Predisposizione mediante tecniche di machine learning di un modello con parametri di shape e action units a partire da una mesh 3D

Riccardo Presotto e Davide Coluzzi

Abstract—Il lavoro proposto si pone come obiettivo la creazione di un modello in grado di separare i parametri di shape ed action units a partire da delle mesh 3D di volti umani. Per raggiungere tale scopo si sono acquisite, tramite Kinect, delle rappresentazioni tridimensionali del viso di 20 utenti, ognuno dei quali ha simulato, con diverso grado di enfatizzazione, sei diverse espressioni facciali.
I dati raccolti sono stati poi etichettati ed organizzati in un tensore a tre dimensioni su cui si é applicata una tecnica di riduzione di dimensionalitá multilineare, l'HOSVD.
Sono stati, quindi, estratti i parametri che caratterizzano le deformazioni del volto inerenti ad un cambio di espressione (action units) e quelli responsabili di una variazione di identitá (shape units).
Grazie a queste operazioni si é stati infine in grado di ricostruire, in modo indipendente, i volti degli utenti presi in esame, mantenendo circa il 90% della varianza, utilizzando un numero nettamente inferiore di dati rispetto al dataset di partenza.

1 Introduzione

Importanza del problema: Le funzioni della comunicazione non verbale, che puó essere considerata il linguaggio delle relazioni, sono molteplici, attraverso di essa si segnalano il tipo di relazione in corso e i mutamenti qualitativi delle relazioni interpersonali. In particolare nella comunicazione non verbale é il volto quello che evidenzia maggiormente le nostre emozioni e i nostri stati d'animo.
Negli ultimi anni sono state eseguite diverse ricerche riguardanti le variazioni nella forma dei volti e quelle dovute alle diverse espressioni. Le applicazioni sono state numerose e in svariati campi, come per esempio in computer vision, grafica 3D o psicologia, medicina ed economia. Ad esempio grazie a queste tecniche si potrebbe essere in grado di creare un assistente virtuale con una identitá fissa, capace di variare in modo dinamico la propria espressione facciale rispettando il contenuto e lo stile della conversazione.
In base all'applicazione pratica sono richiesti specifici modelli in grado di descrivere la forma e le variazioni del volto, legate alle espressioni e all'identitá di una persona.

Approccio seguito: I modelli multilineari sono sicuramente quelli che meglio rappresentano le variazioni statistiche di una volto 3D, permettendo di trattare in modo separato le componenti che definiscono l'identitá del soggetto rispetto a quelle che sono connesse alla sua espressione.
Per poter utilizzare questi modelli statistici é necessario che, per ogni soggetto preso in esame, sia catturato il medesimo set di espressioni e che, grazie all'utilizzo di labels, questi dati siano messi in corrispondenza semantica.
Nel lavoro proposto, i dati necessari sono stati a acquisiti utilizzando "Kinect", uno strumento che grazie a diverse tipologie di sensori installati al suo interno, ha permesso di catturare per ogni utente una rappresentazione tridimensionale del volto.
Tali dati sono stati poi elaborati al fine di estrarre il set di punti 3D legato ad ogni rappresentazione tridimensionale, a cui é stata poi associata una label inerente al soggetto e all'espressione svolta.
Successivamente, i dati raccolti sono stati organizzati in un tensore a tre dimensioni, seguendo un approccio tale da consentire ai modelli di riduzione di dimensionalitá di discriminare in modo separato le variazioni di identitá e di espressione.
Le tecniche utilizzate hanno, quindi, permesso di mantenere per ogni mesh 3D solo le informazioni caratterizzanti (shape ed action units), rendendo possibile la ricostruzione del dataset di partenza utilizzando un numero di dati inferiore, con una approssimazione piú che accettabile.

Contributi:

- Testing e perfezionamento del codice utile all'acquisizione di mesh 3D tramite Kinect;
- Acquisizione del dataset
- Elaborazione ed ordinamento dei dati tramite l'algebra multidimensionale;
- Riduzione di dimensionalitá tramite Higher-order

Riccardo Presotto, Corso di Interazione Naturale, A/A 2017-2018, Universitá di Milano, via Comelico 39/41, Milano, Italy
E-mail: riccardo.presotto@studenti.unimi.i
Davide Coluzzi, Corso di Interazione Naturale, A/A 2017-2018, Universitá di Milano, via Comelico 39/41, Milano, Italy.
E-mail: davide.coluzzi1@studenti.unimi.it

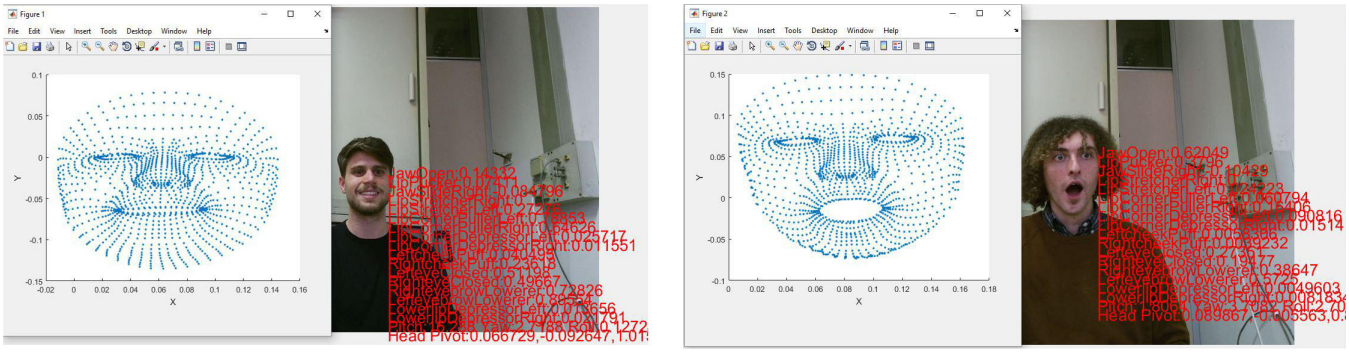

Fig. 1: Interfaccia di acquisizione utilizzata nella fase di sperimentazione

singular value decomposition (HOSVD);
- Ricostruzione di volti utilizzando le shape e le action units ottenute.

