Università degli studi di Cagliari

Dipartimento di Ingegneria Elettrica ed Elettronica
Corso di Laurea Triennale in Ingegneria Elettrica ed Elettronica

Photometric Stereo e Archeologia

Algoritmi e Sperimentazioni

Candidato:
Paolo Floris

Relatore:
Prof. Massimo Vanzi

Correlatore:
Prof. Giuseppe Rodriguez

Anno Accademico 2016/2017

Alla mia famiglia, di sangue e non.
Ma soprattutto a Luisanna,
che altrimenti si offende.

Abstract

A brief introduction to the Photometric Stereo and other 3D reconstruction techniques is given, followed by an extensive coverage of the mathematical concepts needed to explain the algorithm in a fluid way. Ideal and typical acquisition conditions are described, together with tips for achieving better results. Experiments on real and highly valuable archaeological findings (such as *Mont'e Prama Giants* and several ancient Phoenician coins) were conducted and are here analyzed in detail.

Indice

Prefazione

La Photometric Stereo è una tecnica molto avanzata nel campo della *Computer Vision* che consente di costruire, a partire da (almeno) 4 immagini, un modello 3D dell'oggetto ripreso dalle immagini. Nello specifico, questo documento si pone di continuare il lavoro portato avanti negli anni all'Università di Cagliari dai professori G.Rodriguez [1], M.Vanzi e G.Tanda [2], ma anche dagli studenti C.Mannu, R.Dessì e R.Pintus [3] [4] [5]. Grazie a loro la tecnica ha vissuto uno sviluppo considerevole, e in questo lavoro abbiamo cercato di trasportarne i miglioramenti sul campo. Insomma, con questo lavoro l'algoritmo esce di fatto dal suo "stadio embrionale", e attraversa una fase di collaudo, che è cruciale in relazione al progetto Europeo che potrebbe vedere il Dipartimento di Ingegneria Elettrica ed Elettronica (**DIEE**) come leader nella documentazione di siti di interesse archeologico in tutta Europa.
In dettaglio, nel Capitolo 1 verrà si introduce qualitativamente la tecnica, insieme ad altri importanti processi di *3D imaging*, mettendo in risalto i vantaggi della Photometric Stereo nel campo dell'archeologia. Nel Capitolo 2 sono affrontati i concetti matematici alla base di tutto l'algoritmo, di modo che la descrizione del software (nel Capitolo 3) stesso risulti più fluida. Il Capitolo 4 riporta brevemente le modalità di acquisizione delle immagini che consentano di ottenere i risultati migliori, con particolare attenzione alle condizioni di illuminazione. Il Capitolo 5 commenta invece le prove effettuate al Museo Archeologico sui Giganti, e sulle monete al **DIEE**, insieme alle conclusioni e a possibili futuri sviluppi. Preziosa è stata la collaborazione del Dott. Roberto Concas, Direttore del Museo

Archeologico di Cagliari, che ci ha assistito durante le sperimentazioni effettuate su reperti di grande interesse storico e archeologico (alcuni dei famosi *Giganti di Mont'e Prama*). Sentiti ringraziamenti vanno anche al Dott. Luca Alagna, che a sua volta ci ha gentilmente fornito delle antiche monete Fenicie, che hanno costituito un interessantissimo banco di prova per l'algoritmo.
Mi sento di estendere i miei ringraziamenti personali ai miei Relatori ufficiali e alla mia "Relatrice volontaria" Prof.ssa Giovanna Mura, per la loro disponibilità e infinita pazienza. Ultima ma non ultima, la Dott.ssa Carla Mannu, con cui ho avuto il piacere di collaborare e che ha condiviso con me la sua vasta esperienza sul campo.

Capitolo 1

Introduzione

Il primo articolo riguardante la Photometric Stereo risale al 1980 (Woodham [6]). Nell'articolo Woodham risale all'orientazione di una superficie facendo uso di una *Reflectance Map*, cioè utilizzando una sfera fatta dello stesso materiale dell'oggetto che si vuole studiare, che permette di mappare le intensità ottenute nelle immagini direttamente alle normali alla superficie.
Gli articoli che si sono poi succeduti sono innumerevoli e hanno affrontato e risolto dei problemi che ponevano limitazioni importanti alla tecnica, tra cui per esempio la complessità computazionale, e altre criticità riguardanti equipaggiamento e modalità di acquisizione, su cui ci si è concentrati all'Università di Cagliari. [2] [3] [4] [5]
Il procedimento consiste, in maniera molto sintetica, nell'acquisire un set di immagini di un oggetto, riprese con uguale angolazione, ma diverse condizioni di illuminazione.
Esistono due "macro-approcci" nell'implementazione dell'algoritmo che sono stati affrontati dall'Ing. Dessì in [4], e cioè conoscendo la posizione delle luci, oppure non conoscendola. Daremo una panoramica di entrambi, ma approfondiremo principalmente il secondo approccio, perchè è quello che conferisce grande flessibilità nell'applicazione della PS. Infatti utilizzando il primo metodo, serve grande precisione nel posizionamento delle sorgenti luminose, che non è sempre facile (o possibile) ottenere in applicazioni sul

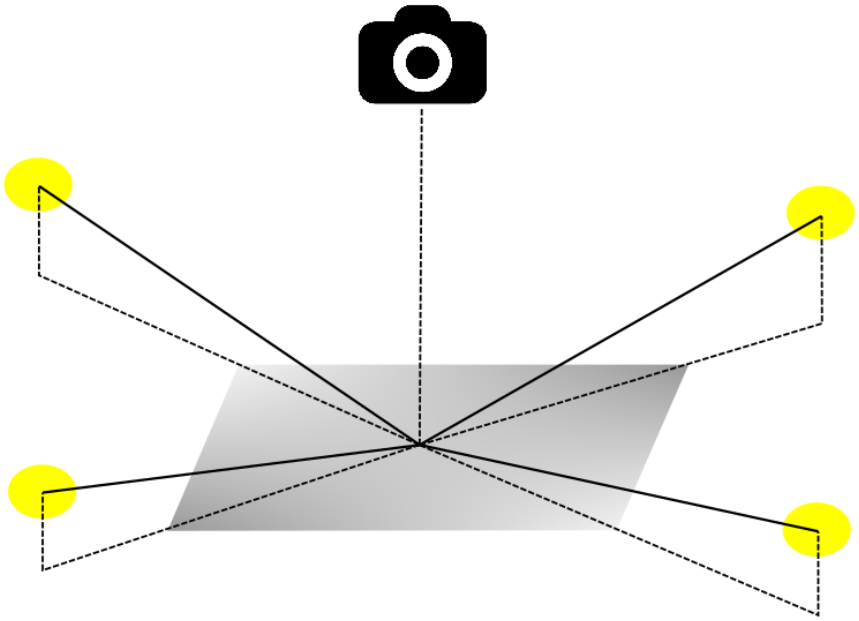


Figura 1.1: Modellizzazione del setup sperimentale

campo. Le ragioni prinicipali che rendono la PS appetibile per l'archeologia sono

- la facilità di acquisizione delle immagini: è sufficiente disporre di una fotocamera e un treppiede, o una struttura fissa, e una fonte luminosa con comandi remoti;
- il basso costo, specialmente se confrontata con le altre tecniche;
- la rapidità di calcolo;
- l'affidabilità dei risultati;
- la possibilità di separare correttamente forma e colore.

Virtualmente, la PS trasporta il sito archeologico nello studio dell'archeologo, ed è inoltre importante sottolineare il crescente bisogno di creare una banca dati elettronica dei reperti e delle opere, in modo da poterli preservare dall'erosione a cui possono essere soggetti, oltre che da catastrofi naturali (e non).

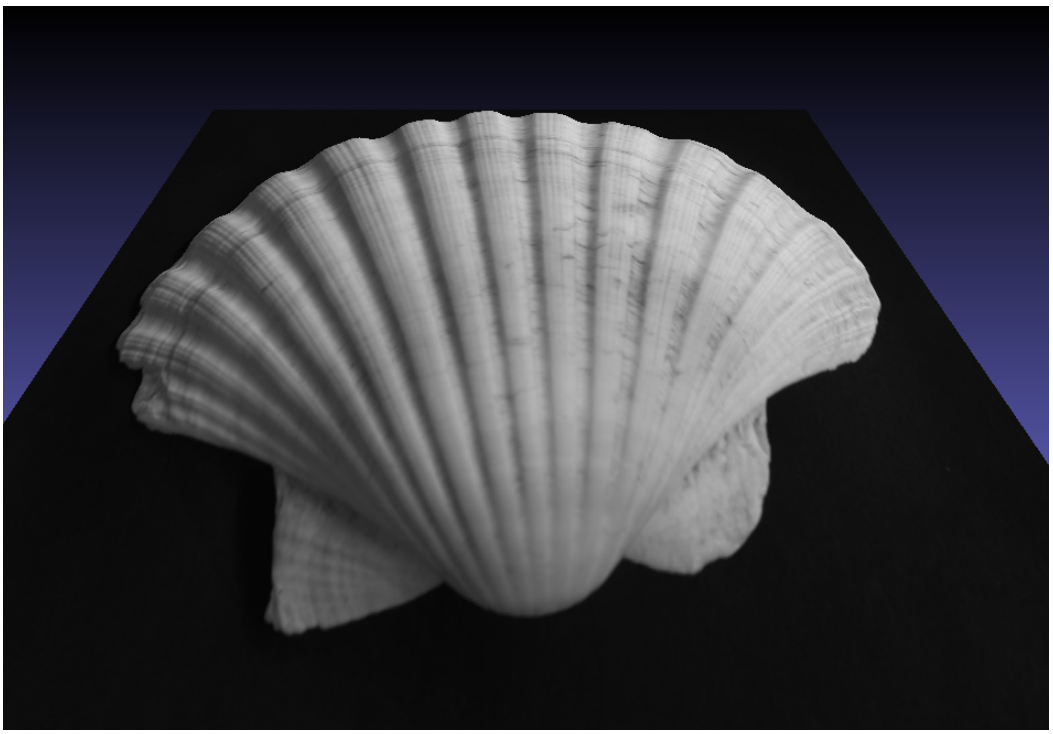


Figura 1.2: Ricostruzione di una conchiglia

1.1 Formulazione del Problema

Il principio fisico su cui la PS pone le fondamenta è la *Legge di Lambert* o *Legge del Coseno*, la quale afferma che l'intensità della radiazione riflessa da una superficie cosiddetta *Lambertiana* è data dal prodotto scalare tra la direzione della sorgente luminosa $\boldsymbol{\ell}$ e la normale alla superficie stessa $\mathbf{n}$.

$$I = \boldsymbol{\ell}^T \mathbf{n} \cos \alpha \tag{1.1}$$

se l'angolo α è quello compreso fra i vettori. In altre parole significa che indipendentemente dal punto di vista, un osservatore percepisce la stessa luminosità provenire da ciascun "pezzetto" di superficie, poichè anche se la luce viene riflessa con minore intensità per angoli crescenti, allo stesso modo spostando il nostro punto di vista nella stessa direzione diminuisce anche l'area di emissione e i due effetti si bilanciano lasciando la *luminanza* (cd/m^2) costante.

Supponiamo di avere n immagini (per comodità in scala di grigio) dell'oggetto che vogliamo ricostruire, ottenute con n direzioni di luce diverse $\boldsymbol{\ell}_j$,

$j = 1, ..., n$. Le immagini digitali hanno dimensioni $m = row \times col$ e le ordiniamo in vettori riga $\mathbf{m}_1$,...,$\mathbf{m}_n \in \mathbb{R}^m$ (ordine lessicografico). Possiamo costruire a questo punto una matrice delle immagini $M = [\mathbf{m}_1;...;\mathbf{m}_n] \in \mathbb{R}^{n \times m}$. Per una superficie *Lambertiana*, per l'i-esimo pixel della j-esima immagine vale la relazione

$$\rho_i \boldsymbol{\ell}_j \mathbf{n}_i = m_{ij}$$

con $i = 1, ..., m$ e $j = 1, ..., n$ dove ρ_i è l'*albedo* che è in sostanza il coefficiente di riflessione della superficie, anche definito come prodotto scalare tra normale alla superficie e il vettore che individua la direzione della luce incidente in quel punto. Sotto l'ipotesi di *Lambertianità* esso è costante in tutta la superficie; non lo è altrimenti, e di questo andrà tenuto conto più avanti. Il vettore $\mathbf{n}_i^T$ è il trasposto del vettore normale alla superficie nel dato punto mentre m_{ij} rappresenta l'intensità della radiazione riflessa da quella porzione di superficie quando illuminato nella direzione $\boldsymbol{\ell}_j$. Il tutto può essere espresso in forma matriciale

$$LNR = M \tag{1.2}$$

dove

$$R = diag(\rho_1, ..., \rho_m) \in \mathbb{R}^{m \times m}, \qquad L = [\boldsymbol{\ell}_1; ...; \boldsymbol{\ell}_n] \in \mathbb{R}^{n \times 3}$$
$$N = [\mathbf{n}_1, ..., \mathbf{n}_m] \in \mathbb{R}^{3 \times m}, \qquad M = [\mathbf{m}_1, ..., \mathbf{m}_n] \in \mathbb{R}^{n \times m}$$

Una volta strutturato il problema il passo successivo è estrarre la mappa dei gradienti, per poi utilizzarla nella ricostruzione vera e propria sviluppando un modello adeguato per l'integrazione dei gradienti attraverso un metodo alle differenze finite, in modo da ottenere la superficie che li ha "generati" in un formato facilmente visualizzabile su un qualsiasi computer.

