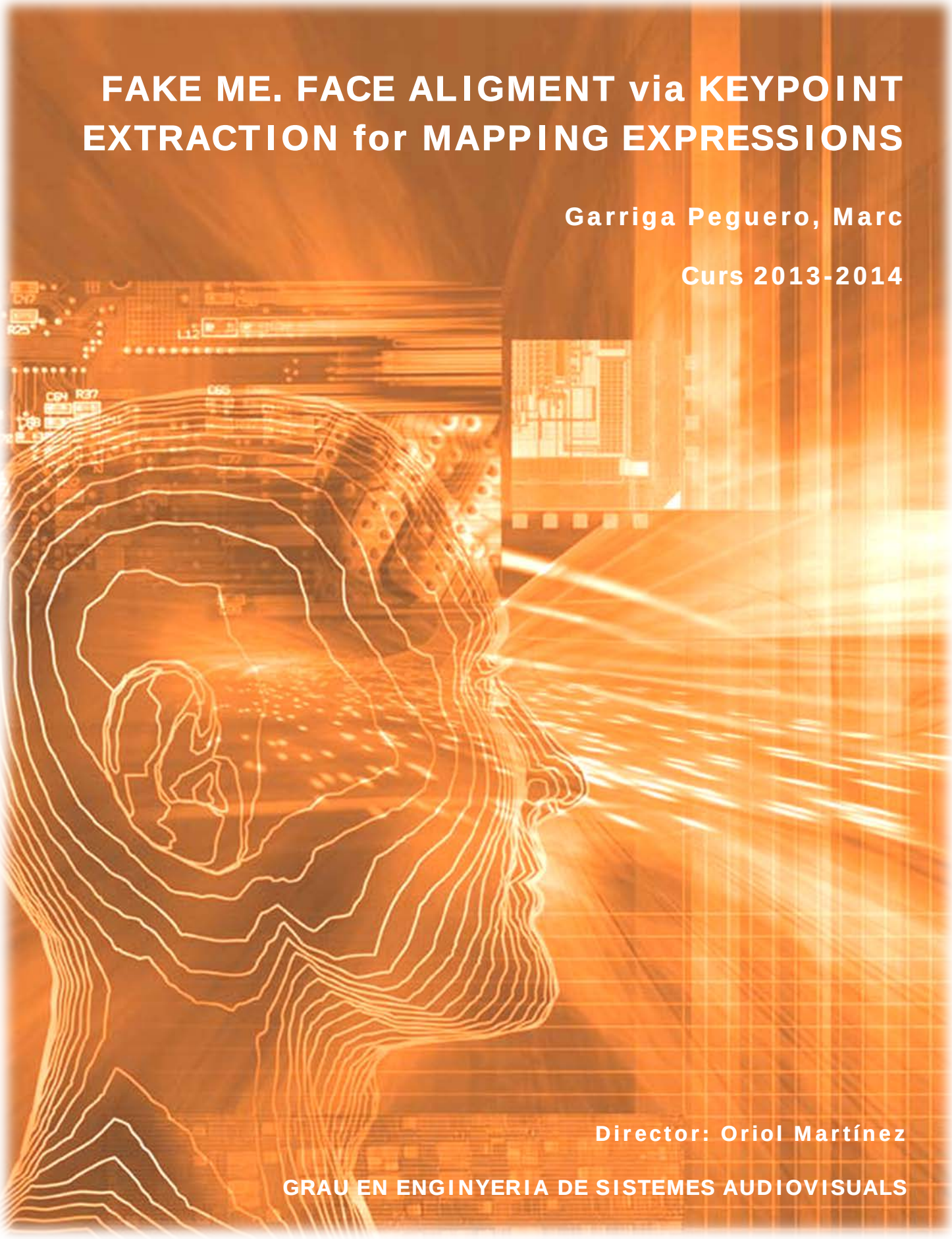
FAKE ME. FACE ALIGMENT via KEYPOINT EXTRACTION for MAPPING EXPRESSIONS

Garriga Peguero, Marc

Curs 2013-2014



Director: Oriol Martínez

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

FAKE ME. FACE ALIGMENT via KEYPOINT EXTRACTION for MAPPING EXPRESSIONS

Marc Garriga Peguero

TREBALL FI DE GRAU

Grau en Enginyeria de Sistemes Audiovisuals

ESCOLA SUPERIOR POLITÈCNICA UPF

2013 - 2014

DIRECTOR DEL TREBALL

Xavier Binefa

Oriol Martínez Pujol

A la meva família, per haver-me
donat forces sempre que m'han faltat.
Als meus amics, per l'alegria que m'han donat.

Per vosaltres.

Agraïments

En primer lloc, agrair al meu director de treball Xavier Binefa l'oportunitat que m'ha brindat en poder treballar en aquest projecte i la seva confiança que ha dipositat en mi.

Agrair sobretot a l'Adrià, en Lluis i en Pol tota l'ajuda que m'han proporcionat des de el primer dia que vaig començar aquest treball, donant-me un cop de mà sempre que ho he necessitat.

Per últim i especialment, vull agrair al meu tutor Oriol tota la paciència que ha tingut amb mi i tot el temps que m'ha dedicat i s'ha preocupat per a que aquest projecte sortís endavant.

Gràcies a tots.

Resum

L'objectiu d'aquest Treball de Fi de Grau (TFG) és la concepció i implementació d'un mètode conegut dins del mon de la Visió per Computador com a *Face Transfer* (Transferència de Cares).

Donades dues cares, l'objectiu del mètode proposat és poder transferir les expressions i característiques facials d'una cara cap a l'altre. O sigui, ser capaç d'aplicar un "mapping" que ens permeti animar la cara o avatar d'un personatge qualsevol a través dels moviments d'una altre cara, per exemple la nostra.

Hi han moltes formes de poder realitzar el *mapping* entre la informació de dues cares. En el nostre cas, proposem fer-ho dividint-les en zones (triangles), mitjançant la detecció d'uns punts característics prèviament definits. Un cop obtingudes les zones, transferim les corresponents textures utilitzant coordenades baricèntriques per realitzar el mapeig de coordenades i una tècnica coneguda com a *Multiband Blending* per fusionar d'una manera suau les textures de les dues cares.

En aquest TFG hem explorat dos escenaris en l'aplicació de la tècnica de *Face Transfer*. El primer escenari és el mapeig de la textura d'una imatge sobre una altre cara en moviment al llarg d'una seqüència de vídeo. El segon és el cas invers, l'animació d'un avatar (una imatge) mitjançant les expressions facials captades en un vídeo.

Abstract

The aim of this TFG is the conception and implementation of a method known in Computer Vision as Face Transfer.

Given two faces, the objective of this method proposed is to be able to transfer of expressions and facial features from a face to another. This means, being able to apply a mapping that allows us to animate face or avatar of any character through the movement of another face, for instance, our face.

There are many ways to map the information between two faces. In our case, we propose to divide the faces into triangular regions, through featuring points detection previously defined. Once we got the regions, we can transfer the correspondent textures using barycentric coordinates to map and a technique known as Multiband Blending to merge both face textures in a smooth way.

In this TFG we have explored two scenarios in the application of Face Transfer technique. First one is a texture mapping of an image over another in movement along a video sequence. The second one is the inverse situation, an avatar animation through the facial expressions captured in a video sequence.

Índex

Llista de figures

Capítol 1

INTRODUCCIÓ

1.1. Motivació

Durant aquests anys de Grau en Enginyeria d'Audiovisuals a la Universitat Pompeu Fabra, assignatures relacionades amb el tractament d'imatge com Processament d'Imatge o Anàlisis i Interpretació d'Imatges m'han sorprès gratament. Aquest és un dels principals motius per els quals he escollit aquest TFG, estretament lligat tot el que he pogut aprendre al llarg del meu Grau en relació al món de la Imatge. Altrament, estan sorgint en aquests darrers anys moltes aplicacions que tenen com a base la detecció de punts característics facials (i també corporals) els quals tenen un ampli ventall d'usos. Usos com aplicacions capaces de classificar expressions facials, de reconèixer atributs facials (sexe, ètnica, edat...) que són útils per a comunicació home-màquina i inclús per a propòsits mèdics.

Aquest creixent ús en el món de les tecnologies de la comunicació i la possibilitat de fer una petita aportació en aquest ampli camp i, en concret, al recent mètode nombrat *Face Transfer*, del qual n'hi han sorprenents resultats, ha decantat la meva elecció per a dur a terme aquest projecte.

1.2. Estat de l'art

El ser humà, durant la plenitud de la seva història, ha fet ús de les expressions facials com una part vital de la comunicació cara a cara, per expressar les seves emocions i el seu estat d'ànim. Tot i que el llenguatge corporal forma també part d'aquesta comunicació entre persones, és la cara la que juga un paper vital en la forma en que ens relacionem amb els demés, i per aquest motiu és quelcom important i rellevant per els humans.

La comunitat científica, i en especial la psicologia, ha mostrat al llarg dels anys especial interès en com els humans utilitzem les expressions de la cara en la

comunicació no verbal. El propi Charles Darwin, pare de la teoria de l'evolució [14], és basa en aquest àmbit per a sustentar els seus arguments sobre com l'ésser humà prové del animal. Ja en la dècada dels 70, el famós psicòleg Ekman [13] va defensar aquesta teoria a partir de la classificació universal (això és, indiferentment de la cultura) de 6 emocions amb les seves conseqüents expressions facials: alegria, ira, por, fàstic, sorpresa i tristesa.