2 Analisi dello stato dell'arte

Modelli multilineari: I modelli multilineari sono stati utilizzati in diversi tipi di applicazioni. Vlasic [1] e Dale [2] hanno utilizzato un modello multilineare per ricostruire facce 3D da video 2D e trasferire espressioni tra soggetti diversi. Mpiperis in [3] ha utilizzato un modello bi-lineare per il riconoscimento di identitá ed espressioni utilizzando scansioni 3D di volti come fonte. Yang in [4] e Bolkart e Wuhrer in [5] sfruttano il disaccoppiamento delle variazioni tra identitá ed espressione per ottenere una rappresentazione compatta per le sequenze di movimento facciali; mentre in [6] i medesimi autori presentano un sistema capace di sfruttare il modello multilineare per la creazione di action e shape units anche in presenza di dati corrotti o mancanti.

Riduzione di dimensionalitá: Tipping e Bishop in [7] introducono la tecnica della PPCA dimostrando come come gli assi principali di un insieme di vettori di dati osservati possono essere determinati attraverso la stima della massima verosimiglianza dei parametri, in un modello variabile latente strettamente correlato all'analisi fattoriale. Liu in [8] estende tali tecniche in uno spazio multidimensionale tramite l'utilizzo di tensori.
Anche in [6] i dati vengono organizzati in un tensore, ma per ridurne la dimensionalitá viene applicata la Higher-Order Singular Value Decomposition (HOSVD).

3 Modello teorico

La variazione facciale puó essere divisa concettualmente in un numero di sottospazi "funzionali": tipologie di variazioni differenti che riflettono dimensioni facciali utili. Alcuni dei possibili spazi selezionabili sono: identitá, espressione (che include tutte le deformazioni transienti del viso), illuminazione e posizione. La finalitá del seguente lavoro é stata l'estrazione automatica dei sottospazi, tralasciando le variazioni relative all'illuminazione e alla posizione, al fine di ottenere una buona riduzione di dimensionalitá che massimizzasse, cosí, le deformazioni dovute ai cambiamenti di identitá ed espressione (shape/action units).

High-Order Singular Value Decomposition:
In questo paragrafo viene introdotto il modello multilineare che é stato applicato al dataset di "facce" acquisito, composto da d_2 identitá e d_3 espressioni. Ciascuno dei volti contenuti é rappresentato attraverso un modello 3D, con n vertici per ognuna delle tre dimensioni. Quando i dati raccolti sono matrici, l'approccio tradizionale prevede di vettorializzare le osservazioni. Ogni $x = (x_1, y_1, z_1, ..., x_n, y_n, z_n)^T$ denota quindi, un viso con n vertici $(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n)$.
Sia $X \in \mathbb{R}^{3n \times d_2 \times d_3}$ un tensore di ordine 3 in cui sono stati organizzati i dati: le diverse identitá sono state associate al secondo modo di X e le diverse espressioni al terzo. In figura (3) viene mostrata una rappresentazione dell'organizzazione dei dati nel tensore X e del comportamento che il modello dovrá assumere.
Per ottenere tale risultato é necessario innanzitutto centrare i dati sottraendo la media $\bar{x} = \frac{1}{d_2 d_3} \sum_{i=1}^{d_2 d_3} x_i$ ai modelli 3D di tutte i volti contenuti nel tensore. L'Higher-Order Singular Value Decomposition (HOSVD) [9] decompone X in un tensore ridotto $M \in \mathbb{R}^{3n \times p_2 \times p_3}$, in cui p_2 e p_3 sono il numero di componenti scelte per la riduzione, inerenti rispettivamente ai sottospazi di identitá ed espressione, e in due matrici di fattori $U_2 \in \mathbb{R}^{d_2 \times p_2}$ e $U_3 \in \mathbb{R}^{d_3 \times p_3}$ come mostrato in Eq. (1) ed in figura (3). Il tensore di dimensione minore M viene chiamato "core tensor", ed é la proiezione multilineare del tensore X dei dati originali sul sottospazio suddiviso in $U_2 \bigotimes U_3$ [10] dove $\bigotimes$ rappresenta il prodotto di