1.2 Non Idealità e Accorgimenti Tecnici

Per quanto riguarda l'illuminazione, affinchè la tecnica funzioni al meglio è preferibile che siano verificate alcune condizioni di ripresa:

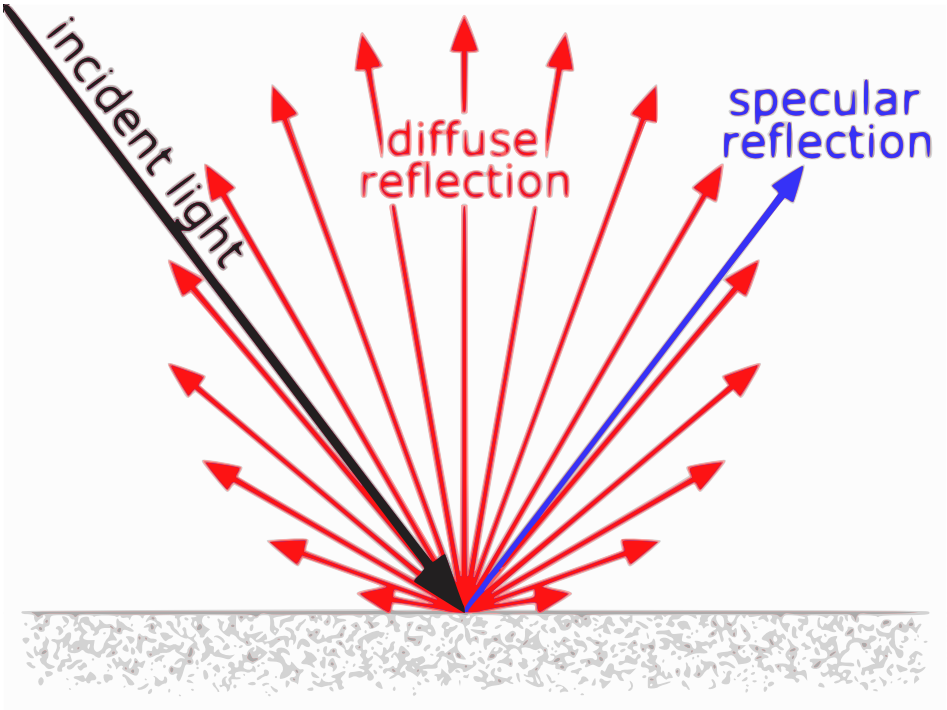


Figura 1.3: Componenti Lambertiane e Speculari

- L'oggetto sia una superficie Lambertiana
- Sia limitato il contributo di luce esterno alle sorgenti utilizzate per il setup sperimentale
- Le sorgenti luminose siano puntiformi e posizionate a distanza sufficiente perchè la superficie sia illuminata uniformemente e i raggi luminosi risultino paralleli

Nella pratica non succede mai che un oggetto sia perfettamente Lambertiano, si hanno spesso delle componenti *speculari*, cioè ci sono delle direzioni in cui la luce riflessa non rispetta il modello di riflessione di Lambert, come illustrato in figura 1.3.

Nelle applicazioni di archeologia abbiamo a che fare con superfici altamente irregolari e uno scenario più probabile è simile a quello in figura 1.4.

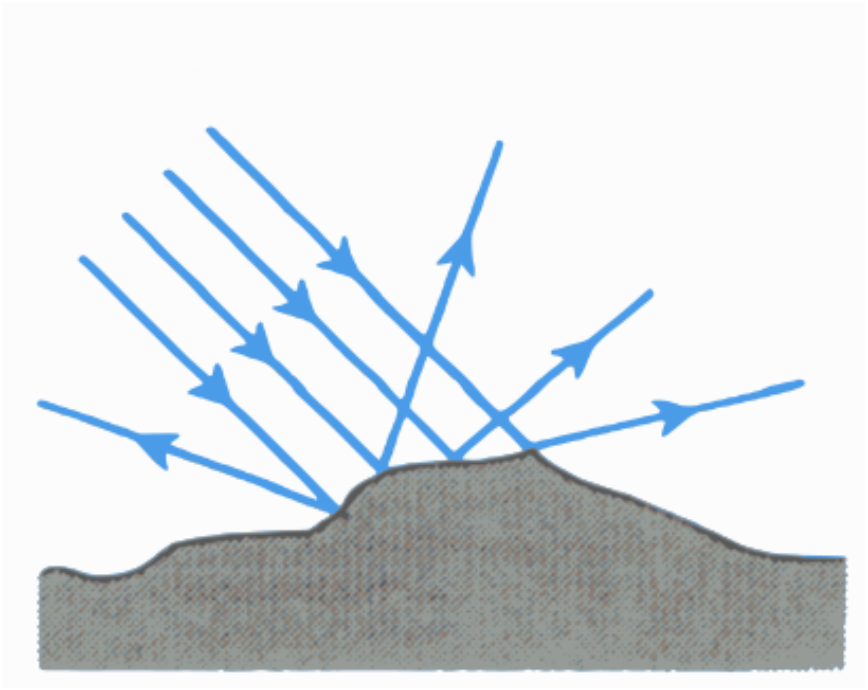


Figura 1.4: Tipica situazione di ripresa

Mentre si riesce a isolare il setup da fonti luminose esterne (anche con metodi più o meno rudimentali, come mostrato in [3]), non è sempre possibile porre le nostre sorgenti a distanza ottimale. Un flash portatile economico genererà tipicamente un fascio conico, che illumina la superficie in modo non uniforme e crea piuttosto delle zone d'ombra i cui corrispondenti pixel avranno valori più bassi rispetto a quello reale. Discorso inverso vale per i pixel corrispondenti alle zone della superficie le cui componenti speculari arrivano al sensore della fotocamera, infatti questi avranno un valore più alto di quello reale. Questo affligge poi anche la qualità della ricostruzione stessa, in quanto questo fenomeno modifica la direzione delle normali spingendole verso la sorgente nel caso di specularità, via dalla sorgente nel caso di un ombra. Le ricostruzioni con eccessive zone d'ombra avranno una componente di distorsione "quasi-sferica" mentre le zone a riflessione speculare risulteranno più piatte di quanto non siano realmente.

Per quanto riguarda invece l'equipaggiamento ottico, è necessario disporre di una fotocamera capace di scattare in modalità manuale, per poter avere controllo sulla luminosità delle immagini; su una qualsiasi fotocamera in

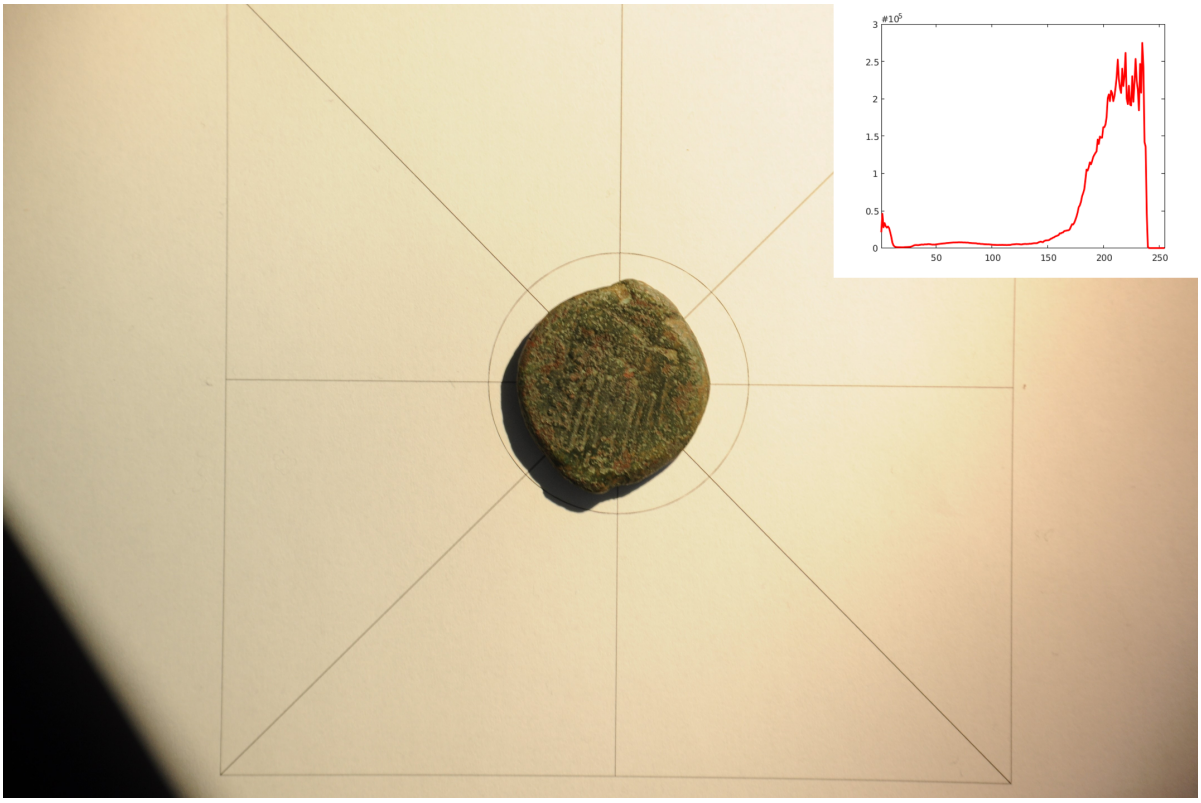


Figura 1.5: Moneta Fenicia e relativo istogramma

commercio (anche entry-level) è possibile, dopo ogni scatto dare uno sguardo all' *istogramma della luminanza*: le ascisse del grafico rappresentano i valori che può assumere ciascun pixel, da 0 (nero) a 255 (bianco), mentre sulle ordinate riporta il numero di pixel che ha quel determinato valore. Il discorso è stato limitato alle immagini in scala di grigio ma è analogo nel caso di immagini a colori, estendo il concetto ai singoli canali (per esempio RGB).

Vogliamo che la maggior parte dei pixel si trovi nella parte sinistra dell'istogramma, cioè che le immagini siano leggermente *sottoesposte*. Ovviamente, quest'affermazione va interpretata con criterio: se lo sfondo è bianco l'istogramma sarà più affollato nella parte destra, ma questo non significa che la foto non sia buona. Un esempio è fornito nella figura 1.5.

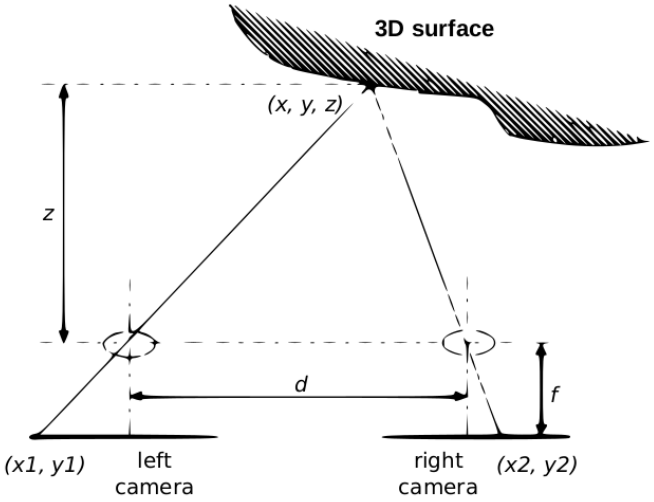
La luminosità delle foto può essere regolata impostando manualmente tempi di scatto, apertura del diaframma e ISO. Aumentando il tempo di scatto più luce riesce ad arrivare ai sensori, il che può essere limitato dall'impo-

stazione sul diaframma: più è piccola l'apertura meno luce attraverserà l'obbiettivo. Tuttavia, fenomeni di diffrazione possono nascere se l'apertura è eccessivamente stretta, creando dei pattern anomali nella luminosità dell'immagine. Infine, è consigliato mantenere dei valori bassi per l'ISO in modo da contenere il rumore che può verificarsi altrimenti. Una trattazione più dettagliata di tutti questi aspetti è fornita in [3].

1.3 Altre tecniche di 3D Imaging

1.3.1 Multiview

In contrapposizione alla Photometric Stereo, questa tecnica richiede che le immagini dell'oggetto siano acquisite da angolazioni differenti. Non si basa quindi su informazioni estratte dalla risposta alle sollecitazioni luminose, ma risolve un problema di *corrispondenza* tra i punti delle immagini disponibili, fondamentalmente mimando il funzionamento dell'occhio umano (che lo fa in maniera specifica con 2 immagini).



La profondità z viene determinata con questa espressione

$$z = \frac{df}{x_1 - x_2}$$

che permette di ricostruire la superficie.

Si può dimostrare che sono sufficienti 2 immagini per avere un problema ben posto. [7] Tuttavia è necessario disporre di un numero maggiore di im-

Figura 1.6: Diversi gradi di ricostruzione di un volto con tecnica Multiview

magini per avere un risultato di buona qualità; con il numero di immagini cresce però anche il carico computazionale poichè dovranno essere calcolate sempre più *mappe di corrispondenza*. Inoltre è necessario scegliere con cura la posizione delle camere perchè piccole imprecisioni possono portare a errori importanti in fase di calcolo.

1.3.2 Laser Scanning

I laser scanner sono degli strumenti molto sofisticati capaci di rilevare la posizione dei punti della superficie in esame. Si delimita l'area da acquisire

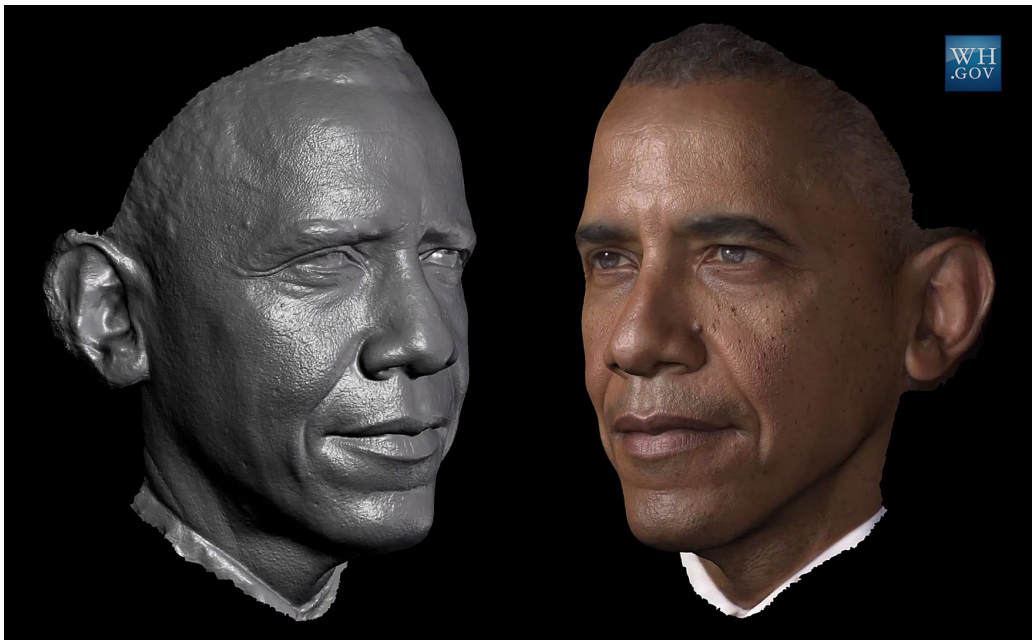


Figura 1.7: Ricostruzione di un volto tramite laser scanning

e si definisce la risoluzione, cioè il numero di punti da rilevare sull'area definita; a questo punto lo strumento raccoglie le sue misurazioni di quota per ogni pezzetto di superficie che è stato definito, ottenendo la superficie voluta, non potendo però "misurare" il colore in alcun modo. Questo si può recuperare integrando nel dispositivo una fotocamera che scatta delle foto con l'obiettivo di sovrapporre il colore al modello 3D creato. Così facendo è però impossibile discernere il reale colore dalle ombre proiettate dallo strumento stesso durante lo scatto delle foto, come è possibile notare in figura 1.7.

