

UNIVERSITÀ DEGLI STUDI DI VERONA

Facoltà di Scienze MM.FF.NN.

Corso di Laurea Specialistica in Sistemi Intelligenti e Multimediali

Tesi di Laurea Specialistica

## **Active Appearance Models**

Aspetti Metodologici e Casi di Studio

**Relatore:**

Umberto Castellani

**Candidato:**

Luca Vezzaro

15 Dicembre 2010



# Riconoscimenti

Questo lavoro non sarebbe stato possibile senza il contributo di Simon Baker<sup>1</sup> e Jing Xiao<sup>2</sup>, che hanno avuto la pazienza e la disponibilità di aiutarmi nell'approfondimento di alcuni aspetti dei loro algoritmi. Il Dr. Xiao, inoltre, ha gentilmente reso disponibile la sua implementazione dell'algoritmo di ricostruzione della forma dal movimento, che è stata usata come benchmark e fonte di ispirazione per le parti più impegnative.

Anche Brian Amberg<sup>3</sup> ha contribuito al successo di questo lavoro, aiutandomi a comprendere alcuni dettagli sul funzionamento della sua implementazione.

---

<sup>1</sup>Microsoft Research (sbaker@microsoft.com)

<sup>2</sup>Epson Research (jing.xiaoj@gmail.com)

<sup>3</sup>University of Basel (brian.amberg@unibas.ch)



# Sommario

Con il termine Active Appearance Model (AAM) si identifica un modello generativo lineare di oggetti deformabili, inizialmente introdotto da Cootes et al. nel 1998. Un AAM fornisce una rappresentazione compatta e altamente rappresentativa di oggetti deformabili e permettere di determinare in modo efficiente i parametri del modello a partire da una generica istanza di un oggetto.

La semplicità concettuale e efficienza degli Active Appearance Models ha reso questi molto popolari, e ha portato negli anni a un continuo lavoro di estensione e miglioramento della formula originale. Sviluppi che, assieme al costante aumento delle prestazioni dell'hardware, ha reso ora possibile il riconoscimento di oggetti deformabili in tempo reale.

Questa tesi fornisce innanzitutto un'introduzione ai fondamenti teorici degli Active Appearance Models, partendo dalla versione originalmente sviluppata da Cootes et al. per poi passare alle più recenti estensioni alla composizione inversa e al riconoscimento 2D+3D.

In seguito si osserva, attraverso alcuni casi di studio, come nessuna delle attuali implementazioni liberamente disponibili di AAM sia allo stesso tempo: portatile, capace di prestazioni real-time e sufficientemente flessibile da essere usata in molteplici applicazioni.

Si procede quindi ad esporre come una simile implementazione sia possibile, delineando in modo passo-passo il processo che ha portato allo sviluppo di un'implementazione completa di Active Appearance Models in tempo reale con ricostruzione simultanea di forma 2D e 3D.



# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Sintesi dei Contenuti . . . . .	2
1.2	Notazione . . . . .	3
<b>2</b>	<b>Active Appearance Models Tradizionali</b>	<b>5</b>
2.1	Acquisizione e Modellazione della Forma . . . . .	6
2.1.1	Allineamento delle Forme . . . . .	7
2.1.2	Modello Statistico di Forma . . . . .	9
2.2	Modellazione dell'Apparenza . . . . .	10
2.2.1	Warp di Immagini . . . . .	11
2.2.2	Modello Statistico di Apparenza . . . . .	12
2.3	Modello Combinato . . . . .	15
2.4	Il Processo di Fitting . . . . .	17
<b>3</b>	<b>Active Appearance Models Rivisitati</b>	<b>19</b>
3.1	Introduzione alla Composizione Inversa . . . . .	20
3.1.1	L'Algoritmo di Lucas-Kanade . . . . .	20
3.1.2	Allineamento di Immagini con Composizione in Avanti . . . . .	22
3.1.3	Allineamento di Immagini con Composizione Inversa . . . . .	22
3.2	Operazioni sui Warp . . . . .	23
3.2.1	Calcolo della Jacobiana . . . . .	24
3.2.2	Inversione . . . . .	26

3.2.3	Composizione . . . . .	26
3.2.4	Estensione alle Trasformazioni Globali . . . . .	27
3.3	Soluzione in Forma Chiusa . . . . .	28
3.3.1	Proiezione dell'Errore Immagine . . . . .	28
3.3.2	Le Immagini di Steepest Descent . . . . .	30
3.3.3	Soluzione Incrementale . . . . .	31
3.3.4	Inversione del Warp Incrementale . . . . .	32
3.3.5	Composizione con il Warp Corrente . . . . .	32
3.4	L'Algoritmo Completo . . . . .	32
<b>4</b>	<b>Active Appearance Models 2D+3D in Tempo Reale</b>	<b>35</b>
4.1	Il Modello di Forma 3D . . . . .	36
4.1.1	Definizione . . . . .	36
4.1.2	Equivalenza tra Modelli di Forma 3D e 2D . . . . .	37
4.2	Ricostruzione della Forma dal Movimento . . . . .	39
4.2.1	Fattorizzazione della Matrice delle Misure . . . . .	40
4.2.2	Vincoli di Rotazione . . . . .	41
4.2.3	Vincoli di Base . . . . .	42
4.2.4	Determinare G . . . . .	44
4.3	Fitting 2D+3D in Tempo Reale . . . . .	45
4.3.1	Vincoli di Coerenza 3D . . . . .	45
4.3.2	Estensione della Composizione Inversa 2D . . . . .	47
4.3.3	Aggiornamento della Soluzione Corrente . . . . .	49
4.3.4	L'Algoritmo Completo . . . . .	50
<b>5</b>	<b>Uso ed Estensione di Implementazioni Pre-Esistenti</b>	<b>53</b>
5.1	L'Implementazione di Cootes . . . . .	54
5.2	L'Implementazione di Stegmann: AAM-API . . . . .	54
5.2.1	Il Dataset IMM . . . . .	56

5.2.2	Fitting con l'AAM-API . . . . .	58
5.2.3	Metriche di Errore . . . . .	58
5.2.4	Casi di Studio . . . . .	60
5.2.4.1	Caso Person-Dependent . . . . .	60
5.2.4.2	Caso Person-Independent . . . . .	62
5.2.4.3	Estensione dell'AAM-API: un'Applicazione Video . . . . .	65
5.2.5	Efficienza di Fitting . . . . .	68
5.3	L'Implementazione di Brian Amberg . . . . .	68
5.4	Altre Implementazioni . . . . .	72
<b>6</b>	<b>Implementazione MATLAB di AAM 2D+3D in Tempo Reale</b>	<b>75</b>
6.1	Scelte Progettuali e Convenzioni . . . . .	76
6.2	Addestramento . . . . .	77
6.2.1	Estrazione delle Mode di Forma . . . . .	77
6.2.2	Estrazione delle Mode di Apparenza . . . . .	78
6.2.3	Pre-Computazioni . . . . .	79
6.2.3.1	Gradiente dell'Apparenza Media . . . . .	79
6.2.3.2	Immagini di Steepest Descent e Hessiana Inversa . . . . .	81
6.3	Fitting 2D in Tempo Reale . . . . .	82
6.3.1	Inizializzazione . . . . .	82
6.3.2	Calcolo dei Parametri Incrementali . . . . .	83
6.3.3	Aggiornamento della Soluzione . . . . .	85
6.3.4	Migliorare la Capacità di Convergenza . . . . .	86
6.4	Ricostruzione della Forma dal Movimento . . . . .	88
6.4.1	Preparazione della Matrice delle Misure . . . . .	88
6.4.2	Vincoli di Rotazione . . . . .	91
6.4.3	Vincoli di Base e Simmetria . . . . .	92
6.4.4	Stima Iniziale di M . . . . .	94
6.4.5	Soluzione . . . . .	95

6.5	Fitting 2D+3D In Tempo Reale . . . . .	98
6.5.1	Inizializzazione . . . . .	98
6.5.2	Stima delle Jacobiane . . . . .	99
6.5.3	Calcolo dei Parametri Incrementali . . . . .	100
6.5.4	Integrazione in Modo Efficace dei Vincoli di Coerenza 3D . . . . .	101
6.6	Risultati Sperimentali . . . . .	102
6.6.1	Fitting 2D . . . . .	102
6.6.1.1	Person-dependent . . . . .	102
6.6.1.2	Person-independent . . . . .	104
6.6.1.3	Complessità e Fitting in Tempo Reale . . . . .	105
6.6.2	Ricostruzione della Forma dal Movimento . . . . .	107
6.6.3	Fitting 2D + 3D . . . . .	109
<b>7</b>	<b>Conclusioni</b>	<b>111</b>
<b>Bibliografia</b>		<b>113</b>

# Capitolo 1

## Introduzione

Una classe di oggetti che rappresenta una particolare sfida per gli algoritmi di riconoscimento di pattern classici sono quelli deformabili, i cui esempi più familiari sono di natura biologica (volti umani, organi, arti...). Il riconoscimento di questi oggetti è difficile perché la variabilità nella posizione delle feature può essere dovuta sia a cambi di posizione o posa dell'oggetto, sia a deformazioni che avvengono nell'oggetto stesso.

Uno dei più popolari metodi per il riconoscimento di pattern applicato a oggetti deformabili è senza dubbio rappresentato dagli Active Appearance Models (AAM) e dalle loro numerose varianti. Generalmente, un Active Appearance Model è costituito da un modello lineare dell'oggetto deformabile, a cui si associa una particolare metodologia di riconoscimento o *fitting* che permette di individuare i parametri ottimali del modello per un particolare caso test. Si tratta quindi di un metodo generativo, essendo il riconoscimento basato sulla generazione di un'istanza che meglio approssima il caso test.

L'*addestramento* di un AAM è la fase in cui si determina il modello lineare dell'oggetto deformabile, attraverso un'analisi statistica dei dati di training che consistono in numerose coppie (*forma,apparenza*). L'apparenza (o texture) dell'oggetto è una rappresentazione del suo aspetto, sotto forma di immagine, mentre la forma localizza le feature di maggior importanza nell'oggetto e ne delimita i contorni, attraverso l'uso di una serie di punti detti *annotazioni* (landmark).

L'addestramento permette a un AAM di “imparare” le possibili deformazioni di forma e apparenza dell'oggetto in esame, e produrre così un modello compatto che permetta di riprodurre tali deformazioni, e molte altre, attraverso la variazione di opportuni parametri.

Il *fitting* consiste nel determinare la combinazione di parametri del modello che meglio approssima un dato caso test, ed è un problema in generale difficile da risolvere. Per questo motivo, la maggior parte dei metodi basati su AAM scelgono di usare tecniche di ottimizzazione locale e/o

## CAPITOLO 1. INTRODUZIONE

approssimazioni. È possibile quindi individuare diverse tipologie di metodi di fitting, a seconda delle tecniche di ottimizzazione usate, o delle caratteristiche di efficienza e capacità di convergenza.

In generale, sono state sviluppate innumerevoli estensioni e variazioni degli AAM originali, e non è nostra intenzione discuterle tutte in modo approfondito in questa sede. Una particolare eccezione è data dall'estensione alla composizione inversa [25], che permette prestazioni di fitting real-time, e rende possibile il fitting simultaneo della forma 3D dell'oggetto [42]. Tali estensioni si sono rivelate indispensabili per rendere possibili prestazioni in tempo reale (o quasi) in un linguaggio di alto livello come MATLAB.

### 1.1 Sintesi dei Contenuti

I Capitoli 2, 3 e 4 si occupano di stabilire il framework teorico necessario.

Nello specifico, il Capitolo 2, introduce gli Active Appearance Models come inizialmente concepiti da Cootes et al. [12] e serve sia da introduzione a chi non è familiare con il metodo, sia a fornire le basi per i due capitoli successivi. Di questi, il Capitolo 3 descrive in dettaglio l'estensione alla composizione inversa degli AAM e il Capitolo 4 estende ulteriormente la composizione inversa con la descrizione del metodo di fitting 2D+3D in tempo reale. Il Capitolo 4 introduce inoltre il concetto di ricostruzione della forma dal movimento, descrivendo in dettaglio l'algoritmo in forma chiusa di Xiao et al. [43].

Il Capitolo 5 fornisce una panoramica sull'attuale stato dell'arte delle implementazioni di Active Appearance Models liberamente disponibili, con un'enfasi su alcuni casi di studio sviluppati allo scopo di familiarizzare con gli AAM e comprenderne i limiti. Alcuni di questi casi di studio hanno richiesto la modifica e l'estensione del codice delle implementazioni, permettendo anche una valutazione della loro caratteristiche di qualità interna (come estensibilità e flessibilità del codice).

Nel Capitolo 6 è descritto in modo passo-passo e dettagliato il processo che ha portato all'implementazione in MATLAB degli AAM 2D+3D [42], concentrandosi sugli aspetti implementativi di ottimizzazione e sulla necessità di dover compensare a alcune omissioni negli articoli, piuttosto che a problematiche di ingegneria del software. La correttezza e efficienza dell'implementazione è successivamente dimostrata attraverso un'analisi comparativa dei risultati prodotti con quelli osservati nei casi di studio del Capitolo 5.

Infine, il Capitolo 7 contiene le riflessioni finali e rappresenta uno sguardo agli sviluppi più recenti degli AAM e ai possibili sviluppi futuri.

## 1.2 Notazione

La notazione segue il più possibile quella usata da Matthews et al. [26], compresi alcuni abusi di notazione che saranno comunque più volte spiegati e non dovrebbero rappresentare un ostacolo alla comprensione.

Se non specificato altrimenti, i vettori sono colonne, e hanno generalmente nome in minuscolo e grassetto. Ad esempio:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Le coordinate 2D sono sempre specificate usando  $u$  e  $v$  e sono generalmente denotate come:

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix}$$

L'uso delle componenti  $x$  e  $y$  è riservata per i vettori 3D:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Una generica forma 2D è costituita da una collezione di vettori 2D, e sarà generalmente denotata da  $\mathbf{x}_i$  o più raramente da  $\mathbf{s}$  (da non confondere con le mode di forma  $s_i$ ):

$$\mathbf{x} = \begin{bmatrix} u_1 & \dots & u_n \\ v_1 & \dots & v_n \end{bmatrix} \quad (1.1)$$

Le matrici hanno nome in maiuscolo e grassetto (ad esempio  $\mathbf{M}$ ), ad eccezione delle forme che hanno sempre nome minuscolo.

Un'apparenza è generalmente denotata da  $\mathbf{g}_i$  o più raramente da  $A$  (mentre  $A_i$  indica una *moda* di apparenza). Se denotata da  $A$  o  $A_i$  l'apparenza può essere in forma di matrice o vettore, a seconda del contesto. In ogni caso, l'operatore di accesso di pixel  $A(\mathbf{u})$  ha sempre lo stesso significato anche se l'apparenza è in forma di vettore.

*CAPITOLO 1. INTRODUZIONE*

## Capitolo 2

# Active Appearance Models Tradizionali

È difficile parlare di storia degli Active Appearance Models, anche se si osserva che modelli parametrici di forma erano già stati usati con successo degli Active Contour Models [20] (anche detti *Snakes*), e dalla loro naturale evoluzione, gli Active Shape Models [11].

Gli Snakes, entità assimilabili alle spline, riescono a posizionarsi (grazie a conoscenza a priori) in corrispondenza di particolari feature di una immagine, vincolati alla minimizzazione di una energia interna che ne assicura la “smoothness”. Gli Active Shape Models hanno migliorato questo metodo sostituendo alla conoscenza a priori un modello statistico addestrato, molto simile a quello usato per la forma negli Active Appearance Models.

Molti metodi precedenti agli AAM hanno inoltre tentato di modellare in modo più o meno lineare l'informazione di apparenza, ma data l'elevata dimensionalità del problema questi erano generalmente limitati a immagini a bassa risoluzione o erano caratterizzati da un numero molto grande di parametri [18]. Limitazioni che sono molto meno accentuate negli AAM.

Cootes et al. [12] hanno inoltre osservato che non è necessario risolvere ogni volta il problema di ottimizzazione completo associato al procedimento di fitting, in quanto diverse istanze di ottimizzazione sono molto simili tra loro. Hanno così proposto un metodo per “imparare” in anticipo come risolvere queste istanze, risultando in un algoritmo di fitting molto efficiente.

## 2.1 Acquisizione e Modellazione della Forma

Molto semplicemente, si definisce forma una collezione di annotazioni (posizioni 2D) che identificano feature o zone di interesse nell'apparenza dell'oggetto. Queste annotazioni sono tradizionalmente specificate manualmente, anche se molti studi sono stati fatti e continuano ad essere fatti per permettere l'automatizzazione (anche solo parziale) del processo. Alcuni di questi metodi tentano di stimare direttamente il modello deformabile a partire dai dati di training (usando metodi di ottimizzazione non-lineari [5] o euristiche [18]), mentre altri tentano di identificare corrispondenze tra feature applicando metodi di allineamento non-rigidi [31, 8].

È essenziale notare come il processo di annotazione sia molto importante, in quanto la qualità e la compattezza del modello prodotto dall'addestramento sono totalmente dipendenti dalla qualità delle annotazioni. Fondamentale è quindi non solo una corretta annotazione, ma anche la scelta delle feature da individuare.

Sfortunatamente, non è possibile definire delle regole generali per la scelta delle feature da annotare, in quanto queste dipendono fortemente dal contesto. Ma, come suggerito da alcuni autori [33], è possibile quantomeno trovare un punto di partenza nell'uso di feature tipiche:

- Punti di riferimento “anatomici” comuni tra le immagini da annotare, ad esempio: angoli della bocca, narici, angoli degli occhi ecc.
- Zone con particolari caratteristiche matematiche come: elevata curvatura, discontinuità, o estremi.
- Punti in cui feature curve o lineari si incontrano per formare delle giunzioni a “T” [13].
- Annotazioni “artificiali” (pseudo-landmarks) spaziate uniformemente tra le annotazioni vere e proprie allo scopo di seguire un contorno importante o facilitare l'annotazione.

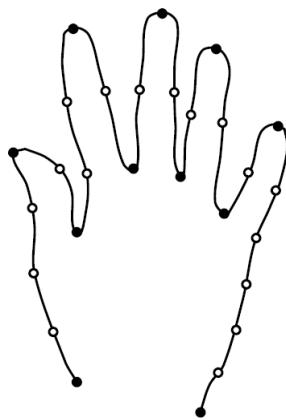


Figura 2.1. Esempio di annotazione di una mano: i punti solidi sono annotazioni in corrispondenza di caratteristiche anatomiche della mano, mentre le altre sono annotazioni artificiali [33].

## 2.1. ACQUISIZIONE E MODELLAZIONE DELLA FORMA

Le  $n$  annotazioni così ottenute si intendono organizzate in forma matriciale con la seguente struttura:

$$\mathbf{x} = \begin{bmatrix} u_1 & \dots & u_n \\ v_1 & \dots & v_n \end{bmatrix} \quad (2.1)$$

dove  $\mathbf{x}$  indica una generica forma,  $u$  corrisponde alla coordinata x dell'annotazione e  $v$  corrisponde alla coordinata y.

Qualunque metodo sia stato usato per l'annotazione (e a prescindere dalle feature scelte), è possibile costruire un modello di forma effettuando un'analisi statistica di Principal Component Analysis (PCA) sulle forme di addestramento, precedentemente allineate tramite analisi procustiana [15].

### 2.1.1 Allineamento delle Forme

L'analisi procustiana (procrustes analysis) permette di allineare tra loro due forme costituite dallo stesso numero di annotazioni, con l'obiettivo di minimizzare la somma quadratica delle distanze tra punti corrispondenti (Figura 2.2). Questa somma, detta distanza procustiana quadratica, è formalmente definita tra le forme  $\mathbf{x}_1$  e  $\mathbf{x}_2$  nel modo seguente:

$$P_d^2 = \sum_{i=1}^n \left[ (u_i^{(1)} - u_i^{(2)})^2 + (v_i^{(1)} - v_i^{(2)})^2 \right] \quad (2.2)$$

dove  $u_i^{(j)}$  indica la coordinata x della i-esima annotazione di  $\mathbf{x}_j$ , e similmente  $v_i^{(j)}$  indica la coordinata y.

Un metodo per allineare al meglio  $N$  forme usando l'analisi procustiana consiste nell'iterare i seguenti passi fino a convergenza:

1. Applicare una traslazione rigida a ogni forma in modo che il suo nuovo baricentro sia l'origine.
2. Scegliere una forma come stima iniziale della forma media  $\bar{\mathbf{x}}$ .
3. Allineare, una alla volta, tutte le forme rispetto alla forma media  $\bar{\mathbf{x}}$  usando l'analisi procustiana.
4. Ricalcolare la forma media a partire dalle forme allineate.
5. Vincolare la nuova forma media allineandola a  $\bar{\mathbf{x}}$  (usando nuovamente l'analisi procustiana) e assegnare a  $\bar{\mathbf{x}}$  il risultato ottenuto.
6. Se la variazione nella forma media è superiore alla soglia di convergenza, ripetere dal passo (3).

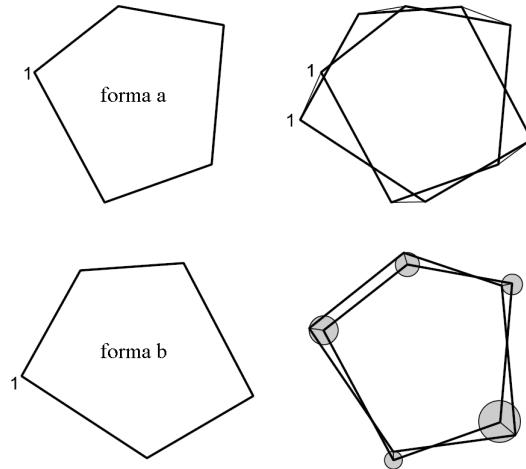


Figura 2.2. Allineamento prodotto dalla minimizzazione della distanza procustiana quadratica [33].

Il baricentro di una forma è semplicemente la media aritmetica dei suoi punti:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} u_i \\ v_i \end{bmatrix} \quad (2.3)$$

Mentre la forma media è ottenuta mediando per ogni coordinata su tutte le  $N$  forme:

$$\begin{bmatrix} \bar{u}_i \\ \bar{v}_i \end{bmatrix} = \frac{1}{N} \sum_{j=1}^N \begin{bmatrix} u_i^{(j)} \\ v_i^{(j)} \end{bmatrix} \quad (2.4)$$

Tutti i passi della procedura di allineamento sono abbastanza comprensibili, anche se potrebbe lasciar perplessi l'ulteriore allineamento effettuato al passo (5). Il motivo è che in questo modo si vincola la forma media a mantenere sempre la stessa orientazione, e questa operazione, assieme alla centratura nell'origine, risolve le ambiguità nella procedura di allineamento. Senza dilungarsi troppo, basti sapere che le ambiguità derivano dalla non unicità della soluzione del problema di allineamento.

Oltre alla presenza di ambiguità, si verifica un problema legato alla non-linearità introdotta dalla procedura di allineamento con alcuni tipi di forme, come nel caso di semplici rettangoli (Figura 2.3). Questo è risolto proiettando nello *spazio tangente*, ossia moltiplicando ogni forma allineata  $\mathbf{x}_i$  per un fattore  $1/(\text{vec}(\mathbf{x}_i)^T \text{vec}(\bar{\mathbf{x}}))$ , dove  $\text{vec}(\mathbf{x})$  è l'operatore di vettorizzazione:

$$\text{vec} \left( \begin{bmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nm} \end{bmatrix} \right) = \left[ \begin{array}{ccccccc} a_{11} & a_{21} & \dots & a_{n1} & \dots & a_{1m} & \dots & a_{nm} \end{array} \right]^T \quad (2.5)$$

## 2.1. ACQUISIZIONE E MODELLAZIONE DELLA FORMA

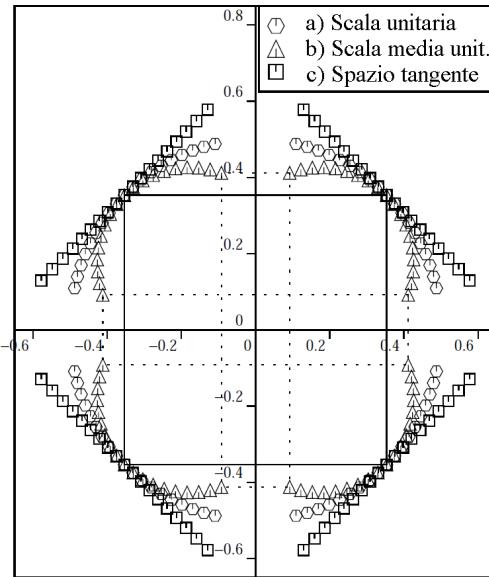


Figura 2.3. Risultati dell'allineamento di un insieme di rettangoli con diversa dimensione e orientazione attraverso l'analisi procustiana [13]:

- a) rettangoli tutti scalati a dimensione unitaria, allineamento basato su rotazione e traslazione.
- b) rettangoli con dimensione media unitaria, allineamento con inclusione della scala (si noti la maggiore non-linearietà).
- c) allineamento come su (b), con proiezione nello spazio tangente.

Un trattamento completo dell'analisi procustiana è al di fuori dei nostri scopi, e si fa riferimento agli articoli di Stegmann [33] e Cootes et al. [13] per i dettagli implementativi.

### 2.1.2 Modello Statistico di Forma

Siano le forme prodotte dal processo di allineamento nuovamente indicate con  $\mathbf{x}_i$ , per  $i = 1 \dots N$ . Da queste, ricavare il modello di forma è una semplice applicazione della Principal Component Analysis (PCA). Sia  $\mathbf{C}$  la matrice di covarianza delle forme:

$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N \text{vec}(\mathbf{x}_i - \bar{\mathbf{x}}) \text{vec}(\mathbf{x}_i - \bar{\mathbf{x}})^T \quad (2.6)$$

con  $\bar{\mathbf{x}}$  media delle forme  $\mathbf{x}_i$ . Siano quindi  $\Lambda_i$  gli autovalori di  $\mathbf{C}$  (ordinati in ordine decrescente) e  $\phi_i$  gli autovettori corrispondenti. Dato un valore percentuale  $\sigma$  (tipicamente compreso tra 95 e 98), che indica la quantità di variazione di forma che deve rappresentare il modello, il numero  $m$  di *mode di forma* è il valore minimo che soddisfa:

$$\sum_{i=1}^m \Lambda_i \geq \frac{\sigma}{100} \sum \Lambda_i \quad (2.7)$$

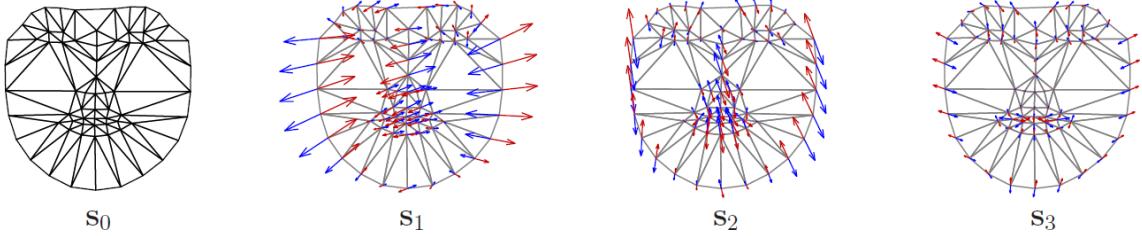


Figura 2.4. Esempio di modello di forma di un AAM: le mode di forma possono essere viste come deformazioni della forma media. Si noti l'impatto decrescente che le mode di forma hanno sul modello, questa è la chiave della compattezza degli AAM [25].

Indicando le mode di forma con  $s_i = \phi_i$  per  $i = 1 \dots m$  e la forma media con  $s_0 = \bar{x}$ , si osserva che una generica forma può essere costruita come:

$$s = s_0 + \sum_{i=1}^m p_i s_i \quad (2.8)$$

che è il modello parametrico di forma con parametri  $p_i$ .

## 2.2 Modellazione dell'Apparenza

L'apparenza è un'immagine, o più precisamente una porzione di immagine, che rappresenta l'oggetto deformabile di interesse. L'immagine può essere in scala di grigi o a colori, e la trattazione è indifferente, essendo i singoli pixel delle apparenze soggetti ad operazioni lineari.

A caratterizzare la porzione di immagine di interesse è la forma associata che, oltre a delimitare i contorni dell'oggetto, identifica le feature che possono essere soggette a deformazioni.

Prima ancora di pensare a come il modello di apparenza possa essere costruito, si osservi che anche solo il calcolo di un'apparenza media è un problema mal posto. Infatti, se si osserva la Figura 2.5, si nota che se le regioni di interesse delle immagini non coincidono, non è possibile ottenere dati statistici significativi sulle apparenze. Si potrebbe comunque obiettare che una semplice trasformazione affine sarebbe adeguata in quel particolare esempio ad allineare le apparenze, ma ciò non è vero nel caso generale a causa della deformabilità dell'oggetto. Per questo motivo, allo scopo di organizzare le apparenze nello stesso "spazio di coordinate" è necessaria una trasformazione non-lineare, detta *warp*, operazione che fa in modo di "mappare" la porzione di immagine delimitata da una forma nello spazio di coordinate di un'altra forma, detta "di riferimento" (nel nostro caso  $s_0$ ).

Si deve inoltre considerare che ci possono essere anche discrepanze tra i valori delle apparenze vere e proprie, dovute a variazioni di luminosità o altri fattori esterni che hanno influito sul processo di acquisizione. Ciò è fonte di una variazione esterna che non ha a che fare con le variazioni

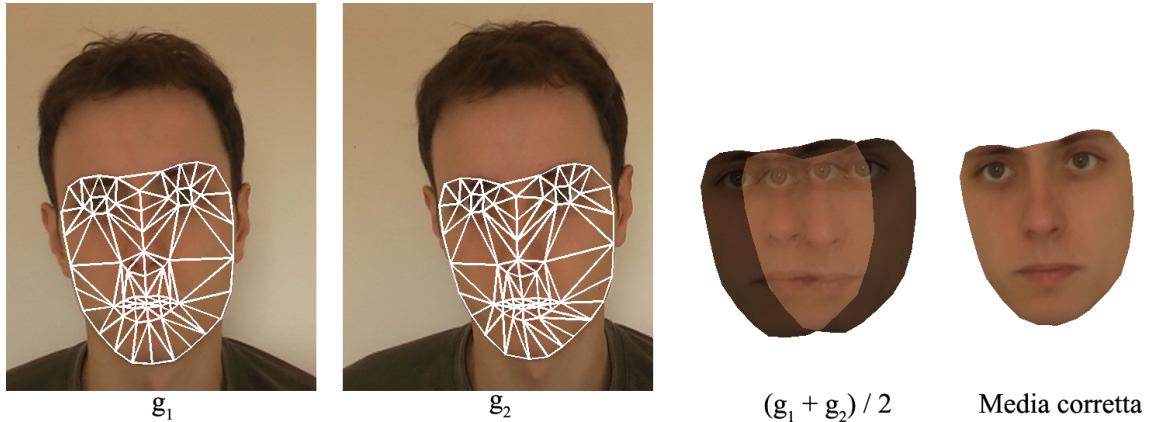


Figura 2.5. Esempio delle difficoltà inerenti all'analisi statistica di apparenze: la semplice media aritmetica non ha significato in questo caso.

nell'apparenza a cui si è interessati (ovvero dovute a deformazioni dell'oggetto), e che si può in larga parte rimuovere tramite una *normalizzazione fotometrica*.

L'addestramento dell'AAM si va quindi a concludere combinando il modello di forma e quello di apparenza in un *modello combinato* di forma e apparenza.

### 2.2.1 Warp di Immagini

Un warp è generalmente definito come la distorsione di un'immagine. Nonostante il termine distorsione possa essere fuorviante, il processo non dev'essere necessariamente distruttivo e non è certamente così per i warp a cui si è interessati in questa sede. Un warp può essere definito tramite una generica funzione  $\mathbf{u}' = f(\mathbf{u})$ , dove  $\mathbf{u}' = [u' \ v']^T$  è usato per accedere all'immagine  $I$ . Si ottiene così una nuova immagine,  $I'$ , formalmente definita nel modo seguente:

$$I'(\mathbf{u}) = I(f(\mathbf{u})) \quad (2.9)$$

Il warp tipicamente usato dagli AAM è di tipo lineare a tratti e fa uso di una mesh triangolare applicata alle forme iniziali (non quelle prodotte dall'allineamento, quindi) e a quella media. Un modo semplice per ottenere questa triangolazione è dato dall'algoritmo di Delaunay, anche se può rivelarsi necessaria la rimozione di alcuni triangoli nel caso in cui il perimetro della forma non sia convesso.

La funzione di warp è detta lineare a tratti perché è realizzata applicando trasformazioni lineari diverse, a seconda del triangolo considerato. Queste trasformazioni lineari si determinano usando le coordinate baricentriche, e rappresentano una relazione lineare univoca tra posizioni riferite alla

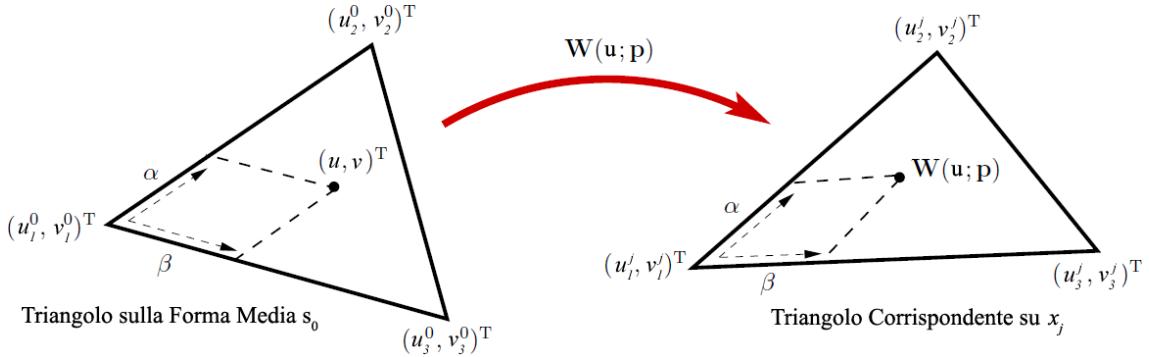


Figura 2.6. Rapresentazione del warp di una coordinata, effettuata usando le coordinate baricentriche del triangolo che la contiene [25].

forma media  $\mathbf{s}_0$  a posizioni riferite alle forme di training (prima dell'allineamento)  $\mathbf{x}_j$ :

$$\mathbf{u}' = f(\mathbf{u}) = \mathbf{u}_1^j + \alpha (\mathbf{u}_2^j - \mathbf{u}_1^j) + \beta (\mathbf{u}_3^j - \mathbf{u}_1^j) \quad (2.10)$$

dove  $\mathbf{u}_1^j$ ,  $\mathbf{u}_2^j$  e  $\mathbf{u}_3^j$  rappresentano i vertici del triangolo considerato in  $\mathbf{x}_j$  e:

$$\alpha = \frac{(u - u_1^0)(v_3^0 - v_1^0) - (v - v_1^0)(u_3^0 - u_1^0)}{(u_2^0 - u_1^0)(v_3^0 - v_1^0) - (v_2^0 - v_1^0)(u_3^0 - u_1^0)} \quad (2.11)$$

$$\beta = \frac{(v - v_1^0)(u_2^0 - u_1^0) - (u - u_1^0)(v_2^0 - v_1^0)}{(u_2^0 - u_1^0)(v_3^0 - v_1^0) - (v_2^0 - v_1^0)(u_3^0 - u_1^0)} \quad (2.12)$$

con  $\mathbf{u}_1^0 = [u_1^0 \ v_1^0]^T$ ,  $\mathbf{u}_2^0 = [u_2^0 \ v_2^0]^T$ ,  $\mathbf{u}_3^0 = [u_3^0 \ v_3^0]^T$  vertici corrispondenti a  $\mathbf{u}_1^j$ ,  $\mathbf{u}_2^j$  e  $\mathbf{u}_3^j$  in  $\mathbf{s}_0$ . Il risultato di questo procedimento matematico è illustrato nella Figura 2.6.

Iterando il procedimento di warp su tutte le coordinate interne a  $\mathbf{s}_0$ , è possibile trasformare tutte le apparenze applicando la Formula 2.9 (ricordando che  $f$  cambia a seconda del triangolo considerato), in modo che queste siano espresse nel sistema di riferimento della forma  $\mathbf{s}_0$ , e permettendo di effettuare le successive analisi statistiche. Queste apparenze allineate sono quindi vettorizzate (Formula 2.5) ad ottenere  $\mathbf{g}_i$  per  $i = 1 \dots N$ .

## 2.2.2 Modello Statistico di Apparenza

Prima di procedere all'analisi statistica delle apparenze prodotte dal warp, è possibile anteporre un processo di normalizzazione fotometrica atto a rimuovere variazioni globali nelle apparenze, applicando una trasformazione lineare sulle apparenze  $\mathbf{g}_i$ :

$$\mathbf{g}'_i = \frac{\mathbf{g}_i - \beta \mathbf{1}}{\alpha} \quad (2.13)$$

## 2.2. MODELLAZIONE DELL'APPARENZA

Questa si realizza calcolando l'apparenza media  $\bar{\mathbf{g}}$ , e facendo in modo che il suo valore medio sia zero:

$$\bar{\mathbf{g}}_{zm} = \bar{\mathbf{g}} - \frac{1}{L} \sum_{i=1}^L \bar{g}_i \quad (2.14)$$

e che la sua varianza sia unitaria:

$$\bar{\mathbf{g}}_{norm} = \frac{\bar{\mathbf{g}}_{zm}}{\sigma}, \quad \sigma = \sqrt{\frac{1}{L} \sum_{i=1}^L \bar{g}_{zm_i}^2} \quad (2.15)$$

( $L$  è il numero di pixel in un'apparenza  $\mathbf{g}_i$ ).

Determinare le intensità normalizzate delle apparenze consiste nella ripetizione del processo iterativo:

1. Stima di  $\bar{\mathbf{g}}_{norm}$  con la Formula 2.15.
2. Per ogni apparenza  $\mathbf{g}_i$ :
  - (a)  $\alpha = \mathbf{g}_i^T \mathbf{g}_{norm}$ .
  - (b)  $\beta = (\mathbf{g}_i^T \mathbf{1}) / L$ .
  - (c) Normalizza  $\mathbf{g}_i$  con la formula 2.13.
3. Ripeti dal passo 1 finché  $\bar{\mathbf{g}}_{norm}$  non è stabile.

L'analisi statistica delle apparenze è analoga quella usata per le forme, anche se l'elevata dimensionalità dei vettori coinvolti può rendere il calcolo della matrice di covarianza (Formula 2.6) intrattabile. Si nota però che l'elevata dimensionalità dei vettori di apparenza implica che in tutti i casi pratici la matrice di covarianza non è a rango pieno, e questo permette l'uso del seguente metodo efficiente per il calcolo di autovalori e autovettori.

Si definisca la matrice delle apparenze  $\mathbf{G}$ :

$$\mathbf{G} = \begin{bmatrix} (\mathbf{g}_1 - \bar{\mathbf{g}}_{norm}) & \dots & (\mathbf{g}_N - \bar{\mathbf{g}}_{norm}) \end{bmatrix} \quad (2.16)$$

La matrice di covarianza delle apparenze sarebbe, in notazione matriciale:

$$\mathbf{C} = \frac{1}{N-1} \mathbf{G} \mathbf{G}^T \quad (2.17)$$

che ha dimensione ragguardevole, al contrario della matrice:

$$\mathbf{C}' = \frac{1}{N-1} \mathbf{G}^T \mathbf{G} \quad (2.18)$$

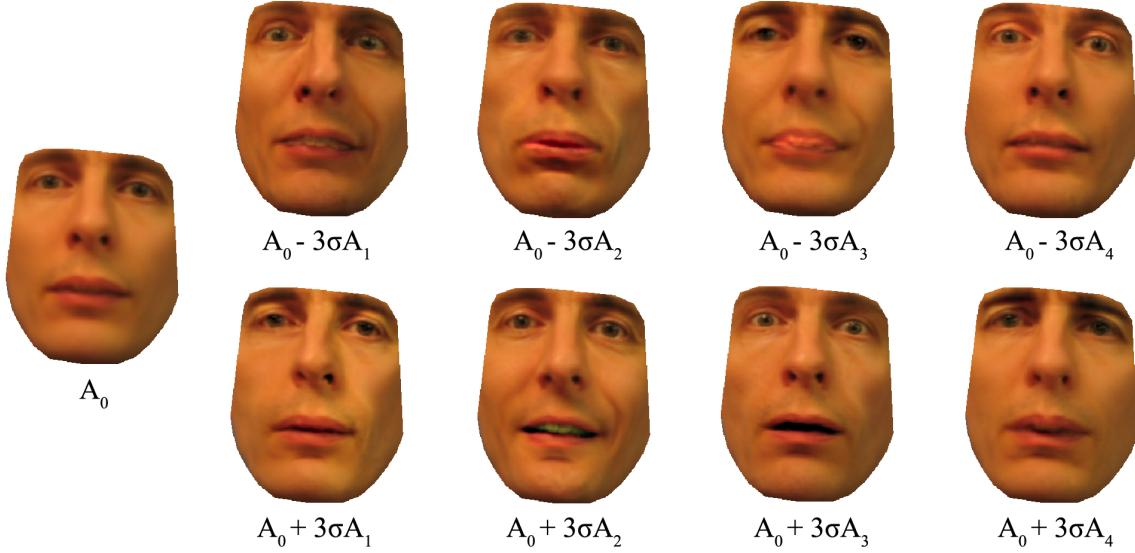


Figura 2.7. Il modello di apparenza di un AAM è costituito dall'apparenza media e dalle mode di apparenza. In questo esempio è possibile vedere l'effetto prodotto sull'apparenza media sommando e sottraendo le prime 4 mode di apparenza di un modello, moltiplicate per 3 deviazioni standard  $\sigma$  (la radice quadrata dell'autovalore corrispondente alla moda).