El món de la visió per computador s'ha interessat en la última dècada per tot el que envolta i significa la comunicació amb expressions facials. S'han proposat sistemes automàtics capaços de detectar les sis expressions facials en imatges o seqüències de vídeo. A més, han sorgit nombroses aplicacions que interpreten la informació que proporciona la cara mitjançant visió per computador. Per exemple, aplicacions que identifiquen subjectes o que inclús detecten malalties. En general, totes aquestes aplicacions es solen segmentar en tres grans fases:

Figura 1.1. *Pipeline* d'un sistema de processament facial.

Com indica la Figura 1.1, el primer que es necessita en una aplicació relacionada amb el processament facial és la pròpia localització de la cara. Per a aquest propòsit, es pot utilitzar el algoritme de *Viola&Jones* [7], molt utilitzat en càmeres digitals. Un cop tenim la cara localitzada, passem a la segona fase: extracció i representació de les dades facials. Aquesta representació és pot fer, per exemple, a partir de la obtenció automàtica de punts facials característics (com la boca, els ulls, el nas, les celles...) i utilitzant les coordenades en la imatge que tenen aquests punts com a representació de l'expressió facial que s'està produint. En la amplia metodologia per a l'obtenció de característiques facials (*facial features*), és poden observar dos grans blocs: l'obtenció per *Appearence Models* (aproximació basada en aparença) [9][15] i per *Shape Models* (aproximació basada en forma) [19].

La primera, l'aproximació basada en aparença, utilitza la informació basada en textura, és a dir, es basa en la informació de la intensitat de color dels píxels d'una imatge facial per a decidir on pot haver-hi una zona potencialment característica de la cara. Per aquest tipus de detecció, la majoria de mètodes fan ús de filtres d'imatge els quals extreuen coeficients útils per a decidir si hi ha una *facial feature* o no. Per exemple, és comú l'ús de *Gabor Filters* [16] o classificadors d'*Ada Boost* [8], que utilitzen *Haar-like filters* en cascada per a calcular diferències entre zones de la cara on s'estan aplicant aquest filtres. La segona, l'aproximació basada en forma, fa ús de models estadístics de la forma que volem detectar (en aquest cas, les característiques facials), els quals defineixen la geometria de la cara en una situació donada mitjançant la deformació d'aquest models estadístics.

Finalment, l'última fase de la Figura 1.1 es tracta del que anomenem com a tasca d'alt nivell. En aquesta fase, s'utilitza la representació de les dades facials per a un propòsit concret com, per exemple, donada una imatge facial, fer la codificació automàtica de l'expressió facial d'aquesta mitjançant aquesta representació de dades. Aquesta memòria va dirigida a la tasca d'alt nivell anomenada *Face Transfer* (transferència de cares). Aquesta tasca d'alt nivell és relativament nova en Processament d'Imatge, però ja hi han algoritmes que implementen amb molta eficàcia aquest efecte facial, com a [1] [2] [3] o [18].

1.3. Descripció del projecte

L'objectiu d'aquest projecte ha estat explorar les diverses tècniques que existeixen en el camp de Processament d'Imatge dedicades a l'animació d'avatars, i implementar-ne una.

En concret, el mètode *Face Transfer*, que tracta d'adaptar les expressions i característiques facials d'una persona cap a un altre. Aquesta tècnica ha estat molt utilitzada en els últims anys en diversos camps i aplicacions i cada cop adquireix més importància.

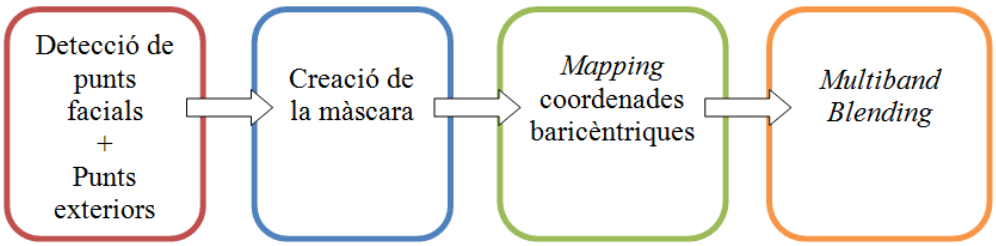


Figura 1.2. Esquema de les fases del projecte *Face Transfer*.

Específicament, en aquest TFC proposem un mètode de *Face Transfer* on, a partir d'una imatge (el nostre avatar), la podem deformar de manera que tracti d'emular les expressions facials d'una persona en una seqüència de vídeo (tals com l'alegria, por, tristesa, fàstic, sorpresa o enfado).

Tal i com es veu en el gràfic de processos (Fig. 1.2), el projecte consta de 4 fases. Primerament, es fa la detecció d'una sèrie de punts facials que ens descriuen millor les característiques de la cara (celles, ulls, nas i boca), més el càlcul de punts addicionals per a tenir una superfície major de *Face Transfer*. Acte seguit, amb aquest punts podem procedir a la creació de la màscara (segona fase), apte per al *mapping*. Aquest mapeig (tercera fase) utilitza les diferents regions de la cara definides per la màscara per a transferir la textura d'una cara origen a una cara destí. Finalment, en l'última fase utilitzem un mètode de processat d'imatge per a mitigar diferències entre textures anomenat *Multiband Blending*. Aquest mètode tracta de crear un efecte de suavitzat en la transició que hi pot haver entre dues imatges.

El mapeig amb coordenades baricèntriques que hem implementat és vàlid en les dues direccions, i per tant ens permet poder jugar amb dos versions de *Face Transfer* amb el mateix mapeig. En la primera versió partim d'una seqüència de vídeo on tractem de mapejar cada *frame* a partir d'una sola imatge. La segona versió és el procés invers al primer; és a dir, l'animació d'un avatar (una imatge) mitjançant les expressions facials que són captades en un vídeo.

1.4. Organització de la memòria

En aquesta memòria expliquem les diverses fases que s'han seguit per a dur a terme les dues versions de *Face Transfer*. Ambdues versions segueixen la mateixa línea de processos, que es mostra en la figura 1.1. La diferència està en la fase de *mapping* de coordenades baricèntriques, primerament degut a que el mapeig és invers en una versió respecte l'altre, i també perquè cada versió conté mètodes diferents degut a les necessitats que proposa cada una.

El primer que s'explica en aquesta memòria és la detecció de punts facials (Capítol 2), i en concret una petita explicació de l'algoritme que fem servir i dels punts característics que utilitzem (Secció 2.1) per a crear les nostres regions facials aptes per al *mapping*. Abans de la creació de la màscara, hem volgut ampliar la nostre superfície fent el *fitting* d'una el·lipse a partir del punts predits per el algoritme de detecció de punts facials utilitzat (Secció 2.2).

En el Capítol 3, s'explica el *mapping* de textures a partir de la teoria de les coordenades baricèntriques (Secció 3.1), i com s'utilitza en cada versió de *Face Transfer* (Secció 3.2 i Secció 3.3).

Per tal de que el efecte de *Face Transfer* sigui més real, hem optat per aplicar el que en Processament d'Imatge és coneix com a *Blending* (Capítol 4) .

Finalment, en el Capítol 5 concloem amb els raonaments i reflexions extrets d'aquest, a més del treball futur.

Capítol 2

DETECCIÓ DE PUNTS FACIALS

Com hem explicat en el Capítol 1, per tal de realitzar el *mapping* entre la textura de les dues cares el nostre primer objectiu es dividir-les en regions equivalents. Per tal de fer això, primer necessitem trobar uns punts característics predefinits (ulls, celles, ...) que ens permetin definir-les i delimitar-les.

Les coordenades d'aquests punts facials ens permeten crear una *mesh* que representi la superfície de la nostra cara amb un conjunt de triangles òptim per al *mapping* de textures (Capítol 3).

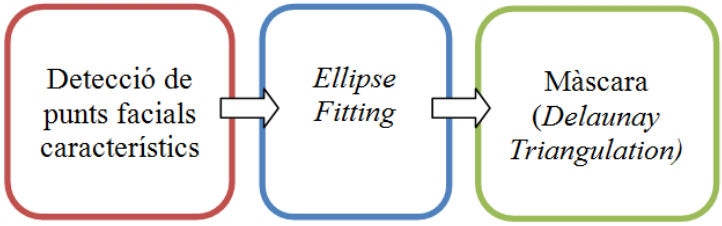


Figura 2.1. Esquema de la creació de la màscara.

El primer pas per dur a terme aquesta tasca és detectar punts característics d'una imatge origen (F_o) i d'una imatge destí (F_d), tals com els ulls, celles, i boca, amb els quals puguem crear aquesta *mesh* que representi la superfície de la nostra cara amb un conjunt de triangles òptim. És bàsic tenir una bona detecció d'aquest punts, degut a que la distància, geometria i disposició entre ells juguen un paper vital en el nostre propòsit, ja que a que si la detecció de punts és bona ens permetrà fer un bon mapeig i que el resultat sigui acceptable.

Per a detectar aquest punts, hem utilitzat l'algoritme proposat per Xiong, et al.[1], on es detecten 49 punts facials [10] (Fig. 2.2). Aquesta funció permet el seguiment (*tracking)* dels punts facials característics a temps real al llarg d'una seqüència de vídeo.

2.1 Descripció dels punts característics utilitzats

L'algoritme[1] proposat per Xiong et al. [10] ens proporciona els 49 punts facials que es poden observar a la Figura 2.2.