Capitolo 2

Strumenti Matematici

2.1 Decomposizione ai Valori Singolari (SVD)

Si supponga $M \in \mathbb{R}^{m \times n}$ con $m > n$: la decomposizione ai valori singolari di M è

$$M = U\Sigma V^T$$

dove

$$U \in \mathbb{R}^{m \times m} \qquad V \in \mathbb{R}^{n \times n}$$

sono matrici ortogonali, cioè $U^T U = I$ e $V^T V = I$, mentre

$$\Sigma = diag[\sigma_1, ..., \sigma_n] \in \mathbb{R}^{m \times n} \qquad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$$

contiene i valori singolari relativi alla matrice M. Si può dimostrare che questi sono la radice quadrata dei valori assoluti degli autovalori della matrice $M^T M$, mentre le colonne di V sono i suoi autovettori. Inoltre

$$MV = U\Sigma V^T V =$$
$$= U\Sigma$$

Tipicamente una delle due dimensioni della matrice M è molto più grande dell'altra e risulta computazionalmente più "economico" calcolare la Decomposizione ai Valori Singolari *ridotta* o *troncata (TSVD)*. Nel caso in

cui $m \gg n$ si ha la decomposizione *'U' troncata* in cui

$$U_n \in \mathbb{R}^{m \times n} \qquad \Sigma \in \mathbb{R}^{n \times n}$$

Questo consente una gestione decisamente migliore della memoria e tempi di computazione significativamente ridotti.
In maniera analoga, nel caso in cui $n \gg m$ (come nel nostro) si ha la decomposizione *'V' troncata* in cui

$$V_m \in \mathbb{R}^{n \times m} \qquad \Sigma \in \mathbb{R}^{m \times m}$$

Si dimostra che la TSVD con fattore di troncamento τ fornisce la migliore approssimazione di rango τ della matrice originale M.
In *Matlab*® [8] la routine per la decomposizione completa viene chiamata con il comando *svd(M)*. Per calcolare la decomposizione troncata è sufficiente aggiungere un parametro *svd(M, 0)*. In questo modo però, il fattore di troncamento non potrà essere scelto manualmente.

2.2 Metodi Iterativi per la Risoluzione di Sistemi Lineari

La strategia di un metodo iterativo per la soluzione di sistemi del tipo

$$A\mathbf{x} = \mathbf{b}$$

è di immaginarla come limite di una successione di vettori soluzione

$$\mathbf{x}_{exact} = \lim_{n \to \infty} \mathbf{x}_{(n)}$$

L'idea generale è di *splittare* la matrice $A = P - M$ con P invertibile. Si parte da una soluzione iniziale $x_{(0)}$ e le successive $x_{(n)}$ si trovano risolvendo

$$\mathbf{x}_{(n+1)} = B\mathbf{x}_{(n)} + \mathbf{f}$$

dove $B = P^{-1}M = I - P^{-1}A$ è detta *matrice di iterazione* e $\mathbf{f} = P^{-1}\mathbf{b}$. Il sistema ora si può anche scrivere come

$$(I - B)\mathbf{x} = \mathbf{f}$$
$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$$

e la matrice P^{-1} viene detta *matrice di precondizionamento* Un metodo converge se il raggio spettrale $\rho(B) < 1$, tanto più velocemente quanto questo si avvicina a zero. Ci accingiamo ora a introdurre dei concetti preliminari alla comprensione del metodo iterativo dei *Gradienti Coniugati Precondizionato*; cominciamo con le forme quadratiche [10].

2.2.1 Forma Quadratica

Una forma quadratica si definisce

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} + c$$

Il gradiente della forma quadratica

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \frac{\partial}{\partial x_2} f(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\mathbf{x}) \end{bmatrix} = \frac{1}{2}A^T\mathbf{x} + \frac{1}{2}A\mathbf{x} - \mathbf{b}$$

è un campo vettoriale che in ogni suo generico punto $\mathbf{x}_p$ è orientato nella direzione di massima crescita di $f(\mathbf{x})$. Ponendo $\nabla f(\mathbf{x}) = 0$, se la matrice A è simmetrica

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = 0$$

si dimostra che la soluzione di $A\mathbf{x} = \mathbf{b}$ è un punto stazionario $f(\mathbf{x})$. Si può dimostrare che se A è anche definita positiva, allora tale soluzione è un minimo. Quindi possiamo trasformare il problema di un sistema lineare in una ricerca di minimo.

2.2.2 Metodo Steepest Descent

Cominciamo con il definire il vettore errore

$$\mathbf{e}_{(i)} = \mathbf{x}_{(i)} - \mathbf{x}$$

e il vettore residuo

$$\begin{aligned}\mathbf{r}_{(i)} &= \mathbf{b} - A\mathbf{x}_{(i)} \\ &= -A\mathbf{e}_{(i)} \\ &= -\nabla f(\mathbf{x}_{(i)})\end{aligned}$$

che come si può notare punta nella direzione opposta rispetto al gradiente, cioè nella direzione di massima decrescita, e questo dovrebbe far intuire il motivo del nome *Steepest Descent, Discesa più Ripida.* Scegliamo a caso un vettore iniziale $\mathbf{x}_{(0)}$ e calcoliamo i successivi con la formula

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{r}_{(i)} \tag{2.1}$$

ovvero ci muoviamo lungo la direzione di ricerca $\mathbf{r}_{(i)}$ sulla quadratica. Dobbiamo scegliere $\alpha_{(i)}$ in modo da minimizzare f lungo questa curva, ovvero

$$\begin{aligned}&\frac{\partial}{\partial\alpha_{(i)}}[f(\mathbf{x}_{(i+1)})] = \\ &= \nabla f(\mathbf{x}_{(i+1)})^T \frac{\partial}{\partial\alpha_{(i)}}\mathbf{x}_{(i+1)} = \\ &= \nabla f(\mathbf{x}_{(i+1)})^T \mathbf{r}_{(i)} = 0\end{aligned}$$

Per determinare gli $\alpha_{(i)}$ sfruttiamo il fatto che $\mathbf{r}_{(i+1)} = -\nabla f(\mathbf{x}_{(i+1)})$ e scriviamo

$$\begin{aligned}&\mathbf{r}_{(i+1)}^T \mathbf{r}_{(i)} = 0 \\ &(\mathbf{b} - A\mathbf{x}_{(i+1)})^T \mathbf{r}_{(i)} = 0 \\ &(\mathbf{b} - A(x_{(i)} + \alpha_{(i)}\mathbf{r}_{(i)}))^T \mathbf{r}_{(i)} = 0 \\ &(\mathbf{b} - A\mathbf{x}_{(i)})^T \mathbf{r}_{(i)} - \alpha_{(i)}(A\mathbf{r}_{(i)})^T \mathbf{r}_{(i)} = 0\end{aligned}$$

$$
\begin{aligned}
(\mathbf{b} - A\mathbf{x}_{(i)})^T \mathbf{r}_{(i)} &= \alpha_{(i)} (A\mathbf{r}_{(i)})^T \mathbf{r}_{(i)} \\
\mathbf{r}_{(i)}^T \mathbf{r}_{(i)} &= \alpha_{(i)} \mathbf{r}_{(i)}^T A\mathbf{r}_{(i)} \\
\alpha_{(i)} &= \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T A\mathbf{r}_{(i)}}
\end{aligned}
$$

E mettendo tutto insieme l'algoritmo si riassume in 3 passaggi

$$\mathbf{r}_{(i)} = \mathbf{b} - A\mathbf{x}_{(i)} \tag{2.2a}$$

$$\alpha_{(i)} = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T A\mathbf{r}_{(i)}} \tag{2.2b}$$

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)} \mathbf{r}_{(i)} \tag{2.2c}$$

Concludiamo notando che possiamo diminuire il carico computazionale aggiungendo una defifinizione ricorsiva per i residui, ovvero premoltiplicando la 2.2c per $-A$ e aggiungendo $\mathbf{b}$ in entrambi i membri ottenendo

$$\mathbf{r}_{(i+1)} = \mathbf{r}_{(i)} + \alpha_{(i)} A\mathbf{r}_{(i)} \tag{2.3}$$

infatti la moltiplicazione $A\mathbf{r}_{(i)}$ che è presente anche nel calcolo degli $\alpha_{(i)}$ in questo modo può essere eseguita una sola volta.

2.2.3 Metodo delle Direzioni Coniugate

Scegliamo un insieme di n direzioni di ricerca tra loro ortogonali $\mathbf{d}_0$,$\mathbf{d}_1$,..., $\mathbf{d}_{(n-1)}$. La differenza con il metodo Steepest Descent, è che mentre a quest'ultimo può capitare di fare 2 passi non consecutivi nella stessa direzione, ogni passo in una determinata direzione compiuto dal metodo CD è della lunghezza giusta, ovvero si giungerà alla soluzione in n passi.

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)} \mathbf{d}_{(i)} \tag{2.4}$$

L'equazione 2.4 è molto simile alla 2.1: stavolta per determinare $\alpha_{(i)}$ sfrutteremo l'ortogonalità fra $\mathbf{e}_{(i+1)}$ e $\mathbf{d}_{(i)}$, cioè fra la distanza che ci separa dalla

soluzione al passo i e la direzione presa nella stesso passo.

$$\mathbf{d}_{(i)}^T \mathbf{e}_{(i+1)} = 0$$
$$\mathbf{d}_{(i)}^T (\mathbf{e}_{(i)} + \alpha_{(i)} \mathbf{d}_{(i)}) = 0$$

e con un procedimento analogo a quello seguito prima troviamo

$$\alpha_{(i)} = -\frac{\mathbf{d}_{(i)}^T \mathbf{e}_{(i)}}{\mathbf{d}_{(i)}^T \mathbf{d}_{(i)}}$$

L'unico problema è che non possiamo conoscere $\mathbf{e}_{(i)}$ a priori, o il problema sarebbe già risolto. La soluzione è sostituire l'insieme iniziale di direzioni di ricerca con uno dove queste sono *A-ortogonali* o *coniugate* (e da qui il nome), cioè per cui valga

$$\mathbf{d}_{(i)}^T A \mathbf{d}_{(j)} = 0$$

Quindi modifichiamo la precedente richiesta di ortogonalità in una di A-ortogonalità

$$\mathbf{d}_{(i)}^T A \mathbf{e}_{(i+1)} = 0$$

e come abbiamo già fatto per lo Steepest Descent, imponiamo $\frac{\partial}{\partial \alpha_{(i)}}[f(\mathbf{x}_{(i+1)})] = 0$ per minimizzare f lungo la direzione $\mathbf{d}_{(i)}$ e otteniamo

$$-\mathbf{r}_{(i+1)} \mathbf{d}_{(i)} = 0 \qquad\qquad \mathbf{d}_{(i)}^T A \mathbf{e}_{(i+1)} = 0$$

ricordandoci che $\mathbf{r}_{(i)} = -A\mathbf{e}_{(i)}$. Otteniamo gli $\alpha_{(i)}$ in un'espressione molto simile a quella ottenuta nel precedente paragrafo

$$\begin{aligned} \alpha_{(i)} &= -\frac{\mathbf{d}_{(i)}^T A \mathbf{e}_{(i)}}{\mathbf{d}_{(i)}^T A \mathbf{d}_{(i)}} \\ &= \frac{\mathbf{d}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{d}_{(i)}^T A \mathbf{d}_{(i)}} \end{aligned}$$

Ora dobbiamo però trovare un metodo operativo per generare effettivamente le direzioni A-ortogonali.

2.2.4 Coniugazione di Gram-Schmidt

Scegliamo n vettori linearmente indipendenti $\mathbf{u}_0$,$\mathbf{u}_1$,...,$\mathbf{u}_{n-1}$ e assegnamo il primo $\mathbf{d}_{(0)} = \mathbf{u}_0$; per costruire i successivi $\mathbf{d}_{(i)}$ l'idea è di prendere $\mathbf{u}_i$ e sottrargli tutto ciò che non è A-ortogonale alle direzioni costruite in precedenza

$$\mathbf{d}_{(i)} = \mathbf{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_{(k)}$$

dove i β_{ik} sono evidentemente definiti per $i > k$. Per trovarli premoltiplichiamo i due membri dell'equazione in questo modo

$$\begin{aligned} \mathbf{d}_{(i)}^T A \mathbf{d}_{(j)} &= \mathbf{u}_i^T A \mathbf{d}_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_{(k)}^T A \mathbf{d}_{(j)} \\ 0 &= \mathbf{u}_i^T A \mathbf{d}_{(j)} + \beta_{ij} \mathbf{d}_{(j)}^T A \mathbf{d}_{(j)}, \qquad\qquad i > j \end{aligned}$$

dove nel secondo passaggio abbiamo sfruttato la A-ortogonalità delle direzioni di ricerca, e troviamo i $\beta_{(ij)}$ come

$$\beta_{(ij)} = -\frac{\mathbf{u}_i^T A \mathbf{d}_{(j)}}{\mathbf{d}_{(j)}^T A \mathbf{d}_{(j)}} \tag{2.5}$$

La computazione così formulata ha complessità $O(n^3)$ e inoltre per la costruzione di ogni direzione bisogna mantenere in memoria tutti i precedenti, rischiando di andare in overflow per sistemi di grandi dimensioni.