Si dimostra [33] che dati gli autovalori non nulli  $\Lambda'_i$  di  $\mathbf{C}'$  e i corrispondenti autovettori  $\phi'_i$ , gli autovalori  $\Lambda_i$  di  $\mathbf{C}$  sono uguali ai  $\Lambda'_i$ , mentre gli autovettori corrispondenti  $\phi_i$  sono dati da:

$$\phi_i = \frac{\mathbf{G}\phi'_i}{\sqrt{\Lambda'_i}} \quad (2.19)$$

Determinato il numero delle mode di apparenza  $l$  tramite la Formula 2.7, e definiti  $\mathbf{A}_0 = \bar{\mathbf{g}}_{norm}$  e  $\mathbf{A}_i = \phi_i$  per  $i = 1 \dots l$  si arriva alla seguente formulazione del modello di apparenza:

$$\mathbf{A} = \mathbf{A}_0 + \sum_{i=1}^l \lambda_i \mathbf{A}_i \quad (2.20)$$

dove, analogamente al modello di forma,  $\mathbf{A}$  indica una generica apparenza e  $\lambda_i$  per  $i = 1 \dots l$  sono i parametri di apparenza.

## 2.3 Modello Combinato

Si è visto come costruire un modello lineare di forma e uno di apparenza in grado di rappresentare deformazioni viste nel training set e molte altre, ma l'addestramento dell'AAM non finisce così. Infatti, nell'incarnazione classica degli AAM, si usa un solo modello lineare *combinato* di forma e apparenza, allo scopo di rendere ancora più compatto e rappresentativo il modello e di facilitare il fitting.

Ricordando che i parametri di forma sono  $p_i$  per  $i = 1 \dots m$ , e i parametri di apparenza sono  $\lambda_j$  per  $j = 1 \dots l$ , si può pensare di disporre i parametri in forma vettoriale a formare i vettori  $\mathbf{p}$  e  $\boldsymbol{\lambda}$ , rispettivamente. Allo stesso modo, le mode di forma e apparenza possono essere espresse in forma matriciale:

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \dots & \mathbf{s}_m \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_1 & \dots & \mathbf{A}_l \end{bmatrix} \quad (2.21)$$

Sfruttando l'ortonormalità delle matrici, si dimostra facilmente che le Formule 2.8 e 2.20 possono essere così riarrangiate:

$$\begin{aligned} \mathbf{p} &= \mathbf{S}(\mathbf{s} - \mathbf{s}_0) \\ \boldsymbol{\lambda} &= \hat{\mathbf{A}}^T(\mathbf{A} - \mathbf{A}_0) \end{aligned} \quad (2.22)$$

La costruzione del modello combinato è basata su un'ulteriore PCA applicata sulla concatenazione, opportunamente pesata, dei parametri di forma e apparenza di ogni campione nel training set:

$$\mathbf{b}_i = \begin{bmatrix} \mathbf{W}_s \mathbf{p}_i \\ \boldsymbol{\lambda}_i \end{bmatrix} \quad (2.23)$$

dove  $\mathbf{p}_i$  e  $\boldsymbol{\lambda}_i$  sono determinati applicando la Formula 2.22 all'i-esimo elemento del training set e  $\mathbf{W}_s$  è un'opportuna matrice di pesatura.

Il motivo per cui si rivela necessaria un'operazione di pesatura è dovuto alla differente variabilità tra i parametri di forma e i parametri di apparenza: questo fa in modo che pari incrementi in un parametro di forma e di apparenza producano cambiamenti di diversa entità nei due modelli. Nella sua forma più semplice  $\mathbf{W}_s = \gamma \mathbf{I}$ , e si richiede necessario il calcolo del solo scalare  $\gamma$ , che è il rapporto tra la variabilità dell'apparenza e quella della forma:

$$\gamma = \frac{\sum \Lambda_{A_i}}{\sum \Lambda_{s_i}} \quad (2.24)$$

(con la variabilità rappresentata dalla sommatoria degli autovalori corrispondenti alle mode di forma e apparenza usate nel modello).

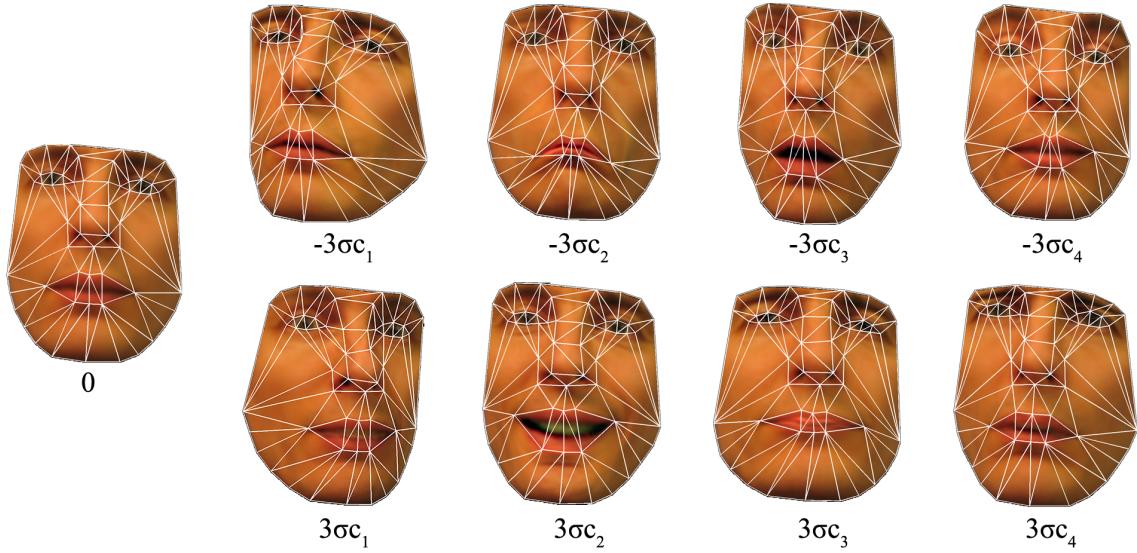


Figura 2.8. Un esempio di modello combinato di un AAM, nuovamente espresso in termini di 3 deviazioni standard dalla media (che stavolta è 0).  $c_i$  indica l'i-esima colonna di  $\mathbf{P}_c$ .

Dati quindi i vettori  $\mathbf{b}_i$  dei parametri concatenati e pesati di ogni campione del training set, e applicata a questi un'ulteriore PCA, si otterranno gli autovalori  $\Lambda_i$  (anche questi in ordine decrescente) e i relativi autovettori  $\phi_i$ . Usando nuovamente la Formula 2.7, si riduce ulteriormente la dimensionalità del problema a  $r$  mode, aumentando quindi la capacità di generalizzazione e la robustezza al rumore.

Si definisce così il modello combinato di forma e apparenza:

$$\mathbf{b} = \mathbf{P}_c \mathbf{c} \quad (2.25)$$

dove  $\mathbf{c}$  sono i *parametri combinati* e:

$$\mathbf{P}_c = \begin{bmatrix} \phi_1 & \dots & \phi_r \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{cs} \\ \mathbf{P}_{cg} \end{bmatrix} \quad (2.26)$$

in cui  $\mathbf{P}_{cs}$  ha  $m$  righe e  $\mathbf{P}_{cg}$  ne ha  $l$ .

La natura lineare del modello permette di ricavare direttamente i parametri  $\mathbf{p}$  e  $\boldsymbol{\lambda}$  a partire da  $\mathbf{c}$ :

$$\begin{aligned} \mathbf{p} &= \mathbf{W}_s^{-1} \mathbf{P}_{cs} \mathbf{c} \\ \boldsymbol{\lambda} &= \mathbf{P}_{cg} \mathbf{c} \end{aligned} \quad (2.27)$$

## 2.4 Il Processo di Fitting

Data un'immagine di test  $I$  e una forma di inizializzazione  $\mathbf{s}_{init}$  sufficientemente vicina alla soluzione, il procedimento di fitting consiste nel trovare la miglior configurazione di parametri  $(\mathbf{p}, \boldsymbol{\lambda})$  dell'AAM e la miglior trasformazione affine  $\mathbf{N}(\mathbf{x}; \mathbf{q})$  che minimizzano l'*errore immagine*  $\delta$ :

$$\delta = \sum_{\mathbf{u} \in \mathbf{s}_0} \boldsymbol{\sigma} \mathbf{I}(\mathbf{u})^2 \quad (2.28)$$

dove:

$$\boldsymbol{\sigma} \mathbf{I}(\mathbf{u}) = A(\mathbf{u}) - I(\mathbf{W}_{\mathbf{s}_0 \rightarrow \mathbf{s}}(\mathbf{u})) \quad (2.29)$$

è l'*immagine di errore*.  $A$  è determinata a partire da  $\boldsymbol{\lambda}$  applicando la Formula 2.20 e, pur essendo  $A$  in forma vettoriale, risulta comodo l'abuso di notazione  $A(\mathbf{u})$  che ha il prevedibile effetto di accedere al pixel corrispondente alla posizione  $\mathbf{u}$ .  $\mathbf{W}_{\mathbf{s}_0 \rightarrow \mathbf{s}}(\mathbf{u})$  è un warp (Sezione 2.2.1), che trasforma coordinate riferite alla forma media  $\mathbf{s}_0$  in coordinate riferite alla forma  $\mathbf{s}$ , che è determinata nel modo seguente:

$$\mathbf{s} = \mathbf{N} \left( \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q} \right) \quad (2.30)$$

Anche se non evidente dalla Formula 2.29, è importante ricordare di applicare la normalizzazione fotometrica descritta nella Sezione 2.2.2 al risultato del warp, per renderlo coerente con il modello di apparenza.

Prima di definire la trasformazione affine  $\mathbf{N}(\mathbf{x}; \mathbf{q})$ , si cerchi di comprendere la motivazione che porta alla sua introduzione, ricordando che durante la costruzione dei modelli di forma e apparenza i dati originari sono stati trasformati e normalizzati in vario modo. Come conseguenza, le forme prodotte dal modello di forma sono centrate nell'origine e hanno dimensioni e orientamento pressoché arbitrarie, mentre le apparenze prodotte dal modello di apparenza sono riferite alla forma media  $\mathbf{s}_0$  e hanno valori di intensità normalizzati e a media nulla. E laddove l'apparenza non crea difficoltà (assumendo che i dati dell'immagine test siano normalizzati correttamente e che sia usato un warp per cambiare sistema di riferimento), la forma deve in qualche modo poter variare anche in orientamento, posizione e dimensione.

Nel caso 2D una trasformazione affine è tipicamente definita da un fattore di scala  $\epsilon$ , una traslazione  $[t_u \ t_v]^T$ , e una rotazione  $\mathbf{R}$  di angolo  $\theta$ ; in questa formulazione la trasformazione identica è data da:  $\epsilon = 1$ ,  $t_u = 0$ ,  $t_v = 0$  e  $\theta = 0$  (o  $\mathbf{R} = \mathbf{I}$ ). Si osserva però che questa rappresentazione non è lineare (a causa di  $\theta$ ) e quindi non è l'ideale per un algoritmo di rifinimento iterativo come quello usato per il fitting di AAM.

Una rappresentazione alternativa più lineare, e quella usata dai parametri di trasformazione affine  $\mathbf{q}$ , è la seguente:  $q_1 = \epsilon \cos(\theta) - 1$ ,  $q_2 = \epsilon \sin(\theta)$ ,  $q_3 = t_u$  e  $q_4 = t_v$ ; in modo che il vettore  $\mathbf{q} = \mathbf{0}$  rappresenti la trasformazione identica.

L'operatore di trasformazione affine  $\mathbf{N}(\mathbf{x}; \mathbf{q})$  non fa altro che applicare la trasformazione descritta dai parametri  $\mathbf{q}$  alla forma o vettore 2D  $\mathbf{x}$ :

$$\mathbf{N}(\mathbf{x}; \mathbf{q}) = \begin{bmatrix} 1 + q_1 & -q_2 \\ q_2 & 1 + q_1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} q_3 & \dots & q_3 \\ q_4 & \dots & q_4 \end{bmatrix} \quad (2.31)$$

I parametri di trasformazione affine  $\mathbf{q}$  rappresentano anch'essi incognite nel problema di fitting e si vanno ad aggiungere alle incognite già presenti, ovvero i parametri di del modellocombinato  $\mathbf{c}$ .

Si potrebbe affermare che la soluzione di un problema con un tale numero di variabili sia estremamente difficile e lenta, e sarebbe sicuramente così nel caso si cercasse di determinare una soluzione globale esatta del problema. Nel caso in cui si accontentasse di determinare una soluzione locale è possibile ottenere ottimi risultati, e con un'alta efficienza, linearizzando la relazione intrinsecamente non lineare tra l'immagine di errore  $\sigma \mathbf{I}$  e il cambiamento incrementale nei parametri  $\Delta \mathbf{c}$  e  $\Delta \mathbf{q}$  che garantisce la massima riduzione dell'errore immagine:

$$\begin{bmatrix} \Delta \mathbf{c} \\ \Delta \mathbf{q} \end{bmatrix} \approx \mathbf{R} \sigma \mathbf{I} \quad (2.32)$$

Gli AAM determinano  $\mathbf{R}$  applicando una *regressione lineare multivariata* a varie combinazioni di valori dei parametri incrementali  $\Delta c_i$  e  $\Delta q_j$  e la relativa immagine di errore prodotta da questi. Una completa trattazione di questo tipo di regressione è al di fuori dei nostri scopi, e si rimanda all'articolo di Stegmann [33] per i dettagli matematici e implementativi.

Concludendo, il fitting AAM è dato dal seguente procedimento iterativo:

1. Inizializza  $\mathbf{s}$ :  $\mathbf{s} \leftarrow \mathbf{s}_{init}$ .
2. Inizializza  $\mathbf{A}$ :  $\mathbf{A} \leftarrow \mathbf{A}_o$  (alternativamente, è possibile applicare la Formula 2.22 a  $I(\mathbf{W}_{\mathbf{s}_0 \rightarrow \mathbf{s}}(\mathbf{u}))$  per inizializzare  $\boldsymbol{\lambda}$  e quindi  $\mathbf{A}$ ).
3. Determina l'immagine di errore  $\sigma \mathbf{I}$  con la Formula 2.29.
4. Determina i parametri incrementali  $\Delta \mathbf{c}$  e  $\Delta \mathbf{q}$  che minimizzano l'errore con la Formula 2.32.
5. Determina  $\Delta \mathbf{p}$  a partire da  $\Delta \mathbf{c}$  usando la Formula 2.27 e determina  $\Delta \mathbf{s} = \sum_{i=1}^m \Delta p_i$ .
6. Determina  $\Delta \boldsymbol{\lambda}$  a partire da  $\Delta \mathbf{c}$  usando la Formula 2.27 e determina  $\Delta \mathbf{A} = \sum_{i=1}^l \Delta \lambda_i$ .
7. Aggiorna  $\mathbf{s}$  e  $\mathbf{A}$  usando i parametri incrementali:  $\mathbf{A} \leftarrow \mathbf{A} + \Delta \mathbf{A}$ ,  $\mathbf{s} \leftarrow \mathbf{s} + \Delta \mathbf{s}$ .
8. Applica la trasformazione globale incrementale con parametri  $\Delta \mathbf{q}$  a  $\mathbf{s}$ :  $\mathbf{s} \leftarrow \mathbf{N}(\mathbf{s}; \Delta \mathbf{q})$ .
9. Ripeti dal punto (3) finché non c'è convergenza o  $\delta = \|\sigma \mathbf{I}\|^2$  scende sotto una certa soglia.

Dove, nei passi (7) e (8), è stata sfruttata la linearità (o quasi linearità) delle operazioni coinvolte per evitare di dover ricavare esplicitamente i parametri di forma e apparenza correnti.

## Capitolo 3

# Active Appearance Models Rivisitati

Una delle più importanti evoluzioni nel campo del fitting AAM è stata l'introduzione della *composizione inversa* (inverse compositional) [3]. Questa tecnica ha permesso l'utilizzo di un approccio più analitico al fitting AAM, rendendo in teoria possibile una soluzione in forma esatta al problema di ottimizzazione locale.

Si vedrà come, attraverso l'uso di alcune approssimazioni nella soluzione analitica, sia possibile ottenere nuovamente una relazione lineare tra immagine di errore e incrementi nei parametri necessari alla minimizzazione. L'aspetto importante è che a differenza dell'approccio empirico di regressione lineare usato dagli AAM tradizionali, l'approccio analitico è (con le dovute assunzioni) una soluzione in forma chiusa. Questa è quindi più semplice da implementare correttamente e ci si aspetta anche che i risultati prodotti siano migliori.

Inoltre, l'uso della composizione inversa permette di proiettare l'immagine di errore in un sottospazio che ne semplifica drasticamente il calcolo, con un significativo incremento nelle prestazioni.

Considerando che la costruzione dei modelli di forma e di apparenza è invariata negli AAM con composizione inversa, si rimanda al Capitolo 2 per la discussione questi aspetti. Si tenga comunque conto che gli AAM che verranno descritti a breve non usano un modello combinato, e per questo motivo essi sono anche detti “indipendenti”.

### 3.1 Introduzione alla Composizione Inversa

Sia  $\mathbf{W}(\mathbf{u}; \mathbf{p})$  il warp che mappa coordinate  $\mathbf{u}$  riferite alla spazio di coordinate della forma media in coordinate riferite alla forma  $\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i$ , realizzato con le modalità descritte nella Sezione 2.2.1. Una composizione di warp può essere definita come:

$$\mathbf{W}_{comp}(\mathbf{u}; \mathbf{p}) = \mathbf{W}(\mathbf{u}; \mathbf{p}) \circ \mathbf{W}(\mathbf{u}; \Delta\mathbf{p}) = \mathbf{W}(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}); \mathbf{p}) \quad (3.1)$$

che è la composizione diretta o in avanti (forward composition). A questo tipo di composizione si contrappone la composizione inversa:

$$\mathbf{W}_{comp}(\mathbf{u}; \mathbf{p}) = \mathbf{W}(\mathbf{u}; \mathbf{p}) \circ \mathbf{W}(\mathbf{u}; \Delta\mathbf{p})^{-1} = \mathbf{W}(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p})^{-1}; \mathbf{p}) \quad (3.2)$$

che è stata inizialmente introdotta nell'ambito dell'allineamento di immagini con l'algoritmo di Lucas-Kanade [22].

Allo scopo di fornire un'introduzione all'uso della composizione di warp nel contesto degli Active Appearance Models, verrà brevemente illustrato il ruolo che svolge nell'algoritmo di Lucas-Kanade.

#### 3.1.1 L'Algoritmo di Lucas-Kanade

L'allineamento di immagini (*template matching*) consiste nell'individuare la porzione di un'immagine test che corrisponde a un'immagine costante data (template). Nello specifico, l'algoritmo di Lucas-Kanade si prefissa di minimizzare l'errore:

$$\sum_{\mathbf{u} \in T} [T(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p}))]^2 \quad (3.3)$$

dove  $T$  è il template e  $I$  è l'immagine test. Cercare il valore di  $\mathbf{p}$  che minimizzi l'espressione è chiaramente un problema non lineare perché, anche nel caso si assumesse che  $\mathbf{W}(\mathbf{u}; \mathbf{p})$  fosse lineare in  $\mathbf{p}$ , i valori di  $I(\mathbf{u})$  non sono in genere in relazione lineare con il valore di  $\mathbf{u}$ .

Allo scopo di linearizzare il problema, l'algoritmo di Lucas-Kanade assume che si disponga di una stima iniziale del valore di  $\mathbf{p}$  e minimizza ripetutamente la seguente misura d'errore rispetto a  $\Delta\mathbf{p}$ :

$$\sum_{\mathbf{u} \in T} [T(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p} + \Delta\mathbf{p}))]^2 \quad (3.4)$$

usando ad ogni iterazione la regola di aggiornamento  $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ .

Effettuando un'espansione di Taylor, la versione linearizzata del problema diventa:

$$\sum_{\mathbf{u} \in T} \left[ T(\mathbf{u}) - I \left( \mathbf{W}(\mathbf{u}; \mathbf{p}) - \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right) \right]^2 \quad (3.5)$$

### 3.1. INTRODUZIONE ALLA COMPOSIZIONE INVERSA

dove  $\nabla I$  è il gradiente dell'immagine test valutato in  $\mathbf{W}(\mathbf{u}; \mathbf{p})$  e  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  è la Jacobiana del warp (il cui calcolo verrà approfondito in seguito) valutata in  $(\mathbf{u}; \mathbf{p})$ .

Si dimostra che la soluzione  $\Delta \mathbf{p}$  del problema linearizzato diventa:

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x} \in T} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [T(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p}))] \quad (3.6)$$

dove  $\mathbf{H}$  è l'approssimazione di Gauss-Newton dell'Hessiana:

$$\mathbf{H} = \sum_{\mathbf{x} \in T} \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[ \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] \quad (3.7)$$

Questa versione dell'algoritmo, illustrata nella Figura 3.1, è detta *additiva in avanti* (forward-additive): additiva perché  $\mathbf{p}$  è aggiornato sommando incrementi ad ogni iterazione; in avanti in quanto il warp  $\mathbf{W}(\mathbf{u}; \mathbf{p})$  mappa verso lo spazio di coordinate dell'immagine  $I$ .

Considerando che in questo algoritmo sia  $\nabla I$  che  $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  dipendono da  $\mathbf{p}$ , è necessario che questi siano ricalcolati ad ogni iterazione (e di conseguenza anche l'Hessiana), con il risultato che l'algoritmo è molto lento.

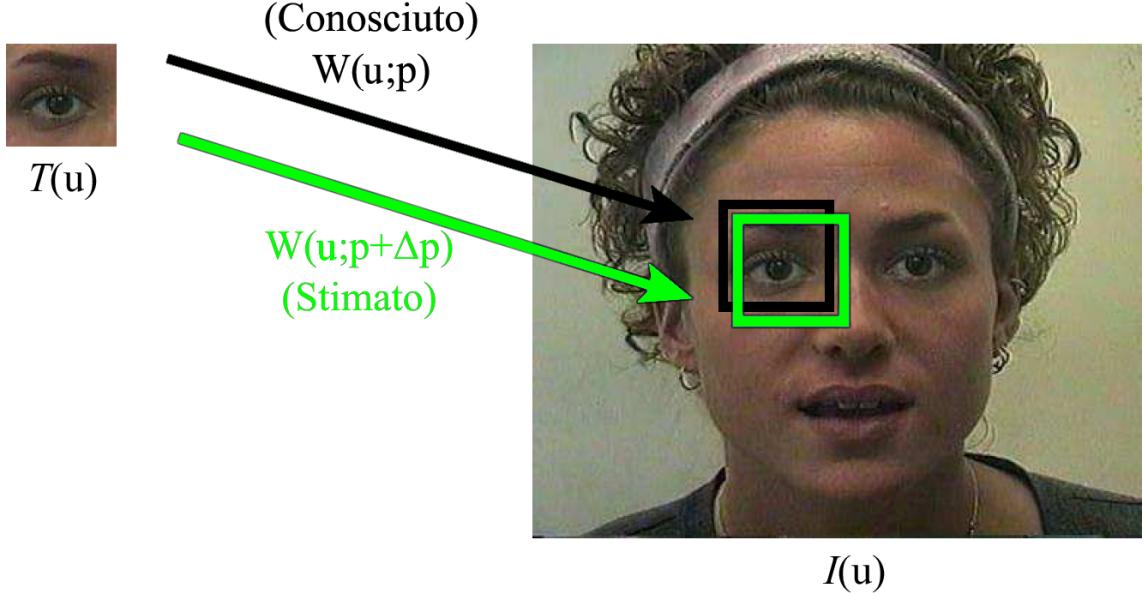


Figura 3.1. Esempio dell'algoritmo di Lucas-Kanade additivo in avanti: un'incremento nei parametri di forma è stimato in modo da allineare il template con la parte di immagine cercata [25].

### 3.1.2 Allineamento di Immagini con Composizione in Avanti

Un algoritmo di Lucas-Kanade modificato in modo da usare la *composizione in avanti* (forward composition) di warp ha come obiettivo la minimizzazione dell'errore:

$$\sum_{\mathbf{u} \in T} [T(\mathbf{u}) - I(\mathbf{W}(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}); \mathbf{p}))]^2 \quad (3.8)$$

e la regola di aggiornamento è data dalla Formula 3.1.

Allo scopo di determinare  $\Delta\mathbf{p}$ , anche questa misura di errore è linearizzata usando Taylor:

$$\sum_{\mathbf{u} \in T} \left[ T(\mathbf{u}) - I(\mathbf{W}(\mathbf{W}(\mathbf{u}; \mathbf{0}); \mathbf{p})) - \nabla I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right]^2 \quad (3.9)$$

dove, per definizione,  $\mathbf{W}(\mathbf{u}; \mathbf{0}) = \mathbf{u}$ .

Il calcolo di  $\Delta\mathbf{p}$  e Hessiana non cambia (Formule 3.6 e 3.7) e il gradiente  $\nabla I$  è sempre valutato in  $\mathbf{W}(\mathbf{u}; \mathbf{p})$ , anche se in questo caso la Jacobiana  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  è valutata in  $(\mathbf{u}; \mathbf{0})$  ed è quindi una costante.

Come si vedrà a breve, la composizione di warp è un'operazione più onerosa di una semplice somma di vettori, ma il fatto di non dover ricalcolare la Jacobiana ad ogni iterazione rende comunque questa versione decisamente più efficiente di quella additiva in avanti.

### 3.1.3 Allineamento di Immagini con Composizione Inversa

Sebbene l'algoritmo di Lucas-Kanade con composizione in avanti non necessiti di ricalcolare la Jacobiana ad ogni iterazione, esso richiede ancora una quantità significativa di tempo di calcolo per determinare il gradiente  $\nabla I$  e l'Hessiana.

La composizione inversa rende possibile un ulteriore miglioramento delle prestazioni, semplicemente “spostando” il warp incrementale dall'immagine test al template:

$$\sum_{\mathbf{u} \in T} [I(\mathbf{W}(\mathbf{u}; \mathbf{p})) - T(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}))]^2 \quad (3.10)$$

e applicando poi la regola di composizione inversa (Formula 3.2) per l'aggiornamento del warp.

È stato dimostrato che questa formulazione è equivalente alla composizione in avanti [4, 3], fatto che si può intuitivamente spiegare considerando che il warp incrementale che si sta stimando è lo stesso della composizione in avanti, ma nella direzione “opposta”.

Applicando anche in questo caso un'espansione di Taylor, si ottiene la versione linearizzata del problema :

$$\sum_{\mathbf{u} \in T} \left[ I(\mathbf{W}(\mathbf{u}; \mathbf{p})) - T(\mathbf{W}(\mathbf{u}; \mathbf{0})) - \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right]^2 \quad (3.11)$$

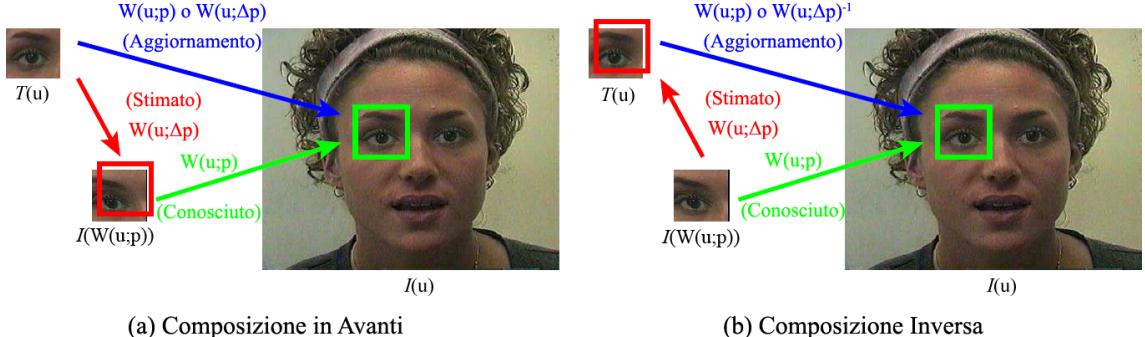


Figura 3.2. Differenza tra composizione in avanti e composizione inversa nell’algoritmo di Lucas–Kanade [25]: (a) nella composizione in avanti  $\Delta p$  è usato per la definizione di un warp incrementale  $W(u; \Delta p)$  da comporre con quello corrente  $W(u; p)$ . (b) nella composizione inversa il warp incrementale stimato, anziché essere riferito a  $I(W(u; p))$ , è riferito a  $A_0$ ; esso è inoltre nella direzione opposta, quindi è necessario invertirlo prima della composizione.

dove  $\nabla T$  è il gradiente del template valutato in  $u$ .

La soluzione  $\Delta p$  è determinata in modo analogo alla composizione in avanti, sostituendo il gradiente di  $I$  con quello di  $T$ :

$$\Delta p = H^{-1} \sum_{u \in T} \left[ \nabla T \frac{\partial W}{\partial p} \right]^T [I(W(u; p)) - T(u)] \quad (3.12)$$

e lo stesso vale per il calcolo dell’Hessiana:

$$H = \sum_{u \in T} \left[ \nabla T \frac{\partial W}{\partial p} \right]^T \left[ \nabla T \frac{\partial W}{\partial p} \right] \quad (3.13)$$

Si osserva che in questi calcoli la Jacobiana  $\frac{\partial W}{\partial p}$  è costante perché valutata in  $(u; 0)$ , così come il gradiente del template  $\nabla T$ , e di conseguenza anche l’Hessiana. Quindi ad ogni iterazione, per il calcolo di  $\Delta p$ , sarà necessario determinare solo la differenza  $I(W(u; p)) - T(u)$ . Il risultato è un algoritmo estremamente efficiente di allineamento di immagini.

## 3.2 Operazioni sui Warp

Si è visto come l’uso della composizione inversa comporti l’applicazione di concetti nuovi come la *composizione*, l’*inversione* e la *derivazione* alla funzione di warp. Nessuna di queste operazioni ha un’interpretazione banale, in quanto la funzione di warp è lineare a tratti e non ha una definizione analitica globale.

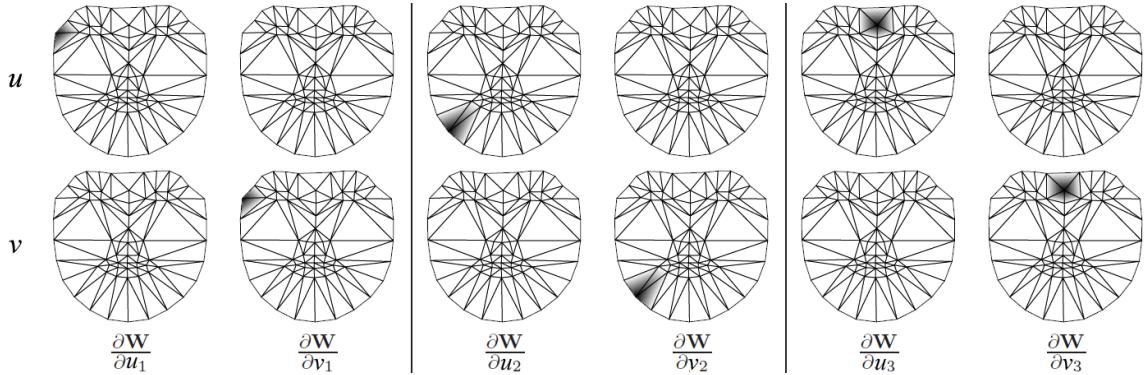


Figura 3.3. Componenti  $x$  e  $y$  delle Jacobiane del warp, calcolate rispetto a 3 diversi vertici (i colori sono invertiti: bianco indica 0, nero indica 1) [25].

### 3.2.1 Calcolo della Jacobiana

Come già visto in precedenza (Sezione 2.2.1), il risultato della funzione warp  $\mathbf{W}(\mathbf{u}; \mathbf{p})$  dipende da  $\mathbf{p}$  in modo abbastanza peculiare. Non è quindi immediatamente evidente il modo in cui si debba determinare la Jacobiana  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  del warp.

Ricordando che una generica forma è definibile come  $\mathbf{s} = [u_1 \ v_1 \ u_2 \ v_2 \ \dots \ u_n \ v_n]^T$ , il calcolo della Jacobiana può essere riformulato usando la regola della catena :

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \sum_{i=1}^n \left( \frac{\partial \mathbf{W}}{\partial u_i} \frac{\partial u_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial v_i} \frac{\partial v_i}{\partial \mathbf{p}} \right) \quad (3.14)$$

(i parametri di  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  sono omessi, ma si ricorda che essi sono coordinate 2D riferite alla forma  $\mathbf{s}_0$ , e la formula è valutata per ogni posizione).

Determinare  $\frac{\partial \mathbf{W}}{\partial u_i}$  e  $\frac{\partial \mathbf{W}}{\partial v_i}$  non è difficile se si riconsidera la Formula 2.10:

$$\mathbf{u}' = \mathbf{u}_i + \alpha (\mathbf{u}_j - \mathbf{u}_i) + \beta (\mathbf{u}_k - \mathbf{u}_i)$$

da cui deriva facilmente:

$$\frac{\partial \mathbf{W}}{\partial u_i} = \begin{bmatrix} 1 - \alpha - \beta \\ 0 \end{bmatrix}, \quad \frac{\partial \mathbf{W}}{\partial v_i} = \begin{bmatrix} 0 \\ 1 - \alpha - \beta \end{bmatrix} \quad (3.15)$$

Ricordando che il calcolo di  $\alpha$  e  $\beta$  (Formule 2.11 e 2.12) avviene sulla forma media  $\mathbf{s}_0$ , risulta evidente che queste Jacobiane sono delle immagini espresse nel sistema di riferimento di  $\mathbf{s}_0$ . Più nello specifico,  $\alpha$  e  $\beta$  sono determinate per tutti i punti  $[u \ v]^T$  interni ai triangoli, considerando un triangolo alla volta. Questo calcolo è ripetuto 3 volte, una per ogni vertice del triangolo, rinominando i vertici in modo da impostare ogni volta  $\mathbf{u}_i$  su un vertice diverso, e portando così al

### 3.2. OPERAZIONI SUI WARP

calcolo di 6 diverse Jacobiane  $\frac{\partial \mathbf{W}}{\partial u_i}$  e  $\frac{\partial \mathbf{W}}{\partial v_i}$ . Questo perché le Jacobiane assumono valori diversi da zero solo nei triangoli che condividono un dato vertice, come evidente dalla Figura 3.3.

Per quanto riguarda le derivate  $\frac{\partial u_i}{\partial \mathbf{p}}$  e  $\frac{\partial v_i}{\partial \mathbf{p}}$ , queste si ricavano direttamente dal legame tra i parametri di forma  $\mathbf{p}$  e una generica forma, che si ricorda essere:

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i \quad (3.16)$$

da cui si deriva facilmente:

$$\begin{aligned} \frac{\partial u_i}{\partial \mathbf{p}} &= \left[ \begin{array}{cccc} s_1^{u_i} & s_2^{u_i} & \dots & s_m^{u_i} \end{array} \right] \\ \frac{\partial v_i}{\partial \mathbf{p}} &= \left[ \begin{array}{cccc} s_1^{v_i} & s_2^{v_i} & \dots & s_m^{v_i} \end{array} \right] \end{aligned} \quad (3.17)$$

dove  $s_j^{u_i}$  indica la coordinata della j-esima moda di forma corrispondente alla componente x del vertice  $\mathbf{u}_i$ , e in modo analogo per la componente y.

Ricapitolando, si avrà che la Jacobiana del warp valutata in  $(\mathbf{x}; \mathbf{0})$  sarà costituita da  $2m$  immagini: 2 immagini, una per componente, per ogni parametro di forma. Un esempio di queste Jacobiane è rappresentato in Figura 3.4.

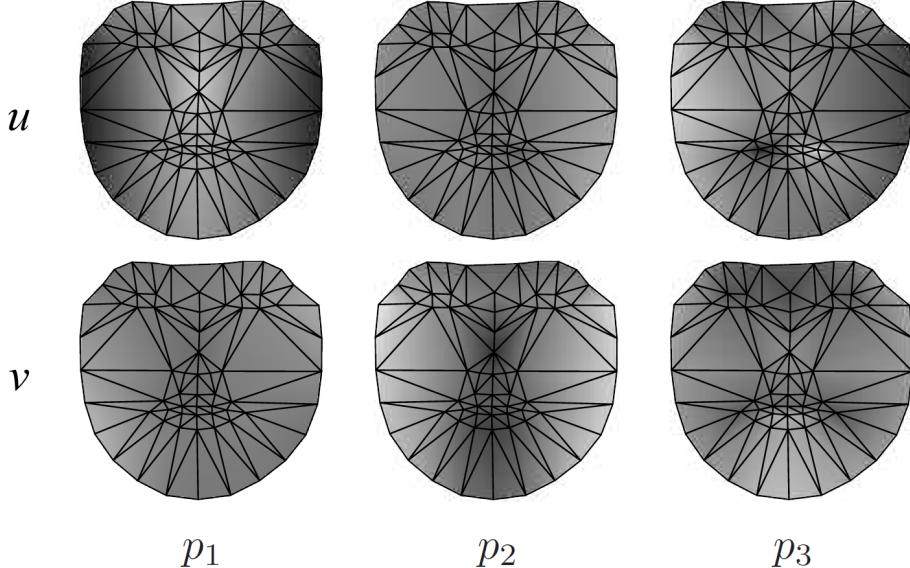


Figura 3.4. Jacobiane del warp  $\frac{\partial \mathbf{W}}{\partial p_i}$  associate ai primi 3 parametri di forma per il modello rappresentato in Figura 2.4. La prima moda di quel modello corrisponde principalmente a una rotazione a destra e a sinistra della testa, la seconda a una rotazione in su e in giù, mentre la terza a movimenti della bocca.

### 3.2.2 Inversione

In precedenza, quando è stata introdotta la composizione inversa di warp (Formula 3.2), non è stato approfondito cosa comporta effettivamente l'inversione di un warp  $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$ . Anche in questo caso si è interessati ad un'operazione approssimata ed efficiente, a partire dall'espansione di Taylor del warp incrementale:

$$\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}) = \mathbf{W}(\mathbf{u}; \mathbf{0}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + O(\Delta\mathbf{p}^2) = \mathbf{u} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + O(\Delta\mathbf{p}^2) \quad (3.18)$$

Effettuando un'ulteriore espansione di Taylor (stavolta sulla composizione di warp) e usando l'approssimazione della Formula 3.18 per il warp incrementale si ottiene che:

$$\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}) \circ \mathbf{W}(\mathbf{u}; -\Delta\mathbf{p}) = \mathbf{u} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + O(\Delta\mathbf{p}^2) = \mathbf{u} + O(\Delta\mathbf{p}^2) \quad (3.19)$$

da cui si deduce che  $\mathbf{W}(\mathbf{u}; -\Delta\mathbf{p})$  è un'approssimazione del primo ordine di  $\mathbf{W}(\mathbf{u}; \Delta\mathbf{p})^{-1}$ .

Si noti però che questo argomento non è rigoroso dal punto di vista matematico, in quanto le due Jacobiane nella Formula 3.19 sono valutate in punti diversi, distanti  $O(\Delta\mathbf{p})$  tra loro, e sono per questo uguali a meno di un ordine zero in  $\Delta\mathbf{p}$ . Ma essendo la sottrazione tra le Jacobiane moltiplicata per  $\Delta\mathbf{p}$ , si ha che l'approssimazione è ancora del primo ordine, ed è quindi accettabile. Si osservi inoltre che anche la composizione di due warp lineari a tratti non è un'operazione matematica ben definita, ma i risultati pratici confermano che l'approssimazione è fondamentalmente corretta.

### 3.2.3 Composizione

Allo scopo di implementare un algoritmo con composizione in avanti o inversa, è fondamentale definire che significato ha la composizione di warp. Si potrebbe pensare di applicare direttamente la definizione di composizione di funzioni e determinare innanzitutto (nel caso della composizione inversa):

$$\Delta\mathbf{s}_0 = - \sum_{i=1}^m \Delta\mathbf{p} \quad (3.20)$$

dove  $\Delta\mathbf{s}_0 = [\Delta u_1^0 \ \Delta v_1^0 \ \dots \ \Delta u_n^0 \ \Delta v_n^0]^T$  è il cambiamento incrementale nella forma media  $\mathbf{s}_0$ .

Per effettuare la composizione, dovrebbe quindi essere sufficiente applicare il warp  $\mathbf{W}(\cdot; \mathbf{p})$  agli spostamenti incrementali prodotti dal warp inverso su  $\mathbf{s}_0$ , anche se a questo punto non è ben chiaro quale triangolo usare per il warp. Scelte diverse porterebbero a risultati diversi, ed è questo il motivo per cui la composizione di warp lineari a tratti è difficile da definire. Come illustrato nella Figura 3.5, una possibile scelta è data dal triangolo di  $\mathbf{s}_0$  contenente la posizione  $[(u_i^0 + \Delta u_i^0) \ (v_i^0 + \Delta v_i^0)]^T$ , ma non è detto che questo triangolo esista.

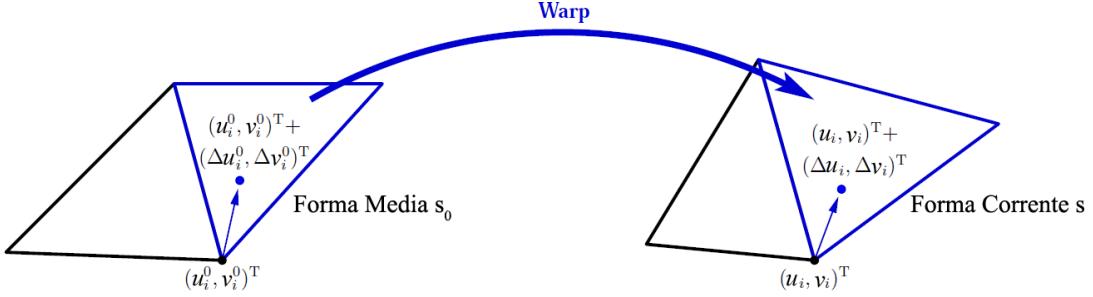


Figura 3.5. Composizione di un warp incrementale con quello corrente: nel migliore dei casi la posizione  $[(u_i^0 + \Delta u_i^0) (v_i^0 + \Delta v_i^0)]^T$  è all'interno di uno specifico triangolo di  $s_0$ , e la composizione è facile da definire [25].

