Shape From Shading

Marta Pibiri, 60/73/65175
Francesca Cella, 60/73/65172

1 Introduzione

Tramite tecniche di computer vision è possibile ottenere la superficie 3D di un oggetto tramite procedure particolari e dedicate. Si sfrutta il concetto di ombra che caratterizza un oggetto e la sua forma, da qui il nome di *Shape from shading*.
Vi sono tre principali procedimenti: la *Stereo Vision*, o *Multivision*, la quale utilizza più telecamere per inquadrare la scena da diverse angolazioni; un'altra tecnica è il *Laser Scanning*, la quale consiste nel movimento di un raggio laser per tracciare i punti dell'oggetto nello spazio 3D, si tratta di una tecnica accurata che dà informazioni sulla forma ma è computazionalmente onerosa.

La tecnica analizzata in esperimento è la *Photometric Stereo* (1), la quale consiste nell'utilizzo di una sola telecamera, con posizione fissa, e diverse fonti di illuminazione poste in diverse angolazioni: tramite le ombre è possibile ricostruire il modello tridimensionale, misurando l'orientamento della sua superficie.

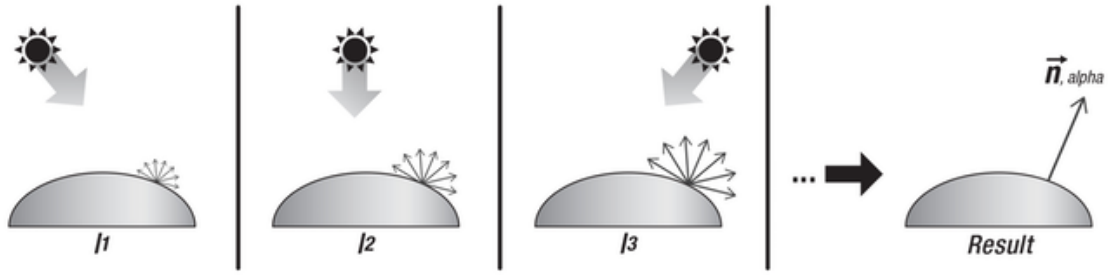


Figure 1: Photometric Stereo Tecnique

I test si concentreranno sul calcolo di incognite necessarie per poi passare allo step di modellazione della superficie; questa parte è compito di uno script esterno. Quest'ultima fase ha scopo visivo e per poter confrontare meglio le diverse metodologie usate per calcolare tali incognite. L'esperimento effettuato è diviso in due parti: nella prima

parte, sono stati implementati due metodi di fattorizzazione QR, quello di Givens e quello di Householder; nella seconda parte, è stata effettuata la risoluzione del sistema, sia con i vari metodi QR che con SVD di MATLAB, che permette di stabilire le normali della superficie a partire dalle immagini e dalle luci in input. Quest'ultimo step viene completato con la visualizzazione 3D dell'oggetto.

2 Analisi del problema

2.1 Legge di Lambert

La legge di Lambert mette in relazione la riflettanza di una superficie, ovvero il suo albedo, valore dipendente dal materiale, con le sue normali e le informazioni sulla luce. La legge si esprime come:

$$i = \alpha \langle \underline{n}, \underline{l} \rangle$$

Se un oggetto viene illuminato di fronte, si vede bene, ma se viene illuminato di lato è buio: al crescere dell'angolo tra la posizione dell'osservatore e la fonte luminosa, l'intensità della riflessione diminuisce.

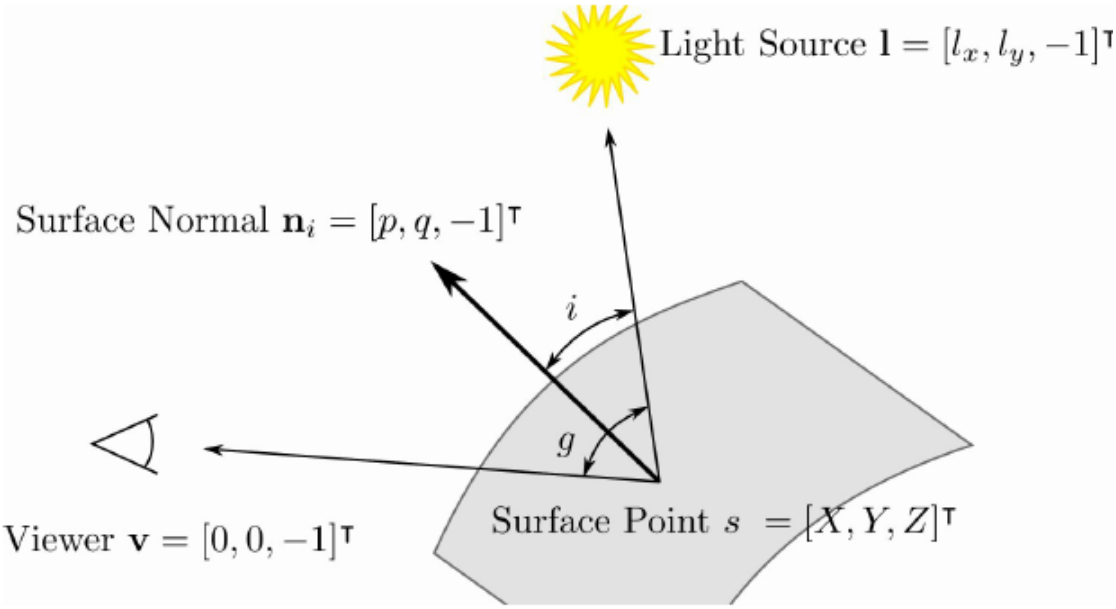


Figure 2: Angolo della luce e angolo dell'osservatore rispetto alla superficie

Si suppone che la fonte luminosa abbia distanza infinita dall'oggetto; un esempio di questo tipo di fonte luminosa è il Sole. È possibile definire ogni pixel tramite la legge di Lambert, in questo caso avremmo:

$$i = \alpha \langle n, l \rangle$$

2.2 Formulazione matriciale

Il singolo pixel di una foto può essere indicato come F_{ij}. Considerando che l'esperimento utilizza un certo numero di foto come input, è preferibile trasformare le immagini in vettori e ottenere una immagine per colonne, indicando quindi il singolo pixel con un solo indice. Formalmente:

$$F_{\mathrm{ij}} \rightarrow F_{\mathrm{k}}$$

dove $i = 1...p$, $j = 1...q$, $k = 1...pq$.
Quindi il singolo pixel può essere espresso tramite legge di Lambert con la seguente formulazione:

$$M_{\mathrm{kr}} = \alpha_{\mathrm{k}} \langle\, n_{\mathrm{k}}, l_{\mathrm{k}} \,\rangle$$

dove $r = 1...q$ indica il numero di foto a disposizione. Le foto a disposizione devono essere tre o più; nel caso in cui si disponesse di più di tre foto, allora il sistema sarebbe sovradeterminato: avremmo un problema con più misure che incognite, per cui nessuna soluzione in senso classico. Il problema può essere formulato come sistema utilizzando le seguenti matrici:

$$DN^{\mathrm{T}}L = M$$

dove D è la matrice diagonale contenente i valori dell'albedo, N è la matrice contenente le normali, nonché incognita e obiettivo dell'esperimento, L è la matrice contenente le luci e M è la matrice contenente le foto.

- Nel nostro esperimento, per semplicità, supponiamo che la matrice degli albedo sia una matrice identità di dimensione $p \times p$.
- I vettori che rappresentano le normali possono essere rappresentati tramite una matrice contenente p righe, ovvero una riga per ogni pixel, e tre colonne.
- I vettori che rappresentano le luci possono essere espressi come matrice $3 \times r$, dove $r = 1...q$, con q uguale al numero di foto a disposizione.
- L'insieme delle immagini 2D che rappresentano l'oggetto può essere considerato come una matrice $p \times q$ dove p è il numero di pixel di ciascuna foto e q è il numero di foto scattate.

