Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Laurea Magistrale in Ingegneria dell' Automazione

RICOSTRUZIONE 3D CON STEREOSCOPIA DINAMICA

Supervisore
Ruggero Carli
Università di Padova

Co-supervisore
Roberto Polesel
Euclid Labs

Candidato
Simone Bonetto

Anno Accademico 2018/2019

Sommario

Sviluppo di un sistema di visione 3D che utilizza una singola telecamera 2D e l'interazione con un sistema robotico di movimentazione della telecamera per la ricostruzione di nuvole di punti di oggetti meccanici per applicazioni di Bin Picking.

Indice

1 Introduzione

Nell'ambito della visione 3D in applicazioni industriali ci sono varie sistemi in commercio che permettono di ricostruire nuvole di punti della zona visualizzata sfruttando la proiezione di pattern laser o ad ultrasuoni. Questi sistemi sono precisi ma hanno lo svantaggio di avere costi elevati e dimensioni importanti.

Nasce, così, l'idea di creare un sistema ad ingombro ridotto tale da poter essere montato a bordo robot.

Si è pensato di basare tale sistema sui principi della stereoscopia, la quale genera nuvole di punti utilizzando due telecamere poste una accanto all'altra senza l'ausilio di nessuna sorgente di luce esterna.

La stereoscopia classica con due telecamere affiancate non rispecchia l'obiettivo di avere un sistema ad ingombro ridotto, a tal proposito si è pensato di creare un sistema a singola telecamera 2D montata a bordo robot che occupa l'ingombro minimo per un sistema di visione e che cattura immagini in posizioni diverse sulle quali poi applicare la stereoscopia.

Questo tipo di configurazione riesce ad aumentare la qualità delle nuvole di punti, rispetto alla stereoscopia classica, attraverso due principali tecniche: l'acquisizione di immagini a distanza maggiore e l'utilizzo di più di due immagini.

Si richiede, quindi, lo sviluppo di un sistema di visione basato esclusivamente su una telecamera 2D che gestisca un numero qualsiasi di acquisizioni e che possa interagire con un apparecchiatura robotica per determinare gli spostamenti della telecamera.

Il sistema deve essere flessibile e completamente configurabile per poter essere adattato a innumerevoli situazioni, inoltre deve poter calcolare in modo autonomo le posizioni robot dove catturare le nuove immagini che permettano di ottenere la migliore nuvola di punti possibile secondo le informazioni a disposizione.

2
Geometria Epipolare e Rettifica

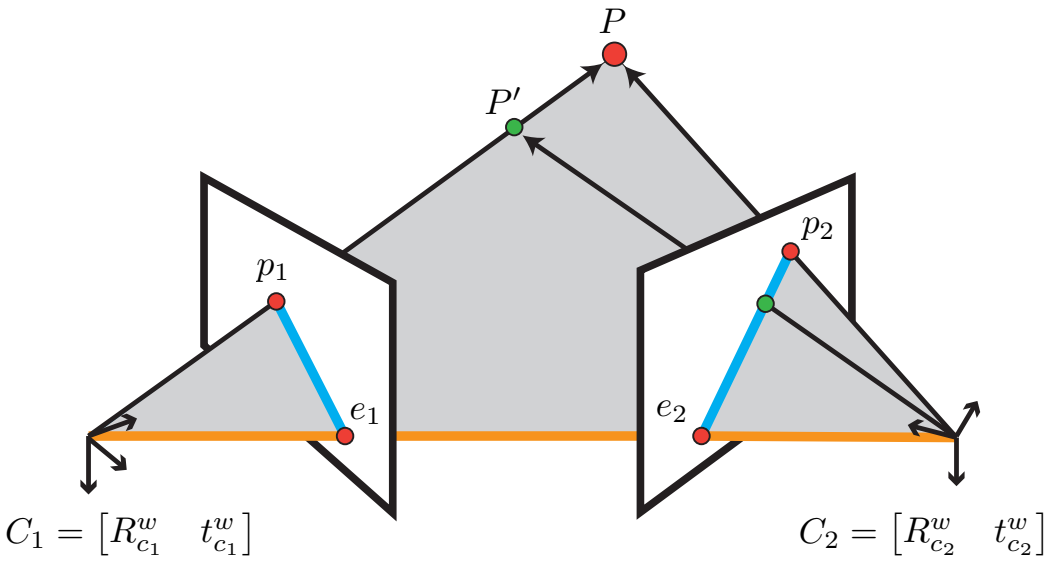


Figura 2.1: Geometria Epipolare: (#) piano epipolare,
(—) baseline, (—) linee epipolari

In sistemi di visione che acquisiscono più di un immagine ci sono delle relazioni interessanti tra le varie telecamere, i punti 3D e le loro proiezioni nel piano delle immagini.
La geometria che descrive la relazione tra una coppia di telecamere è chiamata **geometria epipolare**. Un impostazione base della geometria epipolare è quella illustrata in figura (2.1), dove due telecamere osservano lo stesso punto 3D P, il quale ha proiezione nei relativi piani immagine p_1 e p_2.

I centri focali delle telecamere sono t_1 e t_2, e la linea che li collega è chiamata **baseline**.
Il piano definito dai due centri focali e il punto P è chiamato **piano epipolare**.
I punti dove la baseline interseca i due piani immagine sono noti come **epipoli** e_1 e e_2.
Infine , le linee definite dall'intersezione dal piano epipolare con i due piani immagine sono note come **linee epipolari** l_1 e l_2.

Queste linee appartenenti ad un piano immagine o all'altro, rappresentano tutte le proiezioni di punti 3D che nel piano immagine opposto avrebbero la stessa proiezione. Inoltre tutte le linee epipolari in un piano immagine, definite per diversi punti 3D, hanno come punto in comune l'epipolo.

In situazioni reali, tuttavia non viene fornita la posizione del punto 3D ma conoscendo le posizioni delle telecamere e la proiezione in un piano immagine possiamo definire il piano epipolare e determinare le linee epipolari.

Quindi la conoscenza della geometria epipolare ci consente di creare un forte vincolo tra le coppie di punti nelle 2 immagine senza conoscere la posizione 3D del punto.

Prima di introdurre la matematica della geometria epipolare, viene introdotto il **modello pinhole della telecamera** che permette di mappare un punto 3D, $P^w = [x \quad y \quad z]^T$, in un punto 2D, $p = [u \quad v]^T$ dell'immagine digitale.

Questa mappatura $\mathbb{R}^3 \to \mathbb{R}^2$ è detta **trasformazione proiettiva** M ed è rappresentata dalla seguente relazione:

$$p = \begin{bmatrix} f_u \frac{x^c}{z^c} + c_u \\ f_v \frac{y^c}{z^c} + c_v \end{bmatrix} \quad \text{con} \quad P^c = \begin{bmatrix} R_w^c & t_w^c \\ 0 & 1 \end{bmatrix} \cdot P^w \tag{2.1}$$

che in forma matriciale è espressa da:

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ w \end{bmatrix} = \overbrace{\underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{K} [\, R_w^c \,|\, t_w^c \,]}^{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}^w \implies \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\bar{x}}{w} \\ \frac{\bar{y}}{w} \end{bmatrix} \tag{2.2}$$

dove i parametri c_u e c_v rappresentano la traslazione per spostare l'origine dei punti 2D dal centro del piano immagine (intersezione asse z con il piano) all'angolo in alto a sinistra che è tipicamente l'origine per le immagini digitali; mentre i parametri f_u e f_v convertono i punti del piano immagine espressi in unità fisiche (mm) in punti dell'immagine digitale espressi in pixel. I parametri fin'ora citati sono detti parametri intrinseci.

Le matrici R_w^c e t_w^c, invece, sono parametri estrinseci e trasformano le coordinate del punto 3D espresse nel frame del mondo (w) nelle coordinate espresse nel frame della camera (c). Queste trasformazioni si possono ricavare dalla rototraslazione della telecamera:

$$R_w^c = (R_c^w)^{-1}, \qquad t_w^c = -R_w^c \cdot t_c^w \tag{2.3}$$

Infine notiamo che la trasformazione proiettiva è sviluppata tramite matrici omogenee quindi per ottenere i punti 2D dell'immagine digitale dobbiamo dividere i primi due valori per l'ultima componetene del punto in coordinate omogenee.

Allora le due telecamere C_1 e C_2 del sistema di visione, che consideriamo avere la stessa matrice di parametri intrinseci K, hanno rispettivamente le seguenti trasformazioni proiettive:

$$M_1 = K \cdot [\, R_w^{c_1} \mid t_w^{c_1} \,], \qquad M_2 = K \cdot [\, R_w^{c_2} \mid t_w^{c_2} \,] \tag{2.4}$$

Ora è introdotta la geometria epipolare che si basa sulla **matrice fondamentale** F.
Per la derivazione di questa matrice si considera di essere momentaneamente nel caso semplice in cui la prima camera è posta nell'origine e quindi le due trasformazioni proiettive sono: $M_1 = K\,[\, I \mid 0 \,]$ e $M_2 = K\,[\, R \mid t \,]$.

Di seguito sono riportati i passaggi fondamentali per la derivazione della matrice F.

- Si distingue un punto appartenente all'immagine digitale p espresso in pixel, dal relativo punto nel piano immagine $\bar{p}$ espresso in unità fisiche.
- Il punto $\bar{p}_2^{c_1}$ espresso nel frame della prima camera ha coordinate $\bar{p}_2^{c_1} = R^T \bar{p}_2^{c_2} - R^T t$.
- I vettori $\bar{p}_2^{c_1}$ e $R^T t$ formano il piano epipolare il quale avrà quindi vettore normale $n = R^T t \times (R^T \bar{p}_2^{c_2} - R^T t) = R^T (t \times \bar{p}_2^{c_2})$.
- Anche $\bar{p}_1^{c_1}$ appartiene al piano epipolare, allora vale la relazione $(R^T (t \times \bar{p}_2))^T \bar{p}_1 = 0$ che riscritta introducendo la matrice skew-symmetric $[\cdot]_\times$ diventa $\bar{p}_2^T [t]_\times R \bar{p}_1 = 0$.
- La trasformazione da punti del piano immagine $\bar{p}$ a punti dell'immagine digitale p è: $p = K^{-1} \bar{p}$

Concludendo il vincolo epipolare viene espresso come:

$$p_2^T F p_1 = 0 \quad \text{con} \quad F = K^{-T} [t]_\times R K^{-1} \tag{2.5}$$

Dalla conoscenza della matrice fondamentale è possibile ricavare le linee epipolari: la linea epipolare nella camera C_2 associata al punto p_1 è $l_2 = F p_1$, viceversa $l_1 = F^T p_2$.

Un altra importante caratterista della matrice fondamentale è:

$$F e_1 = F^T e_2 = 0 \tag{2.6}$$

Questo deriva dal fatto che un qualsiasi punto x dell'immagine 1 ha corrispondente linea epipolare $l_2 = Fx$ la quale contiene sicuramente e_2 per definizione, quindi $e_2^T l_2 = e_2^T F x = 0 \ \ \forall x$ allora $e_2^T F = F^T e_2 = \bar{0}$ e analogo per l'altra immagine.

Infine per generalizzare a casi diversi da quello semplice appena illustrato ci basta notare che

$$R = R_1^2 \qquad \text{e} \qquad t = t_1^2 \tag{2.7}$$

Quindi nel nostro caso date le rototraslazioni di due camere C_1 e C_2, le matrici che rientrano nel calcolo della matrice fondamentale si trovano come:

$$R = R_1^2 = (R_{c_2}^w)^{-1} R_{c_1}^w \quad \text{e} \quad t = t_1^2 = (R_{c_2}^w)^{-1}(t_{c_1}^w - t_{c_2}^w) \tag{2.8}$$

2.1 Rettifica classica delle immagini

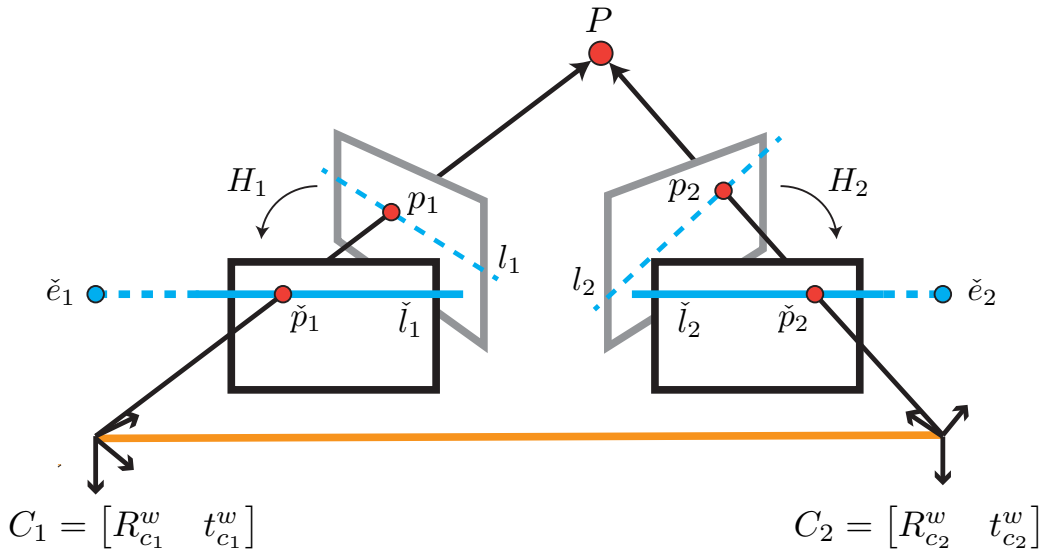


Figura 2.2: Rettifica delle immagini con linee epipolari prima l_i (– –) e dopo $\check{l}_i$ (—) la rettifica

La rettifica delle immagini stereo ha l'obbiettivo di portare tutte le linee epipolari parallele all'asse x in modo da facilitare il processo di ricerca delle corrispondenze.