Una soluzione a queste problematiche consiste nel calcolare il warp  $\mathbf{W} \left( [(u_i^0 + \Delta u_i^0) (v_i^0 + \Delta v_i^0)]^T; \mathbf{p} \right)$  per ogni triangolo che condivide il vertice  $\mathbf{u}_i$ , e fare la media di tutti i risultati. In questo modo la composizione di warp è sempre definita, e come conseguenza della mediatura il risultato è più uniforme.

### 3.2.4 Estensione alle Trasformazioni Globali

Nella Sezione 2.4 sono stati esposti due differenti approcci alla parametrizzazione di una stessa trasformazione globale affine. Un terzo approccio consiste nell'effettuare la trasformazione affine costruendo una funzione di warp che utilizza delle particolari mode di forma. Questo ha il vantaggio di unificare la notazione e di permettere il riutilizzo della teoria alla base del calcolo delle Jacobiane del warp (il perché ciò sia desiderabile sarà più chiaro in seguito).

Si consideri la stessa parametrizzazione usata dagli AAM tradizionali:

$$\mathbf{N}(\mathbf{x}; \mathbf{q}) = \begin{bmatrix} 1 + q_1 & -q_2 \\ q_2 & 1 + q_1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} q_3 & \dots & q_3 \\ q_4 & \dots & q_4 \end{bmatrix}$$

Si dimostra facilmente che questa trasformazione può essere riformulata in modo del tutto simile a un warp nel caso  $\mathbf{x} = \mathbf{s}_0$ :

$$\mathbf{N}(\mathbf{s}_0; \mathbf{q}) = \mathbf{s}_0 + \sum_{i=1}^4 q_i \mathbf{s}_i^* \quad (3.21)$$

con  $\mathbf{s}_1^* = \mathbf{s}_0 = [u_1^0 \ v_1^0 \ \dots \ u_n^0 \ v_n^0]^T$ ,  $\mathbf{s}_2^* = [-v_1^0 \ u_1^0 \ \dots \ -v_n^0 \ u_n^0]^T$ ,  $\mathbf{s}_3^* = [1 \ 0 \ \dots \ 1 \ 0]^T$  e  $\mathbf{s}_4^* = [0 \ 1 \ \dots \ 0 \ 1]^T$ .

È sufficiente poi derivare una trasformazione affine standard a partire dal cambiamento nella forma

$\mathbf{s}_0$  indotto da  $\mathbf{N}(\mathbf{s}_0; \mathbf{q})$ , sfruttando le Formule 2.10, 2.11 e 2.12:

$$\mathbf{N}(\mathbf{u}; \mathbf{q}) = \begin{bmatrix} a_1 + a_2u + a_3v \\ a_4 + a_5u + a_6v \end{bmatrix} \quad (3.22)$$

I dettagli su come determinare i coefficienti della trasformazione affine non sono presentati nell'articolo originale [25], quindi saranno discussi nella parte implementativa (Sezione 6.3.2). Basti sapere che, scelto un triangolo qualsiasi, si ricavano i valori di  $\alpha$  e  $\beta$  associati alla coordinata trasformata e da questi è poi possibile ricavare la relazione lineare con la coordinata non trasformata.

Si anticipa infine che il passo di aggiornamento incrementale del processo di fitting assume che le mode di forma  $\mathbf{s}_i$  e i vettori  $\mathbf{s}_j^*$  siano ortonormali tra loro, quindi i vettori  $\mathbf{s}_j^*$  dovranno essere almeno normalizzati. E anche se, auspicabilmente, i vettori  $\mathbf{s}_i$  e  $\mathbf{s}_j^*$  sono vicini ad essere ortogonali tra loro (in quanto i primi rappresentano una deformazione non-rigida, mentre i secondi una trasformazione rigida), se si desidera minimizzare le imprecisioni durante la fase di fitting è consigliabile anche ortogonalizzare tale insieme di vettori.

Si sottolinea che i vettori  $\mathbf{s}_j^*$  non si vanno ad aggiungere in modo definitivo alle mode di forma, le due cose sono trattate separatamente dall'algoritmo.

### 3.3 Soluzione in Forma Chiusa

La strategia di minimizzazione usata per il fitting differisce sostanzialmente da quella usata negli AAM tradizionali. Infatti, l'uso della composizione inversa modifica drasticamente le operazioni coinvolte e in aggiunta, essendo il modello indipendente, si rende disponibile un'ulteriore opportunità per l'ottimizzazione della procedura. Questo grazie a una particolare operazione di proiezione che "elimina" dal problema di minimizzazione dell'errore immagine la dipendenza dai parametri di apparenza.

#### 3.3.1 Proiezione dell'Errore Immagine

L'obiettivo del fitting è sempre la minimizzazione dell'errore immagine, che è possibile riscrivere in forma vettoriale con l'uso della norma  $\ell^2$ :

$$\sum_{\mathbf{u} \in \mathbf{s}_0} [A(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))]^2 = \|A(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|^2 \quad (3.23)$$

con  $A(\mathbf{u})$  determinata a partire dai parametri  $\lambda$ :

$$A(\mathbf{u}) = A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) \quad (3.24)$$

### 3.3. SOLUZIONE IN FORMA CHIUSA

Qui si è nuovamente abusato della notazione, trattando le apparenze  $A_0$  e  $A_i$  come immagini bidimensionali.

Apparentemente, ci si trova nella stessa situazione degli AAM tradizionali, con la minimizzazione dell'errore immagine che andrebbe effettuata contemporaneamente su  $\mathbf{p}$ ,  $\mathbf{q}$  e  $\boldsymbol{\lambda}$ : un problema difficile da risolvere analiticamente.

Fortunatamente, è possibile rendere il problema trattabile usando una particolare proiezione: sia  $\text{span}(A_i)$  il sottospazio lineare generato dalle mode di apparenza  $A_i$ , e sia  $\text{span}(A_i)^\perp$  il suo complemento ortogonale, la Formula 3.23 può essere riscritta come:

$$\|A(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|_{\text{span}(A_i)^\perp}^2 + \|A(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|_{\text{span}(A_i)}^2 \quad (3.25)$$

dove  $\|\cdot\|_L^2$  indica la norma  $\ell^2$  al quadrato, considerando la proiezione sul sottospazio  $L$ . Si dimostra facilmente che qualsiasi combinazione lineare di appearance è ortogonale a  $\text{span}(A_i)^\perp$ , quindi dal primo termine scompare la dipendenza da  $\boldsymbol{\lambda}$ :

$$\|A_0(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|_{\text{span}(A_i)^\perp}^2 + \|A(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|_{\text{span}(A_i)}^2 \quad (3.26)$$

Per quanto riguarda il secondo termine, si osserva che il suo valore minimo è necessariamente 0, a prescindere da  $\mathbf{p}$ . Infatti la proiezione del residuo su  $\text{span}(A_i)$ , o è una combinazione lineare di mode di apparenze, o è nulla. Di conseguenza, l'espressione è minimizzabile sequenzialmente: si minimizza inizialmente la prima espressione determinando  $\mathbf{p}$  e  $\mathbf{q}$  ottimali, per poi minimizzare la seconda espressione e determinare  $\boldsymbol{\lambda}$ .

Se si assume che le mode di forma siano ortonormali, la soluzione della seconda espressione in forma chiusa è una semplice proiezione del residuo:

$$\lambda_i = \sum_{\mathbf{u} \in \mathbf{s}_0} [I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) - A_o(\mathbf{u})] A_i(\mathbf{u}) \quad (3.27)$$

Al contrario, determinare  $\mathbf{p}$  e  $\mathbf{q}$  che minimizzino la prima espressione della Formula 3.26 è più complicato, in quanto è necessario operare nel sottospazio  $\text{span}(A_i)^\perp$ . Si noti comunque la similità con il problema di allineamento di immagini (Sezione 3.1.3), con la differenza che si deve considerare anche una trasformazione globale, e che la minimizzazione avviene in un sottospazio diverso.

In linea di massima la procedura rimane comunque invariata, e si verifica che è sufficiente proiettare  $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  (e la controparte  $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{q}}$  introdotta da  $\mathbf{q}$ ) sul sottospazio  $\text{span}(A_i)^\perp$ . Così facendo, nel determinare i prodotti scalari usati nella fase di ottimizzazione (simili a quelli della Formula 3.12) si avrà anche la contemporanea proiezione dell'errore immagine.

Non è quindi necessario proiettare esplicitamente anche l'errore immagine, con l'ovvio risparmio di tempo di calcolo (questo fenomeno è detto “*project out*”).

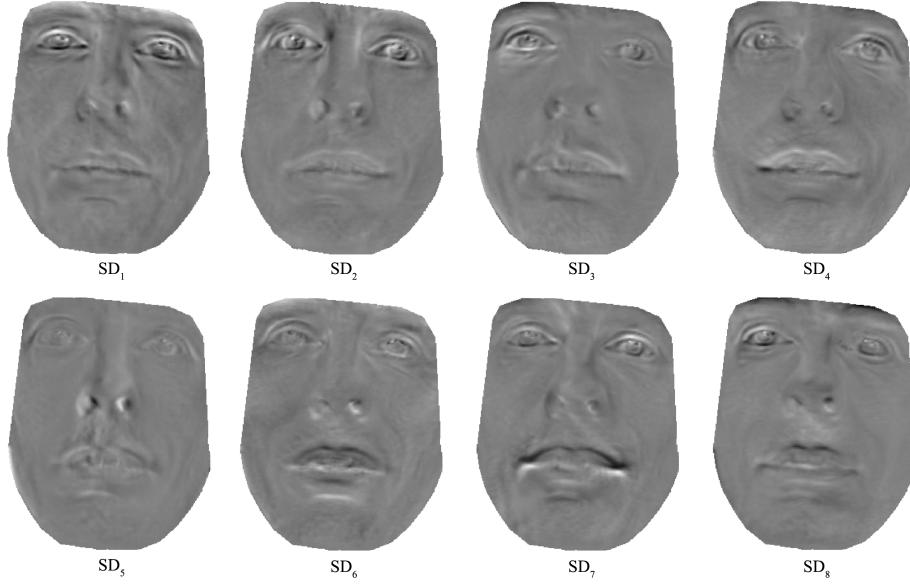


Figura 3.6. Le immagini di steepest descent danno un'indicazione delle zone dell'immagine di errore che più sono influenzate da un particolare parametro  $q_i$  o  $p_i$ . Infatti i parametri incrementalmente sono determinati applicando una trasformazione lineare ai prodotti scalari di queste con l'immagine di errore.

### 3.3.2 Le Immagini di Steepest Descent

Nel seguito si chiameranno immagini di *steepest descent* i risultati della proiezione di  $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  e  $\nabla A_0 \frac{\partial \mathbf{N}}{\partial \mathbf{q}}$  su  $\text{span}(A_i)^\perp$ :

$$SD_j(\mathbf{u}) = \nabla A_0 \frac{\partial \mathbf{N}}{\partial q_j} - \sum_{i=1}^l \left[ \sum_{\mathbf{w} \in \mathbf{s}_0} A_i(\mathbf{w}) \cdot \nabla A_0 \frac{\partial \mathbf{N}}{\partial q_j} \right] A_i(\mathbf{u}), \quad j = 1 \dots 4 \quad (3.28)$$

$$SD_{j+4}(\mathbf{u}) = \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} - \sum_{i=1}^l \left[ \sum_{\mathbf{w} \in \mathbf{s}_0} A_i(\mathbf{w}) \cdot \nabla A_0 \frac{\partial \mathbf{W}}{\partial p_j} \right] A_i(\mathbf{u}), \quad j = 1 \dots m \quad (3.29)$$

dove  $\nabla A_0$  è il gradiente dell'appearance media. Considerando che le Jacobiane della funzione composta  $\mathbf{N}(\mathbf{u}; \mathbf{q}) \circ \mathbf{W}(\mathbf{u}; \mathbf{p})$  sono valutate in  $\mathbf{p} = \mathbf{0}$  e  $\mathbf{q} = \mathbf{0}$ , si dimostra che:

$$\frac{\partial}{\partial \mathbf{q}} \mathbf{N} \circ \mathbf{W} = \frac{\partial \mathbf{N}}{\partial \mathbf{q}} = \sum_{i=1}^n \left( \frac{\partial \mathbf{N}}{\partial u_i} \frac{\partial u_i}{\partial \mathbf{q}} + \frac{\partial \mathbf{N}}{\partial v_i} \frac{\partial v_i}{\partial \mathbf{q}} \right) \quad (3.30)$$

$$\frac{\partial}{\partial \mathbf{p}} \mathbf{N} \circ \mathbf{W} = \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \sum_{i=1}^n \left( \frac{\partial \mathbf{W}}{\partial u_i} \frac{\partial u_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial v_i} \frac{\partial v_i}{\partial \mathbf{p}} \right) \quad (3.31)$$

e si determinano entrambe con le tecniche descritte nella Sezione 3.2.1, in quanto  $\mathbf{N}(\mathbf{u}; \mathbf{q})$  può essere definita come una combinazione lineare delle forme  $\mathbf{s}_i^*$ .

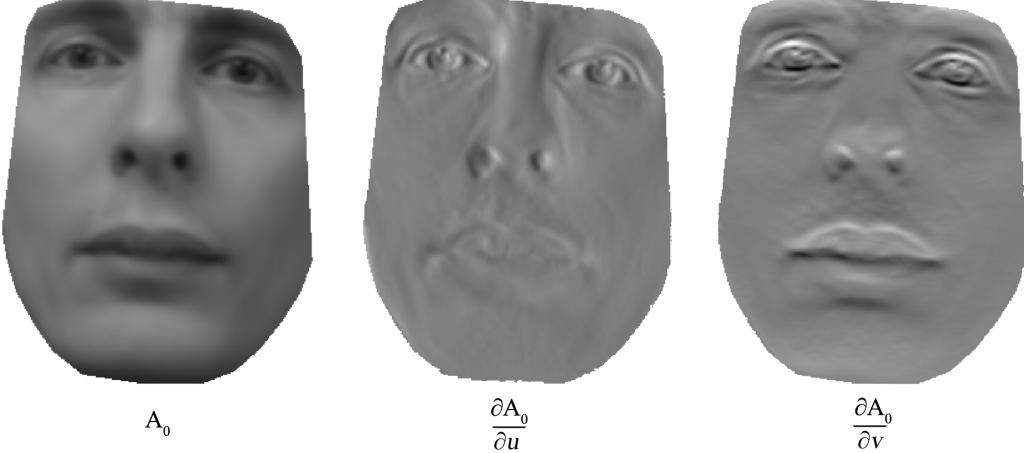


Figura 3.7. Esempio di gradiente dell'apparenza media: valori scuri indicano valori negativi, mentre i valori più chiari indicano valori positivi. Sono state usate immagini monocromatiche per facilitare la presentazione, ma il gradiente può essere facilmente calcolato anche in presenza di colori trattando ogni canale separatamente.

### 3.3.3 Soluzione Incrementale

In modo analogo al problema di allineamento di immagini, dalle immagini di steepest descent si ricava l'Hessiana:

$$\mathbf{H}_{(j,k)} = \sum_{\mathbf{u} \in \mathbf{s}_0} SD_j(\mathbf{u}) SD_k(\mathbf{u}) \quad (3.32)$$

dove  $\mathbf{H}_{(j,k)}$  indica la posizione  $(j,k)$  della matrice  $\mathbf{H}$ .

E la soluzione incrementale in forma chiusa del problema di fitting è data da:

$$\begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{p} \end{bmatrix} = \mathbf{H}^{-1} \mathbf{K} \quad (3.33)$$

che è analoga alla Formula 3.12, con le componenti di  $\mathbf{K}$  ottenute proiettando il residuo sulle immagini di steepest descent:

$$K_i = \sum_{\mathbf{u} \in \mathbf{s}_0} SD_i(\mathbf{u}) [I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) - A_0(\mathbf{u})] \quad (3.34)$$

Osservando che la regola di aggiornamento è ora data da:

$$\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q}) \leftarrow (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) \circ (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \Delta \mathbf{q}, \Delta \mathbf{p})^{-1}) \quad (3.35)$$

Le prossime sotto-sezioni si occuperanno di definire in che modo l'introduzione di una trasformazione globale influenzi la composizione e l'inversione.

### 3.3.4 Inversione del Warp Incrementale

Nella Sezione 3.2.2 si è visto che un'approssimazione del primo ordine di  $\mathbf{W}(\mathbf{u}; \Delta \mathbf{p})^{-1}$  è  $\mathbf{W}(\mathbf{u}; -\Delta \mathbf{p})$ . Procedendo in modo analogo si dimostra che:

$$\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \Delta \mathbf{q}, \Delta \mathbf{p})^{-1} \approx \mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta \mathbf{q}, -\Delta \mathbf{p}) \quad (3.36)$$

sempre con un'approssimazione del primo ordine.

### 3.3.5 Composizione con il Warp Corrente

Applicando l'approssimazione definita dalla Formula 3.36, è possibile riformulare la regola di aggiornamento nel modo seguente:

$$\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q}) \leftarrow (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) \circ (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta \mathbf{q}, -\Delta \mathbf{p})) \quad (3.37)$$

Nonostante questa possa sembrare un'operazione particolarmente complicata, la procedura è analoga a quella usata per la composizione in assenza di trasformazione globale. Si determina innanzitutto il valore di  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{q}, -\Delta \mathbf{p})$  calcolando:

$$\mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{p}) = \mathbf{s}_0 - \sum_{i=1}^m \Delta p_i \mathbf{s}_i \quad (3.38)$$

e:

$$\mathbf{N}(\mathbf{s}_0; -\Delta \mathbf{q}) = \mathbf{s}_0 - \sum_{i=1}^4 \Delta q_i \mathbf{s}_i^* \quad (3.39)$$

per poi ricavare la trasformazione affine associata a  $\mathbf{N}(\mathbf{s}_0; -\Delta \mathbf{q})$  e applicarla a  $\mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{p})$ . Si può quindi applicare la procedura di composizione descritta nella Sezione 3.2.3 alla forma associata al warp  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{q}, -\Delta \mathbf{p})$  e a quella già conosciuta, associata a  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})$ .

## 3.4 L'Algoritmo Completo

Per quanto si sia cercato di rendere il più possibile lineare e semplice la spiegazione dei vari passi che coinvolgono il funzionamento dell'algoritmo, è normale che non possa essere immediatamente evidente il ruolo di alcune operazioni, così come non potrebbe essere perfettamente chiaro quali operazioni siano indipendenti dall'immagine test  $I$  (e quindi pre-calcolabili durante il training) e quali non lo siano.

Segue una rappresentazione schematica delle fasi di training e fitting degli AAM con composizione inversa, ricordando che alcuni dettagli del training sono stati descritti nel capitolo precedente, e che i dettagli implementativi finora omessi saranno descritti nel Capitolo 6:

**Training:**

1. Determina la forma media  $\mathbf{s}_0$  e le mode di forma  $\mathbf{s}_i$  usando le procedure descritte nella Sezione 2.1.
2. Determina i vettori  $\mathbf{s}_j^*$  (come descritto nella Sezione 3.2.4) e ortonormalizza l'insieme  $[\mathbf{s}_1^* \dots \mathbf{s}_4^* \mathbf{s}_1 \dots \mathbf{s}_m]$ . I risultati ottenuti saranno i nuovi  $\mathbf{s}_j^*$  e  $\mathbf{s}_i$ .
3. Determina l'apparenza media  $A_0$  e le mode di apparenza  $A_i$  usando il procedimento descritto nella Sezione 2.2.
4. Calcola il gradiente dell'apparenza media  $\nabla A_0$ .
5. Stima le Jacobiane  $\frac{\partial \mathbf{N}}{\partial \mathbf{q}}$  e  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  e le immagini di steepest descent  $SD_i$ , come descritto nella Sezione 3.3.2.
6. Determina l'Hessiana inversa  $\mathbf{H}^{-1}$  a partire dalla Formula 3.32.

**Fitting:**

1. Data la forma iniziale  $\mathbf{s}_{init}$  pon  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; \mathbf{p}, \mathbf{q}) = \mathbf{s}_{init}$ , a indicare che  $\mathbf{s}_{init}$  è la forma associata al warp corrente.
2. Determina l'immagine di errore  $E(\mathbf{u}) = I(\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) - A_0(\mathbf{u})$  effettuando un warp su  $I$  che mappi la porzione di immagine dal sistema di riferimento della forma  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; \mathbf{p}, \mathbf{q})$  a quello della forma media  $\mathbf{s}_0$ .
3. Calcola il valore di  $\mathbf{K}$  usando la Formula 3.34.
4. Ottieni i valori incrementali  $\Delta \mathbf{q}$  e  $\Delta \mathbf{p}$  moltiplicando l'Hessiana inversa per  $\mathbf{K}$  (Formula 3.33).
5. Aggiorna il warp corrente:  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q}) \leftarrow (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) \circ (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta \mathbf{q}, -\Delta \mathbf{p}))$  applicando le procedure descritte nella Sezione 3.3.5.
6. Ripeti dal punto (2) finché non si ha convergenza, o il numero massimo di iterazioni è stato raggiunto.

CAPITOLO 3. ACTIVE APPEARANCE MODELS RIVISITATI

## Capitolo 4

# Active Appearance Models 2D+3D in Tempo Reale

Tra le tante possibili classi di oggetti deformabili per cui trovano applicazione gli AAM, spiccano senza dubbio i volti umani. Se ci si limita a questo campo, si nota che poco dopo l'avvento degli AAM sono stati sviluppati i 3D Morphable Models (3DMM)[6], anch'essi dei modelli generativi, però incentrati sull'informazione 3D. I 3DMM non possono comunque essere considerati come la “versione 3D” degli AAM, in quanto essi richiedono informazioni molto dense per operare al meglio (tipicamente provenienti da *range scan*) con le conseguenti difficoltà inerenti alla raccolta del training set e il maggior costo computazionale dovuto a una rappresentazione meno compatta.

Si vedrà come, usando un approccio alternativo, Matthews et al. [26] siano riusciti ad estendere gli AAM con composizione inversa in modo da integrare un modello 3D analogo a quello dei 3DMM, senza particolari perdite di efficienza, ottenendo quelli che hanno denominato “AAM 2D+3D”. Essi hanno inoltre dimostrato che un modello di forma 2D ha le stesse capacità di rappresentazione di un modello 3D, al costo di un maggior numero di parametri di forma.

Prevedibilmente, si rivela comunque necessario fornire all'algoritmo 2D+3D un certo tipo di informazione tridimensionale (nello specifico, il modello di forma 3D), e pretendere che questa sia fornita dall'utente scoraggerebbe o renderebbe impossibili molti potenziali usi dell'algoritmo. Per questo motivo, si vedrà come applicare un algoritmo di *ricostruzione della forma dal movimento* (shape from motion recovery) [43] a dati prodotti dal fitting 2D per ricavare le mode del modello di forma 3D.

Un algoritmo di ricostruzione della forma dal movimento, data una sequenza di immagini annotate che rappresentano l'oggetto sottoposto a trasformazioni rigide e non, ripristina la struttura tridimensionale dell'oggetto. Questo processo è possibile a patto che i dati in ingresso rispettino le seguenti condizioni: l'oggetto è sufficientemente vicino (e quindi la distorsione prospettica è

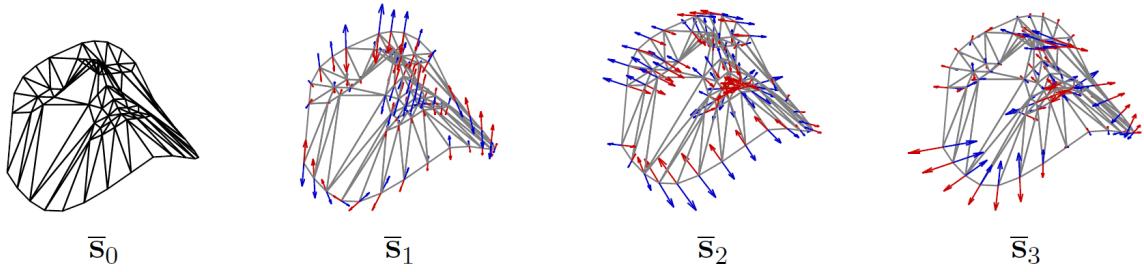


Figura 4.1. Esempio di modello di forma di un 3DMM: le mode di forma possono essere viste come deformazioni della forma media. Dato lo stesso training set, il numero di mode 3D è in genere molto inferiore rispetto al numero di mode 2D. Questo sia per merito della terza dimensione, sia per il fatto che le deformazioni sono indipendenti dall'operazione di proiezione usata [26].

limitata), la quantità e la qualità delle deformazioni presenti nei dati forniti è tale da non rendere il problema mal-condizionato e, ovviamente, i parametri ottici dei dispositivi usati per catturare le immagini sono gli stessi.

Si sottolinea che, essendo l'algoritmo 2D+3D un'estensione dell'algoritmo con composizione inversa, si farà estensivamente uso di riferimenti a formule e tecniche descritte nel capitolo precedente, assumendo che il lettore abbia compreso appieno quei concetti.

## 4.1 Il Modello di Forma 3D

Il modello di forma 3D che verrà trattato in seguito è lineare e del tutto analogo al modello di forma 2D usato dagli AAM. L'unica differenza pratica è che non si dispone in genere dei dati di training 3D, e quindi le mode di forma non possono essere determinate con una semplice analisi statistica.

### 4.1.1 Definizione

In modo analogo alle forme 2D, una generica forma 3D  $\bar{x}$  è rappresentabile in forma matriciale:

$$\bar{x} = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \\ z_1 & \dots & z_n \end{bmatrix} \quad (4.1)$$

in cui si assume ci sia corrispondenza tra le annotazioni 2D e quelle 3D.

#### 4.1. IL MODELLO DI FORMA 3D

Anche nel caso dei 3DMM, si identificano una forma media  $\bar{\mathbf{s}}_0$  e delle mode di forma 3D  $\bar{\mathbf{s}}_1 \dots \bar{\mathbf{s}}_{\bar{m}}$ , allo scopo di costituire il modello lineare:

$$\bar{\mathbf{s}} = \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \quad (4.2)$$

dove  $\bar{p}_1 \dots \bar{p}_{\bar{m}}$  sono i parametri di forma 3D. Questo modello è in realtà solo un piccolo sottoinsieme dei 3DMM, ma è tutto ciò a cui si è interessati in questa sede.

Allo scopo di mettere in relazione le forme 3D con quelle 2D, si consideri un'operazione di proiezione conforme al modello prospettico debole, realizzata attraverso una matrice  $\mathbf{P}$ :

$$\mathbf{P} = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \quad (4.3)$$

e una traslazione dall'origine  $[o_u \ o_v]^T$ .

Il modello prospettico debole comporta che i due assi di proiezione  $\mathbf{i} = [i_x \ i_y \ i_z]$  e  $\mathbf{j} = [j_x \ j_y \ j_z]$  siano tali per cui  $\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j}$  e  $\mathbf{i} \cdot \mathbf{j} = 0$ . Essi sono quindi ortonormali a meno di un fattore di scala.

L'operazione di proiezione è standard ed è semplicemente data da:

$$\mathbf{x} = \mathbf{P} \bar{\mathbf{x}} + \begin{bmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{bmatrix} \quad (4.4)$$

#### 4.1.2 Equivalenza tra Modelli di Forma 3D e 2D

Intuitivamente, verrebbe da pensare che un modello di forma 3D sia più “potente” di un modello 2D. Ma in verità, considerando tutte le possibili proiezioni del tipo prospettico debole, esiste sempre un modello 2D che può generare le stesse forme di un dato modello 3D. La differenza sostanziale è data dal fatto che un modello 2D in grado di fare ciò ha molti più gradi di libertà del modello 3D, e quindi può generare forme che non costituiscono deformazioni “ valide” per l'oggetto considerato.

Prima di dimostrare tutto ciò, si semplifica il modello di proiezione eliminando la traslazione dall'origine  $[o_u \ o_v]^T$ , in quanto questa può essere modellata in un AAM dalla trasformazione globale  $\mathbf{N}(\mathbf{u}; \mathbf{q})$ . La variabilità 2D del modello 3D è di conseguenza data da:

$$\begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \left( \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) \quad (4.5)$$

dove  $i_x, i_y, i_z, j_x, j_y, j_z$  e i parametri di forma 3D  $\bar{p}_1 \dots \bar{p}_{\bar{m}}$  possono variare liberamente tra tutti i valori consentiti.

Riscrivendo la matrice di proiezione come somma di 6 matrici:

$$\begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} = i_x \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + i_y \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + i_z \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + j_x \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + j_y \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + j_z \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.6)$$

la Formula 4.5 diventa una combinazione lineare delle forme:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, \quad (4.7)$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \bar{s}_i, \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \bar{s}_i, \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{s}_i.$$

per  $i = 0 \dots \bar{m}$ . In questo modo si vede che la variabilità 2D del modello 3D può essere rappresentata dalla combinazione lineare delle forme 2D:

$$\begin{aligned} \bar{s}_{6i+1} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, & \bar{s}_{6i+2} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, & \bar{s}_{6i+3} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \bar{s}_i, \\ \bar{s}_{6i+4} &= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \bar{s}_i, & \bar{s}_{6i+5} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \bar{s}_i, & \bar{s}_{6i+6} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{s}_i. \end{aligned} \quad (4.8)$$

per  $i = 0 \dots \bar{m}$  (quindi includendo la forma media  $\bar{s}_0$ ), indicando che sono necessarie al più 6 ( $\bar{m} + 1$ ) mode di forma 2D per riprodurre la stessa variazione di un modello 3D con  $\bar{m}$  mode di forma. Da questo si deduce che, allo scopo di modellare variazioni 2D, un modello 3D è sicuramente più compatto, ma di certo non più potente di un modello 2D. In pratica, comunque, si osserva il numero di mode di forma 2D è solitamente due o tre volte tanto, non sei.

Tuttavia, come dimostrato in Matthews et al. [26], il maggior numero di gradi di libertà del modello 2D permette la generazione di istanze di forme irrealizzabili dal modello 3D, e che sono quindi inconsistenti con l'effettiva geometria dell'oggetto considerato. Questo fenomeno ha anche la conseguenza di rendere il fitting maggiormente soggetto a minimi locali, con una perdita di robustezza.

## 4.2 Ricostruzione della Forma dal Movimento

Date  $N+1$  forme 2D  $\mathbf{s}^0 \dots \mathbf{s}^N$ , il problema di ricostruzione della forma dal movimento consiste nel determinare la miglior combinazione di forme tridimensionali  $\bar{\mathbf{s}}^0 \dots \bar{\mathbf{s}}^N$  e di operazioni di proiezione che le riproducano:

$$\mathbf{s}^i \approx \mathbf{P}_i \bar{\mathbf{s}}^i + \begin{bmatrix} o_u^i & \dots & o_u^i \\ o_v^i & \dots & o_v^i \end{bmatrix} \quad (4.9)$$

dove  $\mathbf{P}_i$  è una matrice di proiezione del tipo prospettico debole e  $[o_u^i \ o_v^i]^T$  è una traslazione dall'origine.

È importante considerare che, come dice il nome, un algoritmo di questo tipo richiede un'ampia varietà di rappresentazioni bidimensionali per poter ricostruire con accuratezza l'oggetto, e addirittura alcuni algoritmi si aspettano che ci sia coerenza temporale tra una forma e la successiva della sequenza. È per questo motivo che di norma si usano sequenze video dell'oggetto come ingresso in questa tipologia di algoritmi, ma questo pone l'ovvio problema di dover annotare un'enorme quantità di immagini.

In questo frangente l'uso di un AAM si rivela estremamente utile: infatti, oltre ad aiutare nell'automatizzazione del processo di annotazione, non è fonte del rumore tipicamente introdotto da un processo di annotazione manuale (aspetto importante, considerando che gli algoritmi di ricostruzione sono spesso sensibili al rumore nei dati di ingresso).

Nel caso di oggetti rigidi, il problema di ricostruzione è relativamente semplice e sono disponibili molti algoritmi affidabili, soprattutto se si considerano proiezioni ortogonali [37]. Infatti, sapendo di avere a che fare con un oggetto rigido, si può assumere che la forma 3D sia unica, e che tutto quello che occorre stimare sono le operazioni di proiezione. Lo stesso non si può dire del caso non-rigido di nostro interesse, in cui la forma 3D cambia a causa di deformazioni, aumentando in modo significativo il numero di incognite e rendendo la soluzione del problema sostanzialmente più difficile.

Così, anche in questo tipo di applicazioni, si è diffuso l'uso di un modello lineare di forma con lo scopo di ottenere una rappresentazione semplice e compatta di oggetti 3D deformabili:

$$\bar{\mathbf{s}} = \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \quad (4.10)$$

(modello di forma 3D che è già stato introdotto in precedenza).

Un'algoritmo di ricostruzione di oggetti deformabili può quindi basarsi sulla conoscenza a priori del modello di forma 3D (eliminando i vettori  $\bar{\mathbf{s}}_0 \dots \bar{\mathbf{s}}_{\bar{m}}$  dalle incognite), oppure può tentare di stimare simultaneamente il modello di forma 3D e le operazioni di proiezione. Considerando che ciò a cui si è interessati nel nostro caso è proprio l'informazione 3D, non ha senso supporre di avere a disposizione i dati che permetterebbero di generare un modello deformabile 3D dell'oggetto.

Dei vari algoritmi in grado di stimare simultaneamente modello deformabile e operazioni di proiezione proposti, quello di Bregler et al. [9] è stato probabilmente il primo ad affrontare il problema, tramite fattorizzazione della matrice delle misure e uso di vincoli sulle matrici di proiezione. Successivamente, Torresani et al. [38] hanno proposto una soluzione basata su un'approccio probabilistico di Expectation-Maximization, strutturato su 3 fasi in cui un tipo di incognite è stimato mentre le altre rimangono invariate. Un approccio non-lineare simile è stato adottato anche da Brand [8], anche se entrambi i metodi usano soltanto *vincoli di rotazione* sulle matrici di proiezione. Xiao et al. [43] hanno dimostrato che i vincoli di rotazione non sono sufficienti, e grazie all'introduzione di ulteriori *vincoli di base*, hanno ottenuto una soluzione in forma chiusa al problema di ricostruzione della forma dal movimento.

Segue la descrizione dell'algoritmo ricostruzione di Xiao et al., dove si ometteranno molti teoremi e dimostrazioni che appesantirebbero una lettura già di per sé molto tecnica. Il lettore interessato può sempre rivolgersi all'articolo di Xiao et al. [43].

#### 4.2.1 Fattorizzazione della Matrice delle Misure

Se si organizzano in una matrice le forme 2D che rappresentano l'oggetto di interesse in movimento, si ottiene quella che viene chiamata *matrice delle misure* (measurements matrix):

$$\mathbf{W} = \begin{bmatrix} u_1^1 & u_2^1 & \dots & u_n^1 \\ v_1^1 & v_2^1 & \dots & v_n^1 \\ \vdots & \vdots & \vdots & \vdots \\ u_1^N & u_2^N & \dots & u_n^N \\ v_1^N & v_2^N & \dots & v_n^N \end{bmatrix} = \begin{bmatrix} \mathbf{s}^1 \\ \vdots \\ \mathbf{s}^N \end{bmatrix} \quad (4.11)$$

Si osservi che se le forme 2D possono essere generate a partire da  $\bar{m}$  mode di forma 3D, dev'essere possibile la seguente fattorizzazione:

$$\mathbf{W} = \mathbf{MB} + \mathbf{t} = \begin{bmatrix} \mathbf{P}_0 & p_1^0 \mathbf{P}_0 & \dots & p_{\bar{m}}^0 \mathbf{P}_0 \\ \mathbf{P}_1 & p_1^1 \mathbf{P}_1 & \dots & p_{\bar{m}}^1 \mathbf{P}_1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_N & p_1^N \mathbf{P}_N & \dots & p_{\bar{m}}^N \mathbf{P}_N \end{bmatrix} \begin{bmatrix} \bar{\mathbf{s}}_0 \\ \vdots \\ \bar{\mathbf{s}}_{\bar{m}} \end{bmatrix} + \begin{bmatrix} o_u^1 & \dots & o_u^1 \\ o_v^1 & \dots & o_v^1 \\ \vdots & \vdots & \vdots \\ o_u^N & \dots & o_u^N \\ o_v^N & \dots & o_v^N \end{bmatrix} \quad (4.12)$$

e ciò si dimostra sostituendo  $\bar{\mathbf{s}}^i = \sum_{j=1}^{\bar{m}} p_j^i \bar{\mathbf{s}}_j$  nella Formula 4.9.

La presenza della componente di traslazione  $\mathbf{t}$  rende difficoltosa la fattorizzazione di  $\mathbf{W}$ , per questo motivo si fa in modo di centrare tutte le forme  $\mathbf{s}^i$  nell'origine, per poi costruire la matrice delle misure *registrata*  $\tilde{\mathbf{W}}$ . Così facendo, la componente di traslazione non è più necessaria e  $\tilde{\mathbf{W}}$  è fattorizzabile nelle matrici  $\tilde{\mathbf{M}}$  e  $\tilde{\mathbf{B}}$ , di dimensione  $2(N+1) \times 3(\bar{m}+1)$  e  $3(\bar{m}+1) \times n$ , rispettivamente.

## 4.2. RICOSTRUZIONE DELLA FORMA DAL MOVIMENTO

Considerando che il numero di mode di forma 3D  $\bar{m}$  è generalmente piccolo, si osserva che il rango di  $\tilde{\mathbf{W}}$  è al più  $3(\bar{m} + 1)$ , e da questo si deduce la seguente relazione:

$$\bar{m} + 1 \leq \left\lfloor \frac{\text{rank}(\tilde{\mathbf{W}})}{3} \right\rfloor \quad (4.13)$$

Una prima stima di  $\tilde{\mathbf{M}}$  e  $\tilde{\mathbf{B}}$  si determina a partire dalla Singular Value Decomposition (SVD) di  $\tilde{\mathbf{W}}$ :

$$\begin{aligned} \tilde{\mathbf{M}} &= \mathbf{U}_{2(N+1) \times 3(\bar{m}+1)} \mathbf{D}_{3(\bar{m}+1) \times 3(\bar{m}+1)} \\ \tilde{\mathbf{B}} &= (\mathbf{V}_{n \times 3(\bar{m}+1)})^T \end{aligned} \quad (4.14)$$

con  $\tilde{\mathbf{W}} = \mathbf{U} \mathbf{D} \mathbf{V}^T$  e dove  $\mathbf{Y}_{a \times b}$  indica la sottomatrice di  $\mathbf{Y}$  di dimensione  $a \times b$  costituita dalle righe di indice  $1, 2, \dots, a$  e dalle colonne di indice  $1, 2, \dots, b$ .

La fattorizzazione non è univoca, è infatti sufficiente una qualsiasi matrice  $3(\bar{m} + 1) \times 3(\bar{m} + 1)$  non singolare  $\mathbf{G}$  per ottenere una nuova fattorizzazione valida:  $\tilde{\mathbf{W}} = \tilde{\mathbf{M}} \mathbf{G} \mathbf{G}^{-1} \tilde{\mathbf{B}}$ .

Per questo motivo, l'algoritmo ha come obiettivo la stima di una matrice  $\mathbf{G}$ , detta *correttiva*, con cui trasformare le  $\tilde{\mathbf{M}}$  e  $\tilde{\mathbf{B}}$  iniziali:

$$\begin{aligned} \mathbf{M} &= \tilde{\mathbf{M}} \mathbf{G} \\ \mathbf{B} &= \mathbf{G}^{-1} \tilde{\mathbf{B}} \end{aligned} \quad (4.15)$$

applicando specifici *vincoli* su  $\mathbf{G}$  per fare in modo che  $\mathbf{M}$  sia costituito da matrici di proiezioni pesate (Formula 4.12), oltre che soddisfare  $\tilde{\mathbf{W}} = \mathbf{M} \mathbf{B}$ .

### 4.2.2 Vincoli di Rotazione

Se si rappresenta la matrice correttiva con  $\bar{m} + 1$  blocchi:  $\mathbf{G} = [\mathbf{g}_0 \dots \mathbf{g}_{\bar{m}}]$ , ognuno di dimensione  $3(\bar{m} + 1) \times 3$ , la relazione tra  $\mathbf{M}$  e  $\tilde{\mathbf{M}}$  espressa dalla Formula 4.15 può essere così riscritta:

$$\tilde{\mathbf{M}} \mathbf{g}_k = \begin{bmatrix} p_k^0 \mathbf{P}_0 \\ p_k^1 \mathbf{P}_1 \\ \vdots \\ p_k^N \mathbf{P}_N \end{bmatrix}, \quad k = 0 \dots \bar{m} \quad (4.16)$$

che sono le  $\bar{m} + 1$  “tri-colonne” di  $\mathbf{M}$ .

Per assicurare che le matrici di proiezione in  $\mathbf{M}$  siano conformi al modello prospettico debole, ogni  $\mathbf{g}_k$  deve fare in modo che le seguenti relazioni siano valide:

$$\left(p_k^j\right)^2 \mathbf{i}_j \cdot \mathbf{i}_j = \left(p_k^j\right)^2 \mathbf{j}_j \cdot \mathbf{j}_j \quad (4.17)$$

(stessa norma degli assi) e:

$$\left(p_k^j\right)^2 \mathbf{i}_j \cdot \mathbf{j}_j = 0 \quad (4.18)$$

(ortogonalità), per  $j = 0 \dots N$ . Dove:

$$\mathbf{P}_j = \begin{bmatrix} \mathbf{i}_j \\ \mathbf{j}_j \end{bmatrix} \quad (4.19)$$

Per imporre questi *vincoli di rotazione* si osserva che  $p_k^j \mathbf{i}_j = \tilde{\mathbf{M}}_{2j+1} \mathbf{g}_k$  e  $p_k^j \mathbf{j}_j = \tilde{\mathbf{M}}_{2j+2} \mathbf{g}_k$  (dove  $\tilde{\mathbf{M}}_i$  è l'i-esima riga di  $\tilde{\mathbf{M}}$ ). Si possono quindi riscrivere le Formule 4.17 e 4.18 nel modo seguente:

$$\tilde{\mathbf{M}}_{2j+1} \mathbf{g}_k \left( \tilde{\mathbf{M}}_{2j+1} \mathbf{g}_k \right)^T = \tilde{\mathbf{M}}_{2j+2} \mathbf{g}_k \left( \tilde{\mathbf{M}}_{2j+2} \mathbf{g}_k \right)^T \quad (4.20)$$

$$\tilde{\mathbf{M}}_{2j+1} \mathbf{g}_k \left( \tilde{\mathbf{M}}_{2j+2} \mathbf{g}_k \right)^T = 0 \quad (4.21)$$

da cui, ponendo  $\mathbf{Q}_k = \mathbf{g}_k \mathbf{g}_k^T$  si determina:

$$\tilde{\mathbf{M}}_{2j+1} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+1}^T = \tilde{\mathbf{M}}_{2j+2} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+2}^T \quad (4.22)$$

$$\tilde{\mathbf{M}}_{2j+1} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+2}^T = 0 \quad (4.23)$$

Grazie a questi passaggi, il problema relativamente astratto di assicurare che la matrice  $\mathbf{M}$  sia costituita da matrici di proiezione (pesate) conformi al modello prospettico debole, è stato ridotto al problema concreto di determinare  $\bar{m} + 1$  matrici  $\mathbf{Q}_k$ , risolvendo dei sistemi lineari.

