or animal faces), a sensible alternative is to use iterative relaxation approaches, such as the one demonstrated by Viaud and Yahia in [VY92]. Additionally, to enforce symmetry in the deformation, we may choose to mirror the triangulation produced for the markers on one half of the face over the other. In practice this is only necessary in specific regions such as the nose and eyebrow arch, where the metric is mostly distorted by the projection.

In order to mimic the connectivity of $\mathcal{M}$, we should be able to specify boundaries so that openings in the mesh can be represented on the control surface (and consequently regarded as discontinuities by the deformation). This is done by defining closed polylines connecting the markers that lie on the corresponding boundary; the resulting segments are preserved in the triangulation by means of forced insertion of midpoints in the Delaunay mesh, as described in [Law77b]. Then the orientation of boundaries is used to

remove the triangles that lie inside the openings, through a simple propagation scheme on the graph defined by facet adjacency.

However, the addition of new points to the set of markers, as part of the segment-preserving triangulation process, brings up the issue of how to provide control input for them throughout the deformation process, consistently with the rest of points whose displacements will be known. Our solution is to parameterise them over the Bézier curve corresponding to the domain edge that is split by their insertion, and derive their location by a top-down evaluation of the hierarchy of edge partitions, computing the midpoint of the curve corresponding to each segment. This is possible since edge curves are constructed prior to the patches themselves, as discussed in the next section.

3.5.4 Conditioning of the control surface

Section 3.5.2 related the degree of continuity of Equation 3.35 to the geometric continuity on the control surface itself. In fact, we can assert G^n continuity by conditioning the control networks of patches assembled according to the topology prescribed in the previous section. These conditioning procedures are a recurrent topic in Surface Fairing, and, in general, in Computer-Aided Design. Peters [Pet90] provides an extensive classification of methods solving this problem for the particular case concerning BIDS: piecewise surfaces composed of triangular Bézier patches. We will now describe those relevant to the current implementation of this deformation technique, enforcing in particular G^1 continuity.
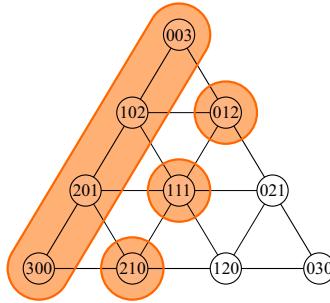
Fig. 3.23: Indexation for the b_j Bézier points of a cubic patch network.

The first aspect to solve is finding the minimum patch degree that may allow for such a conditioning. Taking the uw boundary as referent (with Bézier points labelled as in Figure 3.24), we can demonstrate without loss of generality that cubic degree alone is not sufficient. According to [Far95], vertex conditioning requires coplanarity of all Bézier points incident to the same marker, and since we cannot assume regularity of the topology, these tangent planes have to be predefined. This ultimately reduces the degrees of freedom of Bézier points $\{\mathbf{b}_{210}, \mathbf{b}_{120}, \mathbf{b}_{021}, \mathbf{b}_{021}, \mathbf{b}_{102}, \mathbf{b}_{102}\} \in \mathbb{E}^3$ to two instead of three. Also, the equations yielded by boundary conditions involve these vertices:
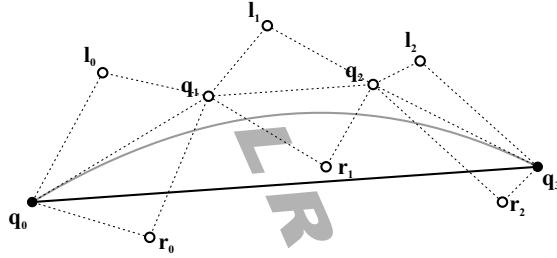
$$
\begin{aligned}
(\alpha_0 \mathbf{l}_0 + (1 - \alpha_0)\mathbf{r}_0) - (\beta_0 \mathbf{q}_0 + (1 - \beta_0)\mathbf{q}_1) \quad &= 0 \\
1/3\left((\alpha_1 \mathbf{l}_0 + (1 - \alpha_1)\mathbf{r}_0) - (\beta_1 \mathbf{q}_0 + (1 - \beta_1)\mathbf{q}_1)\right) & \\
+2/3\left((\alpha_0 \mathbf{l}_1 + (1 - \alpha_0)\mathbf{r}_1) - (\beta_0 \mathbf{q}_1 + (1 - \beta_0)\mathbf{q}_2)\right) \quad &= 0 \\
2/3\left((\alpha_1 \mathbf{l}_1 + (1 - \alpha_1)\mathbf{r}_1) - (\beta_1 \mathbf{q}_1 + (1 - \beta_1)\mathbf{q}_2)\right) & \\
+1/3\left((\alpha_0 \mathbf{l}_2 + (1 - \alpha_0)\mathbf{r}_2) - (\beta_0 \mathbf{q}_2 + (1 - \beta_0)\mathbf{q}_3)\right) \quad &= 0 \\
(\alpha_1 \mathbf{l}_2 + (1 - \alpha_1)\mathbf{r}_2) - (\beta_1 \mathbf{q}_2 + (1 - \beta_1)\mathbf{q}_3) \quad &= 0
\end{aligned}
\tag{3.39}
$$

Since $\mathcal{S}$ can be a closed 2-manifold, no assumptions can be made regarding boundary topology (apart from 0 or 1 neighbours). Bearing this in mind, the linear system in Equation 3.39 presents 12 equations with 28 unknowns, reduced to 22 by the tangent plane conditions at the markers. Therefore, on a per-edge basis, continuity conditioning is underconstrained and, after applying additional strategies (e.g. introducing more constraints), solvable. However, the equations for each of the three edges also involve Bézier points on the oposite edges (see Figure 3.23, orange shading). Taking into account the multiplicity of each of the unknowns in the three concurrent systems, the number of independent variables per edge drops to 10. This makes the whole construction overconstrained, and in the general case, incompatible.

The existing literature presents two alternatives to this problem: either increasing the degree by blending three different patches conditioned independently along each edge, or adding additional degrees of freedom in the form of new vertices splitting each of the patches' domains. In the implementation of BIDS, the splitting alternative is more convenient, for two main reasons:

- the stability of the numerical methods developed alongside this deformation technique benefits from a lower polynomial degree, and

- when treating hardware acceleration (see Chapter 6), memory resources (registers) are overabundant, allowing for the resulting micro-patches to be treated at the same time without significant costs, while improving performance with respect to higher degree computations.

In particular, the partitioning method adopted by BIDS is the classic Clough-Tocher scheme [CT65], which produces three micro-patches by splitting each triangular domain at the barycentre. This results in another instance of the problem for which both the position and tangent plane of the newly added vertex are undetermined, leveraging the balance between constraints and unknowns in the equations for the macro-patch. In
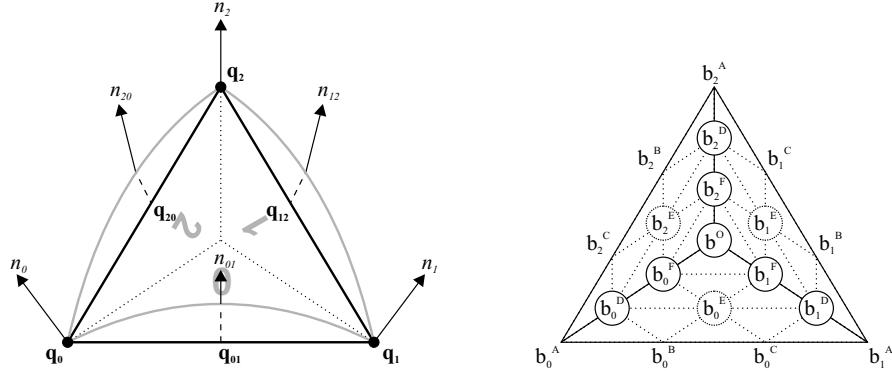
Fig. 3.24: Labelling of elements involved in the Clough-Tocher partitioning scheme.

order to handle this partitioning, together with continuity conditioning amongst neighbouring macro-patches, we will be adopt the following notation:

The introduction of new degrees of freedom allows for the boundary curves of the macro-patch to be specified prior to the construction of the rest of the network. This is a requirement of the procedures outlined in Section 3.5.3, in order to enable a boundary-preserving triangulation that reproduces facial discontinuities. The intermediate Bézier points of each of these cubic curves are computed as the result of Equation 3.40.
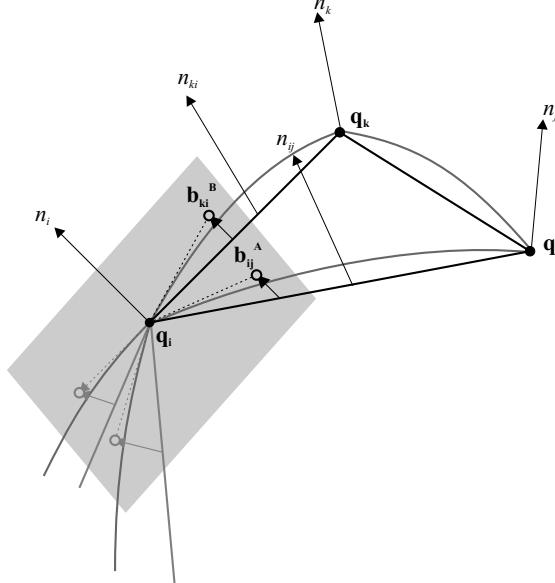
$$\mathbf{b}_{ml}{}^{B} = \mathbf{b}_{lm}{}^{A} = \frac{1}{3}\left(2\,\mathbf{q}_l + \mathbf{q}_m + \frac{(\mathbf{q}_m - \mathbf{q}_l)\cdot n_l}{n_{lm}{}^{A}\cdot n_l}\,n_{lm}{}^{A}\right) \tag{3.40}$$

where the 2nd order interpolated normals $n_{ml}{}^{A}$ and $n_{lm}{}^{B}$ are given by:

$$n_{ml}{}^{B} = n_{lm}{}^{A} = B_0^2\left(\frac{1}{3}\right)n_l + B_1^2\left(\frac{1}{3}\right)n_{lm} + B_2^2\left(\frac{1}{3}\right)n_m$$

with B_i^j standing for the ith univariate Bernstein polynomial of degree j (c.f. Equation 3.20). Vertex and edge normals (n_m and n_{ml}, respectively) are computed by averaging facet normals on the polyhedron constructed in Section 3.5.3.

As for the remaining elements of the layout in Figure 3.24, we will start by characterising the $\mathbf{b}_i^D$s. Let's first define auxiliary points $\mathbf{q}_i^D$ with quadratically interpolated normals $\mathbf{n}_i^D$:

$$
\begin{aligned}
\mathbf{q}_i^D &= \sum_j \left(\frac{1}{9} + \frac{6}{9}\delta_{ij} \right) \mathbf{q}_j \\
n_i^D &= \sum_{l,m,n} n_{lmn} \, B_{lmn}^2 \left(\frac{1}{9} + \frac{6}{9}\delta_{i0}, \frac{1}{9} + \frac{6}{9}\delta_{i1}, \frac{1}{9} + \frac{6}{9}\delta_{i2} \right)
\end{aligned}
$$

where δ_{lm} stands for Dirac's delta. Bézier points n_{ijk} used for the normal interpolation are chosen as the polyhedral vertex and edge normals from Figure 3.24 (left), indexed according to the criteria of Figure 3.23. Using these results, we can derive the corresponding Bézier points $\mathbf{b}_i^D$ consistently with vertex boundary conditions:

$$
\mathbf{b}_i^D = \frac{(\mathbf{q}_i - \mathbf{q}_i^D) \cdot n_i}{n_i^D \cdot n_i} n_i^D + \mathbf{q}_i^D \tag{3.41}
$$

With these constraints, the linear system in Equation 3.39 can be used to rule the conditioning on the edge boundaries of the macro-patch, as the choice of $\mathbf{b}_i^D$s makes it compatible. This binds Bézier points $\mathbf{b}_j^E$ in the patches on both sides of the edge; should any polyhedral boundary be present, conditioning will proceed identically by just applying symmetry.

The remaining elements of the Bézier networks assembled by the partitioning are the $\mathbf{b}_i^F$s and the central point $\mathbf{b}^O$, highlighted in the diagram below. With the constraints imposed by the previously fixed points, the three instances of Equation 3.39 posed by the edges incident to $\mathbf{b}^O$ yield 27 equations, relating the three remaining points, that amount to 9 unknowns, to be added to six others intrinsic to the equations themselves. While this may look incompatible, some of the constraints are indeed redundant as the binding equations form a cyclic system. After discriminating independent constraints, the assembled system admits a unique solution:
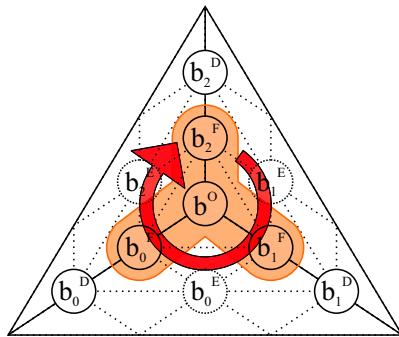
$$
\begin{aligned}
\mathbf{b}_i^F &= \frac{\mathbf{b}_{i-1}^E + \mathbf{b}_i^A + \mathbf{b}_i^E}{3} \\
\mathbf{b}^0 &= \frac{\mathbf{b}_0^F + \mathbf{b}_1^F + \mathbf{b}_2^F}{3}
\end{aligned} \tag{3.42}
$$

where additions in the subindices are to be taken modulus 3.

Equation 3.42 is also consistent with vertex continuity conditions at $\mathbf{b}^O$, as coplanarity is something that trivially follows from the assembly of edge conditioning equations. This completes the construction of each of the macro-patches composing the control surface $\mathcal{S}$ on which the operation of BIDS is based, and by re-evaluating the same G^1 conditioning equations for the construction of $\mathcal{S}'$, C^0 continuity can be asserted for the whole deformation algorithm. Additionally, visual continuity can be improved at little additional cost by applying the procedures outlined in Section 3.5.6.

Should a greater degree of continuity be necessary, a range of blending and splitting strategies can be used to construct G^n piecewise surfaces with $n > 1$. Apart from the extensive survey given in the aforementioned [Pet90], both Farin [Far86] and Velktamp [Vel92] provide an interesting overview of conditioning techniques, with [Vel92] focusing specially on splitting approaches.

3.5.5   Implications on locality

While the main focus of BIDS is achieving a better fitting of the deformed surface and attaining a gradable degree of continuity, the locality of the warp cannot be disregarded. As discussed in the previous section, in particular in Equation 3.40, the conditioning scheme used relies on knowing the tangent plane of the surface at every marker. As usual, the latter is computed by averaging the normals of incident facets from the polyhedron constructed in Section 3.5.3. Implicitly, this expands the effects of displacing one marker from the set of incident macro-patches to all those adjacent to any of them, hence the degree of locality attained with the present G^1 scheme.

Should we provide a higher degree of continuity for the algorithm, by raising that in the control surface to G^2, curvature information would be needed at every marker. The approximated normal at connected markers could then be used to compute this data, and as a result locality would expand one more level across the topology of adjacent patches. This illustrates the balance existing between these two aspects of the deformation, each acting in detriment of the other.

3.5.6   Application to Facial Animation

Similarly to what occurred with Planar Bones, there are additional considerations needed to put BIDS into practice in the context of Facial Animation. This is so not only for the sake of enabling proper deformation, as for instance separating the mesh along facial boundaries, but also to enhance the overall quality of the results whenever handling much more dense representations of facial skin (in comparison to those tackled with Planar Bones).

Visual fairing

The G^1 conditioning approach taken in the current implementation of BIDS guarantees differentiable deformation along most of the extents of the target mesh, apart from those vertices whose projection ($\perp \mathbf{v_i}$ in Equation 3.35) falls in a ϵ-neighbourhood of the boundary between two or more patches. In these cases continuity may degrade down

a) plain distance-based separation   b) aided with normal-based selection
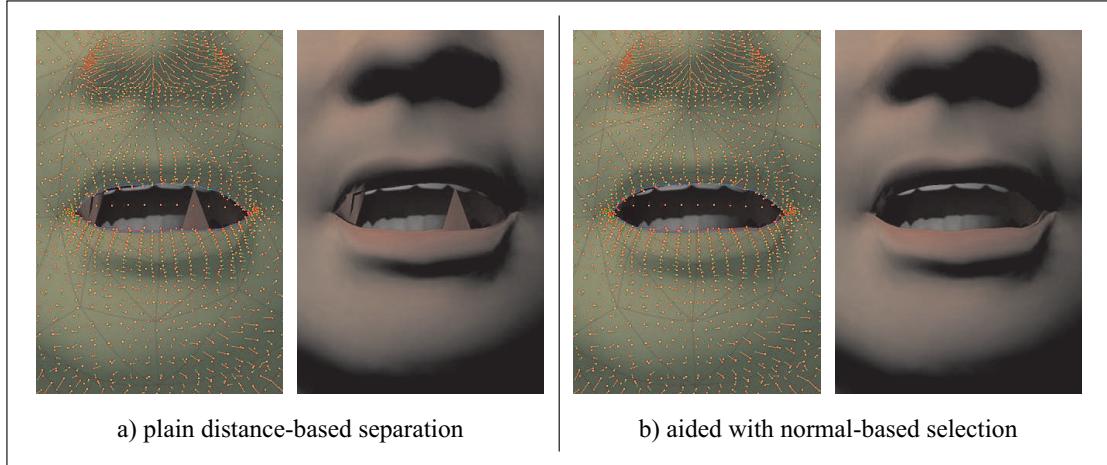
Fig. 3.25: Boundary separation with (b) and without (a) considering coallignment with the normal field of the control surface. (a) clearly exhibits some misclasiffied vertices, which result in polygons ocluding the mouth.

to C^0. In order to mitigate the visual seams that this might create, BIDS deforms the normal field of the surface with the same parameterisation guaranteeing that its resulting image is identically continuous. More explicitly, using the same notation as in Equation 3.35:

$$n'_i = n_i \left(\begin{bmatrix} \frac{d\mathbf{s}_j}{u} \\ \frac{d\mathbf{s}_j}{v} \\ U_j \end{bmatrix}^{-1} \begin{bmatrix} \frac{d\mathbf{s}'_j}{u} \\ \frac{d\mathbf{s}'_j}{v} \\ U'_j \end{bmatrix} \right)^{-T} \Bigg|_{\{u_i, v_i\}} \quad \text{with } \{u_i, v_i\} = \mathbf{s}_j^{-1} \left(\perp_j \mathbf{v_i} \right) \qquad (3.43)$$

The rules of transformation in Equation 3.43 (inverse-transpose) respond not to the tangent space of $\mathbf{s}_j$, but to its dual subspace where the normal field belongs (c.f. [DP90]).

This additional consideration for the deformation of the normal field comes at almost no additional cost with respect to the original formulation (since the involved terms are already computed), and provides smooth visual results like those demonstrated in Figure 3.28.

Boundary separation

One of the most laborious aspects of using Planar bones for Facial Animation is, without doubt, having to produce texture maps to separate facial discontinuities. Now that BIDS has removed the burden of manually specifying the topology of the control structure (see Section 3.5.3), it would be equally desirable to be able to perform boundary separations automatically.

The approximation of the target mesh that the BIDS control surface can provide outclasses its Planar Bones counterpart, yet it does not necessarily provide a good

a) original   b) smoothed countours
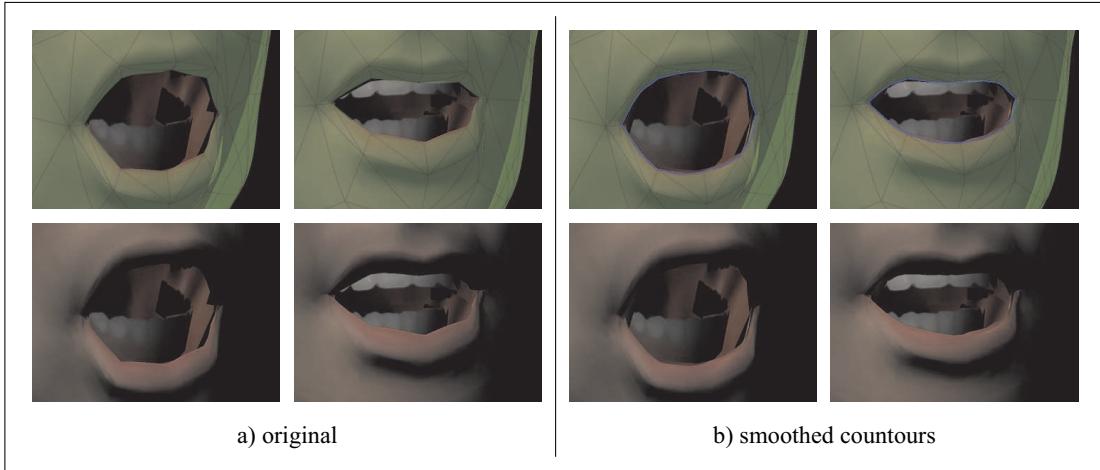
Fig. 3.26: Results of enforcing countour smoothness for facial boundaries, detailing both original and corrected configurations of the control surface (top row) and their respective results on the deformed mesh (bottom).

enough fitting for separating boundaries on a distance criterion alone. However the control structure of BIDS also supports a continuous normal field that can be used as additional criteria for this classification: for any vertex of $\mathcal{M}$ in the neighbourhood of a boundary, several candidate projections are to be considered. Prior to applying any distance-related criteria, all projections whose surface normal diverges more than a threshold angle with respect to the vertex normal itself are to be discarded, and then the procedure follows as usual.

Provided that lip boundaries (e.g. upper and lower lips) have almost opposite normal fields, this method easily automates the only remaining manual process involved in this deformation method, as Figure 3.25 demonstrates.

Contour Fairing

While the conditioning scheme presented in Section 3.5.4 guarantees that the control surface is at least G^1 continuous, it does not necessarily produce smooth contours for domain boundaries that are in reality curved, like for instance the borders of upper and lower lips. When applied to the deformed mesh, this issue causes a breach in the curvature of regions projected onto domain triangles along these boundaries, as illustrated in Figure 3.26.a.

In order to prevent these perceptible flaws, we can make use of some of the unexploited degrees of freedom in the G^1 conditioning scheme, in particular those that allow for macro-patch boundary curves to be specified prior to the construction of the rest of the Bézier network. In every domain edge, intermediate control points $\mathbf{b}_{lm}^A$ and $\mathbf{b}_{lm}^B$ (c.f. Figure 3.24) are only constrained by the tangent planes at points $\mathbf{m}_l$ and $\mathbf{m}_m$. Therefore, for each marker $\mathbf{m}_j$ forming part of a contour line connecting it to $\mathbf{m}_i$ and

$\mathbf{m}_k$, intermediate control points $\mathbf{b}_{ij}^B$ and $\mathbf{b}_{jk}^A$ can be conditioned as follows:
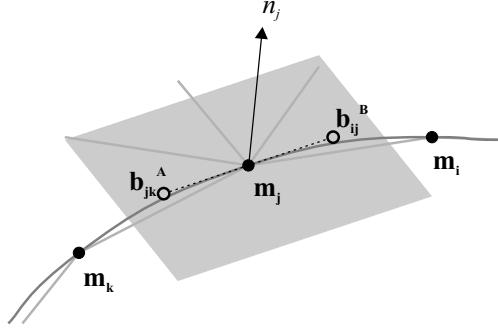
$$
\begin{aligned}
\left(\mathbf{b}_{jk}^A\right)^1 &= \mathbf{m}_j + \left(\left(\left(\mathbf{b}_{jk}^A\right)^0 - \mathbf{m}_j\right) \cdot (\mathbf{o}_k - \mathbf{o}_j)\right)\ \frac{(\mathbf{o}_k - \mathbf{o}_j)}{\|\mathbf{o}_k - \mathbf{o}_j\|^2} \\
\left(\mathbf{b}_{ij}^B\right)^1 &= \mathbf{m}_j + \left(\left(\left(\mathbf{b}_{ij}^B\right)^0 - \mathbf{m}_j\right) \cdot (\mathbf{o}_i - \mathbf{o}_j)\right)\ \frac{(\mathbf{o}_i - \mathbf{o}_j)}{\|\mathbf{o}_i - \mathbf{o}_j\|^2} \qquad (3.44)
\end{aligned}
$$

where $\left(\mathbf{b}_{lm}^R\right)^0$ and $\left(\mathbf{b}_{lm}^R\right)^1$ stand for the original and corrected positions of the corresponding Bézier points, initially computed as in Equation 3.40. Auxiliary points $\mathbf{o}_i$ and $\mathbf{o}_i$ are derived in the following manner:

$$
\begin{aligned}
\mathbf{o}_i &= \mathbf{m}_j - \left((\mathbf{p}_i - \mathbf{p}_j) \cdot n_j\right)\ n_j \\
\mathbf{o}_j &= \mathbf{m}_j - \left((\mathbf{p}_k - \mathbf{p}_j) \cdot n_j\right)\ n_j
\end{aligned}
$$

The sequence of vertices forming part of each smooth contour has to be specified prior to the construction of the patch topology, and they need to be placed along one of the enforced boundaries. Yet they are not equivalent, as some of these boundaries have corners (e.g. eyes) that are not smooth. Figure 3.26.b shows the results of applying this method to the two independent contours of the lips.

Adaptive tessellation

While the range of deformation involved in Facial Animation is relatively small, the application of BIDS may produce warped configurations of the original mesh in which its local smoothness and detail is compromised, as the topology of $\mathcal{M}$ may not be suitable for representing the deformed face. In order to maintain the quality of the discrete approximation of the skin surface by the polygonal mesh, we need to enforce curvature preservation throughout the deformation, so that polygonal detail is consistent with the bending of the skin whenever is deformed.

Preserving curvature can be achieved by mesh tesellation, using an error metric that is consistent with the change of local curvature (see for instance [ON66] and [SZL92]). For this purpose, together with BIDS we utilise the well-known 1 to 3 edge-based subdivision scheme [Hop96], that operates by iteratively selecting candidate edges for splitting, and then remeshing the affected facets according to one of the following three cases (reduced from seven by symmetry):
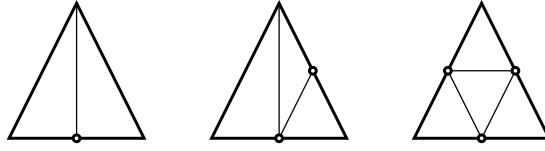
The error metric ruling the selection and splitting of mesh edges has to represent the change of curvature in the mesh, assumed to be locally equivalent to a smooth 2-manifold (and not globally because of the facial discontinuities). Provided this assumption is true, the curvature on any point $\mathbf{p}$ of $\mathcal{S}$ can be expressed as a quotient of the two first fundamental forms in a given direction $v_{\mathbf{p}}$ of its tangent space:

$$k(v_{\mathbf{p}}) = \frac{II(v_{\mathbf{p}})}{I(v_{\mathbf{p}})} \tag{3.45}$$

Both II and I are extrinsic measurements that depend on the metric of the space in which $\mathcal{S}$, however their quotient does not. $I(v_{\mathbf{p}})$ is computed using the metric tensor G of such space, while $II(v_{\mathbf{p}})$ is derived from the shape operator $S(v_{\mathbf{p}})$:

$$
\begin{aligned}
I(v_{\mathbf{p}}) &= v_{\mathbf{p}} \, G \, v_{\mathbf{p}}^{T} \\
II(v_{\mathbf{p}}) &= S(v_{\mathbf{p}}) \cdot v_{p}
\end{aligned}
$$

The shape operator represents the variation of the normal field in a given direction of the tangent space $(S(v_{\mathbf{p}}) = -D_v(U(\mathbf{p})) = -J(U)|_{\mathbf{p}} \, v_{\mathbf{p}})$; computing it will require a local parameterisation of $\mathcal{M}$, and finding a suitable one is not trivial since $\mathcal{M}$ is provided as a polygonal mesh. However, assuming that the curvature of the mesh is not exaggerated (as it occurs for human faces), we can make use of a gross discrete approximation for the whole second fundamental form that does not require any parametric definition. Provided that the embedding space is plainly $\mathbb{E}^3$, for any edge e connecting points $\mathbf{m_i}$ and $\mathbf{m_j} \in \mathcal{M}$, $II(e_{\mathbf{m_i}})$ can be discretely approximated as:

$$II(e_{\mathbf{m_i}}) = \Delta_e(U(\mathbf{m_i})) \cdot e_{\mathbf{m_i}} = (U(\mathbf{m_j}) - U(\mathbf{m_i})) \cdot (\mathbf{m_j} - \mathbf{m_i})$$

and similarly

$$I(e_{\mathbf{m_i}}) = (\mathbf{m_j} - \mathbf{m_i}) \cdot (\mathbf{m_j} - \mathbf{m_i})$$

In Figure 3.27 edges of positive curvature (concave regions) are plotted in green, while their negatively-curved counterparts (convex regions) are coloured red. In order to provide a sign-independent error metric, we can take the following ratio:

$$err(e') = \frac{k(e'_{\mathbf{m_i'}}) - k(e_{\mathbf{m_i}})}{k(e_{\mathbf{m_i}})} \tag{3.46}$$

where $k(e'_{\mathbf{m_i'}})$ stands for the curvature measured in the deformed configuration of e.

The iterative process driving the tessellation will then split edges with an error metric larger than a given threshold and remesh until no more edges are found. Literally, for a threshold value of $th = 0.4$, this means that all edges which experience a change in
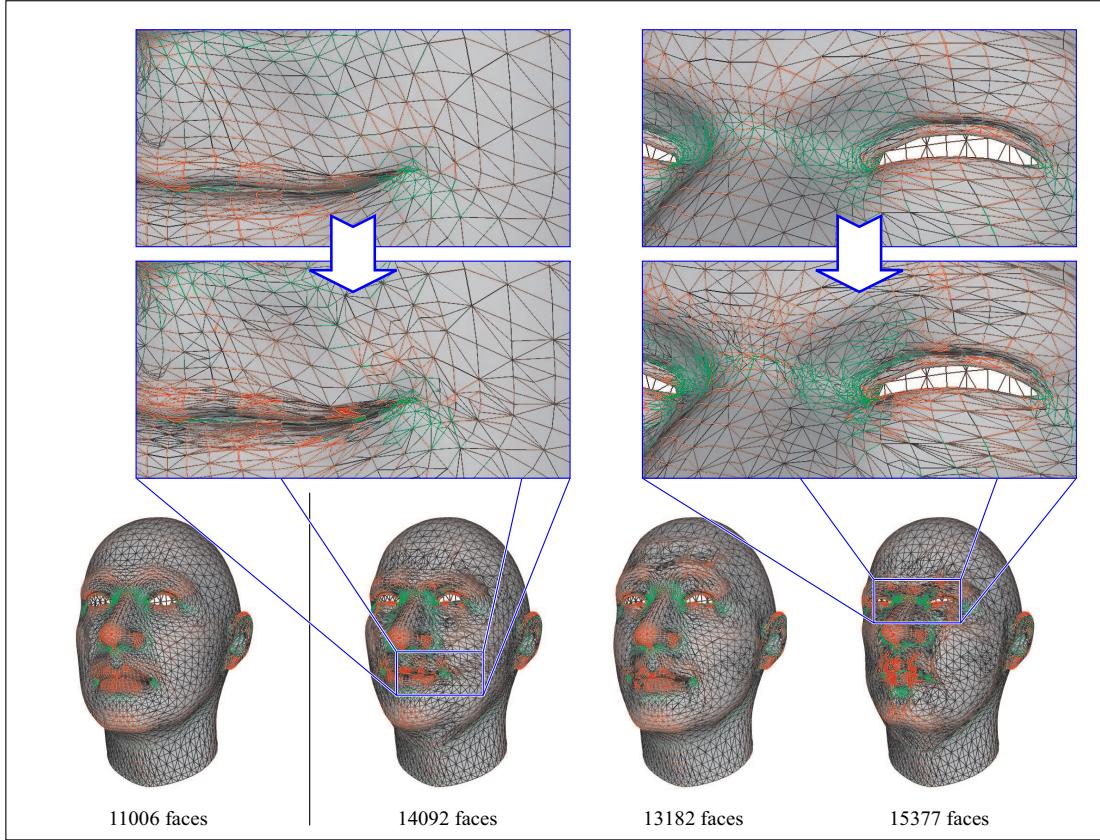
Fig. 3.27: Adaptive tessellation coupled with the deformation that transforms the initial mesh (left) into the expressions on the right. Red and green colouring in the edges denotes positive and negative bending of the normal field, respectively. Insets demonstrate the subdivision in areas of the mesh where the warp exerted significantly distorts local curvature.

curvature larger than 40% with respect to their original configuration will be partitioned. Since this tessellation process runs adaptively in parallel with the animation, special considerations have to be taken in order to maintain a smooth flow of the geometry without *popping* effects. This is achieved by including an error metric term in the equations deriving the new vertices for the split edges:

$$\mathbf{m}'_{ij} = \frac{err(e')}{th}(\mathbf{m}'_{ij})^1 + \left(1 - \frac{err(e')}{th}\right)(\mathbf{m}'_{ij})^0 \tag{3.47}$$

where

$$(\mathbf{m}'_{ij})^0 = \tfrac{1}{2}\mathbf{m}'_i + \tfrac{1}{2}\mathbf{m}'_j$$
$$(\mathbf{m}'_{ij})^1 = \mathbf{m}'_i B_0^3\left(\tfrac{1}{2}\right) + \mathbf{b}_{ij}^{A'} B_1^3\left(\tfrac{1}{2}\right) + \mathbf{b}_{ij}^{B'} B_2^3\left(\tfrac{1}{2}\right) + \mathbf{m}'_j B_3^3\left(\tfrac{1}{2}\right)$$

and

$$\mathbf{b}_{ij}^{A'} = \left(\frac{2\mathbf{m}_i' + \mathbf{m}_j'}{3} + \frac{n_{ij}^{A'} \cdot n_i'}{\|n_{ij}^{A'}\|} \; n_{ij}^{A'} \right) \; \text{with } n_{ij}^{A'} = \;\; \tfrac{2}{3}\mathbf{n}_i' + \tfrac{1}{3}\mathbf{n}_j'$$

$$\mathbf{b}_{ij}^{B'} = \left(\frac{\mathbf{m}_i' + 2\mathbf{m}_j'}{3} + \frac{n_{ij}^{B'} \cdot n_j'}{\|n_{ij}^{B'}\|} \; n_{ij}^{B'} \right) \; \text{with } n_{ij}^{B'} = \;\; \tfrac{1}{3}\mathbf{n}_i' + \tfrac{2}{3}\mathbf{n}_j'$$

In addition, the normal at the new marker has to be identically interpolated for the variation in lighting throughout animation to be smooth:

$$n_{ij}' = \frac{err(e')}{th}(n_{ij}')^1 + \left(1 - \frac{err(e')}{th} \right)(n_{ij}')^0 \tag{3.48}$$

where the midpoint normal $(n_{ij}')^0$ is taken as $\frac{1}{2}n_i' + \frac{1}{2}n_j'$, and its counterpart $(n_{ij}')^1$ is the averaged normal of incident facets after the insertion of $(\mathbf{m}_{ij})^1$. This does not necessarily give a unitary normal in all cases, but Equation 3.47 is robust against this.