Infatti dopo la rettifica delle 2 immagini, preso un punto p in una delle due immagini, il punto corrispondente nell'altra immagine dovrà essere cercato soltanto lungo direzione orizzontale x e alla stessa altezza y.

Le linee epipolari hanno questa proprietà soltanto nel caso in cui gli epipoli di entrambe le immagini sono ad infinito lungo l'asse x.

Quindi il compito della rettifica si riassume nella ricerca di due omografie H_1 e H_2, una per immagine, che portano i rispettivi epipoli all'infinito lungo l'asse x.

Per prima cosa allora dobbiamo trovare i due epipoli, che ricordando la proprietà (2.6), possono essere calcolati dalla decomposizione SVD:

$$F = U \cdot W \cdot V^T \implies e_1 = lastCol(V) \text{ e } e_2 = lastCol(U) \tag{2.9}$$

infatti, ricordando che la matrice fondamentale $F = UWV^T$ ha sempre un autovalore nullo, vale $F \cdot V = U \cdot W \implies F \cdot lastCol(V) = U \cdot lastCol(W) = \bar{0}$ e in modo analogo per l'altro epipolo.

Gli epipoli calcolati sono un vettore omogeneo allora devono essere normalizzati in modo da avere ultima componente 1; i due epipoli risultano nella forma $e = [e_x, e_y, 1]^T$.

Ogni epipolo deve essere mappato lungo l'asse orizzontale nel punto infinito $[f, 0, 0]^T$; esistono molte scelte dell'omografia H che portano a questo obiettivo ma una scelta ragionevole che porta a buoni risultati è quella di scegliere l'omografia come composizione di rotazioni e traslazioni rispetto al centro dell'immagine.

Il primo passo è traslare tutti i punti dell'immagine e quindi anche l'epipolo in modo che il centro immagine sia $[0, 0, 1]^T$; può essere fatto applicando la matrice di traslazione

$$T = \begin{bmatrix} 1 & 0 & -\frac{width}{2} \\ 0 & 1 & -\frac{height}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

L'epipolo risultante $e' = Te$ viene poi ruotato in modo da posizionarlo nel punto $[f, 0, 1]^T$ con la matrice di rotazione

$$R = \begin{bmatrix} \frac{e'_x}{\sqrt{e'^2_x + e'^2_y}} & \frac{e'_y}{\sqrt{e'^2_x + e'^2_y}} & 0 \\ \frac{-e'_y}{\sqrt{e'^2_x + e'^2_y}} & \frac{e'_x}{\sqrt{e'^2_x + e'^2_y}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Infine per ottenere il punto all'infinito desiderato $[f, 0, 0]^T$ viene applicata la seguente trasformazione, la quale è l'unica non riconducibile traslazioni o rotazioni

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{f} & 0 & 1 \end{bmatrix} \tag{2.12}$$

L'omografia finale desidera è la composizione delle matrici appena illustrate:

$$H = T^{-1} \cdot G \cdot R \cdot T \tag{2.13}$$

Questa omografia può essere applicata ad entrambi gli epipoli, con le dovute modifiche, mappandoli entrambi all'infinito ma in questo modo non raggiungiamo l'obiettivo della rettifica perché linee epipolari corrispondenti non avranno la stessa coordinata y.

Per raggiungere l'obbiettivo esatto della rettifica prendiamo H_2 calcolata con il metodo appena illustrato e cerchiamo H_1 che oltre a mappare e_1 all'infinito porta le linee epipolari corrispondenti alla stessa y.

Esiste un teorema il quale afferma che data la trasformazione di rettifica H_2 e la matrice F, la trasformazione H_1 che abbina le linee epiolari ha la forma $H_1 = H_A H_0$ con $H_0 = H_2 M$, $M = [e]_\times F + ev^T$, $v = [1,1,1]^T$ e con H_A matrice arbitraria nella forma

$$H_A = \begin{bmatrix} a_1 & a_2 & a_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Riassumendo siamo arrivati a trovare le due matrici di rettifica nella seguente forma a meno della matrice H_A

$$H_1 = H_A \cdot H_0 \qquad \text{e} \qquad H_2 = T^{-1} \cdot G \cdot R \cdot T \tag{2.15}$$

Per stimare la restante matrice H_A, la rettifica classica[bib], usa dei punti corrispondenti nelle due immagini p_1^i e p_2^i individuati cercando i punti chiave nelle due immagini e associandoli in base ad un descrittore e all'appartenenza alla linea epipolare. Quindi calcola H_A che risolve il seguente problema di minimo:

$$\underset{H_1}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| H_1 p_1^i - H_2 p_2^i \right\|^2 \implies \underset{H_1}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| H_A \bar{p}_1^i - \bar{p}_2^i \right\|^2 \tag{2.16}$$

dove $\bar{p}_1^i = H_0 p_1^i$ e $\bar{p}_2^i = H_2 p_2^i$. Il problema si trasforma nella seguente minimizzazione

$$\underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{n} (a_1 \bar{x}_1^i + a_2 \bar{y}_1^i + a_3 - \bar{x}_2^i)^2 + \cancel{(\bar{y}_1^i - \bar{y}_2^i)^2} \tag{2.17}$$

che è riconducibile facilmente alla minimizzazione ai minimi quadrati $\|\Phi \mathbf{a} - b\|^2$

$$\Phi = \begin{bmatrix} \bar{x}_1^1 & \bar{y}_1^1 & 1 \\ & \vdots & \\ \bar{x}_1^n & \bar{y}_1^n & 1 \end{bmatrix}, \quad b = \begin{bmatrix} \bar{x}_2^1 \\ \vdots \\ \bar{x}_2^n \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \implies \hat{\mathbf{a}} = (\Phi^T \Phi)^{-1} \Phi^T b \tag{2.18}$$

Inoltre per potenziare la stima $\hat{a}$ è possibili implementare RANSAC individuando la stima che ha maggiori inliers secondo un parametro che rappresenta il massimo errore; questo permette di essere robusti nel caso di avere corrispondenze tra le due immagini errate.

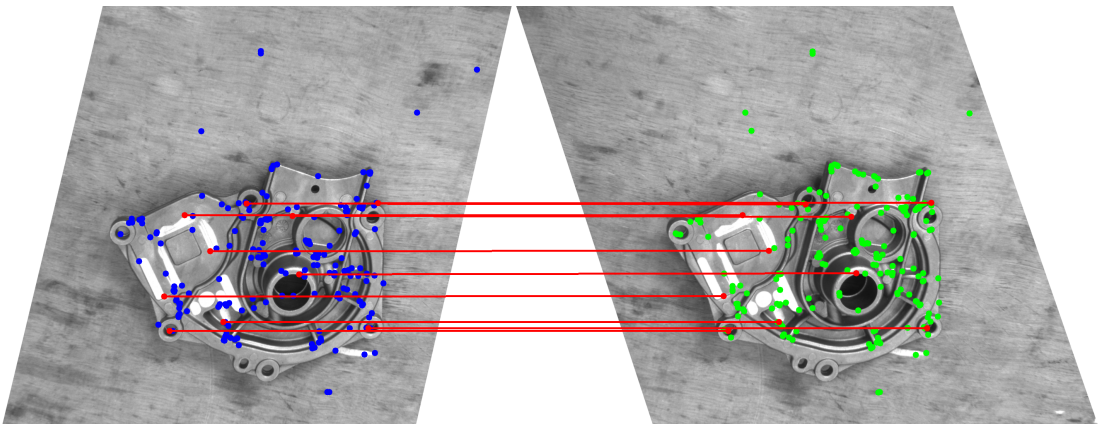


Figura 2.3: Rettifica Classica: da sinistra a destra immagini 1 e 2 rettificate con H_1 e H_2. (—) linee epipolari che collegano punti corrispondenti sono orizzontali

Nella rettifica classica, come appena descritto, per stimare gli ultimi 3 parametri della matrice H_A è necessario individuare dei punti corrispondenti nelle due immagini, ma questo non è un compito semplice in ambienti fortemente monotoni dove l'algoritmo deve operare i quali presentano molti oggetti tutti uguali.

In questa sezione viene proposta una diversa stima della matrice H_A che non richiede la ricerca di punti corrispondenti nelle due immagini.

La matrice omografica H_0, senza la stima della matrice H_A, soddisfa già l'obiettivo di rettifica ponendo pixel corrispondenti alla stessa coordinata x ma questi potrebbero essere notevolmente deformati fino ad non essere visibili e quindi non permettere il matching per stime di H_A errate.

Da questa analisi possiamo ricondurre la stima di H_A ad un problema di ricerca della matrice che introduce la minor deformazione all'immagine di partenza permettendo la miglior visualizzazione.

Le matrici omografiche che non deformano l'immagine in alcun modo sono quelle isometriche:

$$\begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.19}$$

le quali permettono rotazioni e traslazione tramite le corrispondenti matrici R e T ma non creano ne distruggono pixels. Cerchiamo quindi la matrice di rettifica $H_1 = H_A \cdot H_0$ con caratteristiche più simili possibili all'insieme delle omografie isometriche.

Un modo robusto per incapsulare questa similarità è l'analisi dello Jacobiano:

$$J(H_A \cdot H_0, p) = \begin{bmatrix} \frac{\partial \bar{x}}{\partial x} & \frac{\partial \bar{x}}{\partial y} \\ \frac{\partial \bar{y}}{\partial x} & \frac{\partial \bar{y}}{\partial y} \end{bmatrix} \tag{2.20}$$

il quale nelle omografie isometriche vale $J = R$ ed essendo R una matrice di rotazione e quindi ortonormale soddisfa la condizione $R^T R = I$ ovvero con autovalori singolari $\sigma_{1/2}$ entrambi uguali a 1 che non introducono deformazione.

L'idea quindi è trovare H_A che che permette di ottenere $H_1 = H_A \cdot H_0$ con Jacobiano che ha decomposizione in SVD con autovalori singolari più simili possibile a 1, ovvero minimizzando la creazione ($\sigma > 1$) e la distruzione ($\sigma < 1$) di pixels.

Con questo obiettivo valutiamo gli autovalori singolari dello Jacobiano J_i in diversi punti dell'immagini p_i cercando la matrice H_A che minimizza la seguente funzione di perdita:

$$S(a_1, a_2) = \sum_{i=1}^{n} [(\sigma_1(J_i) - 1)^2 + (\sigma_2(J_i) - 1)^2] \tag{2.21}$$

Lo Jacobiano J_i è calcolato prima ottenendo le funzioni fratte $\bar{x}$ e $\bar{y}$ derivanti dall'applicazione di un punto immagine $p_i = [x \quad y \quad 1]$ all'omografia incognita $H(a_1, a_2, a_3) = H_A \cdot H_0$ e poi derivando tali funzioni rispetto alle variabili x e y.

Lo Jacobiano finale è riportato in (2.22) dove è espresso evidenziando le variabili da stimare a_1 e a_2 in composizione con A, B, C, D, E, F che sono dei numeri. Inoltre si nota che giustamente a_3 non è presente, in quanto, in fase di derivazione scompare infatti esso non influisce nelle deformazioni ma rappresenta soltanto una traslazione lungo l'asse x e può essere stimato a posteriori in modo da porre il centro dell'immagini di partenza nel centro dell'immagine rettificata per quanto riguarda la coordinata nell'asse orizzontale.

$$J_i(H_A \cdot H_0, p_i) = \begin{bmatrix} a_1A + a_2B & a_1C + a_2D \\ E & F \end{bmatrix} \tag{2.22}$$

La minimizzazione della funzione di perdita (2.21) viene affrontata tramite due algoritmi: un primo algoritmo che cerca di trovare in modo robusto una buona stima dei parametri a_1 e a_2 sfruttando una linearizzazione agli autovalori singolari e successivamente l'algoritmo di Newton-Raphson per arrivare alla stima ottima dei parametri

Il primo algoritmo, **minimizzazione con linearizzazione agli autovalori singolari**, per semplificare il problema sfrutta la seguente approssimazione:

$$J = UWV^T \quad \Longrightarrow \quad W = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} = U^{-1}JV^{-T} \tag{2.23}$$

dalla quale, conoscendo le matrici J, U, V è possibile ricavare gli autovalori singolari come:

$$\sigma_1 = a_1G + a_2L + M, \qquad \sigma_2 = a_1P + a_2Q + N \tag{2.24}$$

Gli autovalori allora risultano come composizione lineare dei parametri da stimare a_1 e a_2; per questo è possibile ricavare in forma chiusa la soluzione che minimizza la funzione di perdita in questa linearizzazione. Infatti dopo aver calcolato le derivata prime della funzione $S(a_1, a_2)$ in (2.25), è possibile trovare in modo non complesso i parametri che portano tali derivate a zero.

$$\begin{cases} \frac{\partial S}{\partial a_1} = \sum_{1=1}^{n} [2(\sigma_1^i - 1)G^i + 2(\sigma_2^i - 1)P^i] = 0 \\ \frac{\partial S}{\partial a_2} = \sum_{1=1}^{n} [2(\sigma_1^i - 1)L^i + 2(\sigma_2^i - 1)Q^i] = 0 \end{cases} \tag{2.25}$$

Una modifica che incrementa la robustezza dell'algoritmo è quella di calcolare i parametri incogniti, non come quelli che minimizzano la funzione sommatoria $S(a_1, a_2)$, ma come quelli che soddisfano in Least-Square-Sense il vincolo di derivata uguale a 0 per ogni punto p_i valutato.

$$\hat{X} = \underset{X \in \mathbb{R}^2}{\operatorname{argmin}} \|\Phi X - Y\|_2^2 \quad \Longrightarrow \quad \hat{X} = (\Phi^T\Phi)^{-1}\Phi^T Y \quad \text{con} \quad X = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{2.26}$$

L'algoritmo viene sviluppato in modo iterativo, ovvero partendo da parametri incogniti qualsiasi e per ognuno degli n punti valutati p_i, viene calcolato lo Jabobiano J_i e la relativa linearizzazione attorno ai parametri incogniti attuali.