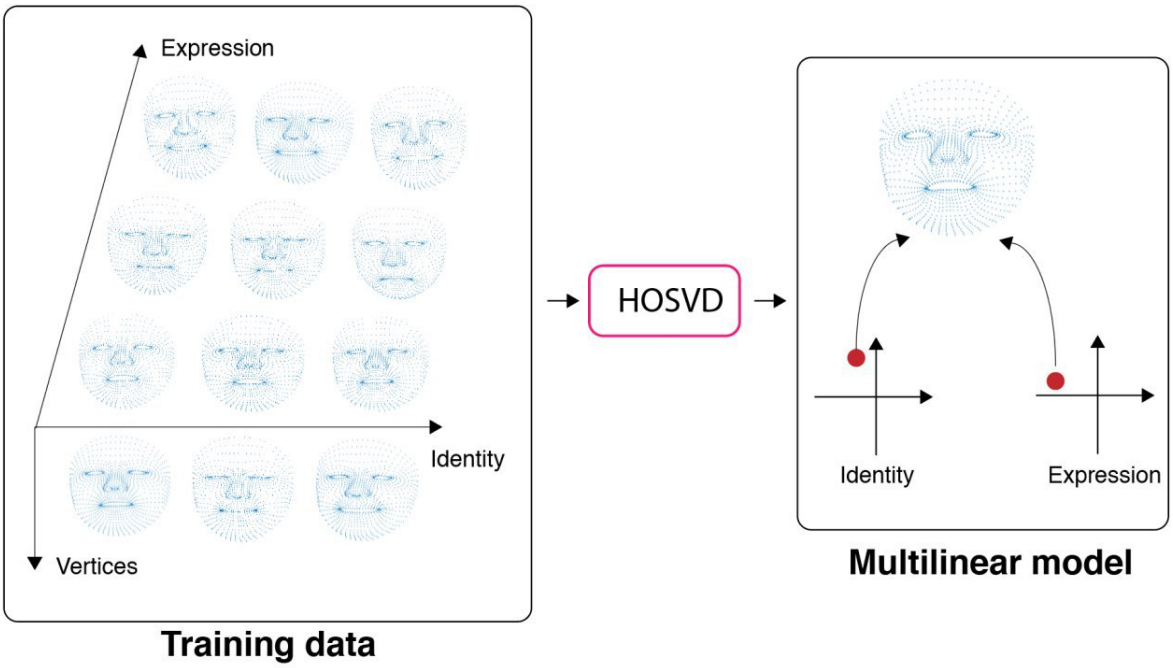

Fig. 2: Rappresentazione del modello multilineare

Kronecker tra matrici[1]:

$$X = M \times_2 U_2 \times_3 U_3 \tag{1}$$

dove $M \times_n U_n$ é l'n-th mode product del tensore M e della matrice U_n.
L'HOSVD procede come segue: il tensore dei dati X viene disposto lungo ogni modo ($k = 1, 2, 3$) in una matrice di "unfolding" indicata con X_k. Prendere il modo-k significa riorganizzare il tensore X in una matrice X_k in cui ognuna delle sue colonne é un vettore di modo-k di X (i vettori di X allineati con l'n-esimo modo formano le colonne di X_k). Dopodiché viene svolta una Singular Value Decomposition (SVD) su ogni X_k, per ottenere le basi ortonormali, come $X_k = U_k S_k V_k^T$. U_k rappresenta gli autovettori ottenuti da $X_k\, X_k^T$, S_k i valori singolari e V_k gli autovettori ottenuti da $X_k^T\, X_k$. Le basi ortonormali sono indicate con U_k con $k = 1, 2, 3$ e avranno come dimensione $d_k \times d_k$. A questo punto troncando le colonne di U_k, cioé prendendone le principali colonne, che rappresentano una completa base ortonormale per il modo k, si ottengono le basi parziali.
A partire da queste é quindi ora possibile ottenere una proiezione multilineare del tensore dei dati X nel sottospazio $U_2 \bigotimes U_3$ come $M = X \times_2 U_2 \times_3 U_3$. Il tensore ridotto M é il "core tensor" che permette di ricostruire correttamente ogni nuvola di punti relativa ad un volto $f \in \mathbb{R}^{3n}$ presi da U_2, i rispettivi coefficienti per l'identitá, $u_2 \in \mathbb{R}^{p_2}$ e da U_3 quelli per l'espressione $u_3 \in \mathbb{R}^{p_3}$. I passaggi necessari sono riassunti nell'algoritmo 1 riportato:

$$f = \bar{x} + M \times_2 u_2^T \times_3 u_3^T \tag{2}$$

1. Il prodotto di Kronecker di due matrici $A \in \mathbb{C}^{I \times J}$ e $B \in \mathbb{C}^{K \times L}$, indicato con $A \bigotimes B$, é una matrice di dimensioni $(IK) \times (JL)$ definita da:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}$$

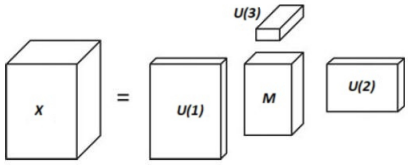

Fig. 3: Rappresentazione della decomposizione operata dalla HOSVD a partire dal tensore originale X

4 Simulazione ed esperimenti

Per l'acquisizione dei dati necessari alla fase di sperimentazione é stato deciso di utilizzare come strumento di cattura "Kinect". Tale scelta é stata fatta per due ragioni principali: prima di tutto, permette di riconoscere automaticamente il volto di un soggetto posizionato davanti ai suoi sensori e, in secondo luogo, ne descrive la conformazione con un pattern di ben 4041 punti 3D (un numero nettamente superiore rispetto a quello fornito ad altri sistemi e tecniche di acquisizione [11]).
Inoltre Microsoft (produttore del dispositivo) mette a disposizione numerose API per sfruttare a pieno le potenzialitá del dispositivo in ambiente Matlab. Ció ha reso