2.2.5 Metodo dei Gradienti Coniugati

Il metodo dei Gradienti Coniugati è analogo alle Direzioni Coniugate, ma con la particolarità nella scelta dei vettori $\mathbf{u}_i = \mathbf{r}_{(i)}$, e per questo è così denominato (ricordiamo $\nabla f(\mathbf{x}_{(i)}) = -\mathbf{r}_{(i)}$). I residui hanno alcune utili proprietà: ogni residuo è ortogonale alle precedenti direzioni di ricerca, ma anche ai precedenti residui stessi. Ma si può dimostrarne una ancora più interessante: il residuo $\mathbf{r}_{(i+1)}$ è A-ortogonale a tutte le direzione precedenti tranne l'ultima al passo i, e questo rende la coniugazione di Gram-Schmidt

molto più semplice a leggera a livello computazionale. L'ortogonalità fra i residui impone

$$\mathbf{r}_{(i)}^T \mathbf{r}_{(j)} = 0, \qquad\qquad i \neq j$$

e combinando con la 2.3 possiamo scrivere

$$\begin{aligned}
&\mathbf{r}_{(i)}^T \mathbf{r}_{(j+1)} = \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} - \alpha_{(j)} \mathbf{r}_{(i)}^T A \mathbf{r}_{(j)} \\
&\alpha_{(j)} \mathbf{r}_{(i)}^T A \mathbf{r}_{(j)} = \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} - \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} \\
&\mathbf{r}_{(i)}^T A \mathbf{r}_{(j)} = -\frac{1}{\alpha_{(i-1)}} \mathbf{r}_{(i)}^T \mathbf{r}_{(i)}, \qquad\qquad i = j+1
\end{aligned}$$

dove si è specificato $i = j+1$ per sottolineare che per $i \geq j+1$ si ottengono soltanto zeri, mentre per $i \leq j+1$ non sono definiti i β_{ij} necessari per la coniugazione. Ne segue quindi ricordando la 2.5

$$\beta_{(i,i-1)} = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\alpha_{(i-1)} \mathbf{r}_{(i-1)}^T A \mathbf{r}_{(i-1)}}$$

e ricordando la definizione degli $alpha_{(i)}$ semplifichiamo

$$\beta_{i,i-1} = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i-1)}^T \mathbf{r}_{(i-1)}}$$

Unendo i tasselli

$$\begin{aligned}
&\mathbf{r}_{(i)} = \mathbf{b} - A\mathbf{x}_{(i)} \\
&\alpha_{(i)} = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T A \mathbf{r}_{(i)}} \\
&\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)} \mathbf{r}_{(i)} \\
&\mathbf{r}_{(i+1)} = \mathbf{r}_{(i)} - \alpha_{(i)} A \mathbf{r}_{(i)} \\
&\beta_{i+1,i} = \frac{\mathbf{r}_{(i+1)}^T \mathbf{r}_{(i+1)}}{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}} \\
&\mathbf{r}_{(i+1)} = \mathbf{r}_{(i+1)} + \beta_{i+1,i} \mathbf{r}_{(i)}
\end{aligned}$$

In *Matlab* la chiamata è x=*pcg*(A,b,tolerance,maxiter,P1,P2), dove "tolerance" e "maxiter" sono necessari per decidere quando fermare le iterazioni, P1 e P2 delle matrici di *precondizionamento*. Infine, si dimostra che la velocità di convergenza del metodo è una funzione del *numero di condizionamento spettrale* $\kappa_2(A)$ definito come il rapporto fra il più grande e il più piccolo degli autovalori della matrice del sistema. In particolare, raggiunge il massimo della velocità quando $\kappa_2(A) \approx 1$ e rallenta invece con la crescita di $\kappa_2(A)$. In questo caso sarà necessario *precondizionare* il sistema.

2.2.6 Precondizionamento

Precondizionare un sistema significa, come suggerisce il termine, migliorarne il numero di condizionamento in modo da poterlo risovere più agevolmente. In formule, significa trasformare il sistema $A\mathbf{x} = \mathbf{b}$ in uno equivalente

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$$

in cui la matrice P sia più facile da invertire rispetto ad A, e la matrice $P^{-1}A$ abbia ovviamente un condizionamento migliore del sistema di partenza. Il problema sorge nel nostro caso, perchè il fatto che P e A siano entrambe simmetriche e definite positive non garantisce che il prodotto $P^{-1}A$ lo sia.
Si supera notando che avendo a disposizione P simmetrica e definita positiva, esiste almeno una fattorizzazione di $P = EE^T$ (per esempio la *fattorizzazione di Cholesky*). Si può dimostrare che le matrici $P^{-1}M$ e $E^{-1}AE^{-(T)}$ hanno gli stessi autovalori, con la garanzia che quest'ultima sia simmetrica e definita positiva. Quindi il sistema si può trasformare in

$$E^{-1}AE^{-(T)}\tilde{\mathbf{x}} = E^{-1}\mathbf{b}$$

che si può quindi risolvere con il metodo CG per $\tilde{\mathbf{x}}$, ed è sufficiente usare poi $\mathbf{x} = E^{-(T)}\tilde{\mathbf{x}}$ per trovare la soluzione originale.

2.2.7 Fattorizzazione di Cholesky

Questa fattorizzazione trasforma la matrice A simmetrica e definita positiva nel prodotto di due matrici trinagolari (superiori o inferiori) tali che

$$U^T U = A \qquad LL^T = A \tag{2.6}$$

con U triangolare superiore ed L triangolare inferiore.
Esiste una variante dell'algoritmo (*fattorizzazione di cholesky incompleta*), in cui la differenza è molto semplice: le matrici U o L sono "costrette" a mantenere lo stesso "pattern" di zeri della matrice A. Questo è ovviamente di grande utilità quando si ha a che fare con matrici sparse, poichè consente di calcolare effettivamente soltanto gli elementi delle matrici U o L nelle posizioni in cui gli elementi di A sono diversi da 0, riducendo i tempi di computazione.

È possibile su *Matlab* chiamare questa routine con un' espressione del

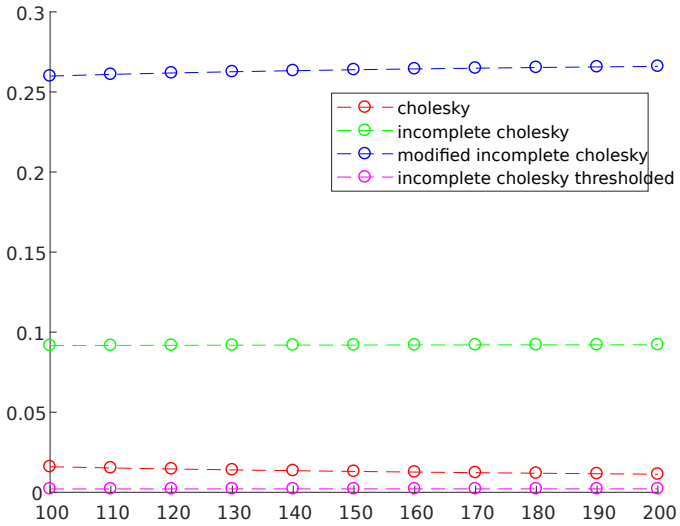


Figura 2.1: $norm(A - LL^T)/norm(A)$ in funzione della dimensione di A, $(threshold = 10^{-3})$

tipo "*C=ichol(A)*" (che di default produce una matrice triangolare inferiore; per richiederne una superiore basta aggiungere il parametro *'upper'*); si possono anche aggiungere altri parametri per modificare ulteriormente l'algoritmo. Il fatto di eliminare dal risultato della fattorizzazione alcuni

elementi come specificato in precedenza può compromettere la veridicità delle equazioni 2.6. A questo si può cercare di ovviare in 2 modi: impostando una soglia di "drop" *C=ichol(A,struct('type','ict','droptol',threshold))*, dove più basso *threshold*, migliore l'approssimazione, oppure apportando una compensazione sulla diagonale *C=ichol(A,struct('michol','on'))*, cioè utilizzando il *Modified Incomplete Cholesky*.

2.3 Problemi ai Minimi Quadrati

Questa denominazione si riferisce a una classe di problemi in cui un sistema lineare

$$A\mathbf{x} = \mathbf{b}$$

con $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$ e $\mathbf{b} \in \mathbb{R}^m$, ha la particolarità che $m > n$, cioè ha più equazioni che incognite. Nel caso in cui gli elementi di $\mathbf{b}$ siano dati sperimentali (quindi affetti da errore), potrebbe non essere più possibile trovare la soluzione esatta. Per questo si imbocca un altra strada: trovare il vettore $\mathbf{x}$ soluzione tale che la norma del residuo sia minima

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2$$

Ovviamente nel caso (senza rumore) in cui $m = n$ questa espressione è certamente uguale a zero.

2.3.1 Metodo delle Equazioni Normali

La quantità $\|A\mathbf{x}-\mathbf{b}\|_2$ è certamente non negativa, e si può minimizzarne il quadrato imponendo il gradiente uguale a zero

$$\begin{aligned}\nabla(\|A\mathbf{x} - \mathbf{b}\|_2^2) = \nabla((A\mathbf{x} - \mathbf{b})^T(A\mathbf{x} - \mathbf{b})) = \\ = 2A^TA\mathbf{x} - 2A^T\mathbf{b} = 0\end{aligned}$$

da cui si ottiene il *sistema normale*

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

Se A ha rango pieno la soluzione è data da

$$\mathbf{x} = A^\dagger\mathbf{b}$$

dove $A^\dagger = (A^T A)^{-1}A^T$ è la cosiddetta *pseudoinversa* o *inversa di Moore-Penrose* di A. Uno svantaggio di questo metodo è il condizionamento della matrice $(A^T A)$ che è pari al doppio di quello di A.

2.3.2 Fattorizzazione QR

Questa fattorizzazione consiste nel riscrivere una matrice $A \in \mathbb{R}^{m\times n}$ come prodotto di una matrice $Q \in \mathbb{R}^{m\times m}$ ortogonale, e una matrice $R \in \mathbb{R}^{m\times n}$ triangolare superiore. La soluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ diventa

$$\begin{cases} Q\mathbf{c} = \mathbf{b} \\ R\mathbf{x} = \mathbf{c} \end{cases}$$

Si trova $\mathbf{c} = Q^T\mathbf{b}$ con una semplice operazione di trasposizione e di conseguenza $\mathbf{x}$ per sostituzione all'indietro. In un problema ai minimi quadrati, avendo fattorizzato $A = QR$

$$\begin{aligned} \|A\mathbf{x} - \mathbf{b}\|_2^2 &= \|Q(R\mathbf{x} - Q^T\mathbf{b})\|_2^2 = \\ &= \|R\mathbf{x} - \mathbf{c}\|_2^2 \end{aligned}$$

il che è ammissibile perchè una matrice ortogonale non modifica la *norma-2* di un vettore; proprio per questa ragione è immediato notare come il condizionamento di R (rispetto alla *norma-2*) è identico a quello di A. Ora è possibile risolvere il sistema quadrato di dimensioni $n \times n$

$$\widetilde{R}\mathbf{x} = \tilde{\mathbf{c}}$$

avendo inserito i primi n elementi di $\mathbf{c}$ in $\tilde{\mathbf{c}}$ e le prime n righe di R in $\widetilde{R}$, ed è semplice dimostrare che se i restanti $m-n$ elementi nel termine noto sono uguali a zero allora $\mathbf{x}$ è la soluzione esatta del sistema, altrimenti lo è nel senso dei minimi quadrati e $\left\| \mathbf{c} - \begin{bmatrix} \tilde{\mathbf{c}} \\ 0 \end{bmatrix} \right\|_2^2$ dà una stima del residuo.

2.4 Equazione di Poisson

È una PDE (Equazione alle Derivate Parziali, dall'inglese *Partial Differential Equation*) che, nel caso di funzioni a due variabili, ha questa forma

$$\Delta f(x,y) = g(x,y)$$

dove $\Delta = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$ è l'operatore *Laplaciano*.
Questa trova innumerevoli applicazioni in fisica (elettrostatica, gravità etc.), e nel nostro caso anche in problemi di Computer Vision, infatti la $f(x,y) = z$ rappresenterà la quota della superficie in funzione del dominio (x,y) delle immagini, mentre la $g(x,y)$ sarà ricavata a partire dalle informazioni che riusciremo a estrarre sui gradienti punto per punto.

2.5 Metodi alle Differenze Finite

L'idea alla base di un *FDM (Finite Differences Method)* è quella di discretizzare un'equazione differenziale con l'obiettivo di approssimarne la soluzione risolvendo un sistema lineare che nasce dalla discretizzazione stessa.
Si supponga di avere una funzione $f(x) : \mathbb{R} \to \mathbb{R}$ derivabile un numero infinito di volte, di cui si vogliono approssimare le derivate. Tale $f(x)$ si

può sviluppare in serie di Taylor intorno a un punto x_0

$$f(x) = \sum_{n}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n = \\ = f(x_0) + \frac{f^{(1)}(x_0)}{1!}(x - x_0) + \frac{f^{(2)}(x_0)}{2!}(x - x_0)^2 + ...$$

Definiamo $h = x - x_0$ come *passo di integrazione* e tronchiamo la serie

$$f(x_0 + h) = f(x_0) + f^{(1)}(x_0)h + O(h^2)$$

e risolvendo per $f^{(1)}(x_0)$

$$f^{(1)}(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} + O(h) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

si ottiene nient'altro che la definizione del rapporto incrementale.
Per risolvere il problema su una macchina, dobbiamo ora trovare i punti che "fittano" meglio la soluzione. Al passo i possiamo avere

$$(\frac{\partial f}{\partial x})_{+i} = \frac{f_{i+1} - f_i}{h} \qquad \text{oppure} \\ (\frac{\partial f}{\partial x})_{-i} = \frac{f_i - f_{i-1}}{h}$$

rispettivamente fitting *in avanti* e *all'indietro*. Si può anche fare un fitting *centrale* operando una "media" dei due

$$(\frac{\partial f}{\partial x})_{\pm i} = \frac{f_{i+1} - f_{i-1}}{2h}$$

E derivando ulteriormente otteniamo un'approssimazione di grado superiore

$$[\frac{\partial f}{\partial x}(\frac{\partial f}{\partial x})]_{\pm i} = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \tag{2.7}$$

2.5.1 Applicazione: Equazione di Poisson

Vogliamo ora discretizzare il problema e applicare delle condizioni al contorno (cioè vogliamo risolvere un *Problema di Cauchy*) in modo da poter ottenere una soluzione unica: scegliamo la *Condizione di Dirichlet*,

cioè imponiamo un valore di f agli estremi del dominio

$$y(x) = f(x) \qquad\qquad \forall x \in \partial\Omega$$

dove $\partial\Omega$ rappresenta la frontiera del dominio. Passiamo al caso in due dimensioni per semplicità, e supponiamo di trovarci in un dominio discreto e rettangolare (come per esempio un'immagine) dato dall'unione di due domini monodimensionali $\Omega = \{x \in [\alpha, \beta] \cup y \in [\gamma, \delta]\}$. Senza perdere di generalità, si possono porre $\alpha = \gamma = 0$ ed anche $\beta = \delta = \varepsilon$, consentendoci di trattare il caso di un dominio quadrato di lato ε nel primo quadrante. In generale, l'espressione che otteniamo è

$$\begin{cases} \frac{\partial^2}{\partial x_i} f_{i,j} + \frac{\partial^2}{\partial y_j} f_{i,j} = g_{i,j} \\ f_{0,j} = f_{\varepsilon,j} = f_{i,0} = f_{i,\varepsilon} = \varrho \end{cases} \qquad \forall (i,j) \in [0, \varepsilon] \qquad (2.8)$$

dove stavolta abbiamo aggiunto il pedice j per indicare la discretizzazione delle y_j, e con $\varrho \in \mathbb{R}$ costante. Ora ricordando la 2.7 e sostituendo nella 2.8

$$\begin{aligned} \frac{\partial^2}{\partial x_i} f_{i,j} + \frac{\partial^2}{\partial y_j} f_{i,j} &= \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{h^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{h^2} \\ &= \frac{f_{i+1,j} + f_{i-1,j} - 4f_{i,j} + f_{i,j+1} + f_{i,j-1}}{h^2} \end{aligned}$$

e a questo punto possiamo risolvere il problema di Cauchy calcolando la soluzione del sistema lineare $A\mathbf{f} = \mathbf{g}$ dove A è la *Matrice di Poisson*, mentre $\mathbf{f}$ e $\mathbf{g}$ sono rispettivamente i vettori contenenti la funzione soluzione e il termine noto.