Vengono calcolati i parametri che meglio soddisfano questi $2n$ vincoli in Least-Square-Sense, poi viene aggiornata la stima attuale di a_1 e a_2 e l'algoritmo riprende iterativamente.

Il secondo algoritmo sfrutta la procedura di **Newton–Raphson** che permette, data una buona stima trovata tramite l'algoritmo precedente, di arrivare ai parametri ottimi.
In questo algoritmo ci servono le due funzioni esatte che esprimono gli autovalori singolari in funzione dei parametri incogniti a_1 e a_2. Gli autovalori singolari sono, per definizione di decomposizione SVD, la radice quadrata di degli autovalori della matrice $J^T J$ dove J è riportata in (2.22) ed assumono la forma:

$$\lambda_{1,2}(J^T J) = \frac{f_1(a_1, a_2) \pm \sqrt{f_2(a_1, a_2)}}{2} \quad \Longrightarrow \quad \sigma_{1,2} = \sqrt{\lambda_{1,2}} \tag{2.27}$$

Trovate queste due funzioni possiamo calcolare le derivate prime e secondo della funzione di perdita $S(a_1, a_2)$ rispetto ai parametri a_1 e a_2 e applicare l'algoritmo iterativo standard di Newton-Raphson:

$$\nabla S(a_1, a_2) = \begin{bmatrix} \frac{\partial S}{\partial a_1} \\ \frac{\partial S}{\partial a_2} \end{bmatrix}, \qquad \nabla^2 S(a_1, a_2) = \begin{bmatrix} \frac{\partial^2 S}{\partial^2 a_1} & \frac{\partial^2 S}{\partial a_1 \partial a_2} \\ \frac{\partial^2 S}{\partial a_2 \partial a_1} & \frac{\partial^2 S}{\partial^2 a_2} \end{bmatrix} \tag{2.28}$$

$$\hat{X}^{k+1} = \hat{X}^k - \triangle_k \qquad \text{con} \qquad \triangle_k = (\nabla^2 S(\hat{X}^k))^{-1} \nabla S(\hat{X}^k) \tag{2.29}$$

Infine avendo le stime ottime di $\hat{a}_1$, $\hat{a}_2$ possiamo calcolarci l'ultimo parametro a_3 in modo che il centro dell'immagine prima della rettifica $C_{img} = [\frac{w}{2} \quad \frac{h}{2} \quad 1]^T$ resti al centro dopo la rettifica almeno per quanto riguarda la componente nell'asse orizzontale:

$$\hat{a}_3 \quad \Longrightarrow \quad C_{img,x} = \frac{(H_A \cdot H_0 \cdot C_{img})[0]}{(H_A \cdot H_0 \cdot C_{img})[2]} \tag{2.30}$$

I due algoritmi sono utilizzati in sequenza per compensare le mancanze uno dell'altro ed arrivare ai parametri ottimi.

Infatti il primo algoritmo ha la proprietà di riuscir a calcolare una stima pseudo ottima partendo da valori iniziali casuali ma di contro, in generale, non riesce ad arrivare alla stima ottima (se iterato per molte iterazioni arriverebbe a parametri molto simili a quelli ottimi).

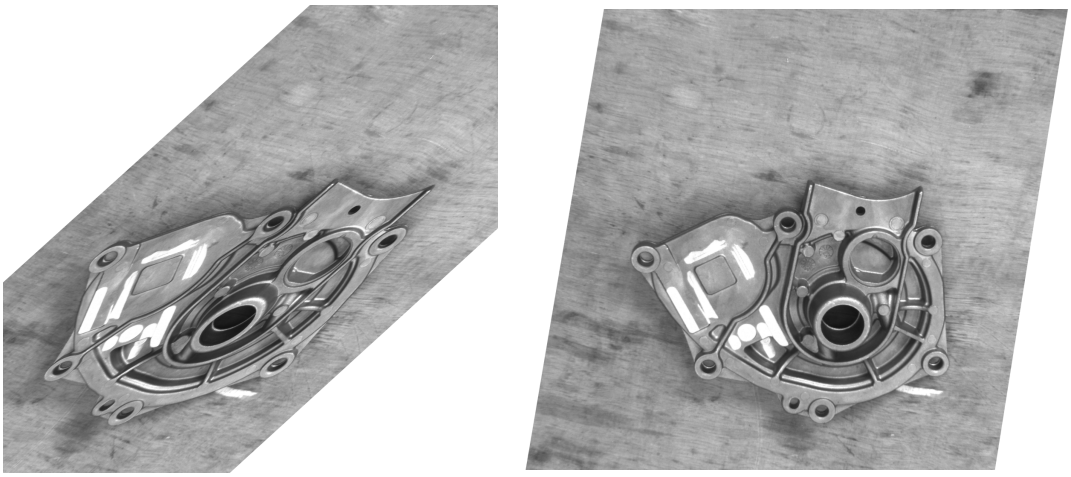
Quindi il primo algoritmo viene utilizzato per inizializzare il secondo algoritmo che altrimenti partendo da stima iniziale casuale calcolerebbe stime assolutamente errate ma che se ben inizializzato arriva in modo robusto ai parametri ottimi.

Le figure di seguito riportano le immagini rettificate dopo 5 iterazioni del primo algoritmo e dopo altre 5 iterazioni del secondo algoritmo partendo dalla stima del precedente e si nota come il primo dopo 5 iterazioni è arrivata ad una stima buona infatti l'oggetto è abbastanza visibile (2.4a) ma non è la stima ottima che invece si vede nell'immagine rettificata del secondo algoritmo (2.4b).

Infine si può osservare che l'immagine finale di questi 2 algoritmi è pressoché identica a quella ottenuta dalla rettifica classica in figura (2.3) ma si ricorda che in questi 2 algoritmi non è stato necessario individuare punti corrispondenti tra le due immagini.

Da notare però che apparentemente la rettifica con punti corrispondenti ha la proprietà aggiuntiva di applicare in modo consapevole anche ridimensionamento lungo l'asse x all'immagine 1 nel caso in cui le due immagini siano prese a distanze molto diverse dall'oggetto. Questo è necessario per agevolare la fase successiva di ricerca delle corrispondenze in particolare per quel gruppo di algoritmi che come vedremo analizzano l'intera immagine.

La rettifica per minimizzazione delle deformazioni invece non ha consapevolezza del ridimensionamento dell'oggetto ma implicitamente lo esegue in quanto il ridimensionamento lungo l'asse y è dato dalla geometria epipolare nel teorema (2.14) lasciando da stimare soltanto la matrice H_A la quale porta modifiche solo lungo l'asse x ma stimando la rettifica con minor deformazione ridimensionerà l'immagine anche lungo l'asse x per adattarsi nel modo migliore al ridimensionamento apportato nell'asse y.



(a) Alg. con linearizzazione

(b) Alg. Newton-Raphson

Figura 2.4: Rettifica minimizzando le deformazioni locali:
immagine 1 rettificata con i due diversi algoritmi

In conclusione viene analizzato e risolto un problema che si riscontra nella rettifica per minimizzazione delle deformazioni locali.

Al termine di tale minimizzazione l'algoritmo può aver converto ad una delle immagini in figura (2.5a), (2.5b), (2.5c) o (2.5d), le quali presentano tutte lo stesso grado di deformazione e quindi hanno tutte ugualmente parametri ottimi per la minimizzazione. Solo una di queste, però, è quella giusta che soddisfa l'obbiettivo di corrispondenza con la rettifica H_2.

Queste immagini differisco tra loro per una rotazione di $180°$ attorno al centro immagine, per un operazione di specchiare l'immagine rispetto l'asse verticale passante per il centro oppure per una composizione di queste.
Queste differenze possono essere codificate rispettivamente attraverso le omografie aggiuntive H_{180}, H_{mirror} e $H_{mirror} \cdot H_{180}$ riportate di seguito:

$$Rot_{180} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Mirr = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{cases} H_{180} = T^{-1} \cdot Rot_{180} \cdot T \\ H_{mirror} = T^{-1} \cdot Mirr \cdot T \end{cases}$$

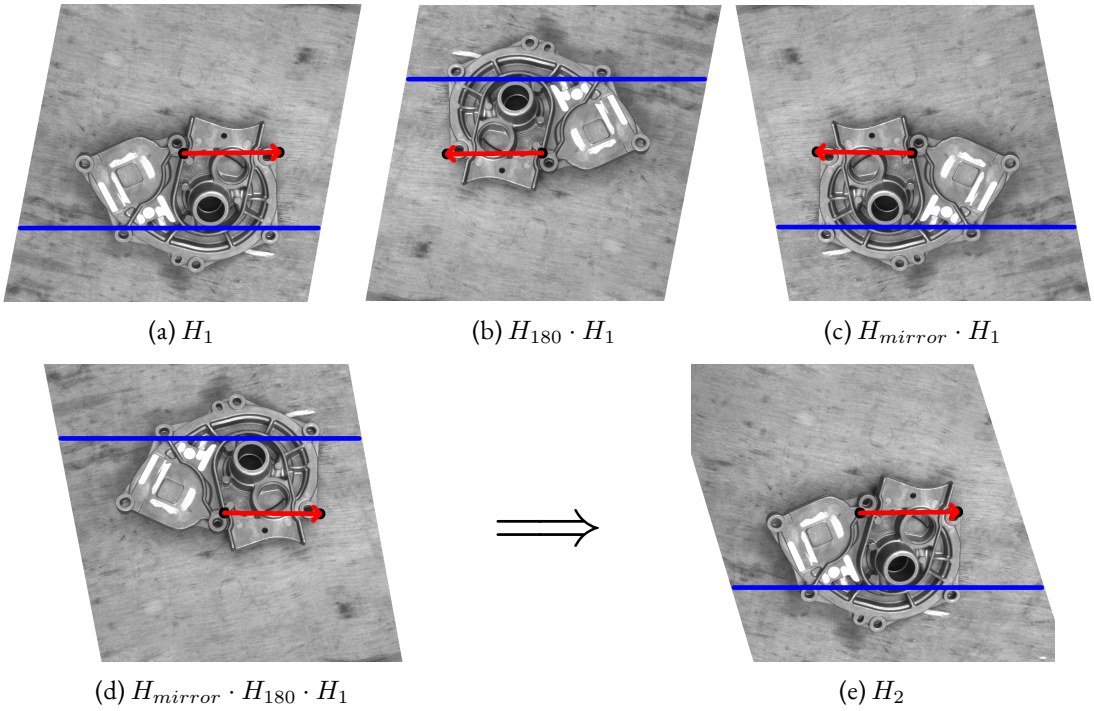
dove T, espressa precedentemente in (2.10), rappresenta la traslazione per riportare il tutto rispetto al centro immagine.

Per eliminare questi gradi di ambiguità si analizzano le possibili omografie $\hat{H}_1 = H \cdot \bar{H}_1$ dove l'omografia $\bar{H}_1$ è quella calcolata al termine della minimizzazione mentre H è quella che codifica le differenze elencate con $H \in \{I, H_{180}, H_{mirror}, H_{180} \cdot H_{mirror}\}$ e si individua quale combacia con l'omografia H_2 riportata in figura (2.5e) secondo alcuni controlli.

Per prima cosa si controlla quale omografia $\hat{H}_1$ produce matrice fondamentale post rettifica $\bar{F} = H_2^{-T} F \hat{H}_1^{-1}$ che induce linee epipolari orizzontali corrispondenti tra l'immagine 1, rettificata con $\hat{H}_1$, e l'immagine 2, rettificata con l'omografia fissa H_2, le quali linee devono avere la stessa coordinata y. Questo ci permette di eliminare i casi in cui all'omografia esatta H_1 è aggiunta un omografia che porta una rotazione di $180°$ come ad esempio quelle in figura (2.5b) e (2.5d).