This edge splitting process gives way to a reorganization of the mesh that also inserts auxiliary edges to preserve triangular topology. These newly-added edges will all require a measurement of their original curvature in order to be able to compute the corresponding error metrics in successive iterations of the tessellation; for this purpose we must be able to refer edges, and in particular their composing vertices, to the mesh topology they originate from. This is solved by using a hierarchical mesh structure that enables such back-tracking, at the cost of implicitly storing the whole process.

Handling this data structure in real time is possible given the relative simplicity of the 1 to 3 partitioning scheme, allowing for a certain regularity (e.g. original edge lines are preserved albeit subdivided, and each child face has a single parent in the original topology). Alternative techniques, like instance $\sqrt{3}$ subdivision [Kob00] allow for better quality meshing; however many of the invariants that facilitate an efficient implementation of the adaptive tessellation are no longer present. Figure 3.27 illustrates their application with a threshold value of 0.4, focusing in on conflict areas like the nasolabial furrow and eye contour.

3.6   Summary

This Chapter has presented a variety of warping algorithms, classifying them with respect to the approach they take to the deformation of facial skin, either by simulating its physical behaviour or using generic geometric warping techniques to convey similar results. Table 3.1 relates some of them to the classification of models of expression introduced in Chapter 2. Sections 3.4 and 3.5 introduced two novel geometric approaches, Planar Bones and BIDS, which exploit a surface-to-surface map in order to be able to reconstruct skin deformation from a reduced set of markers. This paradigm produces good results in mimicking the actual behaviour of the tissue, and enables the use of highly detailed meshes in Facial Animation systems, providing visually plausible results with reasonable computational costs. Additionally, special considerations are taken to maintain the quality of the mesh representation of the skin throughout the deformation.
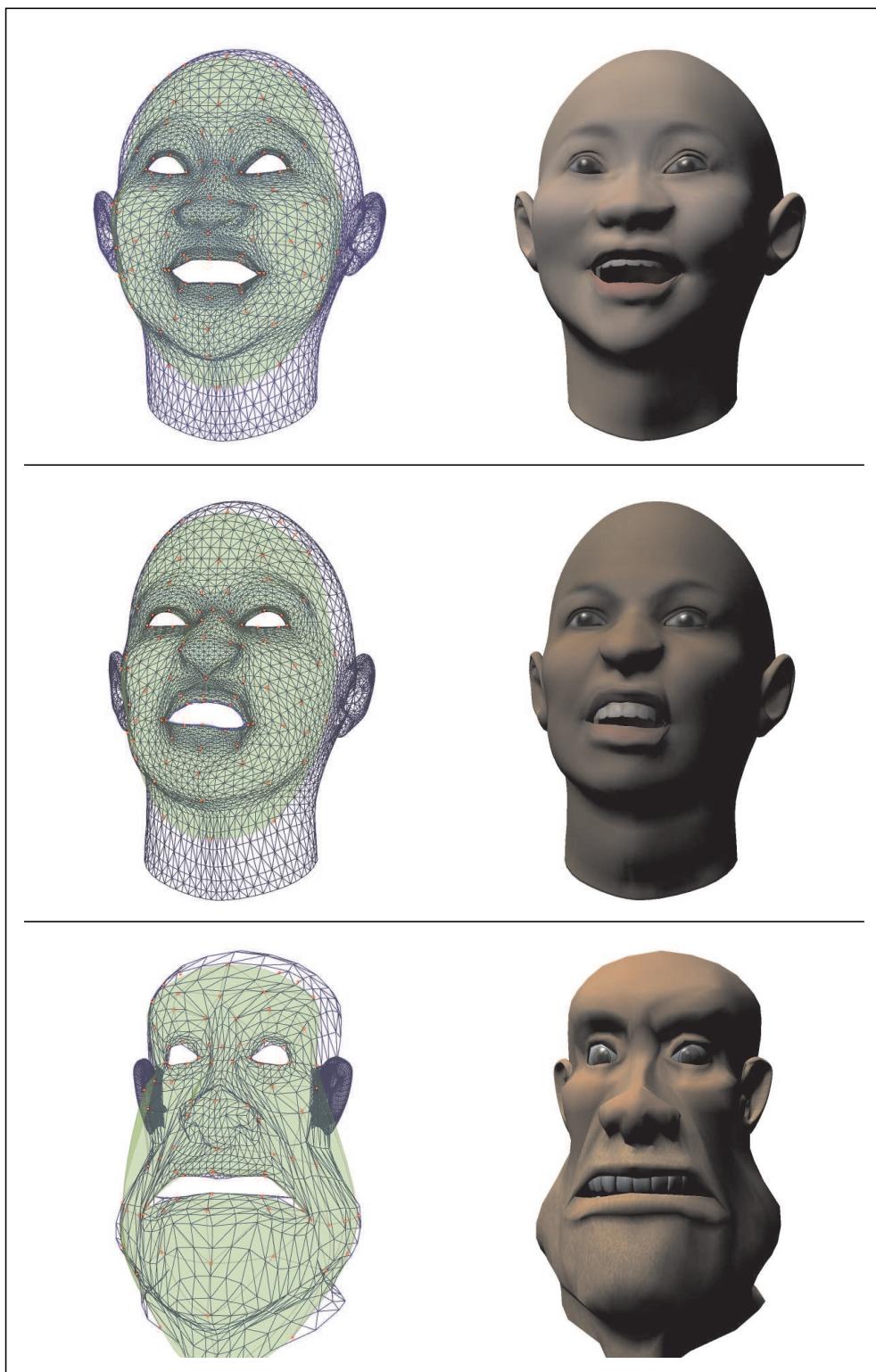
Fig. 3.28: Examples of the application of BIDS with different faces. On the accompanying CD, see **BIDS - MoCap retargetting.avi** and **BIDSwrinkling - EA MoCap.avi**

Tab. 3.1: Relation between some of the deformation techniques described in this Chapter and the model of expression they support.

	Physically-based deformation	Geometric warps
Appearance-based models of expression		
temporally discrete	Lee et al. [LTW93]	Parke [Par78], Pighin[Pig99], Joshi et al.[JTD03]
spatially discrete	Pitterman and Munhall [Pm01]	Williams [Par78], Kokkevis and Singh[KS01]; Planar Bones and BIDS
Anatomically-based models of expression		
	Platt and Badler[PB] Lee et al. [LTW95], Koch et al [KGB98]	Waters[Wat87]

The marker-based approach employed to control these two geometric techniques is a particular embodiment of spatially-discrete models of expression (as described in Chapter 2, yet not generic enough to be controlled by the input captured from a physiognomy different from that being animated. Chapter 4 focuses on providing the necessary techniques to make this possible, by enabling a generalization of Facial Motion Capture. Some of the steps involved in this process depend heavily on the nature of the mapping driving BIDS, creating plenty of references to the contents of the present chapter. This mapping becomes the foundation on which the rest of the thesis is built.

Furthermore, the fine-scale mechanical behaviour of facial skin that geometric approaches oversee will be the main consideration of Chapter 5, aiming to complement the visual results of Planar Bones and BIDS with alternative techniques yielding even more realism.

4. ADAPTING FACIAL MOTION CAPTURE

4.1 Introduction

As Section 2.4.2 described, the adaptability of a spatially-discrete model of expression is not as straightforward as for its anatomically-based counterparts, since the motion of tracked facial features varies largely amongst different physiognomies. In the particular case of Motion Capture using optical markers (c.f. [Wil90, GGW98]), this is an especially acute problem, as by default there is not an inherent structure to the cloud of tracked points that may allow anatomical generalizations. Since performance-based animation methods are one of the principal focuses of this thesis, the current chapter will be dedicated to presenting a solution framework for this problem.

Retargeting Facial MoCap onto a different physiognomy requires not only finding the new point of application of the displacement vector prescribed by every captured marker, but also its scale and direction, represented by the transformation between the tangent spaces of source and target markers. The first of these requirements is normally fulfilled by manually labeling the target face with the marker set used for the capture, but there is not such a trivial translation between their respective tangent spaces.

The MPEG-4 standard [MPE01] proposes a scheme based on anthropomorphic measurements (Facial Animation Parameter Units - see Figure 2.3) that are used to selectively scale vertical and horizontal motion of its particular set of marker points (Facial Definition Parameters - see Figure 2.2). Yet this scaling is purely linear, and more importantly takes place only on the frontal plane of the face; this blatantly ignores the dimensions and shape of its profile, showing that this standard was conceived mainly as a two-dimensional framework, and is not suitable to be used otherwise. Cohen and Massaro [CMC02] follow a practically equivalent procedure.

Assuming that the distribution of markers is dense enough to describe the differences between both physiognomies, we can find more versatile and accurate approaches in higher-order techniques. In [NN01], Noh et al. approach this problem by fitting a polygonal model to the performer's model through Radial Basis Functions centered at the marker points, and then building a linear approximation of the transformation between tangent spaces using the discrete topology incident to the vertices associated to each marker point. This produces a motion mapping that is non-linear with respect to the point of application; however, it also linearises the correspondence between the spatial evolution of source and target markers, disregarding the damping and distortion that occurs as consequence of the differences between their corresponding physiognomies (e.g. lip corners being pulled against cheeks of different size and shape).

The framework presented in this Chapter aims to overcome the limitations of the
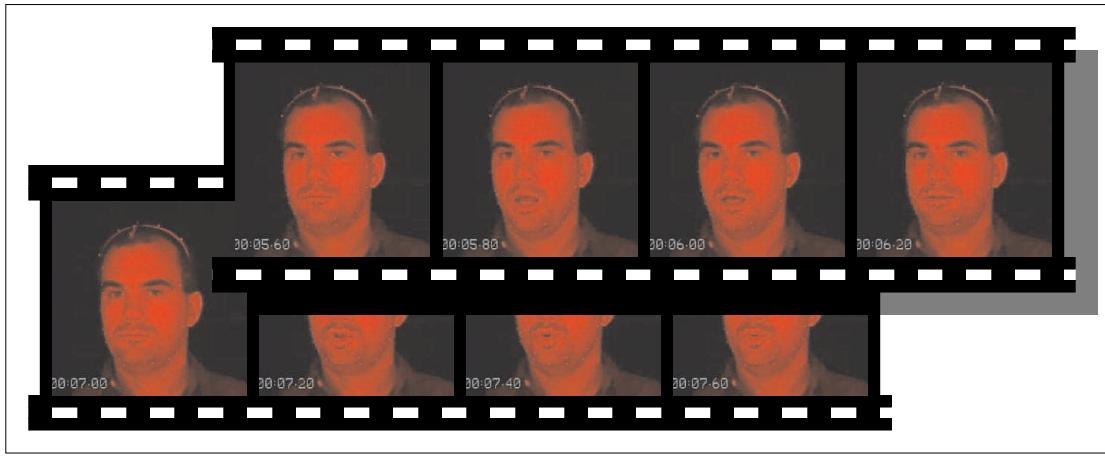
Fig. 4.1: Some of the video sequences supporting the optical tracking of marker points (courtesy of Scott King).

aforementioned approaches by representing the non-linearities of motion mapping, both with respect to the point of application and to the range and direction of displacement. The methods used to this effect (described in Section 4.3) are especially sensitive to the quality of the MoCap data input to the framework, Section 4.2 will introduce additional tools to regularise it, by filtering and reconstructing the motion signal and by separating the components of rigid motion and soft tissue deformation.

4.2   Pre-processing of Facial Motion Capture data

Optical Motion Capture devices [WB97, Opt, Vic] provide a convenient framework for tracking point features arbitrarily distributed over the surface of facial skin. The increasing resolution of Charge Coupled Device (CCD) cameras, together with the use of specific light frequencies and the corresponding reflective materials has allowed for increasingly small markers to be tracked, yielding non-intrusive systems that permit the capture of natural speech and gesture (see for instance [Vic]). However the tracking also becomes much more sensitive, leading to the need for more robust algorithms in order to prevent occlusions and marker misidentifications.

In addition, MoCap data represents not only the local deformation of the facial tissue, but also the spatial motion of the whole head as a rigid body. As outlined in the introduction to this Chapter, it is possible to retarget local deformation because the MoCap markers provide enough information about the dimensions of the performer's face. However, the rigid body motion of the skull remains exclusive to the physiognomy of the performer, since the captured streams do not provide information of the full body structure that could enable the corresponding retargeting. Therefore it is also necessary to provide a way of separating these two components of the markers' motion prior to any further analysis.

4.2.1   Reconstruction of missing fragments

One of the problems derived from the limitations in tracking is the absence of information for some of the markers during particular time segments, either because of unresolved occlusions or lack of resolution to identify markers in a particular orientation.

In order to compensate for these flaws and reconstruct the missing information, we can assume some degree of regularity in the signal represented by the tracked coordinates of each of the missing markers. Then a frequency space transform, like the Discrete Cosine Transformation (DCT, see [ANR74, RY90]), can be applied in a window on both sides of the missing interval $[t_0, t_1)$. For a window of size N, this yields the following characteristic frequencies:

$$\forall k \in \{0..N-1\}, \lambda_k = \sqrt{\frac{1 + \delta_{k0}}{N}} \sum_{i=0}^{N-1} s[w_0 + i] \cos\left(\frac{(2\,i+1)\,\pi\,k}{2\,N}\right) \tag{4.1}$$

where s is the vector of samples for the tracked coordinate, and index w_0 denotes the lower bound of the window interval, equating respectively to $t_0 - N$ and t_1 for the spans to the left and right of the fragment. Taking λ_k^0 and λ_k^1 as the frequency components corresponding to each of these intervals, we can resample the missing fragment by just interpolating the M most characteristic components and extending the resulting signal to fill the gap:

$$\begin{aligned} s[t_0 + i] = \quad & \left(1 - \frac{i}{t_1 - t_0}\right) \sum_{k=0}^{M-1} \sqrt{\frac{1 + \delta_{k0}}{N}}\, \lambda_k^0 \cos\left(\frac{(2\,i+1)\,\pi\,k}{2\,N}\right) + \\ & \frac{i}{t_1 - t_0} \sum_{k=0}^{M-1} \sqrt{\frac{1 + \delta_{k0}}{N}}\, \lambda_k^1 \cos\left(\frac{(2\,(N-i)+1)\,\pi\,k}{2\,N}\right) \end{aligned} \tag{4.2}$$

By choosing $M < N$ we are actually performing a low pass filtering, that attempts to single out the periodic structure of the prefix and suffix windows of the missing fragment, while discarding the noise contained in higher bands. This approach is valid for small segments, yet it can not be used for processing the whole signal; this is caused by the eventual presence of locally strong high frequency components, appearing either as a product of marker misidentification, or because of the vocal articulation itself. Whenever filtering out these high components, the remaining spectrum of the signal exhibits bounce-back effects, as portrayed in Figure 4.3. Instead, we shall adopt another filtering scheme that allows for swift motion to be properly represented, while preventing spurious spikes. The next section introduces a process that is consistent with the dynamic properties of the markers' motion.

4.2.2   Dynamics-based filtering

The differential model of the motion of a given marker $\mathbf{m}_i$ trivially admits the following finite difference approximation:

$$\begin{bmatrix} \mathbf{m}_i(t) \\ \dot{\mathbf{m}}_i(t) \end{bmatrix} = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{m}_i(t - \delta t) \\ \dot{\mathbf{m}}_i(t - \delta t) \end{bmatrix} + \begin{bmatrix} \delta t^2 \\ \delta t \end{bmatrix} \ddot{\mathbf{m}}_i(t - \delta t) \tag{4.3}$$
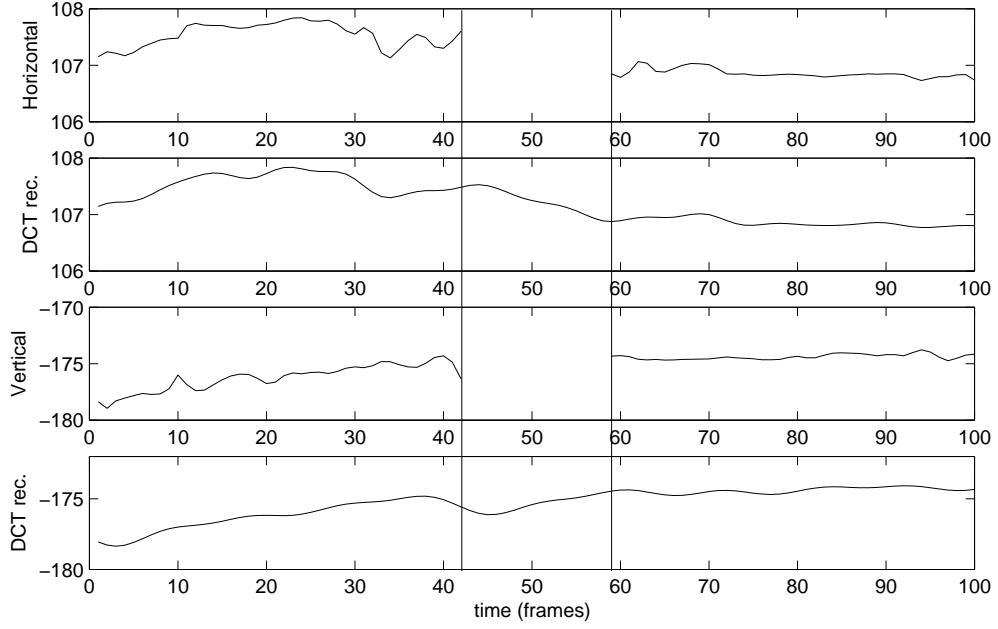
Fig. 4.2: DCT-based reconstruction of a missing fragment in the tracking of a marker point lying on the left corner of the lip; the size of the window is $N = 50$, and the number of characteristic frequencies M is taken as $0.2\,N$.

where $\dot{\mathbf{m}}_u = \frac{\partial \mathbf{m}_u}{\partial t}$ and $\ddot{\mathbf{m}}_u = \frac{\partial^2 \mathbf{m}_u}{\partial t^2}$; δt stands for the time interval used by the capture equipment, which in our particular case is $1/120^{th}$ of a second.

Equation 4.3 provides a state space definition of the relevant marker, represented by its position and speed; this relates to the coordinates $\tilde{\mathbf{m}}_i$ tracked by the optical system as:

$$\tilde{\mathbf{m}}_i = H\,s(t) + n_m, \text{ where } H = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } s(t) = \begin{bmatrix} \mathbf{m}_i(t) \\ \dot{\mathbf{m}}_i(t) \end{bmatrix} \tag{4.4}$$

taking n_m as the meassurement noise, assumed to follow a Gaussian distribution of zero mean and variance R. Similarly, we can consider the second term of Equation 4.3 as zero-mean Gaussian noise in the state transformation, with covariance matrix Q.

Using the framework just outlined, we can apply Kalman filtering [Kal60, May79] to obtain a position estimate that minimizes the measurement error while maintaining consistence with a predefined range of marker accelerations. Rather than a conventional filter, this technique is actually a predictor-corrector scheme for the evaluation of the state model, that dynamically updates the error covariance according to the mispredic-tion. Taking A as the state transition matrix from Equation 4.3, predicted estimates for the state vector $s(t) = [\mathbf{m}_i(t), \dot{\mathbf{m}}_i(t)]^T$ and its corresponding error covariance matrix $P(t)$ can be computed as follows:

$$\begin{aligned} \hat{s}^-(t) &= A\,\hat{s}(t - \delta t) \\ P^-(t) &= A\,P(t - \delta t)\,A^T + Q \end{aligned} \tag{4.5}$$

Using the actual measurement of the marker's position, these estimates can now be updated to reflect the divergence:

$$\begin{aligned} \hat{s}(t) &= \hat{s}^-(t) + K(t)\left(\mathbf{m}_i - H\,\hat{s}^-(t)\right) \\ P(t) &= \left(I - K(t)\,H\right) P^-(t) \end{aligned} \tag{4.6}$$

where the correcting factor $K(t)$, known as the *Kalman gain*, is given by the following expression:

$$K(t) = P^-(t)\,H^T\left(H\,P^-(t)\,H^T + R\right)^{-1}$$

Such a factor is chosen to minimize the matrix norm of the corrected error covariance $P(t) = E[(s(t) - \hat{s}(t))^T\,(s(t) - \hat{s}(t))]$ (for a full derivation see [May79, Jac93]).

Providing values for the covariance matrices of measurement and process errors is on its own quite a sensitive matter, as neither do we known such error metrics for the capturing hardware, nor can we provide a predefined range of facial accelerations for the state model.

However, some gross generalizations can be made in order to approximate these values. Assuming that perceptible frame rates are in the order of 20 frames per second, clusters of 6 frames can be averaged and down-sampled to a single one; then, measurement error can be roughly approximated with respect to the subsampling (taking the deviation from that average). Similarly, speed and acceleration can be computed through discrete approximations in order to give an expression for the second term of Equation 4.3. Taking as a reference an especially fast-paced MoCap stream, the rounded up results obtained by this process are:

$$Q \approx 1E^{-11}\begin{bmatrix} 0.6\ m^2 & 1.2\ m^2/s \\ 1.2\ m^2/s & 2.5\ m^2/s^2 \end{bmatrix};\ R \approx 2.2E^{-7}\ m^2$$

In practice, no more sophisticated methods are required to compute the covariance values, as the results in Figure 4.3 demonstrate. Additional details on the use of Kalman filtering for other Motion Capture contexts can be found in [WB97, WB01], but for the present case the constructed estimation framework proves sufficient.

4.2.3   Estimation and removal of Rigid Body Transformation

The estimation of rigid motion of the head is a well-studied topic for generic frameworks [EDP94, BY95]; to some extent, we simplified this problem by using a rigid marker array (called a *jig*) placed onto the top of the performer's skull. However, the shifting of the scalp tissue caused by movement in the forehead, together with the intrinsic uncertainty of the marker tracking, means that this tool alone is insufficient. Therefore it is necessary to construct a richer estimate by considering additional markers that remain predominantly static under local deformation of the facial tissue, like those placed behind the sideburns or along the base of the neck.

While the Rigid Body Transformation (RBT) exhibited by these markers may not be consistent across the chosen subset, it is possible to define an optimal approximation under the criteria of Mean Square Error (MSE). The parameterisation used to represent
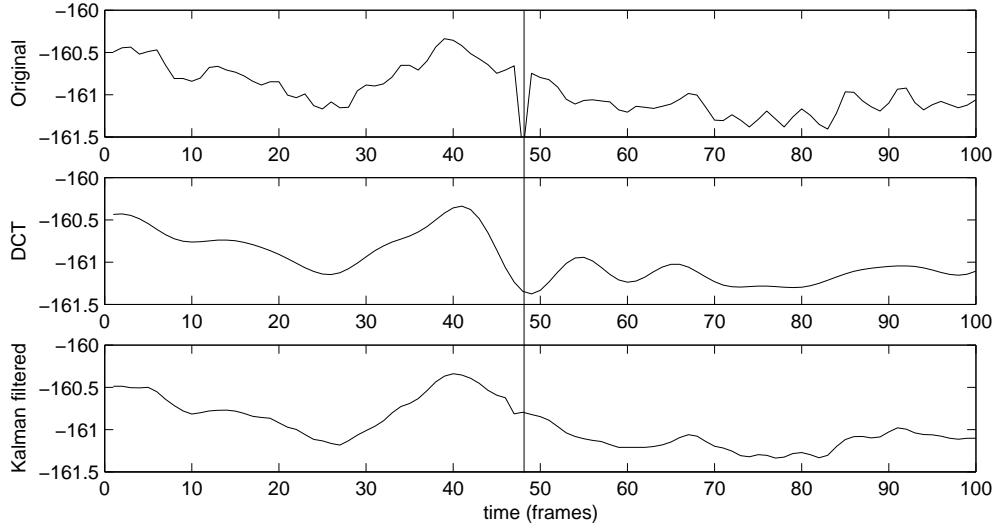
Fig. 4.3: Treating a spike with different filtering approaches: DCT lowest frequencies (20% in the plot) actually track the spurious variation, creating an oscillation in both of its sides, while Kalman filtering (conditioned as in Equation 4.7) maintains the consistency of marker dynamics.

this transformation is especially relevant when it comes to building the estimate, since not all possible choices will provide an affine result through MSE optimisation, as it should correspond to rigid body motion. Conventional approaches like transformation matrices [Bar84] or quaternions composed with translations [Sho85] involve redundant Degrees Of Freedom (DOFs), requiring the enforcement of specific constraints to retain the result of the optimisation in the subspace of affine transformations.

Conveniently, the exponential map [Gra98] provides a parameterisation of arbitrary rotations without additional DOFs, and in conjunction with ordinary translations allows an unambiguous representation of RBTs. The mapping itself operates by relating vectors in $\mathbb{R}^3$ to the subspace S^3 of unitary quaternions, using the following expression:

$$e : \mathbb{R}^3 \to S^3 \qquad\qquad (4.7)$$

$$e(r) \equiv e^r = \begin{cases} [\sin(\|r\|)\frac{r}{\|r\|}, \cos(\|r\|)] & , \|r\| > 0 \\ [0, 0, 0, 1] & , \|r\| = 0 \end{cases}$$

As reported by Grassia in [Gra98], the singularity exhibited in a ϵ-ball of radius close to 0 can be overcome by replacing the expression of the axis of rotation by $sin(1/2\|r\|)/\|r\|r$. The scalar coefficient of the axis is now defined and C^∞-continuous in a neighbourhood of 0, and can be approximated through a power series whenever the quotient is not numerically computable. Other singularities appearing whenever $\|r\| \to 2\pi k$ for $k \in \mathbb{Z}\backslash\{0\}$ are outside the natural range of skull rotations, thus are no reason for concern in the present context.

Defining the state vector s as the concatenation of translation and rotation components (respectively $\{t_x, t_y, t_z\}$ and $\{r_x, r_y, r_z\}$), we can summarize the resulting expressions for the MSE optimisation as:

$$s = \arg\min\left(\sum_k g_k\, g_k^T\right) \Rightarrow \sum_k Jg_k(s)\, g_k^T(s) = 0 \tag{4.8}$$

where the error function $g_k\left([r, t]\right) = f_k\left([q = e^r, t]\right) = q\, \mathbf{m}_k\, q^* + t - \mathbf{m}_k'$, $q*$ stands for the conjugate quaternion of q, and Jf is the Jacobian of f

Abbreviating vector function $Jg_k(s)\, g_k^T(s)$ as $h_k(s)$, we can find a root to Equation 4.8 as a result of an iterative gradient descent process:

$$s_{i+1}^T = s_i^T + \left(\sum_k Jh_k(s_i)^T\, Jh_k(s_i)\right)^{-1} \sum_k Jh_k(s_i)^T\, h_k(s_i) \tag{4.9}$$

whose termination condition is defined as $\sum_k g_k(s_j)\, g_k(s_j)^T < \varepsilon$. The Jacobian of $h_k(s)$ can be expanded as the following 3×6 matrix:

$$Jh_k = Hg_k\, g_k^T + Jg_k\, Jg_k^T \tag{4.10}$$

where Hg_k is the $\frac{1}{2}$-Hessian tensor of the error function. In neighbourhoods of the solution $g_k \to 0$, therefore we can safely discard the first right hand term and compute everything according to the error gradient:

$$Jg_k = Jf_k \begin{bmatrix} Je_{4\times 3} & 0 \\ 0 & I_{3\times 3} \end{bmatrix} \tag{4.11}$$

Substituting the resulting expressions in Equation 4.10 yields a rather large expansion, however, it can be reduced to a low number of instructions by applying optimizing code generators from symbolic languages (e.g. *Waterloo Maple*[1]). Thus RBT estimation can be executed synchronously with the stream retargeting without involving any significant performance degradation.

Finally, rigid body motion is decoupled from local deformation by reverting the estimated RBT on the whole set of markers. This process enables retargeting by projecting motion on the local space of the skull, as described in the first paragraphs of this section. Time coherence along the MoCap stream is ultimately guaranteed by the filtering stage, therefore including any additional considerations to it as part of the removal of RBT would be redundant.

4.3   Non-linear retargeting framework

Once the markers' tracking is stabilized and guaranteed to represent just the soft body deformation of the face, we can proceed to outline the rest of the process that finally

---

[1] Maplesoft, Waterloo Maple inc. `http://www.maplesoft.com/`
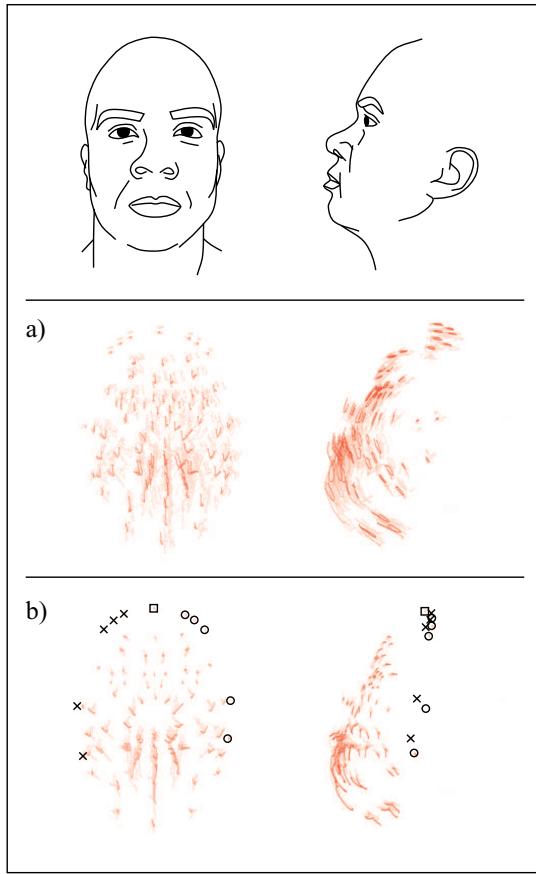
Fig. 4.4: Time-faded trace (front and profile) of a the markers used for a MoCap segment, before (a) and after (b) the removal of Rigid Body Transformation; the marker points used for the stabilization are those placed on top of the forehead and along sideburns (highlighted).

produces the retargeted animation; to this effect we will use the flow diagram in Figure 4.5.

Such a retargetting process takes as inputs the pre-processed MoCap data we want to retarget and a polygonal mesh that represents the target physiognomy; both the motion mapping and geometric deformation techniques used by this approach require that we relate those two faces through a common control structure (Planar Bones control mesh or BIDS control surface), which can be later used as a generalization of the skin geometry, as well as to drive the deformation. Since defining this relation is a rather sensitive process for the quality of the results, Section 4.3.1 introduces a semi-automatic method to transfer the marker labelling from a reference face, marked according to the MoCap source, onto the target mesh. This is depicted by stage *1* of Figure 4.5.

Once the labelling transfer is complete, we will know the point of application in the target mesh for each of the marker displacements retreived for the source. However, as
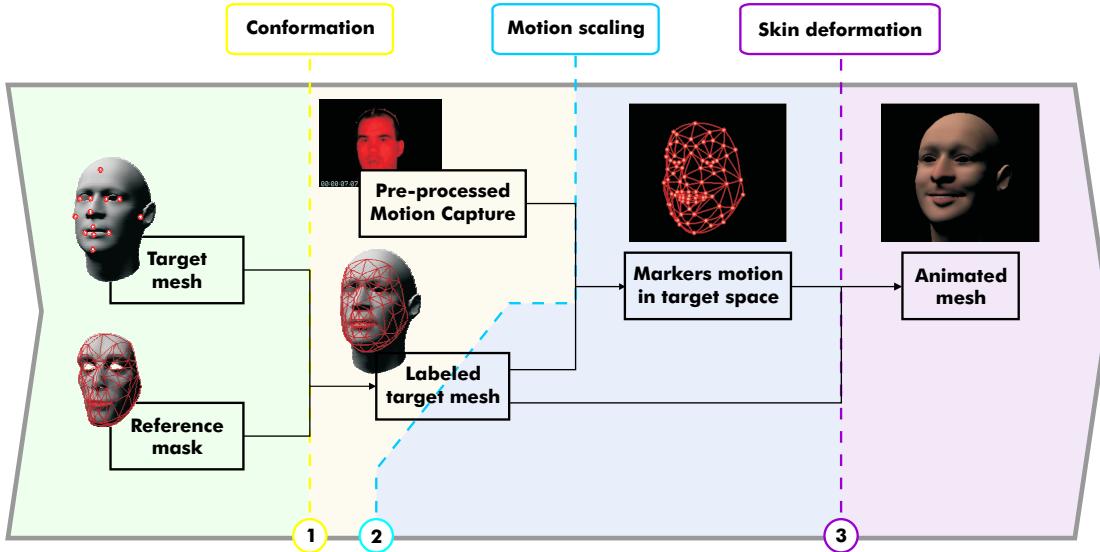
Fig. 4.5: Steps in producing the retargeted animation.

explained in Section , retargetting such vectors also requires adjusting their scale and orientation througout the animation, something that is achieved through a non-linear mapping between their respective tangent spaces (stage *2* of the diagram). As Section 4.3.2 describes, such mapping is constructed using Radial Basis Functions (RBFs); after evaluating it with the input MoCap data, we obtain an expression of the marker motion in the local space of the target mesh. Such expression is then translated into full face animation by using Planar Bones or BIDS, as stage *3* of Figure 4.5 illustrates.

4.3.1   Labelling transfer and control structure conformation

Given a reference model ready labelled with the set of markers used in the MoCap data, we construct a semi-automatic procedure that consistently transfers this labelling to the geometry of the face to be animated. While this approach requires manual labelling of the reference model for every particular marker layout (prescribed by the particular set of MoCap streams), it significantly simplifies the process for any alternative facial geometry animated with the same data. Should we be able to capture a single static shot of the actual geometry of the performer's face, it could be itself used as reference model, ultimately removing the need for manual labelling.

Since the application of BIDS (and Planar Bones) relies on a projective mapping defined between the control structure and the actual geometry being deformed, ideally the same relation exhibited amid the captured face to be animated and its corresponding control surface (or mesh). In order to satisfy this criterion, we proceed by approximating the geometry of the target face with its source counterpart, using BIDS or Planar Bones deformation on this last. Once a proper fit is found, the displaced control structure is taken as the neutral configuration for the target model.