Le matrici  $\mathbf{Q}_k$  sono simmetriche, e ognuna di esse ha  $(9(\bar{m} + 1)^2 + 3(\bar{m} + 1)) / 2$  incognite, quindi si potrebbe pensare che per determinarle è sufficiente che valga  $2(N + 1) \geq (9(\bar{m} + 1)^2 + 3(\bar{m} + 1)) / 2$ . In realtà, come dimostrato da Xiao et al. [43], i  $2(N + 1)$  vincoli sono sufficienti solo nel caso rigido, e sono necessari altri vincoli per assicurare l'unicità della soluzione nel caso non-rigido.

### 4.2.3 Vincoli di Base

Anche se dal punto di vista matematico la cosa non è immediatamente evidente, si potrebbe comunque intuire perché i vincoli di rotazione permettono di determinare univocamente solo le  $N + 1$  matrici di proiezione, lasciando ambigua la soluzione dei parametri del modello deformabile 3D. Infatti, per costruzione, i vincoli di rotazione hanno il solo scopo di assicurare che le matrici di proiezione siano conformi al modello prospettico debole.

## 4.2. RICOSTRUZIONE DELLA FORMA DAL MOVIMENTO

È possibile rimuovere parte dell'ambiguità nella stima del modello deformabile 3D assumendo che le prime  $\bar{m} + 1$  forme 3D rappresentate nella sequenza in ingresso siano le basi  $\bar{s}_0 \dots \bar{s}_{\bar{m}}$ . Questa assunzione non si verifica in generale, ma è sufficiente riordinare la sequenza in modo che le prime  $\bar{m} + 1$  forme 2D siano indipendenti. Un modo per trovare  $\bar{m} + 1$  forme indipendenti consiste nel prendere un insieme di forme, organizzarle in forma matriciale (in modo analogo a  $\mathbf{W}$ ) e calcolarne il numero di condizionamento. Minore è il numero di condizionamento, maggiore è il grado di indipendenza delle forme 2D (e auspicabilmente, anche delle forme 3D).

Se le prime  $\bar{m} + 1$  forme bidimensionali rappresentano la proiezione delle basi del modello di forma 3D, si verifica facilmente che deve valere:

$$\begin{aligned} p_i^i &= \frac{1}{|\mathbf{P}_i|}, \quad i = 0, \dots, \bar{m} \\ p_j^i &= 0, \quad i, j = 0, \dots, \bar{m}, i \neq j \end{aligned} \quad (4.24)$$

dove  $|\mathbf{P}| = i_x i_x + i_y i_y + i_z i_z$ .

Considerando nuovamente la  $k$ -esima tri-colonna di  $\mathbf{M}$ , questo implica che le seguenti relazioni sugli assi delle matrici di proiezione sono vere:

$$\begin{aligned} p_k^i p_k^j \mathbf{i}_i \cdot \mathbf{i}_j &= 1 \\ p_k^i p_k^j \mathbf{j}_i \cdot \mathbf{j}_j &= 1 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i = j = k \quad (4.25)$$

$$\begin{aligned} p_k^i p_k^j \mathbf{i}_i \cdot \mathbf{i}_j &= 0 \\ p_k^i p_k^j \mathbf{j}_i \cdot \mathbf{j}_j &= 0 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i \neq k \quad (4.26)$$

$$\begin{aligned} p_k^i p_k^j \mathbf{i}_i \cdot \mathbf{j}_j &= 0 \\ p_k^j p_k^i \mathbf{j}_i \cdot \mathbf{i}_j &= 0 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i \neq k \text{ oppure } i = j = k \quad (4.27)$$

dove  $i = 0 \dots \bar{m}$  e  $j = 0 \dots N$ . In pratica, se si riuscissero ad imporre anche questi vincoli sulle matrici di proiezione, si otterrebbe che le prime  $\bar{m} + 1$  forme 3D rappresenterebbero una base per il modello di forma. Queste relazioni hanno forma simile alle Formule 4.17 e 4.18 e possono essere riarrangiate in modo analogo, come vincoli su  $\mathbf{Q}_k$ :

$$\begin{aligned} \tilde{\mathbf{M}}_{2i+1} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+1} &= 1 \\ \tilde{\mathbf{M}}_{2i+2} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+2} &= 1 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i = j = k \quad (4.28)$$

$$\begin{aligned} \tilde{\mathbf{M}}_{2i+1} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+1} &= 0 \\ \tilde{\mathbf{M}}_{2i+2} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+2} &= 0 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i \neq k \quad (4.29)$$

$$\begin{aligned} \tilde{\mathbf{M}}_{2i+1} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+2} &= 0 \\ \tilde{\mathbf{M}}_{2i+2} \mathbf{Q}_k \tilde{\mathbf{M}}_{2j+1} &= 0 \end{aligned} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad i \neq k \text{ oppure } i = j = k \quad (4.30)$$

con  $i = 0 \dots \bar{m}$  e  $j = 0 \dots N$ . Questi sono detti *vincoli di base* (base constraints) e, in associazione ai vincoli di rotazione, permettono di determinare in modo univoco le  $\bar{m} + 1$  matrici  $\mathbf{Q}_k$ .

#### 4.2.4 Determinare $\mathbf{G}$

Allo scopo di determinare la matrice correttiva  $\mathbf{G}$ , è necessario conoscere le sue tri-colonne  $\mathbf{g}_k$ . Ricordando che  $\mathbf{Q}_k = \mathbf{g}_k \mathbf{g}_k^T$ , dalla decomposizione SVD  $\mathbf{Q}_k = \mathbf{U} \mathbf{D} \mathbf{V}^T$  si osserva che:

$$\mathbf{g}_k = \mathbf{U} \mathbf{D}^{\frac{1}{2}} \quad (4.31)$$

in quanto  $\mathbf{U} = \mathbf{V}$ , vista la simmetria di  $\mathbf{Q}_k$ .

Essendo la stima di ogni matrice  $\mathbf{g}_k$  prodotto della soluzione di sistemi lineari diversi, si osserva che esse sono espresse in spazi di coordinate differenti. Quindi, prima di poter ricostruire la matrice correttiva è necessario fare in modo che le sue tri-colonne siano in uno spazio di coordinate comune, usando il metodo di analisi procustiana ortogonale [32]. Questo richiede innanzitutto le  $\bar{m} + 1$  versioni delle matrici di proiezione, che si ottengono separandole dai vari pesi  $p_j^i$  e ricordando che le matrici avranno un significato solo in corrispondenza di pesi non nulli.

Gli assi delle matrici di proiezione sono poi ortonormalizzati, per correggere eventuali errori introdotti da procedure numeriche.

Nella separazione tra pesi e matrici di proiezione diventa poi importante determinare il segno dei pesi, in quanto l'uso di un segno errato comporta un drastico cambiamento nella matrice di proiezione associata. Un metodo per stimare questi segni può essere ad esempio l'uso di una misura di correlazione tra matrici di proiezione corrispondenti.

Una volta stimate correttamente le varie versioni delle matrici di proiezione, e con il corretto segno, è possibile definire quelle di riferimento da usare per il metodo di analisi procustiana ortogonale. Se  $\mathbf{P}_{ref}$  è la matrice di riferimento, si è interessati a determinare la matrice ortogonale  $\mathbf{R}$  che meglio mappa una qualsiasi matrice  $\mathbf{P}$  su  $\mathbf{P}_{ref}$ . Nello specifico si intende minimizzare:

$$\|\mathbf{P}\mathbf{R} - \mathbf{P}_{ref}\|_F \quad (4.32)$$

dove  $\mathbf{R}\mathbf{R}^T = \mathbf{I}$  e  $\|\cdot\|_F$  è la norma di Frobenius.

La soluzione del problema di analisi procustiana ortogonale, data la decomposizione SVD  $\mathbf{P}^T \mathbf{P}_{ref} = \mathbf{U} \mathbf{D} \mathbf{V}^T$  è:

$$\mathbf{R} = \mathbf{U} \mathbf{V}^T \quad (4.33)$$

Per maggiori dettagli implementativi relativi a questa parte si rimanda alla Sezione 6.4.

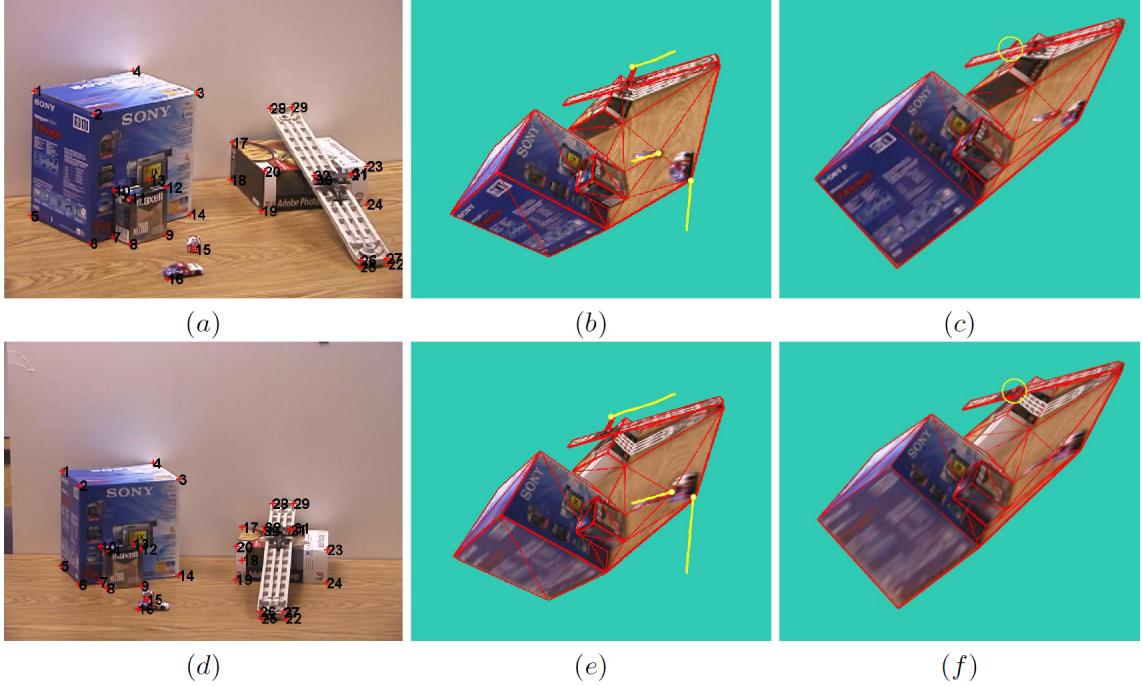


Figura 4.2. Esempio di ricostruzione del movimento di 3 oggetti su sfondo statico. (a) e (d) illustrano 2 delle forme in ingresso all'algoritmo; (b) e (e) sono le corrispondenti ricostruzioni 3D. (c) e (f) sono invece state prodotte dall'algoritmo di Brand et al. [8] che applica soltanto i vincoli di rotazione, e mostrano come l'aeroplano (area cerchiata) sia stato erroneamente posizionato sotto la rampa [43].

## 4.3 Fitting 2D+3D in Tempo Reale

Le estensioni 3D agli AAM con composizione inversa non comportano modifiche alla fase di training, che rimane quella descritta nel Capitolo 3.

La fase di fitting è ancora un problema di ottimizzazione locale e necessita per questo di una stima della forma 3D iniziale e della proiezione iniziale, oltre che delle mode di forma 3D. Il procedimento rimane comunque grossomodo invariato, e si rende necessaria solo la stima di alcune Jacobiane aggiuntive e di una nuova Hessiana. E nonostante questo comporti un costo per iterazione maggiore, la maggiore velocità di convergenza dell'algoritmo fa in modo che siano richieste meno iterazioni per arrivare alla soluzione del problema.

### 4.3.1 Vincoli di Coerenza 3D

Matthews et al. [26] hanno dimostrato che la composizione inversa nel caso 3D è possibile solo a patto che ad ogni iterazione si ricalcoli un diverso gradiente 3D dell'immagine, e quindi non

rappresenta una scelta efficiente per il fitting in tempo reale. È possibile però aggirare questa limitazione riformulando il problema in modo da poter continuare a usare la composizione inversa.

Ricordando che l'operazione di fitting 2D consiste nel determinare i parametri dell'AAM che minimizzino l'errore immagine:

$$\sum_{\mathbf{u} \in \mathbf{s}_0} \left[ A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) \right]^2 \quad (4.34)$$

si potrebbe pensare di vincolare il problema di ottimizzazione in modo da assicurare che la forma 2D sia coerente con il modello 3D dell'oggetto:

$$\min_{\mathbf{P}, \bar{\mathbf{p}}} \left\| \mathbf{P} \left( \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) + \begin{bmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{bmatrix} - \mathbf{N} \left( \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q} \right) \right\|^2 = 0 \quad (4.35)$$

dove  $\|\cdot\|^2$  indica la somma dei quadrati degli elementi.

Le uniche quantità che non sono conosciute ( $m, \bar{m}, \bar{\mathbf{s}}_i, \mathbf{s}_i$ ) o ottimizzate ( $\mathbf{P}, \bar{\mathbf{p}}$ ) nella Formula 4.35 sono  $\mathbf{p}$  e  $\mathbf{q}$ . Si tratta quindi di vincoli cosiddetti “hard” sui parametri  $\mathbf{p}$  e  $\mathbf{q}$ .

Per rendere il problema trattabile, i vincoli sono introdotti nella misura di errore dell'AAM, moltiplicandoli per un peso  $K$  grande (vincoli “soft”):

$$\sum_{\mathbf{u} \in \mathbf{s}_0} \left[ A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) \right]^2 + K \left\| \mathbf{P} \left( \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) + \begin{bmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{bmatrix} - \mathbf{N} \left( \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q} \right) \right\|^2 \quad (4.36)$$

si ha che per  $K \rightarrow \infty$  i vincoli diventano “hard”, quindi per valori di  $K$  sufficientemente grandi, la soluzione tende a quella ideale.

Si osservi che la soddisficiabilità del problema di ottimizzazione vincolato è stata dimostrata nella Sezione 4.1.2, quando si è visto che un modello di forma bidimensionale sufficientemente complesso può generare le stesse forme 2D di un modello 3D.

Anche in questo caso, si applica il procedimento di “project out” sull'errore immagine (Sezione 3.3.1):

$$G(\mathbf{p}; \mathbf{q}) = \|A_0(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}))\|_{\text{span}(\mathbf{A}_i)^\perp}^2 \quad (4.37)$$

e riscrivendo poi i vincoli in forma di funzione:

$$F(\mathbf{q}; \mathbf{p}; \bar{\mathbf{p}}; \mathbf{P}; o_u; o_v) = \left\| \mathbf{P} \left( \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) + \begin{bmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{bmatrix} - \mathbf{N} \left( \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q} \right) \right\|^2 \quad (4.38)$$

si arriva finalmente alla misura di errore definitiva degli AAM 2D+3D in tempo reale:

$$G(\mathbf{p}; \mathbf{q}) + K \sum_{t=u,v} \sum_{i=1}^n F_{t_i}^2(\mathbf{q}; \mathbf{p}; \bar{\mathbf{p}}; \mathbf{P}; o_u; o_v) \quad (4.39)$$

dove  $F_{u_i}(\dots)$  è la componente di  $F$  corrispondente al vertice  $u_i$ , e in modo analogo per  $v_i$ .

#### 4.3.2 Estensione della Composizione Inversa 2D

Dati dei valori iniziali di  $\mathbf{p}$ ,  $\mathbf{q}$  e  $\bar{\mathbf{p}}$ , determinare iterativamente il valore incrementale dei parametri che minimizza la misura di errore è un problema in parte simile al caso 2D. In particolare, si è interessati agli incrementi nei parametri che minimizzano:

$$G(\mathbf{p} + \mathbf{J}_p \Delta \mathbf{p}; \mathbf{q} + \mathbf{J}_q \Delta \mathbf{q}) \quad (4.40)$$

applicando la regola di aggiornamento:

$$\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p} + \mathbf{J}_p \Delta \mathbf{p}); \mathbf{q} + \mathbf{J}_q \Delta \mathbf{q}) \approx \mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}) \circ \mathbf{N}(\mathbf{W}(\mathbf{u}; \Delta \mathbf{p}); \Delta \mathbf{q})^{-1} \quad (4.41)$$

dove l'uguaglianza è vera con una approssimazione del primo ordine.

L'introduzione delle matrici  $\mathbf{J}_p$  e  $\mathbf{J}_q$  in un certo senso aiuta a rendere meno cruda l'approssimazione, soprattutto nel caso di grandi variazioni di posa tra un'iterazione e la successiva. Nello specifico,  $\mathbf{J}_p$  è una matrice  $m \times m$  che esprime la relazione tra un cambiamento incrementale nel valore di  $\Delta \mathbf{p}$  e il cambiamento nel valore dei parametri  $\mathbf{p}$  prodotto dalla composizione. E in modo analogo per la matrice  $4 \times 4 \mathbf{J}_q$  e i parametri  $\mathbf{q}$ .

Tralasciando i dettagli del calcolo di  $\mathbf{J}_p$  e  $\mathbf{J}_q$  (che non saranno determinate direttamente), si verifica che la soluzione incrementale al fitting 2D + 3D è data da:

$$\begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{p} \\ \Delta \bar{\mathbf{p}} \\ \Delta \mathbf{P} \\ \Delta o_u \\ \Delta o_v \end{bmatrix} = -\mathbf{H}_{3D}^{-1} \left( \begin{bmatrix} \mathbf{K}_q \\ \mathbf{K}_p \\ \mathbf{0} \\ \mathbf{0} \\ 0 \\ 0 \end{bmatrix} + K \sum_{t=u,v} \sum_{i=1}^n \mathbf{SD}_{F_{t_i}}^T F_{t_i}(\mathbf{p}; \mathbf{q}; \bar{\mathbf{p}}; \mathbf{P}; o_u; o_v) \right) \quad (4.42)$$

con i *vincoli sulla geometria* steepest descent  $\mathbf{SD}_{F_{t_i}}$  dati dalla concatenazione delle derivate parziali di  $F_{t_i}$ :

$$\mathbf{SD}_{F_{t_i}} = \left[ \frac{\partial F_{t_i}}{\partial \mathbf{q}} \mathbf{J}_q \quad \frac{\partial F_{t_i}}{\partial \mathbf{p}} \mathbf{J}_p \quad \frac{\partial F_{t_i}}{\partial \bar{\mathbf{p}}} \quad \frac{\partial F_{t_i}}{\partial \mathbf{P}} \quad \frac{\partial F_{t_i}}{\partial o_u} \quad \frac{\partial F_{t_i}}{\partial o_v} \right] \quad (4.43)$$

e:

$$\mathbf{H}_{3D} = \begin{bmatrix} H_{2D} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + K \sum_{t=u,v} \sum_{i=1}^n \mathbf{S} \mathbf{D}_{F_{t_i}}^T \mathbf{S} \mathbf{D}_{F_{t_i}} \quad (4.44)$$

I vettori di prodotti scalari  $\mathbf{K}_q = [K_1 \dots K_4]^T$ ,  $\mathbf{K}_p = [K_5 \dots K_{4+m}]^T$  e la matrice  $\mathbf{H}_{2D}$  sono associati alla parte bidimensionale dell'AAM, e sono determinati come descritto in dettaglio nella Sezione 3.3.3.

L'unica parte di  $F(\dots)$  che dipende da  $\mathbf{p}$  e  $\mathbf{q}$  è  $(-\mathbf{N}(\mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q}))$ , quindi  $\frac{\partial F_{t_i}}{\partial \mathbf{q}} \mathbf{J}_q$  e  $\frac{\partial F_{t_i}}{\partial \mathbf{p}} \mathbf{J}_p$  rappresentano il cambiamento (negato) nella composizione dei warp (Formula 4.41) prodotto da  $\Delta \mathbf{q}$  e  $\Delta \mathbf{p}$ , rispettivamente. Per calcolare tali derivate si usa un semplice rapporto incrementale, impostando di volta in volta un singolo parametro  $\delta q_i$  o  $\delta p_i$  ad un valore  $\epsilon$  piccolo (ad esempio 1.0) per poi determinare il warp incrementale  $\mathbf{W}(\mathbf{u}; \delta \mathbf{p})$  e la trasformazione incrementale  $\mathbf{N}(\cdot; \delta \mathbf{q})$ . Si applicano quindi le tecniche esposte nella Sezione 3.3.5 per combinare il warp incrementale con quello corrente ad ottenere la forma  $\delta \mathbf{s}$  associata al warp  $\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q}) \circ \mathbf{N}(\mathbf{W}(\mathbf{u}; \delta \mathbf{p}); \delta \mathbf{q})^{-1}$ .

La stima delle derivate è infine data da:

$$\frac{\partial F}{\partial q_j} \mathbf{J}_q = \frac{\mathbf{N}(\mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q}) - \delta \mathbf{s}}{\epsilon} \quad (4.45)$$

$$\frac{\partial F}{\partial p_j} \mathbf{J}_p = \frac{\mathbf{N}(\mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i; \mathbf{q}) - \delta \mathbf{s}}{\epsilon} \quad (4.46)$$

$\frac{\partial F_{t_i}}{\partial \bar{\mathbf{p}}}$  è più semplice da determinare e si ricava facilmente dalla definizione di  $F(\dots)$ :

$$\frac{\partial F}{\partial \bar{p}_j} = \mathbf{P} \bar{\mathbf{s}}_j \quad (4.47)$$

dove  $\mathbf{P}$  è il valore corrente della matrice di proiezione. Procedendo in modo analogo si vede che:

$$\begin{aligned} \frac{\partial F}{\partial o_u} &= \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \end{bmatrix} \\ \frac{\partial F}{\partial o_v} &= \begin{bmatrix} 0 & \dots & 0 \\ 1 & \dots & 1 \end{bmatrix} \end{aligned} \quad (4.48)$$

La derivata rispetto a  $\mathbf{P}$  è più difficile, in quanto una proiezione prospettica debole ha solo 4 gradi di libertà, ma 6 componenti. Anche in questo caso, è possibile usare una stima approssimata che

è accettabile nel caso incrementale, scomponendo così i 4 gradi di libertà di  $\mathbf{P}$ . Per definizione:

$$\mathbf{P} = \sigma \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \quad (4.49)$$

quindi la derivata rispetto alla scala è data da:

$$\frac{\partial F}{\partial \sigma} = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \bar{\mathbf{s}} = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \left( \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) \quad (4.50)$$

dove  $\bar{p}_i$  sono i parametri di forma 3D correnti.

Gli altri 3 gradi di libertà di  $\mathbf{P}$  sono esprimibili come rotazioni degli assi di proiezione rispetto ai versori del sistema di coordinate. Assumendo che gli angoli di rotazione incrementali  $\Delta\theta_x$ ,  $\Delta\theta_y$  e  $\Delta\theta_z$  da determinare siano sufficientemente piccoli, vale la seguente regola di aggiornamento su  $\mathbf{P}$  (*small-angle updates*):

$$\mathbf{P} \leftarrow \mathbf{P} \left( \mathbf{I} + \Delta\theta_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} + \Delta\theta_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} + \Delta\theta_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \quad (4.51)$$

applicando questa relazione alla Formula 4.38, si ottengono le derivate rispetto agli incrementi sugli angoli:

$$\frac{\partial F}{\partial \Delta\theta_x} = \mathbf{P} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \bar{\mathbf{s}}, \quad \frac{\partial F}{\partial \Delta\theta_y} = \mathbf{P} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \bar{\mathbf{s}}, \quad \frac{\partial F}{\partial \Delta\theta_z} = \mathbf{P} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{\mathbf{s}} \quad (4.52)$$

dove  $\mathbf{I}$  è la matrice identica  $3 \times 3$  e  $\bar{\mathbf{s}}$  è stato definito nella Formula 4.50.

È importante tenere a mente che tutte le derivate di  $\mathbf{SD}_{F_{t_i}}$  sono valutate sui valori correnti dei parametri  $(\mathbf{q}; \mathbf{p}; \bar{\mathbf{p}}; \mathbf{P}; o_u; o_v)$ , quindi devono essere ricalcolate ad ogni iterazione, e come conseguenza anche l'Hessiana inversa. Fortunatamente, la dimensionalità del problema continua a rimanere indipendente dalla grandezza delle apparenze, ed è pari a  $(m + \bar{m} + 6)$ .

### 4.3.3 Aggiornamento della Soluzione Corrente

Come prevedibile, il calcolo della composizione inversa tra il warp incrementale e quello corrente non cambia rispetto alla versione puramente 2D (Sezione 3.3.5). Ad esclusione di  $\mathbf{P}$ , le altre regole di aggiornamento sono semplicemente additive:  $\bar{\mathbf{p}} \leftarrow \bar{\mathbf{p}} + \Delta\bar{\mathbf{p}}$ ,  $o_u \leftarrow o_u + \Delta o_u$ ,  $o_v \leftarrow o_v + \Delta o_v$ .

L'aggiornamento di  $\mathbf{P}$  richiede innanzitutto l'estrazione del valore di scala corrente  $\sigma$  e degli assi normalizzati (Formula 4.49). Dopodichè, applicando la Formula 4.51, si vede che la regola di

aggiornamento è la seguente:

$$\mathbf{P} \leftarrow (\sigma + \Delta\sigma) \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{bmatrix} \begin{bmatrix} 1 & -\Delta\theta_z & \Delta\theta_y \\ \Delta\theta_z & 1 & -\Delta\theta_x \\ -\Delta\theta_y & \Delta\theta_x & 1 \end{bmatrix} \quad (4.53)$$

con  $\mathbf{P}$  che è in seguito ortogonalizzata, in quanto l'aggiornamento della parte di rotazione non preserva necessariamente l'ortogonalità degli assi.

#### 4.3.4 L'Algoritmo Completo

Come per l'algoritmo 2D, si rappresentano in modo schematico le varie fasi dell'algoritmo, ricordando che la parte di training è invariata ed è stata inclusa solo per facilitare la consultazione e per aggiornare la notazione.

Anche se la parte di stima delle mode di forma 3D è stata inserita nella fase di training, si intende soltanto che questa non può essere effettuata durante il fitting, se si desiderano prestazioni real-time. Ma questo non implica che l'operazione debba necessariamente essere effettuata in concomitanza con il training dell'AAM. Anzi, ciò è improbabile se si considera che l'AAM è solitamente usato per generare i dati necessari all'algoritmo di ricostruzione della forma dal movimento.

Ovviamente, non si è neppure obbligati ad usare un algoritmo di ricostruzione della forma dal movimento per specificare il modello di forma 3D, è sufficiente che questo sia effettivamente in grado di generare la stessa tipologia di forme generate dal modello 2D.

#### Training

1. Determina la forma media 2D  $\mathbf{s}_0$  e le mode di forma 2D  $\mathbf{s}_i$  usando le procedure descritte nella Sezione 2.1.
2. Determina i vettori  $\mathbf{s}_j^*$  (come descritto nella Sezione 3.2.4) e ortonormalizza l'insieme  $[\mathbf{s}_1^* \dots \mathbf{s}_4^* \mathbf{s}_1 \dots \mathbf{s}_m]$ . I risultati ottenuti saranno i nuovi  $\mathbf{s}_j^*$  e  $\mathbf{s}_i$ .
3. Determina l'apparenza media  $A_0$  e le mode di apparenza  $A_i$  usando il procedimento descritto nella Sezione 2.2.
4. Calcola il gradiente dell'apparenza media  $\nabla A_0$ .
5. Stima le Jacobiane  $\frac{\partial \mathbf{N}}{\partial \mathbf{q}}$  e  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$  e le immagini di steepest descent  $SD_i(2D)$  come descritto nella Sezione 3.3.2.
6. Determina l'Hessiana inversa  $\mathbf{H}_{2D}^{-1}$  a partire dalla Formula 3.32.
7. Determina la forma media 3D  $\bar{\mathbf{s}}_0$  e le mode di forma 3D  $\bar{\mathbf{s}}_i$  usando, ad esempio, un algoritmo di ricostruzione della forma dal movimento come quello descritto nella Sezione 4.2.

### Fitting

1. Data la forma iniziale 2D  $\mathbf{s}_{init}$ , pon  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; \mathbf{p}, \mathbf{q}) = \mathbf{s}_{init}$ , a indicare che  $\mathbf{s}_{init}$  è la forma associata al warp corrente.
2. Inizializza i parametri di forma 3D  $\bar{\mathbf{p}}$ , la trasformazione prospettica debole  $\mathbf{P}$ , e le traslazioni  $o_u$  e  $o_v$ . Queste informazioni possono essere richieste in ingresso alla procedura, anche se sarebbe preferibile stimarle in qualche modo a partire dalla forma 2D  $\mathbf{s}_{init}$ , ad esempio cercando la trasformazione che minimizzi l'errore:

$$\left\| \mathbf{P}\bar{\mathbf{s}}_0 + \begin{bmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{bmatrix} - \mathbf{s}_{init} \right\|^2$$

che anche in questo caso si può effettuare con un'analisi procustiana [15]. Si noti che, in questo particolare caso,  $\bar{\mathbf{p}} = \mathbf{0}$ .

3. Stima i vincoli sulla geometria steepest descent  $\mathbf{SD}_{F_{t_i}}$  e l'Hessiana invertita  $\mathbf{H}_{3D}^{-1}$  usando le tecniche descritte nella Sezione 4.3.2.
4. Determina l'immagine di errore  $E(\mathbf{u}) = I(\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) - A_0(\mathbf{u})$  effettuando un warp su  $I$  che mappi la porzione di immagine dal sistema di riferimento della forma  $\mathbf{N} \circ \mathbf{W}(\mathbf{s}_0; \mathbf{p}, \mathbf{q})$  a quello della forma media  $\mathbf{s}_0$ .
5. Calcola il valore di  $\mathbf{K}_{2D} = [\mathbf{K}_p \ \mathbf{K}_q \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0}]^T$  usando la Formula 3.34.
6. Determina i valori incrementali:  $\Delta\mathbf{q}$ ,  $\Delta\mathbf{p}$ ,  $\Delta\bar{\mathbf{p}}$ ,  $\Delta\sigma$ ,  $\Delta\theta_x$ ,  $\Delta\theta_y$ ,  $\Delta\theta_z$ ,  $\Delta o_u$ ,  $\Delta o_v$  applicando la Formula 4.42.
7. Aggiorna il warp corrente:  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q}) \leftarrow (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; \mathbf{p}, \mathbf{q})) \circ (\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta\mathbf{q}, -\Delta\mathbf{p}))$  applicando le procedure descritte nella Sezione 3.3.5.
8. Aggiorna gli altri parametri e la forma 3D usando le tecniche descritte nella Sezione 4.3.3.
9. Ripeti dal punto (3) finché non si ha convergenza, o il numero massimo di iterazioni è stato raggiunto.



## Capitolo 5

# Uso ed Estensione di Implementazioni Pre-Esistenti

A più di dieci anni dall'introduzione degli AAM, ci si aspetterebbe di trovare a disposizione numerose implementazioni open-source del metodo. In realtà non è così, e all'atto pratico se ne contano solo un paio di utilizzabili al di fuori di un contesto puramente educativo, e solo una di queste è portabile. Queste implementazioni sono state esaminate a fondo, allo scopo di acquisire familiarità con le applicazioni pratiche degli AAM e di valutare le possibilità di integrare con successo le estensioni di nostro interesse nel codice.

In particolare, è stato possibile individuare tre implementazioni di AAM liberamente disponibili in grado di produrre risultati di buona qualità, più molte altre di incomplete, non funzionanti o non liberamente disponibili. La valutazione delle prestazioni delle due implementazioni di maggiore interesse è poi avvenuta attraverso lo sviluppo di alcuni casi di studio, atti a valutare la qualità dei risultati prodotti in diverse condizioni, e allo scopo di capire quali fossero i limiti degli AAM in generale e delle specifiche implementazioni.

Le implementazioni valutate hanno sicuramente dimostrato di poter ottenere eccellenti prestazioni di fitting, ma allo stesso tempo si sono rivelate poco efficienti e/o difficilmente portabili. Inoltre, si è osservata una generale scarsità di documentazione in parti cruciali del codice, che a volte si è rivelato piuttosto contorto. Scelte progettuali poco orientate all'estensibilità del codice hanno infine comportato l'abbandono dell'iniziale intenzione di integrare in una di queste implementazioni le estensioni 2D+3D.

I casi di studio sono stati comunque fondamentali per approfondire le implicazioni pratiche associate all'uso di AAM, e hanno reso la seguente implementazione degli AAM 2D+3D molto più agevole.

## 5.1 L'Implementazione di Cootes

Tim Cootes, uno degli inventori degli Active Appearance Models, ha realizzato una serie di strumenti<sup>1</sup> per costruire, esaminare e utilizzare AAM. Questi sono purtroppo disponibili esclusivamente in forma binaria, ma si tratta comunque di un ottimo punto di partenza per familiarizzare con le applicazioni pratiche degli AAM.

Data l'impossibilità ad agire sul codice di questa implementazione, non sono stati fatti esami particolarmente approfonditi, limitandosi ad utilizzare le immagini e le annotazioni incluse per addestrare un AAM, esaminarne le mode e provare qualche operazione di fitting.

Ciò che sicuramente impressiona di più in questa implementazione è la varietà di opzioni e strumenti messi a disposizione, e anche se il modo per accedervi non è sempre intuitivo e diretto, la documentazione allegata è solitamente sufficiente a risolvere problematiche di questo tipo. Ottima anche la qualità e la varietà delle 26 immagini indicate, con le relative annotazioni.

Il formato di annotazione usato da questa implementazione è testuale e segue il seguente schema:

```
version: 1
n_points: n
{
  u1 v1
  ...
  un vn
}
```

dove l'origine del sistema di coordinate è in alto a sinistra dell'immagine, con asse Y rivolto verso il basso, e le unità sono i pixel. Al solito,  $u_i$  e  $v_i$  rappresentano, rispettivamente, le coordinate X e Y dell'i-esima annotazione.

## 5.2 L'Implementazione di Stegmann: AAM-API

L'AAM-API<sup>2</sup> è una libreria di classi e funzioni in C++ che implementano funzionalità necessarie per lavorare con gli Active Appearance Models, utilizzando alcune variazioni sull'algoritmo tradizionale sviluppate da Mikkel B. Stegmann nella sua tesi [33].

Alcune delle estensioni proposte da Stegmann sono: la rifinitura della triangolazione di Delaunay per oggetti dai contorni “concavi”, la possibilità di escludere parti di apparenza che non sono di

---

<sup>1</sup>Gli strumenti sono liberamente disponibili al sito: [http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/software/am\\_tools\\_doc/index.html](http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/software/am_tools_doc/index.html)

<sup>2</sup>Liberamente disponibile al sito: <http://www2.imm.dtu.dk/~aam/>

## 5.2. L'IMPLEMENTAZIONE DI STEGMANN: AAM-API

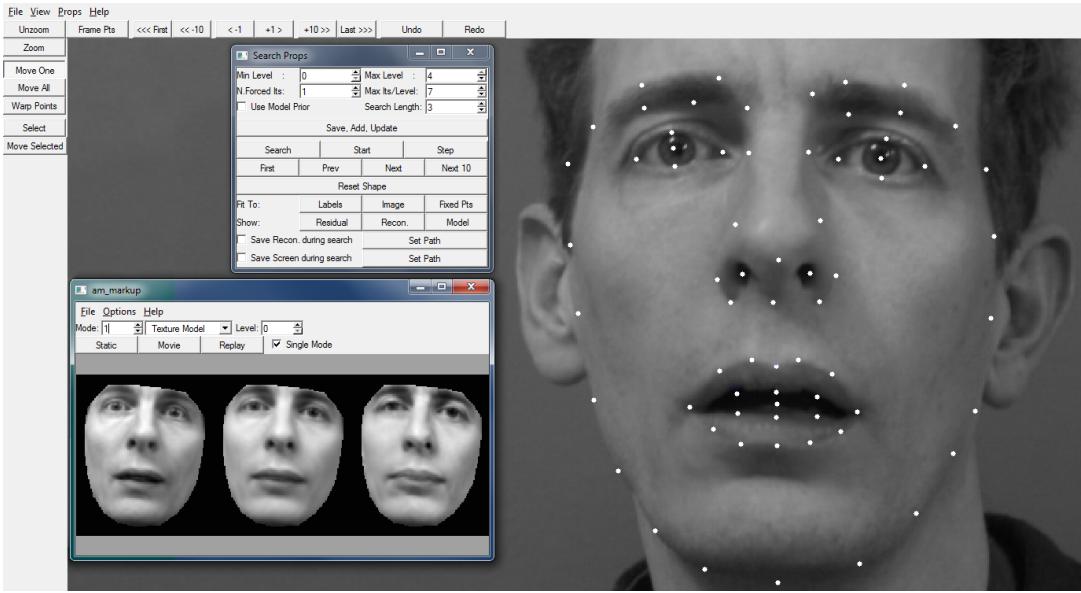


Figura 5.1. Una delle componenti principali dell'implementazione di Cootes: **am\_markup**. Lo strumento è usato principalmente per l'annotazione del training set, eventualmente usando un fitting AAM per ottenere una stima iniziale delle annotazioni. Sono illustrate anche le opzioni disponibili per il fitting e la finestra di visualizzazione dei dettagli del modello usato. L'immagine usata fa parte del training set incluso nell'implementazione.

interesse e l'inclusione di una parte di “bordo” nelle apparenze per migliorare il fit in immagini con sfondo costante (ad es. mediche). Ma il più importante contributo è rappresentato sicuramente dall'algoritmo di inizializzazione automatica della ricerca, che si è dimostrato essere piuttosto valido.

Per quanto riguarda i dettagli più prettamente implementativi, si segnalano innanzitutto le due dipendenze di questa implementazione: DIVA<sup>3</sup>, che è inclusa nel sorgente, quindi all'atto della compilazione non costituisce fonte di particolari complicazioni e VisionSDK<sup>4</sup> [17] (da cui dipende anche DIVA), pacchetto ormai obsoleto per applicazioni di visione computazionale sviluppato dalla Microsoft. Non sorprende quindi che VisionSDK supporti esclusivamente piattaforme Microsoft Windows, rendendo di fatto la AAM-API non portabile.

Nonostante le problematiche legate alla portabilità, questa implementazione si è rivelata estremamente utile per comprendere a fondo il funzionamento degli Active Appearance Models e per capire le tecniche e gli strumenti necessari per realizzare una valida implementazione degli stessi. E il tutto è stato possibile per buona parte del tempo senza particolari sforzi di programmazione, grazie agli strumenti inclusi per l'annotazione di immagini e per l'addestramento e il fitting di AAM. Senza dimenticare l'eccellente dataset IMM [28, 34], che non è allegato all'AAM-API, ma è reperibile anch'esso dal sito di Stegmann.

<sup>3</sup>Homepage del software: <http://www2.imm.dtu.dk/~diva/>

<sup>4</sup>Stegmann fornisce una versione modificata da usare esclusivamente con la AAM-API nel suo sito.

### 5.2.1 Il Dataset IMM

Il dataset IMM è costituito da 240 immagini annotate, suddivise tra 40 persone (di cui 33 maschi e 7 femmine), con 6 immagini per individuo, ognuna appartenente a una particolare combinazione di posa e espressione. Ogni individuo è identificato da un valore numerico, e lo stesso vale per le combinazioni di posa e espressione, che sono identificate nel seguente modo:

1. Posa: frontale; Espressione: neutra.
2. Posa: frontale; Espressione: felice.
3. Posa: ruotata di circa 30 gradi a destra dell'individuo; Espressione: neutra.
4. Posa: ruotata di circa 30 gradi a sinistra dell'individuo; Espressione: neutra.
5. Posa: frontale; Espressione: neutra, fascio di luce concentrato in una zona alla sinistra dell'individuo.
6. Posa: frontale; Espressione: scherzosa (variabile a seconda degli individui).

Alcuni esempi del dataset sono illustrati nella Figura 5.2: l'immagine “16-4m”, ad esempio, rappresenta il sedicesimo individuo (che è maschio) nella posa “4”. Ogni immagine è in risoluzione  $640 \times 480$ , e si osserva che le immagini sono state acquisite in condizioni molto simili tra loro, quindi sono ideali per ottenere una rappresentazione compatta attraverso un'analisi statistica.