Resta l'ambiguità dovuta all'operazione di specchiare che inverte i pixel lungo l'asse x quindi non è eliminata controllando le linee epipolari. Questo effetto è dovuto dall'osservare l'oggetto dalla posizione opposto lungo l'asse z della telecamera ovvero osservando l'oggetto da 'sotto'; questa ambiguità può essere risolta decomponendo le omografia $\hat{H}_1$ restanti nelle roto-traslazioni di spostamento corrispondenti che approssimativamente inducono tali omografie e scegliendo quella che porta la telecamera 1 dopo lo spostamento ad avere l'asse z congruente con la telecamera 2 dopo lo spostamento indotto da H_2. Oppure in modo più semplice si può prendere un punto 3D e riproiettarlo nelle immagini 1 e 2, modificare poi i rispettivi punti 2D secondo le omografie $\hat{H}_1$ e H_2 e trasformare tali punti in vettori rispetto al centro immagine; infine scegliere l'omografia $\hat{H}_1$ con vettore concorde con quello di H_2 lungo x.

Alla fine di tutti i controlli resterà soltanto una omografia $\hat{H}_1$ che induce immagine rettifica corrispondente a quella rettificata con H_2; nel caso specifico sarà l'immagine in figura (2.5a).



(a) H_1 (b) $H_{180} \cdot H_1$ (c) $H_{mirror} \cdot H_1$

(d) $H_{mirror} \cdot H_{180} \cdot H_1$ (e) H_2

Figura 2.5: Eliminazione delle ambiguità analizzando le linee epipolari (—) e l'orientazione rispetto al centro (—>)

3
Calibrazione Telecamera

Nella sezione precedente si è analizzata la geometria epipolare la quale ci permette di eliminare un grado di libertà nella ricerca di corrispondenze tra immagini avendo a disposizione la matrice fondamentale F.

La matrice fondamentale F può essere calcolata in due modi: il primo metodo è quello descritto nella sezione precedente dove è calcolata dalla conoscenza delle posizioni delle telecamere e della matrici di proiezione; il secondo invece ricostruisce la matrice fondamentale solamente dalla immagini avendo però a disposizione dei punti chiave tra le due immagini. Come spiegato precedentemente la ricerca di punti chiave è assolutamente un compito difficile e poco robusto in ambienti monotoni dove ci poniamo di lavorare quindi è scelto di utilizzare il primo metodo.

Si ricorda quindi la formula della matrice fondamentale $F = K^{-T}[t]_{\times}RK^{-1}$ dove K è la matrice di proiezione della telecamera e R e t dipendono dalle posizioni relative tra le due foto acquisite.

La configurazione meccanica a cui gli algoritmi implementati devono interfacciarsi è costituita da un braccio robotico sul quale è montata la telecamera. Dall'interfacciamento con il robot è possibile ricavare i ogni istante la rototraslazione della flangia, zona a cui è applicato il supporto per la telecamera, e quindi la posizione della telecamera a meno di una rototraslazione tra la flangia e la telecamera.

Nelle due sezioni successive vengono analizzate i processi per individuare la matrice di proiezione della telecamera e la rototraslazione tra flangia e telecamera che ci permettono di calcolare in modo preciso la matrice fondamentale.

La calibrazione della telecamera è un passaggio necessario quando si vuole estrarre informazioni metriche da immagini 2D. Viene utilizzato un classico algoritmo con scacchiera illustrato in dettaglio in [bib], di seguito vengono riportati i passaggi fondamentali.

L'algoritmo utilizza diverse immagini inquadranti una scacchiera raccolte mantenendo fissa la scacchiera e cambiando la posizione della telecamera. In questa impostazione è possibile individuare in modo robusto e preciso i punti di intersezione tra gli angoli dei quadrati nelle immagini 2D. Inoltre è possibile esprimere in modo accurato i corrispondenti punti 3D rispetto ad un riferimento posto al primo punto della scacchiera e conoscendo la dimensione reale dei quadrati.

L'algoritmo ha l'obiettivo di calcolare i parametri intrinseci (K) e i parametri estrinseci (R e t) che formano il classico modello pin-hole:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}}_{M} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \underbrace{K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}}_{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.1}$$

dove la matrice di proiezione M è stata semplificata in quanto tutti i punti 3D espressi rispetto al riferimento della scacchiera hanno coordinata $Z = 0$. Come primo passo deve essere individuata la matrice omografica $H = [h_1\ h_2\ h_3]$ che trasforma i punti dal piano della scacchiera al piano immagine:

$$\begin{cases} u_i = \dfrac{h_1^T P_i}{h_3^T P_i} \\ \\ v_i = \dfrac{h_2^T P_i}{h_3^T P_i} \end{cases} \implies \begin{bmatrix} \vdots & \vdots & \vdots \\ P_i^T & 0 & -u_i P_i^T \\ 0 & P_i^T & -v_i P_i^T \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0 \tag{3.2}$$

Ogni punto genera 2 vincoli quindi per soddisfare i 9 gradi di libertà meno il fattore di scala ininfluente in qualsiasi omografia bastano 4 punti ma in pratica ne vengono utilizzati di più e il sistema risolto in modo robusto con SVD: la soluzione del sistema $Ax = 0$ dove x corrisponde alle colonne della matrice H poste in unico vettore colonna, ha come soluzione l'ultima colonna della matrice V che compone la decomposizione $A = UWV^T$ ovvero quella relativa all' autovalore singolare più piccolo.

La matrice H calcolata deve essere decomposta nei parametri intrinseci ed estrinseci. Per far questo ricordiamo che $[h_1\ h_2\ h_3] = \lambda K[r_1\ r_2\ t]$ con λ fattore di scala e imponendo il vincolo di ortonormalità alle colonne della matrice di rotazione ricaviamo le seguenti equazioni:

$$\begin{cases} r_1^T r_2 = 0 \\ r_1^T r_1 = r_2^T r_2 = 1 \end{cases} \implies \begin{cases} h_1^T B h_2 = 0 \\ h_1^T B h_1 = h_2^T B h_2 \end{cases} \quad \text{con} \quad B = (KK^T)^{-1} \tag{3.3}$$

Anche in questo caso viene formulato un sistema $Ax = 0$ con x formata dalle 6 componenti indipendenti della matrice B che è simmetrica. Per la composizione della matrice A si nota che ogni immagine ha matrice H diversa perché la posizione della telecamera è cambiata e si nota inoltre che per ogni immagine è possibile generare 2 vincoli sulla matrice B quindi per individuare in modo univoco la soluzione servono almeno 3 immagini ma come prima è consigliato acquisirne in numero superiore e poi risolvere con SVD.

I 4 parametri della matrice K possono essere ricalcolati in forma chiusa da B sapendo a cosa corrispondono le singole componenti di tale matrice rispetto ai paramenti di K ($B_i = f_i(f_x, f_y, c_x, c_y)$) e poi componendo le varie formule inverse.

I parametri estrinseci per ogni immagine vengono calcolati anche essi come formula inversa delle equazioni in (3.3) ricordando che le colonne ortonormali della matrice R, in aggiunta alle due proprietà già elencate precedentemente, soddisfano anche $\| r_i \| = 1$ e $r_i = r_j \times r_h$ quindi si può arrivare alle seguenti equazioni:

$$\begin{cases} r_1 = \lambda' K^{-1} h_1 \\ r_2 = \lambda' K^{-1} h_2 \\ r_3 = r_1 \times r_2 \\ t = \lambda' K^{-1} h_3 \end{cases} \quad \text{con} \quad \lambda' = \frac{1}{\| K^{-1} h_1 \|} = \frac{1}{\| K^{-1} \| h_2} \tag{3.4}$$

La matrice $\hat{R} = [r_1\ r_2\ r_3]$ così calcolata potrebbe non essere precisamente una matrice di rotazione quindi è necessario cercare la matrice di rotazione esatta più simile a quella calcolata in termini di norma di Frobenius:

$$\underset{R^T R = I}{\operatorname{argmin}} \| R - \hat{R} \|_F^2 \quad \text{con} \quad \hat{R} = UWV^T \quad \Longrightarrow \quad R = UV^T \tag{3.5}$$

Da notare che i parametri estrinseci, a differenza di quelli intrinseci, dipendono da dove è stato individuato il riferimento della scacchiera, ovvero il primo punto, per la specifica immagine. In questa fase non sarebbe necessario individuare nella scacchiera un riferimento in modo univoco ma per la sezione successiva si.

Come mostrato in figura (3.1) il riferimento univoco viene individuato aggiungendo un piccolo cerchio vicino al punto della scacchiera che deve essere individuato come punto del riferimento e traslato verso l'asse x; il cerchio viene individuato in modo robusto cercando solo nelle zone rispetto alla scacchiera dove potrebbe essere presente e successivamente i punti immagine sono ordinati per crescere prima in colonna lungo l'asse x e poi nelle righe successive. In questo modo individuiamo i punti immagine sempre allo stesso modo e gli relazioniamo sempre agli stessi punti 3D ordinati ugualmente ed espressi rispetto al riferimento cosi le rotazioni R e le traslazioni t che otteniamo per ogni immagine rappresentano la trasformazione tra camera e riferimento T^c_{ref}.

Trovati tutti i parametri intrinseci ed estrinseci, essi possono essere raffinati minimizzando la funzione di costo S composta dalla somma degli errori di riproiezione di ogni punto:

$$S = \sum_{i=0}^{N} \sum_{j=0}^{M} \| \, p_{i,j} - f(K, R_i, t_i, P_j) \, \|^2 \tag{3.6}$$

dove N rappresenta il numero delle immagini, M il numero di punti per ogni immagine, $f(\cdot)$ la funzione di riproiezione, p e P rispettivamente i punti in 2D e 3D. La funzione viene minimizzata in modo iterativo con il metodo di Levenberg-Marquardt.

La calibrazione della telecamera, oltre a calcolare parametri intrinseci ed estrinseci, ha anche il compito di rimuovere la distorsione introdotta dalle lenti. Per far questo viene introdotto il modello radiale che definisce la distorsione dell'immagine in funzione della distanza dal centro e di alcuni parametri, nella descrizione di seguito sarà esposto il modello a 2 parametri ma in applicazione reale può essere usato il modello fino a 14 parametri.

Il modello radiale a 2 parametri k_1 e k_2 è il seguente:

$$\begin{cases} \hat{u} = u + (u - u_0)[k_1 r^2 + k_2 r^4] \\ \hat{v} = u + (v - v_0)[k_1 r^2 + k_2 r^4] \end{cases} \quad \text{con} \quad r^2 = \left(\frac{u - u_0}{f_u} \right)^2 + \left(\frac{v - v_0}{f_v} \right)^2 \tag{3.7}$$

dove $(\hat{u}, \hat{v})$ sono i punti individuati nell'immagine soggetti a distorsione, (u, v) sono i punti ideali calcolati come riproiezione dei punti 3D, (u_0, v_0) rappresentano il centro dell'immagine e r è il raggio del punto ideale rispetto al centro.

Per ogni punto 2D in ogni immagine è possibile individuare la seguente coppia di equazioni:

$$\begin{bmatrix} (u - u_0)r^2 & (u - u_0)r^4 \\ (v - v_0)r^2 & (v - v_0)r^4 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \hat{u} - u \\ \hat{v} - v \end{bmatrix} \tag{3.8}$$

quindi dati M punti in N immagini otteniamo $2NM$ equazioni che possono essere inserite in un equazione matriciale $D \cdot k = d$ che viene risolta in Least-Square-Sense con:

$$k = (D^T D)^{-1} D^T d \tag{3.9}$$

Infine questi passaggi appena illustrati possono essere ripetuti fino alla convergenza:

1. Stima dei parametri intrinseci (K) ed estrinseci (R_i, t_i)
2. Stima dei parametri di distorsione (k_1, k_2)
3. Rimuovo la distorsione dalle immagini
4. Riprendo dal punto 1 con le immagini non distorte
5. Continuo fino alla convergenza

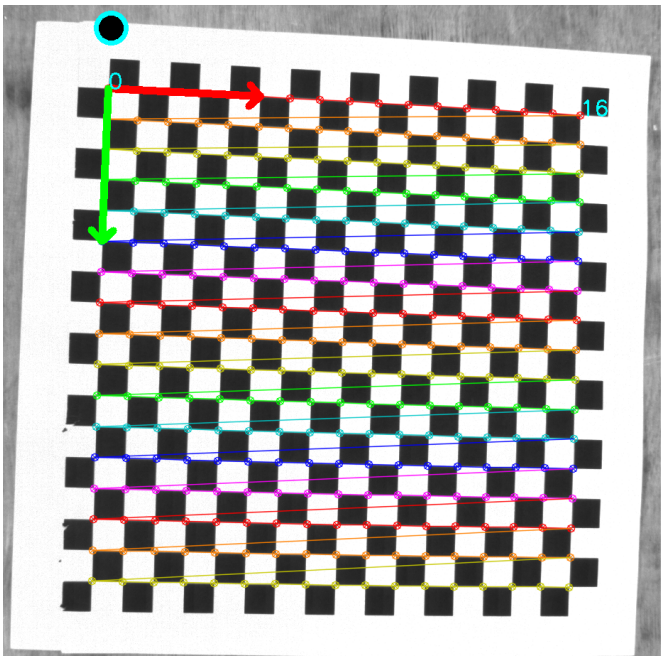


Figura 3.1: Scacchiera con cerchio aggiuntivo per individuare in modo univoco in ogni immagine il riferimento dato dall'asse x (—) e dall'asse y (—)

3.2 Calibrazione della telecamera sul robot

In questa sezione è illustrato come calcolare la matrice di rototraslazione T_c^f la quale permette di individuare la posizione e orientazione della telecamera conoscendo quelle della flangia del robot ($T_c^w = T_f^w \cdot T_c^f$).