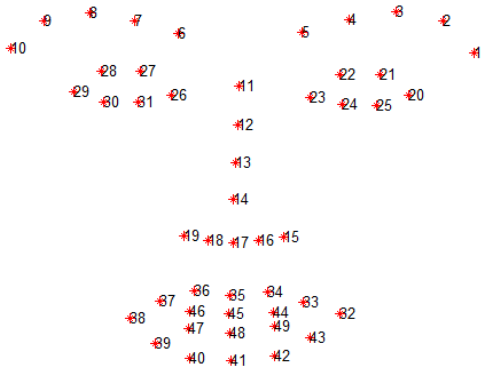


Figura 2.2. 49 punts facials detectats

El seu funcionament està basat en l'optimització de la localització dels punts d'interès a través de l'aplicació de Scale Invariant Feature Transform (SIFT) [6] i l'aprenentatge d'un model de forma no paramètric. Aquest model es composa de 2 regressors per a cada punt facial, que donat el descriptor global calculat amb SIFT (des d'una localització inicial) defineixen el desplaçament en x i en y per tal d'ajustar-se a les localitzacions reals.

Aquest mètode, està considerat l'estat de l'art tant en la detecció com en el seguiment de punts característics facials proveint-nos amb la localització d'aquests d'una manera acurada, robusta i en temps real.

Aquests 49 punts detectats representen la part on hi ha més informació de la cara; és a dir, són aptes per a codificar les expressions facials i serien útils per a classificar la emoció que s'està produint en una persona. Tot i així, aquests punts tenen l'inconvenient

[1] IntraFace: Supervised Descent Method and its Application to Face Alignment

de que només ens proporcionen una superfície que no és suficient per a que el *mapping* de textures doni un resultat acceptable Per aquest motiu, en el següent apartat (Secció 2.2) expliquem el procediment d''estimació d'una el·lipse que hem definit per tal de calcular punts addicionals amb l'objectiu d'ampliar la superfície de la màscara de regions.

2.2 Ampliació dels punts facials mitjançant l'estimació d'una el·lipse

Per a tenir una superfície més ample on fer *Face Transfer*, es necessari afegir punts addicionals per a tenir una malla més ample que mapejar. Com hem explicat inicialment, el detector de punts que hem escollit detecta 49 punts (celles, ulls, nas i boca). Aquest punts són punts interiors, i que ens creen una superfície molt limitada per a fer el *mapping* de textura.

És complicat predir punts que no siguin aquests, degut a que al exterior de la cara no tenim referències suficients per automatitzar una detecció de punts en aquesta zona. Una possible forma podria ser detecció de contorns o *Active Shape Model* i *Active Appearence Model* (ASM, AAM) [9] [19], de forma que podem detectar el contorn de la cara en cada frame. El problema sorgeix quan la cara on volem detectar aquest contorn hi ha barba o molt cabell; el contorn de la cara no esta ben definit i pot resultar difícil una predicció acurada dels punts. A més, cal afegir problemes que poden sorgir degut a la il·luminació del entorn, on el detector es pot confondre i variar la situació d'aquests punts, cosa que afectaria molt negativament el nostre *mapping*.

Per aquest motiu, hem tractat d'ampliar aquesta superfície tractant d'adaptar una el·lipse[2], de la millor manera sobre un conjunt de dades que, en aquest cas, són un

[2] Equació paramètrica de l'el·lipse (centrada en l'origen de coordenades):

$$x = a \cdot \cos(\alpha)$$
$$y = b \cdot \sin(\alpha)$$

on:

- x, y són les coordenades de la corba el·líptica
- a, b són els radis del l'el·lipse
- α és l'anomalia excèntrica de l'el·lipse; $\alpha \in [0,\ 2\pi)$

conjunt de punts facials. A partir d'aquesta el·lipse podem definir punts addicionals als característics que s'adaptin adequadament a cada *pose* de la cara.

2.2.1 Paràmetres de l'el·lipse

Com explica l'anterior apartat, és necessari un conjunt de dades per aproximar una figura geomètrica que millor descrigui aquest. El nostre conjunt C de dades consta dels punts detectats en els ulls i boca. Els punts dels ulls ens permeten centrar l'el·lipse correctament, i generalment ens descriu la major amplitud que pot tenir aquesta figura geomètrica. Els punts detectats en la boca ens son útils per a variar l'altura de l'el·lipse: quan obrim o tanquem la boca aquesta el·lipse tindrà més o menys altura, respectivament.

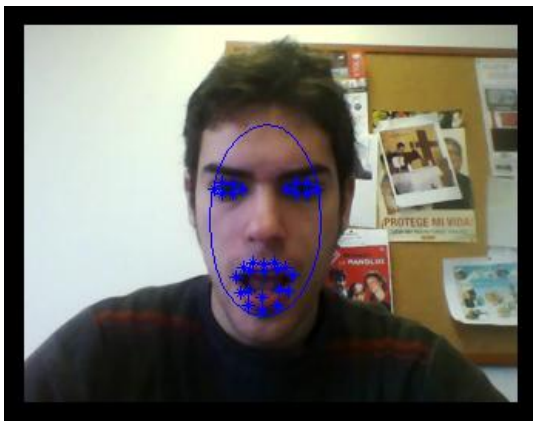


Figura 2.3. El·lipse adaptada sobre un conjunt de punts

Per a determinar el radi tant el radi com la posició de la el·lipse, utilitzem el que en àlgebra lineal s'anomena *Principal Component Analysis* (PCA) [5], que explicarem en el següent apartat (Secció 2.2.2).

2.2.2 Principal Component Analysis (PCA)

Per a calcular la magnitud i direcció dels radis de la nostra el·lipse donat un conjunt de dades C, fem servir PCA utilitzant el mètode de covariança. Aquest procediment estadístic consta d'una sèrie de passos que explicarem en aquest apartat.

El conjunt C de dades és compon d'una matriu $n \times p$. En el nostre cas, la matriu de dades és de $n \times 2$ (és a dir, $p = 2$), degut a que les dades (els conjunt de punts detectats que hem volgut fer servir) són en dos dimensions.

El primer pas és calcular la mitjana empírica de cada dimensió:

$$M[j] = \frac{1}{n} \sum_{i=1}^{n} C[i, j] \tag{2.1}$$

on :

p és el nombre de dimensions

n és el nombre de dades

$j = 1, \ldots, p$

$i = 1, \ldots, n$

El segon pas és centrar les dades a l'origen per a les p files del vector $\overline{C}$ de mitjanes, necessari per a calcular la covariança:

$$B = C - vM^T \tag{2.2}$$

on:

v és un vector unitari de n files

Després, calculem la matriu de covariança. La covariança ens indica la màxima variació entre dues variables aleatòries. Donat el nostre conjunt C, la covariància entre les variables x i y es defineix com:

$$A = \frac{1}{n-1} B^* \cdot B \tag{2.3}$$

on:

B^* és la matriu B transposada i conjugada [3]

Calculada la matriu de covariança A, ja podem estimar els valors i vectors propis aptes per a trobar les direccions de màxima variança. L'objectiu és trobar la matriu V que diagonalitza la matriu quadrada de covariança A:

$$AV = DV \tag{2.4}$$

on:

V és la matriu de vectors propis

D és la matriu diagonal de valors propis de A

[3] $B^* = (\overline{B})^T = \overline{B^T}$, on B^T és la matriu transposada i $\overline{B}$ és la matriu de conjugats complexes.

Els n valors de D que satisfan la equació són els nostres valors propis, i els valors corresponents a V són els nostres vectors propis. En el nostre cas, la matriu A és una matriu de 2-per-2 (Eq. 2.5), amb els valors de covariança (capítol 1.3.1) en x i y del conjunt C de punts predits. La matriu D és la matriu diagonal de vectors propis:

$$D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{2.5}$$

Un cop calculats valors i vectors propis, ja podem definir la magnitud de la nostra el·lipse. Definim els radis a i b de la nostra el·lipse a través de la següent fórmula:

$$a = 2\sqrt{\lambda_1} \qquad b = 2\sqrt{\lambda_2} \tag{2.6}$$

Per a una el·lipse centrada en el origen de coordenades on els seus radis segueixen la direcció dels eixos de coordenades, la nostra base (definida amb els vectors propis els quals són una representació de la direcció dels dos radis de l'el·lipse) serà la següent:

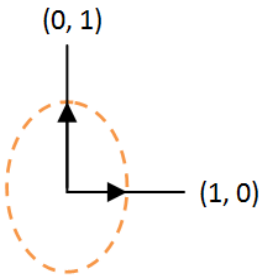


Figura 2.4. Base ortonormal el·lipse.

En canvi, sembla evident que si aquesta el·lipse pateix una transformació (com per exemple una rotació degut als moviments del cap), els dos radis d'aquesta el·lipse tindran un altre direcció. La següent figura és un exemple d'això, on és mostra la nova base ortonormal d'una nova disposició del núvol de punts del conjunt C, en aquest cas una rotació del cap :

Figura 2.5. Base el·lipse sobre el conjunt C de punts

Com indica la Figura 2.5, la base ortonormal (o ortogonal en cas que els vectors no siguin norma) obtinguda mitjançant PCA seran les direccions de màxima variància. Donada la distribució geomètrica dels punts de la cara, les dues direccions principals (que corresponen als eixos que ens defineixen les dades) correspondran als eixos que ens defineixen la llargada i l'amplitud de la cara.

Un cop tenim tots els paràmetres calculats de l'el·lipse que millor s'ajusta amb PCA, generem els punts necessaris a partir de l'equació paramètrica de l'el·lipse.

Capítol 3

MAPPING DE TEXTURES

Amb tot el conjunt de punts característics calculats amb el detector més els punts addicionals a partir d'ajustar una el·lipse, hem de crear un conjunt de regions apte per a mapejar amb coordenades baricèntriques (Fig. 3.1).

