Università degli Studi di Padova

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Corso di Laurea Triennale in
Ingegneria Meccanica e Meccatronica

RILIEVO E RIPRODUZIONE DI OGGETTI TRIDIMENSIONALI

Relatore: Prof. Roberto Oboe

Laureando: Mattia Guidolin

Anno Accademico: 2013/2014

INDICE

1 INTRODUZIONE 1

2 TECNICHE DI MISURAZIONE 3D 3

2.1 Tecniche a contatto 4

2.2 Tecniche senza contatto attive 4

2.2.1 Tempo di volo 4

2.2.2 Triangolazione 5

2.2.3 Luce strutturata 6

2.2.4 Tecniche mediche 6

2.2.5 Tecniche industriali 7

2.2.6 Tecniche topografiche 7

2.3 Tecniche senza contatto passive 7

3 TRIANGOLAZIONE 9

4 SETUP E SOFTWARE UTILIZZATI 13

4.1 Setup 13

4.1.1 Pentax K200D 14

4.1.2 Laser 15

4.1.3 Sensoray Model 626 17

4.1.4 Motore a passo 17

4.1.5 Telecomando a infrarossi 18

4.2 Software 18

4.2.1 MATLAB 18

4.2.2 Simulink 19

4.2.3 MeshLab 21

5 PROCEDURA DI RILIEVO DELLA FORMA 3D 23

5.1 Calibrazione della posizione del fuoco 23

5.2 Lettura delle immagini 24

5.3 Calcolo delle coordinate dei punti 27

5.4 Definizione delle normali 33

5.5 Elaborazione dei dati in MeshLab 35

6 RISULTATI E CONCLUSIONI 39

Bibliografia 43

A APPENDICE A 45

A.1 calibraz_fuoco.m 45

A.2 fullscanner.m 51

1

INTRODUZIONE

I sistemi per l'acquisizione di dati 3D hanno raggiunto negli ultimi anni un elevato grado di precisione. Con lo sviluppo della tecnologia si riesce a processare un numero sempre maggiore di informazioni, fatto che ha reso possibile la creazione di modelli 3D sempre più complessi. Si possono ottenere superfici costituite da decine di milioni di facce.

Nel mercato si trovano un gran numero di scanner 3D, ognuno con dimensioni, ambiti di utilizzo, precisione e costi differenti. Si parte da scanner per uso domestico dal costo di alcune centinaia di euro, fino ad arrivare a scanner professionali da decine di migliaia di euro.

Scopo di questa tesi è dimostrare la possibilità di progettare e costruire uno scanner 3D economico, ottenendo comunque un buon livello di precisione, da poter utilizzare in ambito medico per la ricostruzione di modelli 3D di mani, dita, etc. e per il dimensionamento di protesi. La superficie ricostruita può inoltre essere utilizzata per effettuare stampe 3D. Ovviamente non sarà possibile raggiungere l'accuratezza o la velocità di acquisizione degli scanner in commercio, soprattutto per quanto riguarda i tempi di acquisizione, fortemente limitati dalla velocità di elaborazione e salvataggio dell'immagine della fotocamera.

Tra le varie tecniche conosciute per lo scanning 3D è stata scelta la triangolazione, a causa della semplicità del setup necessario. Gli unici elementi indispensabili per la realizzazione sono infatti una fotocamera ed un laser a linea.

Per ottenere uno scanner più accurato è possibile utilizzare un motore a passo per ruotare l'oggetto da analizzare, così da conoscerne con precisione la posizione. Il motore necessiterà quindi di un sistema di controllo, elemento che porta ad un aumento dei costi. Per lo scanner costruito in questa tesi è stato inoltre utilizzato un telecomando a infrarossi in modo da rendere automatica lo scatto delle foto.

Una volta acquisite le immagini è necessario effettuare alcune operazioni per ottenere il modello 3D dell'oggetto analizzato. Lo script per il calcolo delle coordinate è stato scritto in MATLAB, mentre per la ricostruzione della superficie si fa uso di MeshLab.

In questa tesi, oltre ad una breve esposizione delle tecniche di scanning 3D più utilizzate (Capitolo 2), soffermandosi sulla triangolazione (Capitolo 3), verrà descritto il setup utilizzato per i test in laboratorio (Capitolo 4). Ci si concentrerà quindi sul funzionamento dello script per la determinazione delle coordinate dei punti dell'oggetto e delle normali (Capitolo 5), informazioni necessarie per la ricostruzione del modello 3D. Verranno infine esposti i risultati ottenuti durante i test in laboratorio (Capitolo 6) ed il codice prodotto in MATLAB (Appendice A).

2

TECNICHE DI MISURAZIONE 3D

Lo scanner 3D è un dispositivo che analizza un oggetto fisico al fine di riprodurne un modello tridimensionale digitale. In generale possono essere individuati due momenti nel processo di ottenimento del modello 3D:

- acquisizione dei dati
- ricostruzione del modello poligonale in modo da ottenere una superficie continua.

Le informazioni ricavate dall'acquisizione dei dati permettono di ottenere una nuvola di punti (point cloud) che rappresenta le coordinate (x, y, z) dei punti costituenti la superficie dell'oggetto analizzato. Successivamente, utilizzando software per l'elaborazione di modelli 3D, a partire dalla nuvola di punti è possibile ricostruire una superficie continua.

Per fare ciò sono disponibili diverse tecniche di acquisizione dei dati, ognuna delle quali avente vantaggi, limitazioni e costi diversi, e vari software per l'elaborazione dei dati. Una volta ottenuto il modello 3D è possibile estrapolare le dimensioni dell'oggetto in esame, come ad esempio altezza, larghezza, volume. A causa delle diverse caratteristiche di ogni tecnica di acquisizione non è possibile determinare quale sia la migliore o la peggiore, ma la scelta dipende dall'ambito di utilizzo. Di seguito verranno analizzate le tecniche più utilizzate.

Le tecniche impiegate per la costruzione di scanner 3D si possono dividere in due categorie principali: a contatto e senza contatto. Le tecniche di acquisizione senza contatto possono a loro volta essere divise in tecniche attive, nel caso in cui vengano emesse radiazioni, e passive, nel caso in cui vengano sfruttate le radiazioni ambientali.

2.1 TECNICHE A CONTATTO

Gli scanner 3D a contatto, conosciuti anche come CCM (*Coordinate Measuring Machines*), determinano forma e dimensioni dell'oggetto analizzato attraverso il contatto fisico di una sonda. Il movimento della sonda può avvenire tramite un carrello munito di un braccio rigido, un braccio articolato oppure una combinazione dei due. Dalla conoscenza della posizione del braccio mentre è a contatto con l'oggetto è possibile determinare le coordinate dei punti della superficie. Si può ottenere un'elevata precisione nella misurazione.

Uno dei principali svantaggi però è proprio il contatto fisico con l'oggetto in esame necessario per effettuare la misurazione, il quale potrebbe venire modificato o danneggiato durante l'analisi, per esempio nel caso di materiali fragili o superfici molli (come quelle del corpo umano). Per questo motivo gli scanner basati su tecniche a contatto non sono adatti all'utilizzo in ambito medicale. Inoltre sono relativamente lenti, se paragonati ad altre tecniche di scanning 3D e molto costosi, in quanto necessitano di un elevato numero di sensori ad alta precisione per la determinazione della posizione del braccio durante la scansione.

2.2 TECNICHE SENZA CONTATTO ATTIVE

Gli scanner attivi emettono un qualche tipo di radiazione e ne misurano la riflessione per determinare la forma dell'oggetto in analisi. Generalmente i tipi di emissione utilizzati sono luce, ultrasuoni, raggi X.

2.2.1 *Tempo di volo*

Lo scanner a tempo di volo utilizza un fascio laser per misurare le distanze e determinare quindi la forma di superfici riflettenti al laser. Quest'ultimo è utilizzato per emettere un impulso che verrà riflesso dalla superficie in esame. Viene quindi cronometrato con elevata precisione il tempo in cui la luce riflessa è vista da un rilevatore. Dato che la velocità della luce è nota ($c = 299,792,458$ m/s), mediante l'informazione del tempo è possibile determinare la distanza percorsa dal fascio del laser, che è pari al doppio della

distanza della superficie dal sensore. Siano c la velocità della luce e t il tempo misurato dal sensore; il valore della distanza d può essere calcolato mediante la formula: $d = \frac{t \cdot c}{2}$. Per sondare l'intero campo è necessario ruotare il laser ed effettuare la misurazione del tempo per ogni punto di interesse. La direzione del laser può venire variata ruotando il diodo emettitore, o più convenientemente utilizzando degli specchi di rotazione, così da velocizzare l'operazione.

La principale limitazione di questo metodo è l'impossibilità di ottenere un'elevata precisione nella misurazione, in quanto essa dipende direttamente dall'accuratezza con cui il sensore è in grado di misurare il tempo (1 mm viene percorso in circa 3.3 ps). Generalmente la precisione varia da qualche millimetro a qualche centimetro.

Gli scanner a tempo di volo permettono però di coprire elevate distanze, anche dell'ordine di kilometri. Sono quindi indicati per la misurazione di grandi strutture, come ad esempio edifici o siti archeologici.

2.2.2 *Triangolazione*

La triangolazione è una tecnica che permette il calcolo delle distanze sfruttando le proprietà dei triangoli. Anche gli scanner che utilizzano il principio della triangolazione, così come gli scanner a tempo di volo, fanno uso di un laser per sondare l'oggetto. A differenza di questi ultimi però il laser viene impiegato per illuminare l'oggetto e per mezzo di una fotocamera si cattura un'immagine della posizione del punto (o del fascio) del laser proiettato sulla superficie. La posizione del laser (o la forma del fascio) nell'immagine catturata dal sensore è funzione della distanza dell'oggetto. Mediante più acquisizioni nelle quali viene variata la posizione del laser è possibile determinare la geometria completa dell'oggetto in esame.

Questa tecnica verrà analizzata in dettaglio nel Capitolo 3.

2.2.3 *Luce strutturata*

Gli scanner a luce strutturata proiettano uno schema noto sull'oggetto e ne determinano la profondità in base alla deformazione della griglia proiettata. I pattern, che generalmente consistono in linee parallele, vengono emessi utilizzando un proiettore LCD o altre sorgenti di luce analoghe. Una fotocamera viene utilizzata per catturare l'immagine della deformazione del pattern. Dalla misurazione dello scostamento tra le linee visualizzate sull'oggetto ed il pattern iniziale e conoscendo la geometria del sistema di acquisizione è possibile determinare le coordinate tridimensionali della superficie dell'oggetto in esame. Il processo è analogo a quello utilizzato negli scanner a triangolazione.

I vantaggi degli scanner a luce strutturata sono l'elevata velocità e precisione. Si possono infatti scannerizzare contemporaneamente centinaia di migliaia punti, se non l'intero campo di vista.

La qualità dei risultati dipende però fortemente dal livello di precisione con cui è possibile rilevare la deformazione del pattern proiettato sull'oggetto nell'immagine acquisita. È quindi necessario effettuare un'accurata calibrazione dello scanner prima di procedere all'acquisizione dei dati.

2.2.4 *Tecniche mediche*

La tomografia computerizzata, anche nota come TAC (tomografia assiale computerizzata) è una tecnica utilizzata in ambito medico per generare un'immagine tridimensionale dell'interno di un oggetto (generalmente organi umani) a partire da un grande numero di immagini bidimensionali ottenute emettendo raggi X e misurando la quantità di radiazione che lo attraversa. Questo permette di ottenere risultati insensibili ai fenomeni di riflessione, tipici degli scanner che fanno uso di laser o luce strutturata.

Il principale svantaggio di questo metodo è l'esposizione alle radiazioni dei raggi X, la cui dose deve essere la minima indispensabile per ottenere il risultato desiderato, al fine di limitare i possibili danni al paziente.

Un'altra tecnica molto utilizzata in ambito medico è la risonanza magnetica o MRI (Magnetic Resonance Imaging), la quale produce un maggiore contrasto tra i diversi tipi di tessuto del corpo, sulla base della loro composizione biochimica. È considerata non dannosa nei confronti del paziente, il quale non viene sottoposto a radiazioni ionizzanti (come ad esempio i raggi X).

Tra gli svantaggi dell'utilizzo di questa tecnica ci sono i costi elevati ed i tempi prolungati necessari per l'acquisizione delle immagini.

2.2.5 *Tecniche industriali*

La tomografia computerizzata, come anche la microtomografia, vengono utilizzate anche in ambito industriale per ottenere il modello digitale di un oggetto da utilizzare per effettuare test non distruttivi o particolari misurazioni.

2.2.6 *Tecniche topografiche*

La fotogrammetria è una tecnica di rilevamento 3D che permette la determinazione di forma e dimensioni di un oggetto mediante l'analisi di due fotogrammi della stessa scena ottenuti da punti diversi (fotogrammi stereometrici). A seconda della distanza e del campo di vista dei due sensori, questa tecnica può essere impiegata in molti ambiti, dalla microfotogrammetria utilizzata in laboratorio servendosi di stereomicroscopi alla fotogrammetria terrestre, dove le distanze possono essere dell'ordine di kilometri.

2.3 TECNICHE SENZA CONTATTO PASSIVE

Le tecniche passive non emettono alcun tipo di radiazione, ma si basano sul rilevamento delle radiazioni ambientali riflesse dall'oggetto che si vuole analizzare. Le radiazioni rilevate sono solitamente la luce o gli infrarossi.

I metodi passivi sono molto economici, questo perché nella maggior parte dei casi non necessitano di hardware specializzato, ma solamente di una fotocamera digitale.

Tra le tecniche passive più utilizzate troviamo:

- *Sistemi stereoscopici*: utilizzano due fotocamere che, da punti diversi, guardano la stessa scena. Analizzando le differenze tra le immagini catturate dalle due fotocamere è possibile determinare la distanza di ogni punto visualizzato. Questo metodo è basato sugli stessi principi della visione stereoscopica degli occhi.
- *Sistemi fotometrici*: utilizzano una sola fotocamera per catturare immagini multiple ottenute variando le condizioni di illuminazione, dalla cui analisi è possibile risalire al profilo dell'oggetto.
- *Silhouette techniques*: utilizzano le curve ottenute da una serie di fotografie attorno all'oggetto da esaminare, posizionato su uno sfondo ad alto contrasto. Queste sagome vengono poi combinate in modo da ottenere la superficie completa dell'oggetto. Con questo approccio non si riescono a rilevare alcune concavità.

3

TRIANGOLAZIONE

Il termine *triangolazione* deriva dalla posizione di laser, fotocamera e oggetto da analizzare, i quali formano appunto un triangolo. Una tipica configurazione di un sistema di rilevazione 3D che sfrutta la triangolazione è riportata in Figura 1 e prevede un laser utilizzato per proiettare un fascio sull'oggetto da analizzare, il quale viene fatto ruotare per mezzo di un motore in modo da evidenziarne le diverse zone. Il profilo illuminato dal laser viene quindi catturato con un sensore CCD (*Charge-Coupled Device*). Laser e sensore devono essere posti ad una distanza fissa e nota.