Per questa calibrazione sfruttiamo le immagini utilizzate per la calibrazione precedente dove, per ogni foto acquisita in posizione differente ma lasciando fissa la scacchiera, viene prelevata anche l'informazione sulla rototraslazione della flangia del robot T_f^w espressa rispetto al riferimento del mondo w che di solito è posto nella base del robot. Sempre dalla calibrazione precedente è stato possibile individuare una stima della trasformazione tra la telecamera e il riferimento T_{ref}^c per ogni immagine acquisita.

Nota la posizione del riferimento della scacchiera T_{ref}^w rispetto al mondo e per ogni immagine j è possibile calcolare la posizione della telecamera e di conseguenza la rototraslazione cercata conoscendo la posizione della flangia quando è stata acquisita quell'immagine:

$$T_{c_j}^w = T_{ref}^w \cdot (T_{ref}^{c_j})^{-1} \quad \Longrightarrow \quad T_{c_j}^{f_j} = (T_{f_j}^w)^{-1} \cdot T_{c_j}^w \tag{3.10}$$

e siccome la trasformazione tra telecamera e flangia è fissa ($T_c^f \cong T_{c_j}^{f_j}$) si può trovare quella media tra tutte quelle stimate per ogni immagine j.

Queste metodo non ottiene una buona stima in quanto innanzitutto la rototraslazione T^c_{ref} non è precisa e inoltre difficilmente si riesce ad ottenere la posizione del riferimento della scacchiera in modo preciso, quindi la rototraslazione cercata sarà doppiamente influenzata dalla propagazione di questi errori.

Per calcolare questa matrice viene proposto un algoritmo che minimizza l'errore tra la riproiezione di un punto 3D P^w_i, chiamata $\bar{p}_i$, e il punto p_i individuato in modo univoco nell'immagine:

$$S = \sum_i \left\| \bar{p}_i - p_i \right\|^2, \qquad \text{con} \qquad \bar{p}_i = K[R^c_w | t^c_w] \cdot P^w_i \tag{3.11}$$

Nel equazione del punto di riproiezione $\bar{p}_i$ dobbiamo evidenziare la dipendenza dalla rototraslazione tra la flangia e la telecamera all'interno della funzione di riproiezione $h(\cdot)$:

$$\bar{p}_i = h(T^c_w \cdot P^w_i) = h(T^c_f \cdot T^f_w \cdot P^w_i), \quad h(P) = \begin{bmatrix} f_u \frac{P_x}{P_z} + c_u \\ f_v \frac{P_y}{P_z} + c_v \end{bmatrix} \tag{3.12}$$

dove si nota la presenza dell'inverso della rototraslazione cercata, quindi l'algoritmo stimerà T^c_f e poi basterà applicare l'inversione per riportarsi a quella cercata ($T^f_c = (T^c_f)^{-1}$).

Inoltre i punti 3D P^w_i vengono espressi rispetto alla rototraslazione del riferimento della scacchiera T^w_{ref} conoscendo la dimensione reale dei quadrati l:

$$P^w_i = T^w_{ref} \cdot P^{ref}_i \quad \text{con} \quad P^{ref}_i = \begin{bmatrix} a \cdot l \\ b \cdot l \\ 0 \end{bmatrix} \quad \text{e} \quad \begin{cases} a \in [0, ..., N.cols - 1] \\ b \in [0, ..., N.rows - 1] \end{cases} \tag{3.13}$$

Quindi la funzione di perdita S che l'algoritmo deve minimizzare è una sommatoria d'errore nelle foto N a diverse posizioni e negli M punti della scacchiera:

$$S(T^c_f, T^w_{ref}) = \sum_{j=0}^{N} \sum_{i=0}^{M} \left\| h(T^c_f \cdot T^{f,j}_w \cdot T^w_{ref} \cdot P^{ref}_i) - p_{i,j} \right\|^2 \tag{3.14}$$

Come detto prima la posizione della scacchiera non sarà nota con precisione in quanto in molti casi, sopratutto in situazioni di ricalibrazione con molte apparecchiature già montate nei dintorni, non sarà possibile sfruttare il robot per ottenere la posizione della scacchiera ma si dovrà accontentarsi di una stima approssimativa e anche nei casi in cui si può sfruttare il robot si potrebbe avere una stima poco corretta e la conseguente propagazione d'errore nella stima della rototraslazione che ci interessa.

Viene scelto allora di tenere come variabile da stimare anche la posizione della scacchiera T^w_{ref} assieme alla rototraslazione tra flangia e camera T^c_f e concettualmente questo problema con due rototraslazioni da stimare potrebbe avere problemi di degenerazione, infatti tutti i

punti lungo lo stesso raggio hanno la stessa riproiezione ma avendo più foto in posizioni diverse ci sarà solo un possibile punto conforme a tutte le riproiezioni inoltre anche la struttura fissa e nota della scacchiera evita questo tipo di degenerazione.

La minimizzazione della funzione di perdita in (3.14) ha quindi come variabili due matrici di rototraslazione indipendenti, ognuna delle quali può essere parametrizzata con un vettore di stato; scegliendo la parametrizzazione minima il vettore di stato ha 6 elementi: i primi 3 rappresentano la traslazione e gli altri 3 le rotazioni attorno agli assi con configurazione Roll-Pitch-Yaw:

$$x^1 = \begin{bmatrix} t^c_{f,x} \\ t^c_{f,y} \\ t^c_{f,z} \\ R^c_{f,z} \\ R^c_{f,y} \\ R^c_{f,x} \end{bmatrix}, \quad x^2 = \begin{bmatrix} t^w_{ref,x} \\ t^w_{ref,y} \\ t^w_{ref,z} \\ R^w_{ref,z} \\ R^w_{ref,y} \\ R^w_{ref,x} \end{bmatrix}, \quad x = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} \tag{3.15}$$

Inoltre il vettore di stato non appartiene a $\mathbb{R}^N$ ma a allo spazio delle rototraslazioni $SE(3)$ quindi le consuete minimizzazioni, che agiscono in $\mathbb{R}^N$ non porterebbero a buoni risultati.

Si preferisce allora, dopo aver parametrizzato le rototraslazioni, minimizzare all'interno dello spazio $SE(3)$.

Per introdurre la minimizzazione nello spazio delle rototraslazioni ricordiamo come generalmente agiscono le minimizzazioni iterative in $\mathbb{R}^N$ alle quali poi apporteremo delle modifiche. Un qualsiasi metodo mi minimizzazione iterativo con aggiornamento $x_{k+1} = x_k + \varepsilon$ passa attraverso una linearizzazione della funzione di perdita S nella forma:

$$S(x_k + \varepsilon) = S(x_k) + J_k \cdot \varepsilon \qquad \text{con} \qquad J_k = \left.\frac{\partial S(x_k + \varepsilon)}{\partial \varepsilon}\right|_{\varepsilon=0} \tag{3.16}$$

dove tutti i vettori e le operazioni appartengono alla spazio $\mathbb{R}^N$ e dove lo Jacobiano è calcolato ad incremento ε nullo che sta ad indicare nelle vicinanze della stima attuale x_k.

La minimizzazione nello spazio delle rototraslazioni utilizza la stessa linearizzazione ma ha le seguenti modifiche:

- il vettore di stato parametrizzato $x \in se(3)$ è definito nello spazio detto $Lie\ group$ di dimensione 6
- l'incremento attorno alla linearizzazione $\varepsilon \in se(3)$ è anch'esso definito nello spazio $Lie\ group$ di dimensione 6
- l'operatore d'incremento del vettore di stato $+ : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$ deve essere sostituito con l'operatore $\boxplus : se(3) \times se(3) \to se(3)$; quindi l'aggiornamento della del vettore di stato risulta $x_{k+1} = x_k \boxplus \varepsilon$

- l'operatore $\boxplus$ che lega variabili appartenenti ad $se(3)$ può essere però implementato tra variabili $SE(3)$ adottando le giuste funzioni di mappatura tra i due spazi e e ln:

$$x_k \xrightarrow{e} X_k, \qquad X_{k+1} = e^{\varepsilon} X_k, \qquad X_{k+1} \xrightarrow{ln} x_{k+1} \tag{3.17}$$

Riassumendo, la minimizzazione nello spazio delle rototraslazioni stima ad ogni iterazione l'incremento $\varepsilon \in se(3)$ (6 elementi) ricordando però che l'aggiornamento del vettore di stato avviene in $SE(3)$ (16 elementi) $X_{k+1} = e^{\varepsilon} X_k$:

$$S(e^{\varepsilon} X_k) = S(X_k) + J_k \cdot \varepsilon \qquad \text{con} \qquad J_k = \left. \frac{\partial S(e^{\varepsilon} X_k)}{\partial \varepsilon} \right|_{\varepsilon=0} \tag{3.18}$$

dove la mappatura $e : se(3) \to SE(3), \;\; v = [t \quad \omega]^T \to e^v$ è definita da:

$$e^v = \begin{bmatrix} e^{[\omega]_x} & Vt \\ 0 & 1 \end{bmatrix}, \quad V = I_3 + \frac{1 - cos\theta}{\theta^2}[\omega]_x + \frac{\theta - sin\theta}{\theta^3}[\omega]_x^3, \quad \theta = |\omega| \tag{3.19}$$

Da questa introduzione generale si nota che è necessario calcolare lo Jacobiano J_k della funzione S, che per semplicità può essere riscritta come

$$S(\cdot) = \sum_i \|h_i(\cdot) - p_i\|^2 = \sum_i (h_i(\cdot) - p_i)^T (h_i(\cdot) - p_i) \tag{3.20}$$

dove abbiamo riunito gli indici delle diverse posizioni e dei diversi punti in un unico indice i, e dalla quale deriva facilmente che

$$J_k = \left. \frac{\partial S(\cdot)}{\partial \varepsilon} \right|_{\varepsilon=0} = \sum_i 2(h(\cdot) - p_i)^T \left. \frac{\partial h_i(\cdot)}{\partial \varepsilon} \right|_{\varepsilon=0} \tag{3.21}$$

con $\varepsilon = [\varepsilon_1, \varepsilon_2]^T$ vettore di 12 elementi rappresentanti l'incremento dei 2 vettori di stato parametrizzati x^1 e x^2 con ognuno 6 elementi che nel campo $SE(3)$ sono rispettivamente T_f^c e T_{QR}^w.

Il calcolo dello Jacobiano si è ridotto al calcolo della derivata di $\frac{\partial h(\cdot)}{\partial \varepsilon}$ che può essere divisa nel calcolo separato delle derivate di ε_1 e ε_2:

$$\left. \frac{\partial h(e^{\varepsilon_1} \overbrace{T_f^c \cdot T_w^{f,i} \cdot T_{QR}^w}^{A} \cdot \overbrace{P_i^{QR}}^{p})}{\partial \varepsilon_1} \right|_{\varepsilon_1=0} = \left. \frac{\partial h(g)}{\partial g} \right|_{g=Ap} \cdot \left. \frac{\partial e^{\varepsilon_1} Ap}{\partial \varepsilon_1} \right|_{\varepsilon_1=0} \tag{3.22}$$

$$\left. \frac{\partial h(\overbrace{T_f^c \cdot T_w^{f,i}}^{A} \cdot e^{\varepsilon_2} \overbrace{T_{QR}^w}^{D} \cdot \overbrace{P_i^{QR}}^{p})}{\partial \varepsilon_2} \right|_{\varepsilon_2=0} = \left. \frac{\partial h(g)}{\partial g} \right|_{g=ADp} \cdot \left. \frac{\partial A e^{\varepsilon_2} Dp}{\partial \varepsilon_2} \right|_{\varepsilon_2=0} \tag{3.23}$$

Vengono riportate anche le derivate che servono per comporre le derivate sopra elencate:

$$\frac{\partial h(g)}{\partial g} = \begin{bmatrix} f_u/g_z & 0 & -f_u g_x/g_z^2 \\ 0 & f_v/g_z & -f_v g_y/g_z^2 \end{bmatrix}, \qquad \left.\frac{\partial e^{\varepsilon} Ap}{\partial \varepsilon}\right|_{\varepsilon=0} = \begin{bmatrix} I_3 & -[Ap]_x \end{bmatrix} \tag{3.24}$$

$$\left.\frac{\partial A e^{\varepsilon} Dp}{\partial \varepsilon}\right|_{\varepsilon=0} \approx \begin{bmatrix} I_3 & -[p+d_t]_x \end{bmatrix} \quad \text{con} \quad d_t = traslationOf(D) \tag{3.25}$$

Trovato lo Jacobiano J_k al passo k possiamo calcolare l'incremento ε come quello che, nella linearizzazione trovata , porta la funzione di perdita al passo successivo $k+1$ verso il minimo assoluto che per tale funzione è 0:

$$S_{k+1} = S_k + J_k \cdot \varepsilon = 0 \quad \Longrightarrow \quad \varepsilon = -\frac{1}{\alpha}(J_k^T J_k)^{-1} J_k^T S_k \tag{3.26}$$

dove è stato aggiunto un fattore di scala $1/\alpha$ che riduce la dimensione dell'incremento.