The fitting process just described can be regarded as the result of minimising a cost function measuring the disparity between reference and target meshes. Such a metric can be easily found in averaging the distance between vertices $\mathbf{v}_i$ of the deformed reference mesh $\mathcal{R}$ and their closest points $\Pi_T(\mathbf{v}_i)$ on the surface of the target model. However this approach does not guarantee that the overall shape and structure of the control surface is in any way preserved, allowing for major shifts to happen as long as both face geometries are coincident. In practical terms this may result in the fitted mesh resembling no more than a random surface flattened over the target face.

In order to maintain consistency between both control structures, we rely on an elastic deformable model of the one taken as reference (thereafter referred to as $\mathcal{S}$). The elastic energy involved in the deformation conducted by the fitting process is taken into account to leverage the convergence of the cost function, penalizing displacements that would otherwise alter severely the structure of the reference model. Following the approach taken by Celniker and Gossard in [CG91], we can decompose such an energy functional into different components reflecting linear strain (E), shearing (S) and bending (B), and weight them accordingly to represent the elastic properties of the whole face. The resulting expression combining both cost terms (convergence and elastic energy) is:

$$\begin{aligned} \Phi_c(\mathcal{S}') \quad = \quad & (1-c) \sum_{\mathbf{v}_i \in \mathcal{R}} \left(w_i \, \|\Pi_T(\mathbf{v}_i) - \mathbf{v}_i\| \right) + \\ & c \int_{\mathcal{S}} \left(k_e \, |E(\mathcal{S}') - E(\mathcal{S})| + k_s \, |S(\mathcal{S}') - S(\mathcal{S})| + k_b \, |B(\mathcal{S}') - B(\mathcal{S})| \right) d\mathcal{S} \end{aligned} \qquad (4.12)$$

where $\mathcal{S}'$ is the fitter control structure, c is the coefficient of the elastic term, and w_i stands for a per-vertex weighting proportional to the area of the polygons of $\mathcal{R}$ incident to $\mathbf{v}_i$. $\{k_e, k_s, k_b\}$ represents the contribution of each of the deformation components; similarly to the model built by Lee et al. in [LTW95], $k_e >> k_s, k_b$, yet in this case this relation is more moderate as these coefficients represent the deformation of the whole physiognomy, rather that just the skin: $k_e = 0.15 \ N/m$ and $k_b = k_s = 0.05 \ k_e$.

In the case of BIDS, rather than computing these deformation metrics over the continuum of $\mathcal{S}$, we can take the coarse discretization defined by the triangular domains of the patches and evaluate them as in a polygonal mesh (treating them as if they were Planar Bones):
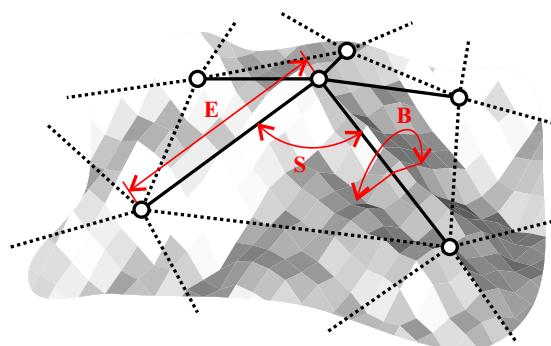
For each edge e_{ij}:

$$E\left(e_{ij}\right) = \left\|\mathbf{m_j} - \mathbf{m_i}\right\|$$

and for any vertex $\mathbf{m_i}$ lying on a triangular facet f_{ijk}:

$$S\left(\mathbf{m_i}, f_{ijk}\right) = \left(\mathbf{m_j} - \mathbf{m_i}\right) \cdot \left(\mathbf{m_k} - \mathbf{m_i}\right)$$

Finally, for each edge e_{ij} connecting facets f_{ijk} and f_{jil}:

$$B\left(e_{ij}\right) = \left(\left(\mathbf{m_j} - \mathbf{m_i}\right) \wedge \left(\mathbf{m_k} - \mathbf{m_i}\right)\right) \cdot \left(\left(\mathbf{m_k} - \mathbf{m_j}\right) \wedge \left(\mathbf{m_l} - \mathbf{m_j}\right)\right)$$

Iterating over the whole topology allows computing the discrete version of the second term of Equation 4.12 as a series of sumations. This approximation is perfectly acceptable, since it follows from Section 3.5.4 that the deformation of the piecewise control surface corresponds to that of its *domain mesh*.

Once the tools to compute Φ_c are provided, the problem could be regarded as finding a root for $\nabla\Phi_c$ in the multi-dimensional space defined by the coordinates of all markers; however, while the elastic energy terms are differentiable, the convergence function is not analytic (since neither it is $\Pi_T : \mathcal{R}$). This discards the use of conventional optimization techniques and relegates us to the use of blind approaches, which are generally far less accurate. The method applied is Simplex Downhill [PTF92], described in detail in Appendix C.

In order to compensate for the limitations of the applicable numeric schemes, a good starting point can be given just by hinting at the location of a reduced set of marker points (hence the semi-automatic designation), and then using RBFs to deform the rest of the reference set, as in [FNK99, NFN00]. We use the 11 fiducial points shown in Figure 4.6 for such purpose. These happen to be a subset of all the marker layouts of the different MoCap data used in this thesis, thereby reducing the dimensionality of the search space in every case. Also, these fiducial points span a representative set of anthropometric measures (see [Far94]), contributing to a meaningful startup deformation for the rest of markers.

Another method that helps in producing a stably convergent approximation is progressively varying the elastic energy coefficient in Φ_c and using the result as a start point for subsequent iterations. Additionally, in order to alleviate the significant computational costs of Π_T, we can map $\mathcal{R}$ onto the target mesh via a cylindrical projection, and retrieve the control facet onto which a given vertex is projected by simply checking a look-up table.

A collection of the fitted models used in Figures 4.9 and 4.10, along with their corresponding retargeted control structures are displayed in Figure 4.7.

4.3.2   Radial Basis Function-based mapping

With the techniques described in the previous section, we can not only map the marker layout of the captured streams onto the target face, but also prescribe homotopic control structures for both source and target physiognomies, using the same domain topology of the reference model. In fact, it is also possible to build a continuous mapping between
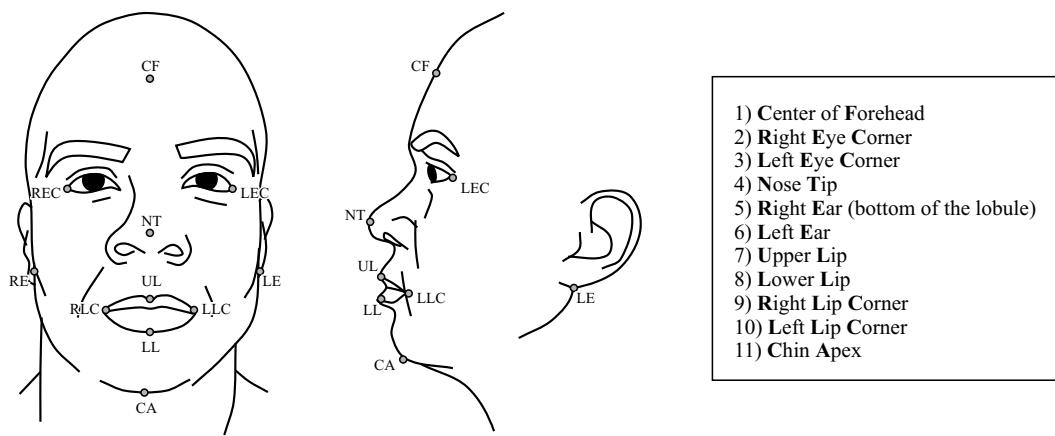
Fig. 4.6: Set of fiducial points used to construct the starting point of the approximation.

1) **C**enter of **F**orehead
2) **R**ight **E**ye **C**orner
3) **L**eft **E**ye **C**orner
4) **N**ose **T**ip
5) **R**ight **E**ar (bottom of the lobule)
6) **L**eft **E**ar
7) **U**pper **L**ip
8) **L**ower **L**ip
9) **R**ight **L**ip **C**orner
10) **L**eft **L**ip **C**orner
11) **C**hin **A**pex
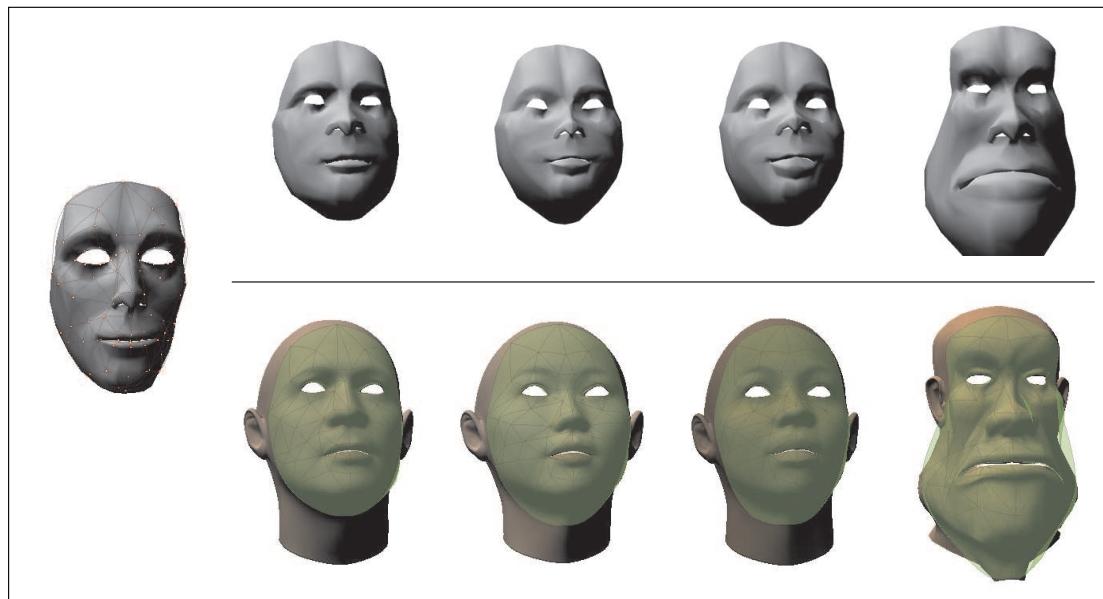
Fig. 4.7: Instances of the reference model (left) fitted to different physiognomies. The bottom row shows the resulting control surfaces (in green) overlaid onto the actual polygonal meshes.

the embedding of both surfaces (or meshes), and use it to derive the transformation between the tangent space of markers, as long as their corresponding displacements fall into a limited range. This is precisely why the removal of Rigid Body Transformation plays such an important role, as the isolated soft body deformation that results from it is relatively small in comparison with the scale of the whole face, rendering this approach valid.

The mapping described in the previous paragraph is constructed by using RBF kernels centered on a set of samples (C) of the undeformed configuration of the source control structure. At the coarsest level, these center points coincide with the markers themselves, yet if there is a large difference in their concentration amidst different regions, the relevant triangular domains are subdivided. Using sub-indices s and t for source and target models, respectively, the target image of any displaced marker $(\mathbf{m}'_i)_s$ is ruled by the following expression:

$$\Psi : \mathbb{R}^3 \to \mathbb{R}^3$$
$$(\mathbf{m}'_i)_t = \Psi\left((\mathbf{m}'_i)_s\right) = \sum_{\mathbf{c}_j \in C} \phi_j\left(\left\|(\mathbf{m}'_i)_s - \mathbf{c}_j\right\|\right) [\alpha_j, \beta_j, \gamma_j] \tag{4.13}$$

RBF kernels ϕ_j are chosen as the Inverse Multi-Quadric (IMQ) family, which exhibits a particularly good behaviour for the interpolation of irregular datasets (c.f. [Kan90]); also, they are C^∞-continuous, preventing irregularities in the retargeted motion. Specifically:

$$\phi_j : \mathbb{R} \to \mathbb{R}$$
$$\phi_j(r) = (r^2 + \delta_j)^{-\mu} \big| \{\mu > 0 \wedge \delta_j > 0\} \tag{4.14}$$

where the factor δ_j stands for the minimum distance from the kernel's center to another element of C (in other words, $\delta_j = \min\left(\|\mathbf{c}_i - \mathbf{c}_j\|\right) | \{\mathbf{c}_i, \mathbf{c}_j \in C \wedge i \neq j\}$). Exponent μ models the spatial decay in the kernel's contribution, and for this particular case it takes a value of 2.

Also, in Equation 4.13, the computation of kernel weighting vectors $[\alpha_j, \beta_j, \gamma_j]$ is prescribed by the correspondence between centers $\mathbf{c}_j$ lying on the source control structure and their target counterparts. This requires the solution of three $\sharp C \times \sharp C$ linear systems, according to the specification given in Section 3.3.3.*Radial Basis Functions*.

By evaluating this RBF-based mapping for markers moving in the local coordinate space of the source face, we implicitly retarget the point of application of such motion according to the dimensions and shape of the target physiognomy, as well as its scale and direction. As the transformation is non-linear, each infinitesimal displacement of the markers is modulated by a different metric tensor that represents the local discrepancy between source and target physiognomies (see Figure 4.8). By doing so, we extend the evaluation of *Motion Vector Direction and Scaling Adjustments* in [NN01] from just the point of application to the whole transit of the markers, thus being more consistent with the mechanical process of facial motion.

4.3.3 Application to the animation of a full facial model

Using the processes outlined in the previous sections, the motion of the common marker set is retargeted onto the local coordinates of the target face, and so is the control
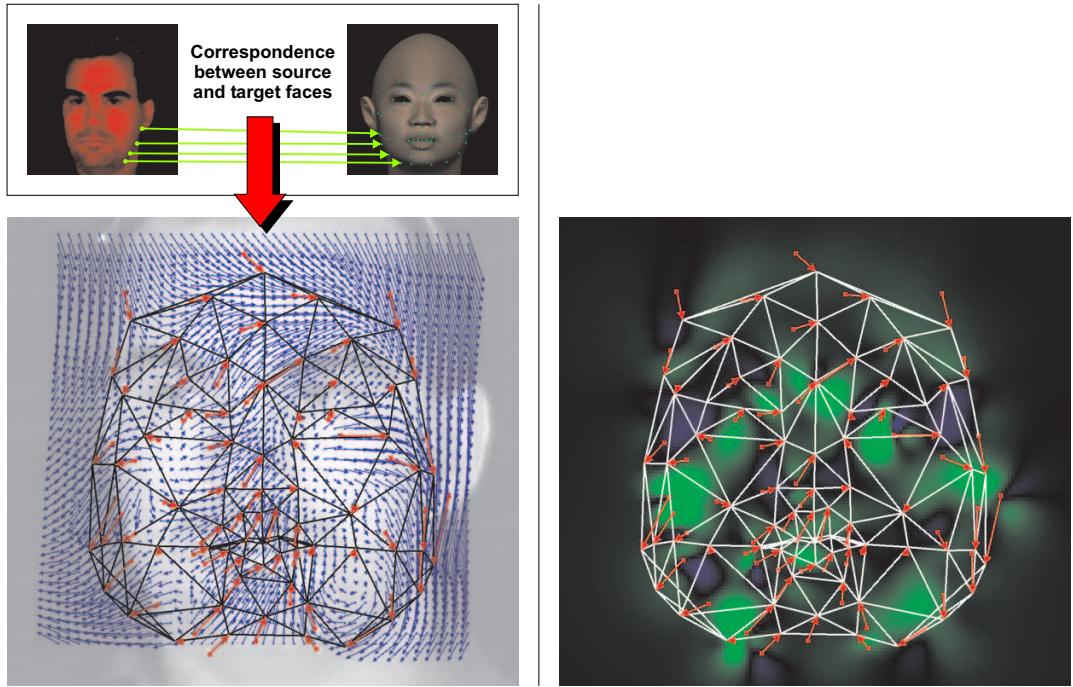
Fig. 4.8: Planar reduction of the retargeting process: the correspondence between sample points in source and target control structures leads to a dense mapping (left) that deforms the tangent space of the source markers into the embedding of the target control structure (in this case, both are identically the plane). The determinant of the metric tensor induced by this mapping (right) reveals general elongation (coloured in green) to represent the larger scale of motion in cheeks and inner mouth, while infinitesimal displacements are compressed (blue) in the corners of eyes and mouth.

structure of BIDS (or Planar Bones). This enables us to apply any of these deformation techniques to deform the rest of the target's face geometry and produce the final animation (see Sections 3.4 and 3.5), with their corresponding benefits over alternative Motion Interpolation approaches (see Section 3.3.3).

In particular, the high degree of automatism brought in by these two warping algorithms contributes largely to the versatility of this retargeting framework, allowing for it to be applied with little costs on a wide variety of faces. However, in the case of Planar Bones, the treatment of discontinuities (see Section 3.4) requires creating a discontinuity map for each of the new physiognomies. In order to circumvent this, we can use the inverse of the Π_T mapping using by the fitting process (c.f. Section 4.3.1) to translate a set of reference discontinuity textures onto the target model. BIDS does not require such additional operations, since the discontinuity handling is conducted according to normal field correspondence, as explained in Section 3.5.6; yet for some physiognomies the disparity threshold may need to be individually revised (e.g. depending on the curvature

of the lips).

In addition to human faces, Facial MoCap captured from an actual performer may be applied to a cartoon-style character. In this case, some of the facial expressions might need to be exaggerated to give a more caricature-like effect, as it corresponds to a cartoon; such exaggeration can be easily attained by just extrapolating the results of the retargeting process using local weighting maps on the control structure, as Figure 4.11 illustrates.

4.3.4   Limitations

The animation framework we have outlined is only applicable to faces that share similar anatomic principia. For example, we could not produce plausible results for a goat's face from retargeted human motion, as the utterances of the animal do not involve lip articulation, while those by the human do. However, very different physiognomies like the realistic and cartoon faces demonstrated in Figure 4.11 can be animated by the same input data, since all of them represent human characters.

In addition, the initial configurations for source and target physiognomies onto which the RBF mapping is constructed have to be equivalent, ideally representing the same neutral expression. Otherwise, the expressions retargeted onto the target face will not be consistent with the originating ones. This can be a problem for cartoon faces, as the original model may exhibit some expressive traits as part of the characterization, requiring the artist to be aware of this particular aspect. Also, it has repercussions on the capture side, as the performer is required to exhibit a neutral pose at the beginning of the stream. This is not trivial, since the mental embodiment of the character to be played unconsciously pushes the actor to anticipate part of the expression. Fortunately, a separate stream with just that neutral pose can be used as reference, as the RBT removal will unify it with any other MoCap sequence.

Further restrictions on the application of this framework are imposed by the marker-based MoCap paradigm adopted. There are regions of the face, like the inner contour of the mouth, and other facial features, such as eye gaze and tongue movement, that cannot be tracked with markers. Treating some of these would invariably need alternative vision approaches, like for instance [KWT88, Qu92, Ess95], while others simply cannot be tracked at all (e.g. interior of the mouth). In particular, the motion of the inner contour of the mouth can be approximated with a mass-spring system [PB81], as the topology is quite simple and allows for accurate real-time solutions; this is the approach adopted so far by the present implementation of framework, yet in some occasions it leads to small lip intersections. Solving this would require a lip contact model, incorporating collision detection, as described for instance by King in [KPO00].

Fig. 4.9: Results of the retargeting contrasted with the actual performance (MoCap data courtesy of Electronic Arts Canada). On the accompanying CD, see **BIDSwrinkling - EA Mocap.avi**

Fig. 4.10: MoCap retargeting results with different facial models.

Fig. 4.11: Cartoon effects by extrapolating the results of retargetting MoCap. The left-most column shows the normal retargetting; after modulating the RBF contribution in certain masked areas (green) the exagerated effects on the right demonstrate cartoonesque expressions for *blowing* (top) and *yelling* (bottom).

4.4   Summary

This Chapter has introduced a retargeting framework that enables adapting a model of facial expression based on spatially discrete features (in particular marker points) across different physiognomies, thus overcoming one of the main limitations of these control paradigms. Such a framework consists of three main different stages:

- Fitting of a common control structure to both source and target faces.

- Construction of an RBF-based mapping relating the embedding of both control surfaces, where marker motion takes place.

- Deformation using a surface-driven geometric warp, controlled by the motion of the retargetted marked points.

The limitations of marker-based MoCap systems require that their input is pre-processed to remove noise and guarantee consistency. At the same time, Rigid Body Transformation has to be decoupled from soft body motion in order to validate some of the assumptions taken in the construction of the RBF-based mapping.

The methods here presented, in conjunction with the deformation techniques described in Chapter 3, permit a versatile and extensive use of performance-based animation techniques, maximizing the usability of captured streams irrespective of the performer's anatomy. While this is unquestionably advantageous for the perceptual realism of the results, there are important aspects of facial motion that cannot be captured with these approaches, such as the fine scale behaviour of the facial tissue.

Following the publication of the details of this framework in [SEK03], similar techniques have appeared in [PS03] and [NJ04]; the main difference between these works and the procedure documented in this Chapter is that they actually build the RBF-mapping over a series of expressions rather than a single neutral one, working in a way that is reminiscent of the temporally-discrete model by Joshi et al. [JTD03] (c.f. Section 2.4.1). Given the relatively small scale of facial soft body motion, it is rather arguable whether this additional degree of complexity of the mapping is necessary.

5. SYNTHETIC WRINKLING

5.1   Introduction

Section 3.3.3 listed three main criteria for attaining realistic results with geometric deformation schemes applied to appearance-based models of expression (namely consistent motion propagation, preservation of tissue smoothness and locality of deformation); however, even if all these criteria are fullfilled, the accuracy of the deformation is ultimately constrained by the density of markers placed on the target mesh. Optical MoCap systems do not allow for the marker distribution to be arbitrarily dense (to prevent misidentification), so there will be aspects of skin behaviour that cannot be represented; with the usual amount of markers (in the order of a few hundred - c.f. Chapter 4), we can model coarse phenomena such as large scale soft tissue deformation and the swelling and protrusion caused by facial muscles (e.g. maxillary pairs under the cheeks), but the resulting deformation will ignore finer elements such as buckling and wrinkling. Despite their small scale, these aspects of skin behaviour play an important role in the perception of facial expression (e.g. forehead creases in a surprise gesture), and therefore any animation system aiming to produce believable results has to take them into consideration.

The elastic properties of facial tissue, described in Section 3.2, cause the visible layer of skin to bend and buckle under linear strains, rather than compress itself. While this behaviour is almost regular in immature tissue (it will occur in the same manner at any point of the skin), the exercise of facial expression combined with the natural process of ageing makes it much more uneven in adult subjects; successive plastic deformation force the collagen fibres of the dermis to reorient themselves along specific crease lines (macro-furrows), making the elastic response significantly feebler in the transversal direction. Consequently, the skin in these areas burrows under the buckling of surrounding tissue, creating the familiar furrowing effect that we perceive as wrinkles. For a more detailed account of facial biomechanics refer to [Eld77].

The process of facial wrinkling is not only irregular in distribution, but also strongly dependent on the directions of skin compression, as the crease lines that appear under a particular pattern of strain may behave in a completely different way given other strain conditions (see Figure 5.1). Such complexity, together with the fine scale at which these phenomena occur, makes reproducing them with a mesh-based mechanical simulation a near impossible task. This is so not only because of the numerical and efficiency problems brought along with the increase of mesh resolution, but also because of the intricacy of analysing and reproducing the elastic behaviour of living tissue at each particular point. Therefore ambitious generic approaches like those proposed by Wu et

Fig. 5.1: Left and right columns of this figure illustrate how wrinkles appear in identical
           patterns under similar directions of compression (vertical in the case of eyebrow
           raising and diagonal in the case of frowning).

al. [WMT94] (c.f. Section 3.2.2) have their scope limited to minimalist, non real-time
local simulations, like the work by Magnenat-Thalmann et al. in [MKL02], rather than
full-fledged Facial Animation.

Since most expressive wrinkles appear in very specific patterns, various researchers
have adopted the idea of moulding the corresponding crease lines in the model represent-
ing the face, either as isocurves in parametric patches [Ber85, VY92] or as curve splines
embedded in the topology of a polygonal mesh [WKM97]. Then, using specific knowl-
edge for the wrinkling in each facial model (e.g. the protrusion of certain vertices when
compressed between parallel crease lines), large scale deformation is complemented with
synthetically-generated furrows without the need for evaluating any mechanical model.

In Section 5.2 of this Chapter we will develop a model-independent approach that
enables us to specify simultaneously different patterns of furrowing according to specific
directions of strain. The generality of this technique follows from the ability to relate a
common control structure (a mesh in the case of Planar Bones, and a piecewise para-
metric surface for BIDS) to different facial physiognomies, introduced in Chapter 4; by
these means, we can can transfer a single definition of wrinkling behaviour given for such
control structure to a large variety of faces that can be approximated by its topology.

While the approach in Section 5.2 requires specifying by hand the relation between
directions of strain and furrowing patterns, Section 5.3 proposes a method that captures
such relation from an actual subject's performance. Furthermore, since the definitions

of wrinkling and strain used by such a method are given with respect to the control structure used by BIDS, the data retrieved from a particular subject can also be applied to different physiognomies (in a similar manner to Section 5.2).

5.2 Area-preserving approach

The first method that we will present in this chapter uses furrowing patterns associated with particular directions of skin compression sustained in specific regions of the face. In the approaches described in Section 5.1, such patterns are transferred onto the target mesh by using offsets along its normal field (a technique known as *displacement mapping*, c.f. [KL96]); however, in our case such a field is not defined onto the mesh itself, but onto the control structure of Planar Bones. This enables us to consider deformation and wrinkling under the same paradigm, as the effect of the Planar Bones warp also occurs along the normal field of each control facet.

This relation between deformation and wrinkling allows constructing the later around one of the mechanical properties of skin: local area preservation under compression; whenever a control mesh is used to represent soft body deformations, the linear strains inflicted on it translate almost directly onto the warped mesh, arbitrarily diminishing the local area of the regions projected over compressed control facets. In order to compensate for this, we use displacement mapping over the Planar Bone control mesh and modulate vertex offsets through a locally-defined shape function. This shape function is constructed by a weighted sum of the furrow patterns relevant to the area, after evaluating the local strains experienced in their corresponding directions. The whole process is summarised in Figure 5.2.

Weighting each of the furrow patterns according to their characteristic direction of occurrence (e.g. vertical in the forehead pattern depicted in Figure 5.2) requires a metric of strain that can be referred to specific directions along the control mesh. This is precisely the focus of Section 5.2.1.

In addition, Section 5.2.2 describes how to use the resulting shape function to modulate vertex offsets in order to preserve local surface area, thereby computing the appropiate furrowing depth under compression.

5.2.1 Generalized formulation of directional strain

For any point in a deformable body, we can define its material (*Lagrangian*) coordinates $\mathfrak{R} \equiv \{X, Y, Z\}$ as its position in Euclidean space while in the rest state. Similarly, spatial (*Eulerian*) coordinates $\mathfrak{r} \equiv \{x, y, z\}$ are those spanned by the moving image of the point while sustaining deformation. Considering spatial coordinates as a function of the Lagrangian parameters and time, their Jacobian takes the form of a 3×3 mixed tensor that is commonly known as the *deformation gradient*:

$$F_j^i = (J(\mathfrak{r}))_j^i = \frac{\partial \mathfrak{r}_i}{\partial \mathfrak{R}_j} \qquad (5.1)$$

Under this notation, the displacement function can be trivially defined as $\mathfrak{u} = \mathfrak{r} - \mathfrak{R}$; its Jacobian, called the *differential strain tensor*, can be related to the deformation gradient

Evaluation of
directional strain

Synthetic
wrinkling

Deformation of
the control mesh

Modulation of the
shape functions
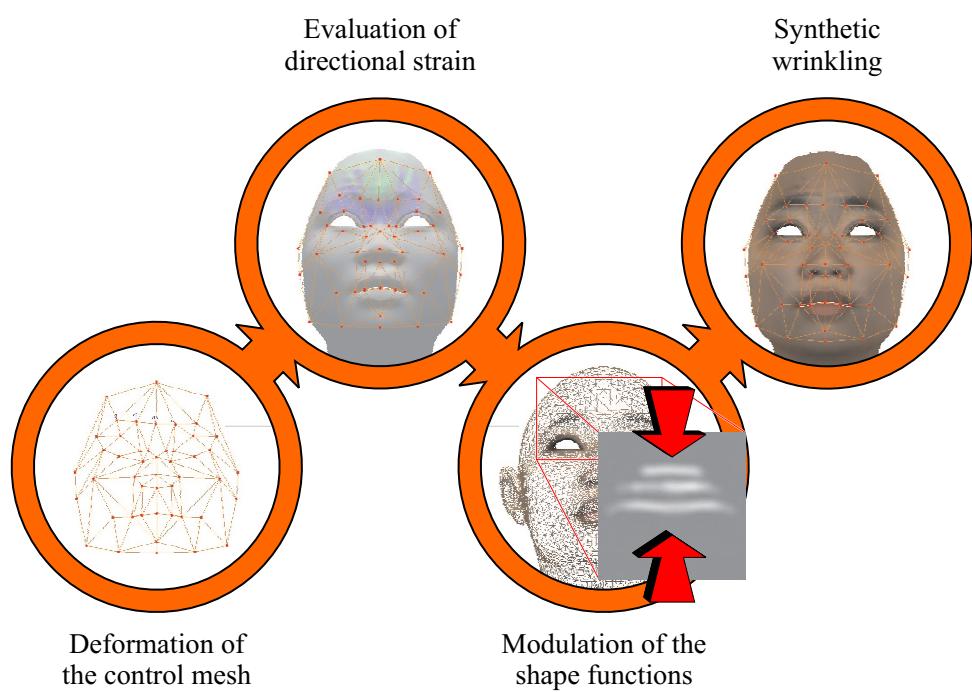
Fig. 5.2: Steps of the area preserving approach to synthetic wrinkling.

as follows:

$$S^i_j = \frac{\partial \mathfrak{u}_i}{\partial \mathfrak{R}_j} = F^i_j - \delta_{ij} \tag{5.2}$$

Using this definition, at any given material point $\mathbf{p}$ we can compute the strain in the direction of a unitary vector v as the directional derivative of the projection of the displacement function over it:

$$\varepsilon_v(\mathbf{p}) = D_{v_\mathbf{p}} v \cdot \mathfrak{u} = v \cdot D_{v_\mathbf{p}} \mathfrak{u} + \mathfrak{u}(\mathbf{p}) \cdot D_{v_\mathbf{p}} v = v \, J(\mathfrak{u})\big|_\mathbf{p} v^T = v \, S\big|_\mathbf{p} v^T \tag{5.3}$$

As we are just concerned with the simpler case of *planar strain* (e.g. the strain along a control facet or in the tangent plane of a parametric surface), Equation 5.3 can be replaced by its restriction to the plane:

$$\varepsilon_v = \frac{\partial \mathfrak{u}_x}{\partial X} \, v_x^2 + \frac{\partial \mathfrak{u}_y}{\partial Y} \, v_y^2 + \left(\frac{\partial \mathfrak{u}_x}{\partial Y} + \frac{\partial \mathfrak{u}_y}{\partial X} \right) \, v_x v_y \tag{5.4}$$

Thus the strain tensor S can be reformulated as a symmetric bilinear form:

$$S = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} \\ \gamma_{xy} & \varepsilon_{yy} \end{bmatrix} \text{ with } \begin{cases} \varepsilon_{xx} = \frac{\partial \mathfrak{u}_x}{\partial X} \\ \varepsilon_{yy} = \frac{\partial \mathfrak{u}_y}{\partial Y} \\ \gamma_{xv} = \frac{1}{2} \left(\frac{\partial \mathfrak{u}_x}{\partial Y} + \frac{\partial \mathfrak{u}_y}{\partial X} \right) \end{cases} \tag{5.5}$$
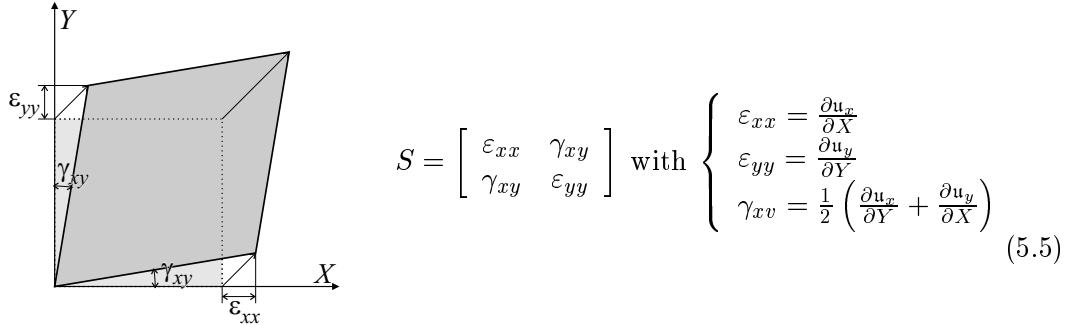
Giving a physical interpretation to the components of this tensor, ε_{xx} and ε_{yy} can be considered as the linear strains along the X and Y directions in Lagrangian coordinates, respectively. γ_{xy} is an angular measure representing the shearing between these two axes.

As with any symmetric 2-form, Equation 5.5 admits Eigenvalue decomposition (c.f. [Mey00]); this yields two characteristic directions of strain, w_1 and w_2, over which this magnitude is purely linear and has real values λ_1 and λ_2, respectively. This decomposition is extremely relevant since we need to differentiate between positive strains, corresponding to tissue elongation, and the negative values that represent compression (and consequently generate wrinkling).

Using the eigen-terms of S, we can define the *differential compression tensor* as the result of clamping the characteristic strains to compression-only values:

$$\bar{S} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}^T \begin{bmatrix} \min(\lambda_1, 0) & 0 \\ 0 & \min(\lambda_2, 0) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{5.6}$$

This quadratic expression allows computing the magnitude of linear compression in any specific direction of the plane, discarding elongation terms. This is exactly what

we need to evaluate the contribution of each predefined furrowing pattern to the final shape function. However the directions of occurence for these patterns are given in a different parameterisation, for which the next section will define equivalent tools. For a further review of deformation mechanics, consult [S P70, Liu00].

5.2.2   Construction of the shape function

Each of the furrow patterns composing the wrinkling model is given in the texture space of the control mesh, defined uniquely as the cylindrical projection of a reference instance. The direction of strain in which a given pattern occurs is also defined in that space; however, since displacement amplitude is computed with respect to the barycentric domain of the each of the control facets, it is both sensible and practical to reparameterise this vector onto the involved control elements.