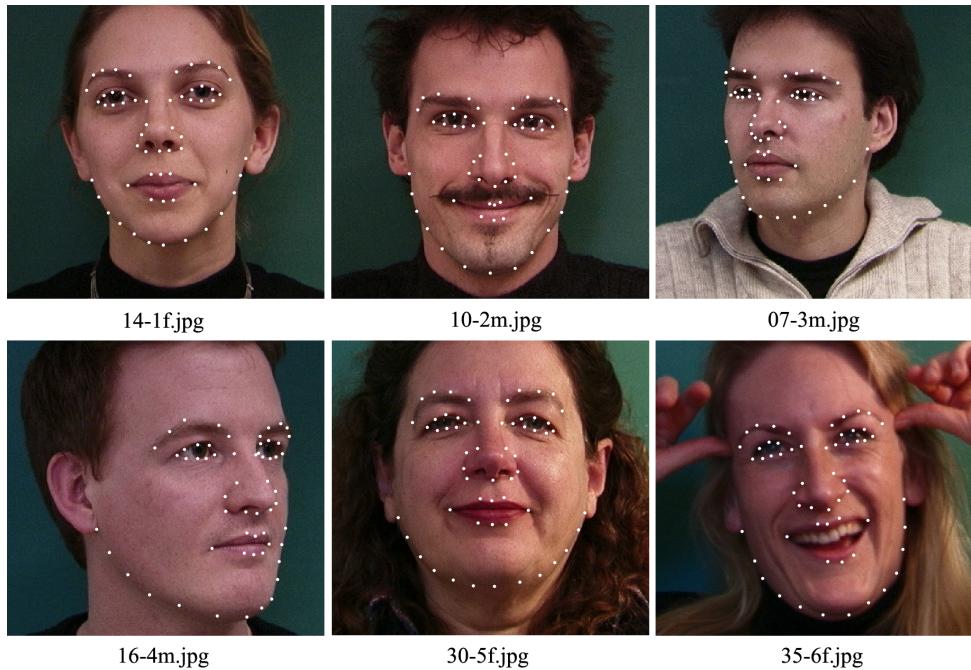


Figura 5.2. Un piccolo sottoinsieme del dataset IMM, che è comunque sufficiente per capire come ogni campione è identificato e annotato.

## 5.2. L'IMPLEMENTAZIONE DI STEGMANN: AAM-API

Per quanto riguarda il formato di annotazione, questo è testuale ed è indubbiamente più complesso rispetto a quello usato da Cootes, in quanto l'implementazione di Stegmann definisce le forme come una serie di curve chiuse o aperte, e non come una semplice serie ordinata di punti. In questa sede non si è interessati ad elencare ogni singolo aspetto del formato, e si rimanda alle relative specifiche [28] per i dettagli omessi.

Brevemente, il formato di annotazione è così definito:

```

 $n$ 
#curva tipo  $u$   $v$  #ann. precedente successiva
...
#curva tipo  $u$   $v$  #ann. precedente successiva
immagine

```

dove:

$n$	Numero di annotazioni specificate dal file.
#curva	Identifica la curva di appartenenza di questa annotazione.
tipo	Bit-field che ha scopi informativi e la cui specifica non ha interesse per le nostre applicazioni.
$u$	Coordinata X <i>relativa</i> dell'annotazione, appartenente al dominio $[0,1)$ . In termini matematici, data una coordinata $x \in [1,w]$ (dove $w$ è la larghezza dell'immagine), la coordinata $u$ è data da $(x - 1) / w$ .
$v$	Coordinata Y relativa dell'annotazione, determinata allo stesso modo della coordinata $u$ , ed espressa considerando l'angolo in alto a sinistra dell'immagine come origine e coordinate Y crescenti verso il basso.
#ann.	Numero identificativo dell'annotazione, può essere zero. È usato per identificare annotazioni corrispondenti alla stessa feature in file diversi.
precedente	Indice dell'annotazione precedente nella curva corrente. Se tale annotazione non esiste, è generalmente uguale a <#ann.>.
successiva	Indice dell'annotazione successiva nella curva corrente. Se tale annotazione non esiste, è generalmente uguale a <#ann.>.
immagine	È il nome, completo di estensione, del file immagine a cui fanno riferimento le annotazioni.

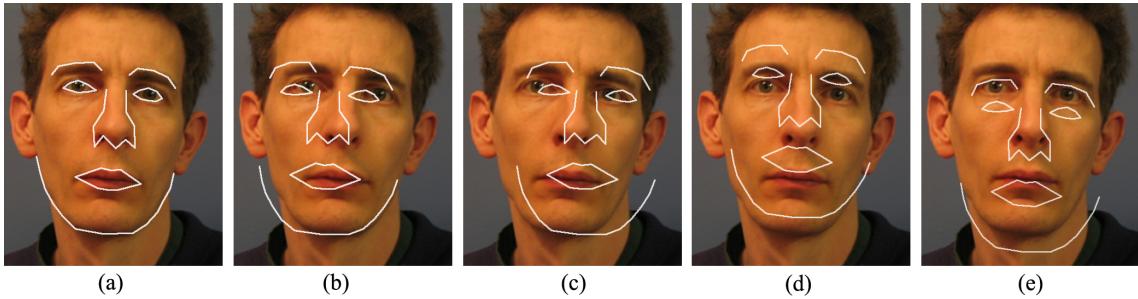


Figura 5.3. Modalità pseudo-automatica di inizializzazione: (a) la forma in ingresso; (b) e (c) spostamento sull'asse x del 10%; (d) e (e) spostamento sull'asse Y del 10%.

### 5.2.2 Fitting con l'AAM-API

Prima di procedere ad esporre le modalità con cui sono state valutate le prestazioni e i limiti di questa implementazione, è opportuno approfondire le tecniche usate dall'AAM-API per effettuare il fitting, altrimenti risulterebbe difficile capire in che modo sono state determinate le misure di errore per i casi di studio.

Si distinguono due diverse modalità di fitting, che differiscono dalla strategia di inizializzazione: *automatica* e *pseudo-automatica*, entrambe basate sull'utilizzo di una forma di riferimento (*ground-truth*) fornita in ingresso, che generalmente indica la soluzione desiderata.

La strategia automatica è relativamente complessa: inizia effettuando 3 iterazioni dell'algoritmo di fitting a partire da diverse posizioni iniziali, per poi scegliere al più 10 di queste con un metodo euristico. Per ognuna delle posizioni scelte, si effettuano altre 10 iterazioni di fitting, e la posizione scelta per inizializzare il fitting vero e proprio è quella associata alla misura di errore minore.

La strategia pseudo-automatica è più semplice ed effettua 4 fitting separati a partire dalla forma fornita in ingresso (Figura 5.3.a) traslata positivamente e negativamente nelle direzioni x (Figure 5.3.b e 5.3.c) e y (Figure 5.3.d e 5.3.e) del 10% rispetto alla dimensione della forma stessa.

Se non specificato altrimenti, nel nostro caso è sempre stata usata la modalità di inizializzazione pseudo-automatica, in quanto la procedura automatica non si è rivelata essere sempre affidabile, in quanto soggetta a “falsi positivi”.

### 5.2.3 Metriche di Errore

L'AAM-API supporta varie metriche di errore, ed è importante capire come queste siano determinate per comprendere i risultati che verranno presentati in seguito. Inoltre, considerando che verranno discusse implementazioni sostanzialmente differenti da questa, è importante capire quando diverse misure di errore siano comparabili o meno con quelle delle altre implementazioni.

## 5.2. L'IMPLEMENTAZIONE DI STEGMANN: AAM-API

Si distinguono essenzialmente due tipologie di misure di errore: l'errore sulla forma e l'errore sull'apparenza (o errore immagine).

Le misure di errore immagine di nostro interesse sono:

- Quadrato della norma  $\ell^2$ :

$$E_{sim} = \sum_{\mathbf{u} \in \mathbf{s}_0} \left[ A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) \right]^2 \quad (5.1)$$

(esattamente lo stesso valore espresso dalla Formula 3.23)

- Distanza di Mahalanobis indipendente:

$$E_{Mah} = \frac{1}{L} \sum_{\mathbf{u} \in \mathbf{s}_0} \frac{\left[ A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) \right]^2}{\sigma_{\mathbf{u}}^2} \quad (5.2)$$

dove  $\sigma_{\mathbf{u}}^2$  è la varianza di apparenza misurata sul pixel  $\mathbf{u}$  durante il training e  $L$  è il numero di pixel considerati. In pratica si tratta di un errore quadratico medio normalizzato rispetto alla varianza.

- Autocorrelazione del residuo:

$$E_{col} = \left| \frac{1}{\sigma_{\mathbf{d}}^2 (L-1)} \sum_i^{L-1} (d_i - \bar{d}) (d_{i+1} - \bar{d}) \right| \quad (5.3)$$

dove:

$$\mathbf{d} = \text{vec}(\mathbf{E}), \quad E(\mathbf{u}) = A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) \quad (5.4)$$

con  $\sigma_{\mathbf{d}}^2$  che rappresenta la varianza di  $\mathbf{d}$  e  $\bar{d}$  la sua media. L'operatore  $\text{vec}()$  è stato definito nella Sezione 2.1.1.

Si ricorda che in questa implementazione le apparenze sono normalizzate, quindi queste metriche sono comparabili ad altre solo se calcolate con apparenze normalizzate allo stesso modo.

Le misure di errore sulla forma sono invece:

- Distanza di Mahalanobis:

$$D_{Mah} = \sum_{\forall \Lambda_{c_i} > 10^{-15}} \frac{c_i^2}{\Lambda_{c_i}} \quad (5.5)$$

dove  $c_i$  sono i parametri combinati prodotti dal fitting e  $\Lambda_{c_i}$  è l'autovalore corrispondente.

- Distanza media tra annotazioni:

$$D_{pt.pt.} = \frac{1}{n} \sum_{i=1}^n \sqrt{\left(u_i - u_i^{ref}\right)^2 + \left(v_i - v_i^{ref}\right)^2} \quad (5.6)$$

misurata rispetto al ground-truth.

- Distanza media tra annotazioni e curve  $D_{pt.crv.}$ : è definita come la distanza euclidea media tra i punti della forma prodotta dal fitting e le curve più vicine della forma di riferimento.

Tutte le misure di errore sulle forme sono calcolate in coordinate assolute, e le unità sono per questo i pixel.

## 5.2.4 Casi di Studio

### 5.2.4.1 Caso Person-Dependent

Durante i primi esperimenti con l'AAM-API, ci si è concentrati sull'analisi delle prestazioni nel caso *person-dependent*, usando quindi un training set di immagini dello stesso individuo. È stato scelto il dataset fornito con l'implementazione di Cootes, escluse le annotazioni. Queste sono state ricreate da zero, allo scopo di acquisire familiarità con la procedura di annotazione e assicurarsi che la qualità dei risultati di fitting ottenuti fossero indipendenti da fattori esterni.

Il processo di annotazione è stato piuttosto macchinoso, vista la scarsa tolleranza agli errori del software di annotazione fornito con l'AAM-API e considerando che errori nell'assicurare la corrispondenza tra le annotazioni hanno causato crash improvvisi delle procedure di fitting e training. E l'assenza di messaggi di errore di alcun tipo non ha di certo facilitato la correzione di questi errori.

Tutte le 26 immagini del dataset di Cootes sono state annotate, e a partire da queste, quattro sotto-insiemi sono stati creati per testare diversi aspetti dell'implementazione. In seguito si farà riferimento a questi nel modo seguente:

- **intero**: Insieme composto da tutte le 26 immagini.
- **dimezzato**: 13 immagini prese dal dataset intero in modo casuale.
- **ridotto**: 6 immagini prese dal dataset intero in modo casuale.
- **posa**: 5 immagini, di cui 4 con posa frontale ed espressione variabile e una con posa più di profilo e espressione neutra.

Dataset	$D_{pt.pt.}$			$D_{pt.crv.}$			$E_{sim}$		
	Medio	Min	Max	Medio	Min	Max	Medio	Min	Max
<b>intero</b>	3.42	1.67	7.63	1.55	0.86	3.34	0.01	0	0.04
<b>dimezzato</b>	4.46	2.48	6.47	1.99	1.09	4.06	0.02	0.01	0.05
<b>ridotto</b>	4.3	3.54	6.85	2.12	1.51	4.55	0.03	0.01	0.06
<b>posa</b>	12.33	12.27	12.51	6.93	6.93	7.12	0.12	0.12	0.12

Tabella 5.1. Risultati del fitting leave-one-out, ottenuti usando la modalità pseudo-automatica, inizializzata con il ground-truth.

I risultati ottenuti dai diversi sottoinsiemi sono riassunti nella Tabella 5.1. La restante parte di questa sezione si occuperà di commentare questi risultati e di descrivere in modo dettagliato le modalità usate per ottenerli.

Per quanto concerne il sottoinsieme **intero**, sono stati addestrati 26 modelli diversi, escludendo ogni volta una differente immagine dal training set e usando questa per il fitting (approccio *leave-one-out*). Il tutto è stato automatizzato usando una piccola applicazione Java realizzata ad-hoc.

Osservando la Tabella 5.1, si nota in generale un'elevata qualità dei risultati, con errori punto-punto costantemente inferiori ai 7 pixel (data l'alta risoluzione dei dati, dell'ordine dei  $300 \times 300$  pixel, una variazione di 7 pixel corrisponde al 2% circa). Fa eccezione il caso illustrato nella Figura 5.4.a, in cui le feature associate alle sopracciglia sono state erroneamente posizionate in una zona inferiore alle stesse.

Il dataset **dimezzato** è stato costruito allo scopo di valutare quanto la dimensione del training set contribuisce alla qualità di fitting, ed eventualmente osservare se e quanto gli AAM sono soggetti al fenomeno di *over-fitting*. Il test è stato effettuato nuovamente con un approccio leave-one-out. La qualità dei risultati si è attestata ancora una volta su valori elevati, con errori punto-punto sempre al di sotto dei 7 pixel, con rare imprecisioni dovute a difficoltà nel riconoscimento delle sopracciglia e del contorno viso (Figura 5.4.b). Non sembra che con training set di questa dimensione si ponga il problema dell'*over-fitting*, comunque.

Il testing leave-one-out con il dataset **ridotto** ha dato risultati anche migliori delle aspettative. Pur contando un training set di sole 5 immagini, l'errore medio sulle annotazioni ha continuato ad attestarsi su valori inferiori ai 7 pixel, anche se si è osservato un incremento nell'errore immagine. Mentre dal punto di vista qualitativo, in questo caso si sono osservati maggiori errori nel riconoscimento degli occhi e del contorno viso rispetto ai casi precedenti, com'è facilmente visibile dalle Figure 5.4.c e 5.4.d.

Il dataset **posa** ha portato all'estremo la riduzione nella variabilità del training set per quanto riguarda la posa. Questo ha inevitabilmente comportato grosse difficoltà nel fitting dell'immagine con posa non rappresentata nel training set. La Tabella 5.1 mostra chiaramente l'entità degli errori, che si attestano su valori 3 o 4 volte superiori a quelli visti con gli altri dataset, mentre la Figura 5.4.e mostra graficamente l'entità dell'errore.

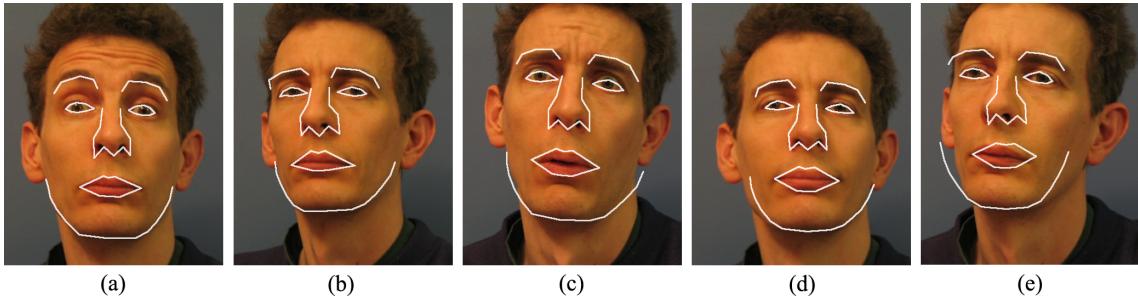


Figura 5.4. Alcuni esempi di fitting errato: (a) Nonostante la grandezza del training set **intero**, in rari casi si osservano errori nel riconoscimento delle sopracciglia; (b) È possibile che la minor dimensione del training set abbia permesso all'AAM di generalizzare meglio nel caso **dimezzato**, in cui gli errori sono qualitativamente più contenuti; (c) e (d) Il dataset **ridotto** mostra evidenti difficoltà ad effettuare il fitting in alcuni casi; (e) La mancanza di variazione di posa nel training set **posa** è fonte di inevitabili errori nel fitting di pose sconosciute all'AAM.

Ulteriori esperimenti sono stati effettuati per valutare la robustezza al rumore, la sensibilità a forti escursioni di luminosità nel training set e l'eventuale impatto di immagini sfocate nel training set. Ognuno di questi esperimenti è stato effettuato alterando opportunamente sia il training set **ridotto**, sia un training set di dimensione maggiore, ottenuto escludendo dal dataset **intero** le 3 immagini che nei test precedenti avevano riscontrato i maggiori errori di fitting, e usando quest'ultime per la valutazione.

In tutti i casi non si sono riscontrati particolari peggioramenti rispetto ai risultati degli altri esperimenti, a dimostrazione dell'efficacia della normalizzazione fotometrica nel compensare a variazioni di luminosità nel training set e della capacità di rimozione del rumore della modellazione multi-risoluzione.

#### 5.2.4.2 Caso Person-Independent

Come facilmente intuibile, il termine *person independent* indica che l'AAM è usato per riconoscere molteplici individui, anche mai visti nel training set.

Nel valutare le prestazioni person-independent dell'AAM-API, è risultata naturale la scelta del dataset IMM: il training set è stato inizialmente costruito estraendo casualmente 5 soggetti dal dataset, che sono stati poi usati per gli esperimenti di fitting.

Nonostante le condizioni indubbiamente favorevoli, con training set composto da immagini appartenenti allo stesso dataset, acquisite in condizioni molto simili, e con inizializzazione del fitting relativamente vicina alla soluzione, la qualità dei risultati prodotti (illustrati nella Tabella 5.2) è risultata sicuramente inferiore a gran parte degli altri esperimenti person-dependent.

La scarsa capacità nel fitting person-independent dell'AAM-API è risultata ancor più evidente usando immagini non appartenenti al dataset IMM: infatti, a partire da un AAM addestrato con

## 5.2. L'IMPLEMENTAZIONE DI STEGMANN: AAM-API

Modalità	$D_{pt.pt.}$			$D_{pt.crv.}$			$E_{sim}$		
	Medio	Min	Max	Medio	Min	Max	Medio	Min	Max
autom.	10.28	3.55	387.56	4.12	1.98	340.85	0.22	0.07	2.35
pseudo.	8.54	3.62	56.47	3.43	1.71	41.89	0.19	0.06	9.21

Tabella 5.2. Risultati del fitting person-independent, usando sia la modalità automatica che quella pseudo-automatica di inizializzazione. La modalità pseudo-automatica si conferma la migliore, sia per il fatto che l'inizializzazione è garantita essere in un dintorno del ground-truth, sia per l'occasionale tendenza della modalità automatica a scegliere cattive posizioni iniziali (influenzando in modo pesantemente negativo sui valori medi e massimi). Questo nonostante l'errore immagine  $E_{sim}$ , generalmente inferiore per la modalità automatica. A suggerisce che tale misura non è sufficiente a determinare la qualità del fitting, fatto che è stato confermato da un caso di studio successivo.

l'intero dataset IMM, sono stati effettuati alcuni tentativi di fitting su immagini da noi acquisite, con il viso del soggetto principalmente al centro e posa e espressione generalmente neutra. La Figura 5.5 mostra chiaramente come, a prescindere dal metodo di inizializzazione usato, e della tipologia di forme usate per l'inizializzazione, i risultati si siano rivelati decisamente inadeguati.

Ulteriori esperimenti hanno avuto lo scopo di isolare i fattori che differenziavano maggiormente le nostre immagini da quelle del training set, per identificare quali avessero il maggior impatto sulle (cattive) prestazioni di fitting.

Una volta identificate delle immagini nel dataset IMM che potessero essere facilmente alterabili con

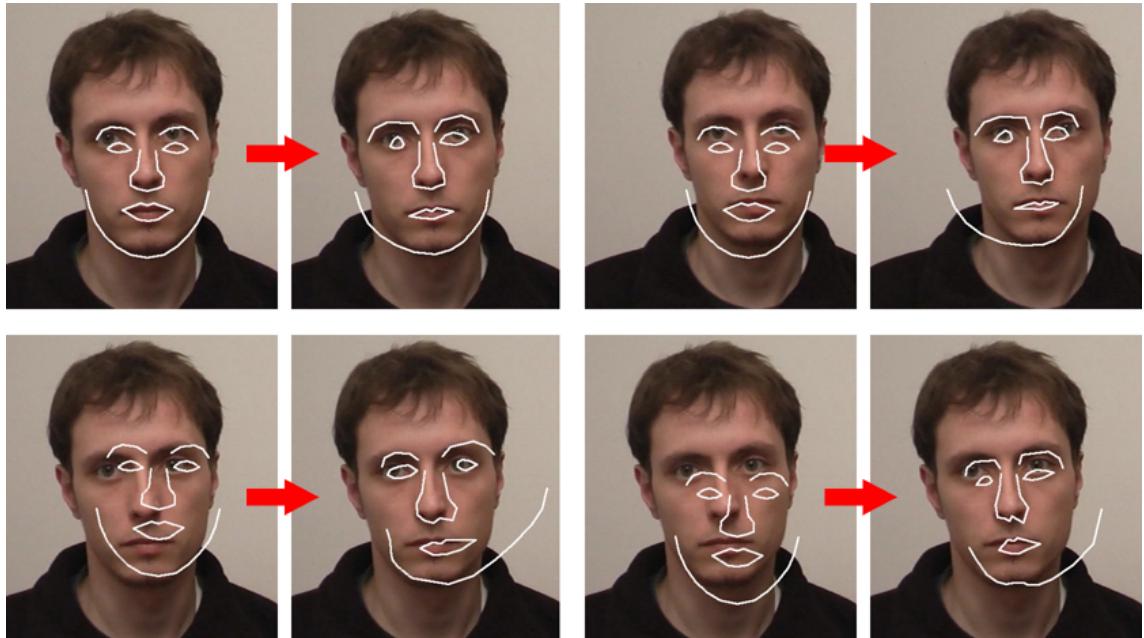


Figura 5.5. I risultati migliori ottenuti dal fitting di immagini da noi acquisite usando un AAM addestrato con l'intero dataset IMM. Le frecce indicano l'applicazione dell'operazione di fitting a partire dalla posizione iniziale segnata.

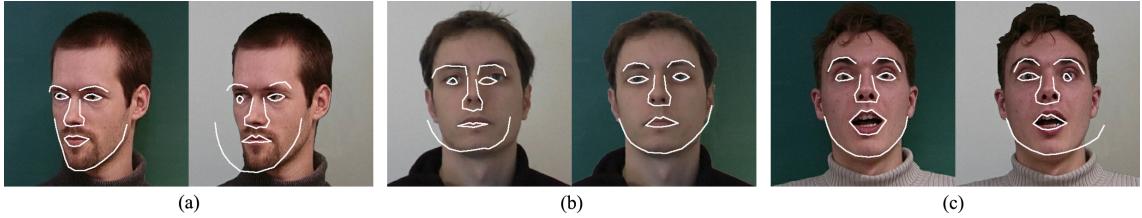


Figura 5.6. Fitting dell’immagine con sfondo originale (a sinistra) comparato con quello con sfondo alterato (a destra): (a), (c) sono immagini del dataset IMM; (b) è una delle immagini da noi acquisite.

tecniche di foto-ritocco, queste sono state manipolate in modo da simulare traslazioni tridimensionali, corrispondenti a traslazioni e cambiamenti di scala bidimensionali, cercando di mantenere il più possibile inalterato lo sfondo. In questo modo si sarebbe potuto verificare se le difficoltà di fitting riscontrate fossero dovute a differenze nel posizionamento di soggetto e/o fotocamera. I risultati di questo esperimento, seppur non di qualità paragonabile a quelli ottenuti con le immagini originali, sono stati nella maggior parte dei casi buoni.

Ritornando ad esaminare le immagini da noi acquisite, non è rimasto che tentare di manipolarle in modo da aumentare la loro somiglianza con le immagini del dataset IMM. A questo scopo sono state alterate luminosità, contrasto e rumore di fondo; anche se la manipolazione che all’atto pratico ha avuto più impatto è stata la sostituzione vera e propria dello sfondo con quello usato dal dataset IMM. Infatti, nelle immagini con sfondo alterato, si è osservato un significativo miglioramento delle prestazioni di fitting, a suggerire che l’AAM avrebbe in qualche modo “imparato” lo sfondo del training set. Un esempio di questo fenomeno è illustrato nella Figura 5.6.b.

Le speculazioni sull’impatto dello sfondo nel fitting sono state confermate da un’esperimento simile a quello appena descritto, considerando alcune immagini del dataset IMM a cui è stato alterato lo sfondo per fare in modo che somigliasse a quello usato nelle nostre immagini. Importanti peggioramenti nella qualità del fitting (due esempi sono illustrati nelle Figure 5.6.a e 5.6.c) hanno confermato che la tipologia di AAM implementata dall’AAM-API è fortemente dipendente dallo sfondo usato durante il training.

In conclusione, l’AAM-API ha dimostrato di poter essere applicata al caso person-independent, ma solo in condizioni molto controllate. Variazioni significative di sfondo, scala, luminosità e rumore hanno contribuito tutte a un rapido deterioramento nei risultati di fitting.

E anche se le variazioni di sfondo non sono state oggetto di studio nel caso person-dependent, si prevede che i risultati non sarebbero molto diversi da quelli ottenuti nel caso person-independent.

### 5.2.4.3 Estensione dell'AAM-API: un'Applicazione Video

Una variazione interessante rispetto al fitting AAM di singole immagini è rappresentata dai video. Definiamo video una sequenza di immagini, acquisite a intervalli di tempo regolari, che sono la rappresentazione di come un oggetto deformabile si è evoluto in un certo periodo di tempo. Un video è quindi costituito da immagini, e su esse è naturalmente possibile effettuare il fitting AAM. L'unica differenza sostanziale tra i fotogrammi di un video e una generica collezione di immagini, è che in un video generalmente si osserva *coerenza temporale*. In altre parole, nei video con coerenza temporale, (che sono quelli a cui si è interessati in questa sede) ogni immagine della sequenza è molto simile alla precedente e alla successiva.

In particolare, si è voluto osservare come fosse possibile sfruttare la coerenza temporale per ovviare a occlusioni temporanee parziali o totali e a eventuali errori di fitting (senza la pretesa di risolvere il problema nel caso generale [16, 44]). Inoltre, la bassa risoluzione ( $480 \times 180$ ), e la scarsa qualità del video usato hanno costituito un'ulteriore sfida alle capacità degli AAM.

Anche gli strumenti forniti per il fitting allegati all'AAM-API hanno in questo caso mostrato segni di inadeguatezza, specialmente nella limitata disponibilità di opzioni per agire sull'inizializzazione del fitting. Per ovviare a questi limiti è stata realizzata un'applicazione C++ ad-hoc, basata sull'AAM-API, con lo scopo di operare le modifiche alle modalità di fitting necessarie ad ottenere i risultati desiderati.

L'utilizzo dell'AAM-API durante la programmazione non differisce molto da una qualsiasi altra libreria, soprattutto se si fa riferimento al sorgente degli strumenti che si occupano di effettuare il training e il fitting. Si è osservato però uno scarso livello di parametrizzazione delle funzionalità, con molte costanti importanti hard-coded nel codice e alcune funzionalità utilizzabili solo modificando e ricompilando il codice della libreria stessa.

Non essendo intenzionati a ricompilare l'intera libreria a ogni modifica del codice, è stata ridefinita all'interno della nostra applicazione la procedura dell'AAM-API destinata al fitting, permettendoci così di modificare liberamente tutta la parte di inizializzazione.

Il video usato per il caso di studio è la ripresa di un esempio di interazione non verbale tra un bimbo e sua madre, della durata di 1 minuto e 23 secondi, e caratterizzato dalla presenza contemporanea di due volti e un buon numero di occlusioni parziali e totali. Si sottolinea che i due volti sono stati trattati separatamente, e in particolare ci si è concentrati esclusivamente sulla madre nelle prime fasi, essendo auspicabile che le strategie di fitting funzionanti per la madre avrebbero funzionato anche per il bimbo.

La costruzione di un training set adeguato per il fitting di un video complesso come quello usato nel nostro caso non è un'operazione banale, soprattutto se non si ha intenzione di annotare centinaia di fotogrammi. Nel nostro caso, il training set è stato costruito in più iterazioni, partendo da 13 fotogrammi scelti ad intervalli regolari nel video: ad ogni iterazione è stato ripetuto il training e il

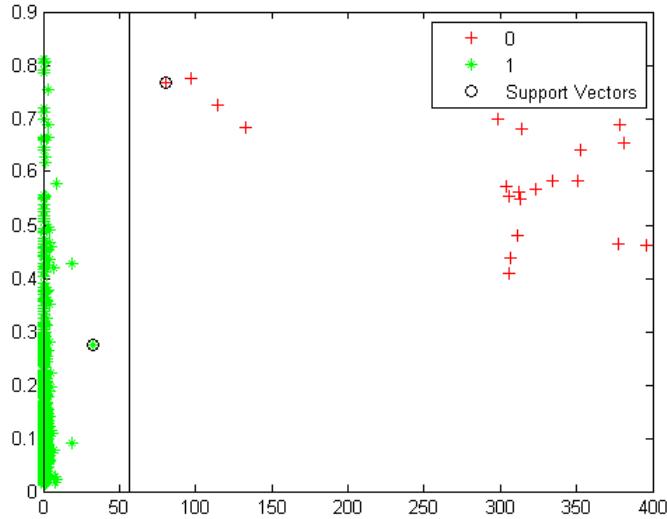


Figura 5.7. Separabilità tra fitting accettabili e fitting non accettabili. In ascissa: traslazione del baricentro della forma rispetto al frame precedente. In ordinata:  $E_{sim}$ . I campioni rappresentanti fitting non accettabili sono le croci, mentre quelli rappresentanti fitting accettabili sono gli asterischi. Questo grafico rappresenta solo due dei quattro parametri usati per la discriminazione.

fitting del video, isolando i fotogrammi che mostravano i maggiori errori di fitting, ed espandendo il training set con quei fotogrammi che meglio rappresentavano la tipologia di pose ed espressioni soggette a errori.

Nell'annotazione del training set, si è osservato come la topologia usata dal dataset IMM non fosse l'ideale in condizioni di cambi di posa molto grandi e di basse risoluzioni, e si è optato per alcune semplificazioni, eliminando le annotazioni più ambigue e difficili da specificare. In totale, sono state annotate 24 immagini per il training set della madre e 37 per il training set del bimbo.

Allo scopo di sfruttare la coerenza temporale, il fitting è stato modificato in modo da usare come forma di inizializzazione per il fotogramma corrente il risultato del fitting sul fotogramma precedente. Questa strategia garantisce i migliori risultati possibili nel caso di perfetta coerenza temporale e assenza di occlusioni o altri fenomeni che possano generare errori di fitting; in caso contrario, si ha un effetto a catena che causa la rapida degenerazione dei risultati. La mancanza di coerenza temporale non ha sicuramente rappresentato una preoccupazione nel nostro caso, mentre non si può dire lo stesso delle occlusioni.

Nel tentare di gestire le occlusioni si presentano molti interrogativi, come ad esempio: dove posizionare la forma durante il fitting se c'è occlusione? Ma ancor prima l'interrogativo più grande è: come individuare le occlusioni prima che queste compromettano definitivamente il fitting? E se questo non è possibile, come sviluppare un meccanismo di ripristino per recuperare la posizione corretta a seguito di un'occlusione o di fitting errato?



Figura 5.8. Gestione delle occlusioni sfruttando la coerenza temporale e discriminazione tra fitting accettabili e non accettabili attraverso SVM: durante le occlusioni, il fitting è effettuato solo se produce un risultato “vicino” a quello del fotogramma precedente, altrimenti si propaga il risultato del fotogramma precedente. Nei rari casi in cui si ha divergenza, questa è individuata usando la stessa metrica usata per la discriminazione, e si procede a una ri-inizializzazione del fitting.

Nel nostro caso, si è rivelato più semplice sfruttare la coerenza temporale per reagire a cambiamenti troppo drastici nelle forme prodotte dal fitting e scegliere se ripristinare la forma di inizializzazione o evitare proprio di fare il fitting, a seconda del valore di una metrica di errore ad-hoc più sofisticata del semplice errore immagine  $E_{sim}$  (la metrica usata di default dall'AAM-API). Usando solo l'errore immagine, infatti, diventa a volte impossibile distinguere tra facce e altre parti dell'immagine, anche nel caso di occlusioni solo parziali.

Non è stato particolarmente difficile creare metriche di errore più robuste, combinando in vario modo: errore immagine, cambiamento incrementale nei parametri di forma e cambiamento incrementale nei parametri di trasformazione globale (scala, traslazione e rotazione). Nessuna di queste è stata però in grado di discriminare in modo affidabile tra risultati di fitting accettabili e meno accettabili. Dopo qualche tentativo di miglioramento delle capacità discriminanti delle metriche di errore, si è reso evidente come una metodologia di *trial and error* non fosse adatta per un problema di tale complessità, e sono stati raccolti dati per determinare che combinazione di metrica/funzione discriminante sarebbe stata necessaria.

I dati ottenuti sono in parte rappresentati nella Figura 5.7, da cui si vede facilmente che i casi accettabili si sono dimostrati *linearmente separabili* da quelli “errati”, suggerendo l’uso di una Support Vector Machine [14] per la discriminazione. Nonostante sia stato necessario diverso tempo per raccogliere i campioni (più di 2000) e classificarli per addestrare la SVM, il risultato finale si è rilevato di ottima qualità, sia per il fitting della madre<sup>5</sup> che per il fitting del bimbo<sup>6</sup>.

<sup>5</sup>Video con i risultati del fitting per la madre: <http://www.youtube.com/watch?v=RJjSNBHTrxk>

<sup>6</sup>Video con i risultati del fitting per il bimbo: <http://www.youtube.com/watch?v=sUQhbqfry6I>

### 5.2.5 Efficienza di Fitting

Uno degli scopi principali dei casi di studio sviluppati per l’AAM-API è stato la valutazione dell’idoneità di questa implementazione a fornire da base per sviluppare gli AAM con composizione inversa e fitting 3D in tempo reale. A questo riguardo, la valutazione della robustezza e della qualità dei risultati prodotti dall’implementazione è importante quanto l’efficienza dell’algoritmo di fitting.

La Tabella 5.3 riassume alcuni dei tempi di fitting<sup>7</sup> associati ai casi di studio descritti in precedenza.

Nel caso person-dependent si osserva che, a una riduzione della dimensione del training set, è associata una riduzione del tempo di fitting; ad indicare che le prestazioni di fitting dipendono in modo non trascurabile dal numero di parametri del modello combinato. Questo spiega anche il maggior tempo di fitting nel caso person-independent, che non è comunque aumentato in modo proporzionale alla dimensione del training set, per merito della capacità degli AAM di generalizzare e mantenere compatti i modelli di forma e apparenza.

Quasi tutte le misure sono state valutate su immagini a risoluzione  $640 \times 480$  (con dimensione effettiva delle apparenze di circa la metà) e queste, per gli standard degli AAM, sono alte risoluzioni. Il fitting video ha infatti mostrato che con immagini di dimensione minore, e apparenze a risoluzione  $100 \times 100$ , o meno, il processo può avvenire in tempo reale.

In conclusione, queste prestazioni non sono da ritenersi eccezionali, per trattarsi di un’implementazione in C++. Si nota comunque che il codice dell’AAM-API non è stato particolarmente ottimizzato, e non è da escludere che le scarse prestazioni siano anche dovute all’algoritmo di fitting “tradizionale”, che è sicuramente più oneroso di quello con composizione inversa.

Dataset	Person-dependent				Person-independent	Video (madre)
	intero	dimezzato	ridotto	posa		
Tempo Medio (ms)	345.55	208.27	130.38	140.29	419.54	16.31

Tabella 5.3. Tempi medi per singola operazione di fitting, in millisecondi. Nel caso person-dependent la convergenza è avvenuta tra le 10 e le 7 iterazioni, nei casi person-independent e video è avvenuta, in media, entro 12 iterazioni.

## 5.3 L’Implementazione di Brian Amberg

Nel loro articolo, Amberg et al. [2] hanno riunito in un unico framework teorico diversi metodi di fitting basati sulla composizione di warp.

Questo lavoro ha portato alla definizione di 4 nuove metodologie di fitting, caratterizzate dall’uso di approcci diversi al calcolo di Jacobiane e Hessiane:

<sup>7</sup>Ottenuti con un processore Intel dual-core a 2.4GHz.

- Metodi di discesa del gradiente:
  - **CoDe**: Jacobiane esatte.
  - **LinCoDe**: Jacobiane approssimate.
- Metodi di Gauss-Newton:
  - **CoNe**: Jacobiane e Hessiana esatte.
  - **CoLiNe**: Jacobiane esatte e Hessiana approssimata.
  - **ICIA**: Jacobiana approssimata e Hessiana approssimata, è il metodo con composizione inversa descritto nel Capitolo 3.

Tutti questi metodi sono stati implementati da Amberg in MATLAB, che ha reso l'implementazione liberamente disponibile<sup>8</sup>. Questa include funzioni di facile uso per effettuare il training e il fitting, ed è compatibile con i dataset IMM [28] e XM2VTSDB [27]. Più precisamente, IMM è compatibile a patto di usare il modello già addestrato allegato con l'implementazione, mentre se si ha a disposizione il dataset XM2VTSDB<sup>9</sup> (che è a pagamento), sono forniti tutti i dati e le procedure richieste sia per il training che per il fitting.

Gli aspetti teorici delle tecniche usate da Amberg non sono stati approfonditi particolarmente, ma la prima cosa che si nota esaminando l'implementazione è che il warp di immagini, anziché essere lineare a tratti, fa uso di *thin plate splines*. La discussione di come questa rappresentazione alternativa influenzi le operazioni di warp è al di fuori dai nostri scopi e si fa riferimento all'articolo di Bookstein [7] per l'approfondimento.

Nel valutare le prestazioni dei vari metodi di fitting di questa implementazione, si è cercato di replicare i migliori risultati della Sezione 5.2.4.3, ma l'assenza dall'implementazione di strumenti per l'annotazione ha reso prima di tutto necessario determinare il formato di annotazione usato, per convertire le annotazioni già in nostro possesso.

Considerando che anche questa implementazione usa un formato testuale, non è stato difficile comprenderne la struttura, costituita sostanzialmente da un elenco delle coordinate x e y delle annotazioni:

```
1 u1 v1
...
1 un vn
```

---

<sup>8</sup>Per ottenere il software è sufficiente seguire le seguenti istruzioni: [http://www.cs.unibas.ch/personen/amberg\\_brian/aam/README](http://www.cs.unibas.ch/personen/amberg_brian/aam/README).

<sup>9</sup>Il dataset XM2VTSDB non è liberamente disponibile, ma va acquistato al sito: <http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/>

## CAPITOLO 5. USO ED ESTENSIONE DI IMPLEMENTAZIONI PRE-ESISTENTI

Altra cosa che distingue in modo significativo questa implementazione da tutte le altre provate, è la presenza di maschere di foreground/background nel training set, ossia immagini il cui canale *alfa* vale 0 nei pixel corrispondenti a sfondo. Amberg ha, in separata sede, puntualizzato che egli ha generato queste maschere applicando l'algoritmo di segmentazione di immagini GrabCut [30]. Nel nostro caso, si è fatto uso del software SegTool<sup>10</sup>, basato su tecniche di taglio di grafi interattive, tra cui anche GrabCut.

A seguito di alcune modifiche a SegTool, atte a rendere il funzionamento più efficiente e automatizzato, è stato possibile creare le maschere, anche se si è osservato che con il training set in nostro possesso (forse a causa di rumore e bassa risoluzione), è stato necessario un intervento manuale di rifinitura per ottenere dei risultati accettabili.

Per quanto riguarda le operazioni di training e fitting AAM, si è rivelato sufficiente far riferimento agli script di esempio inclusi nell'implementazione. In particolare, durante il fitting si osserva facilmente che l'assenza di un metodo di inizializzazione automatico porta l'implementazione a "imbrogliare", usando leggere variazioni del ground-truth come punto di partenza (in modo analogo al metodo pseudo-automatico dell'AAM-API).

Una rapida valutazione delle prestazioni dei vari metodi ha chiaramente evidenziato la superiorità del metodo **CoDe**: questo ha dimostrato di essere molto dinamico e in grado di adattarsi a tutti i cambiamenti di posa e espressione presenti nel video. Questo metodo non si è però differenziato dagli altri in quanto a robustezza, e alla prima occlusione (anche parziale) il fitting ha mostrato la tendenza a divergere verso forme molto grandi o molto piccole. Non avendo alternative, è stato comunque il metodo preferenziale per le operazioni fitting.

La mancanza di un metodo di inizializzazione automatico ha reso di certo più difficile il recupero della forma corretta a seguito di occlusioni molto significative, ma in casi meno gravi la coerenza temporale si è rivelata sufficiente, a patto di avere inizializzato correttamente il fitting del primo fotogramma.

Per poter discriminare i fitting accettabili da quelli che possono generare divergenza in fotogrammi successivi, anche in questo caso si è reso necessario individuare una metrica di errore e una funzione discriminante, possibilmente evitando il lungo processo di raccolta dati necessario all'addestramento di una SVM. Gli elementi considerati per la costruzione di una metrica di errore sono stati:  $E_{sim}$  (Formula 5.1) e  $\Delta\mathbf{q}$  e  $\Delta\mathbf{p}$  (variazione nei parametri prodotta dal fitting rispetto alla forma iniziale).