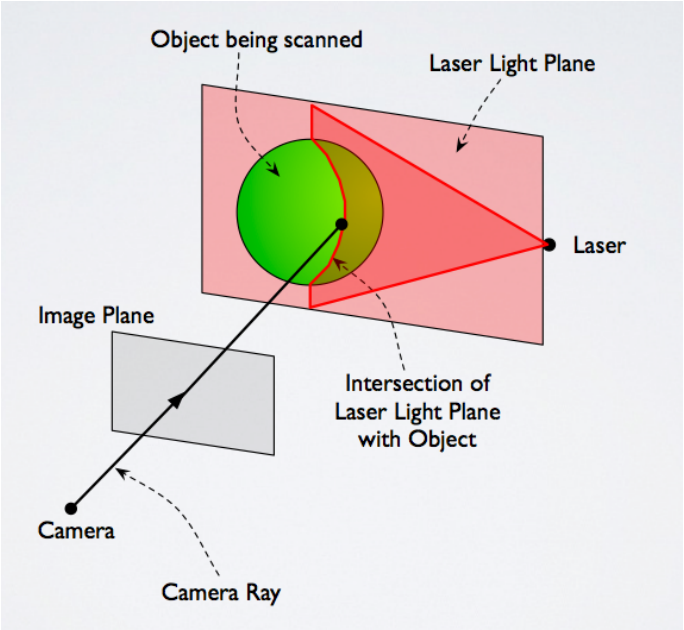

Figura 1: Configurazione tipica di un sistema di triangolazione per lo scanning 3D.

Per mezzo di relazioni trigonometriche è possibile determinare le coordinate (x, y, z) dei punti che costituiscono la superficie dell'oggetto scannerizzato.

Essendo le immagini catturate dalla fotocamera in 2 dimensioni, l'informazione ottenuta non è sufficiente a determinare le coordinate tridimensionali di un oggetto, in quanto sono vincolati solo 2 gradi di libertà. Il laser viene utilizzato per definire un piano Π noto che inibisce il terzo grado di libertà e permette la determinazione delle

coordinate del profilo nelle 3 dimensioni.

Per illuminare l'oggetto da analizzare si possono utilizzare laser a punto, ma generalmente si fa uso di laser a linea per illuminare molti punti contemporaneamente, così da velocizzare significativamente il processo di acquisizione.

Gli scanner che sfruttano la triangolazione possono coprire distanze dell'ordine di qualche metro, molto minori quindi di quelli a tempo di volo. Si riesce però ad ottenere un'elevata precisione nella misurazione, dell'ordine di qualche decina di micrometro. Sono inoltre molto economici, in quanto sono sufficienti una fotocamera ed un laser, fatto che li rende anche facilmente riconfigurabili in caso di necessità, così da adattarsi al tipo di superficie da analizzare. Questo li rende ideali per sondare oggetti di piccole dimensioni, tra cui anche arti, mani o piedi al fine di dimensionare eventuali protesi o effettuare simulazioni sul modello 3D.

Il principale difetto di questo metodo è la possibile presenza di zone d'ombra nelle immagini ottenute. A seconda delle condizioni ambientali (interferenza della luce) e delle caratteristiche dell'oggetto da analizzare (superfici molto oblique o riflettenti, presenza di gole o fessure) è possibile che il fascio laser non sia in grado di raggiungere tutti i punti di interesse [Figura 2]. Per limitare questo inconveniente è necessario posizionare laser e fotocamera più vicini possibile, accettando però in questo caso una diminuzione della precisione. In alternativa è possibile utilizzare 2 fotocamere posizionate diversamente l'una dall'altra [Figura 3], oppure effettuare più scansioni in condizioni differenti, per poi ottenere il profilo completo sovrapponendo i dati acquisiti mediante appositi software per l'editing 3D.

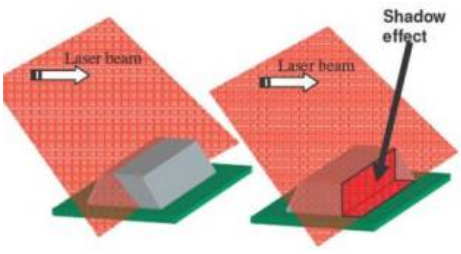

Figura 2: Esempio di zona d'ombra.

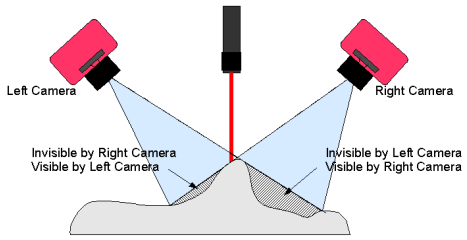

Figura 3: Utilizzo di due fotocamere per evitare zone d'ombra.

La costruzione di uno scanner 3D richiede una conoscenza accurata della geometria dell'obiettivo della fotocamera utilizzata per catturare le immagini che verranno successivamente analizzate.

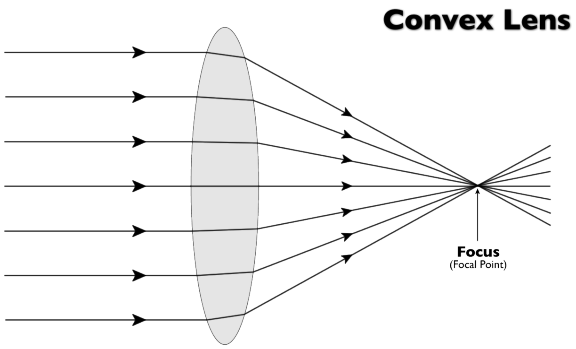

Figura 4: Fuoco di una lente convessa.

Il parametro principale di un obiettivo è la sua distanza focale (o lunghezza focale). Essa è la distanza tra il fuoco e la lente, espressa generalmente in mm. Il fuoco di una lente convessa è il punto in cui convergono tutti i raggi di luce paralleli rifratti dalla lente stessa [Figura 4].

La lente della fotocamera proietta solo una parte della scena nel sensore. Il campo di vista è determinato da due angoli che dipendono dalla distanza focale e dalle dimensioni del piano di proiezione [Figura 5].

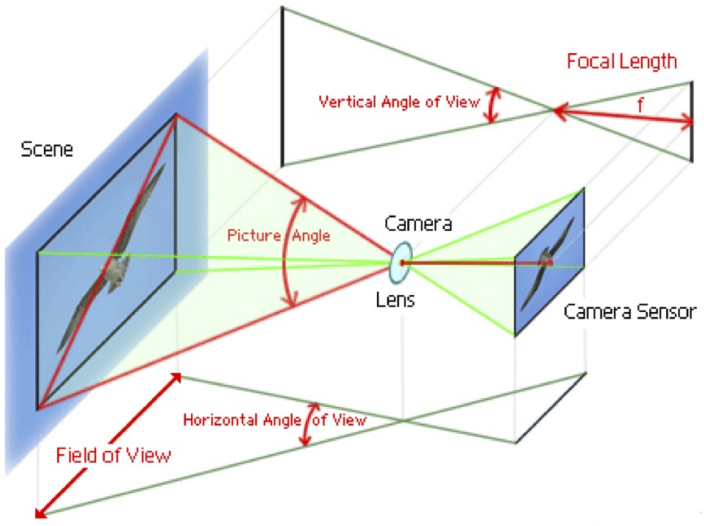

Figura 5: Schema di funzionamento delle lenti in fotografia.

Le informazioni necessarie per la determinazione delle coordinate (x, y, z) della superficie di un oggetto sono quindi:

- posizione del laser
- angolo di inclinazione del laser rispetto alla fotocamera
- posizione del fuoco della fotocamera
- distanza focale dell'obiettivo
- dimensioni e risoluzione del sensore CCD.

Con questi dati a disposizione è possibile definire l'equazione del piano Π in cui giace il fascio del laser e delle rette che collegano i punti del profilo nel piano immagine al fuoco della fotocamera. L'intersezione dei due determina la posizione dell'oggetto nello spazio tridimensionale.

Setup e procedimento utilizzati verranno descritti in dettaglio nei Capitoli 4 e 5.

4

SETUP E SOFTWARE UTILIZZATI

4.1 SETUP

Il setup utilizzato è composto da una reflex Pentax K200D munita di telecomando a infrarossi, un laser a linea ed un motore a passo. Sono stati inoltre utilizzati un generatore di corrente per il comando delle fasi del motore, un alimentatore ed un PC munito di Simulink per il controllo della rotazione del motore a passo e la sincronizzazione del segnale del telecomando a infrarossi. Il collegamento tra PC, motore e telecomando avviene con una scheda Sensoray Model 626.

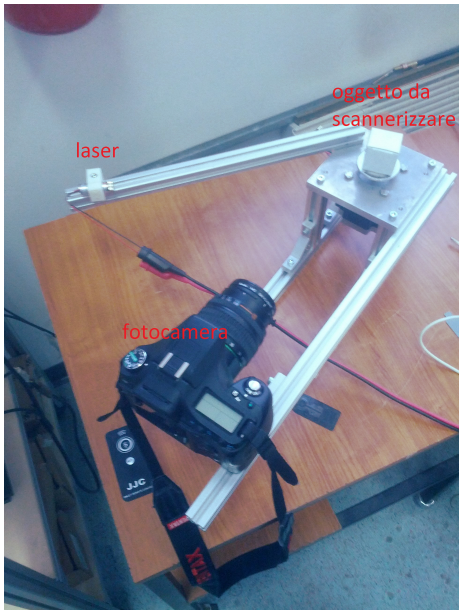

Figura 6: Setup utilizzato.

Il laser è impiegato per produrre il fascio di luce verticale con cui viene illuminato l'oggetto in esame ed è posizionato in modo da formare un angolo di 45° rispetto al piano della fotocamera, i cui parametri sono impostati in modo da ottenere immagini dove è chiaramente visibile solo il profilo dell'oggetto illuminato dal laser [Figura 16 Capitolo 5].

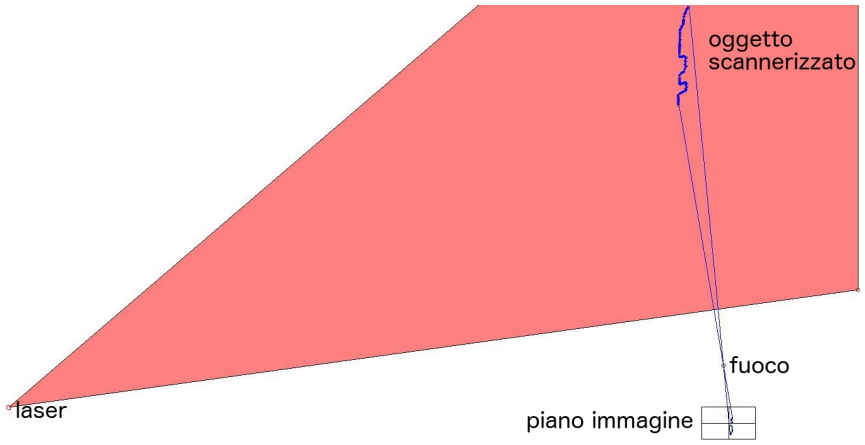

Figura 7: Setup schematizzato in MATLAB.

Il motore è utilizzato per ruotare l'oggetto, così da poter illuminare con il laser ogni punto della superficie, e ad ogni passo, pari ad 1.8° nel caso in esame, il telecomando a infrarossi dà segnale alla macchina fotografica di scattare una foto. Il motore a passo è alimentato da due generatori di corrente collegati ad un alimentatore, utilizzato anche per alimentare il laser. Il motore è comandato in *full step drive* utilizzando Simulink, software usato anche per sincronizzare il segnale del telecomando a infrarossi, comandato con un relè, alla rotazione dell'oggetto, così da ottenere lo scatto di un'immagine per ogni passo del motore in modo automatico. La comunicazione tra computer e motore avviene mediante una scheda Sensoray Model 626.

Il ciclo per ogni immagine è quindi: rotazione dell'oggetto di un passo (1.8°), scatto della foto. Per ottenere il profilo completo dell'oggetto è necessario acquisire un numero di immagini pari a $360/1.8 = 200$. Il ciclo va quindi ripetuto 200 volte, ottenendo un totale di 200 immagini.

4.1.1 *Pentax K200D*

La K200D [19] è una reflex prodotta dall'azienda giapponese Pentax nel 2008.

È dotata di un sensore CCD da 23.5 x 15.7 mm con una risoluzione di 10.2 megapixel. Le dimensioni delle immagini ottenute sono di 3872 x 2592 pixel. Ogni pixel è grande circa 6.06 micrometri. La distanza focale della lente utilizzata è variabile tra 18 e 55 mm.

Figura 8: Pentax K200D, visione frontale.

Figura 9: Pentax K200D, visione dall'alto.

4.1.2 *Laser*

Il laser (acronimo di ***L**ight **A**mplification by **S**timulated **E**mission of **R**adiation*) è un dispositivo in grado di emettere un fascio di luce coerente, monocromatica e unidirezionale. Analizzando più nel dettaglio queste caratteristiche si ha:

- unidirezionalità: la luce emessa si propaga in una direzione ben definita, a differenza delle sorgenti elettromagnetiche tradizionali, le quali emettono in tutte le direzioni;
- monocromaticità: la radiazione emessa presenta sempre la stessa frequenza;
- coerenza: mentre nell'emissione spontanea ogni fotone viene emesso in maniera casuale, nel laser il fascio di luce è composto da fotoni con uguale frequenza e fase che si sommano quindi l'una con l'altra producendo grandi intensità e potenza.

Queste proprietà consentono l'impiego del laser in una grande varietà di applicazioni, come ad esempio lavorazione di materiali, misurazioni industriali, utilizzo nelle fibre ottiche e in ambito medico. L'elevata potenza concentrata in un'area ristretta, permette ai laser di essere impiegati per il taglio e la saldatura di metalli; monocromaticità e coerenza vengono sfruttate per utilizzare i laser come strumenti di misura delle distanze, così come per trasportare informazioni nelle fibre ottiche anche per tratti considerevoli.

La lunghezza d'onda dei laser può variare da 200 nm a 700 nm passando dall'ultravioletto al visibile all'infrarosso. La lunghezza d'onda dei laser utilizzati generalmente per gli scanner 3D a triangolazione è di 635 nm, corrispondente al colore rosso.

Rischi nell'uso dei laser e classificazione

Il contatto con la luce emessa dal laser può provocare disturbi alla pelle e agli occhi. È importante quindi conoscere il livello massimo di radiazione al quale possono essere esposti questi organi, al fine di evitare l'insorgere di potenziali danni.

La grande varietà di lunghezze d'onda, energie e caratteristiche d'impulso dei laser rendono indispensabile, ai fini della sicurezza, il loro raggruppamento in classi di pericolosità. A tale scopo è stato introdotto un parametro chiamato Limite di Emissione Accettabile (LEA), che descrive i livelli di radiazione emessi da un sistema contenente laser, la cui determinazione deve essere effettuata nelle condizioni di utilizzo più sfavorevoli ai fini della sicurezza.

Sono state definite 5 classi, con indice di pericolosità crescente:

- classe 1: (<0.04 mW) laser intrinsecamente sicuri, il livello di esposizione massima permesso non viene mai superato.
- classe 2: (<1 mW) sorgenti che emettono radiazione nell'intervallo tra 400 nm e 700 nm (cioè nel visibile) a bassa potenza, normalmente non in grado di arrecare danni alla vista.
- classe 3A: (<5 mW) laser sicuri per la visione ad occhio nudo. Tuttavia possono danneggiare la vista se visualizzati attraverso strumenti ottici come binocoli, telescopi, microscopi.
- classe 3B: (<500 mW) la visione diretta del fascio di questi laser è sempre pericolosa.
- classe 4: (>500 mW) sono i laser più potenti e pericolosi. Possono causare lesioni alla pelle e costituire pericolo di incendio. È pericolosa l'esposizione anche al solo raggio diffuso.