Algorithm 1 HOSVD algorithm

Input: Un insieme di campioni nel tensore $X \in \mathbb{R}^{3n \times d_2 \times d_3}$ e le dimensioni desiderate dei sottospazi del tensore p_1, p_2 e p_3.
Process:
1: **Centering:** Calcolata la media $\bar{x} = \frac{1}{d_2 d_3} \sum_{i=1}^{d_2 d_3} x_i$:
for $i = 1$ to d_2 **do**
  for $j = 1$ to d_3 **do**
    Si calcoli $X_{ij} = X_{ij} - \bar{x}$
2. **Decomposition:** High Order Singular Value Decomposition
for $k = 1$ to N **do**
  Si calcoli $X_k = U_k S_k V_k^T$
$M' = X \times_2 U_2 \times_3 U_3$
3. **Reconstruction:** Per ogni volto $f \in \mathbb{R}^{3n}$ individuata in X con X_{ij}, si prendano $u_2 \in \mathbb{R}^{p_2}$ da $U_{2(i)}$ e $u_3 \in \mathbb{R}^{p_3}$ da $U_{3(j)}$ e si calcoli:
$f = \bar{x} + M \times_2 u_2^T \times_3 u_3^T$

possibile la scrittura di un codice apposito, che permetta anche ad un utente inesperto di scattarsi autonomamente una "fotografia tridimensionale" del volto.
Nello specifico, per la fase di acquisizione é stata fornita ad ogni utente una lista indicante le sei espressioni universali (felicitá, sorpresa, disgusto, rabbia, paura, tristezza) e gli é stato poi chiesto di simularne tre diverse rappresentazioni, enfatizzando l'espressione in modo crescente.
All'utente, inoltre, sono state fornite le indicazioni per interagire con il codice di acquisizione, che consistono unicamente nell'inserimento di un proprio codice identificativo e nella pressione del tasto "c" per salvare la rappresentazione tridimensionale del proprio viso, vedi figura 1.
Questa scelta é stata fatta per assicurarsi che l'utente, senza alcuna interferenza esterna, salvasse i dati nel momento in cui sentiva di avere simulato nel miglior modo possibile l'espressione richiesta.
Lo script realizzato in Matlab (*AcquireData.mat*) ha poi etichettato in modo autonomo ogni rappresentazione tridimensionale del volto, inserendo per ognuna di esse l'identificativo del soggetto, il nome dell'espressione eseguita ed un codice inerente al grado di enfatizzazione della stessa. Successivamente sono state svolte alcune operazioni di sperimentazione ed esplorazione dei dati acquisiti, e, attraverso alcune conversioni, si sono poi salvati i dati in file in formato CSV.

4.1 Dataset

Il dataset acquisito si compone di un totale di 360 rappresentazioni tridimensionali di volti umani (vedi Figura 4), nello specifico sono stati presi in esame 20 utenti, ognuno dei quali ha eseguito tre differenti gradi di enfatizzazione di sei espressioni facciali.
Ogni campione é formato da 4041 punti espressi in uno spazio tridimensionale che sono stati memorizzati in

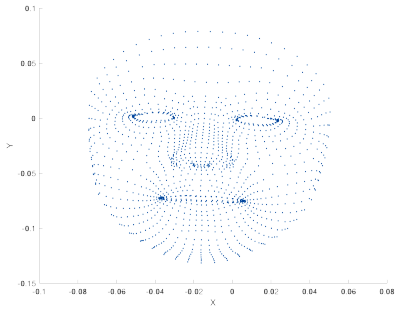

Fig. 4: Rappresentazione tridimensionale di un volto fornita da Kinect

formato tabellare (CSV), e suddivisi in base al proprio asse di appartenenza (x, y, z).
Ogni file é stato inoltre provvisto di una etichetta, formattata in modo tale da renderla nello stesso tempo sia espressiva che facilmente leggibile da una iterazione: il codice assegnato a ogni utente é numero progressivo compreso tra 0 e 19 preceduto dalla lettera u, il nome dell'espressione rappresentata é stato scritto per intero, mentre il livello di enfatizzazione é Anch'esso un numero da 0 a 2 (es: u15_felice_2).
Al fine di rendere i dati utilizzabili dai modelli multilineari presi in esame, essi sono stati poi riformattati in un unico array, ordinando i campioni sui singoli assi in modo da ottenere una sequenza nel seguente formato $(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n)$.

4.2 Architettura del sistema

Il sistema realizzato per questo progetto é composto da tre moduli per implementare la riduzione di dimensionalitá e di due moduli che permettono la ricostruzione del dataset originale e delle singole facce presenti in esso; un esempio é presentato nella figura 5.
I dati grezzi raccolti tramite Kinect vengono elaborati al fine di utilizzare le sole misurazioni utili allo scopo ed essere poi inviate al modulo organizzativo. Questo modulo ha il compito di ordinare in un tensore tridimensionale le mesh 3D catturate, facendo in modo che nelle colonne siano presenti le varie espressioni simulate dallo stesso soggetto, mentre nelle righe i volti di utenti diversi che performano la stessa espressione.
Una volta creato il tensore esso é dato in input al modulo che si occupa della riduzione di dimensionalitá, il quale lo scompone in un sotto-tensore (core-tensor), e tre matrici (una per ogni dimensione del tensore) in cui sono presenti i coefficienti che pesano i valori singolari ottenuti dalla SVD su ciascuno dei modi del tensore.
A questo punto possono essere fatte due operazioni distinte: utilizzare il core-tensor e le tre matrici per ricostruire in modo approssimato il tensore originale, oppure estrarre le shape e le action units per ricreare i singoli volti degli utenti durante lo svolgimento di una specifica espressione.
Nel primo caso basterá prendere i dati in output dal