Partendo dal presupposto che i parametri  $q_3$  e  $q_4$  rappresentano la componente X e Y della traslazione, e che un grande cambiamento nei parametri di forma non è un fenomeno normale se si ha

---

<sup>10</sup>Di Mohit Gupta, Krishnan Ramnath (Robotics Institute, CMU), disponibile al sito: <http://www.cs.cmu.edu/~mohitg/segmentation.htm>

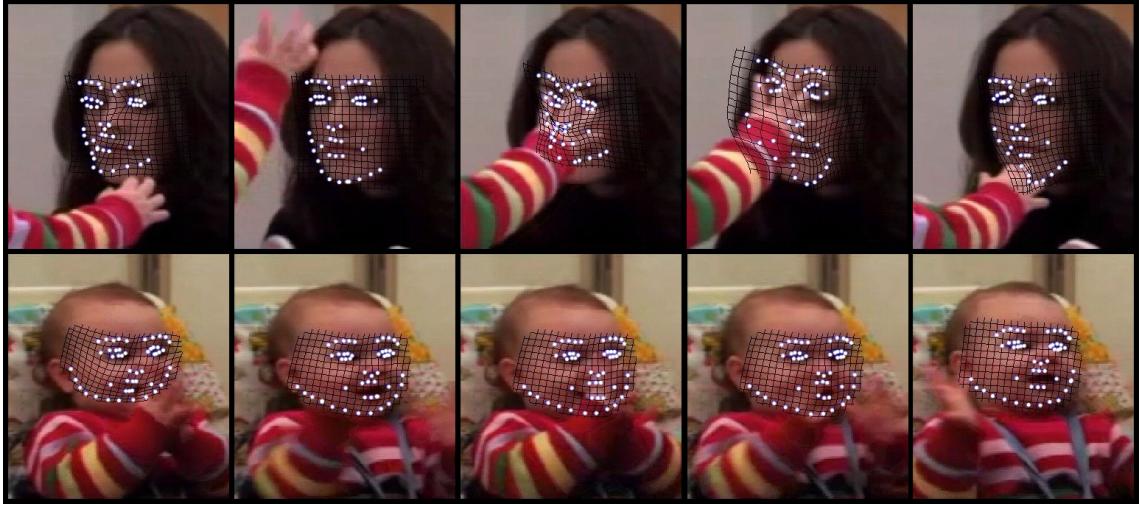


Figura 5.9. Gestione delle occlusioni usando l'implementazione di Amberg. La tecnica è fondamentalmente la stessa illustrata dalla Figura 5.8, ma senza la ri-inizializzazione automatica.

coerenza temporale, la seguente metrica si è dimostrata valida:

$$E_{sim} \left( 0.12 \sqrt{\Delta q_3^2 + \Delta q_4^2} + \frac{0.7}{m} \sum_{i=1}^m \frac{\Delta p_i}{\Lambda_i} + 1 \right) \quad (5.7)$$

dove  $\Lambda_i$  è l'autovalore corrispondente all'i-esima moda di forma. Si noti che, nel caso in cui non ci sia variazione nei parametri di forma e di trasformazione globale, l'errore è  $E_{sim}$ ; inoltre, il fatto che l'errore immagine agisca come un peso sulle altre componenti di errore fa sì che maggiori cambiamenti nella forma siano accettati se questi generano un minor errore immagine.

Considerando che tutti i metodi di fitting tendevano a divergere di fronte a occlusioni anche parziali, non è stato sufficiente usare la metrica di errore per scegliere ad ogni frame il metodo migliore. Per questo motivo, in modo analogo a come fatto con l'AAM-API, gli errori prodotti dai vari metodi di fitting sono stati comparati con l'errore calcolato mantenendo la forma inalterata rispetto al frame precedente. Essendo  $\Delta p$  e  $\Delta q$  dei parametri fondamentali della metrica di errore, ed essendo questi pari a zero se la forma non cambia, si è osservato che spesso in caso di occlusione la scelta di non effettuare il fitting è stata preferibile.

Concludendo, i risultati si sono dimostrati qualitativamente vicini (Figura 5.9) a quelli prodotti dall'AAM-API, sia per la madre<sup>11</sup>, che per il bimbo<sup>12</sup>. Anche se si osserva che il tempo di calcolo necessario è stato decisamente superiore in questo caso, a causa sia dell'uso da parte dell'implementazione di un linguaggio ad alto livello, sia per le tecniche di warp usate, più precise ma anche più complesse. Giusto per dare un'idea, i metodi più veloci, **LinCoDe** e **ICIA** hanno richiesto

<sup>11</sup>Video con i risultati del fitting per la madre: <http://www.youtube.com/watch?v=4qBuKtPyXiQ>

<sup>12</sup>Video con i risultati del fitting per il bimbo: <http://www.youtube.com/watch?v=ham8SDfXUgI>

tempi dai 3 ai 5 secondi per il fitting di una singola immagine del dataset IMM, mentre il metodo più lento, **CoNe**, è arrivato ad impiegare più di 50 secondi per un singolo fit! Considerando invece i singoli fotogrammi del video usato per questo caso di studio, i tempi di fitting si sono aggirati mediamente sul secondo o poco meno<sup>13</sup>.

## 5.4 Altre Implementazioni

Oltre alle tre implementazioni che sono state analizzate più o meno in dettaglio in precedenza, molte altre sono state oggetto di valutazione durante il nostro lavoro. Queste non sono state in grado di soddisfare appieno i nostri requisiti di:

- Qualità del fitting.
- Estensibilità del codice o elevata flessibilità.
- Portabilità immediata o potenziale.
- Ridotto lavoro di adattamento e comprensione del codice necessari all'utilizzo.

Molto brevemente, alcune di queste implementazioni sono:

**RAVL**<sup>14</sup> Recognition And Vision Library è una libreria di classi C++ per applicazioni di visione computazionale, riconoscimento di pattern e altro. Fornisce un supporto preliminare agli AAM, anche se le funzionalità non sono ancora ben integrate nella libreria. Anche dopo aver dedicato diverso tempo nel tentativo di compilare gli esempi relativi agli AAM non si è riusciti ad ottenere alcun risultato; sia per l'assenza dei dati richiesti dagli esempi, sia per continui errori di accesso ai file (in caso si cerchi di fornire dati alternativi a quelli mancanti).

**AAM su Matlabcentral**<sup>15</sup> Implementazione MATLAB degli AAM tradizionali. Il formato di annotazione è complicato ma non impossibile da comprendere, ma anche a seguito della conversione delle annotazioni in nostro possesso, il training e il fitting non hanno prodotto risultati degni di nota. I dati forniti con l'implementazione sono costituiti da immagini di mani, di cui è stato annotato il contorno, e sembra che l'implementazione riesca ad operare solo su questi contorni e non su topologie più generiche.

---

<sup>13</sup>Tutti i tempi sono stati ottenuti con un processore Intel dual-core a 2.4GHz.

<sup>14</sup>Centre for Vision, Speech and Signal Processing (University of Surrey): <http://www.ee.surrey.ac.uk/CVSSP/Software>

<sup>15</sup>Dirk-Jan Kroon, disponibile al sito: <http://www.mathworks.co.uk/matlabcentral/fileexchange/26706-active-shape-model-asim-and-active-appearance-model-aam>

#### 5.4. ALTRE IMPLEMENTAZIONI

**aambuilding**<sup>16</sup> Semplice strumento di addestramento di AAM sviluppato in C++, portabile ma poco utile visto il solo supporto all'addestramento e l'eccessiva semplicità.

**Active Appearance Models Library**<sup>17</sup> Semplice libreria C++ open source che sostiene di implementare sia gli AAM tradizionali che quelli con composizione inversa. Non ha prodotto risultati con i dati in nostro possesso, e non è predisposta per l'uso al di fuori di piattaforme Windows, anche se sembra scritta in modo portabile.

**AAMtools**<sup>18</sup> Non ancora disponibile, implementa le varianti con composizione inversa degli AAM sviluppate da Papandreou et al. [29].

**Paamela**<sup>19</sup> Non ancora disponibile, implementa l'algoritmo con composizione inversa di Matthews et al. [25].

---

<sup>16</sup>JIA Pei, disponibile al sito: [http://en.sourceforge.jp/projects/sfnet\\_aambuilding/](http://en.sourceforge.jp/projects/sfnet_aambuilding/)

<sup>17</sup>Yao Wei, disponibile al sito: <http://code.google.com/p/aam-library/>

<sup>18</sup>George Papandreou, homepage: <http://cvsp.cs.ntua.gr/software/AAMtools/>

<sup>19</sup>Emmanuel Goossaert, blog: <http://codecapsule.com/2010/08/12/active-appearance-models-in-c-plus-plus/>

CAPITOLO 5. USO ED ESTENSIONE DI IMPLEMENTAZIONI PRE-ESISTENTI

## Capitolo 6

# Implementazione MATLAB di AAM 2D+3D in Tempo Reale

I casi di studio descritti nel capitolo precedente hanno mostrato, tra le altre cose, come nessuna delle implementazioni di AAM liberamente disponibili sia allo stesso tempo portabile ed efficiente. Di fronte a questo problema, è stato necessario valutare se fosse effettivamente opportuno modificare un'implementazione pre-esistente in modo da renderla conforme alle nostre necessità, oppure implementare da zero l'algoritmo con composizione inversa.

L'implementazione di Stegmann, pur essendo la più efficiente, non si è rivelata portabile a causa della libreria Vision SDK, usata per ogni operazione geometrica, grafica e di accesso a risorse di sistema. Un porting avrebbe quindi richiesto la sostituzione della libreria Vision SDK con un'alternativa portabile che, molto probabilmente, si sarebbe dovuto riscrivere da zero (questioni di copyright e la scarsa popolarità della libreria suggerivano che difficilmente ci siano stati dei tentativi di porting della stessa).

Una breve ricerca ha confermato l'assenza di una versione portabile della Vision SDK, quindi ci si è trovati a considerare se fosse il caso di riscrivere in modo portabile la porzione di libreria usata dall'AAM-API. La cosa avrebbe sicuramente richiesto una buona quantità di tempo e una profonda conoscenza del funzionamento dell'AAM-API e della Vision SDK, comportando allo stesso tempo il rischio di violare in qualche modo il copyright della libreria Microsoft.

Al contrario l'implementazione di Amberg, pur essendo portabile, è risultata ben lontana dalle prestazioni di tempo reale desiderate, e non ha fornito garanzie circa la possibilità di ottimizzare il codice in modo significativo. Inoltre, la scarsità di commenti nel codice e la natura matematicamente sofisticata dell'algoritmo hanno ostacolato significativamente la comprensione del codice.

## CAPITOLO 6. IMPLEMENTAZIONE MATLAB DI AAM 2D+3D IN TEMPO REALE

Modificare tale codice avrebbe prima di tutto richiesto un approfondimento delle tecniche matematiche più avanzate usate nell'algoritmo, un investimento di tempo che non avrebbe portato frutti nel caso in cui non sarebbe stato possibile ottimizzare significativamente la suddetta implementazione.

Riassumendo, considerata l'entità degli impedimenti temporali, tecnici, o addirittura legali, non sarebbe stata una scelta saggia tentare di integrare l'algoritmo di fitting 2D+3D in una delle implementazioni considerate. Si vedrà quindi la descrizione del processo che ha portato a un'implementazione ex novo degli Active Appearance Models 2D+3D in tempo reale, i cui fondamenti teorici sono già stati descritti nel Capitolo 4.

### 6.1 Scelte Progettuali e Convenzioni

Come linguaggio per l'implementazione è stato scelto MATLAB, per la sua intrinseca portabilità e la rapidità con cui permette lo sviluppo di applicazioni matematiche. Il prezzo di tale flessibilità è una perdita di efficienza anche se, attraverso strategie di programmazione opportune (ed eventualmente tramite MEX-files), è possibile ottenere comunque ottimi risultati.

Essendo l'algoritmo 2D+3D in tempo reale un'estensione dell'algoritmo con composizione inversa, si è scelto di implementare prima quest'ultimo per verificarne la correttezza ma, soprattutto, per assicurarsi che le prestazioni fossero accettabili. Solo dopo aver ottenuto un'implementazione efficiente e corretta dell'algoritmo 2D avrebbe avuto senso implementare l'algoritmo di recupero della forma 3D dal movimento (prerequisito dell'algoritmo 2D+3D) e le estensioni 3D alla procedura di fitting AAM.

Dal punto di vista architettonale, l'implementazione non ha necessitato di particolari accorgimenti, vista la dimensione relativamente contenuta del sorgente, che è stato semplicemente suddiviso in funzioni in modo da assicurare efficienza, convenienza di utilizzo e massimo riutilizzo del codice.

Avendo osservato come le altre implementazioni trascurassero la fase di annotazione, fornendo strumenti di annotazione poco efficienti o omettendoli completamente, si è provveduto ad implementare uno strumento di annotazione ad-hoc. Questo ha migliorato sotto molti aspetti lo strumento di annotazione della AAM-API, che si è dimostrato intollerante agli errori e poco flessibile.

Il miglioramento principale introdotto dal nostro software consiste nella possibilità di usare le annotazioni di altre immagini come punto di partenza, eliminando problematiche legate alla necessità di ricordare l'ordinamento delle annotazioni e il numero di annotazioni usate per alcuni contorni.

La possibilità di modifica delle annotazioni sia sequenziale che basata su selezione permette inoltre una gestione più efficiente delle annotazioni, evitando di dover ricominciare da capo in caso di annotazione errata.

Questi piccoli, ma significativi miglioramenti rendono il processo di annotazione molto più efficiente, grazie alla maggiore tolleranza agli errori e alla possibilità di riutilizzo del lavoro già svolto.

## 6.2 Addestramento

La chiave delle eccezionali prestazioni dell'algoritmo con composizione inversa, e degli AAM in generale, sta nella capacità di concentrare gran parte delle operazioni computazionalmente più onerose nella fase di addestramento del modello. Quindi non sorprende che questa sia anche la parte più impegnativa da implementare, seppur sia richiesto sicuramente meno impegno dal punto di vista dell'ottimizzazione, trattandosi di una fase *off-line* dell'algoritmo.

### 6.2.1 Estrazione delle Mode di Forma

In questa fase si applica l'algoritmo descritto nella Sezione 2.1.1, eseguendo alcune iterazioni dell'algoritmo di analisi procustiana sulle forme di ingresso, in modo da allinearle. Sono disponibili numerose implementazioni dell'analisi procustiana, compresa quella fornita con lo Statistics Toolbox di MATLAB, che è quella da noi usata.

A seguito dell'allineamento, è importante assicurare due cose: che le coordinate siano tutte positive; e che le forme abbiano dimensione limitata e siano il più vicine possibile all'origine. Questi requisiti sono motivati dal fatto che la forma media, ottenuta da queste forme, sarà usata per accedere alle mode di apparenza e all'apparenza media, che sono delle immagini. Per questo motivo è conveniente che le coordinate delle forme siano positive, in quanto fungono da “indici” nelle apparenze. Allo stesso tempo, un fitting efficiente richiede che le apparenze abbiano dimensioni contenute, quindi non è opportuno “sprecare” spazio nelle apparenze.

Sulle forme allineate, fissata la quantità di variabilità di forma che si desidera “coprire”, si può quindi procedere a effettuare la PCA, ad ottenere le  $m$  forme di base  $\mathbf{s}_i$ . È così possibile definire i 4 vettori  $\mathbf{s}_1^* \dots \mathbf{s}_4^*$  associati ai parametri di trasformazione globale (procedimento descritto in dettaglio nella Sezione 3.2.4), e procedere all'ortonormalizzazione dell'insieme delle forme vettorizzate  $[\mathbf{s}_1^* \dots \mathbf{s}_4^* \mathbf{s}_1 \dots \mathbf{s}_m]$ .

Anche in questo caso, è possibile utilizzare diversi metodi per l'ortonormalizzazione, e nel nostro caso si è optato per una semplice implementazione dell'algoritmo di Gram-Schmidt, che nel caso di vettori già prossimi a essere ortogonali ha il vantaggio di produrre vettori molto simili a quelli forniti in ingresso.

Infine, dati i nuovi vettori  $\mathbf{s}_1^* \dots \mathbf{s}_4^*$  e  $\mathbf{s}_1 \dots \mathbf{s}_m$  prodotti dalla procedura di ortonormalizzazione, si definiscono le matrici:

$$\mathbf{S}^* = \begin{bmatrix} \mathbf{s}_1^* & \dots & \mathbf{s}_4^* \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \dots & \mathbf{s}_m \end{bmatrix} \quad (6.1)$$

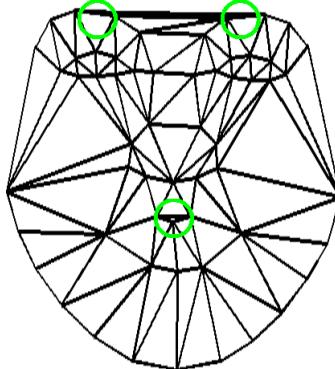


Figura 6.1. Esempi di cattiva triangolazione: triangoli eccessivamente piccoli o sproporzionati sono i casi più comuni. In casi più subdoli, quella che può sembrare una buona triangolazione, può causare difficoltà nel posizionamento di alcune annotazioni in quanto si possono verificare anomalie (*folding*) nella triangolazione.

### 6.2.2 Estrazione delle Mode di Apparenza

Per fare in modo che le apparenze in ingresso siano nello stesso sistema di riferimento, e poter così effettuare le analisi statistiche, è necessario effettuare un warp (Sezione 2.2.1) di ogni apparenza verso il sistema di riferimento della forma media. Prima di effettuare il warp, però, è necessario avere a disposizione una mesh triangolare sulle forme, mesh che può essere fornita alla procedura di training o che può essere determinata usando la triangolazione di Delaunay. In quest'ultimo caso si osserva come la qualità delle annotazioni sia determinante, e laddove topologie migliori producono triangoli ampi e ben distribuiti, topologie meno buone risultano in diversi triangoli sproporzionati (Figura 6.1) che sono fonte di cattivi risultati all'atto della composizione del warp.

Il warp è la componente più critica dell'algoritmo di fitting, in quanto una sua implementazione naïve può essere estremamente inefficiente. Buona parte delle ottimizzazioni applicabili al warp si possono effettuare durante il training, attraverso la costruzione di 3 pseudo-immagini: la prima è una mappa che associa ogni posizione interna alla forma  $s_0$ , al triangolo in cui è contenuta; le altre due sono immagini contenenti, per ogni posizione, le coordinate baricentriche  $\alpha$  e  $\beta$  (determinate usando le Formule 2.11 e 2.12) relative al triangolo di appartenenza.

Determinare le coordinate baricentriche in questa fase non ha costo aggiuntivo, considerando che queste servono anche a individuare il triangolo in cui è contenuta una posizione, osservando che per quel triangolo, e solo quel triangolo, le condizioni  $\alpha \geq 0$ ,  $\beta \geq 0$  e  $(\alpha + \beta) \leq 1$  sono tutte vere.

La procedura di warp vera e propria consiste nel costruire delle nuove immagini, le apparenze allineate, applicando la funzione di warp descritta nella Sezione 2.2.1. Nel caso in cui siano state usate le ottimizzazioni appena descritte, il warp si riduce nell'iterazione su ogni posizione discreta (pixel) interna alla forma media, usando le informazioni memorizzate nelle pseudo-immagini per determinare il triangolo da usare per il warp, e sfruttando le coordinate baricentriche per determinare la

posizione corrispondente nella forma  $\mathbf{x}_i$ :

$$I'_i(\mathbf{u}) = I_i(\mathbf{W}_{\mathbf{s}_0 \rightarrow \mathbf{x}_i}(\mathbf{u})), \quad \forall \mathbf{u} \in \mathbf{s}_0 \quad (6.2)$$

Non essendo  $\mathbf{W}_{\mathbf{s}_0 \rightarrow \mathbf{x}_i}(\mathbf{u})$ , in generale, una posizione discreta, è raccomandabile l'uso di un'interpolazione bilineare per determinare l'intensità dell'apparenza in quella particolare posizione.

Le apparenze allineate sono quindi vettorizzate (l'ordine di “vettorizzazione” di pixel e colori non è importante purché sia consistente durante tutto l'algoritmo) così da poter determinare l'apparenza media  $\mathbf{A}_0$  e le mode di apparenza  $\mathbf{A}_1 \dots \mathbf{A}_l$ , in modo analogo a come fatto per le forme. Questa procedura è descritta in dettaglio nella Sezione 2.2.2.

Si osservi che nel nostro caso non è stato applicato il procedimento di normalizzazione fotometrica alle apparenze ma, nel caso si decidesse di applicarlo, è importante ricordare che anche le apparenze sottoposte a fitting dovranno subire la stessa normalizzazione.

### 6.2.3 Pre-Computazioni

Ai fini di un'efficiente operazione dell'algoritmo di fitting, è fondamentale pre-calcolare tutto ciò che è indipendente dalle singole operazioni di fitting. Come descritto nella Sezione 3.4, le informazioni che si prestano ad essere pre-calcolate durante il training sono: il gradiente dell'apparenza media  $\nabla A_0$ , le Jacobiane  $\frac{\partial \mathbf{N}}{\partial \mathbf{q}}$  e  $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ , le immagini di steepest descent  $SD_i$  e l'Hessiana inversa  $\mathbf{H}^{-1}$ .

#### 6.2.3.1 Gradiente dell'Apparenza Media

Per stimare  $\nabla A_0$  si può procedere in vari modi, ma si osserva che l'algoritmo di *plane fitting*  $3 \times 3$  (lo stesso usato da Matthews et al. [25]) si è dimostrato più accurato rispetto al metodo incluso in MATLAB, anche se è sicuramente più oneroso.

Come dice il nome, un algoritmo di plane fitting  $3 \times 3$  determina il piano che meglio approssima, ai minimi quadrati, i valori di una funzione bidimensionale  $f(x,y)$  valutata su un reticolo quadrato (square lattice) di 9 elementi equispaziati. Il gradiente approssimato  $\nabla f$ , valutato su  $(x,y)$ , è quindi dato dal gradiente del piano ottenuto usando  $(x,y)$  come punto centrale del reticolo  $3 \times 3$ .

Se si assume un piano centrato nell'origine, dalla definizione parametrica del piano si vede che se  $a$  e  $b$  sono soluzioni di un problema di plane fitting tali che  $ax + by \approx f(x,y)$ , ne consegue banalmente che  $\partial f / \partial x \approx a$  e  $\partial f / \partial y \approx b$ . Il problema si riduce quindi a determinare  $a$  e  $b$  tali che il piano sia la migliore approssimazione ai minimi quadrati dei valori assunti dall'immagine nel reticolo, con l'unica accortezza di centrare il sistema di coordinate nella posizione  $(x,y, f(x,y))$  in cui si sta valutando il gradiente, altrimenti diventa necessario stimare anche lo scostamento del piano dall'origine.

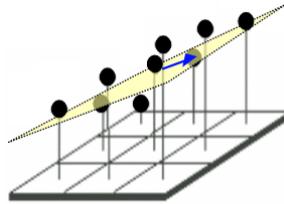


Figura 6.2. Uso di un piano per la stima del gradiente su un reticolo quadrato  $3 \times 3$  [1].

**Esempio:** Sia  $I$  un’immagine di cui si desidera determinare il gradiente, valutato in (33,30). L’immagine assume i seguenti valori nel reticolo  $3 \times 3$ :

$I(32,29) = 39$	$I(33,29) = 42$	$I(34,29) = 42$
$I(32,30) = 37$	$I(33,30) = 42$	$I(34,30) = 40$
$I(32,31) = 40$	$I(33,31) = 43$	$I(34,31) = 38$

A questo punto, si centra il sistema di coordinate nella posizione (33,30), ovvero per ogni punto  $(x,y)$  del reticolo si definisce  $v(x - 33, y - 30) = I(x,y) - I(33,30)$ :

$v(-1, -1) = -3$	$v(0, -1) = 0$	$v(1, -1) = 0$
$v(-1, 0) = -5$	$v(0, 0) = 0$	$v(1, 0) = -2$
$v(-1, 1) = -2$	$v(0, 1) = 1$	$v(1, 1) = -4$

Con questi valori è possibile determinare i coefficienti del piano  $a$  e  $b$  minimizzando:

$$\sum_{x=-1}^1 \sum_{y=-1}^1 [ax + by - v(x,y)]^2 \quad (6.3)$$

È importante fare una precisazione sul calcolo del gradiente nei “bordi” dell’apparenza media, in quanto questa sarà generalmente memorizzata in un’immagine con sfondo costante, ad esempio pari a 0. Applicando la stima del gradiente appena descritta nel bordo dell’apparenza porterà in generale a una stima poco corretta del gradiente, a causa dello stacco con lo sfondo. Per questo motivo, sono state usate le pseudo-immagini calcolate durante il warp delle apparenze per escludere dal reticolo pixel non appartenenti all’apparenza media, ricordando che il problema di plane fitting rimane risolvibile a patto che si considerino almeno 3 valori diversi e non collineari nel reticolo.

Infine, si osservi che nel caso di più canali cromatici il gradiente è stato calcolato indipendentemente per ogni singolo canale RGB. Tuttavia, non è escluso che altri metodi di gestione dell’informazione cromatica possano portare a risultati migliori.

### 6.2.3.2 Immagini di Steepest Descent e Hessiana Inversa

La stima delle Jacobiane del warp rispetto ai parametri  $\mathbf{p}$  e  $\mathbf{q}$  è una diretta applicazione delle Formule 3.31 e 3.30 (Sezione 3.3.2), eventualmente usando le informazioni già calcolate durante la fase di warp per ottimizzare la ricerca dei triangoli e il calcolo delle coordinate baricentriche.

È già stato spiegato come il calcolo delle Jacobiane richieda l'iterazione su tutti i vertici della forma media e, per ogni vertice, la valutazione delle coordinate baricentriche di tutte le posizioni interne ai triangoli che condividono quel particolare vertice. Apparentemente, questo sembra un processo complicato e inefficiente ma, nuovamente, le informazioni calcolate durante la fase di preparazione al warp possono semplificare di molto il tutto. Infatti, anziché iterare sui vertici e cercare i triangoli che condividono un dato vertice, è possibile iterare su tutti i pixel e determinare il triangolo di appartenenza, per poi calcolare  $\frac{\partial \mathbf{W}}{\partial u_i}$  e  $\frac{\partial \mathbf{W}}{\partial v_i}$  per ogni vertice del triangolo.

Anche il calcolo delle immagini di steepest descent è una diretta applicazione delle tecniche descritte nella Sezione 3.3.2, ma essendo questa la fase dell'addestramento più onerosa computazionalmente, essa va implementata in modo efficiente per evitare che l'addestramento richieda diversi minuti, piuttosto che qualche secondo.

Un modo per ottenere una velocità di calcolo ragionevole si ottiene osservando che il calcolo delle Formule 3.28 e 3.29 può essere suddiviso in tre fasi:

1. Calcolo delle sommatorie più interne:  $S_{j,i} = \sum_{\mathbf{w} \in \mathbf{s}_0} A_i(\mathbf{w}) \cdot \nabla A_0 \frac{\partial \mathbf{N}}{\partial q_j}$  e  $S_{j+4,i} = \sum_{\mathbf{w} \in \mathbf{s}_0} A_i(\mathbf{w}) \cdot \nabla A_0 \frac{\partial \mathbf{N}}{\partial p_j}$  per  $i = 1 \dots l$ .
2. Inizializzazione del valore delle immagini di steepest descent:  $SD_j(\mathbf{u}) = \nabla A_0 \frac{\partial \mathbf{N}}{\partial q_j}$  e  $SD_{j+4}(\mathbf{u}) = \nabla A_0 \frac{\partial \mathbf{N}}{\partial p_j}$ .
3. Iterazione delle sottrazioni:  $SD_j(\mathbf{u}) \leftarrow SD_j(\mathbf{u}) - S_{j,i} A_i(\mathbf{u})$  e  $SD_{j+4}(\mathbf{u}) \leftarrow SD_{j+4}(\mathbf{u}) - S_{j+4,i} A_i(\mathbf{u})$  per  $i = 1 \dots l$ .

Si sottolinea che se le apparenze non sono monocromatiche, questi calcoli si ripetono per ogni singolo canale, in modo da ottenere immagini di Steepest Descent con lo stesso numero di canali cromatici delle apparenze.

Allo scopo di massimizzare le prestazioni durante il fitting, le immagini di steepest descent devono in seguito essere vettorizzate e organizzate in forma matriciale:

$$\mathbf{SD} = \begin{bmatrix} \text{vec}(SD_1) & \dots & \text{vec}(SD_{m+4}) \end{bmatrix}^T \quad (6.4)$$

(il motivo della trasposizione sarà più chiaro a breve). Con questa formulazione, il calcolo dell'Hessiana è semplicemente:

$$\mathbf{H} = \mathbf{SD} \cdot \mathbf{SD}^T \quad (6.5)$$

di cui si può facilmente determinare l'inversa.

Infine, si ha che la relazione lineare tra immagine di errore e parametri incrementali può essere espressa da:

$$\mathbf{R} = \mathbf{H}^{-1} \mathbf{SD} \quad (6.6)$$

il cui uso sarà chiarito a breve.

## 6.3 Fitting 2D in Tempo Reale

A differenza dell'addestramento, il fitting è un processo iterativo, anche se la velocità di convergenza è piuttosto alta: nella stragrande maggioranza delle applicazioni pratiche, se dopo 10-20 iterazioni non si ha convergenza, è molto difficile che si possa ottenerla con ulteriori iterazioni.

Ad ogni iterazione, l'implementazione ripete i passi già descritti nella Sezione 3.4, prestando particolare attenzione all'efficienza. Nel nostro caso, questo ha significato usare il più possibile le operazioni vettoriali di MATLAB e l'uso di un MEX-file [24] per le operazioni di warp. Nonostante non si escluda la possibilità di implementare il warp efficientemente in MATLAB, la cosa sembra alquanto improbabile visto il numero di cicli richiesti (proporzionale al numero di pixel nelle apparenze), in cui sono necessarie diverse operazioni di moltiplicazione, divisione e troncamento.

La versione ottimizzata del warp è stata implementata in C++ e questa, pur non essendo concettualmente diversa dall'implementazione MATLAB, si è potuta avvantaggiare di una drastica riduzione nel costo dei cicli e degli accessi alle matrici, risultando in una riduzione nel tempo di esecuzione di un fattore 200 circa.

Segue l'esposizione dettagliata della nostra implementazione dell'algoritmo di fitting con composizione inversa, descritto nel Capitolo 3.

### 6.3.1 Inizializzazione

Ogni metodo di ottimizzazione di ricerca locale richiede una soluzione iniziale vicina a quella ottima, e il fitting con Active Appearance Models non fa eccezione. Anzi, su immagini di dimensione media, si osserva che il raggio di convergenza degli AAM raramente supera i 10 pixel, quindi una buona inizializzazione è estremamente importante.

L'inizializzazione, nel caso specifico degli AAM, potrebbe sembrare particolarmente complicata, con un elevato numero di parametri da stimare. In realtà, le uniche cose che è effettivamente necessario stimare sono la posizione e la dimensione approssimative dell'oggetto da riconoscere. Infatti, dalla Formula 2.22 si deduce che, una volta specificata una forma di inizializzazione, i parametri di apparenza sono univocamente determinati e sono dati dalla proiezione dell'immagine di errore

sulle mode di apparenza. In modo analogo, anche i parametri  $\mathbf{p}$  e  $\mathbf{q}$  iniziali sono univocamente determinabili dalla forma di inizializzazione, applicando la Formula 2.22.

Il problema di determinare posizione e dimensione approssimativa dell'oggetto deformabile di interesse in un'immagine non è comunque banale, soprattutto se si desidera una procedura sufficientemente efficiente da non compromettere le prestazioni real-time della procedura di fitting. Nel nostro caso, per questioni di tempo e per l'intrinseca difficoltà del problema, si è preferito escludere questo aspetto dall'implementazione della procedura di fitting, lasciando la parte di inizializzazione a discrezione dell'utente. Questo può eventualmente rivolgersi a una delle diverse soluzioni di *tracking* in tempo reale, sia nel caso specifico volti umani<sup>1</sup> [41], sia nel caso di oggetti generici, con implementazioni disponibili per OpenCV<sup>2</sup> e MPT<sup>3</sup>. In alternativa, è possibile applicare un approccio simile a quello descritto nella Sezione 5.2.2, sfruttando la stessa procedura di fitting per stimare la posizione dell'oggetto nell'immagine, usando una combinazione di inizializzazioni più o meno casuali e AAM a basse risoluzioni per migliorare efficienza e raggio di convergenza.

### 6.3.2 Calcolo dei Parametri Incrementali

A partire dall'immagine di errore, determinata sfruttando il warp associato alla forma corrente, è sufficiente applicare la Formula 3.33. Questa, nel caso si sia provveduto ad organizzare le immagini di Steepest Descent in forma matriciale (come nella Formula 6.4), si riduce a un singolo prodotto matrice-vettore:

$$\begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{p} \end{bmatrix} = \mathbf{H}^{-1} \mathbf{SD} \cdot \text{vec}(\mathbf{I}(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) - \mathbf{A}_0(\mathbf{u})) \\ = \mathbf{R} \cdot \text{vec}(\mathbf{I}(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})) - \mathbf{A}_0(\mathbf{u})) \quad (6.7)$$

A partire dai parametri di forma e di trasformazione globale incrementali, è possibile determinare le forme incrementali  $\mathbf{N}(\mathbf{s}_0; -\Delta \mathbf{q})$  e  $\mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{p})$ , usando le procedure descritte nella Sezione 3.3.5 per poi ottenere  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta \mathbf{q}, -\Delta \mathbf{p})$ . Quest'ultimo passo richiede il calcolo della trasformazione affine corrispondente a  $\mathbf{N}(\mathbf{s}_0; -\Delta \mathbf{q})$ , procedimento non banale e non descritto in dettaglio da Matthews et al. [25].

L'equazione necessaria si ricava espandendo  $\alpha$  e  $\beta$  all'interno dell'equazione che esprime il warp, con la loro definizione (Formule 2.11 e 2.12):

---

<sup>1</sup>Ad esempio, Raphael Cendrillon ha sviluppato e implementato un algoritmo di face tracking real-time [10], concettualmente simile agli AAM, disponibile al sito: <http://www.cendrillon.org/research/face/index.html>

<sup>2</sup>Open Source Computer Vision, usa l'algoritmo di Lienhart [21]. Maggiori dettagli al sito: <http://opencv.willowgarage.com/wiki/FaceDetection>,

<sup>3</sup>The Machine Perception Toolbox, implementa l'algoritmo di Viola e Jones [40]: <http://mplab.ucsd.edu/grants/project1/free-software/mptwebsite/introduction.html>

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} + \frac{(u - u_i^0)(v_k^0 - v_i^0) - (v - v_i^0)(u_k^0 - u_i^0)}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_j \\ v_j \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) + \frac{(v - v_1^0)(u_2^0 - u_1^0) - (u - u_1^0)(v_2^0 - v_1^0)}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) \quad (6.8)$$

dove  $[u_{i,j,k} \ v_{i,j,k}]^T$  sono i vertici di un qualsiasi triangolo in  $\mathbf{N}(\mathbf{s}_0; -\Delta \mathbf{q})$  e  $[u_{i,j,k}^0 \ v_{i,j,k}^0]^T$  sono i vertici corrispondenti in  $\mathbf{s}_0$ .

Si è quindi interessati ad ottenere la formulazione affine equivalente alla Formula 6.8:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} a_2 & a_3 \\ a_5 & a_6 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} a_1 \\ a_4 \end{bmatrix} \quad (6.9)$$

e si dimostra che i parametri della trasformazione affine sono:

$$\begin{bmatrix} a_1 \\ a_4 \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} + \frac{v_i^0(u_k^0 - u_i^0) - u_i^0(v_k^0 - v_i^0)}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_j \\ v_j \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) + \frac{u_i^0(v_j^0 - v_i^0) - v_i^0(u_j^0 - u_i^0)}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) \quad (6.10)$$

$$\begin{bmatrix} a_2 \\ a_5 \end{bmatrix} = \frac{v_k^0 - v_i^0}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_j \\ v_j \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) + \frac{v_i^0 - v_j^0}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) \quad (6.11)$$

$$\begin{bmatrix} a_3 \\ a_6 \end{bmatrix} = \frac{u_j^0 - u_i^0}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_j \\ v_j \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) + \frac{u_i^0 - u_k^0}{(u_j^0 - u_i^0)(v_k^0 - v_i^0) - (v_j^0 - v_i^0)(x_k^0 - x_i^0)} \left( \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right) \quad (6.12)$$

In questo modo è possibile applicare la trasformazione lineare data dalla Formula 6.9 a  $\mathbf{W}(\mathbf{s}_0; -\Delta \mathbf{p})$ , e ottenere così  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta \mathbf{q}, -\Delta \mathbf{p})$ .

Non rimane che vedere come comporre il warp incrementale con quello corrente per ottenere il risultato dell'operazione di fitting per l'iterazione.

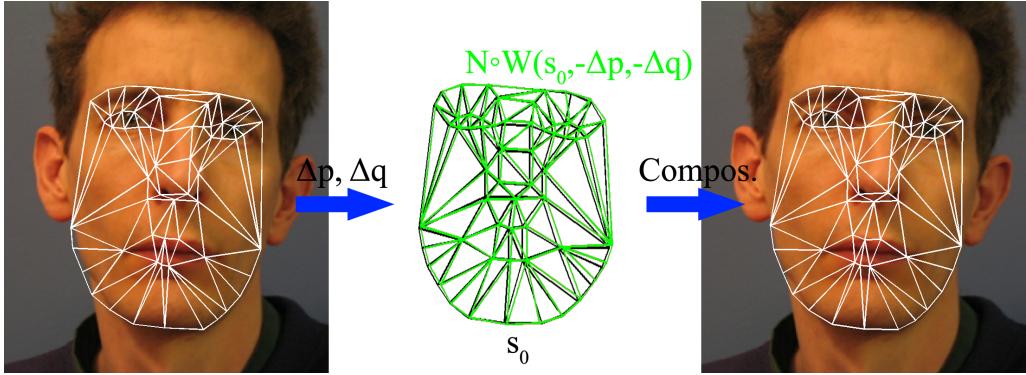


Figura 6.3. Illustrazione di un’iterazione della procedura di fitting: a sinistra, la forma corrente, usata per determinare l’immagine di errore, e da cui si ricava la stima dei parametri incrementali; al centro, la forma associata al warp incrementale, sovrapposta alla forma media; a destra, il risultato della composizione di warp.

### 6.3.3 Aggiornamento della Soluzione

La composizione di warp (Sezione 3.3.5) si basa su un concetto simile al warp lineare a tratti, ed è un’altra funzionalità difficile da implementare efficientemente. Applicando direttamente la definizione, questa consiste nell’iterare su ogni vertice  $\mathbf{u}_i$  della forma associata al warp incrementale  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}; -\Delta\mathbf{q}, -\Delta\mathbf{p})$ , e valutare  $\mathbf{N} \circ \mathbf{W}(\mathbf{u}_i; \mathbf{q}, \mathbf{p})$  su ogni triangolo che include il vertice  $\mathbf{u}_i$ ; la media dei risultati di questi warp produce il nuovo valore di  $\mathbf{u}_i$ . Questa procedura è efficiente solo se è possibile determinare in poco tempo i triangoli che condividono un particolare vertice, tipicamente usando un vettore di liste costruito durante il training.

Una soluzione efficiente che non richiede la creazione di ulteriori strutture dati consiste nell’iterare sui triangoli, piuttosto che sui vertici: per ogni triangolo si valuta il warp ponendo di volta in volta un vertice diverso pari a  $\mathbf{u}_i$ , e rinominando gli altri due come  $\mathbf{u}_j$  e  $\mathbf{u}_k$ ; tutti i risultati della composizione sono quindi accumulati in un’unica forma di cui si calcola una media pesata dividendo per un vettore contenente il numero di volte che il warp è stato valutato per un particolare vertice della forma.

Entrambi i metodi sono ugualmente validi e producono risultati analoghi, anche se il metodo di iterazione sui triangoli nel nostro caso si è dimostrato più efficiente. A prescindere dal metodo usato, se si denota la forma prodotta dalla composizione con  $\mathbf{s}$  e il suo precedente valore (determinato nell’iterazione precedente) con  $\mathbf{s}_{old}$ , è possibile (ma non strettamente necessario) definire una misura di convergenza del fitting. Nel nostro caso si è optato per una misura Root Mean Square:

$$\sqrt{\frac{\|\mathbf{s} - \mathbf{s}_{old}\|^2}{n}} \quad (6.13)$$

e una soglia (inclusiva) di 0.3 per dichiarare la convergenza.

### 6.3.4 Migliorare la Capacità di Convergenza

Un’implementazione che segue rigorosamente la formulazione teorica degli AAM con composizione inversa è probabilmente soggetta a soffrire di problematiche che rendono le prestazioni di fitting piuttosto scarse.

Nella nostra esperienza, si sono osservati in particolare due fenomeni: divergenza del fitting in caso di inizializzazione lontana più di qualche pixel dall’oggetto da riconoscere; e, in certi dataset, un’incapacità dell’algoritmo di stabilizzarsi in corrispondenza della soluzione corretta, anche se inizializzato perfettamente.

Nel primo caso, il fenomeno è dovuto al cambiamento dei parametri di forma da parte dell’algoritmo anche quando la forma corrente è ancora molto lontana dalla soluzione, con la conseguenza che ripetuti aggiornamenti imprecisi della forma portano a divergenza del risultato.

Mentre nel secondo caso, considerando la teoria alla base degli AAM, nulla sembra suggerire che il fitting possa essere soggetto a simili limitazioni. Nella nostra esperienza si è comunque osservato come i risultati dell’algoritmo di fitting diventano sempre più instabili al crescere del numero di mode del modello di forma, o qualora le deformazioni specificate nel training set siano specificate in modo poco accurato o siano “accoppiate” tra loro. Di conseguenza, verrebbe da supporre che essendo il metodo di fitting con composizione inversa basato su un procedimento di ottimizzazione matematico e non empirico, esso in qualche modo richiede che il modello statistico usato per definire l’oggetto deformabile sia perfetto o quasi.

Una soluzione semplice che permette di aumentare il raggio di convergenza dell’algoritmo di fitting consiste nell’ignorare completamente l’aggiornamento dei parametri di forma nelle primissime iterazioni (5, nel nostro caso). Si effettua così un allineamento di immagini del tipo Lucas-Kanade, analogo a quello descritto nella Sezione 3.1.3, che dovrebbe avvicinare la forma di inizializzazione all’oggetto di interesse, senza alterare la forma in modo irreparabile. La Figura 6.4 illustra l’impatto che può avere questo semplice accorgimento, in casi in cui l’algoritmo standard fallisce in modo decisamente evidente.