Un altro metodo per stimare l'incremento ε senza calcolare la derivata seconda può essere quello di linearizzare non direttamente la funzione $S(\cdot)$ ma solamente la funzione interna al modulo $f(\cdot)$:

$$S(\cdot) = \sum_i \|f_i(\cdot)\|^2 \quad \text{con} \quad f_i(\cdot) = h_i(\cdot) - p_i \tag{3.27}$$

La linearizzazione di tale funzione passa attraverso il calcolo dello Jacobiano J_k^f che corrisponde alla derivata prima della funzione $h(\cdot)$ già calcolata: $J_k^f = \frac{\partial f(\cdot)}{\partial \varepsilon}\big|_{\varepsilon=0} = \frac{\partial h(\cdot)}{\partial \varepsilon}\big|_{\varepsilon=0}$.

La linearizzazione della funzione è allora $f_i(x_{k+1}) = f_i(x_k) + J_k^f \cdot \varepsilon$ e otteniamo la funzione S nella forma:

$$S_{k+1} = \sum_i (f_i(x_k) + J_k^{f,i} \cdot \varepsilon)^T (f_i(x_k) + J_k^{f,i} \cdot \varepsilon) = \varepsilon^T H_k \varepsilon + 2 B_k \varepsilon + C_k \tag{3.28}$$

Data questa nuova linearizzazione della funzione di perdita si può stimare l'incremento ε con approccio Gauss-Newthon che soddisfa la condizione di derivata della funzione linearizzata uguale a 0:

$$\frac{\partial S_{k+1}}{\partial \varepsilon} = 2 H_k \varepsilon + 2 B^T = 0 \quad \Longrightarrow \quad \varepsilon = -\frac{1}{\alpha}(H_k)^{-1} B^T \tag{3.29}$$

La fase di aggiornamento delle variabili da stimare secondo l'aggiornamento scelto $\varepsilon = [\varepsilon_1 \; \varepsilon_2]$ è eseguita con la seguente formulazione:

$$T_f^c \longleftarrow e^{\varepsilon_1} T_f^c, \qquad T_{ref}^w \longleftarrow e^{\varepsilon_2} T_{ref}^w \tag{3.30}$$

L'algoritmo quindi partendo dalla stima approssimativa calcolata in (3.10) e per ogni iterazione calcolala l'incremento con i due metodi esposti e valutata preventivamente la funzione S derivante dall'aggiornamento della variabili da stimare con i due incrementi congiuntamente ad una serie di fattori di scala e viene scelto l'incremento ε che ottiene funzione di perdita inferiore. Per la maggior parte delle iterazioni l'incremento scelto appartiene ad una scala dell'incremento calcolato con il primo metodo in quanto esso riesce sempre ad andare verso il minimo assoluto mentre il secondo spesso stima soltanto un minimo locale ed è utile soltanto nelle ultime iterazioni quando siamo già vicini al minimo cercato in quanto riesce ad approssimare con maggior precisione la funzione ed a raffinare il minimo.

Scelto l'incremento migliore le variabili da stimare vengono aggiornate e l'algoritmo continua iterativamente per un numero finito di iterazioni.

L'algoritmo può essere sviluppato utilizzando tutti i vertici della scacchiera oppure come fatto in figura (3.2) usando soltanto i 4 punti agli angoli della scacchiera; questo permette all'algoritmo di essere più veloce in quanto devono essere processati molti meno punti senza perdere qualità nel risultato perché le variabili da stimare applicano solo trasformazioni rigide che non richiedono la presenza di una moltitudine di punti per essere stimate.

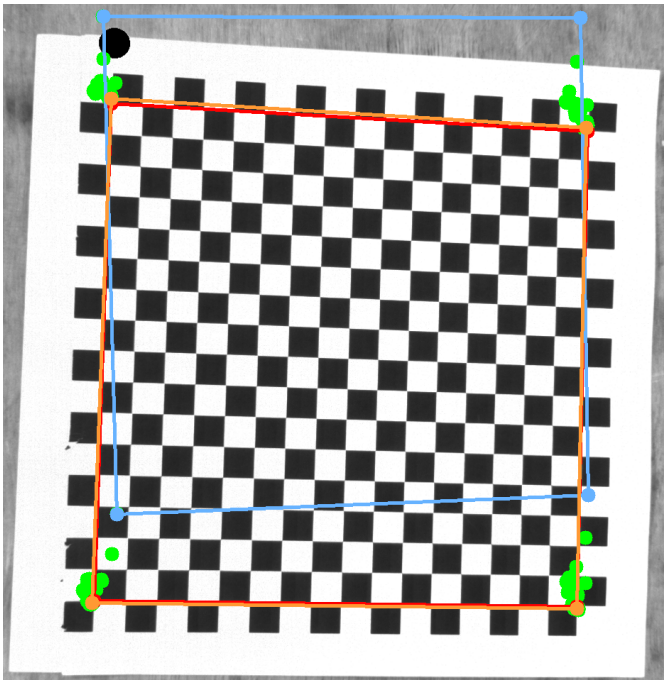


Figura 3.2: Iterazioni dell'algoritmo per calibrazione dei parametri estrinseci.
Sono visualizzati i punti individuati nell'immagine $p_{i,k}$ (—) e i punti di riproiezione $\bar{p}_{i,k}$ alla prima iterazione dell'algoritmo (—), durante l'algoritmo (—) e alla fine (—)

4

Algoritmi per la corrispondenza stereoscopica

La visione stereoscopica è una tecnica atta a ricostruire la tridimensionalità di una determinata area da due fotogrammi catturati in posizioni diverse. L'informazione sulla tridimensionalità di un punto è data dal riconoscere tale punto nelle due immagini, quindi alla base della visione stereoscopica ci sono gli algoritmi che calcolano le corrispondenze tra le due immagini.

Questi algoritmi si possono dividere in due categorie: gli algoritmi **locali** e **globali**. I primi calcolano le corrispondenze analizzando solo le vicinanze dei due punti presi in esame, sono molto veloci ma non restituiscono risultati soddisfacenti. I secondi invece scelgono i punti corrispondenti considerando tutta l'immagine, in particolare minimizzando una funzione di costo che vincola i punti ad essere ordinati e ad avere una certa levigatezza; questi sono algoritmi con costo computazionale elevato ma che ottengono ottimi risultati.

Accanto a quest'ultimi ci sono gli algoritmi **semi-globali** la cui funzione di costo deve essere minimizzata solo in alcune porzioni di immagine accelerando di molto i tempi computazionali e ottenendo risultati che si avvicinano a quelli degli algoritmi globali.

4.1 Semi-Globali Match

Alla sezione degli algoritmi semi globali fa parte l'algoritmo che verrà illustrato e utilizzato, l'algoritmo in questione è chiamato **Semi-Globali Match (GSM)** [bib] di Helsinki Citrulleria.

Innanzitutto gli algoritmi per la corrispondenza stereo hanno come ingresso 2 immagini che saranno nominate $Left$ e $Right$ ed essi hanno il compito di individuare le corrispondenze dall'immagine base $Left$ all'immagine target $Right$; ovvero preso un pixel p dell'immagine $Left$ cercano di trovare la corrispondenza migliore nell'immagine $Right$ all'interno

della linea epipolare che per immagini rettificate si trova in $q = [p_x - d,\ p_y]^T$ dove d rappresenta la disparità ed l'unico parametro da stimare. Il risultato dell'algoritmo sarà quindi una mappa di disparità riportante per ogni cella il valore d relativo al pixel p dell'immagine $Left$.

Nella ricerca delle corrispondenze tra singoli pixel un aspetto importante è l'area che viene confrontata: in generale la qualità dell'accoppiamento è migliore aumentando l'area, questo però porta a calcolare disparità uguale in tutta l'area perdendo definizione nella mappa di disparità.

Questo algoritmo allora confronta singoli pixel, ciò significa che l'unica informazione che può essere usata è l'intensità dei pixel. Metriche che adottano questo tipo di confronto sono Mutua Informati on [bib] o quella proposta da Burchiello and Omasi [bib] nella quale il costo di ogni singolo confronto è la distanza assoluta fra le intensità:

$$C_{BT}(p,d) = |I_p - I_q| \quad \text{con} \quad q = \begin{bmatrix} p_x - d \\ p_y \end{bmatrix} \tag{4.1}$$

quindi avremmo un costo basso se la corrispondenza è buona.

Il calcolo delle corrispondenze confrontando i singoli pixel in generale è ambiguo e calcola molte false corrispondenze. La scelta classica è adottare una funzione di costo, detta energia $E(D)$, che penalizza i cambiamenti di disparità tra punti vicini N_p e che favorisce la levigatezza. Questa funzione che deve essere minimizzata su tutta la mappa di disparità D è riportata di seguito:

$$E(D) = \sum_p \left\{ C_BT(p, D_p) + \sum_{q \in N_p} P_1 \cdot T[|D_p - D_q| = 1] + \sum_{q \in N_p} P_2 \cdot T[|D_p - D_q| > 1] \right\}$$

La funzione quindi, a parità di costo, predilige che pixel vicini abbiano la stessa disparità mentre penalizza con P_1 dove la disparità cambia di un pixels e con P_2 dove cambia di molto, e data l'imposizione $P_1 <= P_2$ sta preferendo superfici curve a salti netti di disparità ($d \in \infty, .., -1, 0, 1, .., \infty$).

Piccoli e grandi cambiamenti di disparità possono avvenire solo se effettivamente la corrispondenza pixel a pixel ha costo C_{BT} talmente basso da essere inferiore alle altre corrispondenze nonostante l'incremento rispettivamente di P_1 e P_2.

La minimizzazione di questa energia nello spazio delle immagini 2D è computazionalmente onerosa e impiega molto tempo, l'algoritmo GSM propone una minimizzazione 1D che ottiene risultati molto simili alla minimizzazione 2D ma che può essere risolta in modo efficiente in tempo polinomiale attraverso la programmazione dinamica.
L'approccio proposto minimizza lungo dei percorsi 1D nell'immagine e in particolare propone un nuovo metodo di aggregazione dei costi nel quale ogni percorso ha gli stessi vincoli risolvendo il problema di venatura lungo le linee epipolari che si verificava per la presenza di forti vincoli lungo le suddette linee e vincoli più deboli negli altri percorsi.

L'aggregazione dei costi $S(p, d)$ per il pixel p e la disparità d, che rappresenta sempre e solo la disparità lungo la linea epipolare, è data dalla somma di tutti i costi di tutti i percorsi 1D

minimi che terminano in p alla disparità d. Questi percorsi sono linee rette nell'immagine di base ($Left$) ma non linee rette nell'immagine in cui cercare le corrispondenze ($Right$), inoltre è interessante notare che è richiesto solo il costo del percorso non il percorso stesso.

Il costo del percorso $L_r(p,d)$ lungo la direzione $r \in \{\pm 1,\ \pm 1\}$ del pixel p alla disparità d è definito ricorsivamente da:

$$L_r(p,d) = C_{BT}(p,d) + min \begin{Bmatrix} L_r(p-r,d) \\ L_r(p-r,d-1) + P_1 \\ L_r(p-r,d+1) + P_1 \\ \min\limits_i L_r(p-r,i) + P_2 \end{Bmatrix} - \min\limits_k L_r(p-r,k) \quad (4.2)$$

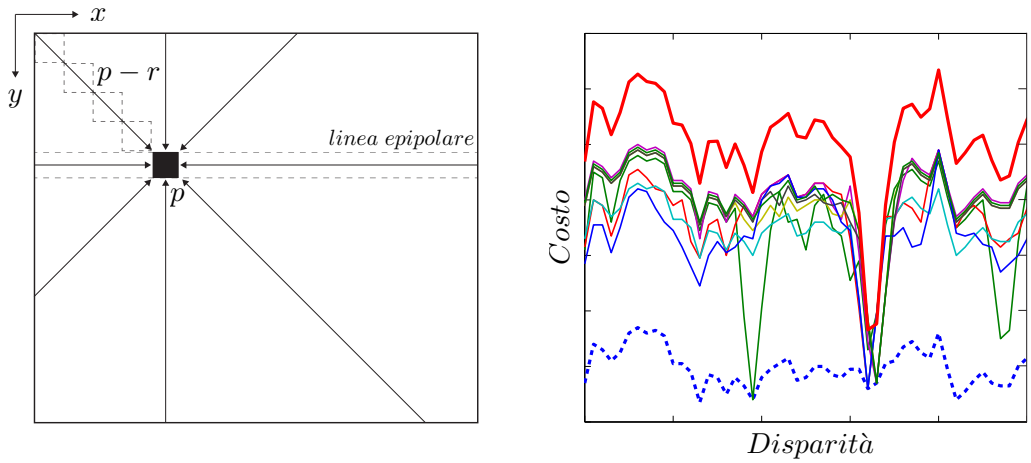
Il primo termine, come descritto prima, rappresenta il costo della singola corrispondenza pixel a pixel. Il secondo prende il costo minimo del precedente pixel $p - r$ nel percorso includendo l'appropriata penalità e che porta al suo interno tutte le informazioni del percorso fino a quel pixel. Infine il terzo termine serve per non far crescere all'infinito il costo lungo il percorso senza modificarne il minimo finale in quanto per tutte le disparità d del pixel p è un termine costante.Il numero di percorsi dovrebbero essere 16 o 8 per avere una buona convergenza ai risultati ottenuti con minimizzazione globale.