Aquestes regions les hem calculat incialment mitjançant l'algoritme de *Delaunay Triangulation* [11]. Aquest algoritme tracta de, donat un conjunt de punts, maximitzar l'angle més petit que pot formar un triangle, de forma que no quedin triangles prims que ens puguin ocasionar regions molt petites que per al *mapping* de baricèntriques resultin insignificants.

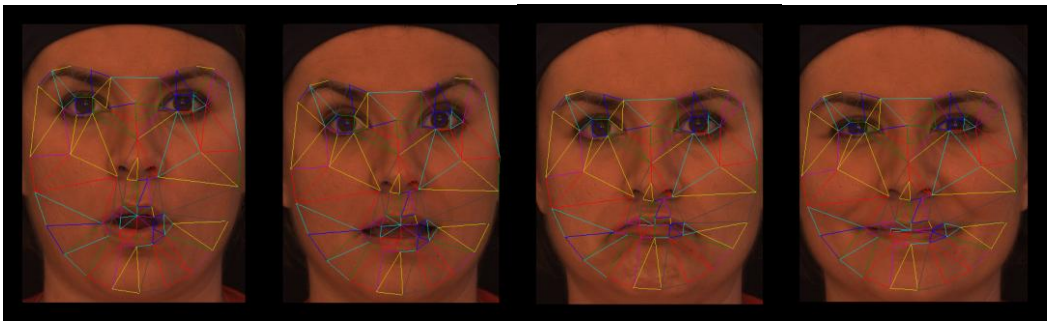


Figura 3.1 . Màscara final. Conté 49 punts facials característics,
2 punts calculats a partir d'una mitja i 13 punts amb *fitting* d'una el·lipse.

La triangulació de *Delaunay* es pot definir també com un conjunt de punts en 2D on, el conjunt de triangles format per aquests punts no conté un altre punt dins del cercle en el qual s'inscriu el triangle.

Un cop hem inicialitzat els triangles de la nostra màscara mitjançant *Delaunay*, és necessari retocar-los a mà per a que quedin simètrics. Això és degut a que en el moment de detectar els punts, no tots tenen una geometria simètrica, i el mapeig es pot veure afectat si no hi ha la mateixa disposició de triangles entre la imatge origen i la destí. Amb la màscara finalitzada, aquesta ja és vàlida per a qualsevol imatge facial donada qualsevol disposició de punts facials calculats.

Aquest capítol explica el *mapping* de textura que s'ha utilitzat per a fer el *Face Transfer*. Primerament, s'explica la teoria en la que es basa aquest mapeig. Després, s'explica les dues versions de *Face Transfer*: la primera, un vídeo animat a partir d'una imatge; la segona, una imatge "avatar" animada a partir de les expressions i gestos facials d'un vídeo.

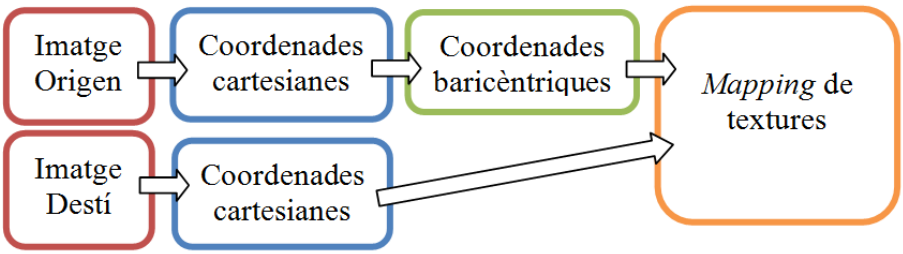


Figura 3.2. Esquema del *mapping* de textures.

La Figura 3.2 mostra el esquema que segueix el *mapping* de textures. Partim de dues imatges: una imatge origen i una imatge destí (que en el Capítol 2 hem anomenat F_o i F_d, respectivament). El primer pas és extreure les coordenades cartesianes de tots els punts que estan dins la màscara corresponent a cada imatge. Després, hem de convertir les coordenades cartesianes de la imatge origen a baricèntriques (Secció 3.1), per a poder dur a terme el *mapping* de textures entre la imatge (que expliquem en les Seccions 3.2 i 3.3).

Les coordenades baricèntriques ens són útils per a mapejar tota la informació que hi ha en una regió triangular i poder-la extrapolar a una altra superfície triangular sense que aquesta tingui la mateixa geometria. Per aquest motiu, si tenim dues malles triangulars, corresponents a les dues cares on volem aplicar *Face Transfer*, és possible aplicar una correspondència entre triangles amb la qual podem fer el *mapping* de la textura d'un triangle de F_o al seu corresponent en F_d .

3.1 Coordenades baricèntriques

Qualsevol punt P dins (o fora) d'un triangle o tetraedre és pot definir com a una combinació lineal dels seus vèrtexs. És a dir, per a un triangle amb vèrtexs ABC un

punt P és pot definir com a $P = uA + vB + wC$, on uvw són les coordenades baricèntriques.

És compleix que $u + v + w = 1$ quan les coordenades baricèntriques estan normalitzades. Si qualsevol de les coordenades uvw és major que 1 o menor que 0, vol dir que el punt P es troba fora del triangle ABC. Les coordenades baricèntriques uvw es poden entendre com les diferents subàrees que conformen un triangle a partir d'un punt P dins d'aquest i els seus vèrtexs (Fig. 3.3).

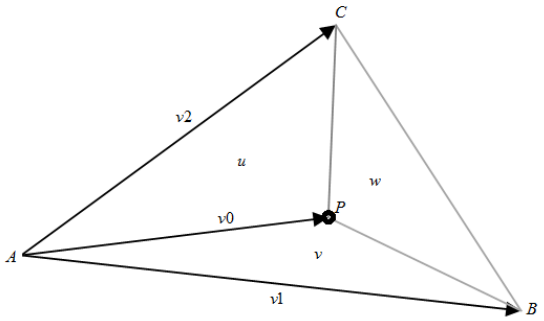


Figura 3.3. Coordenades baricèntriques d'un punt P.

Per aquest motiu, les coordenades baricèntriques són molt útils per a treballar en dominis subtriangulars. com s'explica a [12]. En el nostre cas, per omplir cada triangle de la malla destí amb els corresponents píxels de la malla origen, ja que amb coordenades cartesianes no seria possible fer el mapeig degut que les àrees dels triangles origen i destí són diferents.

Un cop tenim la malla de triangles de F_o i F_d, el següent pas és trobar les coordenades UV que es troben a dins de les malles. Aquestes coordenades UV són cartesianes, i depenent dins de quin triangle estigui se li assigna el identificador d'aquell triangle. Després toca convertir cada coordenada UV a coordenada baricèntrica uvw.

El primer pas és calcular els dos vectors que defineixen cada triangle. Aquests són $v1 = B - A$ i $v2 = C - A$, més el vector $v0 = P - A$. Després d'això, calcular l'àrea del triangle amb el producte vectorial:

$$Area_{total} = \frac{\| v1 \times v2 \|}{2} \tag{3.1}$$

I les diferents subàrees del triangle que conformen les coordenades baricèntriques:

$$u = \frac{Area_{ACP}}{Area_{total}}, \text{ on } Area_{ACP} = \frac{\| v0 \times v2 \|}{Area_{total}}$$

$$v = \frac{Area_{ABP}}{Area_{total}}, \text{ on } Area_{ABP} = \frac{\| v0 \times v1 \|}{Area_{total}}$$

$$w = 1 - Area_u - Area_v$$

$$P = uA + vB + wC$$

(3.2)

Un cop calculat totes les coordenades uvw de la nostra malla, creem una matriu amb les coordenades cartesianes UV, coordenades baricèntriques uvw, components RGB de cada píxel i identificador del triangle al que correspon cada coordenada.

3.2 Mapeig de la textura d'una imatge sobre una altre cara en moviment

El nostre objectiu és fer *Face Transfer* en un vídeo a partir d'una imatge origen. És a dir en una seqüència de *frames* $[F_{d1}, F_{d2}, F_{d3}, ..., F_{dN}]$ aplicarem *Face Transfer* a partir d'una imatge origen F_o.

El primer pas és extreure les coordenades baricèntriques (tal i com s'ha explicat en el apartat anterior) a la cara origen F_o, i poder fer el mapejat corresponent per a cada *frame* de F_d.

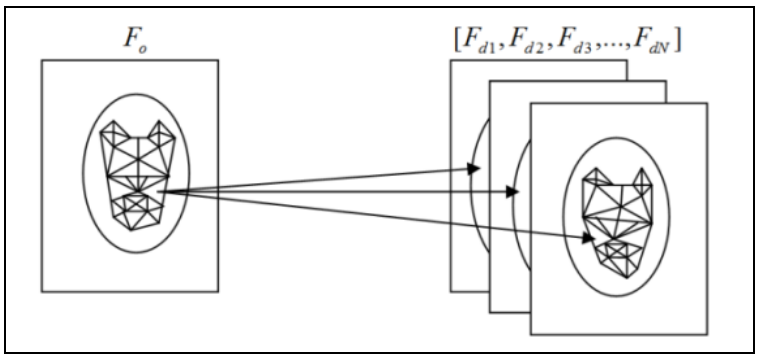


Figura 3.4. *Face Transfer* de F_o cap a $[F_{d1}, F_{d2}, F_{d3}, ..., F_{dN}]$

Per a fer el mappeig, es necessiten les coordenades baricèntriques extretes de F_o , amb les quals buscarem la seva correspondencia en F_d . És a dir per a un punt la imatge origen P_{F_o} amb baricèntriques $(u_{F_o}, v_{F_o}, w_{F_o})$, P_{F_d} es calcula de la seguent forma:

$$P_{F_d} = A_d \cdot u_{F_o} + B_d \cdot v_{F_o} + C_d \cdot w_{F_o} \tag{3.3}$$

on:

A_d, B_d, C_d són els vèrtex d'un triangle en F_d

En aquest punt P_{F_d} de la imatge ens interessa quedar-nos amb el valor RGB, que és el que substituirem en les coordenades de P_{F_o} .