L'ordinanza 16 luglio 1998 pubblicata nella Gazzetta Ufficiale n. 167 del 20 luglio 1998 vieta, su tutto il territorio nazionale, la commercializzazione di puntatori laser o di oggetti con funzione di puntatori laser di classe pari o superiore III (>1 mW), secondo la norma CEI EN 60825; lo stesso provvedimento è emanato nell'Unione Europea dalla

direttiva CEI EN 60825/1 e negli Stati Uniti d'America dal "Chapter 21 CFR (the Code of Federal Regulations)".

4.1.3 *Sensoray Model 626*

La Sensoray Model 626 [20] è una scheda multifunzione di input/output analogico/digitale. Sono presenti 16 canali di input analogici con risoluzione di 16 bit, ognuno dei quali può venire programmato per lavorare tra ±5V o ±10V, 4 output analogici di tensione, con risoluzione di 14 bit, che possono lavorare nel range di ±10V e 48 canali di input/output digitali.

Collegandola in modo da avere in input un computer e in output i generatori di corrente utilizzati per alimentare le fasi motore a passo, è possibile effettuare il controllo di quest'ultimo utilizzando Simulink.

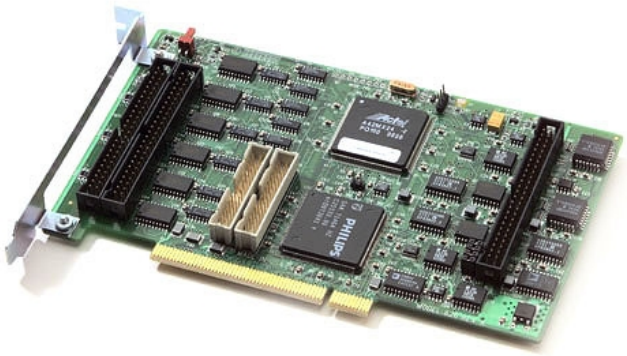

Figura 10: Sensoray Model 626.

4.1.4 *Motore a passo*

Il motore utilizzato è il modello 60SH65-86, un motore a passo ibrido costruito dall'azienda indiana Ark Motion Controls [11], sul cui rotore è stato montato un piatto in modo da sostenere l'oggetto da scannerizzare. L'angolo di passo è di 1.8°, con un'accuratezza del 5%.

4.1.5 *Telecomando a infrarossi*

Per lo scatto delle immagini da remoto è stato utilizzato un telecomando a infrarossi per fotocamere Pentax. Il normale funzionamento prevede la pressione di un tasto per dare il comando alla fotocamera. Per automatizzare il processo di acquisizione delle immagini il telecomando è stato modificato saldando un relè alla scheda, così da poterlo controllare utilizzando Simulink e bypassando la pressione manuale del tasto.

Figura 11: Telecomando a infrarossi per fotocamere Pentax.

4.2 SOFTWARE

I software impiegati per la scrittura dello script di analisi delle immagini, il controllo del motore a passo e l'elaborazione del modello 3D ottenuto dallo scan sono:

- MATLAB [16]
- Simulink [17]
- MeshLab [18].

4.2.1 *MATLAB*

MATLAB (abbreviazione di Matrix Laboratory) è un linguaggio di alto livello e un ambiente interattivo creato alla fine degli anni '70 da Cleve Moler che permette di effettuare calcoli, manipolare matrici, analizzare dati ed implementare algoritmi, oltre ad essere in grado di interfacciarsi con numerosi altri programmi. Il linguaggio e le funzioni incorporate consentono una programmazione più immediata rispetto ai linguaggi tradizionali,

come ad esempio Java o C++, pur mantenendo una sintassi simile.

Il codice realizzato per la determinazione delle coordinate della superficie dell'oggetto in esame, la definizione delle normali e la calibrazione della posizione del fuoco della fotocamera è stato scritto utilizzando MATLAB.

4.2.2 *Simulink*

Simulink è un ambiente grafico che lavora a stretto contatto con MATLAB e supporta la progettazione a livello di sistema, la simulazione, la generazione automatica del codice, il testing e la verifica di sistemi embedded. Viene utilizzato in svariati ambiti, dal model-based design ai sistemi di controllo, per l'elaborazione di segnali sia analogici che digitali e per lo sviluppo e la verifica di sistemi meccatronici.

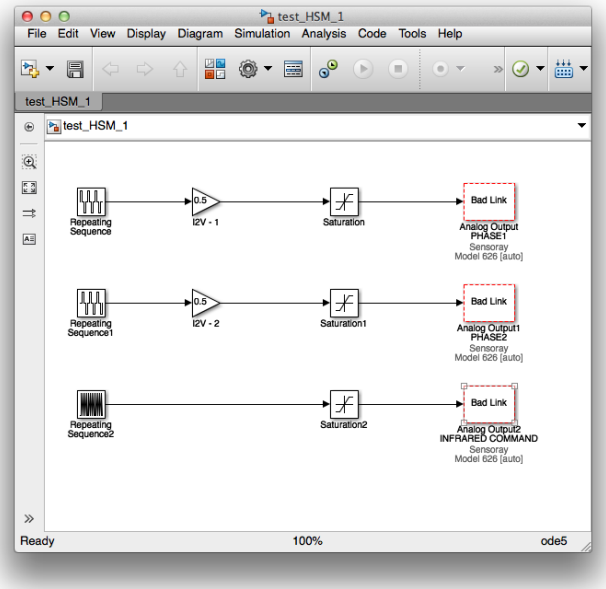

Figura 12: Controllo di motore a passo e telecomando a infrarossi in Simulink.

Per effettuare il controllo del motore a passo e la sincronizzazione del segnale del telecomando a infrarossi della fotocamera si è fatto uso di Simulink.

La generazione dei riferimenti di corrente avviene con 2 blocchi *Repeating Sequence*, uno per ogni fase del motore. Comandandoli con i valori, per la fase 1:

- *Time values:* [0 0 1 1 2 2 3 3 4]
- *Output values:* [0 1 1 0 0 -1 -1 0 0]

e per la fase 2:

- *Time values:* [0 1 1 2 2 3 3 4 4]
- *Output values:* [0 0 1 1 0 0 -1 -1 0]

corrispondenti alle Figure 13 e 14, il motore effettuerà 4 scatti di 1.8° (angolo di passo) per ogni periodo di ripetizione delle sequenze (il primo passo all'istante 1, il secondo all'istante 2, il terzo all'istante 3, il quarto all'istante 4). Modificando i valori di *Time values* è possibile regolare la velocità di rotazione del motore. I riferimenti di tensione generati nel blocco *Repeting Sequence* relativo ad ogni fase passano successivamente per un blocco *Saturation*, al fine di evitare possibili danni al motore. La scheda Sensoray Model 626 produrrà quindi 2 riferimenti di tensione pari a quelli delle 2 *Repeating Sequence*, i quali comanderanno i 2 generatori di corrente collegati alle 2 fasi del motore a passo, producendo il movimento voluto.

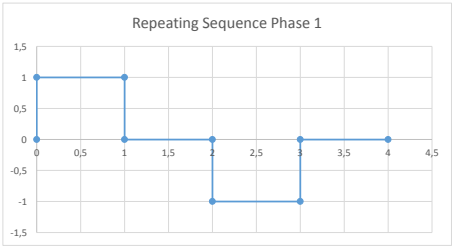

Figura 13: Repeating Sequence Phase 1.

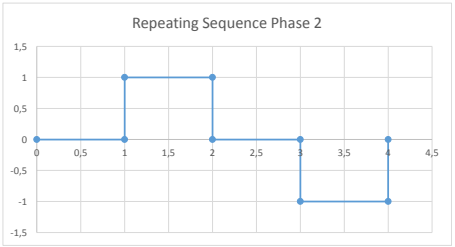

Figura 14: Repeating Sequence Phase 2.

Analogamente il relè saldato al telecomando a infrarossi è comandato con i valori:

- *Time values:* [0 0.5 0.5 1.5 1.5 2.5 2.5 3.5 3.5 4]
- *Output values:* [0 0 1 0 1 0 1 0 1 0].

In questo modo la fotocamera scatterà una foto per ogni passo del motore.

4.2.3 *MeshLab*

MeshLab è un software open source molto utilizzato nel campo della modellazione 3D e nel trattamento di grosse quantità di dati come coordinate o nuvole di punti. Il suo scopo è quello di trattare, organizzare e processare i dati generati dagli scanner 3D, offrendo strumenti per modificare, pulire e semplificare le nuvole di punti ottenute. Il software è stato sviluppato in Italia nel centro di ricerca ISTI – CNR per l'Università di Pisa nel 2005.

Per la visualizzazione dei dati ottenuti dallo script in MATLAB e la successiva ricostruzione della superficie dell'oggetto di test è stato utilizzato MeshLab.

5

PROCEDURA DI RILIEVO DELLA FORMA 3D

Utilizzando dei supporti metallici, fotocamera e laser sono fissati in modo da conoscerne l'esatta distanza dal centro di rotazione del motore a passo (e quindi dell'oggetto che viene fatto ruotare su di esso). Conoscendo la distanza focale dell'obiettivo della macchina fotografica è possibile determinare univocamente la posizione dei punti del profilo dell'oggetto.
Il processo è suddiviso in 4 fasi:

- calibrazione della posizione del fuoco della fotocamera
- lettura delle immagini
- calcolo delle coordinate dei punti
- definizione delle normali.

5.1 CALIBRAZIONE DELLA POSIZIONE DEL FUOCO

Per la determinazione delle coordinate 3D dell'oggetto di test è necessaria la conoscenza delle posizioni del laser e del fuoco della fotocamera rispetto ad un punto di riferimento qualsiasi (nel caso in esame come punto di riferimento è stato scelto il centro di rotazione del motore). Ciò richiederebbe la conoscenza precisa della struttura interna della fotocamera utilizzata. In assenza di questa informazione è possibile comunque calcolare una stima della posizione del fuoco.

Ipotizzando che il fuoco della macchina fotografica si trovi in un punto appartenente all'asse passante per il centro della lente in utilizzo, l'unica incognita da determinare è la distanza nella coordinata y, in quanto le coordinate in x e z sono misurabili sperimentalmente [Figura 15].

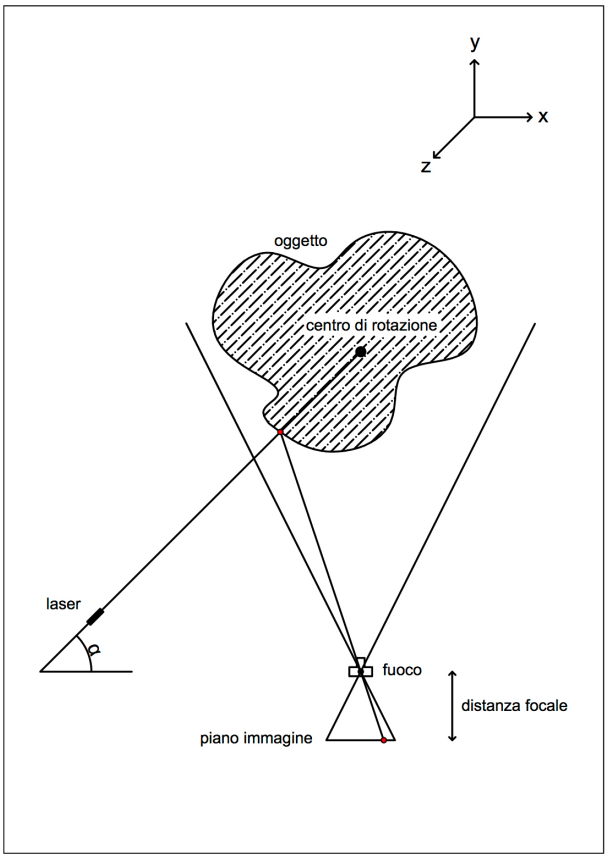

Figura 15: Schema utilizzato per il calcolo delle coordinate 3D.

Altezza, larghezza e profondità del modello 3D dell'oggetto in esame sono funzione della posizione del fuoco. Utilizzando un oggetto di test di cui si conoscono le dimensioni è possibile determinare la posizione del fuoco della macchina fotografica. Ponendo un valore di primo tentativo per la distanza tra fuoco e centro di rotazione ragionevole, effettuando una scansione e successivamente imponendo che l'altezza del modello 3D corrisponda a quella dell'oggetto di test, si ottiene come risultato la distanza effettiva del fuoco dal centro di rotazione. Queste operazioni vengono effettuate nello script MATLAB *calibraz_fuoco.m*.

5.2 LETTURA DELLE IMMAGINI

I parametri della fotocamera devono essere impostati in modo da ottenere immagini in cui sia chiaramente visibile solo il profilo del laser proiettato sulla superficie dell'ogget-

to in esame [Figura 16]. Per ottenere questo risultato è stato impostato per il tempo di apertura del diaframma un valore di 1/60 di secondo. Il valore ottimale necessario ad ottenere immagini in cui sia chiaramente distinguibile il profilo del laser dallo sfondo dipende dalle condizioni di luce e va determinato caso per caso.

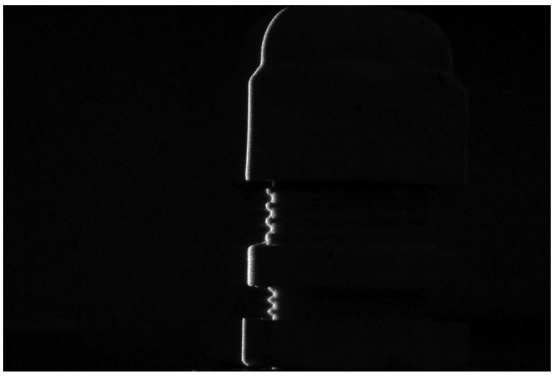

Figura 16: Immagine originale.

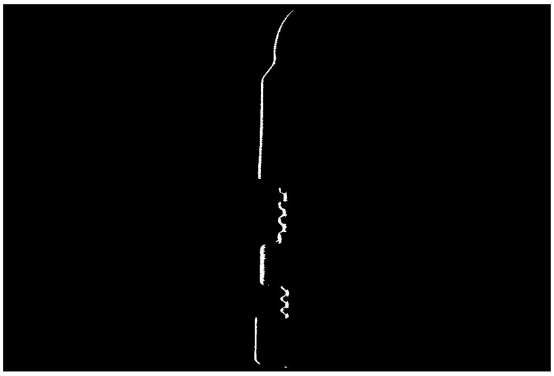

Figura 17: Immagine binaria

Utilizzando MATLAB, l'immagine originale viene convertita prima in una a scala di grigi e successivamente in una binaria, dove il profilo del laser corrisponde ai pixel bianchi (1) ed il resto dell'immagine è completamente nero (0).

Sia $I_{originale}$ l'immagine originale convertita in scala di grigi e $I_{originale}(x)$ un singolo pixel dell'immagine. Per convertirla in binario bisogna prima scegliere un livello di soglia (nel caso in esame pari a 0.5), e successivamente effettuare l'operazione:

$$I_{binaria}(x) = \begin{cases} 0 \text{ se } I_{originale}(x) < soglia \\ 1 \text{ se } I_{originale}(x) \geqslant soglia \end{cases}$$

Essendo la risoluzione della fotocamera pari a 3872×2592 pixel, l'immagine binaria corrisponde ad una matrice di dimensioni 3872×2592, nella quale ogni elemento corrisponde ad un pixel. 0 indica la posizione di un pixel nero, mentre 1 indica la posizione di un pixel bianco [Figura 17].

Procedendo quindi dall'alto verso il basso, viene calcolata la media della posizione dei pixel bianchi, riga per riga, in modo da ottenere un profilo lineare [Figure 18, 19].