È importante scegliere le luci in maniera tale che il rango della matrice L sia pieno, dunque uguale a tre, perché se il rango non è pieno vuol dire che le luci quindi sono ridondanti; questo si riflette sul rango poiché indicherebbe la presenza di righe o colonne dipendenti.

$$\underset{p \times p}{\begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \alpha_{\mathrm{p}} \end{bmatrix}} \quad \underset{p \times 3}{\begin{bmatrix} n_{1\mathrm{x}} & n_{1\mathrm{y}} & n_{1\mathrm{z}} \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ n_{\mathrm{px}} & n_{\mathrm{py}} & n_{\mathrm{pz}} \end{bmatrix}} \quad \underset{3 \times q}{\begin{bmatrix} l_{1\mathrm{x}} & \vdots & \vdots & l_{\mathrm{qx}} \\ l_{1\mathrm{y}} & \vdots & \vdots & l_{\mathrm{qy}} \\ l_{1\mathrm{z}} & \vdots & \vdots & l_{\mathrm{qz}} \end{bmatrix}} = \underset{p \times q}{\begin{bmatrix} m_{11} & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \dots & \dots & m_{\mathrm{pq}} \end{bmatrix}}$$

In generale, l'esperimento consiste nella risoluzione di $p \times q$ equazioni: il risultato di ciascuna equazione è la direzione di ciascun pixel. Dopo aver ottenuto le normali delle

superficie, la matrice viene passata allo script esterno e tramite un insieme di equazioni differenziali e altre metodiche, si otterrà il modello 3D.

Considerando la forma matriciale del problema, per poter calcolare l'incognita, è necessario esplicitare la matrice delle normali:

$$\tilde{N} = DN^{\mathrm{T}}$$

Riprendendo la formulazione matriciale e utilizzando $\tilde{N}$:

$$\tilde{N}L = M$$
$$\tilde{N} = ML^{+}$$

Questo procedimento vale se $r > 3$, altrimenti, se $r = 3$, si usa l'inversa di L:

$$\tilde{N} = ML^{\text{-}1}$$

È consigliabile, però, utilizzare più di tre foto per ridurre l'errore.

La pseudoinversa della matrice delle luci può essere calcolata con il metodo SVD, Singular Value Decomposition:

$$L = U\Sigma V^{\mathrm{T}}$$
$$L^{+} = V\Sigma^{+}U^{\mathrm{T}}$$

$$\Sigma^{+} = \begin{bmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_{\mathrm{p}}} \end{bmatrix}$$

3 Esperimenti con MATLAB

3.1 Primo esperimento

Nella prima fase dell'esperimento si è voluta riporre l'attenzione sulla metodica QR e sulla bontà di tale implementazione su MATLAB.

La fattorizzazione QR viene utilizzata per la risoluzione di sistemi lineari, scompone una matrice A nel prodotto di due matrici, Q e R, dove R è una matrice triangolare superiore e Q è una matrice ortogonale, ovvero $Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I$.
Un sistema lineare $Ax = b$ può essere riscritto come $QRx = b$ e la sua risoluzione può essere espressa come:

$$\begin{cases} Qy = b \\ Rx = y \end{cases}$$

dove $y = Q^{\mathrm{T}}b$. Il metodo QR ha complessità $\mathcal{O}(\frac{2}{3}n^3)$.

3.1.1 Fattorizzazione QR con Householder

Matrice elementare di Householder La matrice viene descritta dalla formula:

$$H = I - 2ww^{\mathrm{T}}$$

Proprietà:

- La matrice H è simmetrica:

$$\begin{gathered} H^{\mathrm{T}} = (I - 2ww^{\mathrm{T}})^{\mathrm{T}}(I - 2ww^{\mathrm{T}}) = \\ = I - 2ww^{\mathrm{T}} - 2ww^{\mathrm{T}} + 4ww^{\mathrm{T}}ww^{\mathrm{T}} = \\ = I - 2ww^{\mathrm{T}} - 2ww^{\mathrm{T}} + 4ww^{\mathrm{T}} = I \end{gathered}$$

- La matrice H è endomorfa, H è simmetrica e ortogonale, infatti:

$$H^{\mathrm{T}}H = I = HH^{\mathrm{T}}$$

La complessità della creazione di tale matrice è $\mathcal{O}(n^2)$. Di seguito l'algoritmo:

Algorithm 1: Householder Matrix

Input: x, n
Output: H, k

1 $\sigma = \|x\|$;
2 $k = -sign(x_1)\sigma$;
3 $\lambda = \sqrt{2\sigma(\sigma + |x|)}$;
4 $w = \frac{x - ke_1}{\lambda}$;
5 $H = I - 2ww^{\mathrm{T}}$;

Fattorizzazione L'obiettivo della fattorizzazione è quello di ottenere una matrice triangolare attraverso $n-1$ passi, dove n è la dimensione della matrice quadrata. Nel caso di matrici rettangolari $m \times n$, con $m > n$, i passi sono n. Si moltiplicano le matrici H ottenute ai vari step per la matrice di partenza A in modo da ottenere la matrice R, il prodotto delle varie matrici H è pari a Q^{T}.

$$H_{\text{n-1}} \dots H_1 A = R$$
$$H_{\text{n-1}} \dots H_1 = Q^{\mathrm{T}}$$

La complessità della fattorizzazione QR di Householder per matrici quadrate è $\mathcal{O}(\frac{2}{3}n^3)$, mentre per matrici rettangolari è $\mathcal{O}(n^2(m - \frac{1}{3}n))$. Di seguito l'algoritmo:

Algorithm 2: Householder QR

Input: A, n
Output: Q, R
1 $Q = I_{\mathrm{n}}$;
2 $A^{(1)} = A$;
3 **for** $i \leftarrow 1$ **to** $n-1$ **do**
4 | *costruisci* H_{i};
5 | $A = H_{\mathrm{i}} A^{(\mathrm{i})}$;
6 | $Q = Q H_{\mathrm{i}}$;
7 **end**
8 $R = A^{(\mathrm{n})}$;

3.1.2 Fattorizzazione QR con Householder

Matrice elementare di Givens La matrice elementare di Givens produce la rotazione del vettore al quale è applicata. La rotazione può essere espressa tramite matrice, G, come:

$$G = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

La matrice elementare, G_{ij}, ha la seguente forma:

$$G_{\mathrm{ij}} = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & c & s & & & \\ & & -s & c & & & \\ & & & & \ddots & & \\ 0 & & & & & & 1 \end{bmatrix}$$

La riga contenente i parametri c e s è la riga i, quella contenente i parametri $-s$ e c è la riga j. Allo stesso modo, la colonna contenente i parametri c e $-s$ è la colonna i, mentre quella contenente i parametri c e s è la colonna j.
Proprietà:

- c e s sono i parametri della rotazione e la somma dei quadrati di questi parametri è pari a 1: $c^2 + s^2 = 1$;
- La matrice è ortogonale: ${G_{\mathrm{ij}}}^{\mathrm{T}} G_{\mathrm{ij}} = I = G_{\mathrm{ij}} {G_{\mathrm{ij}}}^{\mathrm{T}}$.