La aggregazione dei costi $S(p,d)$ viene calcolata con:

$$S(p,d) = \sum_r L_r(p,d) \quad (4.3)$$

e per ogni pixel p viene individuata la disparità d nella quale il costo aggregato è minore.

L'effetto dell'aggregazione dei costi può essere visualizzato in figura (4.1), dove si nota che i costi dei singoli percorsi a diversa disparità hanno minimi incerto mentre la somma di essi ha minimo robusto.



(a) Percorsi terminanti nel pixel p nelle 8 direzioni

(b) Costo dei singoli percorsi in tratto fino mentre costo C_{BT} in (- -) e costo aggregato S in (—)

Figura 4.1: Analisi degli 8 percorsi per un pixel p e per le disparità $d \in \{d_{min}, .., d_{max}\}$

Le disparità così calcolate appartengono ai numeri interi positivi, in aggiunta per avere maggiore precisione, può essere fatta una stima sub-pixel interpolando una curva quadratica nelle disparità vicine e individuando il minimo di tale curva. Infine posso essere implementate alcune elaborazioni postume come controllo di robustezza dei minimi, filtraggio di picchi invalidi e interpolazione di occlusioni o disallineamenti.

In questo algoritmo di fondamentale importanza è la scelta dei parametri P_1 e P_2 i quali permettono di adattarsi più o meno velocemente a salti di disparità e permettono di individuare la disparità in zone dove la corrispondenza tra singoli pixel non è robusta. Soffre comunque in zone molto grandi di no-testure o in zone dove l'immagini è saturata per forte riflessione della luce e in questi casi è consigliato l'utilizzo di proiettori arandomici che creano nuove fresatura in zone monotone e dove si hanno riflessioni.

Di seguito viene riportata un esempio di mappa di disparità ottenuta.

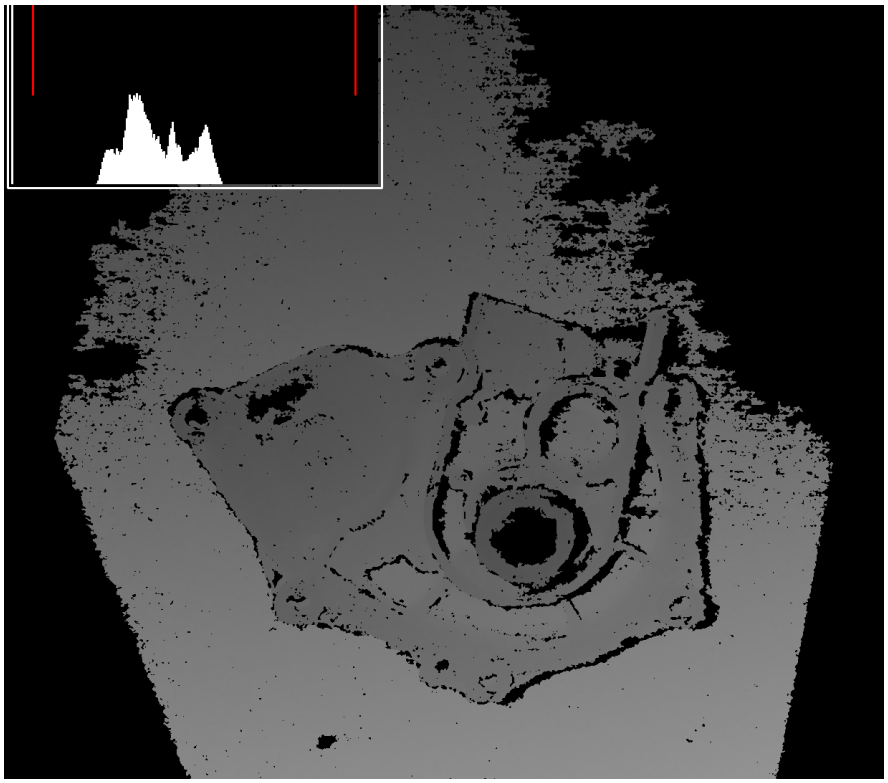


Figura 4.2: Esempio di mappa di disparità con istogramma dei valori di disparità rispetto al range di ricerca

5
Ricostruzione 3D

Questa sezione analizza le metodologie applicate per ricostruire la scena 3D partendo dalla mappa di disparità calcolata dall'algoritmo per le corrispondenze stereo.

La mappa di disparità è una matrice dove ogni pixel contiene un valore rappresentante la disparità $d = x_L - x_R$, oppure con un'altra interpretazione preso un pixel della mappa di disparità, che corrisponde al pixel dell'immagine $Left$ rettificata alle stese coordinate, indica che il pixel corrispondente nell'immagine $Right$ rettificata si trova alla coordinata $x_R = x_L - d$ e alla stessa coordinata y.

Inoltre conoscendo le omografie di rettifica delle due immagini H_L e H_R è possibile trasformare le coordinate, di un dato pixel, da quelle dell'immagine di partenza p a quelle dell'immagine rettificata $\bar{p}$ e viceversa sfruttando l'inversa delle omografie.

Quindi avendo a disposizione queste tre matrici è possibile abbinare pixel corrispondenti tra le immagini di partenza $Left$ e $Right$. Sono riportati di seguito i passaggi per, ad esempio, partendo da un pixel dell'immagine di partenza $Left$ p_L abbinare il corrispondente pixel dell'immagine di partenza $Right$ p_R:

$$p_L \longrightarrow \bar{p}_L = H_L \cdot p_L \longrightarrow \bar{p}_R = \begin{bmatrix} \bar{p}_{L,x} - d(\bar{p}_L) \\ \bar{p}_{L,y} \\ 1 \end{bmatrix} \longrightarrow p_R = (H_R)^{-1} \cdot \bar{p}_R \tag{5.1}$$

Invece, quando è necessario, partendo da un punto $Right$ abbinarci il corrispondente punto $Left$ i passaggi sopra riportati devono essere modificati. Oltre alla banale modifica di sostituzione da L a R e viceversa, differiscono anche nel calcolo della coordinata x. Infatti nel primo caso viene fatta accedendo semplicemente al pixel corrispondente nella mappa di disparità la quale riporta al suo interno l'informazione di corrispondenza per un pixel $Left$ ma non per un pixel $Right$.

Per un pixel $Right$ $\bar{p}_R$ è necessario cercare orizzontalmente alla coordinata $y = \bar{p}_{R,y}$ quale pixel p della mappa di disparità soddisfa $\bar{p}_{R,x} = \tilde{p}_x - d(p)$ e tale pixel è il pixel $Left$ nell'immagine rettificata ($\bar{p}_L = \tilde{p}$).

Questa operazione svolta per tutti i pixel $Right$ è onerosa in termini di tempo, una soluzione alternativa è quella di creare, appena trovata la mappa di disparità originale, un'altra mappa di disparità per accesso da pixel $Right$ in modo che il pixel $Left$ rettificato sia trovato semplicemente come: $\bar{p}_L = [\bar{p}_{R,x} - \hat{d}(\bar{p}_R),\ \ \bar{p}_{R,y},\ \ 1]^T$.

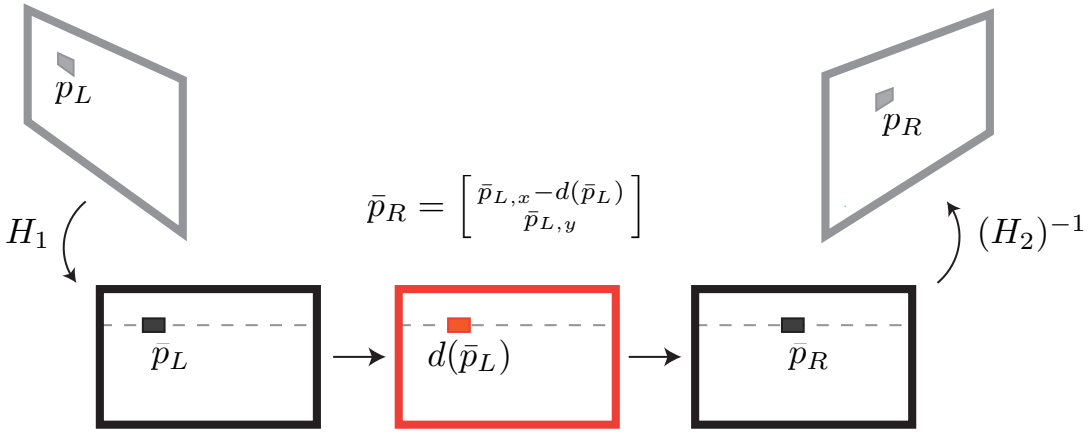


Figura 5.1: Configurazione per ottenere le corrispondenze tra pixel delle immagini di partenza

5.1 Metodi di triangolazione

I metodi di triangolazione sono gli algoritmi attraverso i quali dalle informazioni sulle corrispondenze tra immagini 2D vengono ricostruiti i punti nello spazio 3D.
Dalla lunga analisi precedente si può capire che come metodo di triangolazione non sarà utilizzato quello classico della stereoscopia, il quale partendo semplicemente dalla mappa di disparità calcola la profondità nel piano epiopolare con $z = \frac{B \cdot f}{d}$.

Questa scelta è motivata da due aspetti: il primo riguarda il fatto che questo metodo assume di avere due telecamere con asse ottico parallelo, mentre nella nostra configurazione le telecamere possono avere una qualsiasi rototraslazione e se si cerca di approssimare le omografie di rettifica, che appunto portano le linee epipolari orizzontali cioè asse ottico parallelo, con la compinazione di una rototraslazione e della matrice proiettiva per conferire alle immagini rettificate una posizione fisica si ha sicuramente una perdita di precisione. In secondo luogo non è scelto questo metodo perché non è estendibile a più di 2 viste che, come vedremo a fondo capitolo, è un requisito fondamentale.

Il metodo scelto, invece, usa come informazioni l'abbinamento tra punti nelle immagini di partenza p_L e p_R e cerca il punto 3D che meglio rispecchia queste informazioni con due algoritmi: il primo è un algoritmo con soluzione algebrica lineare mentre il secondo usa un metodo iterativo per ottenere una stima migliore. Il primo algoritmo è molto veloce mentre

il secondo incrementa di molto il tempo di computazione e in pratica sarà utilizzato solamente il primo a parte rari casi in cui è richiesta alta precisione.

Di seguito viene descritto l'**algoritmo lineare** che risolve in modo semplice il problema dell'intersezione tra due o più raggi nello spazio 3D.
Dato un punto immagine $p = [x, y, 1]^T$ e il relativo punto 3D P, ovvero $p = M \cdot P$ con M la matrice di trasformazione proiettiva, vale il principio di parallelismo: $p \times MP = \bar{0}$ in quanto considerando i due fattori del prodotto scalare come matrice omogenea con costante moltiplicativa libera di variare, essi rappresentano lo stesso raggio e quindi sono anche paralleli. Da questo possiamo ricavare i seguenti vicoli:

$$\begin{cases} x(M_3 \cdot P) - (M_1 \cdot P) = 0 \\ y(M_3 \cdot P) - (M_2 \cdot P) = 0 \\ \cancel{x(M_2 \cdot P) - y(M_1 \cdot P) = 0} \end{cases} \tag{5.2}$$

dove M_i rappresenta la riga i-esima della matrice M e dove l'ultimo vincolo può essere eliminato in quanto se vengono sostituiti i primi due in esso, l'equazione introdotta da tale vincolo è soddisfatto per qualsiasi P.

Allora noti i due punti corrispondenti nelle due immagini $p_L = M^L \cdot P$ e $p_R = M^R \cdot P$ possiamo cercare il punto P che soddisfa i vincoli di parallelismo per entrambi i punti:

$$A = \begin{bmatrix} x_L M_3^L - M_1^L \\ y_L M_3^L - M_2^L \\ x_R M_3^R - M_1^R \\ y_R M_3^R - M_2^R \end{bmatrix}, \qquad A \cdot P = 0 \tag{5.3}$$

Il punto P che soddisfa l'equazione è il vettore che appartiene al nucleo di della matrice A e ogni sua versione scalata $P = \alpha \cdot ker(A)$.

Potrebbe succedere che la matrice abbia nucleo vuoto, per risolvere l'equazione in modo più robusto può essere usato SVD, cercando P che minimizza $\|A \cdot P\|^2$ e la soluzione è l'autovettore corrispondente all'autovalore più piccolo della decomposizione di $A = U \cdot W \cdot V^T$, che nella decomposizione classica con autovalori in ordine decrescente corrisponde all'ultima colonna della matrice V. Ricordiamo che SVD minimizza nel vincolo $\|P\| = 1$ quindi il vettore omogeneo calcolato deve essere normalizzato in modo da avere l'ultima componente uguale a 1.
Si nota che questo metodo è estendibile alla triangolazione con un generico numero di punti immagine aggiungendo le relative 2 righe per ogni punto alla matrice A la quale avrà infine dimensione $2N \times 4$ dove N è il numero di punti immagine.