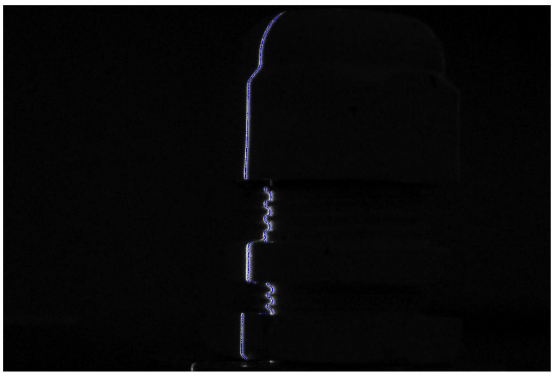

Figura 18: Immagine originale con evidenziato in blu il profilo della media.

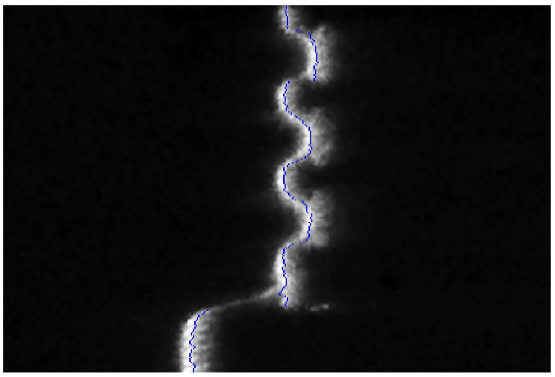

Figura 19: Particolare del profilo della media.

Al fine di minimizzare la complessità computazionale, e quindi incrementare la velocità di esecuzione dello script, è opportuno non analizzare l'immagine nel suo complesso, ma limitare l'area di analisi alla zona occupata dal profilo illuminato dal laser [Figura 20]. Essendo le dimensioni delle foto acquisite pari a 3872×2592 pixel, l'analisi completa richiederebbe $3872 \cdot 2592 \approx 10$ milioni di iterazioni (1 per ogni pixel) per ogni immagine. Un altro vantaggio del limitare l'area di analisi dell'immagine, e quindi dichiarare i pixel superiore e inferiore (che indicano l'altezza massima dell'oggetto in analisi e la

posizione della base su cui è appoggiato) permette di minimizzare le discontinuità tra le immagini acquisite, in modo da ottenere risultati migliori nella successiva generazione della mesh, soprattutto nel caso in cui la sommità dell'oggetto sia piatta.

Siano: $first\_row$, num_row, $first\_column$, num_column rispettivamente i valori dei pixel che indicano la prima e l'ultima riga e la prima e l'ultima colonna da analizzare in ogni immagine.

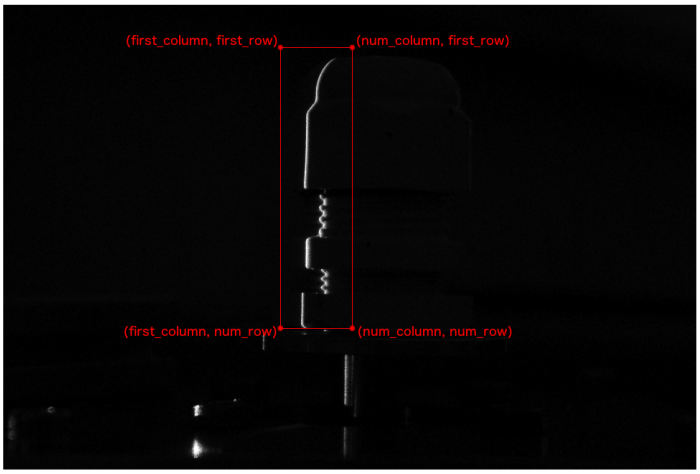

Figura 20: Delimitazione dell'area da analizzare all'interno di un'immagine acquisita dalla fotocamera.

Procedendo da $first\_row$ fino a num_row, lo script riconosce la posizione dei pixel bianchi (cioè quelli illuminati dal laser) riga per riga (row) e ne calcola la media. L'informazione viene quindi memorizzata all'interno della variabile $edge$ di dimensioni $(num\_row - first\_row)$. La coppia di valori $(edge(row), row)$ indica la posizione nel piano immagine dei pixel che formano il profilo medio calcolato [Figure 18, 19].

5.3 CALCOLO DELLE COORDINATE DEI PUNTI

Le immagini catturate dalla fotocamera sono in 2 dimensioni, l'informazione ottenuta vincola quindi solo 2 gradi di libertà. Ciò non è sufficiente a determinare le coordinate 3D di un punto, in quanto è ancora presente un grado di libertà (∞^1 soluzioni). Il laser viene utilizzato per definire un piano Π noto. Esso inibisce il terzo grado di libertà e

permette il calcolo delle coordinate del profilo nelle 3 dimensioni.

Conoscendo la posizione del fuoco (informazione ottenuta dalla calibrazione) e la distanza focale è possibile calcolare le coordinate dei punti del profilo nel piano immagine, rispetto al centro di rotazione dell'oggetto. Esse valgono:

$$\begin{cases} xk = xfuoco - \frac{\frac{sensorwidth}{2} - edge(row)}{tara\_x} \\ yk = yfuoco - focaldist \\ zk = zfuoco + \frac{\frac{sensorheight}{2} - row}{tara\_z} \end{cases}$$

dove $(xfuoco, yfuoco, zfuoco)$ sono le coordinate della posizione del fuoco della fotocamera rispetto all'asse di rotazione del motore, $focaldist$ la distanza focale della lente utilizzata, $sensorwidth$ e $sensorheight$ rispettivamente larghezza e altezza del sensore in pixel, $tara\_x = \frac{larghezza\_sensore\_in\_mm}{larghezza\_sensore\_in\_pixel}$ e $tara\_z = \frac{altezza\_sensore\_in\_mm}{altezza\_sensore\_in\_pixel}$.

A questo punto è possibile determinare le equazioni del piano in cui giace il fascio del laser e delle rette che collegano i punti del profilo nel piano immagine al fuoco. La loro intersezione determina il punto nello spazio tridimensionale corrispondente alla superficie dell'oggetto [Figura 15].

Sia $(xlaser, ylaser, zlaser)$ la posizione del laser rispetto all'asse del motore e α l'angolo che forma rispetto al piano della fotocamera [Figura 15]. Per determinare l'equazione del piano definito dal fascio del laser sono necessari altri 2 punti qualsiasi appartenenti al piano. Siano $(x2, y2, z2)$ e $(x3, y3, z3)$ le coordinate di questi punti. È ora possibile calcolare l'equazione del piano. Essa vale:

$$\Pi : l \cdot (x - xlaser) + m \cdot (y - ylaser) + n \cdot (z - zlaser) = 0$$

dove:

$$l = \begin{vmatrix} y2 - ylaser & z2 - zlaser \\ y3 - ylaser & z3 - zlaser \end{vmatrix}, m = -\begin{vmatrix} x2 - xlaser & z2 - zlaser \\ x3 - xlaser & z3 - zlaser \end{vmatrix}, n = \begin{vmatrix} x2 - xlaser & y2 - ylaser \\ x3 - xlaser & y3 - ylaser \end{vmatrix}$$

Sia $\Pi : a \cdot x + b \cdot y + c \cdot z - d = 0$ questo piano. a, b, c, d sono fissi e noti.

Siano (xf, yf, zf) le coordinate del fuoco della macchina fotografica e (xk, yk, zk) le coordinate di un singolo punto del profilo nel piano immagine definite precedentemente. Le equazioni della retta r passante per i 2 punti sono:

$$r : \begin{cases} \frac{x-xf}{xf-xk} = \frac{y-yf}{yf-yk} \\ \\ \frac{x-xf}{xf-xk} = \frac{z-zf}{zf-zk} \end{cases}$$

Dopo alcuni passaggi si ottiene:

$$r : \begin{cases} (yf-yk)\cdot x-(xf-xk)\cdot y-(yf\cdot xk-xf\cdot yk)=0 \\ \\ (zf-zk)\cdot x-(xf-xk)\cdot z-(zf\cdot xk-xf\cdot zk)=0 \end{cases}$$

L'intersezione del piano del laser e delle rette che collegano il fuoco ai punti del profilo nel piano immagine determina il valore delle coordinate (x, y, z) effettive dell'oggetto in esame. Per determinare il punto di intersezione è sufficiente risolvere il sistema lineare contenente l'equazione del piano Π e le 2 equazioni della retta r. Si ottiene quindi:

$$\begin{cases} a\cdot x+b\cdot y+c\cdot z-d=0 \\ \\ (yf-yk)\cdot x-(xf-xk)\cdot y-(yf\cdot xk-xf\cdot yk)=0 \\ \\ (zf-zk)\cdot x-(xf-xk)\cdot z-(zf\cdot xk-xf\cdot zk)=0 \end{cases}$$

Essendo MATLAB ottimizzato per eseguire calcoli sfruttando le matrici, per risolvere il sistema viene utilizzato il metodo di Cramer.

Siano per comodità di notazione $a1 = (xf-xk), b1 = (yf-yk), c1 = (zf-zk), d1 = (xk\cdot yf-yk\cdot xf), e1 = (xk\cdot zf-zk\cdot xf)$. La matrice completa del sistema è:

$$\begin{pmatrix} a & b & c & d \\ \\ b1 & -a1 & 0 & d1 \\ \\ c1 & 0 & -a1 & e1 \end{pmatrix}$$

I valori delle coordinate nelle 3 dimensioni del punto della superficie dell'oggetto a cui è associata la retta r sono quindi:

$$xxk = \frac{\det\begin{pmatrix} d & b & c \\ d1 & -a1 & 0 \\ e1 & 0 & -a1 \end{pmatrix}}{\det\begin{pmatrix} a & b & c \\ b1 & -a1 & 0 \\ c1 & 0 & -a1 \end{pmatrix}}$$

$$yyk = \frac{\det\begin{pmatrix} a & d & c \\ b1 & d1 & 0 \\ c1 & e1 & -a1 \end{pmatrix}}{\det\begin{pmatrix} a & b & c \\ b1 & -a1 & 0 \\ c1 & 0 & -a1 \end{pmatrix}}$$

$$zzk = \frac{\det\begin{pmatrix} a & b & d \\ b1 & -a1 & d1 \\ c1 & 0 & e1 \end{pmatrix}}{\det\begin{pmatrix} a & b & c \\ b1 & -a1 & 0 \\ c1 & 0 & -a1 \end{pmatrix}}$$

Al punto di coordinate (xk, yk, zk) nel piano immagine corrisponde il punto del profilo dell'oggetto di coordinate (xxk, yyk, zzk). Queste operazioni vanno ripetute per ogni punto del profilo in ogni immagine acquisita.

A seconda della particolare geometria dell'oggetto scannerizzato e della posizione del laser possono essere presenti discontinuità nel profilo ottenuto dall'acquisizione delle immagini [Figura 21]. Questo fatto causerebbe la presenza di buchi nella point cloud, e quindi di imprecisioni nella superficie dopo la ricostruzione in MeshLab. Per evitare che ciò accada lo script controlla la distanza di 2 punti consecutivi del profilo e, nel caso in cui sia maggiore rispetto ad un valore desiderato, li congiunge con un vettore e aggiunge i punti mancanti, così da ottenere un profilo senza interruzioni [Figura 22]. Al fine di ottenere una superficie chiusa il punto iniziale del profilo di ogni immagine deve

essere lo stesso, deve quindi giacere nell'asse di rotazione. Lo stesso ragionamento vale anche per il punto finale del profilo.

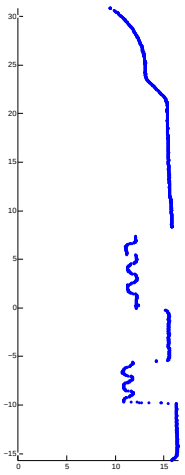

Figura 21: Profilo originale.

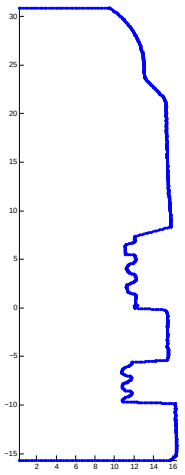

Figura 22: Profilo completo.

Lavorando in un'unica immagine, i punti del profilo giacciono tutti sullo stesso piano (il piano del laser). Ciò significa che dei 3 gradi di libertà, uno è inibito. Per ogni punto è quindi possibile scrivere una delle 3 coordinate in funzione delle altre 2 ($x = f(y, z)$ oppure $y = f(x, z)$ oppure $z = f(x, y)$).

Per semplificare i conti è quindi opportuno non utilizzare tutte e 3 le coordinate, ma solamente 2, senza perdita di alcuna informazione. È possibile effettuare una trasformazione delle coordinate dei punti dal piano Π del laser definito in precedenza ad un piano $\Pi' : y = 0$.

Il punto che nel piano Π aveva coordinate (xxk, yyk, zzk), nel piano Π' avrà coordinate:

$$\begin{cases} X' = \sqrt{xxk^2 + yyk^2} \\ Y' = 0 \\ Z' = zzk \end{cases}$$

È ora possibile determinare la distanza tra due punti consecutivi semplicemente calcolando:

$$dist = \sqrt{(X'(row) - X'(row-1))^2 + (Z'(row) - Z'(row-1))^2}$$

e, nel caso in cui sia maggiore della distanza massima accettabile, aggiungere i punti mancanti al profilo.

Una volta calcolate le coordinate dei punti da aggiungere per ottenere un profilo senza interruzioni, è necessario riportarle dal piano Π' al piano reale, tenendo conto anche della rotazione imposta dal motore a passo. Siccome dopo ogni foto il motore ruota l'oggetto di un passo (1.8°), ad ogni immagine è associato un angolo Θ, che viene incrementato di 1.8° ad ogni immagine successiva.

Questa operazione viene effettuata moltiplicando le coordinate appartenenti al piano Π' per una matrice di rotazione il cui argomento è Θ.

La matrice di rotazione rispetto all'asse z è definita come:

$$R = \begin{pmatrix} \cos(\Theta) & \sin(\Theta) & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

I valori degli elementi della matrice sono diversi per ogni immagine.

Le coordinate reali del punto in esame sono quindi:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \cdot \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}$$

Una volta effettuati i calcoli viene creata la variabile p3 al cui interno sono contenute le coordinate di tutti i punti della superficie dell'oggetto.

Al fine di migliorare l'accuratezza della superficie finale è possibile utilizzare più immagini, diminuendo quindi l'angolo di passo (per esempio 0.9°, corrispondente a 400 foto per ottenere il profilo completo). Questo causa però un aumento dei tempi di acquisizione, che dipendono dalla velocità con cui la macchina fotografica è in grado di recepire il comando di scatto e salvare la foto. Nel caso in esame è stato utilizzato un tempo di 1.5 secondi per ogni foto, al fine di non perdere nessuna immagine durante la

rotazione dell'oggetto, fatto che comporterebbe la ricostruzione di una superficie leggermente errata. Con altre fotocamere potrebbero essere sufficienti tempi minori.

5.4 DEFINIZIONE DELLE NORMALI

La normale ad una superficie in un punto P è definita, nello spazio tridimensionale, come il vettore perpendicolare al piano tangente nel punto.

Per poter ricostruire la superficie 3D di un oggetto non è sufficiente la sola conoscenza delle coordinate dei punti, ma è necessario sapere anche direzione e verso della normale di ogni punto. Questa informazione è necessaria allo script di ricostruzione per sapere quali sono l'interno e l'esterno della superficie da ricostruire.