Di seguito l'algoritmo per il calcolo dei parametri di rotazione c e s:

Algorithm 3: Giv Rot

Input: $x_{\mathrm{i}}, x_{\mathrm{j}}$
Output: c, s

```
if x_j == 0 then
    c = 1;
    s = 0;
else
    if |x_i| > |x_j| then
        t = x_i / x_j;
        z = sqrt(1 + t^2);
        s = 1/z;
        c = t × s;
    else
        t = x_j / x_i;
        z = sqrt(1 + t^2);
        c = 1/z;
        s = t × c;
    end
end
```

Fattorizzazione La fattorizzazione QR di Givens permette di sfruttare la presenza di zeri nella matrice, quindi lavora meglio su matrici sparse. Anche in questo caso, le matrici G ai vari step vengono moltiplicate per A in modo da trovare la matrice triangolare, R. Il prodotto delle matrici G corrisponde a Q^{T}.

$$G_{\mathrm{n\text{-}1\ n}} \dots G_{1\ 2} A = R$$
$$G_{\mathrm{n\text{-}1\ n}} \dots G_{1\ 2} = Q^{\mathrm{T}}$$

Se la matrice fosse stata rettangolare $m \times n$, con $m > n$, l'ultimo elemento sarebbe stato $G_{\mathrm{n\ n}}$.
La complessità della fattorizzazione per matrici quadrate è pari a $\mathcal{O}(\frac{4}{3}n^3)$, mentre per matrici rettangolari è pari a $\mathcal{O}(2n^2(m - \frac{1}{3}n))$.

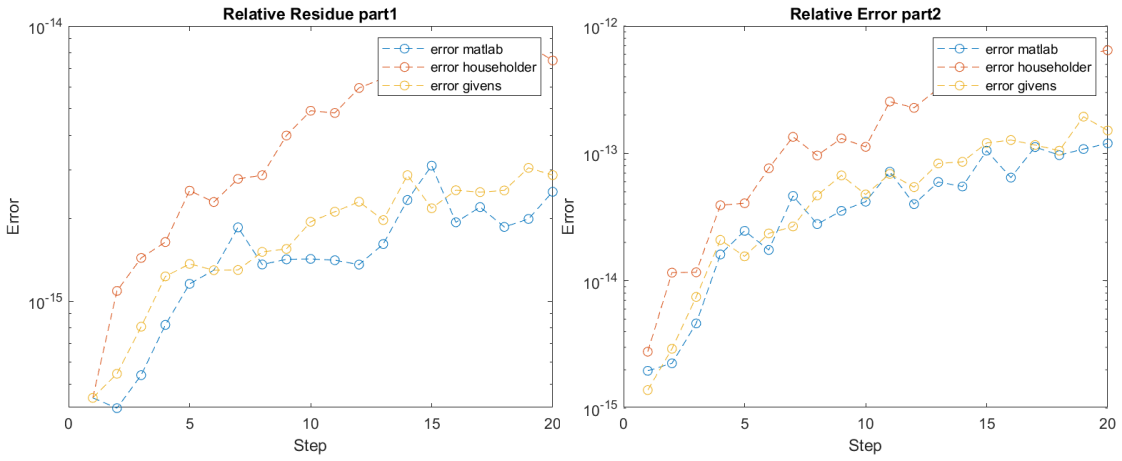
3.1.3 Test

Sono state implementate le due versioni, descritte sopra, dell'algoritmo QR, le quali sono state confrontate con l'implementazione tramite funzione QR di MATLAB (corrisponde

all'implementazione di Householder).
Sono stati prima generati 20 sistemi casuali di dimensione crescente, a partire dalla dimensione 20×10, proseguendo con passo 10 per ciascuna dimensione. I sistemi sono stati fattorizzati con i 3 metodi, ovvero QR MATLAB, QR Householder e QR Givens e sono stai calcolati, per ciascuno:

- il residuo relativo, confrontando la soluzione ottenuta con quella ottenuta con MATLAB utilizzando il backslash;
- l'errore della soluzione del sistema, confrontando la soluzione con quella stabilita inizialmente;
- l'errore della fattorizzazione QR;
- l'errore nella creazione della matrice Q;
- il tempo di esecuzione.

Residuo relativo ed errore relativo Di seguito i risultati del primo test:



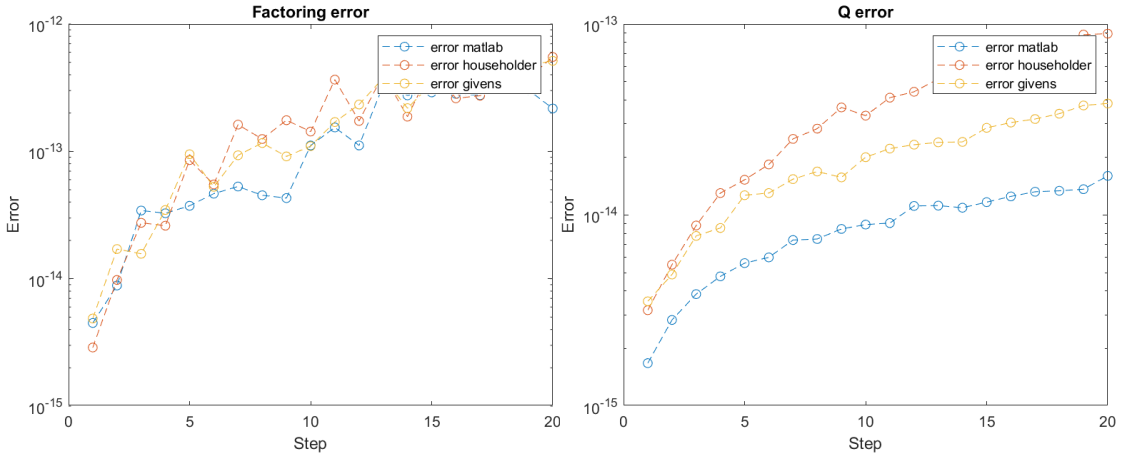
Il grafico *Relative residue* mostra il residuo relativo sul risultato del sistema, ottenuto con le tre procedure (MATLAB qr, Householder QR, Givens QR) in relazione alla risoluzione del sistema implementata in MATLAB, intesa come $A*x = b$. Il residuo è stato calcolata con la seguente formula:

$$e = \frac{\|Ax-b\|}{\|b\|}$$

Per quanto riguarda il grafico *Relative Error part2*, il risultato ottenuto tramite le tre implementazioni è stato confrontato utilizzando la soluzione originale, ovvero quella definita in fase di creazione del sistema casuale. In questo test, l'errore è stato calcolato come:

$$e = \frac{\|xOriginale-x\|}{\|xOriginale\|}$$

Nel primo test e nel secondo, l'errore di qr MATLAB non cresce oltre 10^{-13}.
Per quanto riguarda l'implementazione di Householder, nel primo test i risultati sono molto simili a quelli ottenuti con QR MATLAB mentre, nel secondo test, l'errore non cresce oltre 10^{-13}.
Nell'implementazione di Givens, nel primo test l'errore cresce in maniera abbastanza costante rispetto alla dimensione del sistema, arrivando a 3×10^{-13}, nel secondo invece non ha una crescita costante, il picco massimo arriva a 7×10^{-13}.