For a control facet spanning markers with texture coordinates $\{\mathbf{t}_l, \mathbf{t}_m, \mathbf{t}_n\}$, the barycentric expression of any vector with coordinates $[v_s, v_t]$ in texture space is given as:

$$[v_u, v_v] = \left[\begin{array}{c} \mathbf{t}_l - \mathbf{t}_n \\ \mathbf{t}_m - \mathbf{t}_n \end{array} \right]^{-T} \left[\begin{array}{c} v_s \\ v_t \end{array} \right] ; \; v_w = -v_u - v_v \tag{5.7}$$

Similarly, strain has to be computed onto the same barycentric domain; however, this parameterisation does not match the orthonormal properties of the XY planar frame used in the derivations from Section 5.2.1. Let's consider the jth control facet spanning marker points $\{\mathbf{m}_l, \mathbf{m}_m, \mathbf{m}_n\}$; consistently with the transformation rules for mixed tensors [DP90], the infinitesimal strain tensor takes the following form:

$$S = J(\mathfrak{u}) = \left[\begin{array}{cc} \frac{d\mathfrak{u}^u}{du} & \frac{d\mathfrak{u}^u}{dv} \\ \frac{d\mathfrak{u}^v}{du} & \frac{d\mathfrak{u}^v}{dv} \end{array} \right] = \left[\begin{array}{c} e^u \\ e^v \end{array} \right] \left[\begin{array}{c} \mathfrak{u}(\mathbf{m}_l) - \mathfrak{u}(\mathbf{m}_n) \\ \mathfrak{u}(\mathbf{m}_l) - \mathfrak{u}(\mathbf{m}_m) \end{array} \right]^T \tag{5.8}$$

where e^u and e^v stand for the dual vectors to $e_u = \mathbf{m}_l - \mathbf{m}_n$ and $e_v = \mathbf{m}_m - \mathbf{m}_n$, respectively, in the tangent space of the triangle.

Taking this new definition of strain tensor as a replacement for Equation 5.3, all the subsequent derivations from Section 5.2.1 are applicable in order to compute compression along the directions associated with the occurrence of each furrow pattern. For a given face, let v_i be the vector bringing up the ith affecting pattern; the corresponding weight w_i is computed as follows:

$$w_i = \frac{v_i \; \bar{S} \; (v_i)^T}{\|v_i \; B\|^2}, \text{ with } B = \left[\begin{array}{c} e_u \\ e_v \end{array} \right] \tag{5.9}$$

and $\bar{S}$ standing for the relevant compression tensor derived from S as by Equation 5.6.

Given the set W of wrinkle patterns relevant to an area of tissue undergoing deformation, we can define the parametric expression of the resulting shape function in terms of the displacement representations (p_i) of each of its composing patterns:

$$p : \mathbb{R}^2 \to \mathbb{R}$$
$$p(u, v) = \alpha \; q(u, v) + \sum_W w_i \; p_i(s(u, v), t(u, v)) \tag{5.10}$$

a) Horizontal strain: dominance
of the furrow pattern

b) Vertical strain: unstructured bulging
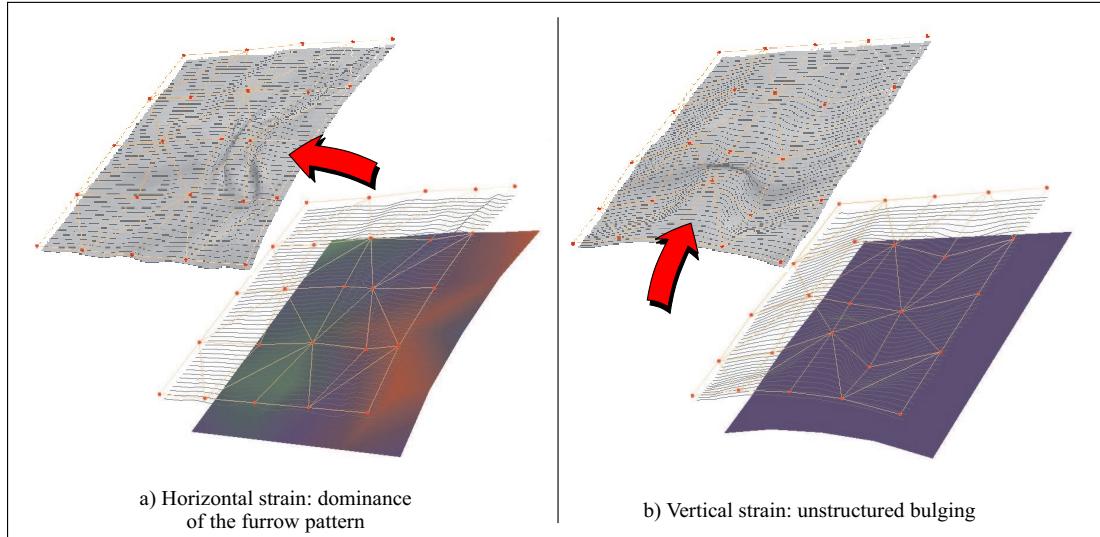
Fig. 5.3: Example of a shape function combining a banded furrow pattern associated with the horizontal direction of the reticule, and a paraboloid function as default displacement to compensate for compression. Positive horizontal strain appears as a green colouration on the domain plane, while negative values are red.

where

$$\alpha = \max\left(0, 1 - \frac{1}{\sharp W}\sum_W w_i\right),$$

u and v span the domain of the control facet, and s and t are the interpolated texture coordinates indexing the furrow patterns. q is chosen as a paraboloid that represents isotropic, unstructured swelling, replacing the contribution of wrinkle patterns whenever none of them occurs with special intensity (strain is null or positive along most of their respective directions of occurence). This paraboloid has its apex placed over the domain's barycentre, and is null at the vertices:

$$q(u,v) = -3u^2 - 3v^2 - 3uv + 3u + 3v \tag{5.11}$$

The results of a shape function constructed in this way are demonstrated in Figure 5.3, where the whole wrinkling technique is applied using a single furrow pattern consisting of vertical banding, and associated to horizontal strain in the grid. Whenever the fragment of tissue is compressed in such direction, the local variation of surface area is compensated through an undulation in the shape of three vertical bars (the furrow pattern); conversely, when compression occurs vertically, just an ordinary fold appears.

5.2.3 Computing the amplitude of displacements

Let's consider T_j as the set of triangular faces from the target geometry which are projected, by application of Planar Bones, onto the jth element of the control mesh.

The initial area of the ith face in that set, spanning vertices $\{\mathbf{p}_l, \mathbf{p}_m, \mathbf{p}_n\}$, can be given in terms of the coordinate frame B_j^0 associated to the relevant control element, defined in Equation 3.31:

$$A_i = \tfrac{1}{2} \left| (\mathbf{p}_m - \mathbf{p}_l) \wedge (\mathbf{p}_n - \mathbf{p}_l) \right| =$$
$$\tfrac{1}{2} \left| \left([\perp_j (\mathbf{p}_m) - \perp_j (\mathbf{p}_l)] B_j^0 \right) \wedge \left([\perp_j (\mathbf{p}_n) - \perp_j (\mathbf{p}_l)] B_j^0 \right) \right| \tag{5.12}$$

where $\perp_j (\mathbf{q}) = [r(\mathbf{q}), s(\mathbf{q}), t(\mathbf{q})] = \mathbf{q} \, (B_j^0)^{-1}$.

Similarly, the area of the deformed triangle could be computed by just replacing B_j^0 in Equation 5.12 by its corresponding image $B_j^{0\prime}$, while preserving the original projected coordinates. As commented before, area is not necessarily preserved by soft body deformations, and in order to compensate for this variation we introduce a displacement along the normal of each control facet:

$$\begin{aligned}
\{r'(\mathbf{q}), s'(\mathbf{q})\} &= \{r(\mathbf{q}), s(\mathbf{q})\} \\
t'(\mathbf{q}) &= t(\mathbf{q}) + h \, p \, (r(\mathbf{q}), s(\mathbf{q}))
\end{aligned} \tag{5.13}$$

where h represents the global amplitude for the control facet, modulated by the shape function p defined in Section 5.2.2.

Also, rigid body transformation is irrelevant for the matter of computing area variations; therefore, without lost of generality, we can assume all deformations occur on the plane. Under this assumption, $B_j^{0\prime}$ can be expressed in terms of its rest (initial) configuration and the deformation gradient F_j:

$$B_j^{0\prime} = F_j^T \, B_j^0 = \begin{bmatrix} 1 + \mathfrak{u}_{,r}^r & \mathfrak{u}_{,r}^s & 0 \\ \mathfrak{u}_{,s}^r & 1 + \mathfrak{u}_{,s}^s & 0 \\ 0 & 0 & 1 \end{bmatrix} B_j^0, \text{ with } \mathfrak{u}_{,\beta}^\alpha = \frac{d\mathfrak{u}^\alpha}{d\beta} \tag{5.14}$$

Conveniently, the differential strain terms in Equation 5.14 are computed as the transpose of tensor S in Equation 5.8, swapping coordinates $\{s, t\}$ by their barycentric counterparts $\{u, v\}$.

Combining all these considerations together with the abbreviation $\Delta_{ij}u = u(\mathbf{q}_j) - u(\mathbf{q}_i)$, we can redefine the area of the deformed i^{th} element of T_j as:

$$A_j' = f \left(\mathfrak{u}_{,r}^r , \mathfrak{u}_{,s}^r , \mathfrak{u}_{,r}^s , \mathfrak{u}_{,s}^s , h \right) = \tag{5.15}$$
$$\tfrac{1}{2} \left| \left([\Delta_{lm}r, \Delta_{lm}s, \Delta_{lm}t + h \, \Delta_{lm}p] \, F_j^T B_j^0 \right) \wedge \left([\Delta_{ln}r, \Delta_{ln}s, \Delta_{ln}t + h \, \Delta_{ln}p] \, F_j^T B_j^0 \right) \right|$$

We now can compute the total area of T_j as the summation of the different A_i' it comprises. This expression is quartic in both strain and amplitude variables, therefore trying to evaluate h by equating it to the initial total area is not analytically possible. Yet both strain and displacement amplitude are relatively small compared to the dimensions of the control facet, making a first order Taylor approximation admissible. Taking the rest conditions as the center of the corresponding polynomial expansion:

$$A'(h, \mathfrak{u}_{,\beta}^\alpha) \simeq \sum_{T_j} A_i' \big|_C + \sum_{\{s,t\}} \mathfrak{u}_{,\beta}^\alpha \sum_{T_j} \frac{\partial A_i'}{\partial \mathfrak{u}_{,\beta}^\alpha} \Big|_C + h \sum_{T_j} \frac{\partial A_i'}{\partial h} \Big|_C$$
$$\text{where } C \equiv \begin{cases} h = 0 \\ \forall \{\alpha, \beta\} \in \{s,t\}, \ \mathfrak{u}_{,\beta}^\alpha = 0 \end{cases} \tag{5.16}$$

The first term on the right hand side of the approximation represents the initial total area, yielding the following approximation for h:

$$h \simeq \frac{\sum_{\{s,t\}} -\mathfrak{u}^{\alpha}_{,\beta} \sum_{T_j} \frac{\partial A'_i}{\partial \mathfrak{u}^{\alpha}_{,\beta}}\Big|_C}{\sum_{T_j} \frac{\partial A'_i}{\partial h}\Big|_C} \tag{5.17}$$

Using the abbreviation $\mathbf{\Delta}(\alpha, \beta) = \Delta_{lm}\alpha\Delta_{ln}\beta - \Delta_{ln}\alpha\Delta_{lm}\beta$, we can evaluate the denominator of the previous quotient by applying:

$$\begin{aligned}
\frac{\partial A'_i}{\partial h}\Big|_C = \frac{1}{\phi} &\left(\mathbf{\Delta}(r,p)\left(\mathbf{\Delta}(r,t)\|e_r\|^2 + \mathbf{\Delta}(s,t)\|e_r \wedge e_s\|\right) \right. \\
&\left. + \mathbf{\Delta}(s,p)\left(\mathbf{\Delta}(s,t)\|e_s\|^2 + \mathbf{\Delta}(r,t)\|e_r \wedge e_s\|\right) \right)
\end{aligned} \tag{5.18}$$

As for the coefficients of linear strains, they follow a similar pattern:

$$\frac{\partial A'_i}{\partial \mathfrak{u}^r_r}\Big|_C = \frac{1}{\phi}\left((\mathbf{\Delta}(r,s)\|e_r \wedge e_s\|)^2 + \mathbf{\Delta}(r,t)\left(\mathbf{\Delta}(r,t)\|e_r\|^2 + \mathbf{\Delta}(s,t)\|e_r \wedge e_s\|\right) \right) \tag{5.19}$$

$$\frac{\partial A'_i}{\partial \mathfrak{u}^s_s}\Big|_C = \frac{1}{\phi}\left((\mathbf{\Delta}(r,s)\|e_r \wedge e_s\|)^2 + \mathbf{\Delta}(s,t)\left(\mathbf{\Delta}(s,t)\|e_s\|^2 + \mathbf{\Delta}(r,t)\|e_r \wedge e_s\|\right) \right) \tag{5.20}$$

and this is even simpler in the case of mixed derivatives:

$$\frac{\partial A'_i}{\partial \mathfrak{u}^r_s}\Big|_C = \frac{1}{\phi}\mathbf{\Delta}(r,t)\left(\mathbf{\Delta}(s,t)\|e_s\|^2 + \mathbf{\Delta}(r,t)\|e_r \wedge e_s\|\right) \tag{5.21}$$

$$\frac{\partial A'_i}{\partial \mathfrak{u}^s_r}\Big|_C = \frac{1}{\phi}\mathbf{\Delta}(s,t)\left(\mathbf{\Delta}(r,t)\|e_r\|^2 + \mathbf{\Delta}(s,t)\|e_r \wedge e_s\|\right) \tag{5.22}$$

In all the expressions listed above, ϕ stands for a constant coefficient that appears both in numerator and denominator of Equation 5.17, therefore it does not need to be taken into account in the evaluation.

All the coefficients for strain terms can be precomputed, as the projection is performed as an offline process. Furthermore, Equation 5.18 can be reduced to linear terms on each of the furrow patterns involved, just by applying Equation 5.10. This all allows the displacement amplitude for each control facet to be computed as a simple iteration over the geometry projected onto it, making the process efficient enough for real-time purposes. Also, numerical stability is guaranteed since the denominator is always non-null, thanks to the contribution of the default parabolic furrow pattern.

5.2.4   Application

In order to enforce displacement continuity along the control mesh, the value of h is computed as a second order Bézier interpolant of averaged vertex and edge values, similarly to the treatment that interpolated normals received in Section 3.5.4. Figure 5.4 illustrates the results of applying a simple shape function consisting of a single furrow pattern, which represents forehead wrinkling for the event of raising eyebrows.
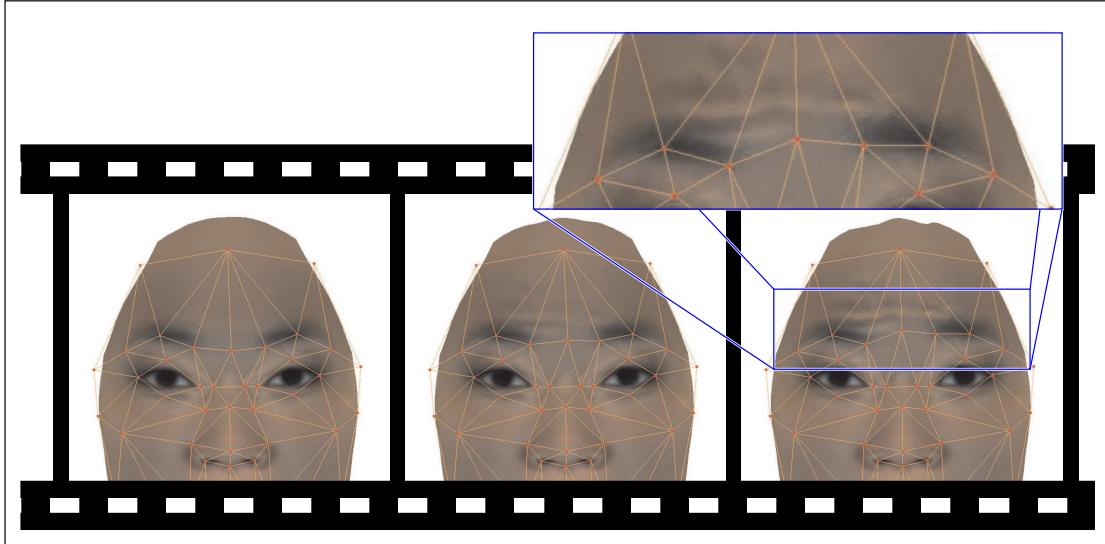
Fig. 5.4: Results of applying the geometry displacement technique outlined in Section 5.2
with the shape function depicted in Figure 5.2. The inset shows how the visual
results are strictly subject to the density and topology of the mesh, making it
a rather inefficient way of representing wrinkling.

The effects produced in such a figure fail to convey the transversal sharpness and lon-
gitudinal smoothness of the crease lines and their surrounding furrowing. This is mainly
because the topology of a generic polygonal mesh representing the skin is not necessar-
ily suitable to portray wrinkle lines. Even if we increase the resolution in an adaptive
manner, for instance as explained in Section 3.5.6, it may well not result in an adequate
triangulation properly representing the shading around creases. The polygonal density
that this may require is overly large, especially in comparison with model-dependent
approaches where furrows are embedded in the mesh.

In addition, the shared effects of concurrent control facets distorts the translation of
furrow patterns onto the target mesh, as each of them bears a different normal field; in
large deformations this could result in the contours of crease lines being smoothed out.
Therefore it would be sensible to use an algorithm that does not require that kind of
blending to guarantee geometric continuity, such as BIDS.

The definition of realistic furrow patterns is also a problem, as they have to be
manually produced and associated with specific directions of strain. In many cases
these directions are not a single one, requiring us to split patterns into many concurrent
components, possibly beyond a number that is practical. Ideally, it should be preferable
to capture such behaviour from facial performances rather than specifying it manually.

Combining all these considerations, the next section will define an alternative ap-
proach that elaborates on the methods already proposed while trying to overcome their
limitations.

5.3   Performance-based approach

Image-based techniques provide a straightforward solution to retrieving fine-scale skin behaviour from facial performance. Animation systems adopting a temporally-discrete model of expression are then able to integrate the skin appearance in a morphable texture model, working in parallel with geometry poses; examples can be found in [Pig99]. However, this approach has the limitation of being bound to specific lighting conditions, restricting significantly its scope of application.

Alternative methods make use of *normal maps*, or the equivalent *bump maps* [Bli78], to represent fine tissue deformations in a way that can be combined with relighting. As Tu et al. [TLY03] demonstrate, normal map data can also be extracted from actual expressions, yet this is again limited to static poses (see as well [NJ04]). Blending wrinkles carries an intrinsic smoothing that ruins the quality of their representation, as noted by Pighin in [Pig99] for the case of photographic textures; this is even more sensitive when it is normals instead of colour data that is interpolated. Specialized blending schemes in frequency space (also in [Pig99]) help in alleviating this problem for morphable texture models, however the idea does not perform as well for normal maps.

The second approach to synthetic wrinkling presented in this Chapter aims to tackle these problems, using similar photographic techniques to derive a causal model of fine-scale tissue behaviour. This is accomplished by relating the deformation metrics described in Section 5.2.1 (identified as the effective cause of skin wrinkling) to the normal map variations that define the visual appearance of wrinkling. Such analysis is possible since in the process of capturing normal maps we also retrieve deformation data from the BIDS control surface, constructed as an approximation of the facial skin thanks to the tracking of optical markers.

The whole performance-based process conducted by this approach can be reduced to the following four steps (as illustrated in Figure 5.5):

1. Conformation of a reference facial mask to the performer.

2. Extraction of normal maps for each expression by composing different views and lighting conditions.

3. Local parameterisation of the variations of the normal field with respect to deformation metrics.

4. Synthesis of the wrinkling effect by evaluating the parametric model over an animated face.

The first three steps are dedicated to the construction of the wrinkling model, and occur as a separate process prior to conventional Motion Capture; then, when MoCap is used to drive animation, the rate of deformation is computed in real-time and used to evaluate the wrinkling model. The results of such evaluation contribute an additional layer (in the form of a resynthesized normal map) applied onto the geometry deformed by BIDS.

Successive sections of this Chapter are dedicated to describing each of these steps in more detail and elaborating further on the techniques involved.
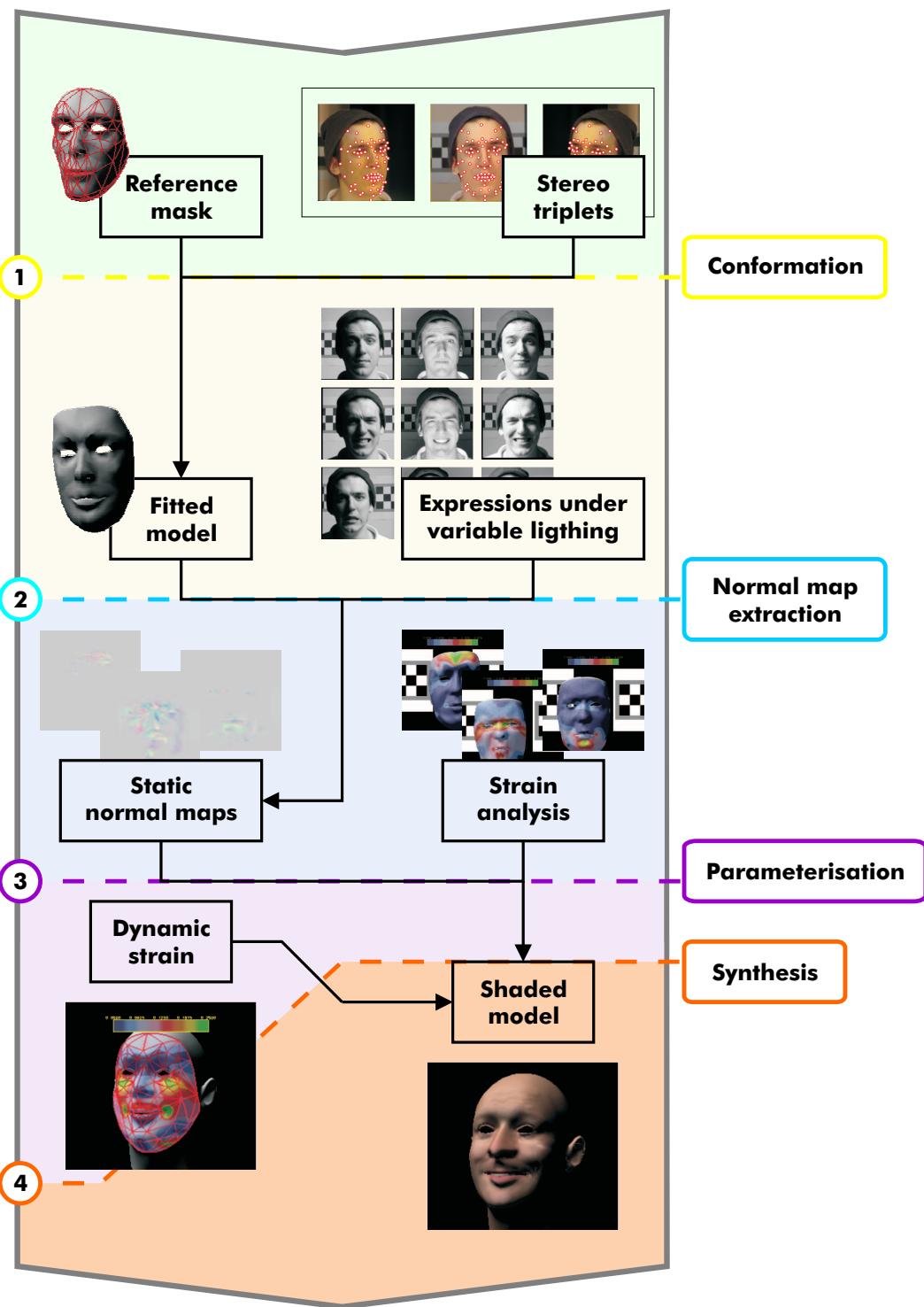
Fig. 5.5: Steps of the performance-based approach to synthetic wrinkling.
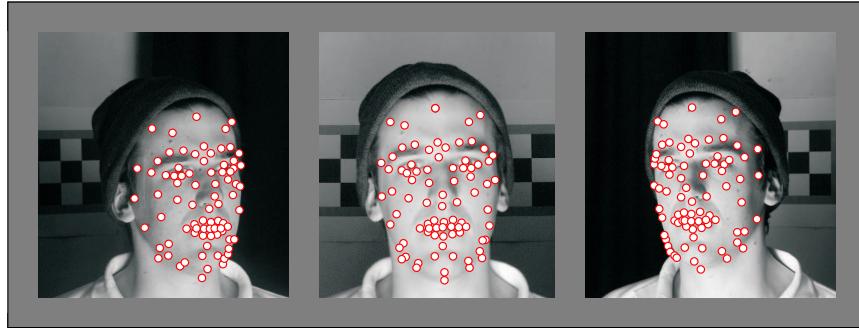
Fig. 5.6: Photograph triplets with labelled markers, enabling their three-dimensional positioning.

5.3.1 Fitting a facial mask to the performer

In order to analyse the deformation of the performer's face when exhibiting a specific expression, we need to be able to retrieve its geometry in some manner. Additionally, the encoding of variations of normal maps requires a supporting surface parameterised over the same domain, as Section 5.3.2 describes in more detail. Capturing an accurate facial model is quite an expensive process, both in hardware and logical resources. Instead, a possible alternative is deriving a coarse approximation by fitting a generic model onto the photographed face, for instance the control surface of BIDS. Furthermore, this would enable deforming a reference polygonal mesh to suit that particular physiognomy (c.f. Section 4.3.1), resulting in a more accurate approximation.

Constructing the BIDS control surface, even when using a fixed patch topology, requires the ability to track the positions of all marker points spanning it. For this purpose the capture setup includes three different cameras, distributed radially around the subject. This enables us to obtain at least two shoots of each of the markers, and then to retrieve their respective three-dimensional positions by reverse stereoscopy.

The application of stereoscopy to this problem assumes accurate knowledge of several camera parameters. These include both position and orientation of the relevant lenses, together with intrinsic characteristics such as focal length and aspect ratio (for digital cameras). As Faugeras et al. describe [FLM92], for the pinhole camera model we can condense all these properties into a single transformation matrix, relating spatial coordinates $\{x, y, z\}$ to their projective plane counterparts $\{u, v\}$. Using homogeneous coordinates:

$$\left. \begin{array}{l} u = \frac{U}{W} \\ v = \frac{V}{W} \end{array} \right\} \quad \text{where } [U, V, W] = [x, y, z, 1] \left[\begin{array}{ccc} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \end{array} \right] \qquad (5.23)$$

For each of the cameras, coefficients c_{ij} can be estimated by identifying a number of
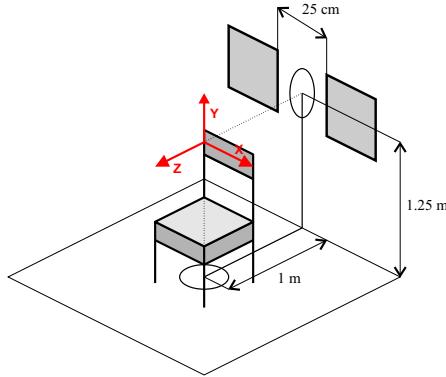
Fig. 5.7: Placement of calibration patterns with respect to the subject in our capture setup.

known points in the projective plane, each of which yields the following two equations:

$$u_k \left(c_{13}x_k + c_{23}y_k + c_{33}z_k + c_{43} \right) = c_{11}x_k + c_{21}y_k + c_{31}z_k + c_{41}$$
$$v_k \left(c_{13}x_k + c_{23}y_k + c_{33}z_k + c_{43} \right) = c_{12}x_k + c_{22}y_k + c_{32}z_k + c_{42} \qquad (5.24)$$

With 2 equations and 12 unknowns per point, we need up to 6 of them to construct the estimate. Additional constraints, like orthonormality, could be put into practice to actually reduce that amount down to 3 (see for instance [FLM92]), yet this would not make a significant difference for the present system.

In the current capture setup (see Figure 5.7), known points are extracted from a pair of calibration targets sitting 1 m behind the origin of coordinates, taken as the position where the performer's face will approximately lie. Both patterns consist of a 3×3 chequerboard measuring 15 cm per side, and they are separated 25 cm from each other in order for at least one of them to appear in every photograph taken. Additionally, cameras are positioned 2 m away the subject, preventing non-linear projective effects such as focal distortion (c.f. [Tsa87] for a detailed description).

After the calibration is performed, the three-dimensional coordinates of a marker point can be retrieved by labelling its position in two of the three different viewpoints. Evaluating Equation 5.24 along with its coordinates on the frontal projective plane yields two equations with three unknowns, and the third constraint can be taken as any of the remaining four identities. Ideally this should be any of the two equations associated with the half of the face where the marker lies, and in particular that involving the horizontal coordinate, as the cameras are rotated around the vertical axis.

Figure 5.8 illustrates the results of adapting a generic model, the popular *DECface* [Wat87], to the particular physiognomy of the subject of the capture. In order to enable this for every single expression, the whole labelling process has to be repeated, making it rather tedious. However, should this be done in combination with Motion Capture hardware, it would render all manual processing unnecessary. Yet most optical MoCap equipment is sensitive to lighting variations, so in many cases such a shortcut may not be possible.

Fig. 5.8: Generic face model (DECface) fitted onto the performer, by application of BIDS together with the tracked markers.

5.3.2 Extracting normal maps from expressions

Following the introduction of photometric stereo by Woodham in [Woo80], successive research has extensively used the Lambertian lighting model for the extraction of shape information from photographs. Rushmeier et al. were the first to introduce its application to the reconstruction of normal maps from an actual object [RTG98], using a fixed lighting set. More recent work by Debevec et al. [DHT00] has extracted full reflectance data from a human face, whose lighting properties differ greatly from the Lambertian model. These results, combined with local separation of diffuse and specular components, provide a better estimate of the normal field on the skin surface. However, the articulated lighting system used in this approach is an overly-strong requirement for the purposes of out approach, leading us to construct the current capture setup around the simpler paradigm described by Rushmeier.

In the case of a Lambertian surface illuminated by a source with radiance L_o, the reflected radiance L_r at a given point is defined by the following equation:
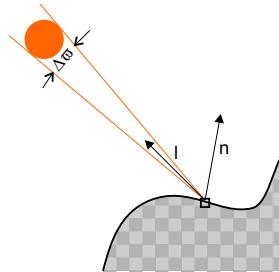
$$L_r = \rho \left(L_o \, \Delta\omega/\pi \right) n \cdot l \tag{5.25}$$

where ρ is the Lambertian reflectance (albedo) at that point, l a unitary vector in the direction of incidence of the light, and n the corresponding surface normal. For all points

A) Concurrent 2 MP cameras, fired remotely

B) Interfit 3200K kit, fitted with 275W tungsten bulbs

D) 50cm. ∅ silver reflectors
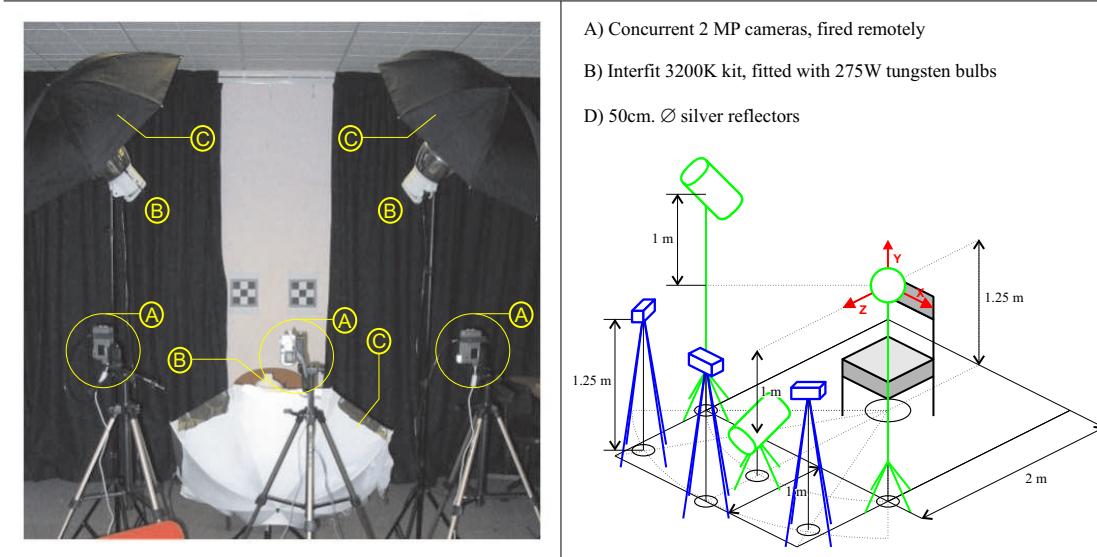
Fig. 5.9: Photographic setup used for the captures

on the surface we can assume the solid angle $\Delta\omega$ to be approximately constant, provided that the light source is sufficiently distant in comparison with the scale of the object being illuminated. In our setup, reflectors are positioned 1.73 m away from the subject, ensuring that solid angle variations are insignificant.

Taking the constant $L_o' = L_o\,\Delta\omega/\pi$ as the effective radiance emitted by any light source in the system, we can then use three of them to derive a straightforward approximation of the surface normal as in Eq. 5.25:

$$
\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{1}{\rho L_o'} \begin{bmatrix} (l_1)_x & (l_1)_y & (l_1)_z \\ (l_2)_x & (l_2)_y & (l_2)_z \\ (l_3)_x & (l_3)_y & (l_3)_z \end{bmatrix}^{-1} \begin{bmatrix} (L_r)_1 \\ (L_r)_2 \\ (L_r)_3 \end{bmatrix}
\tag{5.26}
$$

where $(L_r)_i$ stands for the reflected radiance observed when only the light source with incidence vector l_i is active. In the capture setup depicted in Figure 5.9, the three lights are aligned along the edges of a squashed tetrahedron, whose apex lies on the subject's face. This distribution guarantees a variety of directions sufficient to estimate the skin's normal field accurately [McG98]. For the same reasons that solid angles $\Delta\omega$ can be considered constant, so can incidence vectors l_i.