Per il calcolo della normale di un generico punto P appartenente al profilo dell'oggetto in esame si fa uso di 4 vettori. Siano A e C i punti immediatamente precedente e successivo a P nel profilo dell'immagine n. Siano B il punto del profilo dell'immagine n-1 più vicino a P e D il punto del profilo dell'immagine n+1 più vicino a P [Figure 23, 24]. Si possono ora definire i 4 vettori:
$vett\_1 = \overrightarrow{PA}, vett\_2 = \overrightarrow{PB}, vett\_3 = \overrightarrow{PC}, vett\_4 = \overrightarrow{PD}$.

Effettuando il prodotto vettoriale tra 2 vettori adiacenti, procedendo in verso orario, si ottengono 4 normali in generale diverse tra loro:

$$norm\_12 = vett\_1 \times vett\_2$$

$$norm\_23 = vett\_2 \times vett\_3$$

$$norm\_34 = vett\_3 \times vett\_4$$

$$norm\_41 = vett\_4 \times vett\_1$$

La media di esse approssima il vettore normale cercato relativo al punto P:

$$norm\_p' = \frac{norm\_12+norm\_23+norm\_34+norm\_41}{4}$$

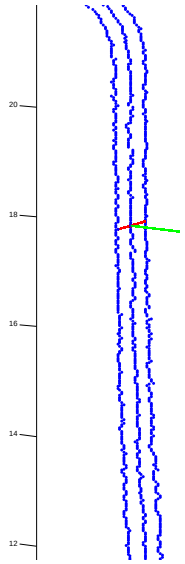

Figura 23: Vettori (in rosso) per il calcolo della normale (in verde).

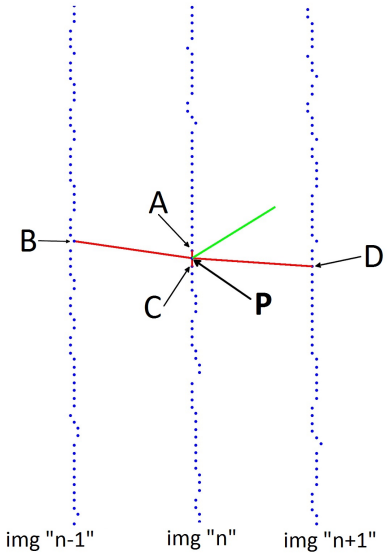

Figura 24: Particolare.

Il vettore ottenuto non ha modulo unitario, bisogna quindi normalizzarlo:

$$\mathrm{norm\_p} = \frac{\mathrm{norm\_p'}}{||\mathrm{norm\_p'}||}$$

In questo modo il modulo di tutte le normali è unitario ed il verso è orientato verso l'esterno.

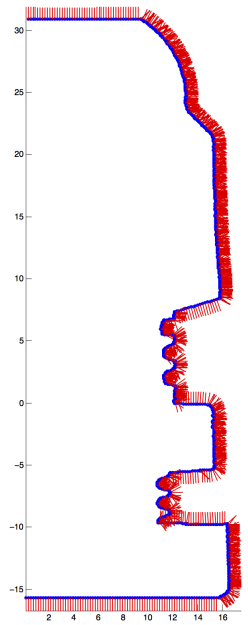

Figura 25: Profilo (in blu) ottenuto da un'immagine e relative normali (in rosso).

Per il primo e l'ultimo punto di ogni immagine, non essendo possibile applicare il procedimento appena descritto (mancano il punto immediatamente precedente nel primo caso e quello immediatamente successivo nel secondo), alle normali vengono assegnati rispettivamente i valori di quella relativa al secondo ed al penultimo punto del profilo.

Le coordinate dei punti e le normali vengono infine salvate in un file .txt secondo la sintassi: "*coord_x coord_y coord_z normal_x normal_y normal_z*".

Ogni riga del file contiene le informazioni necessarie alla visualizzazione di un singolo punto del profilo. Il file ottenuto può essere importato in appositi programmi di rendering 3D (MeshLab nello specifico) per visualizzare la nuvola di punti ed effettuare le operazioni necessarie per la ricostruzione della superficie.

Il codice completo per la calibrazione del fuoco ed il calcolo delle coordinate 3D è riportato nell'Appendice A.

5.5 ELABORAZIONE DEI DATI IN MESHLAB

Per visualizzare la nuvola di punti e ricostruire la superficie 3D dell'oggetto in esame si fa uso di MeshLab.

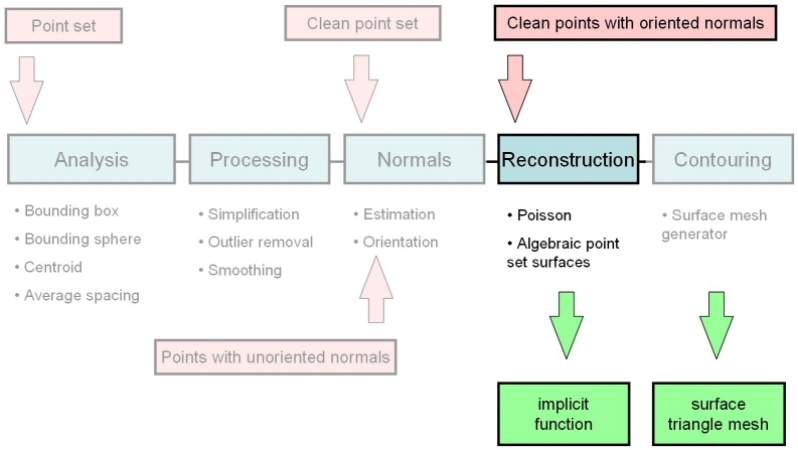

Figura 26: Processo di ricostruzione di una superficie 3D a partire dalla point cloud.

Il processo di ricostruzione di una superficie a partire da una nuvola di punti segue generalmente alcuni passi:

1. Scan dell'oggetto in modo da ottenere la point cloud ed eventualmente le normali
2. Rimozione dei punti indesiderati
3. Semplificazione, in modo da ridurre i tempi di calcolo
4. Smooth, al fine di ridurre l'eventuale rumore nei dati
5. Stima ed orientazione delle normali, nel caso in cui non siano fornite dal dispositivo di acquisizione
6. Ricostruzione della superficie.

Non tutti i passaggi appena descritti sono indispensabili per l'ottenimento della superficie 3D di un oggetto. Nel caso in esame infatti non è stato necessario né rimuovere punti indesiderati, né effettuare semplificazioni, smooth e stime delle normali, in quanto tutte le informazioni necessarie sono state calcolate precedentemente in MATLAB.

Il procedimento per la ricostruzione di una superficie utilizzando MeshLab è descritto di seguito.

All'apertura di MeshLab bisogna selezionare il comando *Import Mesh* e importare quindi il file .txt generato dallo script di MATLAB. Sono ora richiesti 4 parametri, che vanno impostati nel seguente modo:

- *Header row to be skipped:* 0
- *separator between:* SPACE
- *separator between:* X Y Z Nx Ny Nz
- *separator between:* $[0 - 255]$

In questo modo il file verrà letto senza errori.

A questo punto è visibile la nuvola di punti relativa all'oggetto in esame. È possibile visualizzare le normali con il comando *Render* $\rightarrow$ *Show Normal/Curvature*.

Per il calcolo della superficie si utilizza il comando *Filters* $\rightarrow$ *Point Set* $\rightarrow$ *Surface Reconstruction: Poisson*. L'accuratezza della ricostruzione dipende da 4 parametri: *Octree Depth, Solver Divide, Samples per Node, Surface Offsetting*.

- *Octree Depth* determina la precisione con cui verrà effettuata la ricostruzione della superficie, più alto è il valore e più la superficie sarà dettagliata. L'aumento di questo parametro comporta un incremento dei tempi di calcolo.
- *Solver Divide* permette di ottenere una diminuzione delle risorse necessarie per il calcolo della superficie, a costo però di un aumento dei tempi. Per valori di *Octree Depth* maggiori di 8, un valore di *Solver Divide* pari a 7 o 8 può diminuire l'uso di memoria durante il calcolo della superifice.
- *Samples per Node* è un parametro il cui valore ottimale dipende dalla quantità di rumore presente nella point cloud. Un valore più alto di questo parametro permette di ottenere superfici più smooth, e quindi più insensibili al rumore. Nel caso di point cloud con poco rumore, un valore di *Samples per Node* minore permette di ottenere una superficie più dettagliata.
- *Surface offsetting* è un indice di correzione il cui valore può variare tra 0.5 e 2. Un valore pari ad 1 indica assenza di offset.

I valori da assegnare variano da caso a caso e per determinare quelli che producono i risultati migliori è necessario effettuare delle prove. In generale però un maggior valore del parametro *Octree Depth* determina una superficie più dettagliata. Nel caso in esame i valori utilizzati sono: *OD*=12, *SD*=6, *SpN*=1, *SO*=1.

Dopo aver ricostruito la superficie è possibile semplificarla. Il procedimento prende il nome di decimazione e serve a ridurre il numero di triangoli di cui è formata la superficie. Ciò può risultare utile per esempio nel caso in cui si utilizzi la superficie per effettuare una stampa 3D, processo che richiede tempi molto lunghi nel caso di superfici complesse. Per la semplificazione si può utilizzare il comando *Filters* → *Remeshing, Simplification and Reconstruction* → *Quadric Edge Collapse Decimation*. È possibile definire il numero di triangoli di cui dovrà essere costituita la superficie dopo la decimazione o la riduzione percentuale. Un maggior numero di triangoli determina una superficie più dettagliata, mentre un numero minore di triangoli determina una superficie più semplice da analizzare. Il giusto compromesso dipende ovviamente dagli scopi e dalle possibilità computazionali.

Una volta ricostruita la superficie MeshLab permette di esportarla all'interno di un file .stl. In questo modo può essere utilizzata per effettuare simulazioni, misurazioni o dimensionamenti di protesi nel caso in cui l'oggetto scannerizzato sia ad esempio una mano o un dito. C'è inoltre la possibilità di stampre la superficie ottenuta utilizzando una stampante 3D.

6

RISULTATI E CONCLUSIONI

Durante i test è stato utilizzato un oggetto di piccole dimensioni (di altezza pari a 46 mm e diametro di base pari a 32 mm misurando da lato a lato e 35 mm misurando da angolo ad angolo) la cui superficie presenta marcate irregolarità [Figure 29, 30].

Figura 27: Oggetto di test, visione frontale.

Figura 28: Oggetto di test, visione dall'alto.

Verranno di seguito esposti i risultati ottenuti con lo script di MATLAB e la superficie ricostruita in MeshLab.

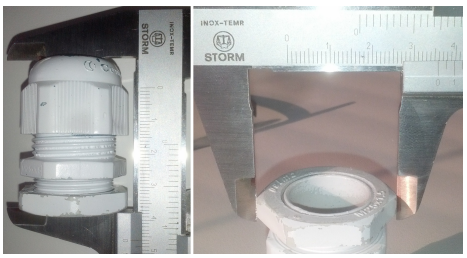

Figura 29: Dimensioni dell'oggetto di test.

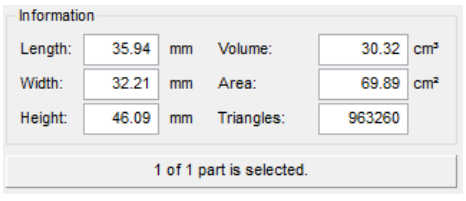

Figura 30: Dimensioni del modello 3D.

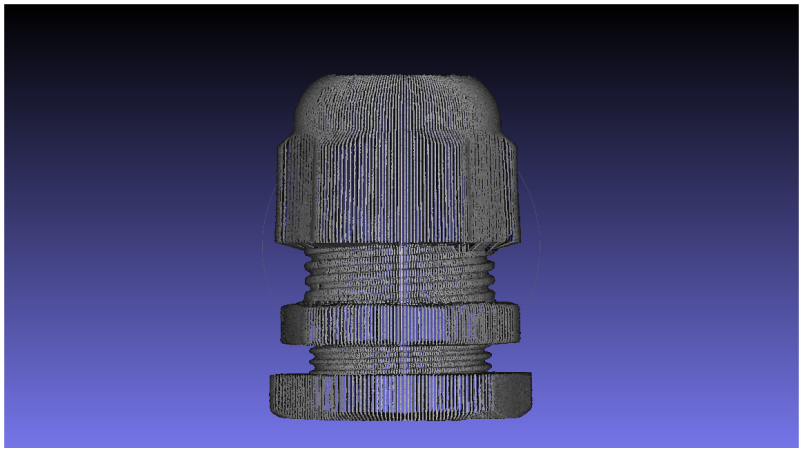

Figura 31: Point cloud in MeshLab.

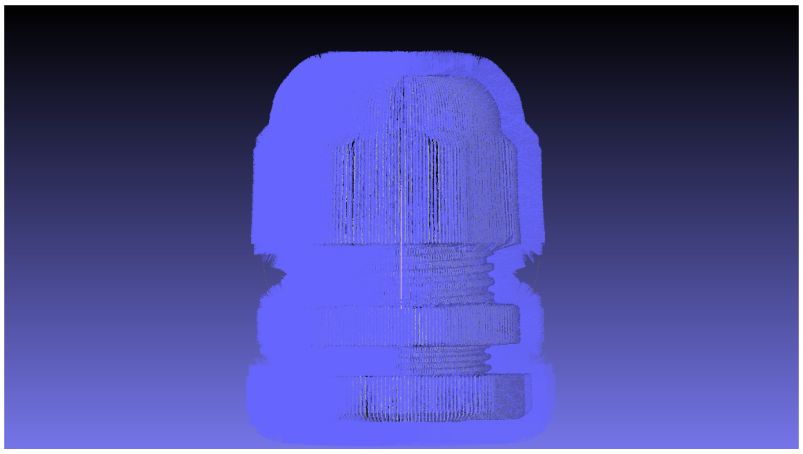

Figura 32: Point cloud e normali in MeshLab.

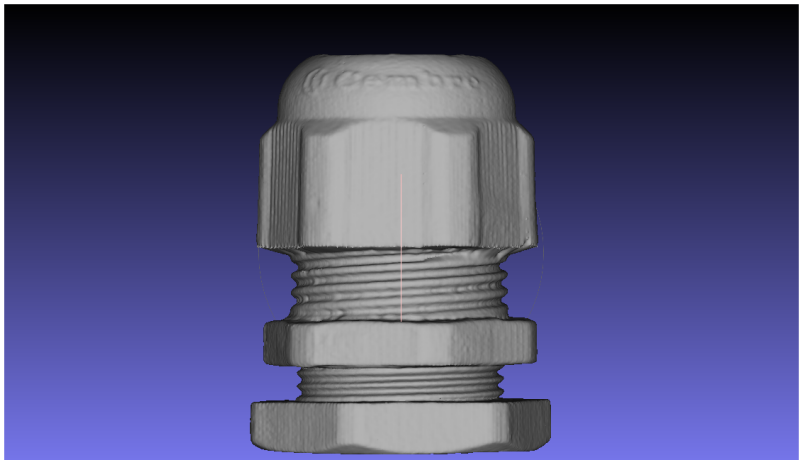

Figura 33: Superficie ricostruita in MeshLab.

Forma e dimensioni della superficie ricostruita corrispondono a quelle dell'oggetto di test, è inoltre possibile leggere nel modello 3D la scritta *Cembre* (di altezza pari a circa 3 mm) presente nella parte superiore dell'oggetto scannerizzato [Figura 34].