Errore assoluto sulla fattorizzazione e sull'ortogonalità di Q Il grafico *Factoring Error* mostra l'errore assoluto sulla fattorizzazione QR ottenuta tramite le 3 implementazioni, confrontando i risultati con la matrice di partenza A. L'errore è stato così calcolato:

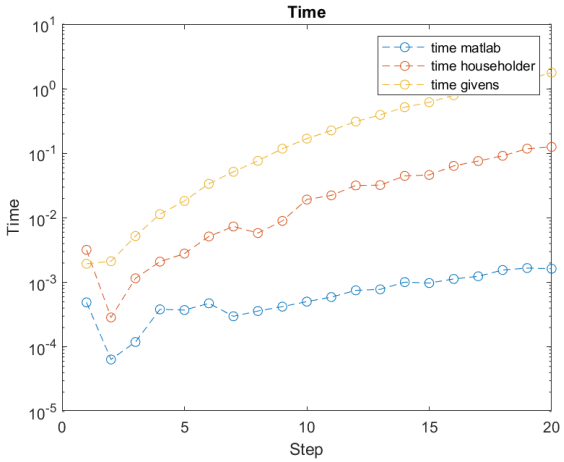
$$e = \|QR - A\|$$

Il grafico *Q error*, invece, mostra l'errore sulla matrice Q ottenuta attraverso le 3 implementazioni: in questo test è stata verificata l'ortogonalità della matrice Q, moltiplicando la sua trasposta (nonché inversa) per se stessa e confrontandola con la matrice identità avente la sua stessa dimensione. Di seguito il calcolo dell'errore:

$$e = \|Q^{\mathrm{T}}Q - I\|$$

In questo test, l'errore di fattorizzazione di MATLAB si mantiene sotto 0.5×10^{-13}, mentre l'errore sull'ortogonalità di Q ha una crescita lenta ma abbastanza costante e si mantiene sotto 0.2×10^{-14}.
L'errore prodotto dal risultato di Householder, per quanto riguarda la fattorizzazione, ha un andamento abbastanza simile a quello di MATLAB con alcuni picchi, ma si mantiene sempre sotto $\times 10^{-13}$; nel test dell'ortogonalità, invece, il metodo di Householder presenta un errore maggiore rispetto agli altri due metodi, con un picco in 1.8×10^{-14}.
Il metodo di Givens non ha una crescita costante: presenta alcuni picchi nell'errore della fattorizzazione, il maggiore è in prossimità di 2×10^{-13}. Per quanto riguarda, invece,

l'errore nell'ortogonalità di Q, la crescita è abbastanza costante e si mantiene sotto 0.6×10^{-14}.



Tempo di esecuzione Infine, il tempo di esecuzione della fattorizzazione QR di MATLAB e di Householder è abbastanza simile, entrambi si mantengono sotto $0.2 secondi$. Per quanto riguarda Givens, invece, il tempo cresce in maniera a abbastanza costante in base alla crescita della dimensione del sistema, arrivando circa a $1.8 secondi$ per il sistema di dimensione massima.

3.1.4 Osservazioni e analisi

In conclusione, è possibile notare come nel primo test l'errore maggiore si verifica con l'utilizzo di Householder, mentre gli altri due metodi presentano errori abbastanza simili; nel secondo test, invece, per quanto riguarda la fattorizzazione, l'implementazione meno precisa è quella di Givens mentre, per quanto riguarda Q, l'implementazione meno precisa è quella di Householder; nel terzo test, il metodo più lento al crescere della dimensione del sistema è Givens: ricordiamo che le matrici casuali scelte per i test sono matrici piene e il metodo di Givens ha una resa migliore su matrici sparse.

3.1.5 Errori calcolati

Relative Residue Part 1 on the system

Step	MatlabResidue	HouseHolderResidue	GivensResidue
1.0000e+00	4.4530e-16	4.4530e-16	4.4530e-16
2.0000e+00	4.0861e-16	1.0896e-15	5.4482e-16
3.0000e+00	5.3852e-16	1.4360e-15	8.0778e-16
4.0000e+00	8.2015e-16	1.6403e-15	1.2302e-15
5.0000e+00	1.1558e-15	2.5218e-15	1.3660e-15
6.0000e+00	1.2959e-15	2.2928e-15	1.2959e-15
7.0000e+00	1.8547e-15	2.7820e-15	1.2983e-15
8.0000e+00	1.3594e-15	2.8698e-15	1.5104e-15
9.0000e+00	1.4190e-15	3.9991e-15	1.5480e-15
1.0000e+01	1.4244e-15	4.9206e-15	1.9424e-15
1.1000e+01	1.4089e-15	4.8304e-15	2.1133e-15
1.2000e+01	1.3580e-15	5.9544e-15	2.2982e-15
1.3000e+01	1.6128e-15	6.4512e-15	1.9712e-15
1.4000e+01	2.3323e-15	6.6381e-15	2.8705e-15
1.5000e+01	3.1090e-15	6.0625e-15	2.1763e-15
1.6000e+01	1.9351e-15	6.5496e-15	2.5305e-15
1.7000e+01	2.1976e-15	7.7648e-15	2.4906e-15
1.8000e+01	1.8613e-15	8.5090e-15	2.5261e-15
1.9000e+01	1.9900e-15	8.4906e-15	3.0513e-15
2.0000e+01	2.4977e-15	7.4930e-15	2.8723e-15

Relative Error Part 2 on the original solution

Step	MatlabError	HouseHolderError	GivensError
1.0000e+00	1.9641e-15	2.7728e-15	1.3864e-15
2.0000e+00	2.2489e-15	1.1582e-14	2.9236e-15
3.0000e+00	4.6242e-15	1.1652e-14	7.4666e-15
4.0000e+00	1.6082e-14	3.9175e-14	2.0923e-14
5.0000e+00	2.4654e-14	4.0615e-14	1.5543e-14
6.0000e+00	1.7452e-14	7.6397e-14	2.3520e-14
7.0000e+00	4.6372e-14	1.3478e-13	2.6689e-14
8.0000e+00	2.7850e-14	9.6331e-14	4.6826e-14
9.0000e+00	3.5389e-14	1.3119e-13	6.7211e-14
1.0000e+01	4.1734e-14	1.1249e-13	4.7550e-14
1.1000e+01	7.1730e-14	2.5500e-13	6.9023e-14
1.2000e+01	3.9954e-14	2.2776e-13	5.4371e-14
1.3000e+01	5.9642e-14	3.2328e-13	8.3306e-14
1.4000e+01	5.4925e-14	4.4463e-13	8.5835e-14
1.5000e+01	1.0505e-13	6.6330e-13	1.2072e-13
1.6000e+01	6.4542e-14	3.1965e-13	1.2730e-13
1.7000e+01	1.1216e-13	4.0580e-13	1.1634e-13
1.8000e+01	9.6842e-14	5.1885e-13	1.0499e-13
1.9000e+01	1.0818e-13	5.9742e-13	1.9410e-13
2.0000e+01	1.1986e-13	6.4525e-13	1.5130e-13

Factoring error

Step	MatlabError	HouseHolderError	GivensError
1.0000e+00	4.4964e-15	2.8866e-15	4.8780e-15
2.0000e+00	8.8532e-15	9.7856e-15	1.7108e-14
3.0000e+00	3.4320e-14	2.7486e-14	1.5706e-14
4.0000e+00	3.2698e-14	2.6102e-14	3.4679e-14
5.0000e+00	3.7392e-14	8.5421e-14	9.4924e-14
6.0000e+00	4.6589e-14	5.4987e-14	5.2722e-14
7.0000e+00	5.2989e-14	1.6217e-13	9.3189e-14
8.0000e+00	4.5228e-14	1.2504e-13	1.1626e-13
9.0000e+00	4.2991e-14	1.7569e-13	9.1302e-14
1.0000e+01	1.1133e-13	1.4305e-13	1.0994e-13
1.1000e+01	1.5417e-13	3.6579e-13	1.6990e-13
1.2000e+01	1.1134e-13	1.7327e-13	2.3322e-13
1.3000e+01	3.3001e-13	4.0200e-13	3.7720e-13
1.4000e+01	2.7522e-13	1.8726e-13	2.1842e-13
1.5000e+01	2.9034e-13	4.5642e-13	3.4029e-13
1.6000e+01	2.8309e-13	2.6095e-13	5.8716e-13
1.7000e+01	2.7583e-13	2.7346e-13	4.4218e-13
1.8000e+01	3.4582e-13	3.2445e-13	4.4782e-13
1.9000e+01	3.0314e-13	3.7743e-13	4.7362e-13
2.0000e+01	2.1718e-13	5.5092e-13	5.1382e-13