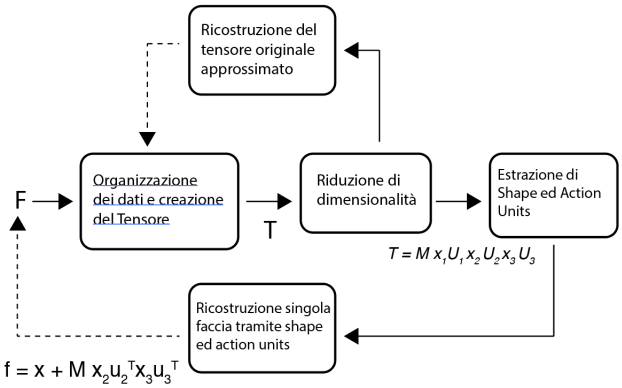

Fig. 5: Architettura del sistema. F rappresenta le varie mesh 3D raccolte; $T \in \mathbb{R}^{3n \times d_2 \times d_3}$ é il tensore che contiene il dataset ordinato; T viene scomposto applicando la $HOSVD$ in un Tensore ridotto M e tre matrici di $U1, U2$ ed $U3$. Tramite questa rappresentazione scomposta di T si possono ricavare le shape(u_3^T) e le action(u_3^T) units che permettono di ricostruire una faccia f nelle modalitá espresse nella formula (vedi algoritmo 1)

.

modulo di riduzione di dimensionalitá e moltiplicare ogni matrice per la moda del core-tensor corrispondente. Nel secondo caso sará invece necessario calcolare la "faccia media" e sommarle la ricostruzione delle variazioni introdotte dalla forma del viso di uno specifico utente mentre simula una certa espressione, svolgendo l'n-mode product tra il core-tensor e le shape e le action units corrispondenti.

4.3 Dettagli implementativi

La parte implementativa del progetto é stata sviluppata utilizzando Matlab, per quanto concerne l'acquisizione dei dati, mentre Python 3.6 per lo sviluppo dell'algoritmo, la rielaborazione dei risultati finali e la creazione dell'interfaccia grafica.
Nella prima fase si fa uso della toolbox Matlab **Kin2**, sviluppata da J. Terven e D. M. Cordova, che permette di interfacciarsi con le API (figura 6) della Kinect [12]. L'invezione di Microsoft Kinect ha rivoluzionato il mondo della computer vision poiché a basso costo permette la combinazione degli spazi di colore e profonditá (RGB-D) ed é stata usata con successo in moltissime applicazioni che includono: tracking e recognition di oggetti, analisi di attivitá umane, motion capture, recognition dei gesti e la realtá aumentata. Kin2 fornisce la maggior parte delle funzionalitá di Kinect come il colore, la profonditá, i raggi infrarossi, il body tracking, l'acquisizione di un modello di punti per le facce e la ricostruzione 3D di esse.
La toolbox é stata, quindi, utilizzata per ottenere un modello preciso del viso e avere a disposizione un buon numero di dati di training. La funzione *Kin2.getHDFaces()* restituisce in output un array strutturato con al massimo sei rappresentazioni, ognuna delle quali possiede diversi campi contenenti informazioni utili - in particolare il campo FaceModel cattura un modello 3D di un volto costituito da 3 x 1347 punti.
La realizzazione dell'algoritmo per ottenere le action/shape units e i test sui risultati ottenuti sono stati realizzati in Python. Innanzitutto, a partire dai dati acquisiti in Matlab e formattati in *CSV* si é creato il tensore. Per la creazione e gestione delle strutture dati si é fatto uso del package di Python *NumPy* che aggiunge supporto per matrici e vettori multidimensionali. Ciascun modello di punti rappresentante un volto é stato vettorializzato, come giá anticipato in 4.1, e poi disposto nel tensore.
Maggiori dettagli e codice in Appendix.
Si é proceduto poi al centramento dei dati, unitamente al calcolo di una "faccia media", ed infine allo svolgimento della SVD sui modi del tensore. Dopo quest'ultima operazione, avendo a disposizione le due matrici di fattori, é stato quindi possibile calcolare la proiezione multilineare del tensore di partenza come riportato nell'equazione 1. Di seguito la porzione di codice:

```
core = np.zeros((4041, 8, 8))

prod2 = n_mode.mode_dot(allTen2, mode2.T, 1)
core = n_mode.mode_dot(prod2, mode3.T, 2)
```

Come si puó notare é stata utilizzata la funzione *mode_dot* del package *tensorly.tenalg.n_mode_product* che permette di svolgere l'n-mode product.
Per quanto concerne la parte di sperimentazione e test, si é fatto ampio uso di *matplotlib*, una libreria per la creazione di grafici per Python e NumPy. Per prima cosa si sono voluti visualizzare i valori singolari ottenuti dalla SVD di ciascun modo e, dopo di ció, si é calcolata la percentuale di varianza catturata da ognuno di questi. Entrambe le informazioni sono state riportate su dei grafici (figura 7) per mezzo del comando *plot* della suddetta libreria.
L'interfaccia grafica é stata realizzata in Python con il package *Tkinter*. In particolar modo nel main dello script "HosvdGUI.py" si predispone una nuova finestra di visualizzazione all'interno di cui viene visualizzato un "volto medio" e i sedici sliders inizialmente settati a zero. Dalle matrici di fattori sono stati ottenuti il massimo e il minimo per ciascuno dei pesi e questi valori sono stati utilizzati per settare l'intervallo entro cui ciascun slider puó essere variato.
Nella funzione "update", poi, vengono letti i valori scelti in ciascuno degli sliders e viene, cosí, ricalcolato e visualizzato il risultato della ricostruzione.