Le problematiche dovute a imprecisioni o rumore nel modello di forma sono significativamente più difficili da risolvere, e la cosa migliore sarebbe prevenirle completamente usando un training set di alta qualità, costituito da immagini con poco rumore, acquisite in condizioni simili e da annotazioni molto precise in cui feature corrispondenti sono specificate in modo non ambiguo. Idealmente, le uniche variazioni di forma introdotte dovrebbero essere dovute a deformazioni dell’oggetto, e ogni immagine del training set dovrebbe introdurre un solo tipo di deformazione, discostandosi il meno possibile dalle altre. A questo scopo, sarebbe preferibile annotare manualmente solo un numero molto limitato di immagini, e usare un AAM costruito a partire da queste per annotare le altre immagini del training set (usando l’algoritmo di fitting), e introducendo variazione di forma solo se strettamente necessario.

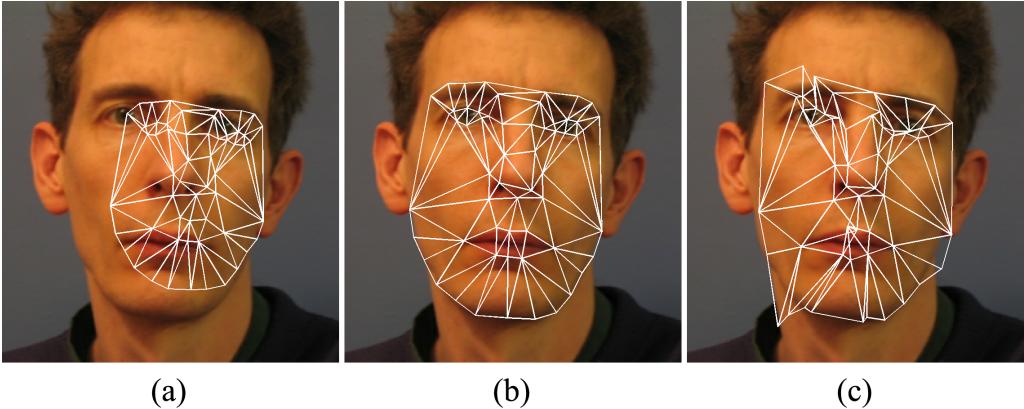


Figura 6.4. Effetto dell'introduzione di un allineamento di immagine nelle prime 5 iterazioni dell'algoritmo di fitting: (a) è la posizione iniziale fornita alla procedura di fitting, (b) è il risultato prodotto al termine di 20 iterazioni, evitando di aggiornare i parametri di forma nelle prime 5; (c) è il risultato del fitting prodotto aggiornando sia i parametri di forma che di trasformazione globale in tutte le 20 iterazioni.

La metodologia di training appena descritta, è più semplice da descrivere in teoria che applicare in pratica, soprattutto quando si ha a che fare con deformazioni complesse (come ad esempio i cambiamenti di posa). Si è riusciti comunque a limitare la tendenza alla divergenza della procedura di fitting agendo sulla fase di aggiornamento della forma, supponendo che se si ha a che fare con un modello di forma non perfetto, le forme sono in un certo senso in contraddizione tra loro, e che un aggiornamento ai parametri di forma associati a queste mode può essere causato da immagini di errore molto simili, portando a instabilità nell'algoritmo. Si è quindi pensato di *smorzare* i parametri incrementali calcolati a partire dall'immagine di errore, in modo da rendere più significative le mode di forma più rappresentate nel training set, e meno significative le altre. Un esempio di smorzamento potrebbe essere:

$$\Delta p_i \leftarrow \frac{\Lambda_i}{\Lambda_1} \Delta p_i \quad (6.14)$$

con  $\Lambda_i$  autovalore corrispondente alla moda di forma  $i$ -esima. In questo modo, il primo parametro di forma sarà aggiornato normalmente (il peso associato è 1), mentre gli altri avranno peso minore. Questo esempio è abbastanza drastico, e risultati di fitting prodotti tendono ad adattarsi molto più difficilmente a deformazioni poco rappresentate nel training set. Per questo motivo, sarebbe preferibile cercare un compromesso smorzando un numero sempre minore di parametri di forma col passare delle iterazioni, eventualmente permettendo a tutte le deformazioni significative di essere ricostruite. Si tratta comunque di un'approccio euristico, che deve essere calibrato a seconda dell'applicazione considerata e che non si sostituisce a un training set di buona qualità. Una comparazione qualitativa di fitting con e senza smorzamento è illustrata nella Figura 6.5.

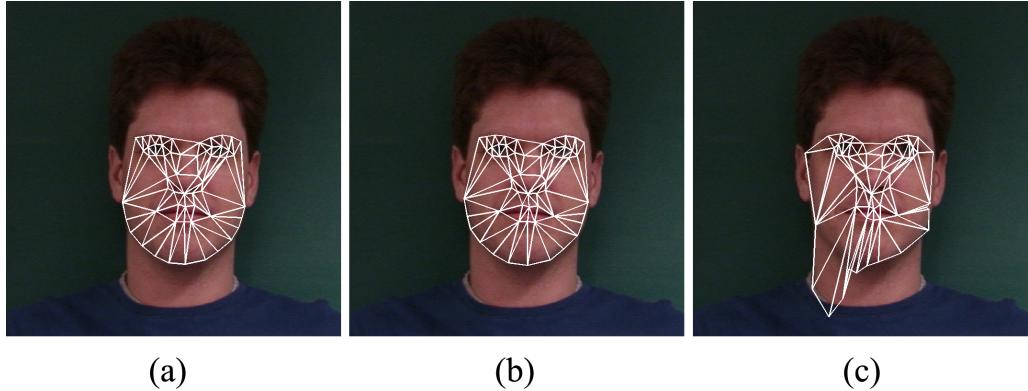


Figura 6.5. Effetto dello smorzamento dei parametri in una situazione di instabilità della procedura di fitting: (a) è la posizione iniziale fornita alla procedura di fitting, (b) è il risultato prodotto al termine di 20 iterazioni, smorzando i parametri incrementali con la Formula 6.14; (c) è il risultato prodotto dall'algoritmo di fitting senza smorzamento.

## 6.4 Ricostruzione della Forma dal Movimento

Si assuma che in ingresso alla procedura siano fornite un buon numero di forme bidimensionali che rappresentano l'evoluzione nel tempo di un oggetto deformabile, ottenute ad esempio applicando il fitting AAM su un video dell'oggetto.

Un'implementazione dell'algoritmo di ricostruzione della forma dal movimento di Xiao et al. [43] (descritto nella Sezione 4.2), senza bisogno di ulteriori informazioni, permette di ricostruire la geometria tridimensionale dell'oggetto di interesse. In particolare, l'aspetto più interessante dell'algoritmo è che determina anche le mode di forma 3D dell'oggetto.

Nel seguito verrà descritto come implementare i procedimenti matematici usati dall'algoritmo descritto nella Sezione 4.2, con particolare enfasi su come ordinare le forme (in modo che le prime  $\bar{m}+1$  siano indipendenti), come specificare in modo efficiente i vincoli e come risolvere le ambiguità inerenti al procedimento.

### 6.4.1 Preparazione della Matrice delle Misure

Date le  $N+1$  forme di ingresso  $\mathbf{s}^0 \dots \mathbf{s}^N$ , si va a costruire la matrice delle misure, che è subito registrata applicando la formula:

$$\tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{s}^0 \\ \vdots \\ \mathbf{s}^N \end{bmatrix} - \begin{bmatrix} \bar{u}^0 & \dots & \bar{u}^0 \\ \bar{v}^0 & \dots & \bar{v}^0 \\ \vdots & \vdots & \vdots \\ \bar{u}^N & \dots & \bar{u}^N \\ \bar{v}^N & \dots & \bar{v}^N \end{bmatrix} = \begin{bmatrix} u_1^0 & u_2^0 & \dots & u_n^0 \\ v_1^0 & v_2^0 & \dots & v_n^0 \\ \vdots & \vdots & \vdots & \vdots \\ u_1^N & u_2^N & \dots & u_n^N \\ v_1^N & v_2^N & \dots & v_n^N \end{bmatrix} - \begin{bmatrix} \bar{u}^0 & \dots & \bar{u}^0 \\ \bar{v}^0 & \dots & \bar{v}^0 \\ \vdots & \vdots & \vdots \\ \bar{u}^N & \dots & \bar{u}^N \\ \bar{v}^N & \dots & \bar{v}^N \end{bmatrix} \quad (6.15)$$

#### 6.4. RICOSTRUZIONE DELLA FORMA DAL MOVIMENTO

dove  $[\bar{u}^i \ \bar{v}^i]^T$  è il baricentro di  $\mathbf{s}^i$ .

Il numero di mode di forma 3D  $\bar{m}$  (se non specificato dall'utente della procedura) è determinato, in assenza di rumore, a partire dalla Formula 4.13:

$$\bar{m} + 1 \leq \left\lfloor \frac{\text{rank}(\tilde{\mathbf{W}})}{3} \right\rfloor \quad (6.16)$$

In generale, le forme in ingresso sono soggette a rumore, e il valore di  $\bar{m}$  determinato dalla Formula 6.16 risulterebbe maggiore di quello effettivo, rendendo il problema di ricostruzione malcondizionato. Perciò, in presenza di rumore si va a sostituire a  $\text{rank}(\tilde{\mathbf{W}})$  il numero di valori singolari di  $\tilde{\mathbf{W}}$  per cui vale  $\sigma_1/\sigma_i < r$ , dove  $\sigma_i$  rappresenta l'i-esimo valore singolare di  $\tilde{\mathbf{W}}$  e i valori singolari si intendono in ordine decrescente ( $r$  è una soglia di sensibilità al rumore, e nel nostro caso un valore di 100 si è rivelato adeguato).

Dato  $\bar{m}$ , è possibile fattorizzare  $\tilde{\mathbf{W}}$  applicando la Formula 4.14, ignorando completamente il valore di  $\tilde{\mathbf{B}}$  (in seguito si vedrà il perché).

Prima di procedere alla specifica dei vincoli necessari a determinare la matrice correttiva  $\mathbf{G}$ , si ricordi (Sezione 4.2.3) che la specifica dei vincoli di base è fondata sull'assunzione che le prime  $\bar{m} + 1$  forme (ovvero le prime  $2(\bar{m} + 1)$  righe di  $\tilde{\mathbf{W}}$ ) siano indipendenti.

Ma prima ancora di vedere come le righe di  $\tilde{\mathbf{M}}$  siano riordinate, in modo da rendere il più efficace possibile la specifica dei vincoli di base, si ritiene opportuno fare una riflessione sulla forma 3D media  $\bar{\mathbf{s}}_0$ . Infatti, se si osserva bene la Formula 4.24, che esprime i vincoli di base, si nota che i parametri di forma 3D  $\bar{p}_i^0$  ricostruiti dovranno essere pari a zero per  $i = 0 \dots \bar{m}$ . Ma questo significa che  $\bar{\mathbf{s}}_0$  non è la forma media (altrimenti  $\bar{p}_i^0$  dovrebbe essere sempre pari a 1), ma una moda di forma del modello semplificato:

$$\bar{\mathbf{s}}^i = \sum_{j=0}^{\bar{m}} \bar{p}_j^i \bar{\mathbf{s}}_j \quad (6.17)$$

A seguito di questa osservazione, si riconosce che non ha più molto senso trattare  $\bar{\mathbf{s}}_0$  separatamente dalle altre mode di forma 3D, e si pone:

$$K = \bar{m} + 1 \quad (6.18)$$

nuovo numero delle mode di forma 3D.

Determinare  $K$  forme 2D che siano semplicemente indipendenti non è difficile, è sufficiente che il numero di condizionamento della matrice costruita a partire da queste (in modo analogo a  $\mathbf{W}$ ) sia diverso da zero. Ma il numero di condizionamento non indica solo se l'insieme di forme è indipendente o meno, dà anche una misura di quanto queste siano indipendenti, perciò ha senso

pensare che si possano ottenere risultati di ricostruzione migliori scegliendo un sottoinsieme di forme con associato il minor numero di condizionamento possibile.

---

**Algorithm 1** Un possibile algoritmo randomizzato per determinare un sottoinsieme di forme con basso numero di condizionamento associato.

---

```

// Numero di condizionamento delle prime 2K righe di  $\tilde{W}$ 
best_sel = {1, ..., 2K};
best_cn = cond( $\tilde{W}_{best\_sel}$ );
// Itera su tutte le coppie di forme, evitando di usare
// le ultime (K - 2)
for (i = 0; i ≤ (N - K + 1); i++) do
    for (j = (i + 1); j ≤ (N - K + 2); j++) do

        // Ripeti K volte la ricerca randomizzata (la scelta di K
        // è arbitraria)
        for (it = 1; it ≤ K; it++) do
            // Indici corrispondenti alla i-esima e j-esima forma
            sel = {2i + 1, 2i + 2, 2j + 1, 2j + 2};
            id = j;

        // Scegli casualmente le restanti (K - 2) forme
        for (n = 3; n ≤ K; n++) do
            // Indice casuale compreso tra (id + 1) e
            // (N - K + n), in modo da avere almeno (K - n)
            // forme a disposizione in seguito
            id =  $\lfloor rand() * (N + n - K - id - 1) + id + 1 \rfloor$ ;

            // Indici alla forma numero 'id'
            sel = sel ∪ {id * 2 + 1, id * 2 + 2};
        endfor

        // Numero di condizionamento associato alle
        // righe in 'sel'
        cn = cond( $\tilde{W}_{sel}$ );

        // (Eventuale) aggiornamento della soluzione
        if cn < best_cn
            best_cn = cn;
            best_sel = sel;
        endif
    endfor
endfor
endfor

```

---

Ovviamente, con valori di  $N$  non banali, la ricerca del sottoinsieme di forme con il più basso numero di condizionamento ha un costo computazionale inaccettabile, visto che è necessaria una ricerca

esaustiva su tutte le possibili combinazioni di forme, quindi l'unica alternativa ragionevole consiste in una ricerca randomizzata.

L'Algoritmo 1 memorizza, in "best\_sel", gli indici delle forme con associato il minor numero di condizionamento, e con questi è possibile riordinare le righe di  $\tilde{\mathbf{M}}$  in modo che siano indipendenti.

Qualsiasi metodo sia stato usato per l'ordinamento è importante osservare che le informazioni necessarie a ripristinare l'ordinamento originale devono essere conservate nel caso si intenda effettuare la ricostruzione 3D di tutte le  $N + 1$  forme.

#### 6.4.2 Vincoli di Rotazione

I vincoli di rotazione servono ad assicurare che le matrici di proiezione ricostruite siano conformi al modello prospettico debole, assicurando che i loro assi di proiezione siano ortonormali (a meno di un fattore di scala comune). La motivazione teorica di questi vincoli è già stata esposta nella Sezione 4.2.2, quindi in seguito si vedrà solo come specificare i vincoli in forma di sistema lineare.

Si osserva che il risultato prodotto da un generico vincolo del tipo  $\mathbf{a}\mathbf{Q}_k\mathbf{b}^T$  può essere così rappresentato:

$$\begin{aligned}
 \mathbf{a}\mathbf{Q}_k\mathbf{b}^T &= \begin{bmatrix} a_1 & \dots & a_{3K} \end{bmatrix} \begin{bmatrix} q_{1,1} & \dots & q_{1,3K} \\ \vdots & \vdots & \vdots \\ q_{3K,1} & \dots & q_{3K,3K} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_{3K} \end{bmatrix} \quad (6.19) \\
 &= b_1 (a_1 q_{1,1} + a_2 q_{2,1} + a_3 q_{3,1} + \dots + a_{3K} q_{3K,1}) \\
 &+ b_2 (a_1 q_{1,2} + a_2 q_{2,2} + a_3 q_{3,2} + \dots + a_{3K} q_{3K,2}) \\
 &+ b_3 (a_1 q_{1,3} + a_2 q_{2,3} + a_3 q_{3,3} + \dots + a_{3K} q_{3K,3}) \\
 &+ \vdots \\
 &+ b_{3K} (a_1 q_{1,3K} + a_2 q_{2,3K} + a_3 q_{3,3K} + \dots + a_{3K} q_{3K,3K}) \\
 &= a_1 b_1 q_{1,1} + a_1 b_2 q_{1,2} + \dots + a_2 b_1 q_{2,1} + a_2 b_2 q_{2,2} + \dots \\
 &= \sum_{i=1}^{3K} \sum_{j=1}^{3K} a_i b_j q_{i,j}
 \end{aligned}$$

da questo, se si considerano le incognite di  $\mathbf{Q}_k$  in forma vettoriale, si ricava la formulazione equivalente:

$$\begin{bmatrix} a_1 b_1 & a_1 b_2 & \dots & a_{3K} b_{3K-1} & a_{3K} b_{3K} \end{bmatrix} \begin{bmatrix} q_{1,1} \\ q_{1,2} \\ \vdots \\ q_{3K,3K-1} \\ q_{3K,3K} \end{bmatrix} = \text{vec}^T(\mathbf{a}^T \mathbf{b}) \cdot \text{vec}^T(\mathbf{Q}_k) \quad (6.20)$$

dove  $vec^T()$  è un operatore di vettorizzazione analogo a  $vec()$ , ma che vettorizza per righe (e genera vettori riga). Si osservi comunque che, per la simmetria di  $\mathbf{Q}_k$ ,  $vec^T(\mathbf{Q}_k) = vec(\mathbf{Q}_k)$ . La scelta di vettorizzare per riga o colonna è arbitraria, e nel nostro caso è stata scelta la vettorizzazione per righe allo scopo di facilitare il confronto con l'implementazione di Xiao.

Applicare la relazione espressa dalla Formula 6.20 ai vincoli di rotazione consiste semplicemente nel formulare il sistema lineare:

$$\begin{cases} vec^T \left( \tilde{\mathbf{M}}_{2j+1} \tilde{\mathbf{M}}_{2j+1}^T - \tilde{\mathbf{M}}_{2j+2} \tilde{\mathbf{M}}_{2j+2}^T \right) vec(\mathbf{Q}_k) = 0 \\ vec^T \left( \tilde{\mathbf{M}}_{2j+1} \tilde{\mathbf{M}}_{2j+2}^T \right) vec(\mathbf{Q}_k) = 0 \end{cases} \quad (6.21)$$

per  $j = 0 \dots N$ . Tutti questi vincoli costituiscono la matrice dei vincoli di rotazione  $\mathbf{C}_{rot}$ , mentre i termini noti (anche se tutti nulli) si identificano con  $\mathbf{b}_{rot}$  e ovviamente l'incognita è data da  $vec(\mathbf{Q}_k)$ .

### 6.4.3 Vincoli di Base e Simmetria

È già stato sottolineato che i vincoli di ortonormalità non sono sufficienti a determinare univocamente  $\mathbf{Q}_k$ , e che solo grazie a dei vincoli aggiuntivi, detti di base, che è possibile vincolare opportunamente il problema e ottenere una soluzione in forma chiusa.

Non c'è molto altro da dire a riguardo e si vede semplicemente che, usando la formulazione per i vincoli definita nella Formula 6.20, i vincoli di base sono:

$$\begin{cases} vec^T \left( \tilde{\mathbf{M}}_{2i+1}^T \tilde{\mathbf{M}}_{2j+1} \right) vec(\mathbf{Q}_k) = 1 \\ vec^T \left( \tilde{\mathbf{M}}_{2i+2}^T \tilde{\mathbf{M}}_{2j+2} \right) vec(\mathbf{Q}_k) = 1 \end{cases} \quad i = j = k \quad (6.22)$$

$$\begin{cases} vec^T \left( \tilde{\mathbf{M}}_{2i+1}^T \tilde{\mathbf{M}}_{2j+1} \right) vec(\mathbf{Q}_k) = 0 \\ vec^T \left( \tilde{\mathbf{M}}_{2i+2}^T \tilde{\mathbf{M}}_{2j+2} \right) vec(\mathbf{Q}_k) = 0 \end{cases} \quad i \neq k \quad (6.23)$$

$$\begin{cases} vec^T \left( \tilde{\mathbf{M}}_{2i+1}^T \tilde{\mathbf{M}}_{2j+2} \right) vec(\mathbf{Q}_k) = 0 \\ vec^T \left( \tilde{\mathbf{M}}_{2i+2}^T \tilde{\mathbf{M}}_{2j+1} \right) vec(\mathbf{Q}_k) = 0 \end{cases} \quad i \neq k \text{ oppure } i = j = k \quad (6.24)$$

In questo caso, si ricorda che i vincoli di base dipendono dal valore di  $k$ , e quindi è necessario definire una matrice dei vincoli  $\mathbf{C}_k$  per ognuna delle matrici  $\mathbf{Q}_k$  incognite (per i termini noti non è necessario purché i vincoli siano disposti sempre allo stesso modo):

$$\begin{bmatrix} \mathbf{C}_{rot} \\ \mathbf{C}_{base}^k \end{bmatrix} vec(\mathbf{Q}_k) = \begin{bmatrix} \mathbf{b}_{rot} \\ 1 \\ 1 \\ \mathbf{0} \end{bmatrix} \equiv \mathbf{C}_k vec(\mathbf{Q}_k) = \mathbf{b} \quad (6.25)$$

#### 6.4. RICOSTRUZIONE DELLA FORMA DAL MOVIMENTO

con  $\mathbf{C}_{base}^k$  matrice dei vincoli di base per un dato  $k$ , con i vincoli non nulli nelle prime due righe.

In realtà, dal punto di vista pratico, i vincoli di base non sono gli ultimi vincoli da considerare prima di poter determinare univocamente le matrici  $\mathbf{Q}_k$ . Infatti, nonostante si sia già detto più volte che le matrici  $\mathbf{Q}_k$  devono essere simmetriche, in nessun modo i vincoli specificati finora impongono la simmetria delle matrici. E questo rende il problema impossibile da risolvere, in quanto ci sono ancora troppi gradi di libertà.

È certamente possibile imporre la simmetria aggiungendo delle righe alle matrici dei vincoli che impongono l'uguaglianza di elementi in posizioni simmetriche, ed è sicuramente l'approccio più semplice. Alternativamente, se si sceglie un approccio un pò meno semplice, è possibile imporre gli stessi vincoli di simmetria e allo stesso tempo velocizzare la successiva soluzione dei sistemi lineari.

Infatti, se si raccolgono gli elementi uguali di  $\mathbf{Q}_k$  nella Formula 6.19, si osserva che il numero di incognite effettive si riduce, e i coefficienti dei vincoli associati a queste incognite sono ottenuti sommando elementi della matrice  $\mathbf{a}^T \mathbf{b}$  in posizioni simmetriche tra loro:

$$\begin{bmatrix} a_1 b_1 \\ (a_1 b_2 + a_2 b_1) \\ (a_1 b_3 + a_3 b_1) \\ \vdots \\ (a_1 b_{3K} + a_{3K} b_1) \\ a_2 b_2 \\ (a_2 b_3 + a_3 b_2) \\ \vdots \\ a_{3K} b_{3K} \end{bmatrix}^T \begin{bmatrix} q_{1,1} \\ q_{1,2} \\ q_{1,3} \\ \vdots \\ q_{1,3K} \\ q_{2,2} \\ q_{2,3} \\ \vdots \\ q_{3K,3K} \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ (a_1 b_2 + a_2 b_1) \\ (a_1 b_3 + a_3 b_1) \\ \vdots \\ (a_1 b_{3K} + a_{3K} b_1) \\ a_2 b_2 \\ (a_2 b_3 + a_3 b_2) \\ \vdots \\ a_{3K} b_{3K} \end{bmatrix}^T \mathbf{q}_k \quad (6.26)$$

Applicare questa tecnica alle matrici  $\mathbf{C}_k$  consiste in una semplice iterazione tra tutte le coppie di indici  $(i,j) \in [1,3K]^2$  con  $i < j$ , sommando alle colonne di  $\mathbf{C}_k$  di indice  $(3K(i-1) + j)$  le colonne di indice  $(3K(j-1) + i)$ . Successivamente, si ridimensionano le  $\mathbf{C}_k$  in modo da eliminare le ultime  $3K(3K-1)/2$  colonne che non contengono più informazioni rilevanti. È facile vedere che queste operazioni corrispondono a sommare coefficienti dei vincoli che, se in forma matriciale, sarebbero in posizioni simmetriche tra loro.

Ovviamente, con questa forma ridotta dei vincoli, la soluzione del problema lineare non sarà più la versione vettorizzata di  $\mathbf{Q}_k$  ma, come si vedrà a breve, questa non è difficile da ripristinare.

#### 6.4.4 Stima Iniziale di $\mathbf{M}$

Le matrici  $\mathbf{Q}_k$  si determinano a partire dalle matrici dei vincoli, risolvendo i sistemi lineari associati ai minimi quadrati. Una semplice soluzione di questo tipo è:

$$\mathbf{q}_k = \mathbf{C}_k^+ \mathbf{b} = (\mathbf{C}_k^T \mathbf{C}_k)^{-1} \mathbf{C}_k^T \mathbf{b} \quad (6.27)$$

dove  $\mathbf{q}_k$  rappresenta il vettore delle incognite uniche di  $\mathbf{Q}_k$  e  $\mathbf{C}_k^+$  è la pseudoinversa di Moore-Penrose della matrice dei vincoli.

Ricostruire le matrici  $\mathbf{Q}_k$  a partire dai vettori  $\mathbf{q}_k$  è semplice se si considera che gli elementi di  $\mathbf{q}_k$  corrispondono alla porzione triangolare superiore di  $\mathbf{Q}_k$ , presi in ordine per riga (o, per simmetria, gli elementi della porzione triangolare inferiore presi per colonna):

$$\begin{bmatrix} [q_{1,1} & q_{1,2} & q_{1,3} & q_{1,4} & \dots & q_{1,3K}] \\ 0 & [q_{2,2} & q_{2,3} & q_{2,4} & \dots & q_{2,3K}] \\ 0 & 0 & [q_{3,3} & q_{3,4} & \dots & q_{3,3K}] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & [q_{3K,3K}] \end{bmatrix} \quad (6.28)$$

da cui, ricostruire anche la parte triangolare inferiore (o superiore se è stata ricostruita per prima la parte inferiore) è banale, sapendo che le matrici  $\mathbf{Q}_k$  sono simmetriche.

Le tri-colonne di  $\mathbf{G}$  possono quindi essere determinate a partire dalla decomposizione SVD di  $\mathbf{Q}_k$ :

$$\mathbf{g}_k = \mathbf{U}_{3K \times 3} \mathbf{D}_{3 \times 3}^{\frac{1}{2}} \quad (6.29)$$

dove  $\mathbf{Q}_k = \mathbf{U} \mathbf{D} \mathbf{V}^T = \mathbf{g}_k \mathbf{g}_k^T$ ,  $\mathbf{U}_{3K \times 3}$  indica la sottomatrice formata dalle prime tre colonne di  $\mathbf{U}$ , e in modo analogo si agisce per  $\mathbf{D}_{3 \times 3}$ . Inoltre si ricorda che essendo  $\mathbf{D}$  una matrice diagonale, l'operazione di radice è data semplicemente dall'applicazione della radice quadrata su ogni elemento della diagonale.

Come è già stato accennato, la soluzione del problema non si conclude con il calcolo delle  $K$  tri-colonne di  $\mathbf{G}$ . In pratica, si arriva presto alla conclusione che per determinare la matrice correttiva finale occorre agire sulle stime intermedie di  $\mathbf{M}$ , fino al punto da rendere il calcolo della  $\mathbf{G}$  definitiva quasi irrilevante. Per questo motivo, d'ora in poi le  $\mathbf{g}_k$  non saranno più considerate, in quanto il

#### 6.4. RICOSTRUZIONE DELLA FORMA DAL MOVIMENTO

loro unico scopo è provvedere alla prima stima delle tri-colonne di  $\mathbf{M}$ :

$$\mathbf{m}_k = \tilde{\mathbf{M}}\mathbf{g}_k = \begin{bmatrix} \tilde{p}_k^0 \tilde{\mathbf{P}}_k^0 \\ \tilde{p}_k^1 \tilde{\mathbf{P}}_k^1 \\ \vdots \\ \tilde{p}_k^N \tilde{\mathbf{P}}_k^N \end{bmatrix} \quad (6.30)$$

in cui i valori di  $\tilde{\mathbf{P}}_k^i$  e  $\tilde{p}_k^i$  sono appunto delle stime di  $\mathbf{P}_k^i$  e  $p_k^i$ , rispettivamente, e non sono necessariamente simili alla soluzione cercata. Infatti, i valori di  $\mathbf{m}_k$  sono ancora lontani dal ricostruire efficacemente la matrice  $\mathbf{M}$ , in quanto sono in sistemi di riferimento diversi (essendo stati stimati in modo indipendente tra loro). Inoltre, è ancora necessario rinforzare l'ortogonalità delle matrici di proiezione e correggere opportunamente i valori dei parametri di forma 3D, in quanto i processi numerici usati non sono perfetti.

I vincoli di base dovrebbero aver assicurato che i coefficienti  $\tilde{p}_k^k$  siano pari a 1, ma all'atto pratico si rivela necessario normalizzare le tri-colonne  $\mathbf{m}_k$  per rinforzare il vincolo:

$$\mathbf{m}_k \leftarrow \frac{\mathbf{m}_k}{\sqrt{(\tilde{p}_k^k)^2 \mathbf{i}_k \mathbf{i}_k^T}} \quad (6.31)$$

con  $\mathbf{i}_k$  primo asse della matrice di proiezione  $\tilde{\mathbf{P}}_k^k$  (la scelta dell'asse è arbitraria).

In seguito, è possibile rinforzare l'ortogonalità degli assi delle matrici di proiezione e allo stesso tempo estrarre le stime dei parametri di forma 3D effettuando una decomposizione SVD su tutte le coppie di righe delle matrici  $\mathbf{m}_k$ :

$$\tilde{p}_k^i \tilde{\mathbf{P}}_k^i = \mathbf{U} \mathbf{D} \mathbf{V}^T = \mathbf{U} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{bmatrix} \mathbf{V}^T \quad (6.32)$$

da cui:

$$p_k^i = \frac{\sigma_1 + \sigma_2}{2} \quad (6.33)$$

$$\mathbf{P}_k^i = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{V}^T \quad (6.34)$$

##### 6.4.5 Soluzione

La Formula 6.32 dovrebbe suggerire che matrici di proiezione con associati pesi più elevati sono numericamente più affidabili, e quindi più adatte ad essere usate come sistema di riferimento per la soluzione del problema di analisi procustiana ortogonale. Questa richiede la scelta di una tri-colonna  $\mathbf{m}_k$ , e l'uso delle matrici di proiezione da essa specificate (Formula 6.34) come riferimento.

## CAPITOLO 6. IMPLEMENTAZIONE MATLAB DI AAM 2D+3D IN TEMPO REALE

Un modo per identificare un'insieme di matrici  $\mathbf{P}_{ref}^0, \dots, \mathbf{P}_{ref}^N$  da usare come riferimento, basato sulla massimizzazione del valore dei parametri di forma 3D, comporta il calcolo dei seguenti valori:

$$\begin{bmatrix} prod_0 \\ \vdots \\ prod_N \end{bmatrix} = \begin{bmatrix} |\prod_{i=0}^{\bar{m}} p_i^0| \\ \vdots \\ |\prod_{i=0}^{\bar{m}} p_i^N| \end{bmatrix} \quad (6.35)$$

e:

$$i_{max} = \underset{i}{argmax} (prod_i) \quad (6.36)$$

$$ref = \underset{j}{argmax} (|p_j^{i_{max}}|) \quad (6.37)$$

Quindi  $ref$  è l'indice della tri-colonna di  $\mathbf{M}$  che ha il massimo valore del parametro di forma 3D per la forma 2D associata alla massima produttoria (Formula 6.35). Questa è una buona scelta perché una produttoria elevata suggerisce che tra i parametri di forma 3D coinvolti ci siano pochi valori vicini allo zero.

Scelto il sistema di riferimento, si può procedere all'allineamento tra i sistemi di coordinate, applicando per ogni insieme di matrici  $\mathbf{P}_i^0, \dots, \mathbf{P}_i^N$  ( $i \neq ref$ ) l'Algoritmo 2. A seguito di questa operazione, tutte le matrici di proiezione si troveranno nello stesso sistema di riferimento, anche se non saranno necessariamente tutte uguali tra loro.

Essendo interessati a ricostruire la matrice  $\mathbf{M}$ , saranno necessarie solo  $N+1$  matrici di proiezione univoche, che si dovranno scegliere tra le  $K(N+1)$  stimate. Questa scelta può avvenire nuovamente basandosi sui valori dei parametri di forma 3D, in particolare scegliendo le matrici di proiezione con associato il parametro maggiore, in valore assoluto:

$$\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_N \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{argmax(p_i^0)}^0 \\ \mathbf{P}_{argmax(p_i^1)}^1, \\ \vdots \\ \mathbf{P}_{argmax(p_i^N)}^N \end{bmatrix} \quad (6.38)$$

Da queste proiezioni, la soluzione per  $\mathbf{M}$  si ottiene dalla Formula 4.12:

$$\mathbf{M} = \begin{bmatrix} p_0^0 \mathbf{P}_0 & p_1^0 \mathbf{P}_0 & \dots & p_{\bar{m}}^0 \mathbf{P}_0 \\ p_0^1 \mathbf{P}_1 & p_1^1 \mathbf{P}_1 & \dots & p_{\bar{m}}^1 \mathbf{P}_1 \\ \vdots & \vdots & \vdots & \vdots \\ p_0^N \mathbf{P}_N & p_1^N \mathbf{P}_N & \dots & p_{\bar{m}}^N \mathbf{P}_N \end{bmatrix} \quad (6.39)$$

---

**Algorithm 2** Analisi procustiana ortogonale tra le matrici di proiezione stimate, preceduta dalla stima del segno dei parametri di forma 3D.

---

```

// Stima dell'angolo tra le matrici
 $\theta = \arccos\left(\frac{\text{trace}\left(\mathbf{P}_{ref}^{i_{max}}(\mathbf{P}_i^{i_{max}})^T\right)}{2}\right);$ 
// Verifica se è il caso di cambiare il segno ad alcuni
// parametri di forma 3D
for (j =  $\bar{m} + 1$ ; j  $\leq N$ ; j++) do
     $\theta^+ = \arccos\left(\frac{\text{trace}\left(\mathbf{P}_{ref}^j(\mathbf{P}_i^j)^T\right)}{2}\right);$ 
     $\theta^- = \arccos\left(\frac{\text{trace}\left(-\mathbf{P}_{ref}^j(\mathbf{P}_i^j)^T\right)}{2}\right);$ 
    // Cambia di segno se la proiezione cambiata di segno
    // è più 'vicina' al riferimento
    if  $|\theta^+ - \theta| > |\theta^- - \theta|$  then
         $\mathbf{P}_i^j \leftarrow -\mathbf{P}_i^j;$ 
         $p_i^j \leftarrow -p_i^j;$ 
    endif
endfor
// Applica il metodo di analisi procustiana ortogonale
 $\mathbf{OP} = \begin{bmatrix} \mathbf{P}_i^{\bar{m}+1} \\ \vdots \\ \mathbf{P}_i^N \end{bmatrix}^T \begin{bmatrix} \mathbf{P}_{ref}^{\bar{m}+1} \\ \vdots \\ \mathbf{P}_{ref}^N \end{bmatrix};$ 
 $\mathbf{U}, \mathbf{D}, \mathbf{V}^T = svd(\mathbf{OP});$ 
 $\begin{bmatrix} \mathbf{P}_i^0 \\ \vdots \\ \mathbf{P}_i^N \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{P}_i^0 \\ \vdots \\ \mathbf{P}_i^N \end{bmatrix} \mathbf{U} \mathbf{V}^T;$ 

```

---

In generale, si è interessati alla soluzione in funzione dell'ordinamento delle righe originario, ed è a questo punto che si usano le informazioni conservate durante la fase di ordinamento (Sezione 6.4.1) per ripristinare l'ordinamento iniziale.

Rimangono da determinare le mode di forma 3D in  $\mathbf{B}$  che, se si applicasse la definizione, sarebbero da ricavare applicando la matrice correttiva invertita a  $\tilde{\mathbf{B}}$ . Ma a questo punto ha poco senso determinare  $\mathbf{G}$ , visto che già si conosce  $\mathbf{M}$ ; inoltre dalla Formula 4.15, risulta evidente che:

$$\mathbf{B} = \mathbf{M}^+ \tilde{\mathbf{W}} \quad (6.40)$$

La correttezza della ricostruzione è stimabile usando la seguente misura di errore relativo:

$$\frac{\|\tilde{\mathbf{W}} - \mathbf{M}\mathbf{B}\|_F}{\|\tilde{\mathbf{W}}\|_F} \quad (6.41)$$

dove  $\|\cdot\|_F$  è la norma di Frobenius. Si osserva comunque che questa stima non è molto affidabile, ed è sempre consigliabile valutare la qualità del modello 3D ottenuto effettuando alcuni fit dell'algoritmo 2D+3D su immagini con rappresentate varie pose e espressioni .

Se lo si desidera, è infine possibile ricostruire tutte le forme tridimensionali associate alle forme 2D applicando la Formula 6.17.

## 6.5 Fitting 2D+3D In Tempo Reale

Le estensioni 3D non comportano nessuna modifica alla procedura di training dell'AAM, la cui implementazione rimane quella discussa nella Sezione 6.2. La fase off-line dell'algoritmo è comunque modificata, e al training dell'AAM si va ad aggiungere il recupero delle mode di forma 3D, discusso nella sezione precedente.

Per quanto riguarda il fitting, la procedura ha a disposizione: l'AAM addestrato, la forma di partenza 2D, l'immagine su cui effettuare il fitting e le mode di forma 3D. In aggiunta a queste informazioni, si osserva che per l'inizializzazione sono richieste: la matrice di proiezione iniziale, la traslazione iniziale e la forma 3D iniziale; che sono informazioni difficili da fornire senza conoscenza a priori. Per questo, si descriverà un semplice metodo per determinare questi parametri senza bisogno di conoscenza a priori.

L'integrazione delle estensioni 3D nella procedura di fitting con composizione inversa non comporta particolari difficoltà, anche se nel nostro caso si sono incontrati alcuni ostacoli di cui si è ritenuto opportuno discutere le implicazioni.

### 6.5.1 Inizializzazione

Come per la procedura di fitting 2D, la forma bidimensionale di inizializzazione (che fornisce il valore iniziale dei parametri  $\bar{\mathbf{p}}$  e  $\bar{\mathbf{q}}$ ) rappresenta un parametro che non è stimato dalla procedura di fitting (cosa ciò comporta è già stato discusso nella Sezione 6.3.1).

Per quanto riguarda l'inizializzazione dei valori di  $\bar{\mathbf{p}}$ ,  $\bar{\mathbf{P}}$ ,  $\bar{o}_u$  e  $\bar{o}_v$ , sarebbe preferibile evitare di richiedere all'utente anche la specifica di questi parametri. Si ha già accennato, nella Sezione 4.3.4, a come sia possibile usare l'analisi procastiana a questo scopo, determinando la combinazione di trasformazione tridimensionale e forma 3D che meglio approssima la forma di inizializzazione bidimensionale  $\bar{\mathbf{s}} = \bar{\mathbf{N}} \circ \bar{\mathbf{W}}(\bar{\mathbf{s}}_0; \bar{\mathbf{q}}, \bar{\mathbf{p}})$ . In particolare, si è scelto di usare per l'inizializzazione la moda di forma 3D associata all'errore quadratico minimo:

$$\underset{i}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{s} \\ \mathbf{0} \end{bmatrix} - \epsilon \mathbf{R} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \bar{\mathbf{s}}_i + \begin{bmatrix} t_u \\ t_v \end{bmatrix} \right\|^2 \quad (6.42)$$

con  $\epsilon$ ,  $\mathbf{R}$  e  $[t_u \ t_v]^T$  stimati attraverso l'analisi procustiana e  $\|\cdot\|^2$  che indica la somma del quadrato dei valori.

Si pone quindi la forma 3D iniziale  $\bar{\mathbf{s}}$  pari alla moda di forma tridimensionale  $\bar{\mathbf{s}}_i$  associata all'errore minimo, e si vede facilmente che la matrice di proiezione e la traslazione da usare per l'inizializzazione si determinano a partire dalla trasformazione affine nella Formula 6.42:

$$\mathbf{P} = \epsilon \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{R} \quad (6.43)$$

$$\begin{bmatrix} o_u \\ o_v \end{bmatrix} = \begin{bmatrix} t_u \\ t_v \end{bmatrix} \quad (6.44)$$

### 6.5.2 Stima delle Jacobiane

Le Jacobiane necessarie alle estensioni 3D dell'algoritmo di fitting non sono costanti, ed essendo necessario ricalcolarle ad ogni iterazione, è di fondamentale importanza che le matrici siano calcolate in modo efficiente. Diventa così essenziale avere a disposizione un'implementazione ottimizzata della composizione di warp, procedura che è usata  $(m + 4)$  volte per ogni iterazione.

Inoltre, la scelta della struttura dati usata per memorizzare le Jacobiane può fare la differenza, soprattutto in MATLAB. A questo scopo è soprattutto importante minimizzare il tempo necessario a memorizzare ogni singola Jacobiana, assicurandosi che componenti diverse della stessa Jacobiana occupino posizioni contigue in memoria, usando ad esempio una matrice tridimensionale:

$$J(t, i, var) = \frac{\partial F_{t_i}}{\partial var} \quad (6.45)$$

Così, nel caso di un linguaggio vettoriale come MATLAB, è possibile determinare e memorizzare tutte le componenti di ogni Jacobiana in modo molto semplice e efficiente:

$$J(*, *, j) = \frac{\partial F}{\partial q_j} \mathbf{J}_q \quad (6.46)$$

$$J(*, *, 4 + j) = \frac{\partial F}{\partial p_j} \mathbf{J}_p \quad (6.47)$$

$$J(*, *, 4 + m + j) = \frac{\partial F}{\partial \bar{p}_j} \quad (6.48)$$

$$J(*, *, 4 + m + \bar{m} + 1) = \frac{\partial F}{\partial \theta_x} \quad (6.49)$$

$$\vdots \quad (6.50)$$

dove  $J(*, *, var)$  è la matrice bidimensionale di indice  $var$ . Si noti che l'ordine delle prime  $m + 4$  variabili dev'essere coerente con quello usato per il training dell'AAM, altrimenti il calcolo

dell'Hessiana  $\mathbf{H}_{3D}$  risulterà errato.