Error of Q matrix

Step	MatlabError	HouseHolderError	GivensError
1.0000e+00	1.6789e-15	3.1786e-15	3.5369e-15
2.0000e+00	2.8250e-15	5.4993e-15	4.8831e-15
3.0000e+00	3.8563e-15	8.8192e-15	7.7793e-15
4.0000e+00	4.7784e-15	1.3036e-14	8.5596e-15
5.0000e+00	5.6051e-15	1.5302e-14	1.2701e-14
6.0000e+00	6.0042e-15	1.8396e-14	1.3037e-14
7.0000e+00	7.4032e-15	2.5039e-14	1.5422e-14
8.0000e+00	7.4941e-15	2.8345e-14	1.6904e-14
9.0000e+00	8.4633e-15	3.6588e-14	1.5745e-14
1.0000e+01	8.9174e-15	3.3149e-14	2.0085e-14
1.1000e+01	9.0808e-15	4.1159e-14	2.2286e-14
1.2000e+01	1.1164e-14	4.4186e-14	2.3350e-14
1.3000e+01	1.1209e-14	5.1553e-14	2.4023e-14
1.4000e+01	1.0925e-14	5.5510e-14	2.4147e-14
1.5000e+01	1.1686e-14	6.3932e-14	2.8618e-14
1.6000e+01	1.2522e-14	6.6037e-14	3.0501e-14
1.7000e+01	1.3243e-14	7.1216e-14	3.1810e-14
1.8000e+01	1.3395e-14	8.0059e-14	3.3929e-14
1.9000e+01	1.3659e-14	8.7997e-14	3.7476e-14
2.0000e+01	1.6036e-14	8.9038e-14	3.8374e-14

Time

Step	MatlabTime	HouseHolderTime	GivensTime
1.0000e+00	4.9080e-04	3.1997e-03	1.9427e-03
2.0000e+00	6.4000e-05	2.8640e-04	2.1270e-03
3.0000e+00	1.1940e-04	1.1588e-03	5.2338e-03
4.0000e+00	3.8260e-04	2.0988e-03	1.1409e-02
5.0000e+00	3.7310e-04	2.7923e-03	1.8298e-02
6.0000e+00	4.7420e-04	5.1587e-03	3.3829e-02
7.0000e+00	2.9860e-04	7.3735e-03	5.1767e-02
8.0000e+00	3.6030e-04	5.8211e-03	7.6674e-02
9.0000e+00	4.2170e-04	8.9769e-03	1.1799e-01
1.0000e+01	5.0450e-04	1.9253e-02	1.6896e-01
1.1000e+01	5.9290e-04	2.2320e-02	2.2632e-01
1.2000e+01	7.5430e-04	3.1813e-02	3.1113e-01
1.3000e+01	7.8120e-04	3.2227e-02	3.9427e-01
1.4000e+01	1.0107e-03	4.4737e-02	5.1868e-01
1.5000e+01	9.7740e-04	4.6324e-02	6.1885e-01
1.6000e+01	1.1332e-03	6.3896e-02	7.9505e-01
1.7000e+01	1.2450e-03	7.6051e-02	9.5793e-01
1.8000e+01	1.5570e-03	9.1609e-02	1.1847e+00
1.9000e+01	1.6697e-03	1.1809e-01	1.4096e+00
2.0000e+01	1.6314e-03	1.2580e-01	1.7808e+00

3.2 Secondo esperimento

Per eseguire i test, i dati iniziali sono: M, L, r, s, N.
M rappresenta la matrice dei dati, ovvero le immagini vettorizzate. L rappresenta il vettore delle luci, r e s le dimensioni delle immagini ed N la normale originale usata poi per confrontare con i risultati ottenuti.
A partire dalle matrici di immagini e luci, è stata ricavata la matrice delle normali della superficie dell'oggetto in questione. Per risolvere il problema sono stati utilizzati il metodo QR con le tre implementazioni viste precedentemente, e il metodo SVD di MATLAB, ovvero la funzione *pinv*, utilizzata per calcolare la pseudoinversa della matrice delle luci.

Tramite uno script esterno, le normali ottenute sono state elaborate in modo tale da produrre la *mesh* che rappresenta la superficie dell'oggetto considerato. Per ciascuna normale ottenuta, la procedura dello script esegue prima una rielaborazione della stessa, tramite differenziazione:

$$\tilde{\underline{n}} = \begin{bmatrix} \frac{\delta f}{\delta y} \\ \frac{\delta f}{\delta x} \\ 1 \end{bmatrix}$$

dove le prime due componenti vengono differenziate e la terza componente viene normalizzata.
Dopo la differenziazione, il profilo tridimensionale della superficie viene ricostruito tramite la risoluzione dell'equazione differenziale parziale di Poisson.

I test effettuati nel secondo esperimento misurano la qualità delle normali ottenute con i vari metodi.

3.2.1 Dataset

I due set sono molto diversi tra loro: il primo è un dataset di immagini sintetiche, il secondo è un dataset leggermente più grande e rappresenta un oggetto reale. Questo è per capire se la natura dei due set può influenzare la bontà del procedimento ma sopratutto se più elementi permettono una rappresentazione più accurata.

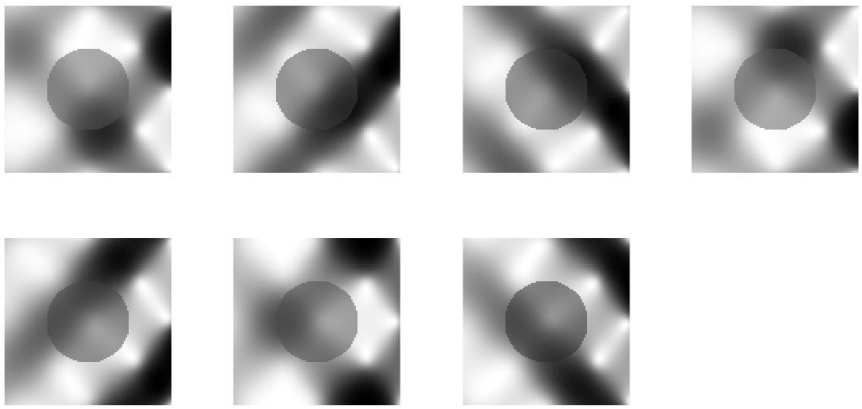


Figure 3: Dataset *superficie1*, 7 immagini 101×101

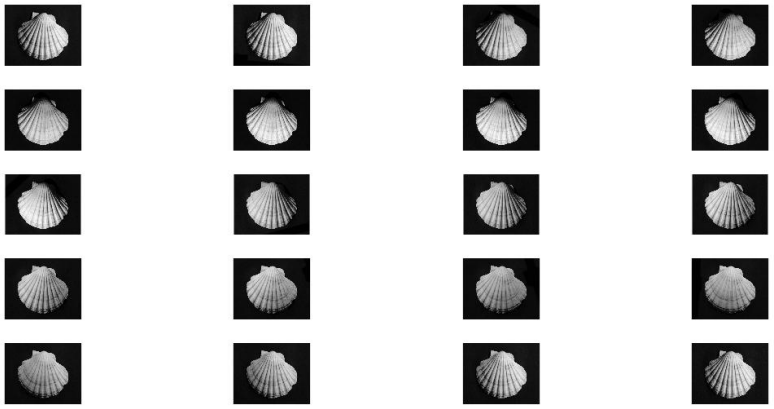


Figure 4: Dataset *conchiglia*, 20 immagini 885×505

3.2.2 Test QR

Nel test QR, il sistema è stato risolto calcolando sia la matrice trasposta di L che la matrice trasposta di M, in modo da ottenere il seguente sistema:

$$L^{\mathrm{T}} N = M^{\mathrm{T}}$$

Per ciascun metodo QR precedentemente considerato - Householder, Givens, MATLAB - è stata calcolata la fattorizzazione, utilizzata poi per risolvere il sistema in modo tale da ricavare la matrice N delle normali.
Gli errori ottenuti per ciascun metodo si mantengono sullo stesso ordine di magnitudine, ciò vale per entrambi i dataset utilizzati. In particolare gli errori calcolati sono: errore sulla fattorizzazione, errore assoluto della normale, relativo e relativo considerando come N corretta quella di MATLAB. Sono stati misurati anche i tempi per capire quale fosse il metodo più veloce. La stessa tipologia di errore è stata calcolata per la metodologia che prevede la risoluzione con SVD (prossimo test).

3.2.3 Test SVD

Nel test SVD, il sistema è stato risolto calcolando la matrice pseudoinversa di destra tramite la funzione *pinv* di MATLAB, la quale utilizza il metodo SVD. La matrice delle normali è stata ottenuta con la seguente procedura:

$$N^{\mathrm{T}} = M \times L^{+}$$
$$N = (N^{\mathrm{T}})^{\mathrm{T}}$$

In questo test è stata confrontata la risoluzione con SVD, quindi *pinv*, con i risultati precalcolati.

3.2.4 Errori calcolati

Dove è presente "NaN" sono valori che non aveva senso calcolare. Ad esempio errore "assoluto della normale" tra la Normale originale e se stessa.

Superficie1
Errori nelle normali superficie1

Tipo	Normale	HouseHolder	Givens	Matlab	SVD
'errore fattorizzazione'	2.5546e+00	2.7374e-15	3.1780e-15	2.8311e-15	3.7886e-15
'errore assoluto della normale'	NaN	6.4804e+02	6.4804e+02	6.4804e+02	6.4804e+02
'errore relativo della normale'	NaN	9.4702e-02	9.4702e-02	9.4702e-02	9.4702e-02
'errore relativo della normale matlab'	NaN	2.4746e-16	3.6016e-16	NaN	NaN
'tempi'	NaN	1.0953e-03	7.0450e-04	4.3090e-04	7.5208e-03

Conchiglia
Errori nelle normali conchiglia

Tipo	Normale	HouseHolder	Givens	Matlab	SVD
'errore fattorizzazione'	4.0735e+03	7.7817e+02	7.7817e+02	7.7817e+02	7.7817e+02
'errore assoluto della normale'	NaN	6.7876e+07	6.7876e+07	6.7876e+07	6.7876e+07
'errore relativo della normale'	NaN	1.1372e+02	1.1372e+02	1.1372e+02	1.1372e+02
'errore relativo della normale matlab'	NaN	4.4953e-16	2.1943e-16	NaN	NaN
'tempi'	NaN	2.2282e-01	2.1615e-01	1.8050e-01	2.6528e-02

Figure 5: *errori*, calcolati per i dataset superficie1 e conchiglia

3.2.5 Ricostruzioni 3D

Queste sono le ricostruzioni di superficie1 ottenute:

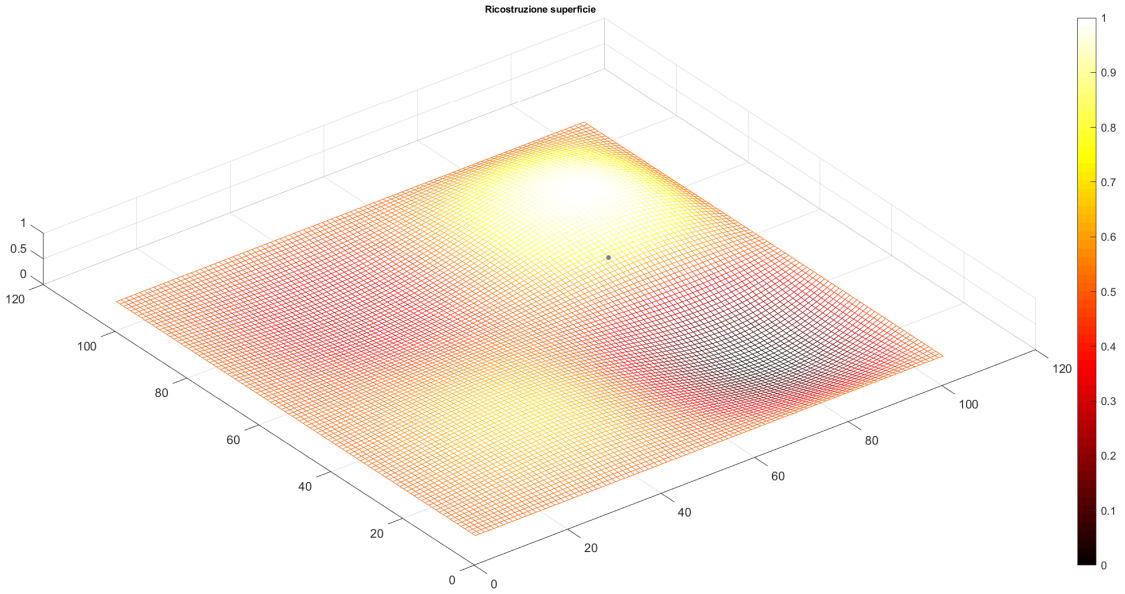


Figure 6: Superficie *superficie1*

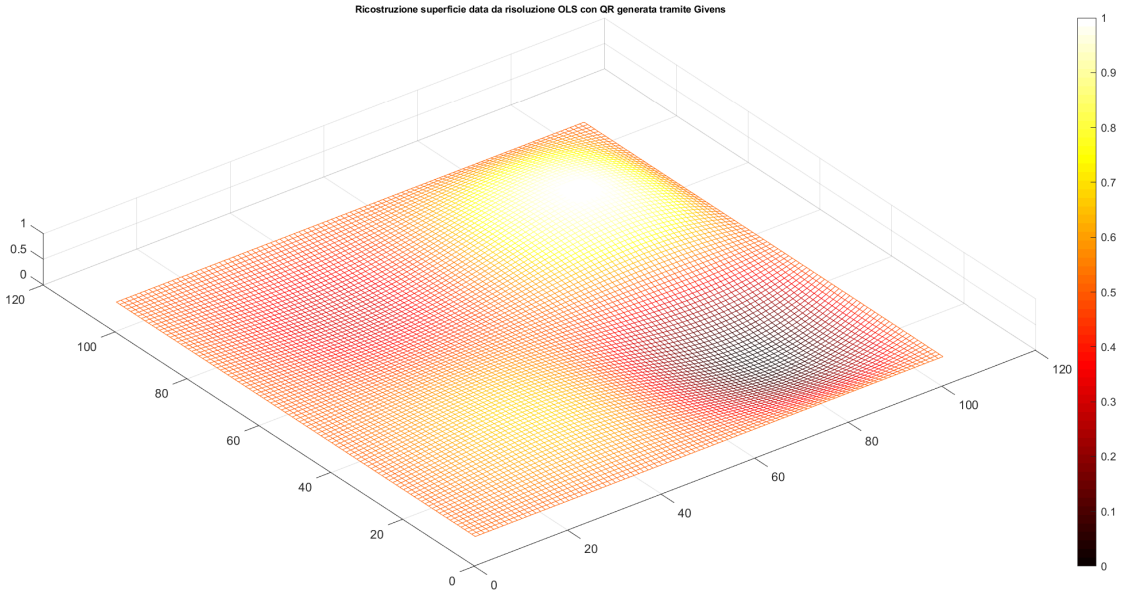


Figure 7: Ricostruzione di *superficie1* con fattorizzazione di Givens

Le ricostruzioni effettuate con i vari metodi, a livello visivo, non presentano differenze particolari, per cui sono state inserite soltanto la ricostruzione originale e quella effet-

tuata tramite fattorizzazione di Givens.
Per osservare la differenza tra la ricostruzione 3D originale e delle ricostruzioni 3D prodotte tramite normale ottenuti dai vari metodi, è stata introdotta anche la visualizzazione della differenza tra la *mesh* originale e la *mesh* sperimentale.