5 Risultati ottenuti

I risultati ottenuti attraverso l'utilizzo del modello multilineare sono innanzitutto visibili nell'interfaccia che é stata realizzata. Sono stati predisposti sedici

sliders - otto per l'identitá e otto per l'espressione - per visualizzare le variazioni catturate. E' possibile, quindi, esplorare i risultati variando i pesi per ciascuna delle units ottenute.
La figura 8 mostra una vista dell'interfaccia realizzata. Variando il valore degli sliders é possibile esplorare i risultati, notando le variazioni catturate da ciascuna unit. In figura 9 si é mostrato il cambiamento nella ricostruzione, aumentando di volta in volta la prima action unit.
Per quanto concerne la scelta del numero di shape/action unit sono state effettuati alcuni test. Innanzitutto, empiricamente, sono state fatte alcune prove di ricostruzione di un particolare volto e si é potuto notare come aumentando di due in due il numero, sia per l'identitá che per l'espressione, l'approssimazione diveniva sempre piú precisa. In particolare in fig. 10 é stato preso in esame un volto che esprime felicitá e si mostra la ricostruzione che é stata effettuata con 2, 4, 8 e 10 unit. Si puó notare come, in particolare il sorriso del volto preso in esame, sia, man mano che si utilizza un numero maggiore, sempre piú simile a quello originale. Risulta chiaro, dunque, che minore é il numero scelto, minori saranno le variazioni catturate e maggiore sará la somiglianza con la "faccia media" piuttosto che con quella originale. Si puó inoltre proseguire dicendo che, dal caso preso in esempio, non risultano significative differenze tra scegliere di mantenerne 8 o 10.
Approfondendo dunque la disamina riguardante i contributi forniti alla ricostruzione é stato confermato che 8 rappresentasse un buon numero sia per quanto riguarda le shape che le action units. Sono stati presi, infatti, tutti i valori singolari ottenuti dai modi 2 e 3 e sono stati rappresentati nei grafici 7 (b) e 7 (d) riportati. Per procedere nella scelta del numero da utilizzare per ogni sottospazio é stata calcolata la percentuale di varianza spiegata e si é deciso di mantenerne 8 per entrambe in quanto é stato mostrato che, in questo modo, é circa del 90%. Nei grafici mostrati 7 (c) e 7 (a) si puó visualizzare la percentuale cumulata di tutti i valori singolari per ciascuno dei due modi. É facilmente visibile come, scegliendo i primi otto, in entrambi i casi, si mantenga circa il 90% della varianza, cioé si ottiene un'ottima approssimazione dell'informazione originale (per i valori di varianza in percentuale si consulti la tabella 1).
Per concludere, si puó affermare di aver ottenuto dei buoni risultati in quanto con una perdita minima (10%) di informazione é stata portata a termine una buona riduzione dei dati. Le dimensioni del tensore di partenza erano, infatti, $4041 \times 20 \times 18$, ció significa che erano contenuti 1.454.760 valori float, dopo le operazioni, invece, sono diventate $4041 \times 8 \times 8$, cioé soltanto 258.624 valori float, che a fronte di una perdita minima rappresentano un ottimo risultato.

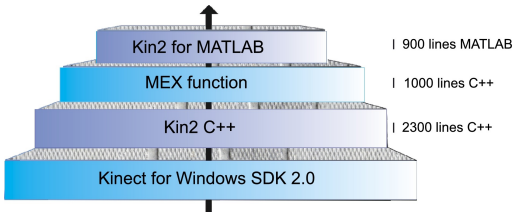

Fig. 6: Diagramma a strati. Kin2 ha un'architettura a quattro layer. Nella parte inferiore si trova Kinect per Windows SDK 2.0; il secondo layer é una versione C++ di Kin2 che incapsula molte delle funzionalitá delle SDK ed effettua una conversione tra le varibili C++ e quelle Matlab.; il layer successivo é composto da una funzione Mex Matlab che fornisce un'interfaccia tra C++ e Matlab; infine, in alto, si trova la toolbox Kin2 per Matlab composta da una classe e funzioni ad alto livello.

6 Commenti conclusivi

Le metodologie utilizzate per l'acquisizione dei dati hanno permesso di lavorare con rappresentazioni tridimensionali formate da 4041 punti, un numero nettamente superiore rispetto ad altre tecniche simili presenti in letteratura.
Questo ha permesso di approssimare con molta precisione sia le variazioni del volto che sussistono tra due soggetti con identitá diversa, sia il movimento dei muscoli facciali responsabili del cambio di espressione. Grazie ai risultati ottenuti é stato inoltre possibile suddividere con precisione le shape e le action units, che potrebbero essere sfruttate come in [2] per trasferire identitá ed espressione tra soggetti, con la differenza nel progetto citato i dati in input provengono da immagini bidimensionali e non mesh 3D come nel nostro caso.
Lo svantaggio di lavorare con una dimensionalitá cosí alta si traduce in un inevitabile allungamento dei tempo computazionali che porta quindi a pensare che questi tipi di procedimenti siano sconsigliati per essere sfruttati run

TABLE 1: Contenuto informativo cumulativo

$\sigma(Id)$	$\%Cum.Variance$	$\sigma(Ex)$	$\%Cum.Variance$
1	30.44378	1	49.15353
2	47.30443	2	59.47071
3	59.84357	3	68.75581
4	67.56120	4	75.95117
5	73.70931	5	81.58384
6	78.29920	6	85.30994
7	81.93849	7	88.61469
8	84.57036	8	90.31999
9	87.01149	9	91.97946
10	89.24354	10	93.38639
11	91.06075	11	94.78949
12	92.61063	12	96.14652
13	93.95190	13	97.19222
14	95.19670	14	97.95579
15	96.23265	15	98.61881
16	97.26084	16	99.17821
17	98.08666	17	99.63476
18	98.88325	18	100
19	99.49492		
20	100		