La Figura 3.5 mostra el resultat del *mapping* de textures sobre diversos *frames* d'una seqüència de vídeo.

Figura 3.5. Resultat del mapping de textures. A la dreta, la imatge origen.

3.3 Animació d'un avatar

En aquest cas de *Face Transfer* es fa servir el mateix mètode però a la inversa. Ara, donat una seqüència de *frames* origen $[F_{o1}, F_{o2}, F_{o3}, ..., F_{oN}]$ i una imatge destí F_d , apliquem *Face Transfer* en la imatge destí. A més, ara no mapejem el color de F_o a F_d, si no que la textura serà de la pròpia imatge F_d, de forma que el resultat serà un avatar 2D animat amb les expressions facials del nostre vídeo original (que anomenarem F_r). Esdevé, a més, la necessitat d'alinear la màscara resultant amb la imatge avatar, degut a que

aquesta no té moviment, i la posició inicial de la màscara F_o s'ha de modificar per a que encaixi correctament amb la imatge avatar. El mètode que hem utilitzat s'explica en la Secció 3.3.1.

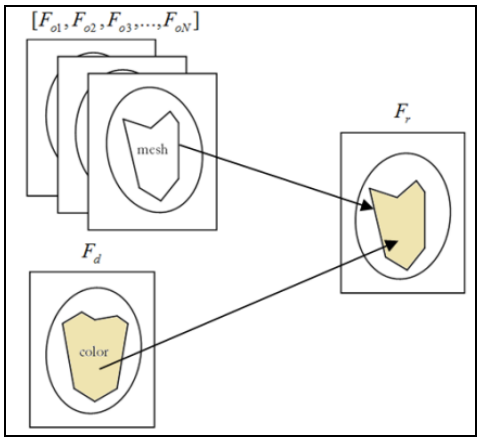


Figura 3.6. *Face Transfer* d'avatar.

Com es pot veure en la Figura 3.6, a partir dels *frames* $[F_{o1}, F_{o2}, F_{o3}, \ldots, F_{oN}]$ del nostre vídeo copiem l'estructura de la malla, i a partir de la imatge F_d mapejem el color dels píxels. Com a resultat, tenim una imatge F_r resultant, on l'expressió facial està donada per la malla de cada frame i el color dels píxels per la imatge estàtica.

Per a mapejar el color dels píxels, el mètode segueix sent el mateix. Un punt P_{F_r} dins la malla es calcularà de la següent forma:

$$P_{F_r} = A_r \cdot u_{F_d} + B_r \cdot v_{F_d} + C_r \cdot w_{F_d} \tag{3.4}$$

Per a calcular la posició de la malla en la imatge resultant, utilitzem anàlisis per Procrustes, que explicarem en el següent apartat.

3.3.1 Anàlisi de Procrustes

L'expressió facial ve donada per la forma que té la malla en cada *frame*. Es necessari fer una bona alineació de la malla per a que la imatge resultant quedi creïble. El mètode que fem servir s'anomena anàlisi de Procrustes [17], i es tracta d'un mètode utilitzat en estadística per a comparar una distribució de formes, núvol de punts... amb el objectiu de trobar la millor correspondència entre ells.

L'anàlisi de Procrustes és un mètode de mínims quadrats, això vol dir que és un mètode que tracta de reduir al mínim una expressió matemàtica, en aquest cas tracta de reduir al mínim el sumatori de les desviacions al quadrat entre dos núvols de punts o figures, mitjançant transformacions d'escala, rotació i translació.

En el nostre cas, volem trobar la millor alineació entre dos grups de núvols de punts, concretament entre la malla de cada frame F_o i la malla de F_d . Aquest núvol de punts no son els 49 punts predits inicialment, si no 11 punts, que es tracten dels punts més rígids d'una cara. S'entén per punts rígids els punts que normalment pateixen menys desviació a l'hora de fer expressions facials respecte una cara amb expressió neutra. En la imatge es poden apreciar els 11 punts escollits per a l'anàlisi de Procrustes:

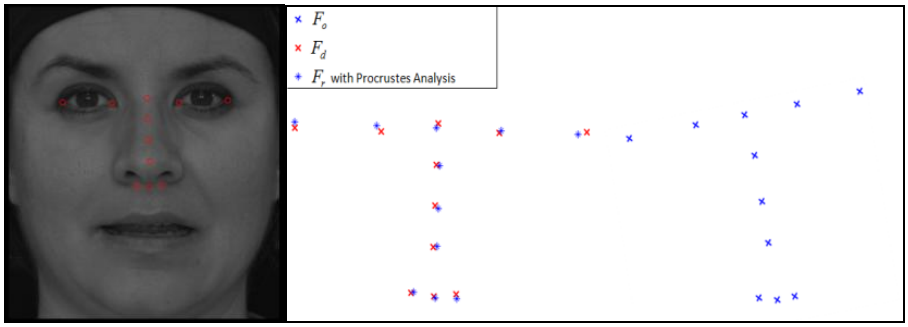


Figura 3.7. Anàlisi de Procustres.

El motiu d'utilitzar el mètode de Procrustes en el mètode de *Face Transfer* (Vídeo a Imatge) la Secció 3.3 es deu al problema centrar correctament la *mesh* triangular de cada *frame* del nostre vídeo la imatge destí (estàtica). Per aquest motiu s'han escollit els punts rígids de cada *mesh* F_d *i* F_o , ja que ens assegura que, al tenir poca desviació, el error minimitzat (Eq. 3.5) entre els punts escollits les dues malles serà més petit que si comparem el total de punts predits.

L'anàlisi de Procrustes tracta de resoldre un problema d'aproximació matricial. Donades dues matrius A i B, volem trobar la matriu P que apropi més la matriu A a B. El objectiu és minimitzar l'Eq. 3.5 a partir d'estimar la matriu de rotació Ω i translació τ:

$$\mathrm{P} = \arg\min_{\Omega,\tau} \| \mathrm{A}\Omega + \tau - \mathrm{B} \| \qquad \textit{subject to } \Omega^T\Omega = \mathrm{I} \tag{3.5}$$

Un cop trobada la transformació que minimitza més el error o desviació entre punts, s'aplica aquesta transformació al total de punts predits en F_o, de manera que tindrem una *mesh* resultant. Finalment, l'últim pas és aplicar el mètode de mapejat de color per baricèntriques explicat en la Secció 3.3 de F_d a F_r.

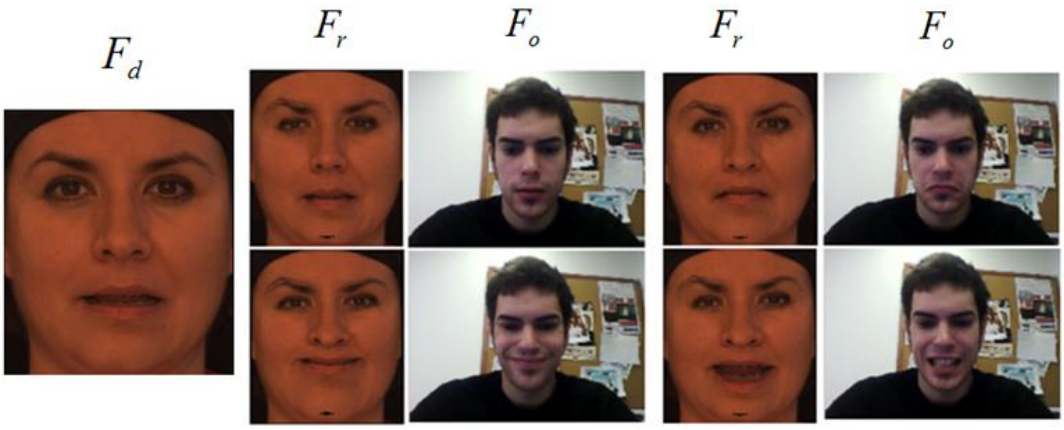


Figura 3.8. Resultat de l'animació d'un avatar amb *Face Transfer*
amb diferents expressions facials.

Capítol 4

MULTIBAND BLENDING

Amb el efecte de suavitzar el canvi de color que hi pot haver entre el color la nostra pell i el color de la imatge en la que li volem fer Face Transfer , hem utilitzat un mètode que s'anomena *Multiband Blending* [4] [Burt i Adelson, 1983]. Aquest mètode tracta de modificar els nivells de color prop dels límits entre dues imatges per obtenir una transició suau entre aquestes.

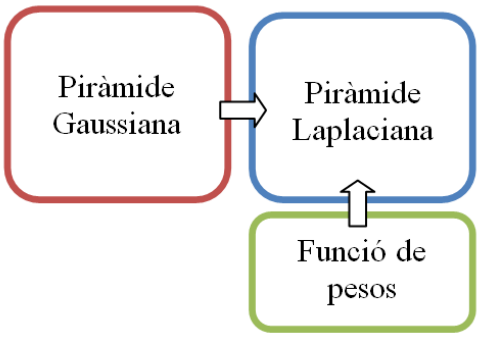


Figura 4.1. Esquema de les fases de *Multiband Blending*

El mètode de *Multiband blending* (Fig. 4.1) utilitza una funció de pesos (Secció 4.1) sobre una imatge Laplaciana de les dues imatges (Secció 4.3) amb la funció de mitigar la transició entre elles. Aquest mètode tracta de mantenir les zones homogènies de color entre dues imatges (aquestes corresponen a les freqüències baixes). En les freqüències altes (que contenen els detalls i contorns de les imatges), és necessari aplicar un suavitzat prop dels eixos per a que el salt entre les dues imatges no es noti brusc, d'aquí el motiu de dividir la imatge en "bandes múltiples" de freqüències a partir del que és conegut com a piràmide Gaussiana (Secció 4.2) i Laplaciana d'una imatge.