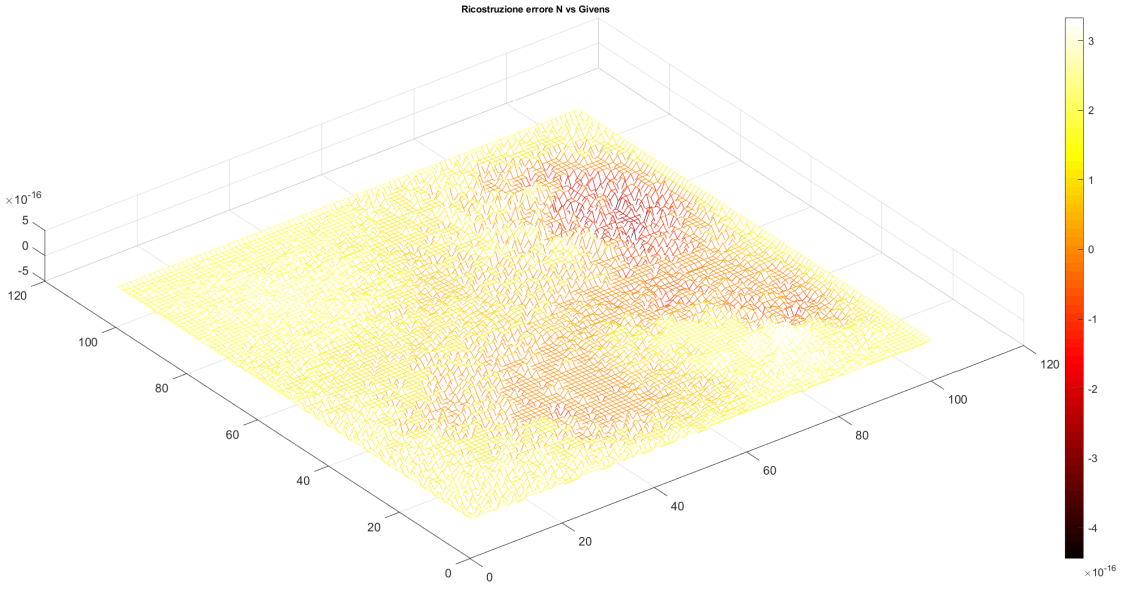


Figure 8: Superficie *superficie1*, differenza tra ricostruzione 3D con fattorizzazione di Givens e ricostruzioni 3D originali

Queste sono le ricostruzioni di conchiglia ottenute:

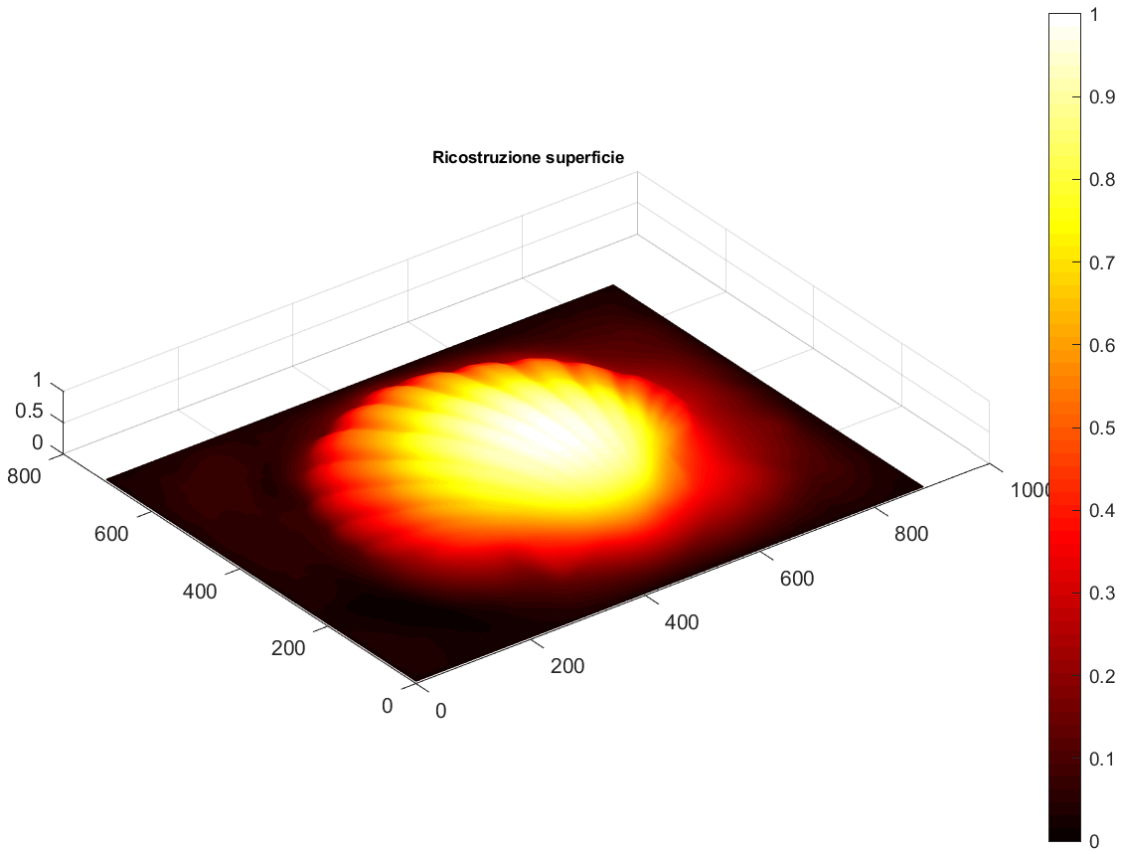


Figure 9: Superficie *conchiglia*

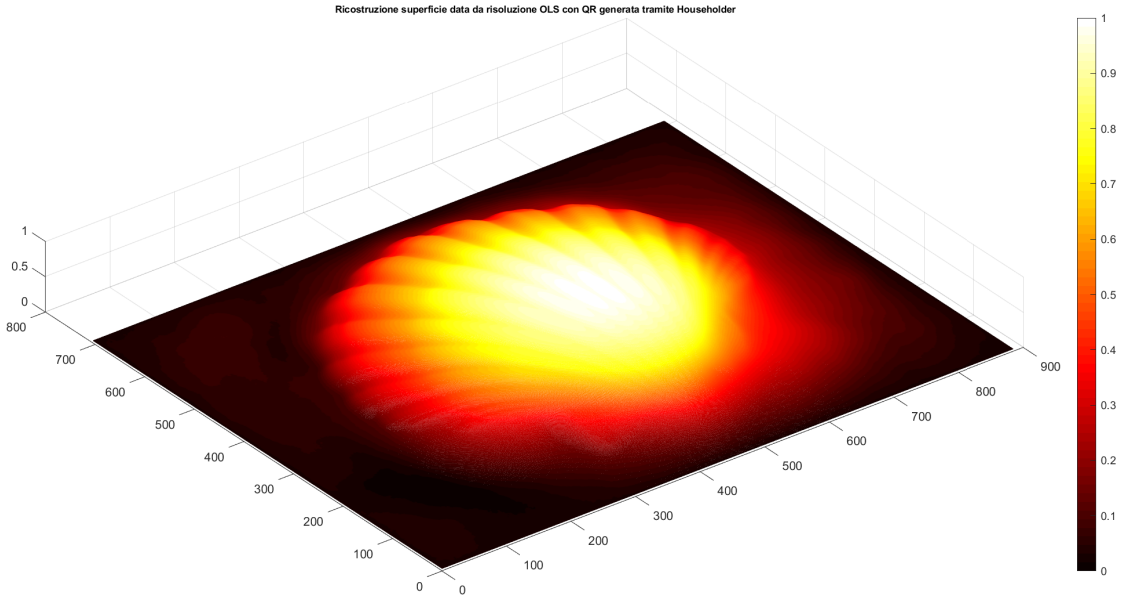


Figure 10: Superficie *conchiglia* con fattorizzazione di Householder

Anche in questo caso, le differenze tra i vari metodi utilizzati non sono così visivamente evidenti. Differenza tra ricostruzione 3D originale e ricostruzione 3D ottenuta dagli esperimenti con il metodo di Householder:

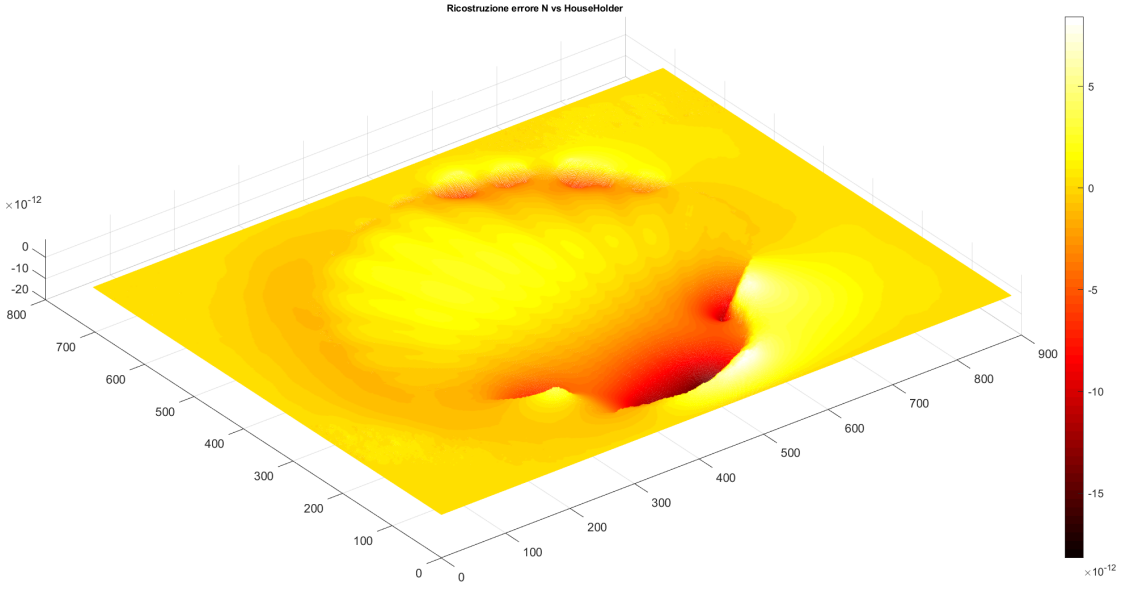


Figure 11: Superficie *conchiglia*, differenza tra ricostruzione 3D con fattorizzazione di Householder e ricostruzioni 3D originali

3.2.6 Osservazioni e analisi

Superficie1 Per quanto riguarda la fattorizzazione, il risultato migliore lo otteniamo con Householder ma non si discosta molto con gli le altre metodologie. Errore assoluto e relativo è costante in tutte e tre le metodologie, 6.4804×10^2 il primo e 9.4702×10^{-2} il secondo. Considerando la matrice MATLAB come quella ottimale, senza confrontarla con SVD, il metodo con la fattorizzazione QR di Householder è migliore rispetto a Givens ma anche in questo caso di poco. Per quanto riguarda i tempi, la metodologia data dalla fattorizzazione di MATLAB è la migliore tra le quattro. La più lenta la metodologia con Householder e, sempre per lentezza, viene seguita dal metodo con SVD. Le ricostruzioni 3D non presentano differenze in un primo momento, anche perché gli errori calcolati ottenuti sono molto bassi. Solo tramite la differenze delle ricostruzioni 3D si è potuto apprezzare qualche piccolo cambiamento tra essi.

Conchiglia Il dataset Conchiglia, presenta risultati lievemente differenti rispetto a Superficie1. L'errore di fattorizzazione è identico per Householder, Givens, MATLAB e SVD. Anche l'errore assoluto e l'errore relativo è uguale in queste metodologie. Vediamo qualche differenza considerando la matrice generata da MATLAB come quella ottimale: in questo caso è Givens ad ottenere risultati migliori ma si differenziano di $2.3010 \times 10_{-16}$ (4.4953×10^{-16} - 2.194×10^{-16}) quindi anche in questo caso non possiamo parlare di una differenza sostanziale. Per quanto riguarda i tempi il migliore è il metodo che prevede l'SVD: questo è un dato interessante ma ha una spiegazione molto semplice. Stiamo lavorando sulla matrice L che ha rango tre, dunque rango basso e questo permette alla funzione *pinv* di sfruttare la SVD troncata, risparmiando memoria e tempo. Con la funzione *pinv* è possibile indicare la tolleranza manualmente, ma avendola lasciata di default, MATLAB stesso si rende conto di quale algoritmo e quali fattori della matrice è meglio sfruttare. Anche in questo caso le differenze visive nella ricostruzioni 3D sono minime; anche calcolando la differenza tra mesh è difficile cogliere grande dettagli. In questo caso, è sicuramente grazie alla presenza di più foto rispetto all'altro dataset che rende il risultato migliore.

3.3 Conclusioni

Le metodologie non presentano grandi differenze per quanto riguarda gli errori parlando del secondo esperimento, sono sicuramente più evidenti nel primo esperimento. La grandezza dei dataset è un dato che influisce sulla bontà del risultato finale ma anche sul tempo perché ovviamente bisogna processare più immagini. Proprio il tempo è un elemento che differenzia le varie metodologie sia nel primo esperimento sia nel secondo ma entrambi sono accomunati dal fatto che nelle metodologie implementate da MATLAB, vengono sfruttati fattori che in una implementazione manuale (Householder e Givens presentate in questa tesina) possono essere estratti dalle matrici in maniera tale da ottenere prestazioni ottimali ed errori minimi.

4 Riferimenti

1. Identifying the lights position in photometric stereo under unknown lighting, A. Concas, R. Dessì, C. Fenu, G. Rodriguez, M. Vanzi

2. Photometric Stereo for 3D Mapping of Carvings and Relieves - Case Studies on Prehistorical Art in Sardinia, Massimo Vanzi and Carla Mannu and Riccardo Dessì

3. Recent improvements in photometric stereo for rock art 3D imaging, R. Dessì, C. Mannu, G. Rodriguez, G. Tanda, and M. Vanzi

4. Dataset *conchiglia* e *superficie1* forniti dal Professor Giuseppe Rodriguez

5. Script esterno *intnormals.m* fornito dal Professor Giuseppe Rodriguez