La Matrice di Poisson è del tipo *Block Toeplitz with Toeplitz Blocks*, cioè *Toeplitz a Blocchi con Blocchi di Toeplitz*. Una matrice di Toeplitz ha tutte le diagonali discendenti da sinistra a destra a elementi costanti. La matrice

A è così costruita

$$A = \begin{bmatrix} T & I & 0 & \cdots & 0 \\ I & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & I \\ 0 & \cdots & 0 & I & T \end{bmatrix}$$

dove T e I sono i blocchi costituenti la Matrice di Poisson; A ha delle "pseudodiagonali" a blocchi costanti, e sua volta il blocco

$$T = \begin{bmatrix} -4 & 1 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -4 \end{bmatrix}$$

è evidentemente anch'esso una matrice di Toeplitz, mentre I è proprio la matrice identità. La matrice A ha dimensioni $n^2 \times n^2$ mentre i suoi blocchi $n \times n$. È triviale notare che la Matrice di Poisson è simmetrica, mentre si può dimostrare che è definita positiva.

In *Matlab* esiste un'apposita routine per costruirla: *A=gallery('poisson',n)*.

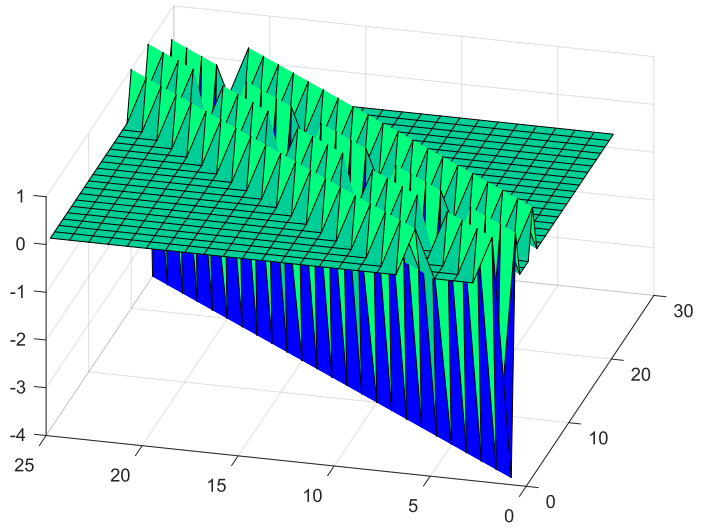


Figura 2.2: Una rappresentazione grafica della Matrice di Poisson

Capitolo 3

L'algoritmo

3.1 Determinazione dei Gradienti

In questo paragrafo ripartiamo dalla equazione 1.2 per risolvere il problema della determinazione dei gradienti sulla superficie punto per punto. Possiamo modellare la nostra superficie come una funzione in due variabili $f(x, y) = z$ e scrivere i gradienti nelle direzioni (x, y) come

$$p = \frac{\partial z}{\partial x} \qquad q = \frac{\partial z}{\partial y} \tag{3.1}$$

e per ogni punto generico $P = (x_P, y_P)$ possiamo definire dei vettori tangenti alla superficie e tra loro perpendicolari $\mathbf{t}_x = [1, 0, p]$ e $\mathbf{t}_y = [0, 1, q]$ tali che il loro prodotto vettoriale $\mathbf{n} = \mathbf{t}_x \times \mathbf{t}_y$ rappresenti la normale nel punto (x_P, y_P)

$$\mathbf{n} = [-p, -q, 1]$$

Ora, a noi interessa soltanto la direzione della normale e non la sua intensità quindi possiamo normalizzare

$$\mathbf{n} = \frac{[-p, -q, 1]}{\sqrt{1 + p^2 + q^2}} \tag{3.2}$$

Con questa premessa possiamo passare ad analizzare il caso in cui la matrice L non sia nota. Tale approccio è noto nella letteratura come *uncon-strained* o *unknown light directions*. Questa metodologia consente di non

doversi preoccupare di misurare in maniera precisa la posizione delle luci, semplificando enormemente la procedura.
Un grande passo avanti in questa direzione fu dato da [9]. L'articolo parte dall'assunto che ogni punto della superficie può ricevere luce da tutte le direzioni. Ma l'insieme di luci ricevute in ciascuna direzione può variare da punto a punto a seconda della specifica orientazione della superficie. Supponiamo che la luce arrivi in direzione $\boldsymbol{\ell}(\theta_\ell, \varphi_\ell)$, mentre venga riflessa in direzione $\mathbf{r}(\theta_r, \varphi_r)$ dove i θ e φ sono le coordinate angolari di sfere unitarie. La luce riflessa da ciascun punto sarà il risultato di una somma di contributi da tutte le direzioni di incidenza. Per questo scriviamo un integrale di convoluzione su una sfera S di raggio unitario

$$\mathbf{r}(\theta_r, \varphi_r) = \int_S \kappa(\boldsymbol{\ell}, \mathbf{r}) \boldsymbol{\ell}(\theta_\ell, \varphi_\ell) \, d\theta_\ell \, d\varphi_\ell$$

che lega la riflettanza dell'*i-esimo* pixel alle componenti direzionali della luce incidente tramite una *matrice di convoluzione* o *nucleo* (dall'inglese *kernel*) di tipo *'half-cosine'* così definito

$$\kappa(\boldsymbol{\ell}, \mathbf{r}) = max(\frac{\boldsymbol{\ell}^T \mathbf{r}}{\|\boldsymbol{\ell}\| \|\mathbf{r}\|}, 0)$$

che è essenzialmente un nucleo *coseno* ma con la particolarità che riporta a zero tutte le componenti con segno negativo; si mostra che questo si comporta come un filtro passa-basso, quindi effettivamente la convoluzione con questo nucleo restituisce soltanto il contributo in bassa frequenza della luce incidente.
L'integrale non può essere risolto in modo esatto ma può appunto essere scomposto in serie mediante armoniche sferiche. Ci limiteremo a una scomposizione del primo ordine, come già riportato in [4].
Partiamo da

$$M = LS_4$$

dove la matrice $L \in \mathbb{R}^{n\times 4}$ contiene le componenti in bassa frequenza della luce incidente, mentre $S_4 \in \mathbb{R}^{4\times m}$ contiene le prime quattro componenti armoniche (*harmonic images*) conseguenti allo sviluppo in serie dell'integrale di convoluzione. Il pedice sta a indicare che lo sviluppo in serie al primo ordine ci porta a lavorare in uno spazio 4D, dove le dimensioni rappresentano effettivamente l'albedo, e le tre componenti spaziali della normale alla superficie.
Il passo successivo è l'applicazione della SVD alla matrice M

$$M = U\Sigma V^T$$

che ci restituisce in Σ le n *principal components* di M. Ricordandoci le proprietà della SVD

$$L = U\sqrt{\Sigma} \qquad S_4 = \sqrt{\Sigma}V^T$$

possiamo applicare tranquillamente la radice quadrata alla matrice dei valori singolari. Ricordandoci anche della particolare struttura della nostra matrice possiamo invece chiamare in causa la TSVD e utilizzare un fattore di troncamento proprio pari a 4 (in realtà il comando *svd(M′, 0)*) non consente di scegliere il fattore di troncamento ma è sufficiente ridimensionare opportunamente le matrici successivamente). Questa scelta è giustificata dal fatto che lo spazio 4D dato dalla TSVD (senza considerare ancora il rumore) fornisce l'approssimazione migliore di rango 4 rispetto alla matrice originale delle immagini. La TSVD ci dà

$$M \approx U_4\Sigma_4 V_4^T$$

Ma allora

$$M \approx \tilde{L}\tilde{S}_4$$

dove $\tilde{L} = U_4\sqrt{\Sigma_4}$ e $\tilde{S}_4 = \sqrt{\Sigma_4}V_4^T$. Questa decomposizione non è unica a meno di una trasformazione lineare

$$M = \tilde{L}A^{-1}A\tilde{S}_4 = LS_4$$

detta *trasformazione di Lorentz*, dove la matrice $A \in \mathbb{R}^{4\times4}$ non singolare, viene detta *matrice di ambiguità*.
Notiamo che ogni colonna $\mathbf{s}= [s_1, s_2, s_3, s_4]^T$ in S_4 soddisfa la relazione $s_1^2 = s_2^2 + s_3^2 + s_4^2$ in quanto, come affermato in precedenza, questi elementi sono rispettivamente l'albedo e le tre componenti delle normali.

$$\mathbf{s}^T J\mathbf{s} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = 0 \tag{3.3}$$

Questo vale per S_4 e non per $\tilde{S}_4$, ma possiamo scrivere una relazione che ci permetta di estendere questo vincolo

$$\mathbf{s} = A\mathbf{q}$$

cioè una trasformazione lineare che riporti effettivamente i punti rappresentati dalle colonne di $\tilde{S}_4$ a giacere sulla sfera unitaria, dove $\mathbf{q}$ è una qualunque colonna di $\tilde{S}_4$. Sostituendo nella 3.3 si ottiene

$$\mathbf{q}^T A^T JA\mathbf{q} = 0$$

In un altro passo ancora, definiamo una matrice $B = A^T JA \in \mathbb{R}^{4\times4}$ tale che

$$\mathbf{q}^T B\mathbf{q} = 0 \tag{3.4}$$

B è simmetrica per costruzione e per questo abbiamo bisogno di 10 elementi anzichè 16 per determinarla (i 4 della diagonale più uno dei 2 triangoli superiore o inferiore). Costruiamo il vettore incognita $\mathbf{b}$ che contiene gli

elementi di B

$$\mathbf{b} = [b_{11}, ..., b_{44}, b_{12}, ..., b_{34}]^T$$

Possiamo sfruttare 3.4 per scrivere un sistema di equazioni a partire da $\widetilde{S}_4$

$$Q\mathbf{b} = 0$$

in cui $Q \in \mathbb{R}^{m \times 10}$ e le sue righe sono così definite $(q_{11}^2, ..., q_{44}^2, 2q_1q_2, ..., 2q_3q_4)$. Da questa relazione troviamo le componenti di B a meno di un fattore di scala

$$\widetilde{B} = \beta B = \beta A^T J A$$

con $\beta \neq 0$ scalare costante. Si può ora sfruttare nuovamente la proprietà di simmetria della matrice B, che trivialmente vale anche per $\widetilde{B}$: tutti i suoi autovalori saranno reali e con una particolarità nel loro segno: uno di essi differirà nel segno dagli altri tre (uno negativo e gli altri positivi o viceversa). Questo perchè

$$\begin{aligned} det(\widetilde{B}) &= \beta det(A^T J A) = \\ &= \beta^4 (-1) det^2 A = \\ &= -\beta^4 det^2 A \end{aligned}$$

Essendo la matrice A non singolare per ipotesi, e poichè $\beta \neq 0$ questa è una quantità sempre negativa e poichè il determinante è effettivamente il prodotto degli autovalori, l'affermazione precedente segue logicamente. Detto questo, possiamo pensare di decomporre ulteriormente B con una fattorizzazione agli autovettori

$$B = W J \Lambda W^T$$

in cui le colonne di W sono gli autovettori di B mentre Λ contiene i valori assoluti dei suoi autovalori. Troviamo finalmente la matrice di ambiguità $A = \sqrt{\Lambda} W^T$ e con essa $L = \tilde{L} A^{-1}$ ed $S_4 = A\widetilde{S}_4$.