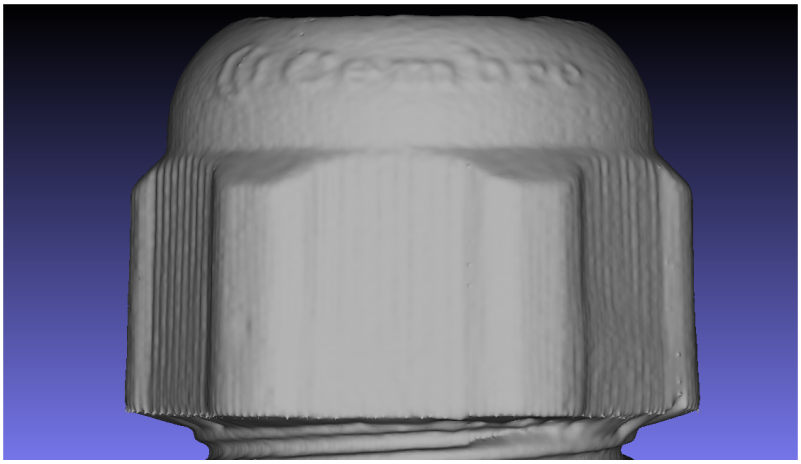

Figura 34: Particolare della superficie in MeshLab.

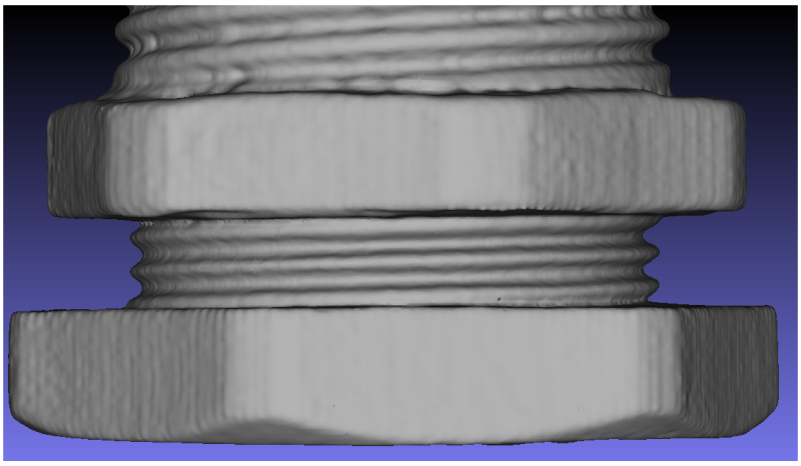

Figura 35: Particolare della superficie in MeshLab.

Tuttavia sono presenti alcune righe verticali in zone in cui l'oggetto è piatto. La causa di questa incongruenza è dovuta alla distanza tra le immagini (la quale dipende dall'angolo di passo del motore). Con un passo $\alpha_P = 1.8°$, approssimando l'oggetto analizzato ad un cilindro di altezza h pari a 46 mm e di diametro d pari a 35 mm è possibile calcolare una stima della distanza dei profili ottenuti in 2 immagini consecutive. Essa è pari a [Figura 36]:

$$dist = 2 \cdot \frac{d}{2} \cdot \sin(\frac{\alpha_P}{2})$$

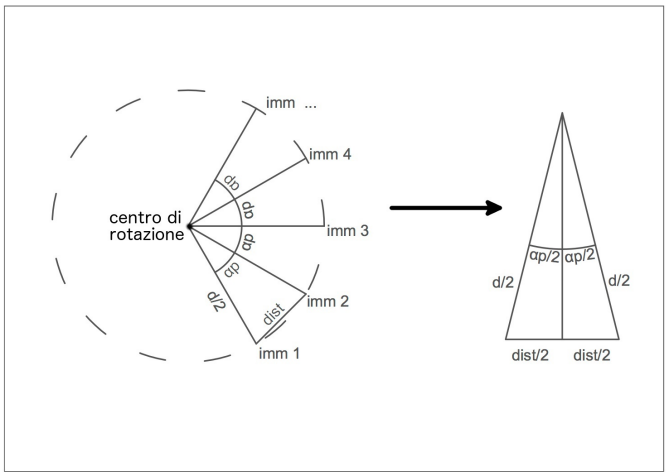

Figura 36: Schema per il calcolo della distanza tra i profili di 2 immagini consecutive.

Sostituendo i valori dell'oggetto di test si ottiene una distanza dei punti dei profili di 2 immagini pari a circa 0.55 mm. La distanza tra 2 punti consecutivi del profilo di una singola immagine si può invece calcolare approssimativamente con la formula:

$$dist = h/n\_pixel\_verticali\_area\_occupata\_dall\_oggetto$$

e, nel caso in esame, è pari a $46/1600 \approx 0.03$ mm. La distanza tra 2 punti del profilo adiacenti orizzontalmente è circa 18 volte maggiore di quella tra 2 punti adiacenti verticalmente, come è possibile notare nella Figura 24 Capitolo 5.

Comandando il motore in *half stepping* [Figura 38], così da avere un angolo di passo pari a 0.9° a cui corrisponde un totale di 400 immagini per ricostruire il profilo completo dell'oggetto, si può dimezzare la distanza media tra le immagini, ottenendo quindi una precisione migliore, a costo però di un raddoppio dei tempi di acquisizione.

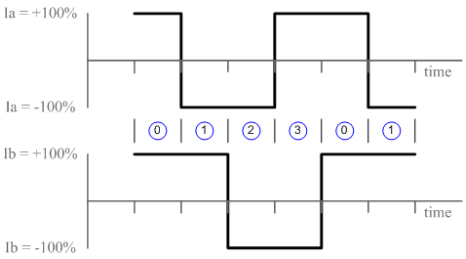

Figura 37: Full step drive.

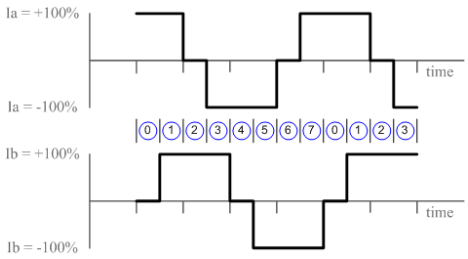

Figura 38: Half step drive.

BIBLIOGRAFIA

[1] Pierre Alliez, Laurent Saboret, Gaël Guennebaud, *Surface Reconstruction from Point Sets*, http://doc.cgal.org/latest/Surface_reconstruction_points_3/, 26/08/14

[2] Ing. Piergiorgio Aprili, *Sicurezza laser*, http://www.ac.infn.it/sicurezza/rel/LASER.pdf, 11/08/14

[3] Dimitri Batani, *Il rischio da laser: cosa è e come affrontarlo*, https://www.unimib.it/upload/Semin_sicur_laser.pdf, 20/08/14

[4] Daniele Bartolucci (2009), *Principi di laser scanning 3D*, pp. 11-15

[5] Fausto Bernardini, Holly E. Rushmeier (2002), *The 3D Model Acquisition Pipeline*, http://www1.cs.columbia.edu/~allen/PHOTOPAPERS/pipeline.fausto.pdf, 25/07/14

[6] Nicola D'Apuzzo (2006), *Overview of 3D surface digitization technologies in Europe*, http://hometrica.ch/publ/2006_3dimg.pdf, 28/07/14

[7] Michael Kazhdan, Matthew Bolitho and Hugues Hoppe (2006), *Poisson Surface Reconstruction*, http://research.microsoft.com/en-us/um/people/hoppe/poissonrecon.pdf, 19/08/14

[8] Jean-Marc Peallat (Vi Technology), *Solder Print: 3D SPI and Process Control*, http://www.pcb007.com/pages/columns.cgi?clmid=%20&artid=60668&_pf_=1, 30/07/14

[9] 3DScanCo, http://www.3dscanco.com/about/3d-scanning/, 28/07/14

[10] SAL3D Shape Analysis Library, http://www.aqsense.com/docs/docu/MergerTool.html, 04/08/14

[11] Ark Motion Controls, http://www.arkmotion.com/, 13/08/14

[12] FabScan open source 3D scanner, http://blog.ponoko.com/2011/12/19/fabscan-open-source-3d-scanner/, 05/08/14

[13] Pentax K200D Specifications, http://www.dpreview.com/articles/2407499942/pentaxtk200d#specs, 13/08/14

[14] Focal Length, http://www.dpreview.com/glossary/optical/focal-length, 06/08/14

[15] Las Cumbres Observatory, Global Telescope Network, https://lcogt.net/spacebook/refracting-telescopes, 06/08/14

[16] MATLAB, http://www.mathworks.it/products/matlab/, 14/08/14

[17] Simulink, http://www.mathworks.it/products/simulink/, 14/08/14

[18] MeshLab, http://meshlab.sourceforge.net/, 14/08/14

[19] Ricoh imaging, Pentax K200D, http://www.ricoh-imaging.it/it/reflex-digitali/PENTAX-K200D.html, 12/08/14

[20] Sensoray | embedded electronics, http://www.sensoray.com/products/626.htm, 12/08/14

[21] Zaber Microstepping Tutorial, http://www.zaber.com/wiki/Tutorials/Microstepping, 28/08/14

APPENDICE A

Il codice di MATLAB per la calibrazione della posizione del fuoco e lo script dello scanner 3D sono riportati di seguito.

A.1 CALIBRAZ_FUOCO.M

```
%%%%%%
% calibraz_fuoco.m determina la posizione del fuoco rispetto al centro di
% rotazione dell'oggetto da analizzare. Dopo aver indicato su quale
% immagine effettuare la calibrazione e' necessario impostare il valore
% dell'altezza effettiva dell'oggetto misurata in mm (variabile 'altezza')
% ed un valore realistico di prova per la posizione del fuoco (variabile
% 'yfuoco'). I parametri first_row, num_row, first_column, num_column
% servono a limitare l'analisi dell'immagine all'area di interesse in cui
% giace il profilo dell'oggetto illuminato dal laser. L'output di questo
% script e' 'yfuoco_teor', il cui valore indica la posizione nella
% coordinata y del fuoco rispetto al centro di rotazione.
%%%%%%


clear all;
close all;
clc;


% Oggetto di test
n_img=8087; % Immagine da analizzare
first_row=240;
num_row=1805;
first_column=1550;
num_column=1950;
cartella='oggetto'; % Cartella in cui si trovano le immagini da analizzare
```

```
altezza=46; % Altezza reale dell'oggetto di test in mm

% Coordinate del fuoco [mm]
xfuoco=-3.5;
yfuoco=-291.5; % Posizione del fuoco di primo tentativo
zfuoco=0;

% Coordinate del laser [mm]
xlaser=-217;
ylaser=-217;
zlaser=0;

theta=pi/4; % Angolo formato dal laser rispetto al piano della fotocamera [rad]

passo=1.8; % Passo del motore [gradi]

sensorw=23.5; % Larghezza sensore [mm]
sensorwpx=3872; % Larghezza sensore [pixel]
sensorh=15.7; % Altezza sensore [mm]
sensorhpx=2592; % Altezza sensore [pixel]

tarax=sensorwpx/sensorw; % [pixel/mm]
taraz=sensorhpx/sensorh; % [pixel/mm]

% Distanza focale [mm]
focaldist=55;

% Coordinate del centro di rotazione [mm]
xcr=0;
ycr=0;
zcr=0;

% Punti utilizzati per definire l'equazione del piano del laser
x1=xcr-cos(theta)*450; %A
y1=ycr-sin(theta)*450;
z1=zcr;
x2=xcr+cos(theta)*100; %B
y2=ycr+sin(theta)*100;
```

```
z2=zcr+150;
x3=xcr+cos(theta)*100; %C
y3=ycr+sin(theta)*100;
z3=zcr-150;

% Piano del laser
a=(y2-y1)*(z3-z1)-(z2-z1)*(y3-y1);
b=-((x2-x1)*(z3-z1)-(z2-z1)*(x3-x1));
c=(x2-x1)*(y3-y1)-(y2-y1)*(x3-x1);
d=a*x1+b*y1+c*z1;



% Applico fullscanner.m alla prima immagine in modo da ottenere le
% coordinare del punto piu' alto e piu' basso dell'oggetto in modo da
% confrontare l'altezza ottenuta con l'altezza reale

% Leggo l'immagine
A=imread(fullfile(cartella ,sprintf('IMG_%d.JPG',n_img)));
B=rgb2gray(A);
C=im2bw(B, 0.5);

% Inizializzo la variabile edge che conterra' la posizione dei pixel bianchi
% (profilo del laser), dopo averne calcolato la media riga per riga
edge = zeros(1,num_row);

for row = first_row:num_row
    sum = 0;
    count = 0;

    for column = first_column:num_column
        sum = sum + column*C(row,column);
        if C(row,column)
            count = count + 1;
        end

        if count == 0
            edge(row) = 0;
        else
```

```
            edge(row) = round(sum/count); % Nella riga 'row', il pixel del profilo
                dell'oggetto si trova nella posizione 'edge(row)'
        end
    end
end

% Inizializzo variabili xk,yk,zk, che conterranno le coordinate dei punti del
    profilo nel piano immagine
xk = zeros(1,length(edge));
yk = zeros(1,length(edge));
zk = zeros(1,length(edge));

% Inizializzo variabili xxk,yyk,zzk, che conterranno le coordinate dei punti
% del profilo reali
xxk = zeros(1,length(edge));
yyk = zeros(1,length(edge));
zzk = zeros(1,length(edge));

% Distanza in pixel dei punti nel piano immagine dal centro del piano immagine
deltax = zeros(1,length(row));
deltaz = zeros(1,length(row));


for row = 1:length(edge)
    if edge(row) ~= 0 && row ~= sensorhpx/2
        deltax(row)=-(sensorwpx/2-edge(row));
        deltaz(row)=-(row-sensorhpx/2);

        % Posizione nel piano immagine dei pixel del profilo
        xk(row)=xfuoco-deltax(row)/tarax;
        yk(row)=yfuoco-focaldist;
        zk(row)=zfuoco-deltaz(row)/taraz;

        a1=xfuoco-xk(row);
        b1=yfuoco-yk(row);
        c1=zfuoco-zk(row);
        d1=xk(row)*yfuoco-yk(row)*xfuoco;
        e1=xk(row)*zfuoco-zk(row)*xfuoco;
        matr= [a   b   c
```

```
                b1 -a1  0
                c1  0  -a1];
        matrx=[d   b   c
                d1 -a1  0
                e1  0  -a1];
        matry=[a   d   c
                b1  d1  0
                c1  e1 -a1];
        matrz=[a   b   d
                b1 -a1  d1
                c1  0   e1];
        % Interseco piano del laser e rette che collegano i punti nel
        % piano immagine al fuoco (Cramer)
        xxk(row)=det(matrx)/det(matr);
        yyk(row)=det(matry)/det(matr);
        zzk(row)=det(matrz)/det(matr);
    elseif xxk(row)==0 && yyk(row)==0 && zzk(row)==0 && row>1
        xxk(row)=xxk(row-1);
        yyk(row)=yyk(row-1);
        zzk(row)=zzk(row-1);
        % (xxk,yyk,zzk) sono le coordinate reali dell'oggetto
    end
end

% Determino da dove iniziano le coordinate dei punti
ii=1;
for jj=1:length(edge)-1
    if xxk(jj)==0 && yyk(jj)==0 && zzk(jj)==0
    ii=ii+1;
    end
end


% Primo punto del profilo
alfa1=xfuoco-xk(ii);
beta1=yfuoco-yk(ii);
gamma1=zfuoco-zk(ii);
delta1=xk(ii)*yfuoco-yk(ii)*xfuoco;
ni1=xk(ii)*zfuoco-zk(ii)*xfuoco;
```

```
% Ultimo punto del profilo
alfa2=xfuoco-xk(num_row);
beta2=yfuoco-yk(num_row);
gamma2=zfuoco-zk(num_row);
delta2=xk(num_row)*yfuoco-yk(num_row)*xfuoco;
ni2=xk(num_row)*zfuoco-zk(num_row)*xfuoco;

% Matrici per la soluzione del sistema lineare utilizzato per determinare
% le coordinare effettive del profilo dell'oggetto
matr1=[a       b      c;
       beta1  -alfa1  0;
       gamma1  0     -alfa1];

matr2=[a       b      c;
       beta2  -alfa2  0;
       gamma2  0     -alfa2];

matrz1=[a       b      d;
        beta1  -alfa1  delta1;
        gamma1  0      ni1];

matrz2=[a       b      d;
        beta2  -alfa2  delta2;
        gamma2  0      ni2];


% Risolvo il sistma lineare imponendo l'altezza reale in modo da ottenere come
    risultato la posizione del fuoco
syms yfuoco
[yfteor]=solve(-altezza==((-a*alfa2*ni2+b*(xk(num_row)*yfuoco-yk(num_row)*xfuoco)*
    gamma2+d*(yfuoco-yk(num_row))*0+d*alfa2*gamma2-b*(xk(num_row)*yfuoco-yk(num_row
    )*xfuoco)*0-b*(yfuoco-yk(num_row))*ni2)/det(matr2))-((-a*alfa1*ni1+b*(xk(ii)*
    yfuoco-yk(ii)*xfuoco)*gamma1+d*(yfuoco-yk(ii))*0+d*alfa1*gamma1-b*(xk(ii)*
    yfuoco-yk(ii)*xfuoco)*0-b*(yfuoco-yk(ii))*ni1)/det(matr1)));
yfuoco_teor=double(yfteor)
```