4.1 Funció de pesos

Per aplicar *Multiband Blending* hem de partir de dues imatges. En el nostre cas una de les imatges és la *mesh* on apliquem el color per baricèntriques i l'altre és la imatge destí F_d. Necessitem una funció de pes (*weighting funtion*) multiplicada a cada imatge per a

crear aquest efecte. Aquesta funció de pesos (Fig. 4.2) convergeix en un mateix punt, en el límit entre la *mesh* i F_d, on cada imatge té un 50% de del pes. A mida que ens distanciem del límit on s'ajunten les dues imatges, una de les dues adquireix major pes que l'altre, de manera que estem donant més importància a la que té major pes.

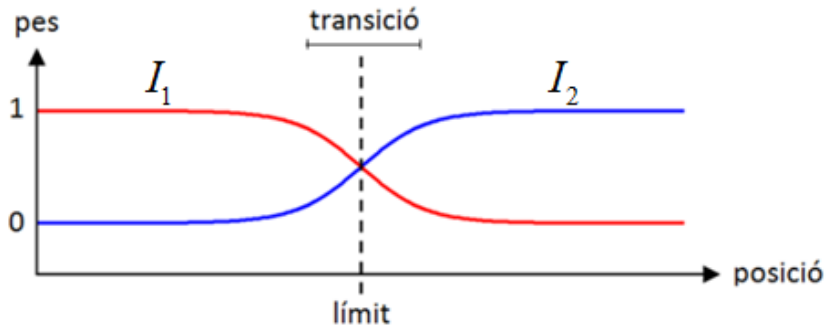


Figura 4.2. Funció de pesos de les imatges I_1 i I_2

Veient el gràfic (Fig. 4.2), I_1 representaria la *mesh* i I_2 representaria cada frame del nostre vídeo F_d, per exemple. A l'esquerra del límit I_1 té màxima importància i per tant ve multiplicada per el màxim factor de pes. La zona de transició es la zona on els pesos varien de 0 a 1 i al revés. A mida que ens apropem al límit, a la zona de transició entre una imatge i l'altre, la importància d'I_2 va creixent al augmentar gradualment el seu pes, mentre que alhora decreix el pes en I_1. Ja en el límit, les dues imatges tenen el mateix pes assignat (0.5 en pes normalitzat).

La mida en la que establim la transició juga també un paper important en el efecte de suavitzat. És la zona on s'aplica el *blending*. Si és massa petita, el límit que denota la separació entre la *mesh* i la imatge on es fa *Face Transfer* es veurà massa brusc. Per el contrari, si la transició és massa allargada, pot ocórrer que es vegin les característiques d'ambdues imatges en les dues bandes, és a dir, les característiques d'I_1 en I_2 i al revés. Per aquest motiu s'ha d'escollir una mida òptima de la finestra de transició de manera que minimitzi el efecte provocat per els dos casos anteriors.

Com estem parlant d'una transició en imatge, és a dir una matriu 2D, el mètode que hem escollit és crear una matriu de pesos també en 2D. Aprofitant la *mesh* de punts detectats

en la nostra cara, podem definir una màscara on els píxels que cauen dins d'aquesta tinguin pes 0 (negre), mentre que els que estan a fora tinguin pes 1 (blanc).

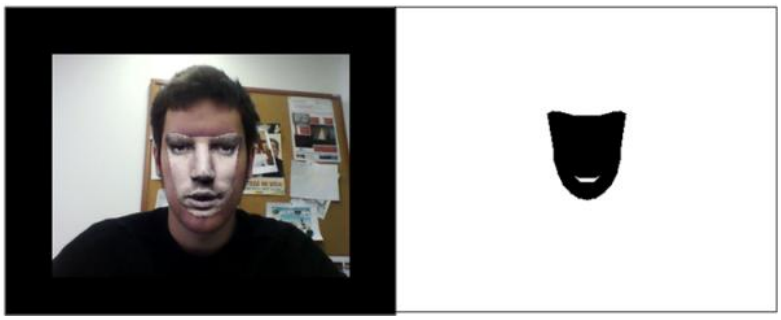


Figura 4.3. A l'esquerra, *Face Transfer* sense *Multiband Blending*.
A la dreta, màscara de pesos 2D del actual *frame*.

Un cop tenim la màscara, necessitem crear una transició suau entre el màxim i mínim pes assignat. Per fer-ho, filtrem la imatge per aplicar *blur* a la màscara, de manera que tinguem una transició molt suau on inicialment teníem el salt de 1 a 0, tal i com es veu en la següent imatge on s'ha aplicat *blur* a la màscara de la Figura 4.3:

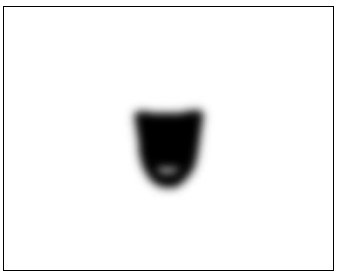


Figura 4.4. *Blur* en la màscara depesos 2D de la Figura 4.3.

A la nostra màscara de pesos 2D (Fig. 4.4) se li ha aplicat un *blur* al llarg de tota la superfície, i no sol als contorns d'aquesta. El motiu de fer això és mitigar les transicions entre il·luminacions que puguin haver-hi entre les dues imatges superposades. A més, si apliquem *blur* a tota la màscara, ens assegurem que el resultat final tindrà un to de pell més uniforme al llarg de la superfície.

Finalment, tal i com s'aprecia en les Figures 4.3 i 4.4, hem optat per deixar les regions de la màscara que pertanyen al interior de la boca com a part de la imatge que pertany a la seqüència de vídeo. Amb això obtenim un major realisme al obrir i tancar la boca ja que és veuran les nostres dents, llengua, etc. Si no ho féssim, es possible que es produïssin distorsions en el cas de que la imatge origen (el nostre avatar) tingui la boca

tancada, i el mapeig d'aquesta zona quedaria defectuós, ja que no hi hauria textura en el interior de la boca per a fer el *mapping* correctament.

4.2 Piràmide Gaussiana

La transició explicitada en el apartat anterior es modela amb el que es coneix com a Piràmide Gaussiana, molt utilitzat en Processament d'Imatge. Es tracta de descompondre la imatge a partir d'una sèrie de convolucions amb filtres Gaussians (normalment de 5x5) (Eq. 4.3), de forma que obtindrem un set de imatges filtrades en passa baix. Cada sub-image, es calcula amb la següent equació:

$$G_l = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(n,m) \cdot G_{l-1}(i-m, j-n) \tag{4.1}$$

on:

G_l és la imatge actual

G_{l-1} és la imatge anterior

w és el filtre Gaussià

El filtre de Gauss es representa amb el següent gràfic:

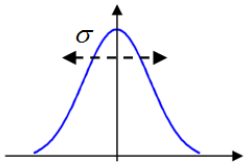


Figura 4.5. Filtre de Gauss.

I la seva equació és la següent:

$$w(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}} \tag{4.2}$$

on:

σ és la desviació estàndard de la gaussiana

Per a dos dimensions (en el cas d'una imatge), l'equació del filtre de Gauss és el producte de dos Gaussianes d'una dimensió:

$$w(x,y) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma}} \cdot \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{y^2}{2\sigma}} = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

Després de cada convolució, la mida imatge resultant és redueix a la meitat respecte el nivell anterior utilitzant submostreig (*downsampling*) de factor 2. Això vol dir que per a cada imatge s'elimina un píxel de cada dos, reduint la imatge en un factor de 1/2. El motiu per el qual es submostreixa havent aplicat un filtre pas baix previ (filtre Gaussià) és per evitar el efecte d'*aliasing* (teorema de Nyquist-Shannon, Eq. 4.4) en la imatge, que implica mostreixar amb una freqüència f_m com a mínim major que el doble de la freqüència màxima del senyal ($f_{\max}$), això és:

$$f_m \geq 2 \cdot f_{\max} \tag{4.4}$$

Convolucionant abans per una Gaussiana ens assegurem d'eliminar les freqüències altes i que al fer el submostreig no ens aparegui l'efecte d'*aliasing*.

Tenim doncs, una seqüència d'imatges $G_o, G_1 \ldots G_N$, on G_o és la imatge original, que compondran la piràmide d'N nivells.

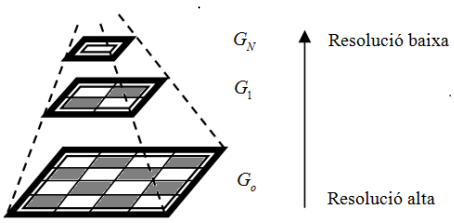


Figura 4.6. Piràmide Gaussiana d'N nivells.
A mida que puja el nivell la imatge té menys resolució.

Com es pot veure en la Figura 4.6, a mida que els nivells de la piràmide són més alts la imatge té menys resolució (més *blur*), ja que s'està aplicant un filtre Gaussià amb major desviació estàndard seguit d'un submostreig en factor 2.