In addition to these considerations for normal estimates to be accurate, we have to enforce Lambertian properties on the skin surface by masking specular reflection. Ideally, this could be done using orthogonal polarizing filters both in lighting kits and camera lenses; however there is a more affordable option, which is applying a thin layer of matte foundation on the performer. This also has the advantageous side effect of evening out the albedo, improving the precision of the estimate.
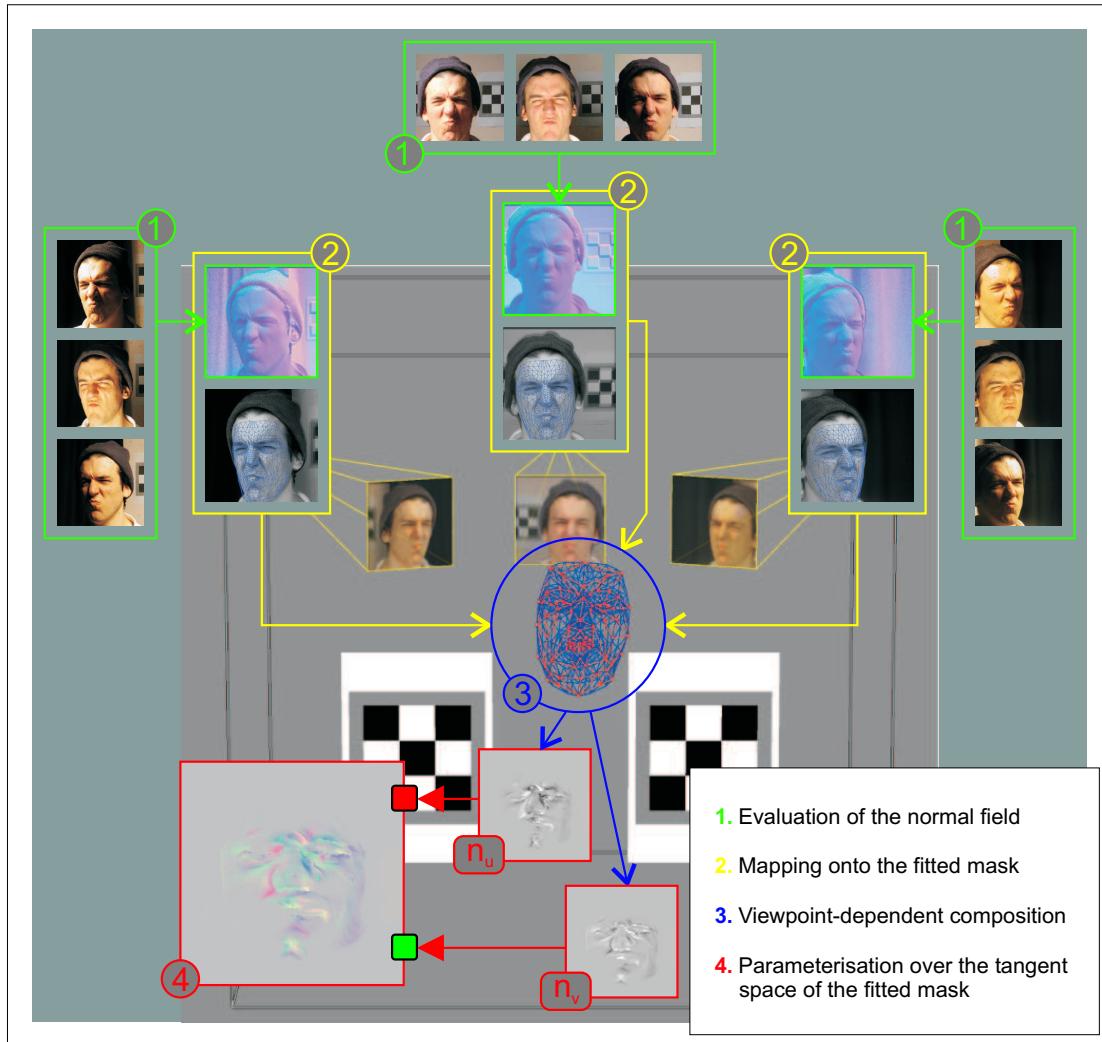
Fig. 5.10: Steps in the production of normal maps from photographs under variable light-ing, combining three different viewpoints. Normal field data is represented in the RGB colour space, mapping coordinate intervals $[-1..1]$ onto unsigned bytes. Similarly, normal map components $\{n_u, n_v\}$ are expressed as red and green values.

From normal fields to a normal map

After evaluating Equation 5.26 with input from the three different lighting conditions that are reproduced for each captured expression, the results will be referred to the image domain of each of the three camera viewpoints (see Figure 5.10.1). Yet we aim to parameterise normals over a unique texture space, mapped onto the reference face model, in order to make possible the transfer of data across different physiognomies.

This reparameterisation is possible thanks to a mapping relating image space to the fitted mask, which is defined by reversing its projection onto each of the viewports (see Figure 5.10.2). This results in three different normal field definitions applied onto the polygons of the reference model, and consequently onto its texture space. In order to compose them into a single one, we can use the averaged normals at the fitted vertices to measure the alignment between the skin surface and each of the camera planes. The local contribution of each of the three normal fields is then weighted according to these factors.

Finally, the combined expression of the normal fields, parameterised over the reference texture space, is encoded in the tangent space of the fitted mask (see Figure 5.10.4). This enables us to express the captured data as a distortion of the natural normal field of such surface (mask), and to transplant it to a different one.

Tangent space of the fitted mask

Consider the ith polygonal face of the fitted model, spanning vertices $\{\mathbf{p}_l, \mathbf{p}_m, \mathbf{p}_n\}$ with texture coordinates $\{\mathbf{t}_l, \mathbf{t}_m, \mathbf{t}_n\}$. The first two axes of the tangent space according to texture parameterisation are the rows of the following matrix:

$$\left[\begin{array}{c} e_u \\ e_v \end{array} \right] = \left[\begin{array}{c} \mathbf{t}_m - \mathbf{t}_l \\ \mathbf{t}_n - \mathbf{t}_l \end{array} \right]^{-1} \left[\begin{array}{c} \mathbf{p}_m - \mathbf{p}_l \\ \mathbf{p}_n - \mathbf{p}_l \end{array} \right] \tag{5.27}$$

The remaining axis e_w can be taken as the face normal, which is trivially orthogonal to the other two. In order to have a smooth definition of the tangent frame field, axes are averaged at every vertex, orthonormalized, and then interpolated back over each polygonal face. This vector data can be rendered back onto the reference texture space over which the combined normal field is defined. Then the encoding is directly performed on a per-texel basis, leading to the final expression of the normal map.

5.3.3 Deformation analysis

When performing a particular expression, the rate of deformation of the skin surface can be approximated by performing strain analysis on the corresponding fitted mask. However, we do not have an analytic definition of the displacement function for this model in order to compute the derivatives required by the infinitesimal strain tensor (Equation 5.8). While discrete approximations give a solution to this problem, they are also highly dependent on the polygonal topology of the reference model, exhibiting significant flaws whenever this is irregular.

Opportunely, the strain tensor on the tangent plane of any model deformed by BIDS corresponds to the equivalent metric on the control surface, and performing the strain evaluation on such a control structure yields a smooth and regular tensor field. For each micro-patch $\mathbf{s}_i$ composing this piecewise surface, the displacement function can be computed by evaluating its parametric expression with the displacements of the corresponding Bézier points. Labelling barycentric coordinates as $\{r, s, t\}$:

$$
\mathfrak{u}_i(r, s) = \mathbf{s}_i'(r, s) - \mathbf{s}_i(r, s) = \tag{5.28}
$$

$$
\sum_{l,m,n}^{3} (\mathfrak{u}_i)_{lmn} B_{lmn}^3(r, s, 1 - (r + s))
$$
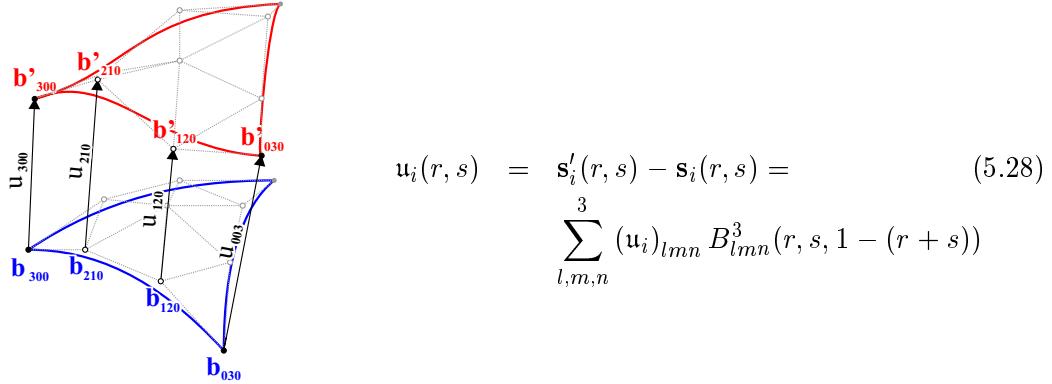
The natural choice of basis for the tangent plane of the micro-patch $\mathbf{s}_i$ is the curvilinear coordinate system defined by $\{e_r, e_s\} = \{\frac{d\mathbf{s}_i}{dr}, \frac{d\mathbf{s}_i}{ds}\}$. Using the notation introduced in Figure 3.19 for subpatches, we can now reinstate the strain tensor definition in Equation 5.8 as:

$$
S_i = J(\mathfrak{u}_i) = \begin{bmatrix} (\mathfrak{u}_i)_{,r}^r & (\mathfrak{u}_i)_{,s}^r \\ (\mathfrak{u}_i)_{,r}^s & (\mathfrak{u}_i)_{,s}^s \end{bmatrix} =
$$

$$
\begin{bmatrix} e^r \\ e^s \end{bmatrix} \begin{bmatrix} \frac{d\mathfrak{u}_i}{dr} \\ \frac{d\mathfrak{u}_i}{ds} \end{bmatrix}^T = 3 \begin{bmatrix} e^r \\ e^s \end{bmatrix} \begin{bmatrix} (\mathbf{s}_i'^r - \mathbf{s}_i{}^r) - \left(\mathbf{s}_i'^t - \mathbf{s}_i{}^t\right) \\ (\mathbf{s}_i'^s - \mathbf{s}_i{}^s) - \left(\mathbf{s}_i'^t - \mathbf{s}_i{}^t\right) \end{bmatrix}^T \tag{5.29}
$$

Analogously to e^u and e^v in Equation 5.8, e^r and e^s stand for the duals of e_r and e_s in the tangent space of $\mathbf{s}_i$.

However, the normal map data collected as described in Section 5.3.2 is expressed in the reference texture space $\{u, v\}$, rather than in the tangent plane of the control surface. Mapping the strain tensor field onto $\{u, v\}$ is quite simple, just by interpolating the texture coordinates of the markers over the patch domain; yet the change of coordinates also requires the strain components to be referred to the new axes. Following the rules of transformation for mixed tensors:

$$
S(u, v) = \begin{bmatrix} e^u(u, v) \\ e^v(u, v) \end{bmatrix} \begin{bmatrix} \mathfrak{u}_{,r}^r(r(u,v), s(u,v)) & \mathfrak{u}_{,s}^r(r(u,v), s(u,v)) \\ \mathfrak{u}_{,r}^s(r(u,v), s(u,v)) & \mathfrak{u}_{,s}^s(r(u,v), s(u,v)) \end{bmatrix} \begin{bmatrix} e_u(u, v) \\ e_v(u, v) \end{bmatrix}^T \tag{5.30}
$$

with $\{e_u, e_v\} = \left\{ \frac{d\mathbf{s}}{dr}\frac{\partial r}{\partial u} + \frac{d\mathbf{s}}{ds}\frac{\partial s}{\partial u}, \frac{d\mathbf{s}}{dr}\frac{\partial r}{\partial v} + \frac{d\mathbf{s}}{ds}\frac{\partial s}{\partial v} \right\}$. After performing this conversion, the symmetric form of the resulting strain tensor (see Equation 5.5) can be subjected to eigen-decomposition to represent just compression, as in Equation 5.6. The outcome of this process is independent of the scale of the captured face, as it is referred to the

Fig. 5.11: Strain diagrams showing the modulus of compression, computed as the determinant of the tensor field $\bar{S}$ defined over the fitted mask. This happens to be a meassurement of local area reduction (see Section 5.2.1), arising with more intensity in the areas coded with colours to the right of the inset bar (down to 1/4 of the original area). Directions of maximum compression are oriented perpendicularly to the isocurves of such a scalar field.

tangent space induced by the reference texture map. This enables us to construct a unique representation of the rate of deformation for all physiognomies to which the same facial mask can be fitted (all those than can be labelled with the reference marker set).

5.3.4   Building a functional model of the captured data

Using the methods described in the previous sections, we have produced a model-independent representation of the normal field variations when performing a set of 16 facial expressions, together with the components of the corresponding compression tensor fields. Since both collections of data are referred to the same texture space, it is possible to express each of them as a local function of the other. In this particular case we aim to model the normal data, constituting a visual representation of wrinkling, as a function of the magnitude and direction of local strains that drive the actual phenomena, condensed in the components of the compression tensor. Assuming that this function is analytic, each of the components $\{n_u, n_v\}$ of the normal map at a given point $\mathbf{t}$ of texture space will admit a polynomial approximation in the form:

$$\hat{n}_i(\bar{\varepsilon}_{uu}, \bar{\varepsilon}_{vv}, \bar{\gamma}_{uv})|_{\mathbf{t}} = \left(\sum_l \alpha_l^i \, (\bar{\varepsilon}_{uu})^l + \sum_m \beta_m^i \, (\bar{\varepsilon}_{vv})^m + \sum_n \lambda_n^i \, (\bar{\gamma}_{uv})^n + n_i^0 \right)\Bigg|_{\mathbf{t}} \qquad (5.31)$$

where n_j^0 are the normal map coordinates in the neutral position, captured under null strain. Mixed terms are discarded in this approximation, since we model the components of strain as non-intercorrelated variables (thereby all mixed moments are null); the purpose of this assumption is to prevent the model from being biased by the specific choice of poses retreived for its construction (e.g. performing more frowns that brow-raises).

Coefficients $\alpha_l^i\big|_{\mathbf{t}}$, $\beta_m^i\big|_{\mathbf{t}}$ and $\lambda_n^i\big|_{\mathbf{t}}$ can be estimated by minimizing the following error metric combining the values $\{n_u^j, n_v^j\}$ of the different normal maps at $\mathbf{t}$:

$$err_i(\mathbf{t}) = \sum_{j=1}^{16} \left\| \hat{n}_i(\bar{\varepsilon}_{uu}^j, \bar{\varepsilon}_{vv}^j, \bar{\gamma}_{uv}^j)\big|_{\mathbf{t}} - n_i^j\big|_{\mathbf{t}} \right\| \tag{5.32}$$

where $\left\{ \bar{\varepsilon}_{uu}^j\big|_{\mathbf{t}}, \bar{\varepsilon}_{vv}^j\big|_{\mathbf{t}}, \bar{\gamma}_{uv}^j\big|_{\mathbf{t}} \right\}$ represent the compression tensor values at $\mathbf{t}$ measured for the jth pose. This minimization is an instance of the well known *Least Squares Optimization* (LSO) problem [PTF92]. Therefore, coefficient vector $C^i = \left[\alpha_1^i \cdots \alpha_L^i, \beta_1^i \cdots \beta_M^i, \lambda_1^i \right.$ $\left. \cdots \lambda_N^i \right]$ can be computed through the generic solution for LSO:

$$C^i = \arg\min(err_i) = \left(\sum_{j=1}^{16} n_i^j\, S_j \right) \left(R + \sum_{j=1}^{16} S_j\, S_j^T \right)^{-1} \tag{5.33}$$

taking S_j as the vector of strain powers $\left[\bar{\varepsilon}_{uu}^j \cdots (\bar{\varepsilon}_{uu}^j)^L, \bar{\varepsilon}_{vv}^j \cdots (\bar{\varepsilon}_{vv}^j)^M, \bar{\gamma}_{uv}^j \cdots (\bar{\gamma}_{uv}^j)^N \right]$ measured in the jth pose, for which the corresponding sample of the normal map is given as $\{n_u^j, n_v^j\}$. In addition, R stands for the usual regularization matrix in the form $diag(\epsilon)$ with $\epsilon \to 0$.

All previous derivations have been performed for generic ranges $[1..L]$, $[1..M]$ and $[1..N]$ of indices l, m, n in Equation 5.31, respectively. Yet in practical application these intervals have to be fixed. Finding the suitable degree of the polynomial approximation can be regarded as trying different values for $\{L, M, N\}$, and then choosing the combination that yields a more accurate global estimate of the captured normal data. Taking W and H as the horizontal and vertical dimensions of texture space (in texels), we can evaluate the *mean squared error* of $\hat{n}_j^i$ as:

$$mse_j^i = \frac{1}{WH} \sum_u \sum_v \left(\hat{n}_j^i\big|_{[u,v]} - n_j^i\big|_{[u,v]} \right)^2 \tag{5.34}$$

In order to measure the scale of the results of Equation 5.34, they can be compared with the variance of the corresponding normal component, identically averaged over the texture domain (σ_i^2). The following plots portray the ratio of these two magnitudes for each of the captured poses, with varying configurations of the degree tuple $\{L, M, N\}$:

Error in the approximation of n_u   **Error in the approximation of n_v**
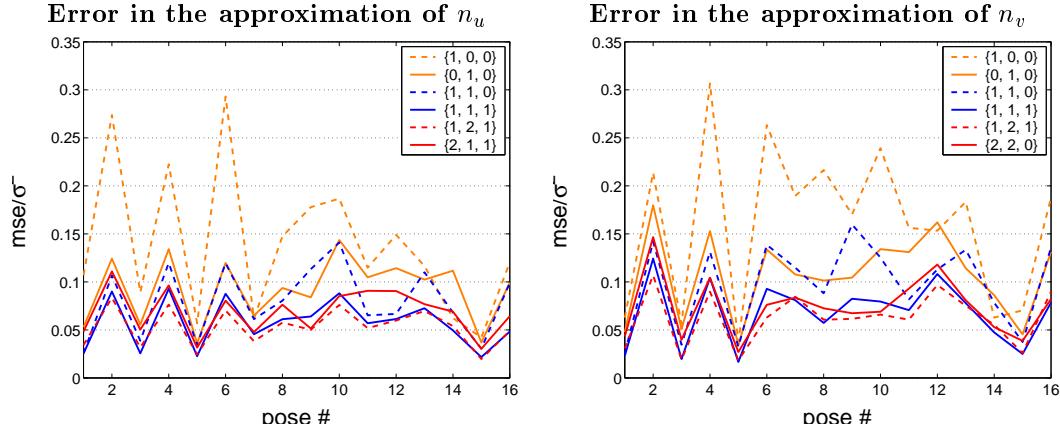
The diagrams above illustrate how rough linear approximations can be built on $\bar{\varepsilon}_{uu}$ and $\bar{\varepsilon}_{vv}$ alone ($\{1,0,0\}$ and $\{0,1,0\}$, repectively), but only $\{0,1,0\}$ shows reasonable convergence with more accurate models. This indicates that compression along the vertical direction of the texture (corresponding to the same direction in the face) is the more predominant source of wrinkling. A linear expression of both terms of strain provides a lesser error, while including shearing ($\bar{\gamma}_{uv}$) in the model doesn't significantly improve accuracy, proving that this factor is not so relevant to the wrinkling process (as could be expected from the elastic properties of the tissue).

Higher degree polynomial approximations, of which only $\{2,2,0\}$ and $\{1,2,1\}$ are shown here, hardly outperform the trilinear case. This can be explained as the process itself being quasi-linear on the components of the strain tensor. The residual error that bounds the performance of the polynomial models constructed responds to the portion of the normal map that is not related to the measured local compression; however, being such a small fraction of the variance ($1/20$), it is clearly negligible. Therefore we can conclude that the rate of change of the normal map along the deformation can be accurately represented by low-degree local approximations. An Chapter 4 describes, this is something that facilitates its evaluation with the limited resources of dedicated graphics hardware.

5.3.5   Application

Previous sections presented a procedure to analyse the skin compression exhibited by different facial expressions under a unique reference system. This was enabled by taking a fixed piecewise topology for the control surface and adapting it to a predefined set of tracked markers; subsequently, the strain analysis performed on this surface was related to a reference texture space, providing the framework for a uniform representation.

Similarly, any facial model can be labelled with the same marker layout used before, and have a control surface fitted with an identical topology. Thereafter, any deformation exerted on it by BIDS can be subjected to strain analysis in the same reference space; such strain data can then be taken as input for Equation 5.31 in order to synthesise the corresponding normal map. Furthermore, since the control surface also acts as a bearer of the reference texture coordinates, these can be consistently mapped onto the new
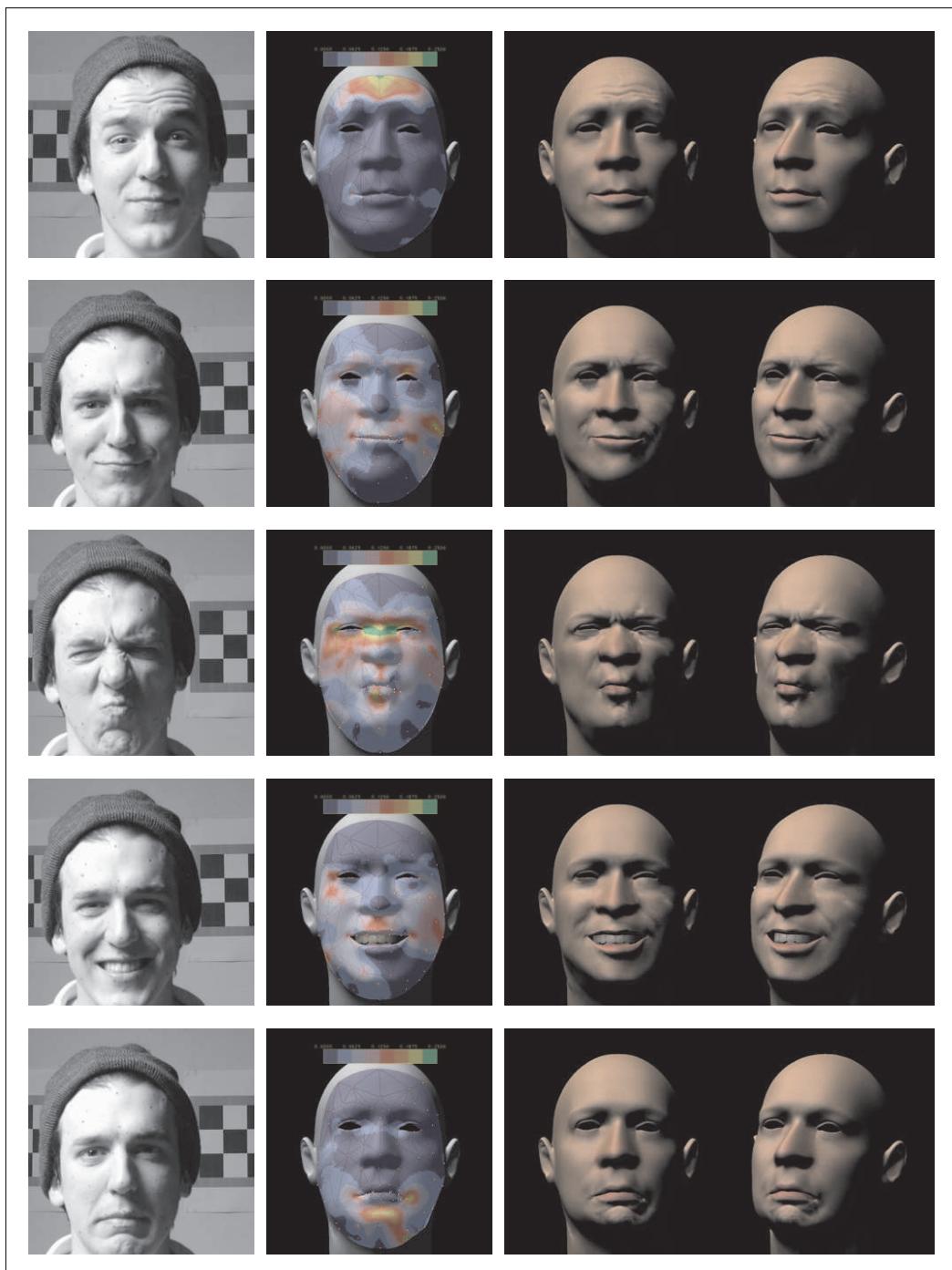
Fig. 5.12: Results obtained by combining BIDS deformation with the wrinkling model proposed in Equation 5.31.
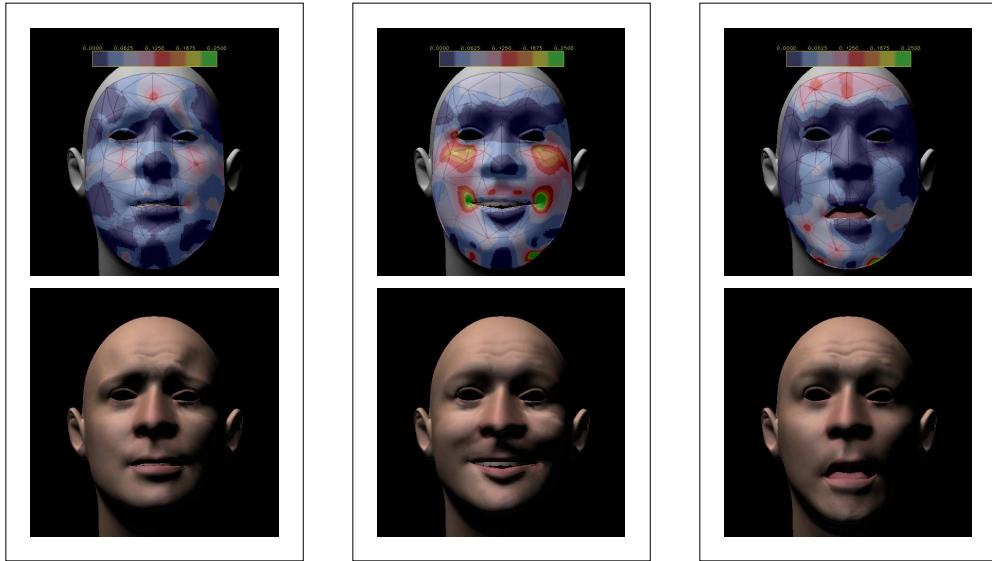
Fig. 5.13: Wrinkling model evaluated on a facial mesh controlled by a piecewise surface with the same topology and marker layout used in its construction.

model in order for it to convey the resulting wrinkling effect. This is achieved at the shading stage of the rendering pipeline by reconstructing the distortion represented by the normal map over the tangent frame field induced by reference texture coordinates. Further details on this are given in Chapter 6

Using the techniques explained in Chapter 4, we can retarget some of the captured expressions onto an arbitrary facial mesh, in order to demonstrate the coherence of the wrinkling model when applied to a similar physiognomy. Figure 5.12 illustrates this. Additionaly, new expressions retargeted from Motion Capture are portrayed in Figure 5.13. While these examples are produced using the original marker layout, it is also possible to apply the model to Motion Capture driven by different marker sets, as long as their texture coordinates are consistently chosen (e.g. derived from the same texture projection of the reference mesh). Figure 5.14 show the results for such datasets.

Despite the fact that the model of skin wrinkling proposed in this Section is independent of its polygonal representation, it is still linked to the specific way facial tissue wraps the subject's physiognomy. Not all individual skins react in the same manner to compression, as their mechanical properties may vary largely; furrows appear in greater or lesser degree depending on factors like dermis hydration, thickness of underlying fat layers, or, more significantly, age (c.f. [Eld77]). Also, some particular crease patterns are exclusive to a few individuals, and even amongst those their location may differ. However, in the same manner that different texture maps can be applied to change the appearance of a facial model, *wrinkle maps* containing the coefficients used by Equation 5.31 can be used to change the way a facial model portrays fine-scale tissue deformation.
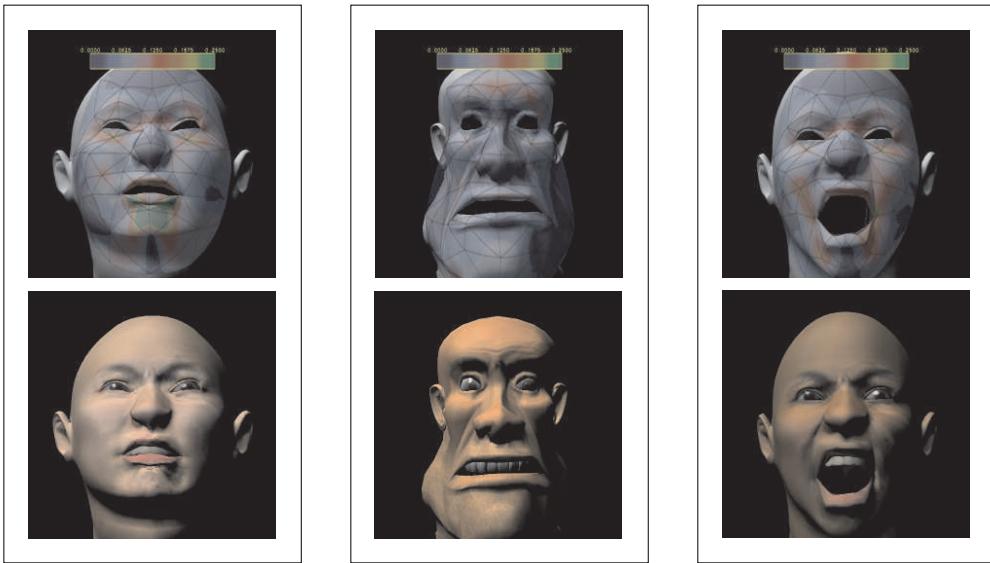
Fig. 5.14: Wrinkling model evaluated on a variety of meshes labelled with a marker layout different from that used in its construction.

5.3.6 Weaknesses and limitations

Unquestionably, one of the main constraints limiting the quality of this approach is the resolution with which the cameras can capture facial skin; given the small scale of some crease lines, this has to be reasonably high in order to capture the shading variations in the furrowing around them. In addition, the distance from the subject at which cameras have to placed (not to interfere with the lighting equipment) requires optical zoom in order to use their maximum resolution. With the entry level hardware used in the current setup (2 megapixel digital cameras with 3x augmenting lenses), the area of interest will have roughly one thousand pixels height out of the 1200×1600 total resolution.

Furthermore the combination of three different views, besides improving the normal estimate by using at any moment the camera that is best aligned with the skin, does also introduce seams as the captured crease lines may not align properly across views. In order to compensate for this, we have to scale down the resolution of the shots, finally producing a normal map of 512×512 samples. Another alternative could be to reduce the angle separation between the views, yet this would also result in worse estimates for the position of the markers used to analyse strain and map the normal field onto texture space. Ideally, integrating more cameras should mitigate all these problems, as long as space and budget allow.

While some crease lines may not be captured because of their small scale, there is another factor to bear in mind in this process, which is the quality of the approximation provided by the fitted reference mask. There are certain critical areas, such as the confluence of the eyebrow arch and the nose, where the curvature of the face is relatively

a) Fitted mask vs. actual physiognomy   b) Extracted normal map   c) Resynthesized normal map applied to a different model
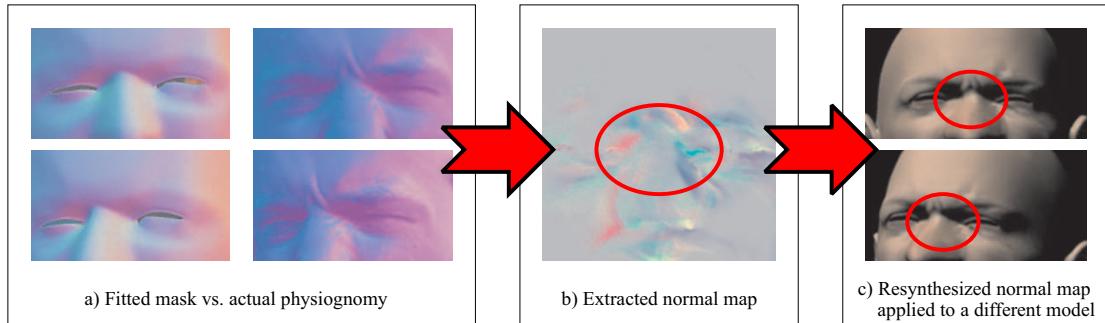
Fig. 5.15: Errors caused by misapproximation of the facial geometry: the example in a) illustrates how the actual deformation in the eye depressions is much more drastic than in the fitted mask; as a result, the extracted normal map (b) does not portray sharp creases on the top on the nose. Finally, in the re-synthesised model (c) the parameterisation has smoothed out all seams of wrinkling, as the data conflicted with other less acute samples of frowning.

high. If these regions are not properly approximated with the reference model, major misalignments may occur in the combination stage, which brings in a strong gradation of two of the views. As a result we lose most of the detail of the furrowing below and around this particular point (Figure 5.15 describes this in more detail). Sensible solutions to the fitting problem are using as reference mask a geometric model that is more consistent with the face of the subject (e.g. a raster scan), and increasing the density of markers in the particular area where the problem arises, so that BIDS produces a more accurate result.

The marker layout also plays an important role besides mapping the captured data onto a reference space, which is related to the proper analysis of strain. While coarse distributions may be sufficient to produce a good geometric approximate of the deformation in regions like the forehead, the planar motion of tissue is extremely relevant to the wrinkling phenomena experienced there. Therefore, if the local marker concentration is not dense enough, this system would not be able to generate the richness of strain patterns occurring in those areas, and relate them to the corresponding furrowing. This is also relevant for synthesis, as the compression tensor fields have to be coherent between the captured model and that onto which the effect is reproduced. Otherwise the skin behaviour may differ.

For instance, in the model used in Figure 5.12, six markers track the skin motion on the forehead, and are able to differentiate frowning and surprise patterns. However, the MoCap marker set used for Figure 5.14 only has two active points in this region (the rest are fixed), resulting in some of the frowning detail not being portrayed, as well as mid-eyebrow furrows only appearing slightly in surprise configurations.

Another aspect to bear in mind is that the performance-based model constructed before is conceived for static expressions, and therefore some dynamic aspects of wrinkling are not considered. For instance, skin viscosity also affects the creation of wrinkles,

making them perceptible for a short time after the causing compression is relieved. This can actually be modelled with a time-fading model for strain analysis, yet estimating the actual rate of fading remains impossible without actual dynamic analysis. Video techniques would be required for this particular purpose.

5.4   Summary

This Chapter has introduced two different approaches complementing the geometric warping methods presented in Chapter 3, by recreating facial creases and furrows according to the ratio of deformation sustained by the skin surface. The first method attempted to reproduce these features by displacing skin geometry along the normal field of a control structure, while the second approach chose instead bump mapping techniques in combination with a performance-based framework. The results of this second method attain a level of realism in representing facial expression that is otherwise unavailable with purely geometric approaches. In addition, it also integrates particularly well with the performance-based systems expounded in Chapter 4, allowing us to construct a comprehensive framework in one single capture facility.