Come affermato in precedenza, le 4 righe di S_4 rappresentano in ordine l'albedo e le tre componenti delle normali negli m punti della superficie. Segue che

$$\begin{cases} s_1 = \rho \\ s_2 = N_x \\ s_3 = N_y \\ s_4 = N_z \end{cases}$$

ma ricordando la 3.2

$$\begin{cases} s_1 = \rho \\ s_2 = \frac{-p}{\sqrt{1+p^2+q^2}} \\ s_3 = \frac{-q}{\sqrt{1+p^2+q^2}} \\ s_4 = \frac{1}{\sqrt{1+p^2+q^2}} \end{cases}$$

e risolvendo per p, q si ottiene

$$p = -\frac{s_2}{s_4} \qquad q = -\frac{s_3}{s_4}$$

Un'implementazione di tutto questo procedimento è riportata in [4]; tuttavia, se si fa riferimento alla 1.2 ci si aspetta che la matrice delle immagini abbia al più $rank(M) = 3$ e questo è tanto più vero quanto più le immagini rispettano il modello proposto (questi aspetti verranno trattati più avanti). A tal proposito si è trovato numericamente e senza ancora nessuna dimostrazione formale, che ponendo

$$L = V_3 \qquad N = U_3\Sigma_3$$

dove il significato del pedice indica proprio che si è applicata una TSVD con fattore di troncamento pari a 3, si ottiene una buona approssimazione dei gradienti estraendoli direttamente dalla seconda e terza colonna della matrice N, nell'ipotesi di avere posizionato le luci con un pattern regolare, e avendo mantenuto un'inclinazione costante rispetto al piano in cui giace

l'oggetto. [2]
Utilizzare un numero maggiore di immagini apporta benefici in termini di addolcimento del rumore, riducendo anche la distorsione delle normali, fornendo una migliore approssimazione. C'è un unico svantaggio nello scegliere questo approccio: i gradienti sono determinati a meno del segno e della direzione, cioè non sappiamo a prescindere quale sia il gradiente $\frac{\partial z}{\partial x}$ e quale invece $\frac{\partial z}{\partial y}$. Questo genera quindi un'ambiguità che per ora riusciamo a fugare soltanto con il metodo "brute force", cioè ricostruendo la superficie (con le modalità che presto verranno esposte) provando una per una le otto combinazioni di gradienti possibili. Per questo è consigliabile, durante la "elaborazione di prova", ridurre con un fattore di scala la dimensione delle immagini in modo da abbassare i tempi di calcolo in questa fase. Una volta stabilita la combinazione giusta si ripete l'elaborazione utilizzando un adeguato fattore di scala (deve essere un buon compromesso fra qualità della ricostruzione e prestazioni della macchina, quindi questa scelta sta all'utente). Si è infine trovato che acquisizioni di oggetti diversi effettuate con la stessa configurazione di illuminazione producono la stessa combinazione di gradienti.

3.1.1 Albedo

Finora non abbiamo fatto cenno all'albedo. In questa fase dell'algoritmo abbiamo effettivamente ricavato la matrice L che contiene le direzioni di incidenza della luce. Torniamo per un attimo indietro alla 3.2: abbiamo esattamente $m = row * col$ sistemi lineari da risolvere

$$\rho_j L\mathbf{n}_j = \mathbf{i}_j \quad j = 1, ..., m$$

dove gli $\mathbf{n}_j \in \mathbb{R}^3$ sono i vettori normali punto per punto, mentre gli $\mathbf{m}_j \in \mathbb{R}^n$ sono vettori contenenti i valori dei pixel corrispondenti nelle n immagini

acquisite, mentre $L \in \mathbb{R}^{n\times 3}$. Risolviamo

$$L^{-1}\mathbf{m}_j = \rho_j \mathbf{n}_j = \mathbf{t}_j$$

e si ricavano i gradienti e l'albedo come

$$p_j = -\frac{t_{1j}}{t_{3j}}, \quad q_j = -\frac{t_{2j}}{t_{3j}}, \quad \rho_j = \sqrt{t_{1j}^2 + t_{2j}^2 + t_{3j}^2}$$

anche se a noi interessa effettivamente soltanto l'albedo e le normali, che troviamo tramite la 3.2. In realtà per quanto riguarda l'albedo, quello che facciamo è ricavarlo per ciascun canale quindi

$$\begin{cases} L^{-1}\mathbf{m}_{Rj} = \rho_{Rj}\mathbf{n}_j \\ L^{-1}\mathbf{m}_{Gj} = \rho_{Gj}\mathbf{n}_j \\ L^{-1}\mathbf{m}_{Bj} = \rho_{Bj}\mathbf{n}_j \end{cases}$$

Chiaramente questo è solamente possibile nel caso $n = 3$ poichè altrimenti l'operazione di inversione non ha significato. Tuttavia se $n > 3$ il problema è sovradeterminato si può risovere utilizzando il metodo della *fattorizzazione QR*

$$\begin{aligned} T &= (QR)^{-1}M \\ &= R^{-1}Q^T M \end{aligned}$$

con $T \in \mathbb{R}^{3xm}$ da cui possiamo ricavare gradienti a albedo punto per punto per ciascun canale di luce. In *Matlab* è sufficiente usare l'operatore *backslash* $T = R\backslash(Q^T M)$, che sceglie automaticamente il metodo migliore per la risoluzione.

Per dare un'idea dei tempi di calcolo riportiamo in figura 3.1 i risultati di una prova effettuata su una macchina con architettura AMD, processore A10 + 8GB RAM, utilizzando quattro immagini e variandone la dimensione fino a 1000×1000.

Proviamo a riassumere questa "macrofase" del procedimento:

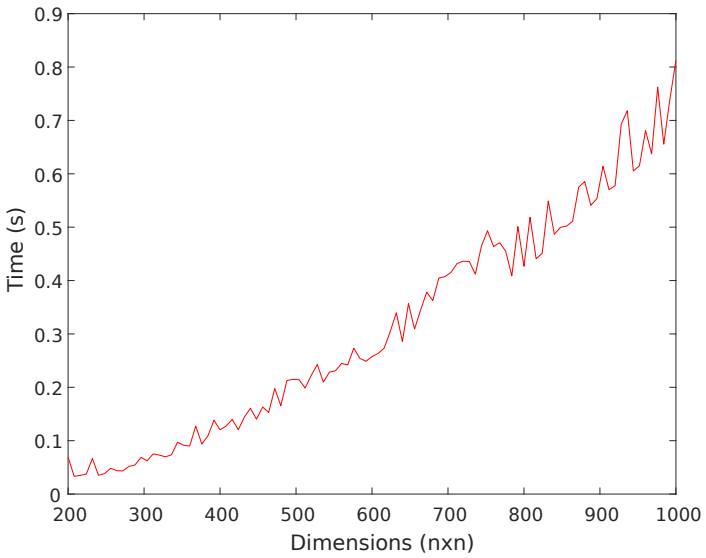


Figura 3.1: Tempi di elaborazione di normali, gradienti e albedo in funzione della dimensione delle immagini

1. Costruiamo la matrice delle immagini $M = [\mathbf{m}_1, ..., \mathbf{m}_n] \in \mathbb{R}^{n \times m}$;
2. Applichiamo la TSVD a $M \approx U_3 \Sigma_3 V_3^T$
3. Poniamo $L = V_3 P$ e $N = U_3 \Sigma_3$;
4. Estraiamo i gradienti $\mathbf{p}, \mathbf{q}$ da N
5. Tramite L si possono ricavare le normali e l'albedo
6. Stabiliamo "brute force" la combinazione corretta dei gradienti

3.2 Ricostruzione della Superficie

Il problema della ricostruzione della superficie è un problema di integrazione di equazioni differenziali, che possiamo trasformare in un sistema lineare, facilmente risolvibile in parecchi modi. Tuttavia, gli argomenti inseriti nel precedente capitolo non sono stati scelti a caso, infatti il nostro modello prevede la discretizzazione con un metodo del secondo ordine dell'equazione di Poisson, e la risoluzione del sistema lineare risultante con il metodo iterativo del Gradiente Coniugato Precondizionato. La scelta di

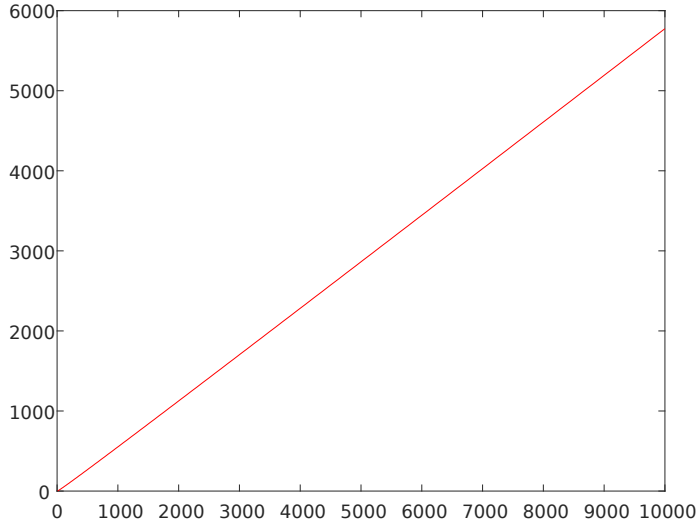


Figura 3.2: Condizionamento della matrice di Poisson in funzione delle sue dimensioni

questo metodo iterativo, non è casuale, infatti deriva da quella di utilizzare la matrice di Poisson, di cui abbiamo visto le proprietà più importanti: simmetrica, definita positiva e sparsa. Un altro aspetto interessante che si può notare è quello sul condizionamento di questa matrice: il suo condizionamento cresce linearmente con le sue dimensioni, e questo rende indispensabile l'utilizzo del precondizionamento per avere delle prestazioni computazionali elevate.

Ricordando l'equazione 2.8 che abbiamo trovato nel caso generale, dobbiamo ora applicarla alla nostra situazione

$$\begin{cases} \frac{\partial^2}{\partial x_i} z_{i,j} + \frac{\partial^2}{\partial y_j} z_{i,j} = \eta_{i,j}(p,q) \\ z_{0,j} = z_{row,j} = z_{i,0} = z_{i,col} = \hat{z} \quad \forall i \in [0, row], \quad \forall j \in [0, col] \end{cases}$$

dove abbiamo soltanto cambiato notazione: $z_{i,j}$ per le quote della superficie, il termine noto $\eta_{i,j}(p,q)$, che ricaveremo a breve, è una funzione dei gradienti ricavati nel passo precedente, (row, col) sono le dimensioni verticali e orizzontali delle immagini in ingresso (quindi il dominio stavolta sarà così definito $\Omega = \{[0, row-1] \cup [0, col-1]\}$).
Prima di tutto, dobbiamo notare che i gradienti sono stati ottenuti in forma di vettore, e prima di procedere abbiamo bisogno di riordinarli in una

matrice di dimensioni $row \times col$. Una volta effettuato il *reshape*, facendo riferimento alla 3.1 il nostro problema può essere riscritto

$$\frac{\partial}{\partial x_i} p_{i,j} + \frac{\partial}{\partial y_j} q_{i,j} = \eta_{i,j}$$

e applicando un semplice FDM diventa

$$\frac{p_{i+1,j} - p_{i,j}}{h} + \frac{q_{i,j+1} - q_{i,j}}{h} = \eta i, j$$

A questo punto abbiamo ottenuto una matrice $\eta \in \mathbb{R}^{row \times col}$ che contiene il termine noto dell'equazione di Poisson. Questa matrice è in generale rettangolare, mentre la matrice A di Poisson è quadrata per definizione, quindi non ci resta che rendere η quadrata aggiungendo degli zeri, ed effettuare nuovamente un *reshape* in forma di vettore colonna $\eta \in \mathbb{R}^{n^2}$, dove $n = max(row, col)$. Ora il sistema

$$A\mathbf{z} = \eta$$

è pronto per essere risolto con le modalità discusse nei capitoli precedenti, ovvero Fattorizzazione Cholesky Incompleta Modificata di A, e soluzione del sistema con il metodo PCG. Otteniamo le $\mathbf{z}$ in un vettore, e ancora una volta effettuiamo un *reshape* che lo riporti in forma di matrice con le sue reali dimensioni (cioè le stesse delle immagini di partenza).

Provando a riassumere i passaggi

1. Costruzione del termine noto η in forma matriciale, *zero-filling* e *reshape* in un vettore colonna
2. Fattorizzazione Cholesky Incompleta Modificata della matrice di Poisson
   - L=*ichol*(A,struct('michol','on'))
3. Soluzione del sistema con metodo dei Gradienti Coniugati Precondizionato e *reshape*
   - z=*pcg*(A,eta,tolerance,maxiter,L,L')

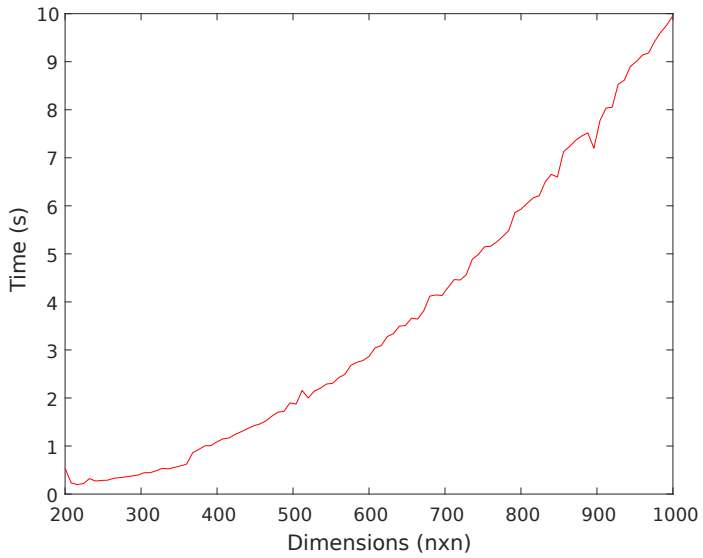


Figura 3.3: Tempo di integrazione dell'equazione di Poisson in funzione della dimensione

Non resta che trasformare la matrice Z delle altezze in un formato compatibile con un software di visualizzazione 3D. Nella nostra attività di ricerca abbiamo utilizzato il software *Meshlab* [11] per visualizzare file in formato *.ply* [12].

3.2.1 Formato PLY

Nell'ambito della Computer Vision esistono innumerevoli formati diversi per visualizzare oggetti tridimensionali, ne è stato scelto uno che fosse il più semplice possibile, e allo stesso tempo con un alto grado di compatibilità. Noto come *Polygon Format* o *Stanford Format*, questo descrive un oggetto come una lista di vertici, facce e altri elementi (per esempio margini, tipo di materiale), e a ognuno di questi sono associati delle proprietà come per esempio le coordinate (x, y, z) o il colore. La struttura di base è composta da:

- *Intestazione*
- *Lista dei Vertici*
- *Lista delle Facce*
- *Eventuali liste di ulteriori elementi*

In *Matlab* è sufficiente convertire le informazioni sulle altezze, contenute nella nostra matrice finale Z, in informazioni "poligonali", cioè in vertici e facce con la funzione *surf2patch*, aggiungere le informazioni sul colore grazie all'albedo, e stampare il tutto su un semplice file di testo secondo le regole imposte dal formato.
Con questo sistema le ricostruzioni ottenute con il nostro algoritmo possono essere visualizzate semplicemente nella gran parte dei software di *mesh-processing*. Alcuni esempi completi di file *.ply* possono essere trovati a questo link.