4.3 Piràmide Laplaciana

Un cop construïda la piràmide Gaussiana de les imatges, el següent pas és invertir el procediment anterior, és a dir, expandir la piràmide des de el nivell mes baix (G_N). El objectiu és construir el que s'anomena piràmide Laplaciana d'imatges. En aquesta piràmide s'emmagatzemen els detalls de la imatge; freqüències altes que defineixen els

contorns de la imatge, al contrari de la piràmide Gaussiana, on apliquem *blur* a cada nivell per tenir les freqüències baixes a cada nivell.

El procediment per obtenir aquestes imatges és el següent: es parteix de l'últim nivell de la piràmide Gaussiana G_N, aquesta imatge s'expandeix per, acte seguit, restar-la amb el nivell anterior G_{N-1}, amb el qual obtenim la imatge del nivell L_{N-1} de la nostra piràmide Laplaciana. La imatge L_N prové de la pròpia imatge del nivell G_N degut a que és l'últim nivell de la piràmide. És fa el mateix procediment amb la imatge G_{N-1}, i així fins arribar al nivell 0, on tenim la imatge original (sense filtrat ni *downsampling*). La diferència d'imatges entre cada nivell forma la piràmide de imatges Laplaciana (Fig. 4.7).

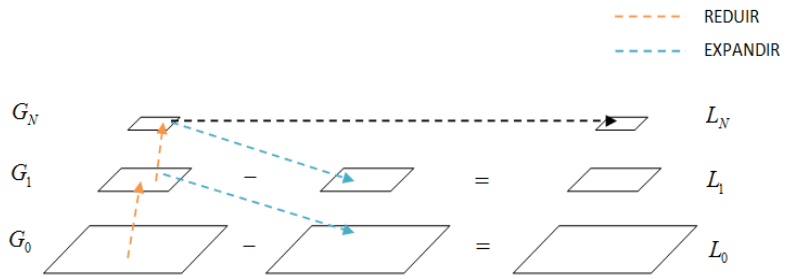


Figura 4.7. Procediment per a obtenir la Piràmide Laplaciana.

Com partim de la imatge més petita, és a dir, la que té menys mostres, i volem restar la diferencia amb el nivell anterior (que en té més) és necessari un sobremostreig (*upsamping*) d'aquesta imatge. Això és, bàsicament, el procés d'EXPANDIR (Eq. 4.5), mentre que el procés de REDUIR (Eq. 4.1) és el anteriorment explicat en la Secció 4.2: filtre Gaussià + *downsampling*.

El sobremostreig que s'ha d'aplicar a cada imatge de la matriu Gaussiana és de factor 2, ja que és necessari doblar el nombre de mostres per a poder obtenir la diferència. Fer un sobremostreig implica crear noves mostres (píxels en la imatge) que no existeixen i, per tant, s'ha de fer una interpolació dels píxels més propers en aquella mostra que inicialment no tenim.

Cada imatge del procés EXPANDIR (Eq. 4.5) s'obté mitjançant la següent formula:

$$G_l^{'} = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} G_{l-1}\left(\frac{2i+m}{2}, \frac{2j+n}{2}\right) \tag{4.5}$$

Finalment, es calcula la diferència (Eq. 4.6) entre nivells equivalents per construir la piràmide Laplaciana.

$$L_l = G_l - G_l^{'} \tag{4.6}$$

4.4 Resultat final

Arribat a aquest punt, tenim dues piràmides Laplacianes: LA i LB, que corresponen a les dues imatges a les que volem aplicar *blending*. Com es veu en la figura següent, cada conjunt d'imatges LA i LB és parteixen per la meitat, mentre que en el seu centre és fa el *blending* mitjançant la funció M, que en aquest cas és la màscara que creem per a cada *frame*, explicat en la Secció 4.1. Els píxels que estiguin sobre el eix que delimita el canvi entre les dues imatges en cada nivell rebran un 50% de pes, i com més s'allunyin d'aquest rebran un pes menys significatiu respecte a l'altre imatge.

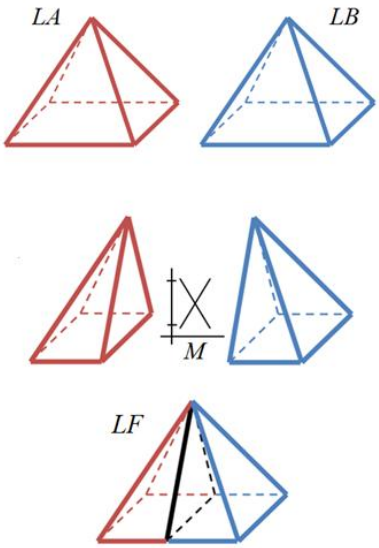


Figura 4.8. *Blending* de les Piràmides LA i LB.

La piràmide resultant LF (Eq. 4.7), que és el *blending* dels dos conjunts d'imatges en cada nivell, s'expressa amb la següent equació:

$$LF_l(i,j) = M_l(i,j) \cdot LA(i,j) + (1 - M_l(i,j)) \cdot LB_l(i,j) \tag{4.7}$$

Finalment, només queda expandir (Eq. 4.8) i sumar (Eq. 4.9) consecutivament cada nivell de la piràmide Laplaciana LF fins arribar al nivell base (on la imatge te la mida original) per a obtenir la imatge resultant (Fig. 4.9):

$$LF_l^{'} = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} LF_{l-1}\left(\frac{2i+m}{2}, \frac{2j+n}{2}\right) \tag{4.8}$$

$$LF_l = LF_l + LF_l^{'} \tag{4.9}$$

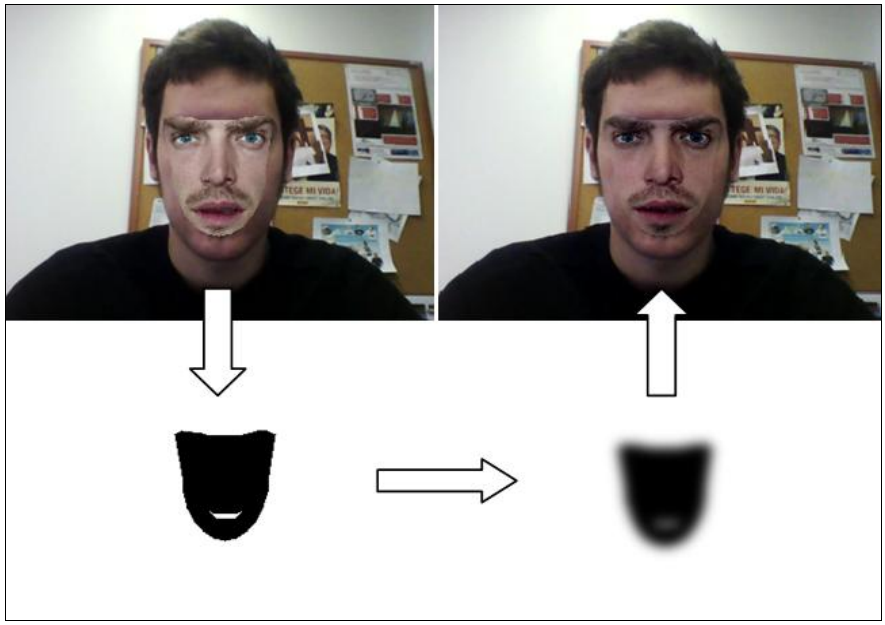


Figura 4.9. Resultat final del *Blending*.
Comparació sense haver aplicat la màscara de pesos 2D a la piràmide Laplaciana (esquerra) i havent-la aplicat (dreta).

Capítol 5

RESULTATS EXPERIMENTALS

En aquest Capítol exposem els resultats, explicant les seves fortaleses i també els seus errors. Com es pot veure, els resultats de la primera versió mostren que el nostre mètode de *Face Transfer* és adaptable a qualsevol imatge facial. A més, si ens fixem en les imatges avatar, no fa falta que la cara expressi una emoció neutre per al *mapping* de textures, si no que podem partir de qualsevol imatge amb qualsevol emoció i deformar la seva textura a partir d'aquesta com ens convingui amb un resultat acceptable.