However, bump mapping does not reproduce the actual geometry of the wrinkling phenomena, but just the corresponding shading effects. This is satisfactory in most circumstances, but in close, non-frontal takes we might want to appreciate the bulge of some of the furrows. For that purpose, the displacement mapping technique described in Section 5.2 would be helpful, after integrating the extracted normal maps as height maps. Yet the density of the mesh may not be enough for an accurate representation. A sensible alternative could be to combine both techniques, separating high and low frequency components of the normal map, and then representing these as displaced geometry while conveying sharper features with the usual bump mapping technique.

The parameterisation experimentally obtained in Section 5.3.4 can be implemented in a rather efficient way, requiring not much more data and computational resource than conventional static bump-maps. Chapter 6 will describe an implementation entirely built on dedicated graphics hardware, analysing the rate of deformation and generating the corresponding normal maps in real-time.

6. HARDWARE-ACCELERATED FACIAL ANIMATION

6.1   Introduction

As the introductory Chapter outlined, one of the main focuses of this work is developing techniques that can be efficiently implemented in real-time. Dedicated hardware plays an important role in this task, providing specialized computing resources that operate alongside the rendering pipeline, and have direct access to the dedicated graphics memory. This prevents performance bottlenecks created by the saturation of the graphics bus when updating large deformed geometries from their temporary storage in the system's main memory. Furthermore, it also frees up part of the central processor's computing power, allowing it to perform other tasks such as Motion Capture retargeting or speech synthesis.

Besides the aspects concerning deformation algorithms, the programmable capabilities of dedicated graphics hardware also allow specific per-pixel shading to be implemented in real-time. This is required to represent facial creases and furrows as normal maps, as described in Chapter 5. In addition, part of the evaluation of the performance-based wrinkling model presented in Section 5.3 can be deferred to the rasterization stage, contributing to a further integration of the complete Facial Animation system in the hardware framework.

However, the sheer benefits of graphics hardware do not come without additional costs. The specialization of the computing resources provided, whilst advantageous performance-wise, turns out to be constraining for any generic usage out of their usual transformation and shading context. This requires very specific reformulations of algorithms in order to benefit from them. Additionally, the rigid programming model graphics hardware exports, embedded in the rendering pipeline itself, takes its toll in the way data has to be formatted and pre-processed before these algorithms can even run.

Precisely, Sections 6.3 and 6.4 of this Chapter will be dedicated to discuss the particular problems of endowing the techniques described in Chapters 3 and 5 with hardware acceleration. But first, Section 6.2 will introduce the specifications of the framework we will be working with.
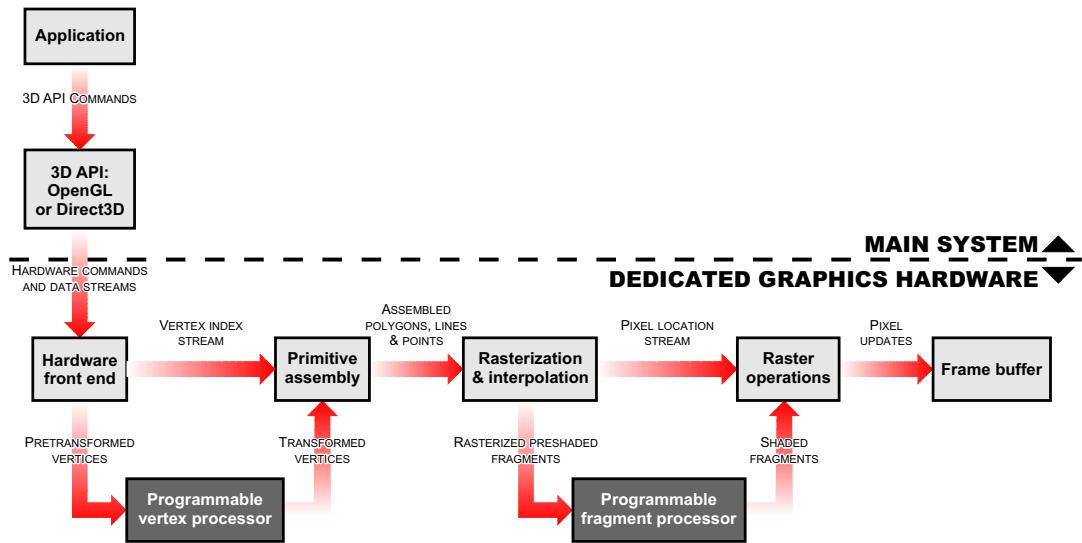
Fig. 6.1: Flow diagram for the architecture of dedicated graphics hardware. Source: [nvi02]

6.2   The Cg/HLSL programming model

The original design of 3D-graphics accelerators was mostly limited to a simple triangle setup and rasterization pipeline, operating in conjunction with in-card colour and depth buffers. When the performance of such devices improved beyond the rate at which the processor could provide data, alternative designs (notably the GeForce 256 by nVidia) added geometric operations like transformation and lighting to the functionality of the hardware pipeline to balance it out. This also meant that some of the dedicated memory of the card, usually reserved just for textures and buffers, could be used to store geometry and process it without relying on the underlying central resources.

Subsequent iterations of this design saw the scale of integration soar, incorporating not only more advanced features, like superscalar operations and parallel pipelines, but also some flexibility by permitting part of the pipeline to be reprogrammed. Initially, the Ti models of the GeForce 3 series enabled limited reprogramming of the transformation and lighting stage. The GeForce 4 models that followed, along with ATI's 8500 series, translated this philosophy to the pixel shading stage, yielding the flow model in Figure 6.1.

Ever since, this architecture has been widely adopted by many other manufactures (S3, Matrox, XGI...) as it was standardised by the tightly hardware-dependent specification of DirectX (c.f. [Bra03]). Successive efforts have been focused on strengthening the framework by allowing programs with a larger number of instructions, and by providing more registers to carry their input. Also, the programming model has been improved by including some of the standard features in conventional general-purpose processors;

Tab. 6.1: Features of programmable vertex units

Vertex shader version:	1.4	2.0	2.a	3.0
Supported hardware:	NV25	R300	NV3x (FX 5K)	NV40 (FX 6K)
	(GeForce 4)	(ATI 9200)	R350 (9700+)	R360 (X300+)
Maximum program length:	128	256	256	512+
Temporary registers:	12	12	16	32
Constant registers:	96	256	256	256+
Static branching:	Yes	Yes	Yes	Yes
Dynamic branching:	No	No	Yes	Yes
Predicates:	No	No	Yes	Yes

Tab. 6.2: Features of programmable fragment units

Pixel shader version:	2.0	2.a	2.b	3.0
Supported hardware:	R300	NV3x	R360	NV40
	(ATI 9200)	(GeForce FX 5K)	(ATI X300+)	(FX 6K)
Maximum program length:	32 + 64	512	512	512+
Temporary registers:	12	22	32	32
Constant registers:	32	32	32	224
Static branching:	Yes	Yes	Yes	Yes
Dynamic branching:	No	No	No	Yes
Predicates:	No	Yes	No	Yes

namely dynamic flow control and predicates for conditional registry assignment (refer to [ita00] for further details). Following the specifications of DirectX, we can track the evolution of programmable vertex units in Table 6.1. Similarly, the developments in fragment processors are depicted in Table 6.2.

Before the arrival of DirectX 9.0, vertex and fragment programming was done in the assembly language of a virtual machine compliant with the corresponding shader model, which was specially cumbersome and difficult to maintain. In response, nVidia teamed up with Microsoft and developed a higher level progamming language: Cg (as in "C for graphics"). Cg allows defining vertex and pixel shaders using similar syntax to the well-known C programming language [KR88]; in particular, the operations of scalar data types such as `float` and `double` are extended to work with tuples (e.g. `float4`, `double2`) that represent geometric entities (vertices, normals, texture coordinates, etc.). Furthermore, a limited matrix algebra is included to work with such tuples, providing some vector operations such as dot products (`dot`) and normalisations (`normalize`), vector-matrix multiplications (`mul`) and other simple matrix operations (e.g. `transpose`).

The flow of Cg programs is contained in the body of the `main` function, taking as parameters per-vertex or per-pixel data stored in the hardware registers, and returning a single output (transformed and lit vertices in the case of vertex programs, and pixel colour in the case of fragment programs). Constant parameters that are common to all vertices and fragments of a single primitive can be tagged as `uniform`, so that they are kept in a specific set of registers whose values are fetched just once for each rendering

call. For more details on the syntax and application of Cg, consult [FK03].

Microsoft embedded Cg in DirectX, rebranding it as HLSL (High-Level Shading Language); this allowed Cg developers to program the cards of third party manufacturers in a similar fashion to nVidia models of the same generation. After the break-up of the joint venture between these two companies, HLSL remains as the standard for DirectX developers, while the equivalent Cg is supported in OpenGL [1] [SA96] by the extensions made to this standard by nVidia.

Despite their continuous update, the present programming models for both vertex and fragment units are still limited by the architecture outlined at the beginning of this section, as well as by the paradigm exported by the relevant graphics API (DirectX or OpenGL). In particular, the data submitted to graphics hardware has to combine both geometric and topologic deformation, and that requires that all primitives (triangles, lines, points) processed at the same time share identical programs and constant (uniform) parameters. This will have a significant effect in our hardware-accelerated implementations of the algorithms previously presented in this work, as the following sections describe in further detail.

6.3 Hardware-accelerated implementation of Planar Bones

The relative simplicity of Planar Bones (see Section 3.4) allows for a fairly efficient hardware implementation. Yet every single vertex of a given polygon can be attached to a different collection of control facets, or more precisely, to particular affection regions of such control facets. According to the restrictions on primitive handling, we cannot process vertices of the same polygon independently; therefore all control facets involved in a rendered polygon have to be taken into consideration at the same time by the vertex program performing its deformation. This is possible using variable indexation, as illustrated by the following vertex program:

```
// Per-vertex input data
struct VTX_IN
{
  float4 vtx : POSITION; // position in world coordinates
  float3 nrm : NORMAL; // normal
  float2 tc : TEXCOORD0; // texture coordinates

  float4 inds0 : TEXCOORD1; // control face region indices
  float4 wgts0 : TEXCOORD2; // control face region weights
};


VTX_OUT main(
  // transposed matrices of each of the control face regions
  uniform float3x4 M_Ts[MAX_REGS],
  // vertex data
          VTX_IN vin,
```

---

[1] http://www.opengl.org

```
  // lighting and transformation data
  ... )
{
VTX_OUT vout;

float4 inds_ = vin.inds0;
float4 wgts_ = vin.wgts0;

float4 vtx = float4(0.0, 0.0, 0.0, 1.0);
float3 nrm = float3(0.0, 0.0, 0.0);

  // Computing the contribution of each of the four _
  // most characteristic control face regions
  for(int i = 0; i < 4; ++i)
  {
    vtx.xyz += mul(M_Ts[inds_.x], vin.vtx)*wgts_.x;
    nrm.xyz += mul(vin.nrm, pseudo_inv(transpose((float3x3)M_Ts[inds_.x])))*wgts_.x;

    inds_ = inds_.yzwx;
    wgts_ = wgts_.yzwx;
  }

// Lighting and transformation
...
```

As commented in the code, this implementation considers only the four most significant affection regions affecting each vertex (referenced through VTX_IN::inds0); should those not exist, their corresponding weightings will be zero (in VTX_IN::wgts0).

The function pseudo_inv performs an ad-hoc inverse for the transformation matrices of affection regions, characterised by having a third row unitary and orthogonal to the plane spanned by the other two.

```
float3x3 pseudo_inv(float3x3 M)
{
float3 pi = cross(M[0], M[2]);
float3 pj = cross(M[1], M[2]);

  return float3x3(pi*1.0/dot(M[0], pi), pj*1.0/dot(M[1], pj), M[2]);
}
```

Changes of uniform parameters, like for instance the matrices of the control facet regions, underpin the performance of this implementation. Therefore it is convenient to arrange all faces with the same affecting regions into batches and submit them as a block. However, as Figure 6.2.a illustrates, this can yield an enormous number of partitions, in the same numerical scale as individual polygons.

Yet the number of affection regions that the shader can proccess simultaneously is much larger than those generally involved in the deformation of a single polygon/batch. Exploiting this fact we can actually process several batches at the same time, as long

as their affecting regions fall into the same set. The maximum number of regions that can be considered by a given instance of the vertex program is thereby limited by the number of input registers the architecture can provide. Table 6.1 gives a detailed account of such resources. Putting aside the storage needed for per-vertex data, transformation and lighting, shader model 2.0 allows `M_Ts` to hold a maximum of 28 3×3 affection region matrices, while subsequent versions increase this amount to at least 80.

However, the topology of a control mesh spanning the typical amount of markers (around 80 points) is made of roughly 100 control facets, yielding six times that amount of affection regions. Therefore in no case could all the geometry of the face be deformed with just one instance of the program.

In order to minimize the number of uniform parameter changes, and distribute them evenly along the deformation timeline, we need to cluster these groups into coherent partitions. For such purpose, we will adopt an optimization approach using a greedy clustering algorithm.

Consider R_i as the set of regions affecting the polygon p_i (constructed as the union of the corresponding sets for each of its vertices); we now define the macro-set R_i' as the union of the smallest R_js containing R_i, namely:

$$R_i' = \bigcup R_j \,\Big|\, R_i \subseteq R_j \wedge (\forall R_k \,|\, R_i \subseteq R_k, \ \sharp R_j \geq \sharp R_k) \tag{6.1}$$

After this first trivial aggregation, we can define a distance function between different R_i's as $d(R_l', R_m') = \sharp R_l' \cap R_m'$. Using this metric, the greedy algorithm proceeds by picking the set that is common to the smaller number of polygons, and aggregating it to its closest counterpart. If the cardinality of the resulting region set exceeds the limit imposed by the architecture, a new second term is selected. This proceeds until no further unification is possible, in which case the current set is added to the final solution.

The region set clusters produced by this procedure define as well a set of polygons in the deformed mesh that can be processed at the same time. The reduction in the number of batches achieved is illustrated in Figure 6.2. Additionally, the relatively similar size of these partitions allows for further parallelism in the hardware operation, by filling up the pipeline stalls that ocurr when loading data for a given cluster with the processing of previously submitted ones.

Furthermore, there is another aspect that can help significantly towards the simplification of this partitioning, and that is the maximum number of region contributions considered for each deformed vertex. In the shader code presented at the beginning of this section that amount was fixed to four, yet this amount can prove rather limiting. As discussed in Chapter 3, the smoothness of the deformation achieved with planar bones relies on an appropriate blending of the contribution of affecting regions, and discarding some of them can result in noticeable artifacts such as those displayed in Figure 6.3. For our current control mesh layout, 8 regions per vertex proves to be a more than adequate amount, and this can be easily implemented by duplicating the corresponding loop in the vertex program.

The resulting vertex program, also including transformation and lighting, takes 117 instructions. This means that it can run both vertex shader models 2.0 and 3.0. On a machine based on a Intel Pentium 4 1.9 Ghz., and equipped with a GeForce 4600 Ti
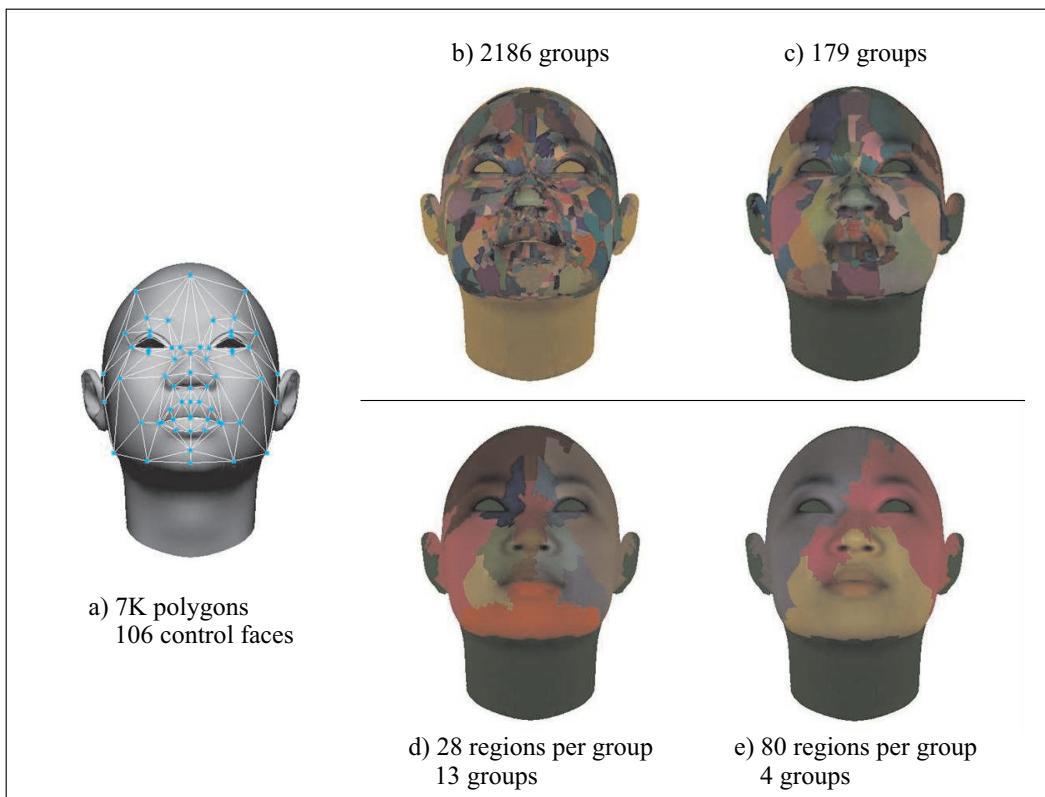
Fig. 6.2: Details of the clustering process for maximizing batch sizes: a) original model with its corresponding control mesh; b) polygons grouped according to their different combinations of affecting control facet regions (using just the 8 most characteristic); c) macro-groups after the initial aggregation step; d, e) final partitions matching the resources of Vertex shader models 2.0 and 3.0 (c.f. Table 6.1)

4 regions per vertex   8 regions per vertex
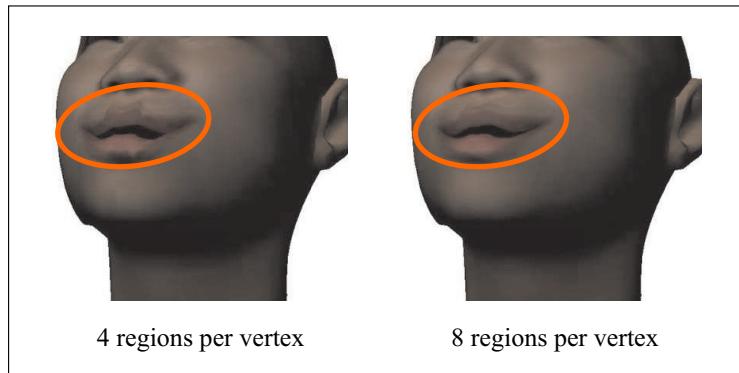
Fig. 6.3: Differences between the deformation performed considering a maximum of 4 and 8 regions per face.

(NV25), the performance achieved using a 640×640 window is 54 frames per second. On the same machine equipped with a GeForce FX 5200 (NV30), the same shader processes 80 regions at a time, yielding a framerate that is only limited by the frequency of vertical refresh (> 100 fps).

6.4 Hardware accelerated implementation of performance-based wrinkling and BIDS

Chapter 5 presented a framework in which BIDS were used to deform the facial model and then the corresponding wrinkling detail was added as a normal map, generated as result of the evaluation of its strain. Implementing this process in hardware involves both vertex and fragment shader programs, performing deformation and shading, and also computing strain data and communicating it between these two units, which does not class as any of their usual applications.

While this implementation results in a single pair of programs reproducing the required functionality altogether, this section will try to dissect them into the three most relevant parts: BIDS deformation, strain computation and evaluation of the normal map model.

6.4.1 BIDS as a vertex program

One of the simplifications that BIDS introduces with respect to the Planar Bones paradigm is the fact that every deformed vertex relates to a single point in the control structure, and therefore there is no need for an iteration evaluating the contribution of different control elements. However, computing the deformation caused by a single micro-patch of our conditioning scheme (see 3.5.2) is in itself much more computationally expensive, hence this simplification is not so effective.

In addition, the data per micro-patch is much larger than that required for a Planar Bones affection region (10 input registers instead of three), yet there are half as many

micro-patches per element of the control topology as there are affection regions. Ultimately, each macro-patch bears data for 19 registers (reduced from 30 by eliminating repeated elements across micro-patches), while the corresponding Planar Bones control facet takes up 18 (six regions), which balances out the data intake of both approaches.

In order to minimize changes of uniform parameters, we can process all three micro-patches of the same macro-patch at the same time, by making use of the predication ability of vertex processors in assigning the values of temporary registers:

```
// Micro-patch data
struct MU_PATCH
{
  float4 bA, bB, bC;
  float4 bD, bE;
  float4 bF;
};


// Per-vertex input data
struct VTX_IN
{
  float3 vtx_rsn; // paramereterised vertex position

  float3 tgt_rsn; // tangent frame induced by the reference texture coords,
  float3 bnm_rsn; // paramereterised over the control surface's tangent space at {r,s}
  float3 nrm_rsn;

  float mp_idx; // micro-patch index

  float2 tc; // ordinary texture coordinates

  // Cached from projection
  float4 aR, aS, aT; // sub-patch samples in the initial configuration
};


VTX_OUT main(
  // macropatch data
  uniform MU_PATCH mu_nets[3],
  uniform float4 b0,
  // reference coordinates of the markers
  uniform float2 rcs[3],
  // vertex data
          VTX_IN vin,
  // L&T data
  uniform float3 lgt_n, // light vector in world coordinates - prenormalized
  uniform float4 eye_pos, // camera position in world coordinates
  uniform float4x4 MVP) // model-view-projection matrix
{
float3 rst =
  float3(vin.vtx_rsn.x, vin.vtx_rsn.y, 1.0 - (vin.vtx_rsn.x + vin.vtx_rsn.y));
```

```
// Selecting the control net for the corresponding micro-patch
float4 b300, b210, b120, b030, b201, b111, b021, b102, b012;
float2 r100, r010, r001 = bez_1(rcs[0], rcs[1], rcs[2], (1.0/3.0).xxx);

  // the compiler translates the following comparisions as predicated writes to each
  // register holding a Bézier Point of the micro-patch (bijk)
  if(vin.mp_idx == 0) { b300 = mu_nets[0].bA; ... b030 = mu_nets[1].bA; ... }
  if(vin.mp_idx == 1) { b300 = mu_nets[1].bA; ... b030 = mu_nets[2].bA; ... }
  ...

// Computing subpatch samples for the deformed net
float4 bR, bS, bT;

  bR = bez_2(b300, b210, b120, b201, b111, b102, rst);
  bS = bez_2(b210, b120, b030, b111, b021, b012, rst);
  bT = bez_2(b201, b111, b021, b102, b012, b0, rst);

// Tangent frame of the surface at {r, s}, used for lighting
float3x3 T;

  T[0] = (3*(bR - bT)).xyz;
  T[1] = (3*(bS - bT)).xyz;
  T[2] = normalize(cross(T[0], T[1])); // normal

float3x3 T_i = pseudo_inv(float3x3(T[0], T[1], T[2]));

// Evaluating the deformed vertex position
float4 vtx;

  vtx.xyz = bez_1(bR, bS, bT, rst).xyz + vin.vtx_rsn.z*T[2]; vtx.w = 1;

// Strain computation
...
// Transformation and lighting
...
```

Functions `bez_1` and `bez_2` are simply implementations of de Casteljau's algorithm [Far95]; alternatively we could store the precomputed weights each of the Bézier points, yet this would only benefit the quadratic case, causing no significant improvements.

```
float4 bez_2(
  float4 b200, float4 b110, float4 b020,
  float4 b101, float4 b011,
  float4 b002,
  float3 rst)
{
  return bez_1(
    bez_1(b200, b110, b101, rst), bez_1(b110, b020, b011, rst),
    bez_1(b101, b011, b002, rst),
    rst);
}
```

```
float4 bez_1(
   float4 b100, float4 b010,
   float4 b001,
   float3 rst)
{
   return rst.x*b100 + rst.y*b010 + rst.z*b001;
}
```

Similarly to the Planar Bones implementation, this BIDS vertex program could manage several macro-patches at the same time, enlarging the batches it processes; however the increase in selection code would push the program length over the 256 instructions boundary, rendering it not applicable for the NV30 architecture this work has targeted.

In addition, the primitive-handling restrictions that the pipelined programming model imposes require that all vertices in a given batch relate to the same macro-patch. For that purpose the deformed geometry has to be split so that polygons do not span vertices projected over different patches. Such splitting is performed along the boundaries between the normal coverages of contiguous patches (see Section 3.5.2), locating every edge of the model going across them and inserting a new vertex at the intersection point. Then re-meshing is performed through the same 1 to 3 scheme described in Section 3.5.6.

Finding the intersection point $\mathbf{q}$ between a given polygon edge e and the boundary of contiguous $C(\mathbf{s}_i)$ is equivalent to intersecting such a segment with the osculating plane spanned by the normal and tangent vector fields applied onto the boundary curve:
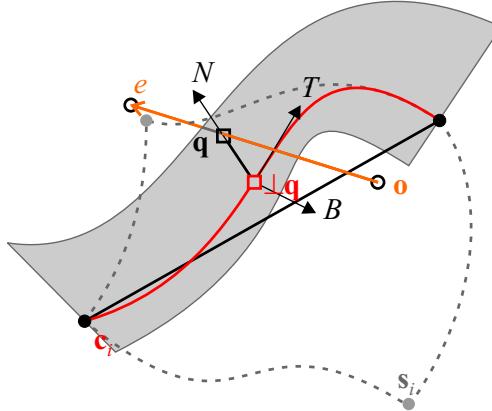
$$\mathbf{q} \equiv \begin{cases} \mathbf{q}_A = \frac{T_i \, (\mathbf{c}_i - \mathbf{o})^T}{T_i \, e^T} \, e + \mathbf{o} & \text{if } e \, \tilde{\parallel} \, T_i \\[2em] \mathbf{q}_B = \frac{B_i \, (\mathbf{c}_i - \mathbf{o})^T}{B_i \, e^T} \, e + \mathbf{o} & \text{if } e \, \tilde{\perp} \, T_i \end{cases} \qquad (6.2)$$

Taking $\mathbf{c}_i(t)$ as the parametric expression of the boundary curve between the two patches, the tangent vector field T_i can be computed as $\mathbf{c}_{,t} \circ \mathbf{c}^{-1}$, and similarly the pseudo-binormal field B_i responds to the cross product $T_i \wedge U_i$. Using these definitions, we can find $u_{\mathbf{o}} = \mathbf{c}^{-1}(\mathbf{o})$ as the root of the following equation:

$$(\mathbf{q}_{,t} - T_i \circ \mathbf{c}_i) \cdot (\mathbf{q} - \mathbf{c}_i) = 0 \qquad (6.3)$$

Segmented face (left) and detail of the polygonal topology before and after
partitioning (above).
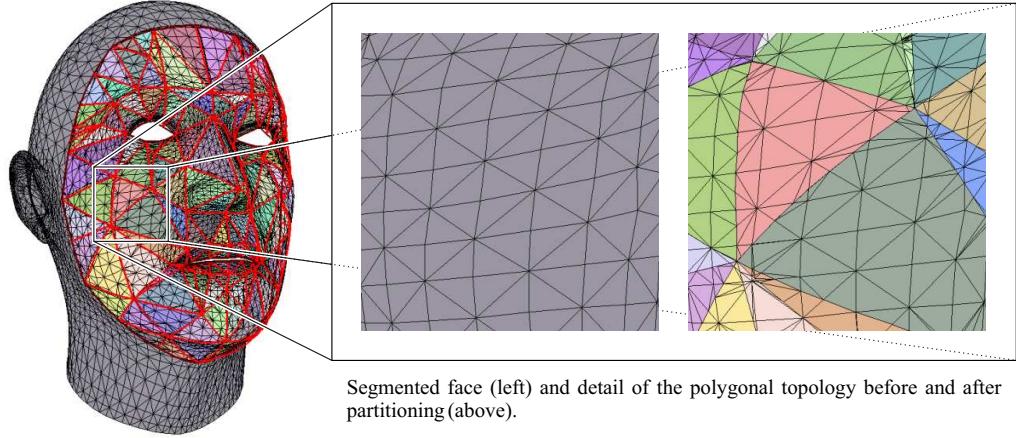
Fig. 6.4: Detials of the mesh preprocessing enabling a hardware-accelerated implemen-
      tation of BIDS.

where

$$
\mathbf{q}_{,T_i} = \begin{cases} \dfrac{((U_i \circ \mathbf{c})_{,t} \cdot (\mathbf{c}-\mathbf{o}))(e \cdot U_i \circ \mathbf{c}) - (e \cdot (U_i \circ \mathbf{c})_{,t})((\mathbf{c}-\mathbf{o}) \cdot U_i \circ \mathbf{c})}{(e \cdot U_i \circ \mathbf{c})^2} & \text{if } v \overset{\sim}{\parallel} U_i (\perp \mathbf{q}) \\[2em] \dfrac{((B_i \circ \mathbf{c})_{,t} \cdot (\mathbf{c}-\mathbf{o}))(e \cdot B_i \circ \mathbf{c}) - (e \cdot (B_i \circ \mathbf{c})_{,t})((\mathbf{c}-\mathbf{o}) \cdot B_i \circ \mathbf{c})}{(e \cdot B_i \circ \mathbf{c})^2} & \text{if } v \overset{\sim}{\perp} U_i (\perp \mathbf{q}) \end{cases}
$$

For the sake of brevity, the full derivation of the solution for Equation 6.3 is provided
in Appendix B. Applying the partitioning to a facial model consisting of 11k vertices,
deformed by a control surface with the same reference topology used in Chapter 5, yields
the segmentation portrayed in Figure 6.4. This process adds 1253 new vertices ($+11\%$),
which is a fair trade-off for a hardware-accelerated implementation. Also, the adaptive
subdivision scheme presented in Section 3.5.6 is not possible with the inflexible topology
handling imposed by the programming model of the hardware, yet the performance gains
allow for the facial model to be pre-subdivided, yielding results of comparable quality.

6.4.2   Per-vertex strain analysis

Section 5.3.3 demonstrated how the strain tensor on the tangent plane of the polygonal
mesh can be approximated by an equivalent computation on its projection onto the
control surface. This can be performed at little additional cost in the same vertex
program that carries out BIDS, by evaluating Equation 5.30 with the same sub-patch
samples used to deform the vertex. Naming sub-patches $\{\mathbf{s}^r, \mathbf{s}^s, \mathbf{s}^t\}$ as $\{\texttt{aR}, \texttt{aS}, \texttt{aT}\}$, and
similarly $\{(\mathbf{s}')^r, (\mathbf{s}')^s, (\mathbf{s}')^t\}$ as $\{\texttt{bR}, \texttt{bS}, \texttt{bT}\}$, the strain computation functionality can be
provided by appending the following code fragment to the implementation of BIDS in
Section 6.4.1:

```
...

// Computing compression tensor components
float2x2 Juv; // Jacobian of ref. texture coordinates expressed in parametric space

  Juv[0] = r100 - r001;
  Juv[1] = r010 - r001;

float2x3 Jdisp; // Jacobian of the displacement function expressed in parametric space

  Jdisp[0] = (float3)((bR - vin.aR) - (bT - vin.aT));
  Jdisp[1] = (float3)((bS - vin.aS) - (bT - vin.aT));

// Strain tensor expressed in the reference texture space
float2x2 S = mul(mul(inv(Juv), (float2x2)mul(Jdisp, T_i)), Juv);

  vout.cmp_tb = eigen_sym(float3(S[0][0], S[1][1], 0.5*(S[0][1] + S[1][0])));

...
```

`eigen_sym` is a function performing eigenvalue decomposition for 2×2 symmetric matrices, and clamping the characteristic components to the negative interval of the real line in order to represent just compression:

```
// Eigen-decomposition for 2x2 symmetric matrices
#define S_EPSILON 0.0001

static float3 eigen_sym(float3 s)
{
float2x2 EV; float l0, l1;

  if(s[2]*s[2] >= S_EPSILON)
  {
  float F0 = s[0] + s[1], F1 = sqrt((s[0] - s[1])*(s[0] - s[1]) + 4.0*s[2]*s[2]);

    l0 = 0.5*(F0 + F1); l1 = 0.5*(F0 - F1);
    EV = float2x2(1.0, (l0 - s[0])/s[2], 1.0, (l1 - s[0])/s[2]);
  }
  else
  {
    l0 = s[0]; l1 = s[1];
    EV = float2x2(1.0, 0.0, 0.0, 1.0);
  }

float2x2 V = float2x2(min(-l0, 0.0), 0.0, 0.0, min(-l1, 0.0));
float2x2 C = mul(mul(inv(EV), V), EV);

  return float3(C[0][0], C[1][1], C[0][1]);
}
```

Samples `aR`, `aS` and `aT` are cached per vertex, as they can be computed at the time of the projection; however, they could actually be re-evaluated along with the deformed sub-patches, just by inputting the initial control nets to the shader as uniform parameters. This would save a considerable amount of redundant information, but would also increase the length of the program. Since this is a rather important constraint, we are forced to take this option. Alternatively, the number of instructions could be reduced by delegating the eigen-decomposition onto the fragment program, but that would create a bottleneck in the rasterization that would not be acceptable.

6.4.3 Evaluating and applying the normal map

We need to produce compression data in the vertex shading stage in order to be able to evaluate the wrinkling model described in Section 5.3 while rendering skin fragments. However this is not sufficient, as the implementation of normal maps also demands the lighting data to be expressed in the tangent space of the mesh, in order for the per-pixel operations to be reasonably efficient (see [nvi02]). Therefore the lighting and transformation steps in the vertex program have the following form:

```
...

  vout.vtx = mul(MVP, vtx);

// texture & reference coordinates

  vout.tc = vin.tc;
  vout.rc = bez_1(r100, r010, r001, rst);

// Evaluating the deformed tangent space basis (aS induced by rcs) using inverse-
// transpose
float3x3 TBN;

  TBN = mul(float3x3(vin.tgt_rsn, vin.bnm_rsn, vin.nrm_rsn), T_i);
  // optionally, a re-orthonormalization could be placed here

float3 eye_n = normalize((float3)(vtx - eye_pos));

  vout.lgt_tbn = mul(lgt_n, TBN);
  vout.hlf_tbn = mul(normalize(eye_n + lgt_n), TBN);

  return vout;
}
```

With light and half vectors expressed in the same coordinate frame as the distorted surface normal given by the normal map (bump normal), we can compute the lighting using Phong's equation for the Bidirectional Reflection Distribution Function (BRDF, see [Bli97]):

$$I = k_s (L \cdot N) + k_d (L \cdot H)^p \qquad (6.4)$$

where I stands for the perceived intensity at a given point, and L and H are light and

half vectors, as computed before. Coefficients k_s and k_d stand for the specular and diffuse reflective properties of the illuminated material, and p is the power of shininess that determines the concentration of specular reflection. As the skin reflective properties are predefined and assumed constant for all its extents, we can pre-compute the terms of this equation in a lookup table indexed by the two dot products, and then store it as a texture to save per-pixel operations; in the code this texture is called `illumMap`, and dereferenced as a 2D-array using the function `tex2D`.