A.2 FULLSCANNER.M

```
%%%%%%
% fullscanner.m determina le coordinate dei punti e delle normali
% dell'oggetto scannerizzato. I parametri richiesti sono il numero di
% immagini totale (num_img), la prima immagine (start), la posizione del
% fuoco e del laser rispetto al centro di rotazione dell'oggetto, e l'area
% di interesse dove giace il profilo illuminato dal laser in ogni immagine
% (da impostare coi parametri first_ row, num_row, first_column,
% num_column, i quali indicano i valori del primo e dell'ultimo pixel in
% verticale e in orizzontale dell'area di interesse). Dopo aver fatto
% partire lo script verra' richiesto il nome del file .txt all'interno del
% quale verranno salvate le coordinate 3D dei punti della superficie
% dell'oggetto. Il file .txt verra' creato all'interno della cartella
% MATLAB.
%%%%%%


clear all;
close all;
clc;

filename = input('Inserire nome file txt: ','s'); % File in cui verranno salvate le
    coordinate dei punti e delle normali
delete(filename)

% Oggetto di test
num_img=200; % Numero di immagini acquisite
start=8087; % Prima immagine
stop=start+num_img-1; % Ultima immagine
first_row=240;
num_row=1805;
first_column=1550;
num_column=1950;
cartella='oggetto'; % Cartella che contiene le immagini (percorso: .../MATLAB/
    cartella)
```

```
% Coordinate del fuoco della fotocamera [mm]
xfuoco=-3.5;
yfuoco=-287.5; % 'yfuoco' determinata con la calibrazione
zfuoco=0;

% Coordinate del laser [mm]
xlaser=-217;
ylaser=-217;
zlaser=0;

theta=pi/4; % Angolo formato dal laser rispetto al piano [rad]

passo=1.8; % Passo del motore [gradi]

sensorw=23.5; % Larghezza sensore [mm]
sensorwpx=3872; % Larghezza sensore [pixel]
sensorh=15.7; % Altezza sensore [mm]
sensorhpx=2592; % Altezza sensore [pixel]

tarax=sensorwpx/sensorw; % [pixel/mm]
taraz=sensorhpx/sensorh; % [pixel/mm]

focaldist=55; % Distanza focale [mm]

% Coordinate del centro di rotazione [mm]
xcr=0;
ycr=0;
zcr=0;

% Punti utilizzati per definire l'equazione del piano del laser
x1=xcr-cos(theta)*450; %A
y1=ycr-sin(theta)*450;
z1=zcr;
x2=xcr+cos(theta)*100; %B
y2=ycr+sin(theta)*100;
z2=zcr+150;
x3=xcr+cos(theta)*100; %C
y3=ycr+sin(theta)*100;
z3=zcr-150;
```

```
% Piano del laser
a=(y2-y1)*(z3-z1)-(z2-z1)*(y3-y1);
b=-((x2-x1)*(z3-z1)-(z2-z1)*(x3-x1));
c=(x2-x1)*(y3-y1)-(y2-y1)*(x3-x1);
d=a*x1+b*y1+c*z1;



for zz=start:stop

    n_img=zz-start+1;

    % Leggo l'immagine
    A=imread(fullfile(cartella ,sprintf('IMG_%d.JPG',zz)));
    % Converto l'immagine in scala di grigi
    B=rgb2gray(A);
    % Converto l'immagine ins cala di grigi in immagine binaria (1=bianco, 0=nero)
    C=im2bw(B, 0.5);

    % Inizializzo la variabile edge che conterra' la posizione dei pixel bianchi (
        profilo del laser), dopo averne calcolato la media riga per riga
    edge = zeros(1,num_row);

    for row = first_row:num_row
        sum = 0;
        count = 0;

        for column = first_column:num_column
            sum = sum + column*C(row,column);
            if C(row,column)
                count = count + 1;
            end

            if count == 0
                edge(row) = 0;
            else
                edge(row) = round(sum/count); % Nella riga 'row', il pixel del
                    profilo dell'oggetto si trova nella posizione 'edge(row)'
```

```
        end
    end
end

% Inizializzo variabili xk,yk,zk, che conterranno le coordinate dei punti del
    profilo nel piano immagine
xk = zeros(1,length(edge));
yk = zeros(1,length(edge));
zk = zeros(1,length(edge));

% Inizializzo variabili xxk,yyk,zzk, che conterranno le coordinate dei punti
    del profilo reali
xxk = zeros(1,length(edge));
yyk = zeros(1,length(edge));
zzk = zeros(1,length(edge));

% Distanza in pixel dei punti nel piano immagine dal centro del piano immagine
deltax = zeros(1,length(row));
deltaz = zeros(1,length(row));


for row = 1:length(edge)
    if edge(row) ~= 0 && row ~= sensorhpx/2
        deltax(row)=-(sensorwpx/2-edge(row));
        deltaz(row)=-(row-sensorhpx/2);

        % Posizione nel piano immagine dei pixel del profilo
        xk(row)=xfuoco-deltax(row)/tarax;
        yk(row)=yfuoco-focaldist;
        zk(row)=zfuoco-deltaz(row)/taraz;

        a1=xfuoco-xk(row);
        b1=yfuoco-yk(row);
        c1=zfuoco-zk(row);
        d1=xk(row)*yfuoco-yk(row)*xfuoco;
        e1=xk(row)*zfuoco-zk(row)*xfuoco;
        matr= [a   b   c
               b1 -a1  0
               c1  0  -a1];
```

```
        matrx=[d   b   c
               d1 -a1  0
               e1  0  -a1];
        matry=[a   d   c
               b1  d1  0
               c1  e1 -a1];
        matrz=[a   b   d
               b1 -a1  d1
               c1  0   e1];
        % Interseco piano del laser e rette che collegano i punti nel piano
            immagine al fuoco (Cramer)
        xxk(row)=det(matrx)/det(matr);
        yyk(row)=det(matry)/det(matr);
        zzk(row)=det(matrz)/det(matr);
    elseif xxk(row)==0 && yyk(row)==0 && zzk(row)==0 && row>1
        xxk(row)=xxk(row-1);
        yyk(row)=yyk(row-1);
        zzk(row)=zzk(row-1);
        % (xxk,yyk,zzk) sono le coordinate reali dell'oggetto
    end
end


% Determino da dove iniziano le coordinate dei punti
ii=1;
for jj=1:length(edge)-1
    if xxk(jj)==0 && yyk(jj)==0 && zzk(jj)==0
    ii=ii+1;
    end
end



% Riempio i buchi
p1 = zeros(num_row-ii+3,3);

k = [double(xxk); double(yyk); double(zzk)]';
```

```
p1(1,:) = [0, 0, k(ii,3)]; % Aggiungo come primo punto un punto che giace nell'
    asse di rotazione

for jj=ii:num_row
    p1(jj-ii+2,:)=[sqrt((k(jj,1)-xcr)^2 + (k(jj,2)-ycr)^2),0,k(jj,3)];
end

p1(end,:)=[0, 0, k(num_row,3)]; % Aggiungo come ultimo punto un punto che giace
     nell'asse di rotazione

precision=0.2; % Un valore piu' piccolo di questo parametro causa un maggiore
    numero dei punti che verranno aggiunti al profilo al posto dei buchi

p2(1,:) = p1(1,:);

% Se n_passi>1, cioe' la distanza tra 2 punti consecutivi e' maggiore
% della massima desiderata, aggiungo i punti mancanti
for jj=2:length(p1)
    n_passi = ceil(sqrt((p1(jj,1)-p1(jj-1,1))^2+(p1(jj,3)-p1(jj-1,3))^2)/
        precision);
    if n_passi > 1
        p_temp = [0, 0, 0];
        for kk=1:n_passi-1
            p_temp(kk,:) = [(p1(jj,1)-p1(jj-1,1))*(kk)/n_passi+p1(jj-1,1), 0, (
                p1(jj,3)-p1(jj-1,3))*(kk)/n_passi+p1(jj-1,3)];
        end
        p2 = [p2; p_temp];
    end
    p2 = [p2; p1(jj,:)]; % p2 contiene le coordinate del profilo dell'oggetto
        dopo aver chiuso eventuali buchi (profilo senza interruzioni)
end

angle = 360-(zz-start)*passo;
ang = angle*pi/180;
% Matrice di rotazione
R =[cos(ang)  sin(ang)  0;
    -sin(ang) cos(ang)  0;
    0         0         1];
```

```
    posiz=[R*p2']'; % Moltiplicando i risultati ottenuti per la matrice di
        rotazione R tengo conto della rotazione dell'oggetto imposta dal motore a
        passo


    for hh=1:length(posiz)
        p3(n_img,hh,:) = posiz(hh,:); % Salvo tutte le informazioni nella variabile
             p3 secondo la sintassi [n_img n_riga xyz]
    end

    clear p2
    clear norm

    calcolo_coordinate_punti=num_img-(zz-start+1)
end




%%%% Normali prima immagine
% Inizializzo i vettori che mi serviranno per il calcolo delle normali,
% cosi' da velocizzare i successivi calcoli
vett_prec=zeros(length(p3),3);
vett_succ=zeros(length(p3),3);

vett1=zeros(num_img,length(p3),3);
vett2=zeros(num_img,length(p3),3);
vett3=zeros(num_img,length(p3),3);
vett4=zeros(num_img,length(p3),3);

i_min_prec=zeros(1,length(p3));
i_min_succ=zeros(1,length(p3));

prod_vet_mat_12=zeros(num_img,length(p3),3);
prod_vet_mat_23=zeros(num_img,length(p3),3);
prod_vet_mat_34=zeros(num_img,length(p3),3);
prod_vet_mat_41=zeros(num_img,length(p3),3);

norm12=zeros(length(p3),3);
```

```
norm23=zeros(length(p3),3);
norm34=zeros(length(p3),3);
norm41=zeros(length(p3),3);
norm_mean1=zeros(length(p3),3);
norm_mean=zeros(length(p3),3);

vett_prec(:,:)=p3(num_img,:,:);
vett_succ(:,:)=p3(2,:,:);


for nn = 1:length(p3)
    % Per ogni punto del profilo nell'immagine 1 determino la posizione dei
    % punti piu' vicini nelle immagini precedente e successiva
    l = length(p3);
    aa = cat(2,ones(l,1)*p3(1,nn,1),ones(l,1)*p3(1,nn,2),ones(l,1)*p3(1,nn,3));
    vett_diff_prec = vett_prec - aa;
    vett_diff_succ = vett_succ - aa;

    mod_prec = vett_diff_prec(:,1).*vett_diff_prec(:,1) + vett_diff_prec(:,2).*
        vett_diff_prec(:,2) + vett_diff_prec(:,3).*vett_diff_prec(:,3);
    mod_succ = vett_diff_succ(:,1).*vett_diff_succ(:,1) + vett_diff_succ(:,2).*
        vett_diff_succ(:,2) + vett_diff_succ(:,3).*vett_diff_succ(:,3);
    [val_min_prec, i_min_prec(nn)]=min(mod_prec); % val_min_prec indica la
        posizione del punto piu' vicino nell'immagine precedente
    [val_min_succ, i_min_succ(nn)]=min(mod_succ); % val_min_succ indica la
        posizione del punto piu' vicino nell'immagine successiva

    if nn>1 && nn<length(p3)

        % Vettori che collegano il punto di cui si vuole calcolare la
        % normale ai 4 punti ad esso adiacenti
        vett1(1,nn,:)=[(p3(num_img,i_min_prec(nn),1)-p3(1,nn,1)), (p3(num_img,
            i_min_prec(nn),2)-p3(1,nn,2)), (p3(num_img,i_min_prec(nn),3)-p3(1,nn,3)
            )];
        vett2(1,nn,:)=[p3(1,nn-1,1)-p3(1,nn,1), p3(1,nn-1,2)-p3(1,nn,2), p3(1,nn
            -1,3)-p3(1,nn,3)];
        vett3(1,nn,:)=[(p3(2,i_min_succ(nn),1)-p3(1,nn,1)), (p3(2,i_min_succ(nn),2)
            -p3(1,nn,2)), (p3(2,i_min_succ(nn),3)-p3(1,nn,3))];
```

```
vett4(1,nn,:)=[p3(1,nn+1,1)-p3(1,nn,1), p3(1,nn+1,2)-p3(1,nn,2), p3(1,nn
    +1,3)-p3(1,nn,3)];

% Prodotto vettoriale dei vettori appena calcolati, in modo da
% ottenere 4 normali
prod_vet_mat_12(1,nn,:)=[0 -vett1(1,nn,3) vett1(1,nn,2)
                         vett1(1,nn,3) 0 -vett1(1,nn,1)
                         -vett1(1,nn,2) vett1(1,nn,1) 0]*[vett2(1,nn,1)
                             vett2(1,nn,2) vett2(1,nn,3)]';
norm12(nn,:)=prod_vet_mat_12(1,nn,:)/sqrt((prod_vet_mat_12(1,nn,1))^2+(
    prod_vet_mat_12(1,nn,2))^2+(prod_vet_mat_12(1,nn,3))^2);

prod_vet_mat_23(1,nn,:)=[0 -vett2(1,nn,3) vett2(1,nn,2)
                         vett2(1,nn,3) 0 -vett2(1,nn,1)
                         -vett2(1,nn,2) vett2(1,nn,1) 0]*[vett3(1,nn,1)
                             vett3(1,nn,2) vett3(1,nn,3)]';
norm23(nn,:)=prod_vet_mat_23(1,nn,:)/sqrt((prod_vet_mat_23(1,nn,1))^2+(
    prod_vet_mat_23(1,nn,2))^2+(prod_vet_mat_23(1,nn,3))^2);

prod_vet_mat_34(1,nn,:)=[0 -vett3(1,nn,3) vett3(1,nn,2)
                         vett3(1,nn,3) 0 -vett3(1,nn,1)
                         -vett3(1,nn,2) vett3(1,nn,1) 0]*[vett4(1,nn,1)
                             vett4(1,nn,2) vett4(1,nn,3)]';
norm34(nn,:)=prod_vet_mat_34(1,nn,:)/sqrt((prod_vet_mat_34(1,nn,1))^2+(
    prod_vet_mat_34(1,nn,2))^2+(prod_vet_mat_34(1,nn,3))^2);