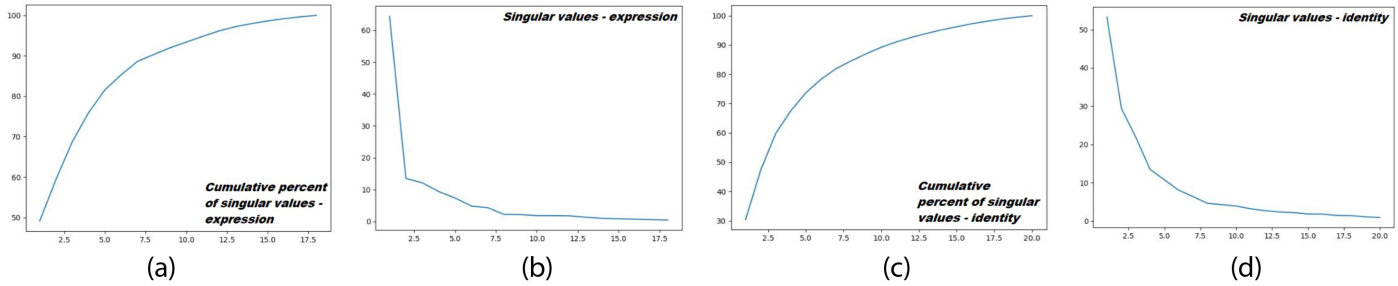

Fig. 7: (a) Percentuale cumulata di tutti valori singolari ottenuti dalla svd del modo 2 del tensore (espressioni); (b) Valori singolari ottenuti dal modo 3 del tensore, quello relativo alle espressioni; (c) Percentuale cumulata di tutti valori singolari ottenuti dalla svd del modo 2 del tensore (identitá); (d) Valori singolari ottenuti dal modo 2 del tensore, quello relativo alle identitá

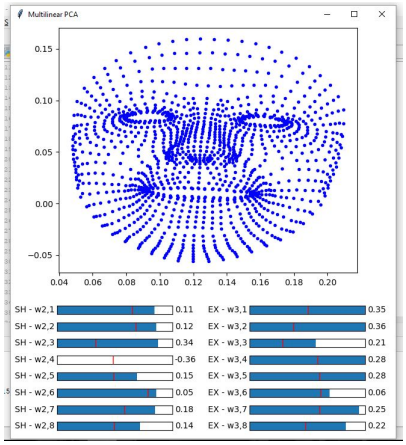

Fig. 8: Vista dell'interfaccia realizzata. Sono stati predisposti 16 sliders attraverso cui é possibile variare le shape e le action units.

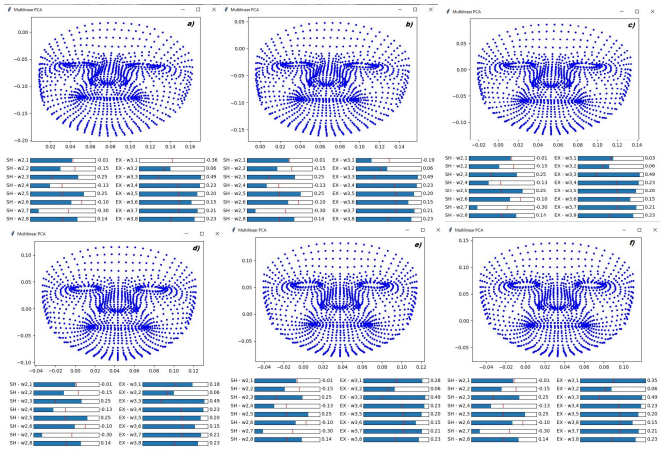

Fig. 9: Variazione catturata dalla prima action unit. La ricostruzione avviene aumentando gradualmente solo il peso della prima action unit. Si puó notare come le caratteristiche principali catturate sono le variazioni degli occhi e della bocca. Nella figura (a), infatti, gli occhi e la bocca sono quasi totalmente chiusi, mentre man mano che si aumenta il peso si aprono sempre di piú - e inoltre, appare un sorriso sempre meno accentuato - fino al massimo rappresentato dalla figura (f).

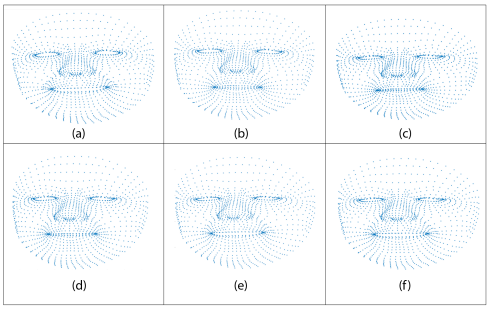

Fig. 10: (a) immagine originale; (b) faccia media; $(c), (d), (e)$ ed (f) ricostruzione della faccia originale usando rispettivamente $2, 4, 8$ e 10 componenti principali di shape e action units.