Other textures that will be required for shading the skin are those containing the neutral normal map (`n0Map`) and diffuse colour (`diffMap`). As the degree of the polynomial approximation used by the wrinkling model is trilinear, only one coefficient will be needed per strain component to compute n_i as in Equation 5.31. Therefore, two texture maps packing three coefficients each (`ctMap` and `cbMap`) will suffice. However, the colour components encoding textures range in the interval $[0..1]$, while wrinkling coefficients do so along the whole real line. While there is no actual bound for such coefficients, we can clamp their range to $[-8..8]$ without a significant loss of quality, as the precision lost in the scaling corresponds to levels of strain that do not cause perceptible wrinkling. Alternatively we could use floating point textures, but this would require four times as much storage space.

The final fragment program that results from implementing the evaluation of the wrinkling approximation along with the normal map shading itself is the following:

```
// Scaling of the coefficient maps (their data is not unitary)
#define SCL_COEFS 16.0


// Per-vertex data
struct VTX_IN // VTX_OUT in the vertex program
{
  float4 vtx : POSITION; // deformed vertex in normalized projective space

  float2 tc : TEXCOORD0; // ordinary texture coordinates
  float2 rc : TEXCOORD1; // reference texture coordinates

  float3 cmp_tb : COLOR; // components of the compression tensor: [e_uu, e_vv, g_uv]
  float3 lgt_tbn : TEXCOORD2; // light vector in the tangent space induced by the rcs
  float3 hlf_tbn : TEXCOORD3; // half-vector in the tangent space induced by the rcs
};


VTX_OUT main(
               VTX_IN vin, // per-vertex data
  const uniform sampler2D diffMap, // ordinary diffuse texture (skin colour)
  const uniform sampler2D n0Map, // neutral normal map
  const uniform sampler2D ctMap, // coefficients [alpha^u_1, beta^u_1, gamma^u_1]
  const uniform sampler2D cbMap, // coefficients [alpha^v_1, beta^v_1, gamma^v_1]
  const uniform sampler2D illumMap) // illumination look-up map
{
float3 nrm = tex2D(n0Map, vin.rc).xyz;
```

```
// Normal components in the tangent plane
float nt = dot(SCL_COEFS*(tex2D(ctMap, vin.rc).xyz - float3(0.5, 0.5, 0.5)),
               vin.cmp_tb) + nrm.x;
float nb = dot(SCL_COEFS*(tex2D(cbMap, vin.rc).xyz - float3(0.5, 0.5, 0.5)),
               vin.cmp_tb) + nrm.y;

// Reconstructed bump normal in tangent space
float3 bumpNrm_tbn =
  normalize(float3(nt, nb, 1.0)); // this could be replaced by a N-R pseudo-norm

// Indices of the illumination map
float diff = dot(vin.lgt_tbn, bumpNrm_tbn);
float spec = dot(vin.hlf_tbn, bumpNrm_tbn);

// Computing illumination
float4 illum = tex2D(illumMap, float2(diff, spec));
float shadow = saturate(4.0*vin.hlf_tbn.z); // additional shadowing term

VTX_OUT vout;

  // modulating the color texture
  vout.colour.rgb = shadow*(illum.xyz*tex2D(diffMap, vin.tc).rgb + illum.www);
  vout.colour.a = 1.0;

  return vout;
}
```

6.4.4   Application

With the methods proposed, the test program running in a 640×640 window achieves a performance of 80 fps on a system equipped with an AMD Athlon XP 2600+ processor and a GeForce FX 5200 graphics card. This almost doubles the 43 fps achieved by the equivalent software implementation (that uses the same fragment program, but runs BIDS on the main processor).

The main bottleneck continues to be changes in uniform parameters. While there are in effect sufficient input registers in the NV30 vertex programmable unit to handle up to 11 macro-patches, the selection logic will cause the program to shift over the maximum length for that chipset, as it is already rather long (242 instructions). Such a limitation is not present in subsequent chipsets, therefore this would be a sensible extension to easily improve overall performance.

Another alternative to handle large vertex programs is concatenating their results by storing intermediate values (e.g. BIDS-deformed vertices) in a vertex buffer. However this will imply not only several passes, but also separating BIDS deformation from strain computation, and incurring lots of redundant evaluations that will jeopardize the convenience of approximating the strain tensor by means of the control surface.

6.5   Summary and conclussions

The preceding sections have described the task of porting the synthetic part of most of the algorithms developed in this work to a dedicated hardware architecture. While in some cases this has proven relatively simple, as these methods were conceived for performance, combining multiple elements together as in Section 6.4 posed a significantly more daunting problem. Coping with limited hardware resources has led to many special considerations resulting in an extensive use of their capabilities, as the length of the vertex program implementing BIDS and performance-based wrinkles ascertains. Some of these constraints will be relieved with the evolution of graphics chipsets, however the methods adopted to overcome them will still be qualitatively valid.

Another aspect that has received special attention in this Chapter is the reduction of geometry batches, in order to prevent stalls in the graphics pipeline when feeding new control data. Some of the problems faced in this particular endeavour were related to the limitations of the whole graphics architecture, and the way it handles both topology and geometry. The new generation of accelerators, in particular the ATI R500 chipset, aims to enhance the programming model and overcome this rigidity. However, at the time of producing this thesis such hardware is still at the specification stage, and there is no graphics API that can support this functionality.

7. CONCLUSIONS AND FUTURE WORK

7.1   Construction of a Facial Animation system

This thesis has presented a series of novel techniques in the fields of animation retargeting (Section 4.3.2), geometric deformation (Sections 3.4, 3.5 and 5.2) and skin shading (Section 5.3) that contribute towards a complete Facial Animation framework, driving the animation of facial skin with optically-based Motion Capture (MoCap). The design of such a framework, illustrated in Figure 7.1, has been driven by a series of specific properties which it aims to attain. Namely, these are:

1. *Perceptual realism*,

2. *generality and reusability*,

3. *a high degree of automation* and

4. *real-time performance*.

The following sections will describe how specific stages of our framework approach these properties, and to which extent they satisfy them.

7.1.1   Perceptual realism

Optically-based facial MoCap solutions, such as the commercial system Vicon 8 [Vic] whose input we have used for the examples given in this thesis, provide a realistic description of the movement of specific features of the face. However, there are aspects of facial motion, such as expressive wrinkling, that cannot be captured with the resolution provided by these systems. This is the reason why we divide the operation of the framework into two tiers, tackling the *large scale deformation* of facial skin and the synthesis of *fine tissue detail* separately.

On the side of large scale deformation, we need to translate the motion information that MoCap provides for specific marker points to the rest of the face (stage 3 in Figure 7.1). Achieving this in a realistic manner is a challenging task for which Planar Bones (Section 3.4) and BIDS (Section 3.5) have been specially conceived. These deformation techniques overcome the limitations of their interpolatory counterparts (see Section 3.3.3) by propagating motion in a way that is coherent with the connectivity of facial tissue, preserves its overall smoothness, and guarantees locality of the deformation according to the constraints imposed by the marker layout.

Representing the fine-scale behaviour of facial tissue using a polygonal mesh turns out to be largely inefficient (see Section 5.2.4). Instead, we reproduce this phenomenon with
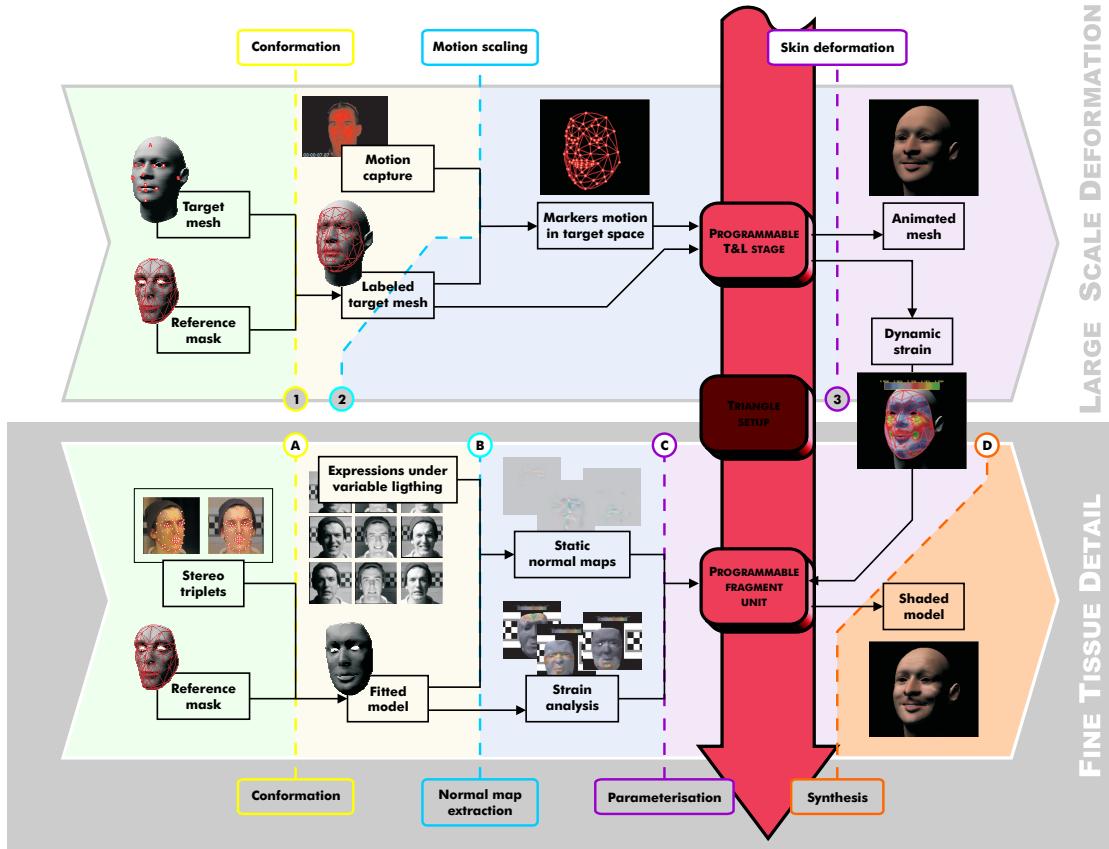
Fig. 7.1: Diagram of the Facial Animation framework integrating the techniques proposed in this thesis. The vertical arrow illustrates how the two tiers of the system accommodate their synthetic stages into the architecture of a hardware-accelerated graphics pipeline.

normal maps [Bli78] applied onto the coarsely animated mesh resulting from the previous tier of the framework. For this purpose, we construct a performance-based model of skin wrinkling (Section 5.3.4 - stage C in the diagram) that relates the magnitude and direction of local strains to the perceived effect. Retrieving the data necessary to build such a model requires specialised capture techniques, that we described in Sections 5.3.1 and 5.3.2 (corresponding to stages A and B). This approach improves significantly the perception of certain expressions, especially those denoting emotion, given the large perceptual importance of facial wrinkles and tissue buckling in such situations.

7.1.2 Generality and reusability

The retargeting methods described in Section 4.3 provide the ability to express MoCap data retrieved from different individuals under a common framework. Together with the Rigid Body Transformation estimation and removal process described in Section

4.2.3, this system allows defining a canonical model of facial motion, a tool of significant importance for the analysis of facial MoCap data.

In addition, the reverse process means that a given MoCap stream captured from a particular subject can be adapted onto any different facial physiognomy labelled with a compatible marker layout, in a way that is consistent with the anatomic differences observed between the original performer and the target model (stage 2 in Figure 7.1). This maximises the reusability of MoCap data, enabling us to drive the animation of different synthetic characters with a unique set of streams, with the consequent reduction in memory requirements that this implies.

7.1.3   High degree of automation

As described in Section 3.3.2, many of the Facial Animation approaches used in the Computer Graphics industry involve a considerable amount of human manipulation, resulting in high production costs. One of the characteristics of the system developed in this thesis is that such user interaction in the animation process is kept to a minimum.

Both Planar Bones and BIDS deformation techniques intrinsically provide a control of locality that defeats the need for model regionalization present in other motion interpolation approaches (c.f. Section 3.3.3); this allows introducing new models with significant ease. Furthermore, the model fitting approach described in Section 4.3.1 (stage 1 in the diagram) permits the labelling of a predefined model to be transferred onto any new physiognomy, requiring just the input of a minimal set of marker points (see Figure 4.6) for the complete model setup.

Finally, the only remaining per-model operation in the application of Planar Bones is the fitting of a discontinuity map, that in some occasions may result specially cumbersome or even impossible (e.g. whenever the lips projected onto texture space overlap). Opportunely, the better approximation of the skin's geometry achieved by the BIDS control surface allows for a fully automatic approach to be implemented.

7.1.4   Real-time performance

One of the main aspects driving the construction of this system is the omnipresent need for a real-time implementation; while there are several stages concerning observation analysis and model setup that bear significant computation times, all synthetic processes are efficient enough to be implemented at interactive frame-rates. In particular, both deformation algorithms proposed have linear computational costs with respect to the complexity of the model they are applied to. Similarly, the retargeting process can also be done in real-time, after caching the results of an initial cubic computation (with respect to the number of marker points).

Furthermore, BIDS and Planar Bones have been implemented on dedicated graphics architectures (c.f. Sections 6.3 and 6.4), meaning that dense facial models featuring more than 10^4 polygons can be used. These implementations, in combination with a hardware accelerated version of the wrinkling model (see Section 6.4 - stage D), produce results of near photorealistic quality for interactive contexts, showcased in the accompanying animations.

7.2 Thoughts on evaluation frameworks

As with many of the results produced in Computer Graphics, the evaluation of the output of this system is largely subjective, and the task of constructing a framework to verify its adequacy is undermined by the many other limitations of computer-generated representations that would influence any experimental subjects. Audio-visual studies such as those by Cohen and Massaro in [CMC02] are not applicable as the Motion Capture input mimics identically the vocal tract, while emotional measurement approaches like the work by Pandzic et al. in [PAY02] prove highly inconclusive.

In practice, parts of the system such as the synthesis of retargeted expressions and their corresponding wrinkling can be contrasted with the original poses of the captured performer, as long as both facial physiognomies bear a certain similarity. This is demonstrated in Figure 5.12. More objectively, the error in reproducing the captured fine scale behaviour can be measured as in the graphs in Section 5.3.4, yet these metrics are also subject to observation flaws (e.g. misalignment of the crease lines in the data reconstructed from different viewpoints).

As for the evaluation of animation, the supporting video `ea_MoCap.mpg` showcases synthetic motion against the captured source. Should a more objective process be necessary, a framework could be constructed in which speech or emotion portrayed by an individual A is retargeted onto B, and compared with the direct result of this last subject uttering the same words or posing the same gesture. However, for this to be possible it would be necessary to be able to capture the actual physiognomy of B, and the hardware necessary for such purpose (e.g. non-invasive laser scanners [1] [2]) is in short supply.

7.3 Scope of application and potential further developments

As commented in Section 7.1.4 the real-time nature of the methods embedded in this system shape it as a useful tool in the construction of interactive content, such as Human-Computer Interaction (HCI) mechanisms, Computer Games or Virtual Reality simulations. However, the significant degree of automation achieved within this framework makes it also an interesting contribution for the creation of off-line content, such as films or TV productions. Undeniably, some of the particular aspects of this work (e.g. rendering) have yet to reach the level of quality generally achieved in these non real-time contexts, however the approaches to MoCap retargeting and tissue deformation will scale up graciously with a greater density of marker points.

The major limitation constraining the application of the current framework in a fully interactive context is the fact that only captured speech and emotion can be reproduced. The generation of new speech data has been tackled with a limited domain approach combining existing fragments (see [ESM04b] and [ESM04a]), yet the synthesis of completely new data remains an unsolved problem. However, as outlined by Section 7.1.2, the retargeting framework contributed by this thesis provides a unique commodity for the analysis of facial motion independently of particular physiognomies and rigid move-

---

[1] `http://www.nvision3d.com/`
[2] `http://www.viewpoint.com/`

ment, allowing the application of statistical models such as those frequently used in conventional speech synthesis (see for instance [Law06]). For this purpose, an extensive body of captured data would be necessary in order to be able to study phoneme utterances under a wide range of conditions, but combined with proper labelling it would also allow to isolate cross-individual traces of emphasis and emotion, thus producing a powerful synthetic tool.

Additionally, the motion of other facial elements contributing to facial expression could be reproduced to a larger extent. Currently eye gaze is not considered, and doing so would require observation methods of its own, as marker points cannot be placed on pupils (see [HFE00, JZ02]). Also, tongue movement is just inferred from the markers along the mouth contour; properly capturing this feature is not feasible from an optical point of view, since most of its extent remain hidden in the buccal cavity. A successful approach could be a semi-automatic process, combining tongue posing driven by speech analysis with manually-specified motion for specific gestures.

Manual intervention has been avoided as much as possible in the design of this framework, as Section 7.1.3 discusses, however in some circumstances allowing a further degree of interaction can prove useful, as was demonstrated in Section 4.3.3 with cartoon-like expressions. Endowing the system with the flexibility necessary to introduce more artistically produced input is another interesting aspect to bear in mind for future iterations.

APPENDICES

A. PROJECTION OF A VERTEX OVER A BÉZIER PATCH

A.1   Posing the problem

As outlined in Section 3.5, the projection of a given vertex $\mathbf{q} \in \mathcal{M}$ onto the ith patch of the control surface $\mathcal{S}$ is ruled by the following minimizing expression:
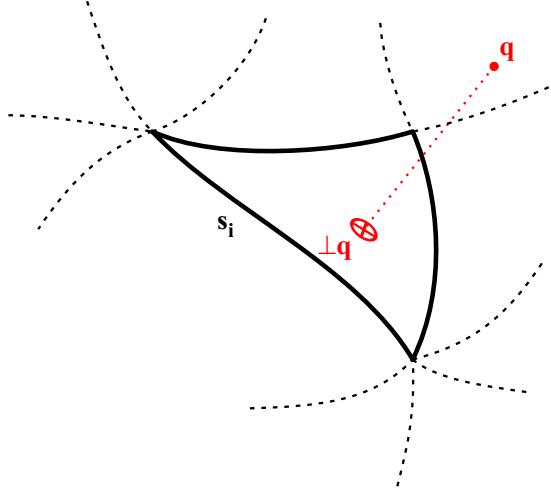
$$\{u, v\} = \mathbf{s}_i^{-1} (\perp \mathbf{q}) = \arg\min \left(F(u, v) \equiv (\mathbf{s}_i(u, v) - \mathbf{q})^2 \right) \tag{A.1}$$

where the parametric expression of the patch is given as in Equation A.2.

$$\mathbf{s}_i(u, v) = \mathbf{S}_i(u, v, 1 - (u + v)) = \sum_{p,q,r}^{n} \mathbf{s}_{pqr}^{j} \ B_{pqr}^{n}(u, v, 1 - (u + v)) \tag{A.2}$$

Local minima of Equation A.1 would be reached whenever both components of ∇F are identically 0. As their derivation Equation A.3 illustrates, this corresponds to points where the distance vector is orthogonal to the surface's tangent plane, thus parallel to is normal field U_i. Assembling these conditions in a single functional G:

$$\left. \begin{aligned} \nabla F_u = 2 \, \tfrac{d\mathbf{s}_i}{du} \, (\mathbf{s}_i - \mathbf{q})^T = 0 \\ \nabla F_v = 2 \, \tfrac{d\mathbf{s}_i}{dv} \, (\mathbf{s}_i - \mathbf{q})^T = 0 \end{aligned} \right\} \Rightarrow G(u, v) \equiv \left(\frac{d\mathbf{s}_i}{du} \, (\mathbf{s}_i - \mathbf{q})^T \right)^2 + \left(\frac{d\mathbf{s}_i}{dv} \, (\mathbf{s}_i - \mathbf{q})^T \right)^2 = 0 \tag{A.3}$$

A.2   Deriving a numerical solution

Even for low-degree cases such as the cubic expressions used in Section 3.5, G does not have an analytic solution (e.g. yields an equation of degree 10), requiring the application of a numerical scheme. Given the regularity of Bézier patches, the gradient descent approach [PTF92, GW04] is most suitable; taking a rough initial approximation $[u_0, v_0]$, we proceed iteravely as by:

$$[u_{j+1}, v_{j+1}] = [u_j, v_j] - \frac{G(u_j, v_j)}{\nabla G(u_j, v_j) \ (\nabla G(u_j, v_j))^T} \ \nabla G(u_j, v_j) \qquad (A.4)$$

until $G(u_j, v_j)$ is close enough to 0. The gradient ∇G has its two components given by Equations A.6 and A.6.

$$\nabla G_u = \frac{\partial G}{\partial u} \quad = \quad 2 \left(\frac{d^2 \mathbf{s}_i}{du^2} (\mathbf{s}_i - \mathbf{q})^T + \frac{d\mathbf{s}_i}{du} \frac{d\mathbf{s}_i}{du}^T \right) \left(\frac{d\mathbf{s}_i}{du} (\mathbf{s}_i - \mathbf{q})^T \right) +$$
$$2 \left(\frac{d^2 \mathbf{s}_i}{dudv} (\mathbf{s}_i - \mathbf{q})^T + \frac{d\mathbf{s}_i}{dv} \frac{d\mathbf{s}_i}{du}^T \right) \left(\frac{d\mathbf{s}_i}{dv} (\mathbf{s}_i - \mathbf{q})^T \right) \qquad (A.5)$$

$$\nabla G_v = \frac{\partial G}{\partial v} \quad = \quad 2 \left(\frac{d^2 \mathbf{s}_i}{dv^2} (\mathbf{s}_i - \mathbf{q})^T + \frac{d\mathbf{s}_i}{dv} \frac{d\mathbf{s}_i}{dv}^T \right) \left(\frac{d\mathbf{s}_i}{dv} (\mathbf{s}_i - \mathbf{q})^T \right) +$$
$$2 \left(\frac{d^2 \mathbf{s}_i}{dvdu} (\mathbf{s}_i - \mathbf{q})^T + \frac{d\mathbf{s}_i}{du} \frac{d\mathbf{s}_i}{dv}^T \right) \left(\frac{d\mathbf{s}_i}{du} (\mathbf{s}_i - \mathbf{q})^T \right) \qquad (A.6)$$

The derivatives of the bi-parametric expression of the patch relate to its barycentric formulation $\mathbf{S}_i$ through following identities:

$$\frac{d\mathbf{s}_i}{du} = \frac{\partial \mathbf{S}_i}{\partial u} - \frac{\partial \mathbf{S}_i}{\partial w}; \ \frac{d\mathbf{s}_i}{du} = \frac{\partial \mathbf{S}_i}{\partial u} - \frac{\partial \mathbf{S}_i}{\partial w}$$
$$\frac{d^2 \mathbf{s}_i}{du^2} = \frac{\partial^2 \mathbf{S}_i}{\partial u^2} + \frac{\partial^2 \mathbf{S}_i}{\partial w^2} - 2 \frac{\partial^2 \mathbf{S}_i}{\partial u \partial w}; \ \frac{d^2 \mathbf{s}_i}{dv^2} = \frac{\partial^2 \mathbf{S}_i}{\partial v^2} + \frac{\partial^2 \mathbf{S}_i}{\partial w^2} - 2 \frac{\partial^2 \mathbf{S}_i}{\partial v \partial w}$$
$$\frac{d^2 \mathbf{s}_i}{dudv} = \frac{\partial^2 \mathbf{S}_i}{\partial u \partial v} + \frac{\partial^2 \mathbf{S}_i}{\partial w^2} - \frac{\partial^2 \mathbf{S}_i}{\partial u \partial w} - \frac{\partial^2 \mathbf{S}_i}{\partial v \partial w} \qquad (A.7)$$

and, as [Far95] points out, the partial derivative terms in A.7 can be easily computed as recursive differences of the intermediate points in the *De Casteljau*'s algorithm used to evaluate $\mathbf{S}_i$.

For the cubic case, even a rough projection of the vertex on the supporting domain triangle of $\mathbf{s}_i$ would be sufficient as a start-up point to guarantee convergence. Should the degree of the patch increase, paraboliodal descent methods in the style of Muller's [GW04] may prove more accurate.

As for the numerical stability, the gradient gain quotient in Equation A.4 would only approach 0 whenever the neighbourhood of $\mathbf{s}_i(u, v)$ is locally equidistant to $\mathbf{q}$, and that is only verified in points of $\mathscr{E}(\mathbf{s}_i)$, which are explicitly omitted from the domain of the projection (c.f. Section 3.5.2).

B. INTERSECTION OF A SEGMENT WITH THE OSCULATING BOUNDARY PLANE OF A BÉZIER PATCH

B.1 Posing the problem

Without loss of generality, we can assume that the intersection of a segment with the osculating boundary plane of a Bézier patch happens along the wu-boundary of the patch domain; taking the curve $\mathbf{c}_i$ as the restriction of its bi-parametric expression $\mathbf{s}_i$ with $v = 0$, the u coordinate of the projected intersection point $\perp \mathbf{q}$ can be found as:
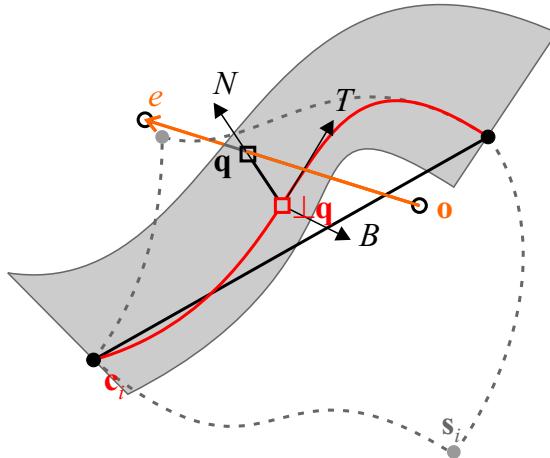
$$u = \mathbf{c}_i^{-1}(\perp \mathbf{q}) = \arg\min(F(u) \equiv (\mathbf{q} - \mathbf{c}_i)\,(\mathbf{q} - \mathbf{c}_i)^T) \tag{B.1}$$

Numerical stability requires that we consider two possible hypothesis for $\mathbf{q}$, depending on the orientation of e with respect to the bounding plane. Taking T_i as the tangent vector field of the curve, N_i as the restriction of the surface normal $U \circ \mathbf{s}_i$, and B_i as the pseudo-binormal field induced by this last, $\mathbf{q}$ is enunciated as:

$$\mathbf{q} \equiv \begin{cases} \mathbf{q}_A = \dfrac{T_i\,(\mathbf{c}_i - \mathbf{o})^T}{T_i\,e^T}\,e + \mathbf{o} & \text{if } e \;\tilde{\parallel}\; T_i \\[2em] \mathbf{q}_B = \dfrac{B_i\,(\mathbf{c}_i - \mathbf{o})^T}{B_i\,e^T}\,e + \mathbf{o} & \text{if } e \;\tilde{\perp}\; T_i \end{cases} \tag{B.2}$$

where T_i is computed as $\frac{\partial \mathbf{c}_i}{du}$ and $B_i = T_i \wedge N_i$.

Taking derivatives, the local minima of F would be found at the roots of the function G, where:

$$G \equiv \frac{dF}{du} = 2 \left(\frac{d\mathbf{q}}{du} - T_i \right) (\mathbf{q} - \mathbf{c}_i)^T = 0 \qquad (B.3)$$

Equation B.3 admits the higher-dimensionality interpretation of the distance vector $d \equiv \mathbf{q} - \perp \mathbf{q}$ being orthogonal to its gradient, which its also consistent with the minimization process it represents. As for the computation of $\frac{d\mathbf{q}}{du}$, the two hypothese described in Equation B.2 can be applied again, yielding the following expressions:

$$\frac{d\mathbf{q}_A}{du} = \frac{\left(\frac{dT_i}{du} (\mathbf{c}_i - \mathbf{o})^T + T_i \, T_i^T \right) \left(T_i \, e^T \right) - \left(T_i \, (\mathbf{c}_i - \mathbf{o})^T \right) \left(\frac{dT_i}{du} \, e^T \right)}{\left(T_i \, e^T \right)^2} \, e \qquad (B.4)$$

$$\frac{d\mathbf{q}_B}{du} = \frac{\left(\frac{dB_i}{du} (\mathbf{c}_i - \mathbf{o})^T \right) \left(B_i \, e^T \right) - \left(B_i \, (\mathbf{c}_i - \mathbf{o})^T \right) \left(\frac{dB_i}{du} \, e^T \right)}{\left(B_i \, e^T \right)^2} \, e \qquad (B.5)$$

Finally, derivatives of the tangent, normal and pseudo-binormal vector fields defined along the boundary curve can be evaluated as follows:

$$\begin{aligned}
\frac{dT_i}{du} &= \left. \frac{d^2 \mathbf{s}_i}{du^2} \right|_{v=0} \\
\frac{dN_i}{du} &= \frac{dT_i}{du} \wedge \left. \frac{d\mathbf{s}_i}{dv} \right|_{v=0} + T_i \wedge \left. \frac{d^2 \mathbf{s}_i}{du\,dv} \right|_{v=0} \\
\frac{dB_i}{du} &= \frac{dT_i}{du} \wedge N_i + T_i \wedge \frac{dN_i}{du} \qquad (B.6)
\end{aligned}$$

and the remaining terms are computed according to the guidelines given in Equation A.7.

B.2 Deriving a numerical solution

Substituting Equations B.4 and B.5 back into the expression of G yields an identity that, for the cubic case we are concerned with, cannot be put in closed form (since it bears a polynomial order of 6th degree or higher). Once more, a numeric solution is necessary; taking a Newton-Raphson approach [PTF92, GW04], an initial approximate u_0 of the root can be improved by iterating the following process:

$$u_{j+1} = u_j - \frac{G(u_j)}{\frac{dG}{du}(u_j)} \qquad (B.7)$$

until $G(u_j)$ falls below a certain treshold. The computation of the derivative of the objective function can be expressed as:

$$\frac{dG}{du} = \left(\frac{d^2 \mathbf{q}}{du^2} - \frac{dT_i}{du} \right) (\mathbf{q} - \mathbf{c}_i)^T + \left(\frac{d\mathbf{q}}{du} - T_i \right) \left(\frac{d\mathbf{q}}{du} - T_i \right)^T \qquad (B.8)$$

In order to elucidate the second order derivative of $\mathbf{q}$, the coefficient of vector e in its first order counterpart will be abbreviated using $\mathcal{N}$ to denote the numerand, and $\mathcal{D}$ for the denominator. Therefore, applying the quotient rule:

$$\frac{d^2q}{du^2} = \frac{d}{du}\left(\frac{dq}{du} \equiv \frac{\mathcal{N}}{\mathcal{D}}\, e\right) = \frac{\frac{d\mathcal{N}}{du}\,\mathcal{D} - \mathcal{N}\,\frac{d\mathcal{D}}{du}}{\mathcal{D}^2}\, e \tag{B.9}$$

where for each of the hypothesis for $\mathbf{q}$:

$$\begin{aligned}
\frac{d\mathcal{N}_A}{du} &= \left(\frac{d^2T_i}{du^2}\,(\mathbf{c}_i - \mathbf{o})^T + 3\,\frac{dT_i}{du}\,T_i^T\right)(T_i\,e^T) - \left(T_i\,(\mathbf{c}_i - \mathbf{o})^T\right)\left(\frac{d^2T_i}{du^2}\,e^T\right) \\
\frac{d\mathcal{D}_A}{du} &= 2\left(\frac{dT_i}{du}\,e^T\right)(T_i\,e^T)
\end{aligned} \tag{B.10}$$

$$\begin{aligned}
\frac{d\mathcal{N}_B}{du} &= \left(\frac{d^2B_i}{du^2}\,(\mathbf{c}_i - \mathbf{o})^T + \frac{dB_i}{du}\,T_i^T\right)(B_i\,e^T) - \left(B_i\,(\mathbf{c}_i - \mathbf{o})^T\right)\left(\frac{d^2B_i}{du^2}\,e^T\right) \\
\frac{d\mathcal{D}_B}{du} &= 2\left(\frac{dB_i}{du}\,e^T\right)(B_i\,e^T)
\end{aligned} \tag{B.11}$$

and the higher order derivatives of the vector fields not covered by Equation B.6 can be computed as:

$$\begin{aligned}
\frac{d^2T_i}{du^2} &= \left.\frac{d^3\mathbf{s}_i}{du^3}\right|_{v=0} \\
\frac{d^2N_i}{du^2} &= \left.\frac{d^2T_i}{du^2} \wedge \frac{d\mathbf{s}_i}{dv}\right|_{v=0} + 2\,\frac{dT_i}{du} \wedge \left.\frac{d^2\mathbf{s}_i}{du\,dv}\right|_{v=0} + T_i \wedge \left.\frac{d^3\mathbf{s}_i}{du^2\,dv}\right|_{v=0} \\
\frac{d^2B_i}{du^2} &= \frac{d^2T_i}{du^2} \wedge N_i + 2\,\frac{dT_i}{du} \wedge \frac{dN_i}{du} + T_i \wedge \frac{d^2N_i}{du^2}
\end{aligned} \tag{B.12}$$

B.3   Special conditions for cubic boundaries

For the case of a cubic boundary edge, further simplifications can be applied, resulting in an easier computation of the derivative terms involved in the expressions presented above. Assuming as well that the intersecting boundary is that corresponding to the uw-edge of the domain, we define the following quadratic curves:

$$\begin{aligned}
\mathbf{c}_i^u &= \mathbf{s}_{102}\,B_0^2(u) + \mathbf{s}_{201}\,B_1^2(u) + \mathbf{s}_{300}\,B_2^2(u) \\
\mathbf{c}_i^v &= \mathbf{s}_{012}\,B_0^2(u) + \mathbf{s}_{111}\,B_1^2(u) + \mathbf{s}_{210}\,B_2^2(u) \\
\mathbf{c}_i^w &= \mathbf{s}_{003}\,B_0^2(u) + \mathbf{s}_{102}\,B_1^2(u) + \mathbf{s}_{201}\,B_2^2(u)
\end{aligned} \tag{B.13}$$

As Farin demonstrates in [Far95], the boundary derivatives of the patch can be expressed in terms of the differences between simulatenous evaluations of these curves, yielding the following expressions:
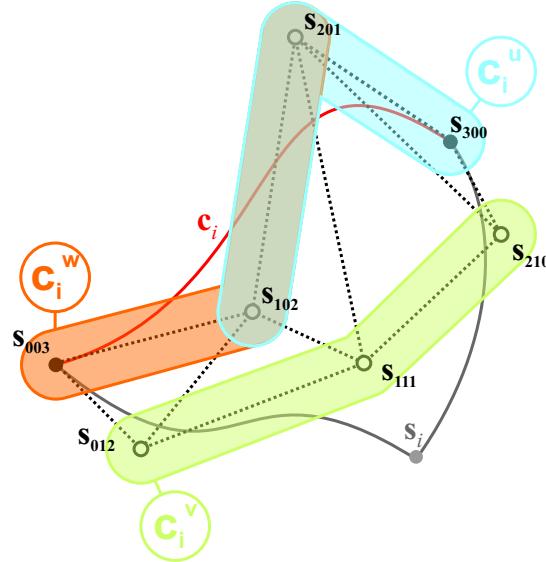
$$\left.\frac{d\mathbf{s}_i}{du}\right|_{v=0} = 3\,(\mathbf{c}_i^u - \mathbf{c}_i^w) \qquad \text{(B.14)} \qquad\qquad \left.\frac{d\mathbf{s}_i}{dv}\right|_{v=0} = 3\,(\mathbf{c}_i^v - \mathbf{c}_i^w) \qquad \text{(B.15)}$$

$$\left.\frac{d^2\mathbf{s}_i}{du^2}\right|_{v=0} = 3\left(\frac{d\mathbf{c}_i^u}{du} - \frac{d\mathbf{c}_i^w}{du}\right) \qquad\qquad \left.\frac{d^2\mathbf{s}_i}{dudv}\right|_{v=0} = 3\left(\frac{d\mathbf{c}_i^v}{du} - \frac{d\mathbf{c}_i^w}{du}\right)$$

$$\left.\frac{d^3\mathbf{s}_i}{du^3}\right|_{v=0} = 3\left(\frac{d^2\mathbf{c}_i^u}{du^2} - \frac{d^2\mathbf{c}_i^w}{du^2}\right) \qquad\qquad \left.\frac{d^3\mathbf{s}_i}{du^2dv}\right|_{v=0} = 3\left(\frac{d^2\mathbf{c}_i^v}{du^2} - \frac{d^2\mathbf{c}_i^w}{du^2}\right)$$

and once again, curve derivatives can be efficiently computed through recursive differences of the intermediate points of de Casteljau's algorithm.