prod_vet_mat_41(1,nn,:)=[0 -vett4(1,nn,3) vett4(1,nn,2)
                         vett4(1,nn,3) 0 -vett4(1,nn,1)
                         -vett4(1,nn,2) vett4(1,nn,1) 0]*[vett1(1,nn,1)
                             vett1(1,nn,2) vett1(1,nn,3)]';
norm41(nn,:)=prod_vet_mat_41(1,nn,:)/sqrt((prod_vet_mat_41(1,nn,1))^2+(
    prod_vet_mat_41(1,nn,2))^2+(prod_vet_mat_41(1,nn,3))^2);

% Calcolo della media delle 4 normali calcolate
norm_mean1(nn,:)=(norm12(nn,:)+norm23(nn,:)+norm34(nn,:)+norm41(nn,:))/4;
% Normalizzo il vettore appena calcolato, norm_mean(nn,:) contiene
% le coordinate (x,y,z) del vettore normale relativo al punto di
% coordinate p3(n_img,nn,:)
```

```
        norm_mean(nn,:)=-norm_mean1(nn,:)/sqrt((norm_mean1(nn,1))^2+(norm_mean1(nn
            ,2))^2+(norm_mean1(nn,3))^2);

        % Nel caso in cui il calcolo della normale porti ad un valore indefinito (
            per esempio in presenza di vettori nulli) copio il valore della normale
             calcolata nel punto precedente
        if isnan(norm_mean(nn,1)) || isnan(norm_mean(nn,2)) || isnan(norm_mean(nn
            ,3))
            norm_mean(nn,:)=norm_mean(nn-1,:);
        end
        norm_mean(1,:)=norm_mean(2,:);
        norm_mean(end,:)=norm_mean(end-1,:);
    end

    n3(1,nn,:) = norm_mean(nn,:); % Salvo tutte le informazioni nella variabile n3
        secondo la sintassi [n_img n_riga nxnynz]
    if p3(1,nn,1)~=0 && p3(1,nn,2)~=0 && p3(1,nn,3)~=0
        slv=[p3(1,nn,1), p3(1,nn,2), p3(1,nn,3), norm_mean(nn,1), norm_mean(nn,2),
            norm_mean(nn,3)]; % La variabile slv contiene, per ogni riga: coord_x
            coord_y coord_z norm_x norm_y norm_z
        save(filename, 'slv','-append','-ASCII') % salvo slv all'interno del file .
            txt definito a inizio script
    end
end


%%%% Normali seconda - penultima immagine
% Stessi ragionamenti della prima immagine
for n_img=2:num_img-1
    calcolo_normali=num_img-(n_img+1)

    vett_prec(:,:)=p3(n_img-1,:,:);
    vett_succ(:,:)=p3(n_img+1,:,:);


    for nn = 1:length(p3)
        l = length(p3);
        aa = cat(2,ones(l,1)*p3(n_img,nn,1),ones(l,1)*p3(n_img,nn,2),ones(l,1)*p3(
            n_img,nn,3));
```

```
vett_diff_prec = vett_prec - aa;
vett_diff_succ = vett_succ - aa;

mod_prec = vett_diff_prec(:,1).*vett_diff_prec(:,1) + vett_diff_prec(:,2).*
    vett_diff_prec(:,2) + vett_diff_prec(:,3).*vett_diff_prec(:,3);
mod_succ = vett_diff_succ(:,1).*vett_diff_succ(:,1) + vett_diff_succ(:,2).*
    vett_diff_succ(:,2) + vett_diff_succ(:,3).*vett_diff_succ(:,3);
[val_min_prec, i_min_prec(nn)]=min(mod_prec);
[val_min_succ, i_min_succ(nn)]=min(mod_succ);

if nn>1 && nn<length(p3)

    vett1(n_img,nn,:)=[(p3(n_img-1,i_min_prec(nn),1)-p3(n_img,nn,1)), (p3(
        n_img-1,i_min_prec(nn),2)-p3(n_img,nn,2)), (p3(n_img-1,i_min_prec(
        nn),3)-p3(n_img,nn,3))];
    vett2(n_img,nn,:)=[p3(n_img,nn-1,1)-p3(n_img,nn,1), p3(n_img,nn-1,2)-p3
        (n_img,nn,2), p3(n_img,nn-1,3)-p3(n_img,nn,3)];
    vett3(n_img,nn,:)=[(p3(n_img+1,i_min_succ(nn),1)-p3(n_img,nn,1)), (p3(
        n_img+1,i_min_succ(nn),2)-p3(n_img,nn,2)), (p3(n_img+1,i_min_succ(
        nn),3)-p3(n_img,nn,3))];
    vett4(n_img,nn,:)=[p3(n_img,nn+1,1)-p3(n_img,nn,1), p3(n_img,nn+1,2)-p3
        (n_img,nn,2), p3(n_img,nn+1,3)-p3(n_img,nn,3)];


    prod_vet_mat_12(n_img,nn,:)=[0 -vett1(n_img,nn,3) vett1(n_img,nn,2)
                                 vett1(n_img,nn,3) 0 -vett1(n_img,nn,1)
                                 -vett1(n_img,nn,2) vett1(n_img,nn,1) 0]*[
                                     vett2(n_img,nn,1) vett2(n_img,nn,2)
                                     vett2(n_img,nn,3)]';
    norm12(nn,:)=prod_vet_mat_12(n_img,nn,:)/sqrt((prod_vet_mat_12(n_img,nn
        ,1))^2+(prod_vet_mat_12(n_img,nn,2))^2+(prod_vet_mat_12(n_img,nn,3)
        )^2);

    prod_vet_mat_23(n_img,nn,:)=[0 -vett2(n_img,nn,3) vett2(n_img,nn,2)
                                 vett2(n_img,nn,3) 0 -vett2(n_img,nn,1)
                                 -vett2(n_img,nn,2) vett2(n_img,nn,1) 0]*[
                                     vett3(n_img,nn,1) vett3(n_img,nn,2)
                                     vett3(n_img,nn,3)]';
```

```
        norm23(nn,:)=prod_vet_mat_23(n_img,nn,:)/sqrt((prod_vet_mat_23(n_img,nn
            ,1))^2+(prod_vet_mat_23(n_img,nn,2))^2+(prod_vet_mat_23(n_img,nn,3)
            )^2);

        prod_vet_mat_34(n_img,nn,:)=[0 -vett3(n_img,nn,3) vett3(n_img,nn,2)
                                     vett3(n_img,nn,3) 0 -vett3(n_img,nn,1)
                                     -vett3(n_img,nn,2) vett3(n_img,nn,1) 0]*[
                                         vett4(n_img,nn,1) vett4(n_img,nn,2)
                                         vett4(n_img,nn,3)]';
        norm34(nn,:)=prod_vet_mat_34(n_img,nn,:)/sqrt((prod_vet_mat_34(n_img,nn
            ,1))^2+(prod_vet_mat_34(n_img,nn,2))^2+(prod_vet_mat_34(n_img,nn,3)
            )^2);

        prod_vet_mat_41(n_img,nn,:)=[0 -vett4(n_img,nn,3) vett4(n_img,nn,2)
                                     vett4(n_img,nn,3) 0 -vett4(n_img,nn,1)
                                     -vett4(n_img,nn,2) vett4(n_img,nn,1) 0]*[
                                         vett1(n_img,nn,1) vett1(n_img,nn,2)
                                         vett1(n_img,nn,3)]';
        norm41(nn,:)=prod_vet_mat_41(n_img,nn,:)/sqrt((prod_vet_mat_41(n_img,nn
            ,1))^2+(prod_vet_mat_41(n_img,nn,2))^2+(prod_vet_mat_41(n_img,nn,3)
            )^2);

        norm_mean1(nn,:)=(norm12(nn,:)+norm23(nn,:)+norm34(nn,:)+norm41(nn,:))
            /4;
        norm_mean(nn,:)=-norm_mean1(nn,:)/sqrt((norm_mean1(nn,1))^2+(norm_mean1
            (nn,2))^2+(norm_mean1(nn,3))^2);


        if isnan(norm_mean(nn,1)) || isnan(norm_mean(nn,2)) || isnan(norm_mean(
            nn,3))
            norm_mean(nn,:)=norm_mean(nn-1,:);
        end
        norm_mean(1,:)=norm_mean(2,:);
        norm_mean(end,:)=norm_mean(end-1,:);
    end

        n3(n_img,nn,:) = norm_mean(nn,:); % [n_img n_riga nxnynz]
        if p3(n_img,nn,1)~=0 && p3(n_img,nn,2)~=0 && p3(n_img,nn,3)~=0
```

```
            slv=[p3(n_img,nn,1), p3(n_img,nn,2), p3(n_img,nn,3), norm_mean(nn,1),
                norm_mean(nn,2), norm_mean(nn,3)];
            save(filename, 'slv','-append','-ASCII')
        end
    end
end


%%%% Normali ultima immagine
% Stessi ragionamenti della prima immagine
vett_prec(:,:)=p3(num_img-1,:,:);
vett_succ(:,:)=p3(1,:,:);


for nn = 1:length(p3)
    l = length(p3);
    aa = cat(2,ones(l,1)*p3(num_img,nn,1),ones(l,1)*p3(num_img,nn,2),ones(l,1)*p3(
        num_img,nn,3));
    vett_diff_prec = vett_prec - aa;
    vett_diff_succ = vett_succ - aa;

    mod_prec = vett_diff_prec(:,1).*vett_diff_prec(:,1) + vett_diff_prec(:,2).*
        vett_diff_prec(:,2) + vett_diff_prec(:,3).*vett_diff_prec(:,3);
    mod_succ = vett_diff_succ(:,1).*vett_diff_succ(:,1) + vett_diff_succ(:,2).*
        vett_diff_succ(:,2) + vett_diff_succ(:,3).*vett_diff_succ(:,3);
    [val_min_prec, i_min_prec(nn)]=min(mod_prec);
    [val_min_succ, i_min_succ(nn)]=min(mod_succ);

    if nn>1 && nn<length(p3)

        vett1(num_img,nn,:)=[(p3(num_img-1,i_min_prec(nn),1)-p3(num_img,nn,1)), (p3
            (num_img-1,i_min_prec(nn),2)-p3(num_img,nn,2)), (p3(num_img-1,
            i_min_prec(nn),3)-p3(num_img,nn,3))];
        vett2(num_img,nn,:)=[p3(num_img,nn-1,1)-p3(num_img,nn,1), p3(num_img,nn
            -1,2)-p3(num_img,nn,2), p3(num_img,nn-1,3)-p3(num_img,nn,3)];
        vett3(num_img,nn,:)=[(p3(1,i_min_succ(nn),1)-p3(num_img,nn,1)), (p3(1,
            i_min_succ(nn),2)-p3(num_img,nn,2)), (p3(1,i_min_succ(nn),3)-p3(num_img
            ,nn,3))];
```

```
vett4(num_img,nn,:)=[p3(num_img,nn+1,1)-p3(num_img,nn,1), p3(num_img,nn
    +1,2)-p3(num_img,nn,2), p3(num_img,nn+1,3)-p3(num_img,nn,3)];


prod_vet_mat_12(num_img,nn,:)=[0 -vett1(num_img,nn,3) vett1(num_img,nn,2)
                               vett1(num_img,nn,3) 0 -vett1(num_img,nn,1)
                               -vett1(num_img,nn,2) vett1(num_img,nn,1)
                                   0]*[vett2(num_img,nn,1) vett2(num_img,nn
                                   ,2) vett2(num_img,nn,3)]';
norm12(nn,:)=prod_vet_mat_12(num_img,nn,:)/sqrt((prod_vet_mat_12(num_img,nn
    ,1))^2+(prod_vet_mat_12(num_img,nn,2))^2+(prod_vet_mat_12(num_img,nn,3)
    )^2);

prod_vet_mat_23(num_img,nn,:)=[0 -vett2(num_img,nn,3) vett2(num_img,nn,2)
                               vett2(num_img,nn,3) 0 -vett2(num_img,nn,1)
                               -vett2(num_img,nn,2) vett2(num_img,nn,1)
                                   0]*[vett3(num_img,nn,1) vett3(num_img,nn
                                   ,2) vett3(num_img,nn,3)]';
norm23(nn,:)=prod_vet_mat_23(num_img,nn,:)/sqrt((prod_vet_mat_23(num_img,nn
    ,1))^2+(prod_vet_mat_23(num_img,nn,2))^2+(prod_vet_mat_23(num_img,nn,3)
    )^2);

prod_vet_mat_34(num_img,nn,:)=[0 -vett3(num_img,nn,3) vett3(num_img,nn,2)
                               vett3(num_img,nn,3) 0 -vett3(num_img,nn,1)
                               -vett3(num_img,nn,2) vett3(num_img,nn,1)
                                   0]*[vett4(num_img,nn,1) vett4(num_img,nn
                                   ,2) vett4(num_img,nn,3)]';
norm34(nn,:)=prod_vet_mat_34(num_img,nn,:)/sqrt((prod_vet_mat_34(num_img,nn
    ,1))^2+(prod_vet_mat_34(num_img,nn,2))^2+(prod_vet_mat_34(num_img,nn,3)
    )^2);

prod_vet_mat_41(num_img,nn,:)=[0 -vett4(num_img,nn,3) vett4(num_img,nn,2)
                               vett4(num_img,nn,3) 0 -vett4(num_img,nn,1)
                               -vett4(num_img,nn,2) vett4(num_img,nn,1)
                                   0]*[vett1(num_img,nn,1) vett1(num_img,nn
                                   ,2) vett1(num_img,nn,3)]';
norm41(nn,:)=prod_vet_mat_41(num_img,nn,:)/sqrt((prod_vet_mat_41(num_img,nn
    ,1))^2+(prod_vet_mat_41(num_img,nn,2))^2+(prod_vet_mat_41(num_img,nn,3)
    )^2);
```

```
        norm_mean1(nn,:)=(norm12(nn,:)+norm23(nn,:)+norm34(nn,:)+norm41(nn,:))/4;
        norm_mean(nn,:)=-norm_mean1(nn,:)/sqrt((norm_mean1(nn,1))^2+(norm_mean1(nn
            ,2))^2+(norm_mean1(nn,3))^2);


        if isnan(norm_mean(nn,1)) || isnan(norm_mean(nn,2)) || isnan(norm_mean(nn
            ,3))
            norm_mean(nn,:)=norm_mean(nn-1,:);
        end
        norm_mean(1,:)=norm_mean(2,:);
        norm_mean(end,:)=norm_mean(end-1,:);
    end

    n3(num_img,nn,:) = norm_mean(nn,:); % [n_img n_riga nxnynz]
    if p3(num_img,nn,1)~=0 && p3(num_img,nn,2)~=0 && p3(num_img,nn,3)~=0
        slv=[p3(num_img,nn,1), p3(num_img,nn,2), p3(num_img,nn,3), norm_mean(nn,1),
             norm_mean(nn,2), norm_mean(nn,3)];
        save(filename, 'slv','-append','-ASCII')
    end
end

% Salvo tutte le vaiabili nel file p3n3_oggetto.mat
save('p3n3_oggetto.mat')
```