time, preferendo invece un approccio batch.
Uno degli sviluppi futuri potrebbe essere usare una decomposizione di Tucker tramite l'Higher Order Orthogonal Iteration (HOOI), passando come input il tensore e il numero di componenti scelte per la riduzione. La HOOI, a differenza della HOSVD, non agisce parallelamente sui modi, ma sequenzialmente, questo porta ad una maggiore efficienza computazionale. Trova infatti un'ottima approssimazione tra il tensore di partenza e la sua decomposizione iterando fino a convergenza l'operazione di troncamento e la SVD sui modi del tensore stesso, come riportato nell'algoritmo 2.
Inoltre si sono verificate alcune problematiche nella creazione del dataset, infatti alcuni soggetti non sono riusciti a scattarsi le foto tridimensionali mantenendosi esattamente perpendicolari ai sensori della Kinect, ció ha portato ad una lieve deformazione sull'asse x della "faccia media" che ha influito nella ricostruzione dei campioni dopo la riduzione di dimensionalitá.
Inoltre il dataset utilizzato si compone di "soli" 360 esempi di volti umani, un numero decisamente sufficiente per testare la validitá del modello, ma non per testarne l'efficienza effettiva in comparazione con altri modelli che si sono potuti avvalere di un dataset decisamente piú ampio.
Inoltre le espressioni prese in esame si rifanno unicamente alle sei espressioni universali, numero che

potrebbe essere sicuramente ampliato al fine di ottenere dei risultati che potrebbero essere utili per scopi piú eterogenei.
In conclusione, possiamo affermare che il lavoro presentato é riuscito a dimostrare che grazie alla riduzione di dimensionalitá applicata su una ricostruzione di punti 3D di un volto umano, con 8 shape ed 8 action units, si riesce effettivamente a descrivere con una perdita informativa del solo 10%, le variazioni che sussistono tra utenti che eseguono diverse espressioni. Ció ha quindi consentito sia la ricostruzione del dataset originale, utilizzando meno dati rispetto a quelli di partenza, sia la creazione di volti ibridi in cui vengono trasferite identitá ed espressione provenienti da soggetti diversi.

Algorithm 2 HOOI algorithm

Input: Un insieme di campioni nel tensore $X \in \mathbb{R}^{3n \times d_2 \times d_3}$ e le dimensioni desiderate dei sottospazi del tensore p_1, p_2 e p_3.
1. **Iniziatilizing:** Scelti U_2 e U_3 iniziali con colonne ortonormali.
2. **Decomposition:** Higher Order Orthogonal Iteration
while Fino a convergenza (max 100 iterazioni) **do**
Calcola $N = X \times_2 U_2^T \times_3 U_3^T$
Calcola $U_1 = SVD(p_1, N_{(1)})$
Calcola $N' = X \times_1 U_1^T \times_3 U_3^T$
Calcola $U_2 = SVD(p_2, N'_{(2)})$
Calcola $N'' = X \times_1 U_1^T \times_3 U_2^T$
Calcola $U_3 = SVD(p_3, N''_{(3)})$
$M' = N'' \times_3 U_3^T$

APPENDIX
NOTA SULLA CREAZIONE DEL DATASET

Apertura dei file (CSV) in base ai nomi assegnativi:

```
matrix = np.zeros((20, 18, 4041))
while True:
    if expr_count == 0:
        exp_name = 'felice'
    if expr_count == 3:
        exp_name = 'triste'
    if expr_count == 6:
        exp_name = 'spaventato'
    if expr_count == 9:
        exp_name = 'arrabbiato'
    if expr_count == 12:
        exp_name = 'disgustato'
    if expr_count == 15:
        exp_name = 'sorpreso'
    if expr_count == 18:
        user_count = user_count + 1
        expr_count = 0;
        exp_name = 'felice'
    if user_count == 21:
        break;
    count_mod = operator.mod
                    (expr_count, 3)
    user_name = ('u' + str (user_count
              + 1) + '_' + exp_name
              + '_' + str(count_mod))
    path = 'C:/Users/Davide Coluzzi/
             Desktop/Progetto_Presotto
         _Coluzzi/Matlab/Dataset_Csv/'
                 + user_name + '.csv'
    expr_count = expr_count + 1
```

Organizzazione dei dati nel tensore:

```
csv = np.genfromtxt(path,
                 delimiter=",")
file_count = file_count + 1;
one_line = np.array([])
    for i in range(0, 1347):
        for j in range(0, 3):
            one_line = np.append
               (one_line, csv[j, i])
    matrix[riga][colonna] = one_line
    colonna = colonna + 1
    if colonna == 18:
        riga = riga + 1
        colonna = 0
```

REFERENCES

[1] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 426–433, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073209

[2] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 130:1–130:10, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024164

[3] I. Mpiperis, S. Malassiotis, and M. G. Strintzis, "Bilinear models for 3-d face and facial expression recognition," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 498–511, Sept 2008.

[4] F. Yang, L. Bourdev, E. Shechtman, J. Wang, and D. Metaxas, "Facial expression editing in video using a temporally-smooth factorization," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 861–868.

[5] T. Bolkart and S. Wuhrer, "3d faces in motion: Fully automatic registration and statistical analysis," *Computer Vision and Image Understanding*, vol. 131, pp. 100 – 115, 2015, special section: Large Scale Data-Driven Evaluation in Computer Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314214001404

[6] ——, "A robust multilinear model learning framework for 3d faces," *Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[7] T. M. E. B. C. M. Michael E. Tipping, Christopher M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, January 1999.

[8] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, Jan 2013.

[9] L. De Lathauwer, *Signal processing based on multilinear algebra*. Katholieke Universiteit Leuven Leuven, 1997.

[10] T.-L. Chen, S.-Y. Huang, H. Hung, and I.-P. Tu, "An introduction to multilinear principal component analysis," , vol. 52, no. 1, pp. 24–43, 2014.

[11] R. Y. B. Z. Baoying Ma, Junfeng Yao, "The improvement of parameterized face model of candide based on mpeg-4 and facs," *International Journal of Electrical Energy, Vol. 2, No. 2*, June 2014.
[12] J. R. Terven and D. M. Córdova-Esparza, "Kin2. a kinect 2 toolbox for matlab," *Science of Computer Programming*, vol. 130, pp. 97–106, 2016.