C. SIMPLEX DOWNHILL METHOD

The Simplex Downhill method is a blind minimization scheme for multivariate real-valued functions, in the form $f : \mathbb{R}^n \to \mathbb{R}$, where for blind we mean that it does not require knowing the derivatives of the objective function. This makes it suitable for optimizing non-analytic error metrics, such as those described in Section 4.3.1. The technique operates by trying to enclose the solution in a n-simplex (a convex body of $n+1$ vertices in $\mathbb{R}^n$), that is initially constructed around a start-up approximation $\mathbf{p}_0$:

$$\forall i \in \{1 \cdots n\}, \ \mathbf{p}_i = \mathbf{p}_0 + \lambda e_i \tag{C.1}$$

where e_i denotes the ith unit vector of the search space. The values of the objective function at each of the $\mathbf{p}_i$s gives a discrete measure of the local variation in the environment of the simplex's centroid, which is similar to finite differences; according to these values vertices are ordered from maximum to minimum, and a series of different steps can be taken for the simplex to converge to the solution:

initial simplex   reflection   reflection and expansion

contraction   multiple contraction
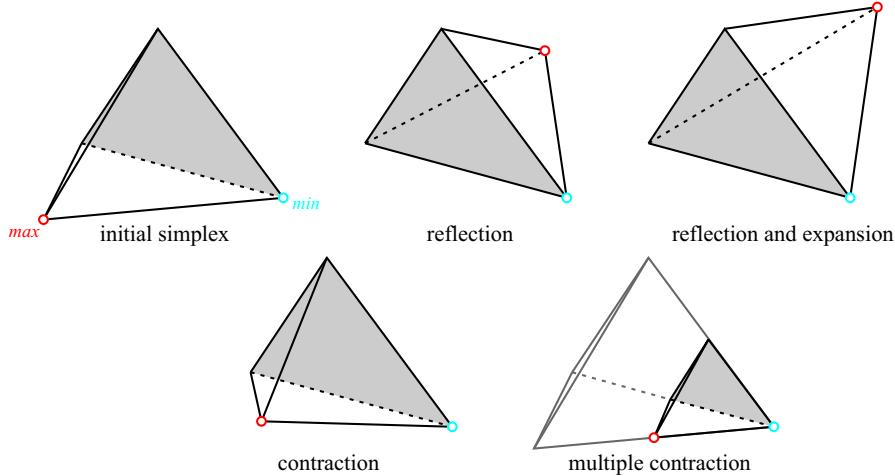
- *reflection* - transposes the maximum of the simplex's vertices across the hyper-face defined by the n remaining $\mathbf{p}_i$s, thus advancing its centroid in a direction of local descent (hence the *downhill* term).

- *reflection and expansion* - whenever a reflection yields a worse new vertex, this steps flips back its effect doubling its magnitude, comprising a larger parcel of the search space in order to overcome a local minimum

- *contraction* - the simplex contracts along one of its edges in the direction opposite to the maximum vertex

- *multiple contraction* - all $\mathbf{p}_i$s shift towards the minimum vertex, closing up on a potential solution

The heuristics that decidewhich step to adopt depends greatly on the optimization scenario, although the implementation provided in [PTF92] provides a generic framework. Severe limitations are imposed by the implicit linearization, restricting the scope of application of this method to local optimizations; should the minimization landscape be irregular enough to allow for several local minima, multiple processes can be spawned in the neighbourhood of a candidate solution in order to check for a global alternative. This may not be very efficient, but since it is an online process this is not such a significant factor.

BIBLIOGRAPHY

[ADR94]   N Arad, N Dyn, D Reisfeld, and Y Yeshurun. "Image warping by Radial Basis Functions: application to facial expressions." In *Computer Vision, Graphics, and Image Processing. Graphical Models and Image Processing*, volume 56, pp. 161–172, 1994.

[ANR74]   N Ahmed, T Natarajan, and K R Rao. "Discrete Cosine Transform." In *IEEE Transactions on Computers*, volume 23, pp. 90–93, 1974.

[ASW93]   T Akimoto, Y Suenaga, and R Wallace. "Automatic creation of 3D facial models." In *IEEE Computer Graphics and Applications*, volume 13(5), pp. 16–22, 1993.

[AW00]   George Arfken and Hans Weber. *Mathematical Methods for Physicists - Fifth Edition*. Springer-Verlag, 2000.

[B78]   P Bézier. "General distortion of an ensemble of biparametric surfaces." In *Computer-Aided Design*, volume 10, pp. 117–120, 1978.

[Bar84]   A H Barr. "Global and local deformations of solid primitives." In *Proceedings of ACM SIGGRAPH'84*, pp. 21–30, 1984.

[BBP01]   Gaspard Breton, Christian Bouville, and Danielle Pelé. "FaceEngine: a 3D facial animation engine for real time applications." In *Proceedings of WEB3D'01*, pp. 15–22. ACM, 2001.

[BCS97]   C Bregler, M Covell, and M Slaney. "Video rewrite: driving visual speech with audio." In *Proceedings of ACM SIGGRAPH'97*, pp. 353–360, 1997.

[Ber85]   P Bergeron. "Controlling facial expressions and body movements in the computer-generated animated short Tony de Peltrie." In *Tutorials of ACM SIGGRAPH'85*, 1985.

[Ber00]   M de Berg. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[BH94a]   D E Breen and D H House. "A particle-based model for simulating the draping behaviour of woven cloth." In *Textile Research Journal*, volume 64(11), pp. 663–685, 1994.

[BH94b]   D E Breen and D H House. "Predicting the drape of woven cloth using interacting particles." In *Proceedings of ACM SIGGRAPH'94*, pp. 365–372, 1994.

[Bli78]   J Blinn. "Simulation of wrinkled surfaces." In *Proceedings of ACM SIGGRAPH'78*, pp. 286–293, 1978.

[Bli97]   J Blinn. "Models of light reflection for computer synthesized pictures." In *Proceedings of ACM SIGGRAPH'97*, pp. 192–198, 1997.

[Blo02]   Jules Bloomenthal. "Medial-based vertex deformation." In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 147–151. ACM Press, 2002.

[Boa83]   Mary L Boas. *Mathematical Methods in the Physical Sciences - Second Edition*. John Wiley & Sons, 1983.

[Boe88]   W Boehm. "Local smooth surface interpolation: a classification." In *Computer-Aided Design*, volume 20, pp. 370–372, 1988.

[Bow81]   A Bowyer. "Computing Dirichlet tesellations." In *The Computer Journal*, volume 24, pp. 162–166, 1981.

[Bra03]   Bargen Bradley. *Inside DirectX - reference book*. Microsoft Press, 2003.

[BV99]   V Blanz and T Vetter. "A morphable model for the synthesis of 3d faces." In *Proceedings of ACM SIGGRAPH'99*, pp. 187–194, 1999.

[BV03]   V Blanz and T Vetter. "Face recognition based on fitting a 3D morphable model." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 25, pp. 1063–1074, 2003.

[BW98]   D Baraff and A Witkin. "Large steps in cloth simulation." In *Proceedings of ACM SIGGRAPH'98*, pp. 43–54, 1998.

[BWK01]   D Baraff, A Witkin, and M Kass. "Physically Based Modelling, ACM SIGGRAPH 2001 Course Notes.", 2001.

[BY95]   M J Black and Y Yacoob. "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion." In *ICCV*, pp. 374–381, 1995.

[CET98]   T Cootes, G Edwards, and C Taylor. "Active appearance models." In *Proceedings of the European Conference on Computer Vision*, pp. 484–498, 1998.

[CG91]   G Celniker and D Gossard. "Deformable curve and surface finite elements for free-form shape design." In *Proceedings of ACM SIGGRAPH'91*, pp. 257–266, 1991.

[CHP89]   J Chadwick, D R Haumann, and R E Parent. "Layered construction for deformable animated characters." In *Proceedings of ACM SIGGRAPH'89*, pp. 234–243, 1989.

[CLK01]   B Choe, H Lee, and H S Ko. "Performance-driven muscle-based facial animation." In *Journal of Visualization and Computer Animation*, pp. 67–79, 2001.

[CMC02]   M Cohen, D Massaro, and R Clark. "Training a talking head." In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, pp. 499–510, 2002.

[Coq90]   S Coquillart. "Extended Free-Form Deformation: a sculpturing tool for 3D geometric modelling." In *Proceedings of ACM SIGGRAPH'90*, pp. 187–193, 1990.

[CT65]   R Clough and J Tocher. "Finite element stiffness matrices for analysis of plates in blending." In *Proceedings of Conference on Matrix Methods in Structural Analysis*, pp. 71–88, 1965.

[CYM92]   M Carignan, Y Yang, N Magnenat-Thalmann, and D Thalmann. "Dressing animated synthetic actors with complex deformable clothes." In *Proceedings of ACM SIGGRAPH'92*, pp. 92–104, 1992.

[Dev97]   R W Devaul. *Cloth dynamics simulation: a Masters Thesis*. Visualization Science Program, College of Arquitecture, Texas A&M University, 1997.

[DHT00]   Paul Debevec, Tim Hawkins, Chris Tchou, Haarm Pieter Duiker, Westley Sarokin, and Mark Sagar. "Acquiring the reflectance field of a human face." In *Proceedings of ACM SIGGRAPH'00*, pp. 145–156, 2000.

[DP90]   C T J Dodson and T Poston. *Tensor Geometry: The Geometric Viewpoint and its Uses - Second Edition*. Springer-Verlag, 1990.

[Duc62]   G B A Duchenne. *The mechanisms of human facial expression*. Paris. (edited and translated by R. Andrew Cuthbertson, Cambridge Univ Press, 1990), 1862.

[EDP94]   I Essa, T Darrell, and A Pentland. "Tracking facial motion." In *Proceedings of the Workshop on Motion of Non-rigid and Articulated Objects*, pp. 36–42. IEEE Computer Society, 1994.

[EF78]   P Ekman and W V Friesen. *Facial action coding system*. Consulting Psychologists Press inc., 1978.

[Eld77]   H R Elden. *Biophysical properties of skin*. Wiley-Interscience, 1977.

[EM03]   J Edge and S Maddock. "Image-based talking heads using Radial Basis Functions." In *Proceedings of EGUK'03*, pp. 74–80. Eurographics, 2003.

[EM04]   J Edge and S Maddock. *Constraint-based synthesis of visual speech*. ACM SIGGRAPH'04 Sketches Programme, 2004.

[EPM98]   M Escher, I S Pandzic, and N Magnenat-Thalmann. "Facial deformations for MPEG-4." In *Proceedings of Computer Animation 1998, (CA'98)*, pp. 56–. IEEE Computer Society, 1998.

[ES04]   D H Eberly and K Shoemake. *Game Physics*. Morgan Kaufmann, 2004.

[ESM04a]   J Edge, M Sánchez, and S Maddock. *Animating speech from motion fragments*. Research Memorandum CS-04-10. Department of Computer Science, University of Sheffield, 2004.

[ESM04b]   J Edge, M Sánchez, and S Maddock. "Reusing motion capture data to animate visual speech." In *Symposium on Language, Speech and Gesture for Expressive Characters, part of the AISB 2004 Convention: Motion, Emotion and Cognition*, pp. 66–74. University of Leeds, 2004.

[Ess95]   I A Essa. *Analysis, interpretation, and synthesis of facial expressions*. PhD thesis, Massachusetts Institute of Technology, 1995.

[Far86]   G Farin. "Triangular Bernstein-Bézier patches." In *Computer Aided Geometric Design*, volume 3, pp. 83–127, 1986.

[Far94]   L Farkas. *Anthropometry of the Head and Face*. Raven Press, 1994.

[Far95]   G Farin. *Curves and Surfaces for Computer-Aided Geometric Design - A Practical Guide - Fourth Edition*. Academic Press, 1995.

[FK03]   Randima Fernando and Mark J Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-time Graphics*. Addison Wesley, 2003.

[FLM92]   O D Faugeras, Q T Luong, and S J Maybank. "Camera self-calibration: theory and experiments." In *Proceedings of the European Conference on Computer Vision '92*, pp. 321–334, 1992.

[FNK99]   D Fidaleo, J Noh, T Kim, R Enciso, and U Neumann. "Classification and volume morphing for performance-driven facial animation." In *International Workshop on Digital and Computational Video (DCV'99)*, pp. 235–239, 1999.

[For03]   J Fordham. "Middle earth strikes back." In *Cinefex*, volume 92, pp. 71–142, 2003.

[GGW98]   B Guenter, C Grimm, D Wood, H Malvar, and F Pighin. "Making faces." In *Proceedings of ACM SIGGRAPH'98*, pp. 55–66, 1998.

[GM01]   Stephane Garchery and Nadia Magnenat-Thalmann. "Designing MPEG-4 Facial Animation Tables for Web Applications." In *Proceedings of Multimedia Modeling 2001*, pp. 39–59, 2001.

[Gra98]   F Sebastian Grassia. "Practical parameterization of rotations using the exponential map." In *Journal of Graphics Tools*, volume 3, pp. 29–48, 1998.

[GW04]   C F Gerald and P O Weatley. *Applied Numerical Analysis, Seventh Edition.* Addison Wesley, 2004.

[HDB96]   D H House, R W DeVaul, and D E Breen. "Towards simulating cloth dynamics using interacting particles." In *International Journal of Clothing Science and Technology*, volume 8, pp. 75–94, 1996.

[HFE00]   A. Haro, M. Flickner, and I. Essa. "Detecting and tracking eyes by using their physiological properties, dynamics, and appearance." In *Proceedings of CVPR 2000*, pp. 163–168, 2000.

[HG75]   R L Hardy and W M Gofert. "Least squares prediction of gravity anomalies, geoidal undulations, and deflections of the vertical multiquadric harmonic functions." In *Geophysical Research Letters*, volume 2, pp. 423–426, 1975.

[Hop96]   H Hoppe. "Progressive meshes." In *Proceedings of ACM SIGGRAPH'96*, pp. 99–108, 1996.

[ita00]   *Intel Itanium Architecture Manual - Volume 1: Application Architecture,* (*http://www.intel.com/design/itanium/manuals/iiasdmanual.htm*). Intel Literature Center, 2000.

[Jac93]   O L R Jacobs. *Introduction to Control Theory - Second Edition.* Oxford University Press, 1993.

[JTD03]   P Joshi, W C Tien, M Desbrun, and F Pighin. "Learning controls for blend shape based realistic facial animation." In *Proceedings of ACM SIGGRAPH'03*, 2003.

[JZ02]   Q Ji and Z Zhu. "Eye and gaze tracking for interactive graphic display." In *Int. Symp. on Smart Graphics*, pp. 139–148, 2002.

[KA91]   T Kurihara and K Arai. "A transformation method for modeling and animation of the human face from photographs.", 1991.

[Kal60]   R E Kalman. "A new approach to linear filtering and prediction problems." In *Transactions of the ASME - Journal of Basic Engineering*, volume 82(D), pp. 35–45, 1960.

[Kan90]   E J Kansa. "Multiquadric - a scattered data approximation scheme with applications to computational fluid dynamics II." volume 19 (8/9), pp. 147–161, 1990.

[KGB98]   R M Koch, M H Gross, and A A Bosshard. "Emotion Editing using Finite Elements." In *Proceedings of Eurographics'98*, pp. 295–302, 1998.

[KGC96]   R M Koch, M H Gross, F R Carls, D F von Büren, G Fankhauser, and Y I H
          Parish. "Simulating facial surgery using the Finite Elements Method." In
          Proceedings of ACM SIGGRAPH'96, pp. 421–428, 1996.

[KGM01]   S Kshirsagar, S Garchery, and N Magnenat-Thalmann. "Feature point
          based mesh deformation applied to MPEG-4 facial animation." In *DE-
          FORM/AVATARS, IFIP Conference Proceedings*, volume 196, pp. 24–34.
          Kluwer, 2001.

[KHS01]   K Kahler, J Haber, and H P Siedel. "Geometry-based muscle modeling for
          facial animation." In *Proceedings of Graphics Interface'01*, pp. 37–46, 2001.

[KL96]   V Krishnamurthy and M Levoy. "Fitting smooth surfaces to dense polygon
          meshes." In *Proceedings of ACM SIGGRAPH 1996*, pp. 313–324, 1996.

[Klo86]   F Klok. "Two Moving Coordinate Frames for Sweeping along a 3D Trajec-
          tory." In *Computer Aided Geometric Design*, volume 3, pp. 217–229, 1986.

[KM77]   R Kent and F Minifie. "Coarticulation in recent speech production models."
          In *Journal of Phonetics*, volume 5, pp. 115–135, 1977.

[KMM92]   P Kalra, A Mangili, N Magnenat-Thalmann, and D Thalmann. "Simula-
          tion of facial muscle actions based on rational free form deformations." In
          Proceedings of Eurographics'92, pp. 59–69, 1992.

[Kob00]   Leif Kobbelt. "$\sqrt{3}$ Subdivision." In *Proceedings of ACM SIGGRAPH'00*, pp.
          103–112, 2000.

[KPO00]   S King, R Parent, and B Olafsky. "An anatomically-based 3d parametric lip
          model to support facial animation and synchronized speech." In *Proceedings
          of Deform 2000*, pp. 7–9, 2000.

[KR88]   B W Kernighan and D M Ritchie. *The C Programming Language, Second
          Edition*. Prentice Hall, Inc., 1988.

[KWT88]   M Kass, A Witkin, and D Terzopoulos. "Snakes: active contour models." In
          International Journal of Computer Vision, volume 1, pp. 321–331, 1988.

[Lan99]   H P Langtangen. *Computational Partial Differential Equations*. Springer-
          Verlag, 1999.

[Law77a] C L Lawson. "Software for C1 surface interpolation." In *Mathematical Soft-
          ware III (John R. Rice, editor)*, pp. 161–194. Academic Press, 1977.

[Law77b] C L Lawson. "Software for C1 Surface Interpolation." In *Mathematical Soft-
          ware III (John R. Rice, editor)*, pp. 161–194. Academic Press, 1977.

[Law06]   N D Lawrence. *Large scale learning with the Gaussian process latent variable
          model*. Technical Report no CS-06-05, University of Sheffield, 2006.

[LC04]   C Larboulette and M P Cani. "Real-time dynamic wrinkles." In *Proceedings of CGI'04*, pp. 522–525. IEEE Computer Society, 2004.

[LCF00]   J P Lewis, M Cordner, and N Fong. "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation." In *Proceedings of ACM SIGGRAPH'00*, pp. 165–172, 2000.

[LCJ94]   F Lazarus, S Coquillart, and P Jancène. "Axial deformations: an intuitive deformation technique." volume 26, pp. 607–613, August 1994.

[Liu00]   I S Liu. *Continuum Mechanics*. Springer-Verlag, 2000.

[LTB96]   J Luettin, N A Thacker, and S W Beet. "Locating and tracking facial speech features." In *Proceedings of the International Conference on Pattern Recognition*, pp. 652–656, 1996.

[LTW93]   Y Lee, D Terzopoulos, and K Waters. "Constructing physics-based facial models of individuals." In *Proceedings of Graphics Interface'93*, pp. 1–8, 1993.

[LTW95]   Y Lee, D Terzopoulos, and K Waters. "Realistic modeling for facial animation." In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 55–62. ACM Press, 1995.

[May79]   P S Maybeck. *Stochastic models, estimation and control*. Academic Press, 1979.

[McG98]   G McGunnigle. *The classification of texture surfaces under varying illumination direction*. PhD thesis, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 1998.

[Mey00]   C D Meyers. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial & Applied Mathematics, 2000.

[MJ96]   R MacCracken and K Joy. "Free-Form Deformations with lattices of arbitrary topology." In *Proceedings of ACM SIGGRAPH'96*, pp. 181–188, 1996.

[MKL02]   Nadia Magnenat-Thalmann, Prem Kalra, Jean Luc Lévêque, Roland Bazina, Dominique Batisse, and Bernard Querleux. "A computational skin model: fold and wrinkle formation." In *IEEE Transactions on Information Technology in Biomedicine*, volume 6(4), pp. 317–323, 2002.

[MLT88]   N Magnenat-Thalmann, R Laperriere, and D Thalmann. "Joint-Dependent Local Deformations for Hand Animation and Object Grasping." In *Proceedings of Graphics Interface'88*, pp. 26–33, 1988.

[MMA89]   N Magnenat-Thalmann, H Minh, M Angelis, and D Thalmann. "Design, transformation and animation of human faces." In *The Visual Computer*, volume 5, pp. 32–39, 1989.

[MPE01]   MPEG. "ISO/IEC 14496-2 - Information technology - Coding of audio-visual objects - Part 2: Visual. Second edition.", 2001.

[MPT88]   N Magnenat-Thalmann, N E Primeau, and D Thalmann. "Abstract muscle actions procedures for human face animation." In *The Visual Computer*, volume 3(5), pp. 290–297, 1988.

[MT97]   L Moccozet and N Magnenat Thalmann. "Dirichlet Free-Form Deformation and their application to hand simulation." In *Proceedings of Computer Animation 1997 (CA'97)*. IEEE Computer Society, 1997.

[NFN00]   J Y Noh, D Fidaleo, and U Neumann. "Animated deformations with Radial Basis Functions." In *Proceedings of VRST 2000*, pp. 166–174, Seoul, Korea, 2000.

[NJ04]   K Na and M Jung. "Hierarchical retargetting of fine facial motions." In *Proceedings of Eurographics'04*, pp. 687–695, 2004.

[NN01]   J Noh and U Neumann. "Expression cloning." In *Proceedings of ACM SIGGRAPH'01*, pp. 277–288, 2001.

[nvi02]   *Cg Toolkit 1.1 (`http://developer.nvidia.com/object/cg_toolkit_1_1.html`)*. nVidia Corp, 2002.

[ON66]   B O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[Opt]   *OPTOTRACK*. Northern Digital Inc (`http://www.ndigital.com/`).

[Par74]   F Parke. *A parametric model for human faces*. PhD thesis, University of Utah, 1974.

[Pat92]   M Patel. *Making faces: the facial animation, construction and editing system*. Technical Report 92-55, Department of Mathematical Sciences, University of Bath, 1992.

[PAY02]   I S Pandzic, J Ahlberg, and L You. "Subjective evaluation of the expressiveness of animated face models (`http://www.tel.fer.hr/users/ipandzic/frames/papers/vc02.pdf`)." In *the Visual Computer Journal - submitted*, 2002.

[PB81]   N Platt and S Badler. "Animating facial expressions." In *Proceedings of ACM SIGGRAPH'81*, pp. 245–252, 1981.

[PB88]   J Platt and A Barr. "Constraint Methods for Flexible Models." In *Proceedings of ACM SIGGRAPH'88*, pp. 279–288, 1988.

[Pet90]   J Peters. "Local smooth surface interpolation: a classification." In *Computer Aided Geometric Design*, volume 7, pp. 191–195, 1990.

[Pet02]   J Peters. "Geometric continuity." In *Handbook of Computer Aided Geometric Design, Chapter 8*. Elsevier, 2002.

[Pig99]   F Pighin. *Modeling and animating realistic faces from images*. Department of Computer Science and Engineering, University of Washington, 1999.

[PLG91]   E C Patterson, P C Litwinowicz, and N Greene. "Facial animation by spatial mapping." In *Computer Animation 91*, pp. 31–44. Springer-Verlag, 1991.

[PM01]   M Pitermann and K Munhall. "An inverse dynamics approach to face animation." In *Journal of the Acoustical Society of America*, volume 110, pp. 1570–1580, 2001.

[Pol]   *POLARIS*. Northern digital Inc (`http://www.ndigital.com/`).

[PP01]   S Pasquiarello and C Pelachaud. "Greta: a simple facial animation engine." In *6th Online Conference on Soft Computing in Industrial Applications*, 2001.

[PS03]   H Pyun and S Shin. "An example-based approach for facial expression cloning." In *Proceedings of SCA'03*, pp. 167–176, 2003.

[PSS99]   F Pighin, R Szeliski, and D H Salesin. "Resynthesizing facial animation through 3D model-based tracking." In *ICCV (1)*, pp. 143–150, 1999.

[PTF92]   William H Press, S A Teulkolsky, and B P Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.

[PW91]   M Patel and P J Willis. "Faces: facial animation, construction and editing system." In *Proceedings of Eurographics'91*, pp. 33–45, 1991.

[Qu92]   G Quénot. "The orthogonal algorithm for optical flow detection using dynamic programming." In *Proceedings of IEEE conference on Acoustics, Speech and Signal Processing*, pp. 249–252, 1992.

[RPF97]   D Recasens, M Pallares, and J Fontdevila. "A model of lingual coarticulation based on articulatory constraints." In *Journal of the Acoustical Society of America*, volume 102, pp. 544–561, 1997.

[RTG98]   H Rushmeier, G Taubin, and A Gueziec. "Applying shape from lighting variation to bump capture." In *Proceedings of Eurographics Rendering Workshop 1998*, pp. 35–44, 1998.

[RY90]   K R Rao and P Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, 1990.

[S P70]   J N Goodier S P Timoshenko. *Theory of Elasticity - 3rd Edition*. McGraw-Hill, 1970.

[SA96]   M Segal and K Akeley. *The OpenGL Graphics System: A Specification (Version 1.1)*. Silicon Graphics, Inc. (http://www.sgi.com/Technology/openGL/glspec/glspec.html), 1996.

[SBM95] Malcolm Sambridge, Jean Braun, and Herbert McQueen. "Geophysical parameterization and interpolation of irregular data using natural neighbours." In *Geophysical Journal International*, volume 122, pp. 837–857, 1995.

[SCP00] H Seo, F Cordier, L Philippon, and M Magnenat-Thalmann. "Interactive modelling of MPEG-4 deformable human body models." In *Proceedings of Deform 2000*, pp. 120–131. Kluwer Academic Publishers, 2000.

[SCS03] Jonathan Starck, Gordon Collins, Raymond Smith, Adrian Hilton, and John Illingworth. "Animated statues." In *Machine Vision and Applications*, volume 14, pp. 248–259, 2003.

[SEK03] M Sánchez, J Edge, S King, and S Maddock. "Use and re-use of facial animation motion capture data." In *Proceedings of Video, Vision and Graphics'03*, pp. 135–142. IEEE Computer Society, 2003.

[SEM04] M Sánchez, J Edge, and S Maddock. *Realistic performance-driven facial animation using hardware acceleration*. Technical Report CS-04-10, Department of Computer Science, University of Sheffield, 2004.

[SF98] K Singh and E Fiume. "Wires: a geometric deformation technique." In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 405–414. ACM Press, 1998.

[Sho85] Ken Shoemake. "Animating rotation with quaternion curves." In *Proceedings of ACM SIGGRAPH'85*, pp. 245–254, 1985.

[Sib80] R Sibson. "A vector identity for the Dirichlet tessellation." In *Math. Proc. Cambridge Philosophy Society*, volume 87, pp. 151–155, 1980.

[SK00] K Singh and E Kokkevis. "Skinning characters using Surface oriented Free-Form Deformations." In *Proceedings of Graphics Interface 2000*, pp. 27–34, 2000.

[SM03] M Sánchez and S Maddock. "Planar Bones for MPEG-4 Facial Animation." In *Proceedings of EG UK'03*, pp. 81–88. Eurographics, 2003.

[SMK97] M U Ramos Sánchez, J Matas, and J Kittler. "Statistical chromacity models for lip tracking with B-Splines." In *Proceedings of the International Conference on Audio- and Video-based Biometric Person Authentication*, 1997.

[SP86] T W Sederberg and S R Parry. "Free-Form Deformation of solid geometric models." In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 151–160. ACM Press, 1986.

[SPC97] F Scheepers, R E Parent, W E Carlson, and S F May. "Anatomy-based modeling of the human musculature." In *Proceedings of ACM SIGGRAPH'97*, pp. 163–172, 1997.

[SZL92]   W J Schroeder, J A Zarge, and W E Lorensen. "Decimation of triangle meshes." In *Proceedings of ACM SIGGRAPH'92*, pp. 65–70, 1992.

[TC03]   J M Thorne and D J Chatting. "Prometheus: facial modelling, tracking and puppetry - poster." In *Proceedings of Video, Vision and Graphics'03*. IEEE Computer Society, 2003.

[TF88]   D. Terzopoulos and K. Fleischer. "Deformable models." In *The Visual Computer 1988*, volume 4, pp. 306–331, 1988.

[TH98]   H Tao and T Huang. "Bezier volume deformation model for facial animation and video tracking." In *Proceedings of IFIP Workshop on Modeling and Motion Capture Techniques for Virtual Environments (CAPTECH'98)*, pp. 242–253, 1998.

[TLY03]   Pei Hsuan Tu, I Chen Lin, Jeng Sheng Yeh, Rung Huei Liang, and Ming Ouhyoung. "Expression detail mapping for realistic facial animation." In *Proceedings of 8th International conference on CAD/graphics*, pp. 48–60, 2003.

[TPB87]   D Terzopoulos, J Platt, A Barr, and K Fleischer. "Elastically deformable models." In *Proceedings of ACM SIGGRAPH'87*, pp. 205–214, 1987.

[Tsa87]   R Y Tsai. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses." In *IEEE Journal of Robotics and Automation*, volume 3, pp. 323–344, 1987.

[Tur50]   Allan Turing. "Computing machinery and intelligence." In *Mind*, pp. 433–460, 1950.

[VCM95]   P Volino, M Courchesne, and N Magnenat-Thalmann. "Versatile and efficient techniques for simulating cloth and other deformable objects." In *Proceedings of ACM SIGGRAPH'95*, pp. 137–144, 1995.

[Vel92]   R C Veltkamp. "Survey of continuities of curves and surfaces." In *Computer Graphics Forum*, volume 11, pp. 93–112, 1992.

[Vic]   *Vicon 8*. Vicon Motion Systems (`http://www.vicon.com/`).

[VM99]   P Volino and N Magnenat-Thalmann. *Fast Geometrical Wrinkles on Animated Surfaces*. Short Communication. MIRALab, University of Geneva, 1999.

[VY92]   M L Viaud and H Yahia. "Facial animation with wrinkles." In *3rd Workshop on Animation, Eurographics'92*, Cambridge, UK, 1992.

[Wat81]   D F Watson. "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes." In *The Computer Journal*, volume 24, pp. 167–172, 1981.

[Wat87]   K Waters. "A muscle model for animation three-dimensional facial expression." In *Proceedings of ACM SIGGRAPH'87*, pp. 17–24, 1987.

[WB97]   G Welch and G Bishop. "SCAAT: incremental tracking with incomplete information." In *Proceedings of ACM SIGGRAPH'97*, pp. 333–344, 1997.

[WB01]   G Welch and G Bishop. *An Introduction to the Kalman Filter.* ACM SIG-GRAPH 2001 Course Notes, 2001.

[Web00]   J Weber. "Run-time skin deformation (`http://www.intel.com/ial/3dsoftware/index.htm`)." In *Proceedings of Game Developers Conference 2000*, 2000.

[Wil90]   Lance Williams. "Performance-driven facial animation." In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pp. 235–242. ACM Press, 1990.

[WKM97]   Yin Wu, Prem Kalra, and Nadia Magnenat-Thalmann. "Physically based wrinkle simulation and skin rendering." In *Eurographics Workshop on Animation and Simulation*, pp. 69–79. Springer, 1997.

[WKM99]   Yin Wu, Prem Kalra, Laurent Moccozet, and Nadia Magnenat-Thalmann. "Simulating wrinkles and skin aging." In *The Visual Computer*, volume 15, pp. 183–198, 1999.

[WL93]   Y Wang and C Leon. *Langwidere: A Hierarchical Spline Based Facial Animation System with Simulated Muscles. Master's thesis.* Department of Computer Science, University of Calgary, 1993.

[WMT94]   Yin Wu, Nadia Magnenat-Thalmann, and Daniel Thalmann. "A plastic-visco-elastic model for wrinkles in facial animation and skin aging." In *Proceedings of the 2nd Pacific Conference on Fundamentals of Computer Graphics*, pp. 201–213. World Scientific Publishing Co., Inc., 1994.

[Woo80]   R J Woodham. "Photometric method to determining surface orientation from multiple images." In *Optical Engineering*, volume 19, pp. 139–144, 1980.

[YBS03]   S Yoshizawa, A G Belyaev, and H P Seidel. "Free-form skeleton-driven mesh deformations." In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, 2003.

[Zie89]   C O Zienkiewicz. *The finite element method. - Vol.1 : Basic formulation and linear problems - Fourth Edition.* McGraw-Hill, 1989.