Capitolo 4

Prove sul Campo

Ci accingiamo ad analizzare i risultati ottenuti dal software sul campo. Ciascun soggetto è stato ripreso cercando di rispettare il più possibile le pratiche descritte nei capitoli precedenti, riguardanti le condizioni di illuminazione, la distanza e inclinazione delle sorgenti, e l'equipaggiamento ottico. Per ogni modello 3D verrà riportato l'intero set di immagini e relativo istogramma, l'immagine a colori relativa all'albedo e uno o più screenshot significativi acquisiti sull'editor 3D *Meshlab*.

4.1 Gigante A

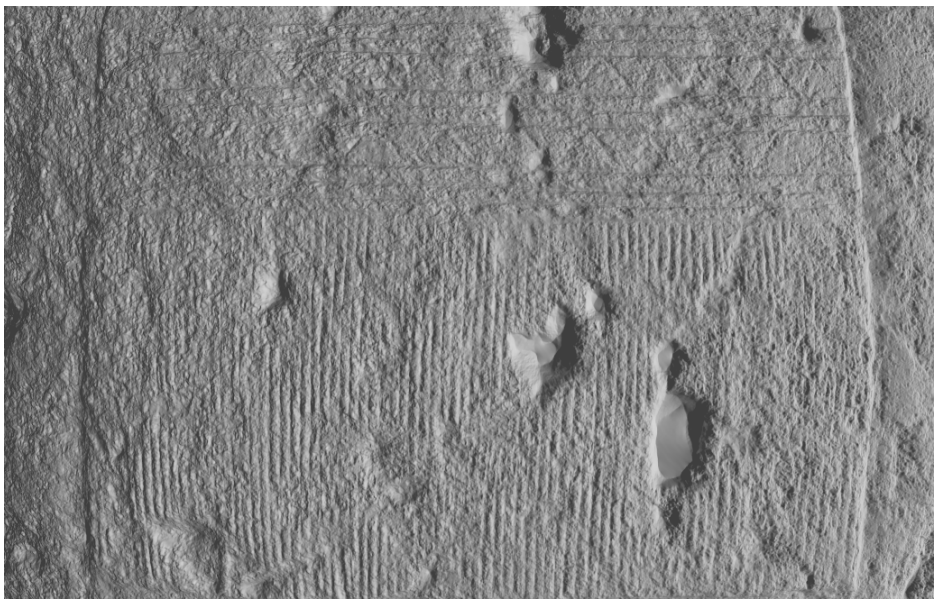


Figura 4.1: Tutte le incisioni sono ben visibili

Per questo Gigante abbiamo preso come soggetto le incisioni che porta sull'addome, che era di particolare interesse per l'alta "densità di dettagli". Dallo screenshot in figura 4.1 si può apprezzare l'elevata accuratezza nei

dettagli, ma è anche evidente dall'andamento dei bordi la "bombatura" della superficie, infatti nel set di partenza è distinguibile la natura conica del fascio. Per quanto riguarda l'albedo, c'è una "ovvia" limitazione, nel senso che dove la luce non arriva, non può recuperare informazioni; ci si

riferisce in particolare ai buchi che si trovano nell'addome di questo gigante, in cui la luce radente non si riesce ad addentrare in nessuno dei quattro scatti. Altre zone della superficie sono invece bene illuminate in 2-3 scatti, mentre sono in ombra nel quarto e in casi come questi il colore finale ne risulta parzialmente influenzato, in misura proporzionale alla "intensità dell'ombra" e del numero di scatti in penombra. Il lettore è invitato a individuare le porzioni di superficie interessate da questo fenomeno. Al di là, di queste considerazioni, il risultato dell'albedo è più che soddisfacente.

4.2 Gigante B

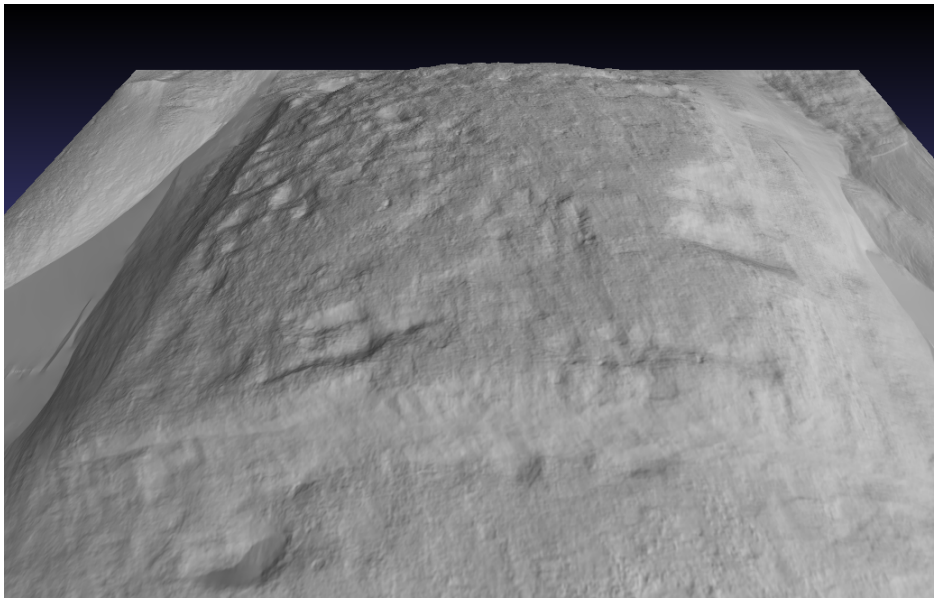
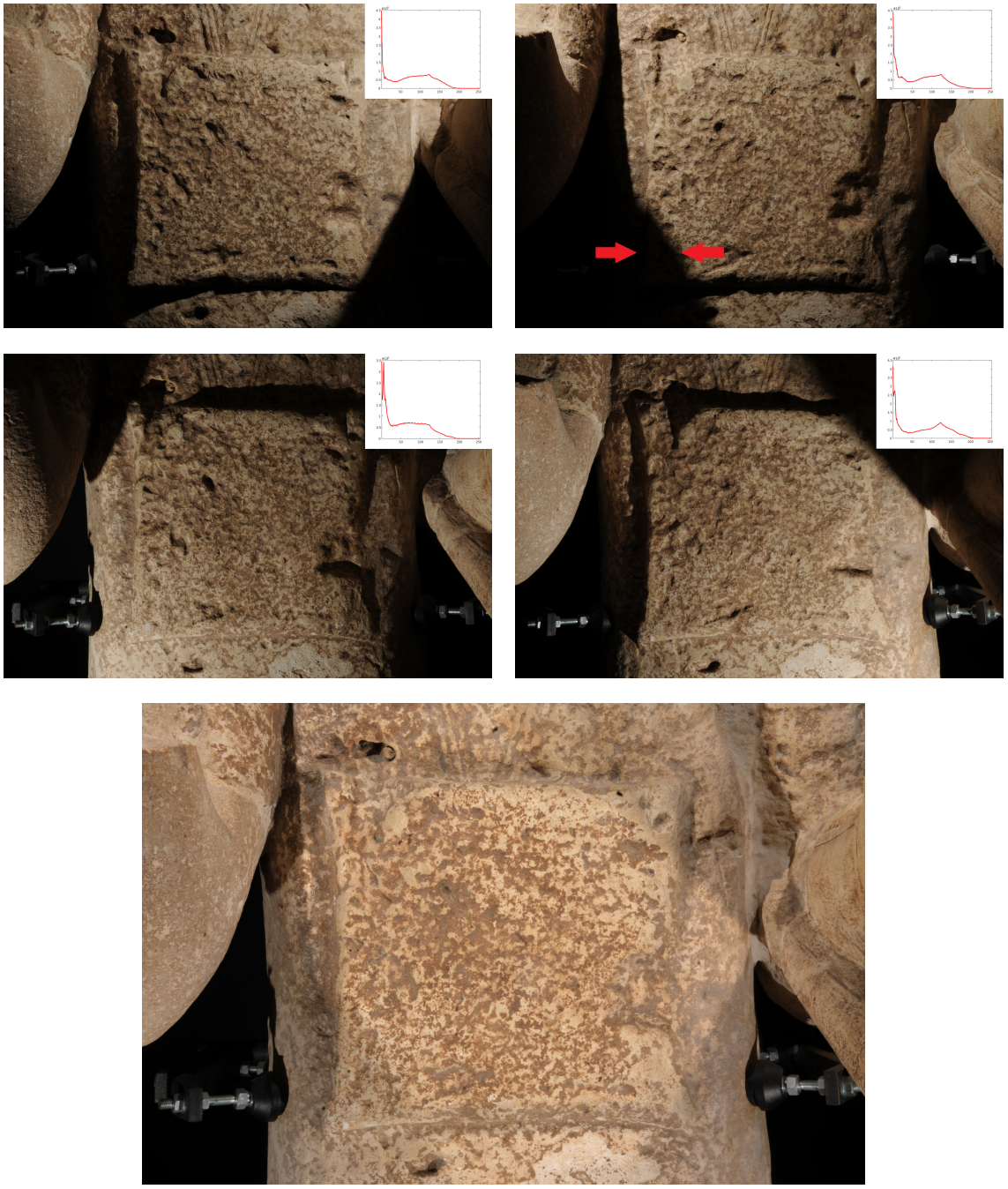


Figura 4.2: La porzione all'angolo in basso a destra è ancora più ricurva

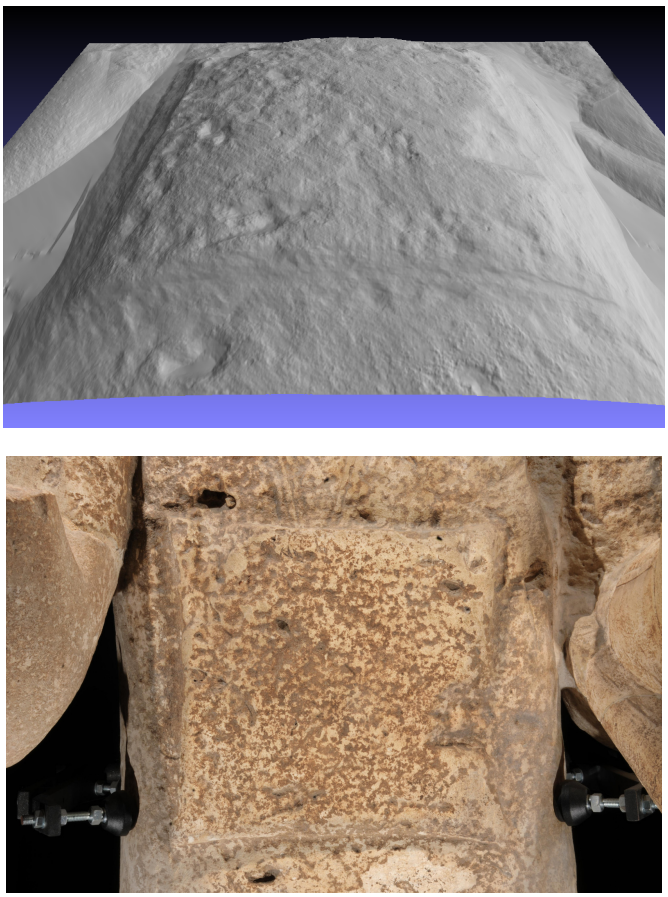
Per questo Gigante abbiamo scelto l'addome, che presenta una superficie particolarmente butterata e rappresenta una sfida per la tecnica su diversi fronti, ed anche l'impugnatura dell'arco. Nel primo tentativo sull'addome abbiamo commesso un errore in ripresa di cui ci siamo resi conto in seguito: per la conformazione della statua stessa, è stato problematico puntare le luci in modo da non proiettare delle ombre indesiderate sul soggetto. La ricostruzione in figura 4.2 ne ha risentito nel calcolo delle normali, nelle modalità accennate nel capitolo precedente, e la stessa porzione evidenzia-

ta nel secondo scatto presenta una colorazione più scura nell'albedo (anche nella porzione in alto a destra, seppure in misura minore). Abbiamo poi



preso un secondo set di immagini stavolta prestando attenzione alle ombre proiettate dalle braccia del Gigante. Ora l'unico difetto è legato al fascio conico che ancora una volta genera l'effetto di bombatura, mentre l'albedo stavolta ha fornito un ottimo risultato.

Passiamo ora all'impugnatura dell'arco, che è stata particolarmente sor-



prendente, infatti questa presenta quelle che a colpo d'occhio sembrerebbero delle semplici incisioni vuote, ma sono invece piene di terriccio che il restauro non ha eliminato. Risulta ben visibile dalla ricostruzione in figura 4.3 come in alcune di queste incisioni il terriccio "strabordi" e venga fuori in rilievo. Anche il risultato sul fronte albedo è stato eccezionale, come