Non c'è molto altro da dire a riguardo del calcolo delle Jacobiane, in quanto si basa sull'applicazione di semplici operazioni vettoriali, o dei principi di composizione inversa; tutte cose che sono già state approfondite nella Sezione 4.3.2.

Ci si sofferma piuttosto sulla Jacobiana  $\frac{\partial F}{\partial \sigma}$ : questa non solo non è necessaria, ma rende anche singolare la matrice  $\mathbf{H}_{3D}$ , in quanto il modello di forma 3D che si sta considerando in questa implementazione è quello descritto nella Formula 6.17, in cui non c'è modo di distinguere tra un'operazione di scala applicata alla forma e una applicata ai parametri di forma 3D. Il parametro di scala della matrice di proiezione è quindi fonte di ambiguità (a meno di imporre dei limiti nella variabilità dei parametri  $\bar{\mathbf{p}}$ ), e deve essere considerato una costante se si vuole evitare di rendere l'Hessiana singolare.

Infine, si osserva che le Jacobiane  $\frac{\partial F}{\partial o_u}$  e  $\frac{\partial F}{\partial o_v}$  sono costanti e non ha senso ricalcolarle ad ogni iterazione.

### 6.5.3 Calcolo dei Parametri Incrementali

Il calcolo dell'Hessiana  $\mathbf{H}_{3D}$  (Formula 4.44) non è facilmente ottimizzabile, ma grazie al ridotto numero di dimensioni delle matrici coinvolte, il costo computazionale di questa parte è trascurabile rispetto a quello del calcolo delle Jacobiane.

Mentre per quanto riguarda il calcolo dei parametri incrementali di un'iterazione, è sufficiente applicare la teoria descritta nella Sezione 4.3.2: determinati i valori  $\mathbf{K}_{2D} = [\mathbf{K}_p \mathbf{K}_q \mathbf{0} \mathbf{0} \mathbf{0} \mathbf{0}]^T$  a partire dalla versione 2D dell'Hessiana inversa (Formula 3.34), si procede a moltiplicare la versione 3D dell'Hessiana inversa per i parametri incrementali di steepest descent:

$$\begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{p} \\ \Delta \bar{\mathbf{p}} \\ \Delta \theta_x \\ \Delta \theta_y \\ \Delta \theta_z \\ \Delta o_u \\ \Delta o_v \end{bmatrix} = -\mathbf{H}_{3D}^{-1} \begin{bmatrix} \mathbf{K}_q \\ \mathbf{K}_p \\ \mathbf{0} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + K \sum_{t=u,v} \sum_{i=1}^n \mathbf{S} \mathbf{D}_{F_{t_i}}^T F_{t_i}(\mathbf{p}; \mathbf{q}; \bar{\mathbf{p}}; \mathbf{P}; o_u; o_v) \quad (6.51)$$

con il calcolo dei prodotti matriciali  $\mathbf{S} \mathbf{D}_{F_{t_i}}^T F_{t_i}(\dots)$  che diventa piuttosto contorto, nel caso si sia usata una matrice tridimensionale per memorizzare le Jacobiane, in quanto:

$$\mathbf{S} \mathbf{D}_{F_{t_i}} = \begin{bmatrix} J(t,i,1) & \dots & J(t,i,m+4+\bar{m}+5) \end{bmatrix} \quad (6.52)$$

La costruzione di questi vettori genera indubbiamente problemi di coerenza cache, con conseguente perdita di efficienza, per non parlare delle numerose copie di dati richieste. Apparentemente, questo è il prezzo da pagare se si intende massimizzare l'efficienza del calcolo delle Jacobiane, in quanto non sembra sia possibile trovare una rappresentazione che sia efficiente in entrambe le situazioni.

Per quanto riguarda l'aggiornamento della soluzione 2D, il procedimento è identico a quello dell'algoritmo puramente 2D, che è stato discusso ampiamente nella Sezione 6.3.3, mentre l'aggiornamento della forma 3D è, intuitivamente:

$$\bar{\mathbf{s}} \leftarrow \bar{\mathbf{s}} + \sum_{i=0}^{\bar{m}} \Delta \bar{p}_i \quad (6.53)$$

L'aggiornamento degli altri parametri è già stato discusso nella Sezione 4.3.3, ma riassumendo brevemente si avrà:

$$o_u \leftarrow o_u + \Delta o_u \quad (6.54)$$

$$o_v \leftarrow o_v + \Delta o_v \quad (6.55)$$

e:

$$\mathbf{P} \leftarrow \mathbf{P} \begin{bmatrix} 1 & -\Delta\theta_z & \Delta\theta_y \\ \Delta\theta_z & 1 & -\Delta\theta_x \\ -\Delta\theta_y & \Delta\theta_x & 1 \end{bmatrix} \quad (6.56)$$

operazioni seguite dall'ortogonalizzazione di  $\mathbf{P}$ .

#### 6.5.4 Integrazione in Modo Efficace dei Vincoli di Coerenza 3D

Nella sezione precedente sono stati omessi tutti i dettagli riguardanti il peso  $K$  dei vincoli di coerenza 3D. Questo perché, nel nostro caso, non è stato possibile individuare un valore che funzionasse in generale.

Baker ha specificato che un valore di  $K$  buono è grande, e tale da rendere il termine di errore 3D (Formula 4.35) un centinaio di volte maggiore dell'errore immagine proiettato. Si vede però che in pratica, nelle prime iterazioni dell'algoritmo, l'errore 3D può essere dello stesso ordine di grandezza o minore di quello immagine, quindi un valore di  $K$  che rende il rapporto tra errore 3D e errore immagine vicino a 100 non è grande.

E in ogni caso, a prescindere dal valore di  $K$ , si è notato che l'algoritmo tende a far convergere la soluzione 2D verso la proiezione di quella 3D attuale, ignorando del tutto o quasi il fitting 2D. E al crescere del valore di  $K$ , l'effetto si è mostrato sempre più accentuato.

Per questo motivo, si è scelto di usare un approccio di separazione nell'aggiornamento dei parametri analogo a quello applicato all'algoritmo di fitting 2D (Sezione 6.3.4), allo scopo di conservare l'accuratezza della soluzione bidimensionale. In pratica, la nostra soluzione ignora l'aggiornamento

dei parametri 3D durante le prime 10 iterazioni dell'algoritmo, e lavora esclusivamente sui parametri  $\mathbf{p}$  e  $\mathbf{q}$  in modo assolutamente identico all'algoritmo di fitting 2D. Mentre nelle ultime iterazioni ignora i parametri 2D e lavora esclusivamente sui parametri 3D.

Grazie a questo stratagemma è stato possibile minimizzare efficacemente entrambi i termini di errore, senza compromettere l'accuratezza del fitting bidimensionale, e con il vantaggio aggiuntivo di eliminare la necessità di determinare un valore adeguato per il parametro  $K$ .

## 6.6 Risultati Sperimentali

Qualità, correttezza e efficienza dell'implementazione sono state valutate in più occasioni, usando diversi dataset.

In seguito verranno esposti i risultati di maggior rilevanza ottenuti, cercando di adottare misure di errore compatibili con quelle già usate per valutare le prestazioni dell'AAM-API (Sezione 5.2.4) e con quelle usate da Matthews et al. nel loro articolo [25]. In quest'ultimo caso, si sottolinea la scarsa rilevanza di una comparazione dei risultati, in quanto ottenuti con dataset e tecniche diverse.

### 6.6.1 Fitting 2D

#### 6.6.1.1 Person-dependent

Le prestazioni person-dependent sono state valutate sullo stesso dataset usato dal caso di studio descritto nella Sezione 5.2.4.1, sempre con una metodologia leave-one-out.

L'addestramento dei 26 AAM richiesti, a partire da 25 immagini annotate manualmente, ha richiesto circa 45 secondi per ciascuno, e si è osservato che in media sono state necessarie 16 mode per il modello di forma e 21 mode per il modello di apparenza (mediamente memorizzato con una risoluzione di  $309 \times 338$  pixel, in formato RGB).

A differenza del caso di studio della Sezione 5.2.4.1, sono state valutate le prestazioni anche a partire da una maggiore varietà forme iniziali, progressivamente più lontane dal ground-truth per quanto riguarda posizione, rotazione e scala. L'entità della variazione applicata al ground-truth è stata calcolata a partire da un fattore  $k$  compreso tra 1 e 10, a parametrizzare una trasformazione affine:

- traslazione: nulla o di  $3k$  pixel in due direzioni casuali
- rotazione: nulla o di  $\pm k\pi/80$  radianti rispetto al baricentro
- scala: assente o del  $\pm 2k\%$

## 6.6. RISULTATI Sperimentali

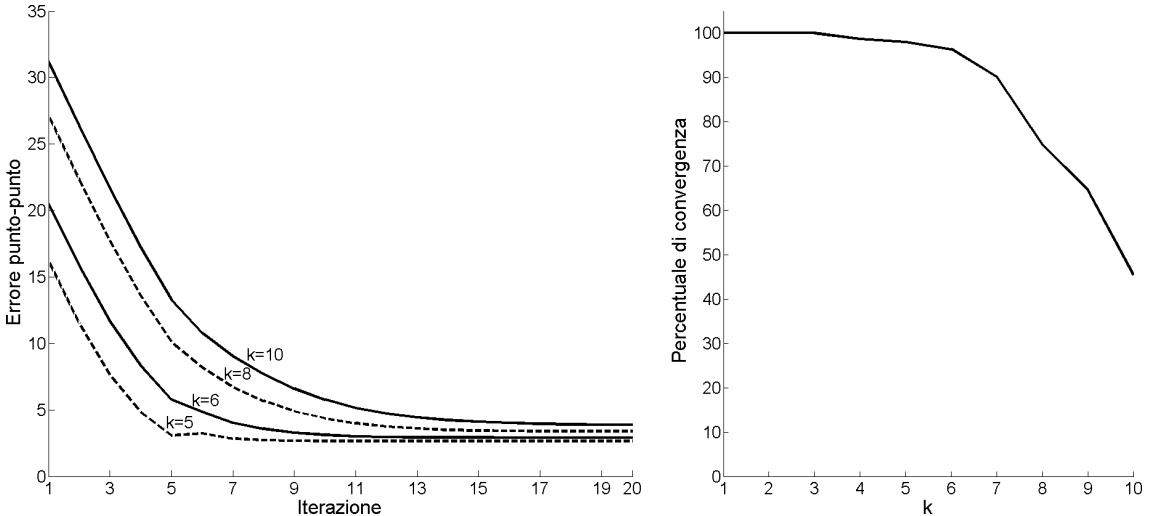


Figura 6.6. A sinistra: velocità di convergenza per diversi valori di  $k$ . A destra: percentuale di convergenza del fitting person-dependent in funzione dell'indice  $k$  (che esprime l'entità dello spostamento rispetto al ground-truth). Si è dichiarato un esperimento convergente quando l'errore punto-punto  $D_{pt.pt}$  prodotto dal fitting è stato inferiore ai 5.5 pixel.

Per ogni esperimento leave-one-out, sono state applicate tutte le trasformazioni affini definite dalle possibili combinazioni di questi parametri (ad eccezione di quella identica) al ground-truth per ottenere le forme con cui inizializzare 20 iterazioni dell'algoritmo di fitting. È stata quindi valutata la velocità di convergenza e la percentuale di test che hanno portato a convergenza del fitting, in funzione del valore di  $k$ . I risultati di questi test sono illustrati nella Figura 6.6.

I dettagli sull'errore punto-punto in funzione di  $k$  sono visibili nella Tabella 6.3. Si osserva che i valori di  $D_{pt.pt}$  medi per  $k$  piccoli sono generalmente bassi, ma a differenza degli esperimenti di Matthews et al. [25], non tendono mai a zero e sembrano assestarsi sempre su valori compresi tra 1.2 e 2.5. Questo è probabilmente dovuto al rumore nelle annotazioni usate per il training, che sono state specificate manualmente (si veda la Sezione 5.2.4.1), anziché essere prodotte da un AAM.

$D_{pt.pt.}$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$	10%
Medio	2.425	2.446	2.482	2.54	2.665	3.008	3.971	6.392	8.02	11.65	3.552
Min	1.222	1.219	1.209	1.12	1.204	1.205	1.224	1.241	1.211	1.246	1.215
Max	3.872	3.918	4.005	4.36	8.118	9.149	59.06	81.75	70.39	88.56	12.627

Tabella 6.3. Errore punto-punto medio, minimo e massimo in funzione dell'indice di distanza dal ground-truth  $k$ . L'ultima colonna rappresenta i risultati ottenuti inizializzando solo con una traslazione del 10% rispetto alla dimensione della forma nelle direzioni degli assi X e Y, sia positiva che negativa.

L'ultima colonna della Tabella 6.3 illustra i risultati ottenuti usando la stessa strategia di inizializzazione usata nella verifica delle prestazioni person-dependent dell'implementazione di Stegmann. L'errore medio e minimo sono molto simili, mentre l'errore massimo segnala una maggiore variabilità nei risultati prodotti dalla nostra implementazione.

Durante le misurazioni è stata monitorata anche l'efficienza di fitting, con la procedura che ha richiesto in media<sup>4</sup> 603 millisecondi per effettuare 20 iterazioni di fitting. Confrontati con i 345 millisecondi prodotti sullo stesso dataset da parte dell'AAM-API (Sezione 5.2.5), non sembra un grande risultato, ma bisogna considerare il numero di iterazioni effettuate in quel caso (compreso tra 7 e 10), e che l'AAM-API è implementata in C++.

#### 6.6.1.2 Person-independent

Allo stesso modo del caso di studio descritto nella Sezione 5.2.4.2, il training set è stato costruito a partire dall'intero dataset IMM (Sezione 5.2.1), escludendo da questo 5 soggetti che sono stati usati per valutare le prestazioni di fitting. A partire dalle 210 immagini e forme di training, l'addestramento del modello ha prodotto 38 mode di forma e 94 mode di apparenza (con risoluzione  $227 \times 205$  e in formato RGB), in un tempo di poco superiore ai 2 minuti.

La Tabella 6.4 mostra i risultati prodotti da 20 iterazioni dell'algoritmo di fitting, usando la stessa modalità di inizializzazione usata dall'AAM-API in configurazione pseudo-automatica, e usando lo smorzamento dei parametri di forma descritto nella Sezione 6.3.4. Si osserva che, rispetto ai risultati prodotti dall'AAM-API, l'errore punto-punto medio è decisamente superiore con un errore punto-punto che si è attestato su valori inferiori ai 4.5 pixel (errore al di sopra del quale si può dire che non c'è stata convergenza) solo nel 25% dei casi valutati.

Per quanto riguarda il tempo di fitting, la minore risoluzione del modello ha fatto sì che il fitting abbia richiesto, in media, quasi la metà del tempo rispetto al caso person-dependent, più precisamente 363.2 millisecondi. Un tempo di fitting simile a quello osservato con l'AAM-API quindi, anche se in quel caso il numero di iterazioni considerate era in media 12.

$D_{pt.pt.}$		
Medio	Min	Max
14.466	2.994	69.988

Tabella 6.4. Errore punto-punto medio, minimo e massimo prodotti dalla nostra implementazione del fitting 2D in condizioni di training set e inizializzazione identiche a quelle del caso di studio della Sezione 5.2.4.2.

<sup>4</sup>Ottenuti con un processore Intel dual-core a 2.4GHz.

### 6.6.1.3 Complessità e Fitting in Tempo Reale

Precedentemente, si ha già accennato a questioni relative l'efficienza della nostra implementazione. Si cercherà ora di definire in modo più o meno formale la complessità della procedura di fitting con composizione inversa, verificando empiricamente che questa è corretta.

Dato un modello di apparenza a risoluzione  $w \times h$  con  $nc$  canali cromatici, e un modello di forma con  $m$  mode di forma, si osserva facilmente che la complessità dell'algoritmo di fitting è definita dalle seguenti componenti:

- Warp dell'immagine test:  $O(w \cdot h \cdot nc)$
- Calcolo dell'immagine di errore:  $O(w \cdot h \cdot nc)$
- Calcolo dei parametri incrementali moltiplicando una matrice di dimensione  $(m + 4) \times (w \cdot h \cdot nc)$  per l'immagine di errore vettorizzata:  $O((m + 4)(w \cdot h \cdot nc)) = O(m \cdot w \cdot h \cdot nc)$
- Composizione del warp su una mesh con  $nt$  triangoli:  $O(nt)$

Le altre operazioni effettuate dalla procedura di fitting hanno costo costante e trascurabile, come sarà dimostrato a breve.

La teoria della complessità specifica che diverse componenti che caratterizzano la complessità di un algoritmo possono essere raccolte in un'unica misura solo se le costanti moltiplicative considerate nei diversi casi hanno ordine di grandezza simile. In questo, caso le costanti non sono necessariamente simili, in quanto una parte del codice è eseguita in ambiente di scripting MATLAB, mentre altre porzioni di codice rappresentano operazioni implementate in modo ottimizzato nei toolbox di MATLAB, oppure sono MEX-files in C/C++. Nonostante questo, è comunque difficile commettere errori, in quanto si possono distinguere solo due tipologie di costi computazionali: quelli che dipendono da  $(w \cdot h \cdot nc)$  e quello che dipende da  $nt$ .

Per questo, si tratta solo di capire qual'è l'equilibrio tra le due tipologie di costi all'interno della definizione di complessità:

$$O(a \cdot m \cdot w \cdot h \cdot nc + b \cdot nt) \quad (6.57)$$

Si ha che in tutti i casi pratici di nostro interesse  $nt \ll (m \cdot w \cdot h \cdot nc)$ , quindi a meno che la costante  $b$  associata alla composizione sia di diversi ordini di grandezza maggiore di  $a$ , la complessità generale dell'algoritmo dovrebbe essere dominata dalle operazioni sull'immagine. La Figura 6.7, che mostra l'andamento del tempo di fitting in funzione di  $(m \cdot w \cdot h \cdot nc)$  da noi misurato<sup>5</sup>, chiarisce immediatamente ogni dubbio, confermando quello che intuitivamente era abbastanza prevedibile.

---

<sup>5</sup>Ottenuti con un processore Intel dual-core a 2.4GHz.

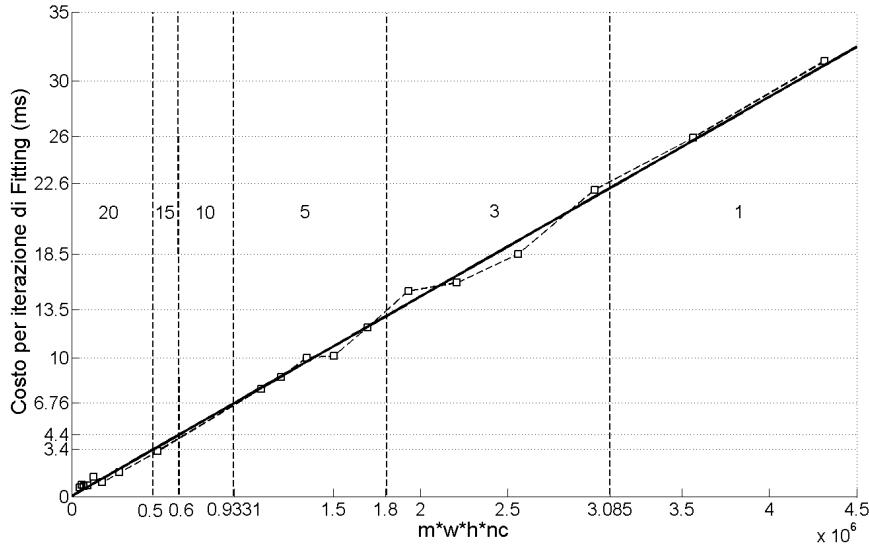


Figura 6.7. Tempo di fitting in funzione di  $m \cdot w \cdot h \cdot nc$ , il cui andamento sostanzialmente lineare conferma la nostra misura di complessità. Sono rappresentate inoltre le soglie entro le quali è possibile il fitting in tempo reale (a 15 fotogrammi al secondo o superiori), a seconda di quante iterazioni si intende effettuare. Ovviamente questi dati sono indicativi, con CPU più veloci si avrà che l'inclinazione della retta sarà più bassa, mentre con CPU più lente sarà più alta e le soglie di tempo reale cambieranno di conseguenza.

La complessità totale è quindi lineare rispetto al prodotto tra dimensione delle apparenze e numero di mode di forma:

$$O(m \cdot w \cdot h \cdot nc) \quad (6.58)$$

La Figura 6.7 illustra inoltre quella che si può definire “soglia di tempo reale”, ovvero il limite sul valore di  $(m \cdot w \cdot h \cdot nc)$  che non è possibile superare all'atto della creazione del modello se si desidera ottenere prestazioni di fitting in tempo reale, a seconda del numero di iterazioni dell'algoritmo di fitting. A questo riguardo, essendo il termine “tempo reale” abbastanza ambiguo, nel nostro caso si considera una prestazione di riconoscimento di 15 fotogrammi al secondo o superiore come di tempo reale.

Osservando la Figura 6.6, si nota che generalmente già dopo la decima iterazione l'algoritmo di fitting produce una soluzione molto vicina a quella ottima, e soprattutto nel contesto di un'applicazione a tempo reale, il miglioramento prodotto da ulteriori iterazioni potrebbe non valere il sacrificio in termini di prestazioni. Nel nostro caso, è perciò possibile eseguire il fitting con 10 iterazioni in tempo reale a patto che valga:  $(m \cdot w \cdot h \cdot nc) \leq 933100$ . Se si considera un modello di forma tipico con 20 mode di forma, si avrà che le apparenze dovranno avere al più 15551 pixel

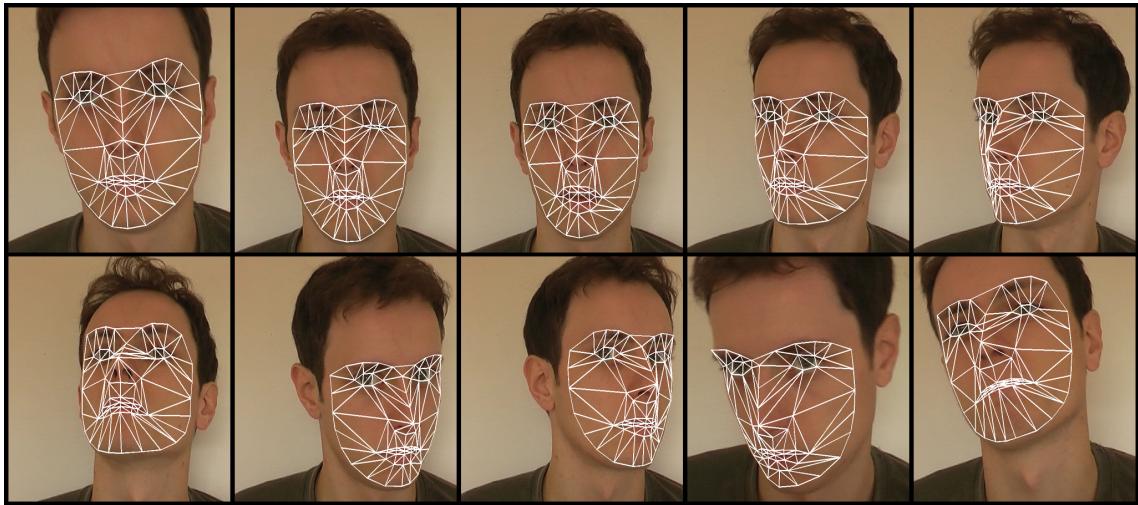


Figura 6.8. Selezione di fotogrammi presi dal video, con le relative annotazioni prodotte dal fitting.

se in formato RGB o 46655 pixels se in scala di grigi, ovvero risoluzioni dell'ordine dei  $124 \times 124$  o  $216 \times 216$  pixel, rispettivamente.

Considerando che la CPU usata per le misurazioni non è più molto recente, e che una moderna CPU può attualmente eseguire operazioni in virgola mobile fino a 2.8 volte più velocemente<sup>6</sup>, e considerando che la complessità è quadratica rispetto alla risoluzione delle apparenze, si stima che alle condizioni sopra elencate, il limite attuale di risoluzione per le apparenze sia intorno ai  $207 \times 207$  pixel, se in formato RGB, e  $361 \times 361$  pixel, se in scala di grigi.

### 6.6.2 Ricostruzione della Forma dal Movimento

I dati necessari all'algoritmo di ricostruzione della forma dal movimento sono stati ottenuti applicando la procedura di fitting 2D a un video della durata di 2 minuti e 20 secondi, caratterizzato da numerose variazioni di posa e espressione. In questo modo, si è cercato di massimizzare la quantità di informazioni a disposizione dell'algoritmo di ricostruzione, con la speranza di ottenere dei buoni risultati. Alcuni fotogrammi del video usato sono illustrati nella Figura 6.8.

Il fitting del video ha comportato alcune difficoltà, specificamente dovute alla non convergenza della procedura se inizializzata esclusivamente tramite coerenza temporale, ovvero usando il risultato del fotogramma precedente. Queste problematiche sono state risolte introducendo dei “checkpoint”, punti del video in corrispondenza dei quali il fitting è stato ri-inizializzato.

<sup>6</sup>Rapporto tra i MFLOPS dichiarati, considerando un i7-965 e il Core2Duo E6600 da noi usato. Fonte: <http://www.intel.com/support/processors/sb/cs-023143.htm>

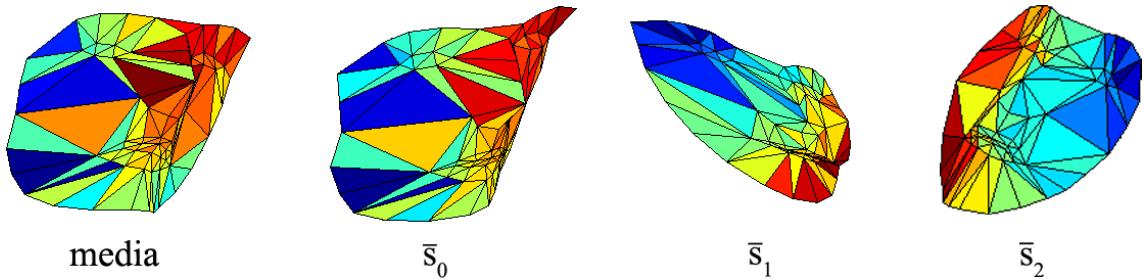


Figura 6.9. Forma 3D media e mode di forma del modello 3D ricavate dalla sequenza video (il modello è quello definito dalla Formula 6.17, che *non* include la forma media). Si noti la scarsa quantità di informazione tridimensionale codificata, dovuta alla necessità da parte dell'algoritmo di usare una selezione delle forme 3D ricostruite come basi del modello. NOTA: L'orientamento delle forme 3D in questa illustrazione è puramente arbitrario, e ha il solo scopo di enfatizzare la tridimensionalità (o non tridimensionalità) delle forme.

I primi test di ricostruzione su dati parziali hanno fornito informazioni molto utili, e hanno soprattutto evitato lavoro di espansione del training set dell'AAM che sarebbe comunque stato sprecato (o quasi).

La prima osservazione deriva dal rango della matrice delle misure registrata: questo ha suggerito che le mode di forma 3D ricostruibili effettivamente (Formula 6.16) erano 3 o al massimo 4, a prescindere da eventuali incrementi nella quantità di dati forniti all'algoritmo di ricostruzione.

In secondo luogo, è stato evidente come l'assunzione che  $K$  forme 3D prodotte dalla ricostruzione possano rappresentare una base per il modello di forma 3D rappresenti un serio limite alla quantità e qualità di variazione di forma rappresentabile.

Infatti, potendo scegliere solo tra 3 o 4 forme da usare come basi per il modello 3D, si osserva che a seconda della scelta fatta dall'Algoritmo 1 si ha: o una ristrettissima variabilità nel modello 3D, a causa di forme troppo simili o rappresentanti deformazioni poco significative o rare; o un accoppiamento estremo di alcune deformazioni che non sono in genere in relazione tra loro, rendendo quasi inutili alcune basi 3D.

Per questi motivi, l'unico modo per determinare una buona ricostruzione consiste nell'eseguire ripetutamente l'algoritmo di ricostruzione della forma dal movimento, nella speranza che l'algoritmo randomizzato faccia una buona scelta, o ripiegare su una ricerca esaustiva, che comunque non è garantito fornisca la soluzione migliore. In generale, per i motivi appena descritti, la variabilità di forma del modello 3D ricostruito molto difficilmente potrà rappresentare accuratamente sottili cambiamenti di espressione che sono facilmente catturabili da un modello 2D.

A seguito di queste osservazioni, si è scelto di concentrare lo sforzo di ricostruzione su 409 fotogrammi (16 secondi circa) caratterizzati da una buona variabilità nella posa, escludendo le variazioni di espressione più complesse degli altri fotogrammi che sarebbero comunque state ignorate o avrebbero rischiato di compromettere il modello, introducendo accoppiamenti di deformazioni e riducendo le possibili variazioni di posa.

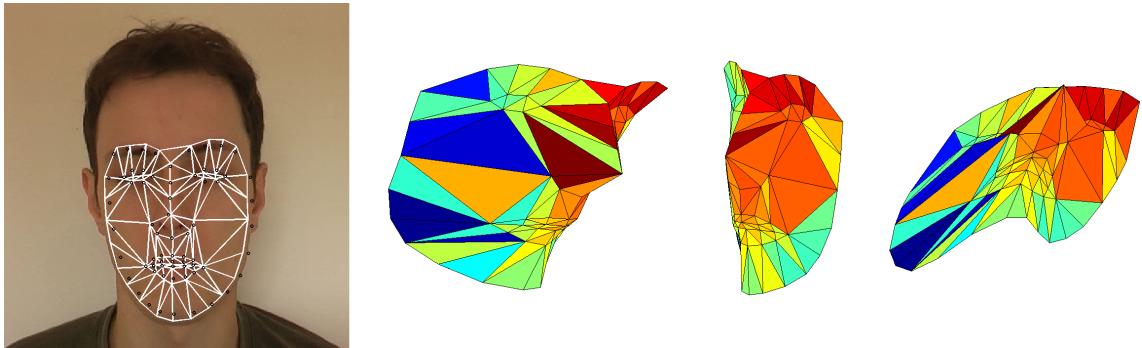


Figura 6.10. Fitting 2D+3D su un fotogramma del video usato per la ricostruzione della forma dal movimento: i cerchietti neri indicano la proiezione della forma 3D ricostruita. Si nota subito che il modello 3D non è riuscito a ricostruire gli occhi chiusi; inoltre, si osservano imprecisioni nella zona della bocca e asimmetria nella forma 3D generata

Le 3 mode ricostruite sono illustrate nella Figura 6.9. A questa ricostruzione è stato associato un errore relativo (Formula 6.41) di circa 0.4, un errore non trascurabile, che sottolinea la difficoltà da parte dell'algoritmo nel codificare opportunamente tutte le variazioni 2D usando una rappresentazione 3D soggetta a così tante limitazioni.

### 6.6.3 Fitting 2D + 3D

Per quanto riguarda la valutazione delle prestazioni, poco si può dire sulla qualità del fitting 3D, e non essendoci metro di paragone affidabile, ci si è dovuti limitare a una valutazione qualitativa.

In particolare, è stato problematico ottenere buone prestazioni di fitting 3D a causa delle difficoltà nel ricavare delle mode di forma 3D valide, per i motivi descritti in precedenza. Infatti, nonostante l'algoritmo abbia dimostrato la capacità di convergere anche nella parte di fitting tridimensionale, questo non è in generale riuscito ad adattarsi a deformazioni della bocca o degli occhi, appunto perché queste deformazioni non sono state incluse nelle mode di forma 3D dall'algoritmo di ricostruzione della forma del movimento.

Se si considera invece la porzione di fitting 2D, si osserva che prevedibilmente, l'accuratezza non cambia rispetto all'algoritmo puramente bidimensionale, a patto di svolgere indipendentemente il fitting (come descritto nella Sezione 6.5.4).

La Figura 6.10 illustra in modo emblematico tutte le problematiche associate al fitting 3D appena descritte.



# Capitolo 7

## Conclusioni

Gli Active Appearance Models hanno dimostrato di essere degli ottimi strumenti per il riconoscimento di oggetti deformabili, e nonostante questa trattazione si sia limitata esclusivamente ai volti umani, non c'è motivo di credere che i metodi descritti non sarebbero altrettanto efficaci se applicati ad altre tipologie di oggetti deformabili.

In particolare, i casi di studio effettuati sull'implementazione di Stegmann hanno mostrato l'eccellente accuratezza e capacità di generalizzazione degli AAM tradizionali, mentre non è stato possibile dire lo stesso degli AAM con composizione inversa, che si sono dimostrati affidabili e robusti solo con certi dataset, mentre con altri hanno mostrato prestazioni altalenanti o insufficienti. Evidentemente, come puntualizzato anche da Amberg et al. [2], la maggiore efficienza computazionale degli AAM con composizione inversa è ottenuta a spese di una minore capacità di convergenza (e una conseguente maggiore sensibilità al rumore nei dati di training).

In generale, uno dei maggiori punti deboli degli AAM è dato dalla fase di addestramento, che richiede un lavoro di annotazione manuale lungo e tedioso, e in cui lievi imprecisioni possono portare a imperfezioni nei modelli di forma e apparenza. Fenomeno che, come si è visto, ha un impatto decisamente significativo negli AAM con composizione inversa. E anche se sono stati proposti diversi metodi per l'addestramento automatizzato di AAM [31, 8, 18, 5], questi sono ancora lontani dal produrre risultati paragonabili a quelli ottenuti usando tecniche tradizionali di annotazione. Quindi sembra che l'annotazione supervisionata o completamente manuale resterà il metodo preferenziale per addestrare gli AAM, almeno finché i metodi automatizzati non riusciranno a produrre risultati di qualità comparabile.

Una strada alternativa al training automatizzato potrebbe essere lo sviluppo di modelli *general-purpose*, costruiti con dataset di grande dimensione e di alta qualità, eliminando in larga parte la necessità di addestramento per molte applicazioni. Se si considera ad esempio il caso dei volti umani, che è quello applicativo di maggiore interesse, in molti casi non si può pretendere di poter

## CAPITOLO 7. CONCLUSIONI

addestrate un AAM per ogni soggetto specifico da considerare. Superare questo limite e riuscire ad addestrare degli AAM che, seppure in condizioni abbastanza controllate, possano riconoscere in modo affidabile soggetti non presenti nel training set sarebbe di estrema utilità in molti campi applicativi, dalla medicina, all'affective computing, ai sistemi di sicurezza.

I metodi di fitting considerati non sono però stati in grado di fornire garanzie nel riconoscimento di soggetti non presenti nel training set, soprattutto di fronte a cambiamenti nelle condizioni di illuminazione, rumore di fondo e/o variazioni peculiari (occhiali, barba, piercing...) rispetto al training set. Per questo motivo, è di fondamentale importanza la ricerca nel ridurre l'influenza delle variazioni dovute all'illuminazione [35, 19] o nel migliorare in generale le capacità di generalizzazione degli AAM, tipicamente usando modelli di forma e apparenza più sofisticati di quelli lineari [39].

Altro ostacolo alle possibili applicazioni pratiche degli AAM è la difficoltà nel gestire occlusioni di vario tipo, problema che è tuttora oggetto di ricerca, anche se i risultati finora ottenuti sono di tutto rispetto [16, 44, 23]. Parte integrante nella gestione delle occlusioni, come illustrato dai nostri casi di studio, è costituita dallo sviluppo di metriche robuste per la valutazione della qualità del riconoscimento [36].

Anche se l'analisi non è stata molto approfondita, si è dimostrato che la ricostruzione della forma dal movimento sembra essere ancora un problema difficile da risolvere per gli oggetti deformabili. In particolare, anche se i vincoli di base permettono di rimuovere l'ambiguità dalla ricostruzione negli algoritmi basati su fattorizzazione della matrice delle misure, la scelta di definire il modello di forma 3D attraverso la scelta di soltanto 3 o 4 forme ricostruite rende molto difficile la creazione di modelli di buona qualità. Sarebbero quindi desiderabili algoritmi di ricostruzione della forma dal movimento meno vincolati al rango della matrice delle misure, e che siano meno propensi ad accoppiare deformazioni fondamentalmente correlate nelle mode di forma 3D.

Le difficoltà nella creazione di un buon modello di forma 3D hanno reso difficile la verifica dell'efficacia dell'algoritmo di fitting 2D+3D, anche se questo non ha mostrato difficoltà a convergere verso una soluzione sensata (a patto di usare particolari accorgimenti). In ogni caso, non si è considerata prioritaria la realizzazione di un'implementazione real-time dell'algoritmo, anche se i risultati del profiling hanno suggerito che, con le dovute ottimizzazioni, le estensioni 3D comporterebbero un incremento nel tempo di calcolo di circa il 20% rispetto al fitting 2D, che è un valore simile a quello osservato da Xiao et al. [42].

In conclusione, vista la scarsa disponibilità di implementazioni di Active Appearance Models sufficientemente flessibili ed efficienti da essere usate in diversi contesti applicativi, la nostra implementazione potrebbe sicuramente dimostrarsi di utilità per chi lavora nel campo del riconoscimento di oggetti deformabili. E da quanto ci risulta, è anche l'unica implementazione predisposta al fitting contemporaneo di forma 2D e 3D. Inoltre, l'intrinseca portabilità dell'implementazione la rende indubbiamente interessante per chi lavora in ambienti diversi da Microsoft Windows e non può quindi fare uso dell'AAM-API, specialmente se si considera l'elevata efficienza di fitting in confronto all'implementazione di Brian Amberg.

# Bibliografia

- [1] Geoworld, October 2005.
- [2] B. Amberg, A. Blake, and T. Vetter. On compositional image alignment, with an application to active appearance models. *CVPR'09, IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.
- [3] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090 – 1097, 2001.
- [4] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, 2004.
- [5] Simon Baker, Iain Matthews, and Jeff Schneider. Automatic construction of active appearance models as an image coding problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):1380 – 1384, 2004.
- [6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.
- [7] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.
- [8] Matthew Brand. Morphable 3D models from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 456–463, 2001.
- [9] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 690–696, 2000.
- [10] Raphael Cendrillon and Brian C. Lovell. Real-time face recognition using eigenfaces. In *Proceedings of the SPIE International Conference on Visual Communications and Image Processing*, 2000.
- [11] T. F. Cootes and C. J. Taylor. Active shape models - ‘smart snakes’. In *Proceedings of the British Machine Vision Conference*, pages 266–275, 1992.
- [12] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In *Computer Vision - ECCV'98*, volume 1407 of *Lecture Notes in Computer Science*. 1998.

## Bibliografia

- [13] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. 2001.
- [14] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2 edition, 2001.
- [15] Colin Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(2):285–339, 1991.
- [16] Ralph Gross, Iain Matthews, and Simon Baker. Active appearance models with occlusion. *Image and Vision Computing*, 24(1):593–604, 2006.
- [17] Vision Technology Group. *The Microsoft Vision SDK, Version 1.2*. Microsoft Research, 2000.
- [18] Michael J. Jones and Tomaso Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *Int. J. Comput. Vision*, 29(2):107–131, 1998.
- [19] F. Kahraman and M. B. Stegmann. Towards illumination-invariant localization of faces using active appearance models. In *IEEE NORSIG 2006, 7th Nordic Signal Processing Symposium, Reykjavik, Iceland (submitted)*, 2006.
- [20] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
- [21] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 900–903, 2002.
- [22] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
- [23] Martin Urschler Horst Bischof Markus Storer, Peter M. Roth and Josef A. Birchbauer. Efficient robust active appearance model fitting. *Book Communications in Computer and Information Science (CCIS)*, 68, 2010.
- [24] The MathWorks, Inc. *MATLAB 7 External Interfaces*, 2010.
- [25] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60:135–164, 2004.
- [26] Iain Matthews, Jing Xiao, and Simon Baker. 2D vs. 3D deformable face models: Representational power, construction, and real-time fitting. *Int. J. Comput. Vision*, 75:93–113, 2007.
- [27] K. Messer, J. Matas, J. Kittler, J. Lüttin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In *Second International Conference on Audio and Video-based Biometric Person Authentication*, pages 72–77, 1999.
- [28] M. M. Nordstrøm, M. Larsen, J. Sierakowski, and M. B. Stegmann. The IMM face database – an annotated dataset of 240 face images. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.
- [29] G. Papandreou and P. Maragos. Adaptive and constrained algorithms for inverse compositional active appearance model fitting. In *Proc. IEEE Int. Conf. on Comp. Vision and Pat. Rec. (CVPR)*, 2008.

- [30] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.
- [31] Daniel Rueckert, Alejandro F. Frangi, and Julia A. Schnabel. Automatic construction of 3D statistical deformation models using non-rigid registration. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 77–84, 2001.
- [32] Peter Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.
- [33] M. B. Stegmann. Active appearance models: Theory, extensions and cases. Master’s thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2000.
- [34] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME - a flexible appearance modelling environment. *IEEE Trans. on Medical Imaging*, 22(10):1319–1331, 2003.
- [35] Ronny Stricker, Christian Martin, and Horst-Michael Gross. Increasing the robustness of 2D active appearance models for real-world applications. In *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ICVS ’09, pages 364–373, 2009.
- [36] Barry-John Theobald, Iain Matthews, and Simon Baker. Evaluating error functions for robust active appearance models. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 149 – 154, April 2006.
- [37] C. Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(1):137–154, 1992.
- [38] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Learning non - rigid 3D shape from 2D motion. In *Advances in Neural Information Processing Systems 16*. 2004.
- [39] L. van der Maaten and E. Hendriks. Capturing appearance variation in active appearance models. In *CVPR4HB10*, pages 34–41, 2010.
- [40] Paul Viola and Michael J. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.
- [41] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [42] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2D+3D active appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 535–542, 2004.
- [43] Jing Xiao, Jinxiang Chai, and Takeo Kanade. A closed-form solution to nonrigid shape and motion recovery. Technical report, Robotics Institute, 2003.
- [44] Xin Yu, Jinwen Tian, and Jian Liu. Active appearance models fitting with occlusion. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 4679 of *Lecture Notes in Computer Science*, pages 137–144. 2007.