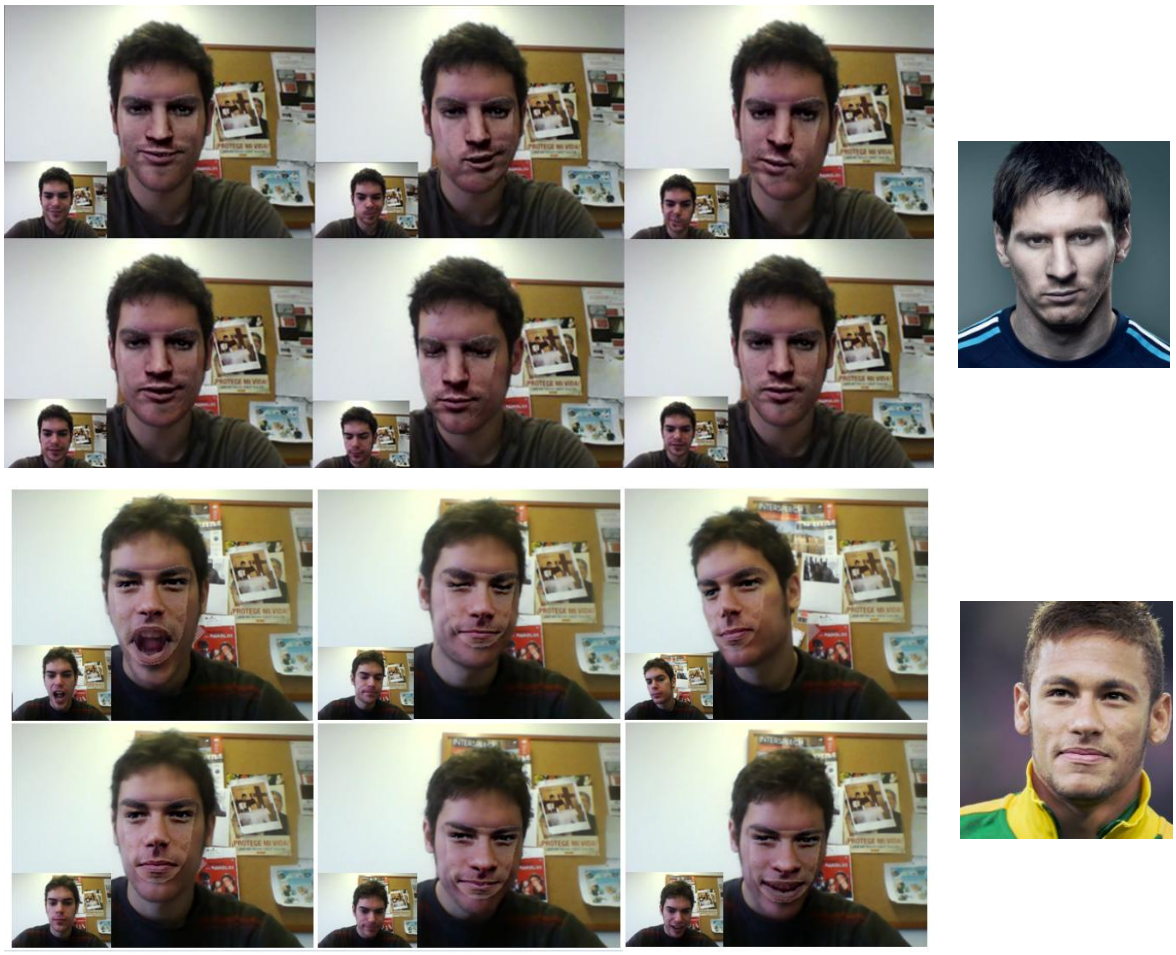


Figura 5.1. Resultats del *mapping* de textures sobre una seqüència de vídeo.

En el la segona versió de *Face Transfer*, on animem directament l'avatar a partir de les expressions facials captades en un vídeo, és pot veure com no hem aconseguit tant dinamisme en com en la primera versió. Això es degut a que la cara que volem animar

és estàtica, i nomes variem la seva expressió facial. Aquesta poca flexibilitat que ofereix ens obliga a més a trobar la millor alineació de la nostra malla amb la imatge avatar. Per a poder encaixar la malla de cada *frame* de la seqüència amb la posició de la cara avatar, hem fet servir l'anàlisi de Procrustres sobre una sèrie de punts (que hem denominat rígids degut a que pateixen poca desviació en el moviment de la cara) .

Figura 5.2. Resultats del *mapping* de textures sobre un avatar.

En l'última imatge, s'observen diversos casos d'errors. En la primera i segona imatge (començant per l'esquerra), veiem una deformació de la malla com a conseqüència d'una mala predicció de punts. En el primer cas aquesta deformació és deu a que la il·luminació confon el predictor de punts facials i com a conseqüència els punts addicionals també estan mal calculats. En el segon cas es degut a que el predictor detecta una possible *feature* en el escenari que no és. En la tercera imatge, es pot veure un error en el càlcul de la malla triangular, i com a conseqüència, un mapeig de la textura erroni. Una possible forma de solucionar aquest error seria no mapejar la textura de l'interior de la boca, igual que hem fet amb la primera versió.

Figura 5.3. Errors del *mapping* de textures.

Capítol 6

CONCLUSIONS

Durant aquest projecte hem comprovat les nombroses aplicacions i mètodes que han sorgit durant els últims anys per al reconeixement de cares i punts facials. Específicament hem concebut e implementat un mètode de "Face Transfer" (Transferència de Cares).

El nostre mètode, al igual que la majoria, té els seus avantatges i també les seves limitacions. Hem comprovat que és un mètode que, tenint en compte que parteix d'una sola imatge per a fer el mapeig i sense cap entrenament previ d'un model, dona un resultat força real inclús per a punts de vista que no estan contemplats en una imatge 2D. Per exemple, al rotar la cara horitzontal o verticalment en una seqüència de vídeo (aquestes rotacions poden ser tant dintre com fora del pla xy), obtenim un bon resultat en el *mapping* de textures, tot i que la imatge origen de la que partim mostri una *pose* de la cara frontal. Gràcies a això, el nostre mètode de *Face Transfer* és capaç d'animar una seqüència de vídeo partint de qualsevol imatge facial. Hem vist mètodes que apliquen *Face Transfer* amb un entrenament previ de models. Per exemple, a [2] i [18] s'utilitza un model d'entrenament per a generar un set d'expressions prototípiques, i calcular la forma de l'avatar a partir d'una deformació d'aquestes.

Respecte a les seves limitacions, s'han de tenir en compte possibles errors en oclusions o en entorns on la il·luminació no és constant, que poden donar peu a detecció de punts erronis i com a conseqüència la creació d'una malla triangular defectuosa per al *mapping* de textures. També cal comentar, sobre el mètode de *fitting* d'una el·lipse per a afegir punts addicionals, que tot i que és un mètode computacionalment ràpid i dona resultats més que acceptables, és evident que hi han mètodes més exactes. El càlcul de punts per *Appearence Model*, explicat a [9] o a partir de calcular els punts amb un model geomètric, mitjançant *Shape Model*, com es fa en [19] detecten el contorn de la cara amb major exactitud. Tot i així, aquest mètodes són sensibles a canvis d'il·luminació i poc exactes per a textures que no estiguin ben definides en els contorns de la cara, com pot ser barba, cabell...

En quant a la versió 2 de *Face Transfer* que hem implementat, on variem les expressions facials d'un avatar a partir dels moviments captats en una seqüència de vídeo, té una limitació evident: les expressions captades han de contenir una *pose* semblant a la de la imatge on volem transferir-les. Això vol dir, que si la imatge avatar és frontal, també ho haurà de ser la *pose* de la nostra cara al llarg de la seqüència de vídeo.

Per aquest motiu, com a treball futur, proposem crear un model d'avatar amb diferents *poses* i/o expressions facials. D'aquesta manera, seria possible escollir a cada moment quina posició i expressió és la més adequada per a mapejar a partir d'una situació similar a la nostra cara.

Bibliografia

[1] Yanlin Weng, Chen Cao, Qiming Hou and Kun Zhou. *Real-time Facial Animation on Mobile Devices*. Graphical Models, Elsevier, 2013.

[2] Jason M. Saragih, Simon Lucey and Jeffrey F. Cohn. *Real-time Avatar Animation from a Single Image*. Automatic Face and Gesture Recognition and Workshops, pages 117-124. IEEE, 2011.

[3] M. Saragih, Simon Lucey and Jeffrey F. Cohn. *Face Alignment through Subspace Constrained Mean-Shift*. Computer Vision, pages 1034-1041. IEEE, 2009.

[4] Peter J. Burt and Edward H. Adelson. A Multiresolution Spline *With Application to Image Mosaics*. ACM Transactions on Graphics (TOG), Vol. 2, Num. 4, pages 217-236. ACM, 1983.

[5] I.T. Jollife. *Principal Component Analysis*. Springer-Verlag, 1986.

[6] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, Vol. 60, Num. 2, pages 91-110. Springer, 2004.

[7] Paul Viola and Michael J. Jones. *Robust Real-Time Face Detection*. International Journal of Computer Vision, Vol. 57, Num. 2, pages 137-154. Springer, 2004.

[8] Michel Valstar, Brais Martinez, Xavier Binefa and Maja Pantic. *Facial Point Detection using Boosted Regression and Graph Models*. Computer Vision and Pattern Recognition (CVPR), pages 2729-2736. IEEE, 2010.

[9] I. Matthews and S. Baker. *Active appearance models revisited*. International Journal of Computer Visio*n*, Vol. 60, Num. 2 pages 135-164. 2004.

[10] M. Nguyen, J. Perez and F. D. la Torre. *Facial feature detection with optimal pixel reduction svm*. Proc. IEEE Int'l conf. on Automatic Face and Gesture Recognition, pages 1–6, 2008.

[11] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppey, Michael Lounsberyz and Werner Stuetzle. Multiresolution *Analysis of Arbitrary Meshes*. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pages 173-182. ACM, 1995. (DT)

[12] O. Weber, M. Ben-Chen, C. Gotsman, K. Hormann. *A Complex View of Barycentric Mappings*. Computer Graphics Forum, Vol. 30. Num. 5, pages 1533-1542. 2011.

[13] Ekman, Paul and Friesen, Wallace V. *Constants across cultures in the face and emotion.* Journal of personality and social psychology, 1971.

[14] Charles Darwin. *The Expression of the Emotions in Man and Animals.* Oxford University Press, 1998.

[15] T. Cootes, G. Edwards, and C. Taylor. *Active Appearance Models*. Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, Num. 6, pages 681-685. IEEE, 2001.

[16] T. Wu, M. Bartlett, and J. Movellan. *Facial Expression Recognition using Gabor Motion Energy Filters. In* Computer Vision and Pattern Recognition Workshops, pages 42-47. IEEE, 2010.

[17] John C. Gower, Garmt B. Dijksterhuis, *Procrustes Problems*, Oxford University Press, 2004.

[18] Chen Cao, Yanlin Weng, Stephen Lin, Kun Zhou. *3D Shape Regression for Real-Time Facial Animation*. ACM TOG, Proc. SIGGRAPH, Vol. 32, Num. 4, page 41. 2013.

[19] I. Kostia, I. Pitas. *Real Time Facial Expression Recognition from Image Sequences using Suport Vector Machines*. Vol. 2, pages 966. IEEE, 2005.