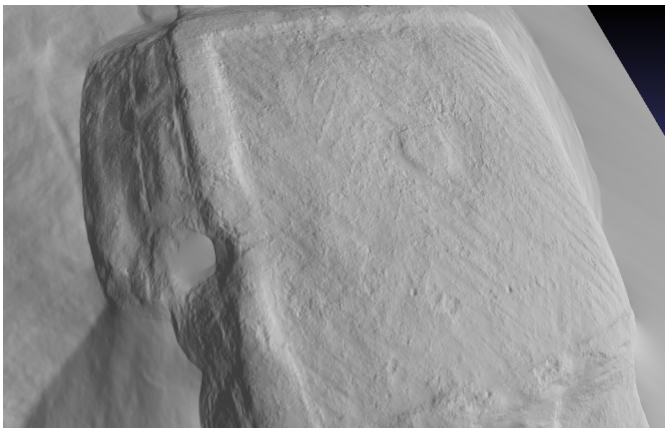


Figura 4.3: Dal particolare si notano alcuni "rilevi" di terriccio

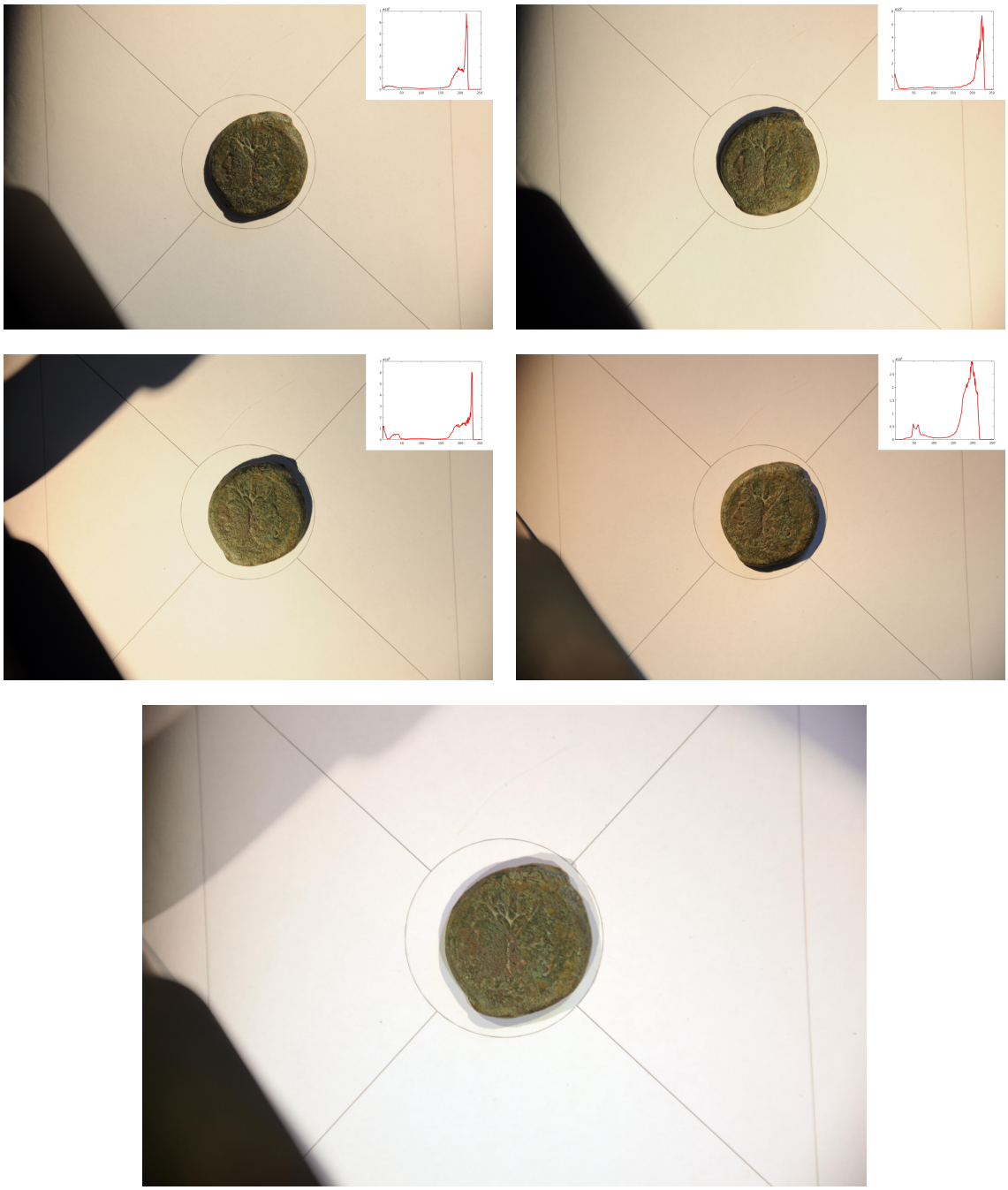
apprezzabile in figura, e in effetti le foto di questo particolare set sono

particolarmente fedeli alle linee guida esposte qui e in [3]. Tuttavia, la curvatura della superficie stona sempre un po'.

4.3 Moneta A

C'è un motivo molto semplice per cui le monete rappresentano un soggetto ancora più adatto per la nostra tecnica: sono di piccole dimensioni e

il fascio luminoso del flash la illumina con ottima approssimazione in modo uniforme; risulta di fatto immune all'effetto bombatura a cui si è fatto cenno in precedenza, come si può vedere in 4.4. Al di là della scarsa cura



prestata alle nostre ombre durante la ripresa, il foglio bianco offre la possibilità di apprezzare ancora meglio l'operato dell'algoritmo sul fronte albedo. Infatti in nessuno dei quattro scatti presenta la colorazione bianca, ma è anzi tendente al rosa. Nell'albedo è invece quasi completamente bianco (a

eccezione delle zone d'ombra, comprese quelle proiettate dalla moneta sul foglio che causano il fenomeno a cui si è fatto cenno in precedenza).

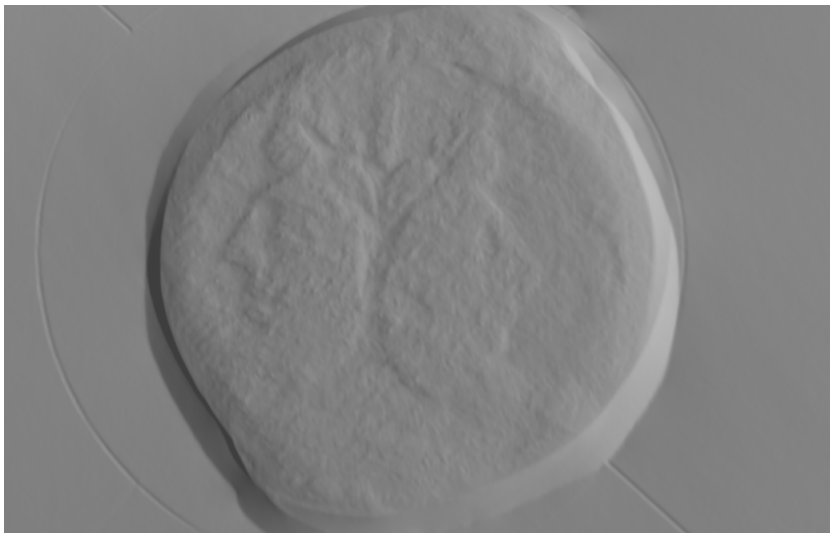


Figura 4.4: Nel bordo superiore è visibile il marchio impresso dal conio

4.4 Moneta B

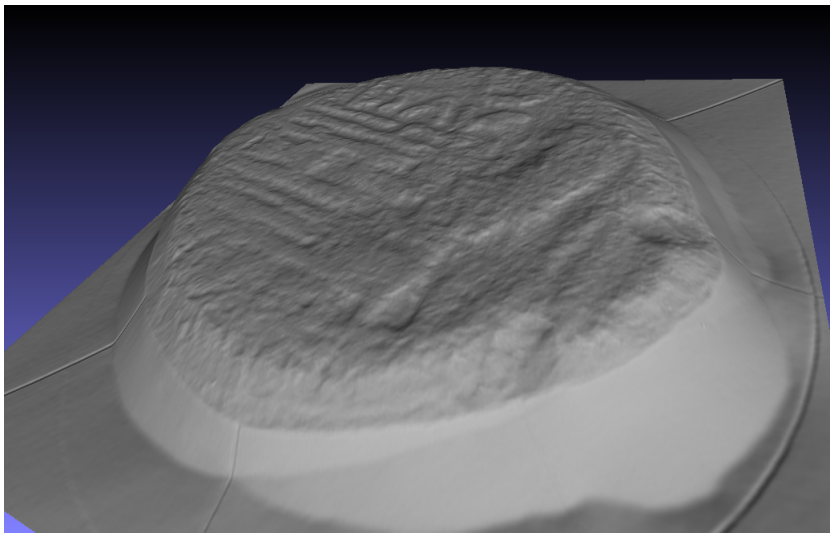
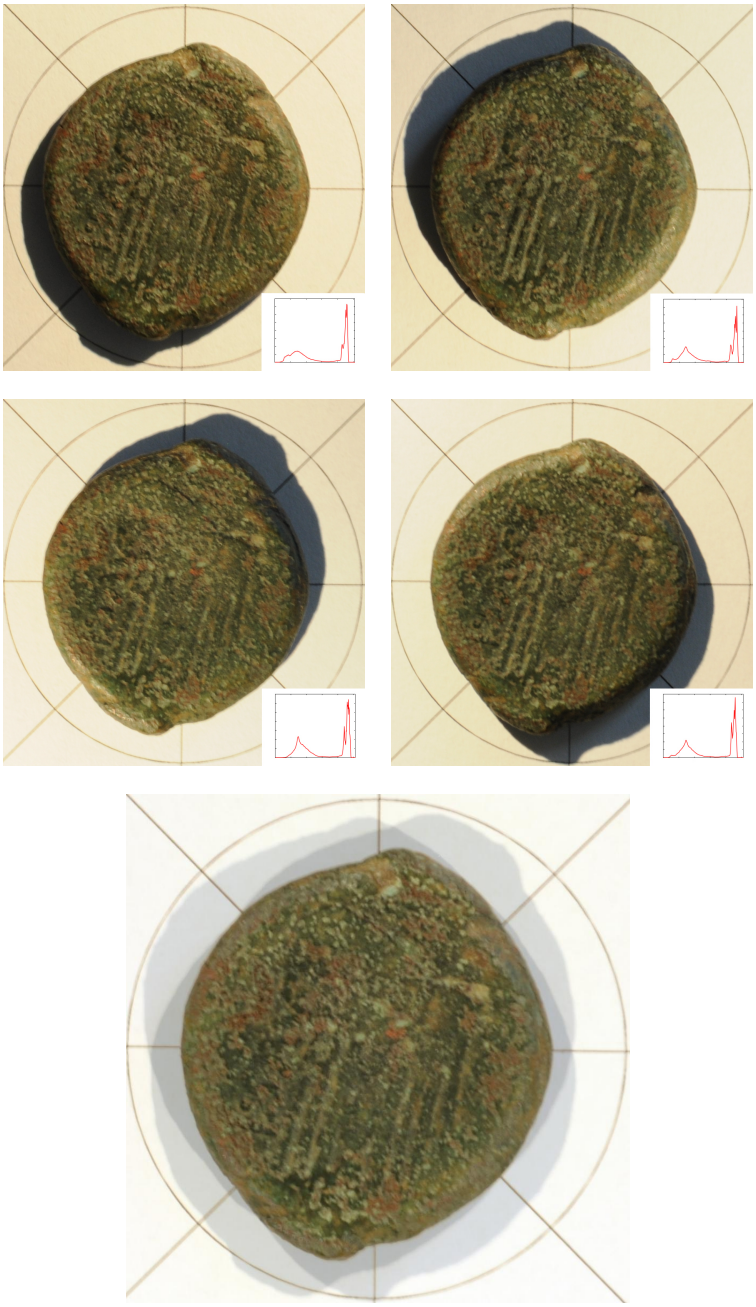


Figura 4.5: Le ombre proiettate dalla moneta falsano la ricostruzione sul contorno

Essendo il nostro soggetto soltanto una piccola parte dell'intera scena, è in realtà buon senso ritagliare l'immagine (facendo attenzione a far corrispondere correttamente i pixel nelle varie figure) in modo da dover elaborare molti meno pixel senza modificare affatto la qualità dell'immagine. In

realtà si potrebbe anche creare delle "maschere" *ad hoc* per ciascun soggetto, cioè colorare di nero tutto ciò che non ci interessa nella scena. Entrambe queste operazioni sono facilmente eseguibili per esempio su *GIMP*. [13]

Il risultato dell'albedo è ancora una volta ottimo.



Capitolo 5

Conclusioni

Abbiamo visto come, seguendo delle istruzioni estremamente semplici, sia possibile ottenere dei modelli 3D piuttosto fedeli nella forma e nel colore, facendo utilizzo di un algoritmo di complessità relativamente bassa. Anche allontanandosi dalle condizioni ideali i risultati restano abbondantemente sufficienti, e anche quando questo dovesse accadere è facilissimo rendersene conto in tempi brevissimi. Una delle direzioni da seguire al momento potrebbe essere quella sulla correzione dell'albedo in presenza di ombre/componenti speculari: in tal senso si potrebbe pensare di studiare, implementare ed eventualmente migliorare l'algoritmo proposto da [14]. Testare il software su oggetti di così grande valore archeologico e culturale ci ha permesso di dimostrare che la tecnica è pronta per essere ampiamente utilizzata su questo campo anche da non addetti ai lavori. A questo proposito, è d'obbligo tradurre il codice *Matlab* in un linguaggio *Standalone* in modo da rendere disponibile il software al di fuori del nostro ambiente di sviluppo. L'idea al momento è di sviluppare il programma in *Python*, per diversi motivi: esiste un certo grado di affinità tra i due linguaggi a livello di sintassi, le librerie di matematica di *Python* sono precompilate (come quelle di *Matlab*) e per questo non si dovrebbe riscontrare una caduta eccessiva nelle prestazioni.

Bibliografia

[1] G.Rodriguez, "Algoritmi Numerici", Pitagora Editrice Bologna, 2008

[2] R.Dessì, C.Mannu, G.Rodriguez, G.Tanda, M.Vanzi, "Recent improvements in photometric stereo for rock art 3D imaging". Digital Applications in Archaeology and Cultural Heritage vol.2, pp. 132-139, 2015

[3] C.Mannu, "Photometric Stereo for 3D Mapping of Carving and Relieves", 2014

[4] R.Dessì, "Algoritmi ottimizzati per la Photometric Stereo", 2014

[5] R.Pintus, M.Vanzi, "Advances in Photometric Sereo", 2007

[6] R.J.Woodham, "Photometric method for determining surface orientation from multiple images". Optical Engineerings vol.19, no.1, pp. 139-144, 1980

[7] A.Chambolle "A uniqueness result in the theory of stereo vision: coupling shape from shading and binocular information allows unambiguous depth reconstruction". Annales de l'I.H.P Analyse non linéare 11(1):1–16, 1994

[8] The MathWorks Inc., MATLAB 2017a, Natick, Massachusetts, United States.

[9] R.Basri, D.Jacobs, I.Kemelmacher, "Photometric Stereo with general, unknown lighting. International Journal of Computer Vision 72.3 (2007): 239-257.

[10] J.R.Shewchuk, "An introduction to the conjugate Gradient method without the agonizing pain", 1994

[11] P.Cignoni, M.Callieri, M.Corsini, M.Dellepiane, F.Ganovelli, G.Ranzuglia, "Meshlab: an open-source mesh processing tool." Proceedings of the 2008 Eurographics Italina Chapter Conference, ISBN: 978-3-905673-68-5, pp. 129-136, DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

[12] G.Turk "The PLY Polygon File Format" Georgia Institute of Technology 1998

[13] The GIMP Team, GIMP 2.8.22, http://gimp.org/, 1995-2017

[14] S.Barsky, M.Petrou, "The 4-Source Photometric Stereo Technique for Three Dimensional Surfaces in the Presence of Highlights and Shadows", IEEE Transactions On Pattern Analysis and Machine Intelligence, vol.25, no.10, 2003