Questo metodo potrebbe non trovare il punto 3D che interpola al meglio le informazioni ricevute. Il metodo **non lineare** invece cerca iterativamente il punto P che minimizza

l'errore di riproiezione nelle immagini:

$$\hat{P} = \underset{P}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| M^i P - p_i \right\|^2 \tag{5.4}$$

La funzione non può essere minimizzata con il metodo lineare least-square in quanto le riproiezioni richiedono la divisione per l'ultima componente. Possiamo allora minimizzare la funzione di perdita con l'algoritmo Gauss-Newton, il quale iterativamente approssima la funzione d'errore $e = M\hat{P} - p$, in un intorno della stima corrente, con:

$$e(\hat{P} + \delta_P) \approx e(\hat{P}) + \frac{\partial e}{\partial P}\delta_P \tag{5.5}$$

Il problema allora si traduce nella ricerca del vettore di aggiornamento δ_P che, in questa iterazione, minimizza la funzione di perdita riporta in (5.6). Questo vettore può essere calcolato in least-square-sense con la formulazione sotto riportata.

$$\hat{\delta}_P = \underset{\delta_P}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| \frac{\partial e}{\partial P}\delta_P + e(\hat{P}) \right\|^2 \quad \Longrightarrow \quad \hat{\delta}_P = -(J^T J)^{-1} J^T e \tag{5.6}$$

$$e = \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} p_1 - M_1\hat{P} \\ \vdots \\ p_N - M_N\hat{P} \end{bmatrix}, \qquad J = \frac{\partial e}{\partial P} = \begin{bmatrix} \frac{\partial e_1}{\partial P_1} & \frac{\partial e_1}{\partial P_2} & \frac{\partial e_1}{\partial P_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial e_N}{\partial P_1} & \frac{\partial e_N}{\partial P_2} & \frac{\partial e_N}{\partial P_3} \end{bmatrix} \tag{5.7}$$

L'algoritmo quindi partendo da una stima approssimativa che può essere quella calcolata con il metodo lineare e per ogni iterazione esegue i seguenti passaggi: linearizza la funzione di perdita attorno alla stima corrente, calcola il vettore d'aggiornamento che secondo la funzione linearizzata porta alla minimizzazione dell'errore di riproiezione rispetto a tutti i punti e aggiorna la stima $\hat{P}^{i+1} = \hat{P}^i + \delta_P$. L'algoritmo riprende gli stessi passaggi dalla stima aggiornata e continua fino alla convergenza o per un certo numero di iterazioni.

5.2 Triangolazione per multi viste

Le richieste progettuali evidenziate ad inizio documento richiedono la ricostruzione della nuvola di punti a partire da un numero generico di viste. Per soddisfare questa richiesta è stata sviluppata una struttura a grafo dove ogni nodo rappresenta un immagine acquisita e dove ogni arco rappresenta l'informazione di corrispondenza tra i due nodi a cui è collegato. Questa informazione di corrispondenza viene data dai tre elementi: immagine di disparità, matrice di rettifica per immagine destra e matrice di rettifica per immagine sinistra come evidenziato ad inizio sezione e nella figura (5.1).

Per scelta progettuale il numero di nodi, ovvero di immagini acquisite, e con esso anche quali archi devono essere presenti, ovvero tra quali immagini si è cercato la corrispondenza,

è un parametro scelto dall'utente che configura il sistema. Questa scelta è dovuta al fatto che immagini prese in posizioni diverse potrebbero inquadrare aree diverse e quindi la ricerca delle corrispondenze essere superflua.

In questo paragrafo verrà analizzato come ricostruire la nuvola di punti avendo le informazioni contenute nel grafo.

Come detto il grafo possiede diversi archi e quindi più di una matrice delle corrispondenze, queste informazioni devono essere unite in modo da creare un unica nuvola di punti.

Un primo possibile metodo potrebbe essere quello di generare una nuvola di punti per ogni matrice delle corrispondenze e poi nello spazio 3D unirle sovrapponendo le parti comuni. Questo metodo avrebbe costi computazionali onerosi per l'unione nello spazio 3D e inoltre non sfrutterebbe tutte le informazioni 2D a disposizione.

Il metodo impiegato invece esplora il grafo e triangola ogni singolo punto 3D sfruttando tutte immagini in cui è stato individuato, infatti è chiaro che una triangolazione a più di due immagini ha errore inferiore rispetto ad una triangolazione a 2 immagini.

Vengono riportati i passaggi che l'algoritmo esegue.

Scelto un nodo di partenza e per ogni pixel della corrispettiva immagine individua le corrispondenze nei nodi collegati a quello di partenza. Questo viene fatto attraversando l'arco in questione ovvero sfruttando le informazioni date dall'immagine di disparità e dalla due matrici di rettifica relative a tale arco come spiegato ad inizio sezione. Ricorsivamente dalle nuove corrispondenze trovate si esplorano allo stesso modo tutti gli archi collegati fino a che la catena non si interrompe. A questo punto con le corrispondenze trovate si triangola con il metodo lineare esposto precedentemente o eventualmente poi con quello non lineare.

Esauriti tutti i pixels del nodo di partenza vengono analizzati gli altri nodi per vedere se ci sono pixels non processati dalle precedenti esplorazioni che verranno gestiti allo stesso modo appena elencato oppure se ci sono catene di nodi completamente separate da quella di partenza che quindi dovranno essere processate.

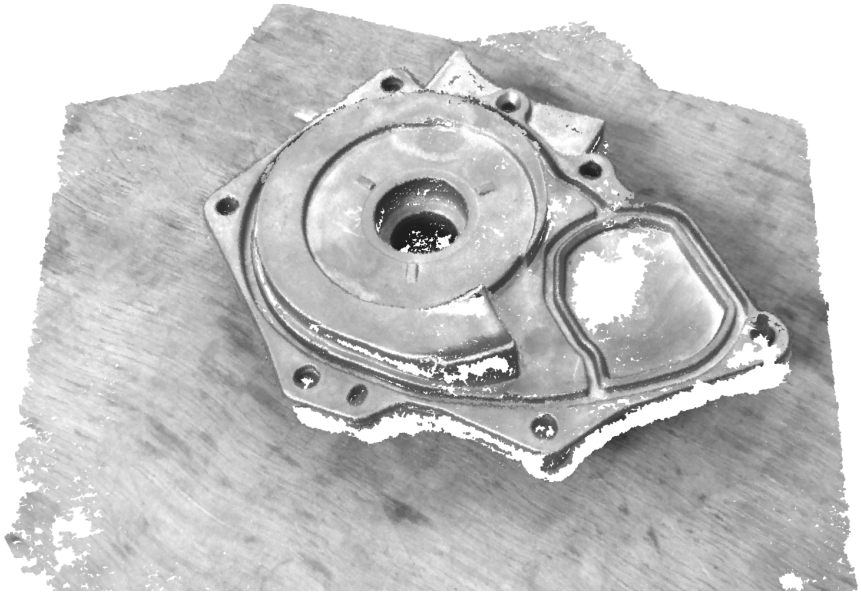


Figura 5.2: Esempio di Point Cloud ottenuta dove ai singoli punti 3D è associato il colore corrispondente

6

Computazione automatica di nuove viste

In questa sezione sono analizzati dei metodi che permettono il calcolo di nuove posizioni della telecamera dove acquisire nuove immagini con l'obiettivo di ottimizzare la nuvola di punti finale.

Innanzitutto è evidenziato il fatto che da una singola immagine 2D, senza alcuna informazione a priori, non è possibile computare una nuova vista che cerchi di inquadrare un area comune da un altra posizione; questo perché la trasformazione proiettiva che mappa punti 3D in immagini 2D non ci da alcuna informazione sulla profondità a cui si trova area inquadrata.

L'applicazione industriale, a cui le richieste progettuali fanno parte, presuppongono di conoscere il modello 3D dell'oggetto quindi esistono tecniche che partendo da un immagine 2D o da una nuvola di punti iniziale permettono di identificare una stima della rototraslazione dell'oggetto nel mondo.

I due algoritmi di questa sezione calcolano le nuove rototraslazioni della telecamera utilizzando questa stima e il modello 3D dell'oggetto ma sono generici per qualsiasi tipo di oggetto e hanno rispettivamente i seguenti obiettivi: il primo è studiato per migliorare la triangolazione di ogni singolo punto mentre il secondo punta a coprire aree dell'oggetto non inquadrate.

6.1 Algoritmo per migliorare la triangolazione

Questo algoritmo vuole ottenere una nuova posizione camera che inquadri al meglio l'oggetto e permetta una buona triangolazione in relazione alle precedenti immagini acquisite in diverse posizioni della telecamera ma note.

Avendo a disposizione il modello 3D dell'oggetto si può ricavarsi una nuvola di punti dell'oggetto dove per ogni punto è presente anche l'informazione del vettore normale in tale

punto. Supposto poi di avere una buona stima della posizione dell'oggetto è possibile traslare la nuvola di punti in tale posizione ed ottenere solo i punti visti da tutte le precedenti camere.

Vengono riportati sinteticamente i passaggi per ottenere questi punti: il modello 3D viene campionato e per ogni punto viene trovata la normale individuata interpolando un piano nei punti vicini, la nuvola di punti con le relative normali è traslata nella posizione dell'oggetto stimata e successivamente è riportata nel riferimento della prima camera in questione e sono calcolati i punti visti dalla telecamera.

Questi punti sono calcolati come quelli appartenenti alla superficie inferiore dell'oggetto nel riferimento della telecamera (nel riferimento del mondo corrispondono alla superficie superiore) che è calcolata in modo rapito inserendo tutti i punti in una griglia 3D ed accedendovi solo verticalmente partendo dal basso e prendendo la prima occorrenza per ogni colonna. In questa operazione abbiamo assunto che tutti i raggi di proiezione sono paralleli che, per distanze telecamera-oggetto mediamente elevate, è una buona approssimazione. Questi punti possono essere ulteriormente filtrati eliminando i punti in cui l'angolo tra la normale e il raggio che punta a tale punto, ovvero $r = C - p$ dove C è la posizione della camera, è superiore a $90°$ quindi non è assolutamente visibile nell'immagine. Queste operazioni vengo ripetute nelle altre camere trovando l'insieme di punti visibili da tutte le precedenti immagini.

Il primo obiettivo di questo algoritmo, ovvero quello di inquadrare al meglio l'oggetto, può essere formulato matematicamente cercando la posizione nella quale tutti i raggi di riproiezione di ognuno di questi punti nella camera hanno angolo rispetto alla normale di tale punto più simile possibile all'angolo $0°$ in quanto un punto è visualizzato nel modo migliore se la telecamera è posta sopra alla sua superficie verticalmente.

Il secondo obbiettivo invece è avere una buona triangolazione per ogni punto, ricordando che la triangolazione viene effettuata trovando l'intersezione tra raggi di diverse immagini, definiamo buona triangolazione quando i raggi sono il meno paralleli possibile perché più paralleli sono più aumenta l'errore nel calcolo del punto di intersezione.

Il raggio di riproiezione di un punto 3D nel piano immagine può essere interpretato come il vettore differenza tra tale punto e il centro focale della telecamera, che identifica appunto la posizione della telecamera, ($r = C - p$). Da questa interpretazione i raggi non dipendono in alcun modo dalla rotazione della telecamera che quindi sarà analizzata solo a fine algoritmo. Inoltre per definire l'angolo tra 2 vettori nello spazio 3D sfruttiamo l'interpretazione geometrica del prodotto scalare:

$$v_1 \cdot v_2 = v_1^T v_2 = |v_1||v_2| cos\phi \quad \Longrightarrow \quad cos\phi = \frac{v_1^T v_2}{\sqrt{v_1^T v_1}\sqrt{v_2^T v_2}} \tag{6.1}$$

dove ϕ rappresenta l'angolo tra i due vettori.
Il problema viene formulato come la ricerca della posizione telecamera X che minimizza la

funzione di perdita:

$$\min_{X \in S} \sum_{i=0}^{N} \left\{ \left[\frac{n_i^T (X - p_i)}{\sqrt{(X - p_i)^T (X - p_i)}} - 1 \right]^2 + \gamma \sum_{k=0}^{M} \left[\frac{\overline{(C_k - p_i)}^T (X - p_i)}{\sqrt{(X - p_i)^T (X - p_i)}} - A^* \right]^2 \right\}$$

La funzione di perdita è composta dal primo addendo che incapsula lo scostamento dai raggi di riproiezione $(X - p_i)$ dal raggio di inquadramento ottimo identificato dal vettore normale n_i, ovvero penalizza quelle posizioni che portano a coppie raggio-normale con coseno dell'angolo diverso da $cos(0°) = 1$.

Il secondo addendo invece penalizza le posizioni che portano a coppie raggio rispetto alla nuova camera X e raggio rispetto alle vecchie camere C_k con coseno dell'angolo diverso dal coseno di un angolo scelto. Questo angolo scelto rappresenta l'angolo desiderato tra la nuova camera e le precedenti camere il quale aumenta proporzionalmente alla qualità della triangolazione ma inversamente alla qualità del corrispondenze 2D.

Il parametro γ è un parametro scelto che permette di dare più importanza o all'inquadratura ottima o alla buona triangolazione.

Infine la funzione viene minimizzata in X appartenente alla sfera S la quale ha centro corrispondente alla stima dell'oggetto data e raggio uguale alla distanza a cui la telecamera è stata messa a fuoco; questo ci permette di evitare le posizioni degeneri in cui la telecamera è molto distante dall'oggetto quindi raggi molto verticali alla superficie ma appunto che non permetterebbero una buona messa a fuoco l'oggetto.

La funzione da minimizzare risulta complessa per l'obiettivo posto quindi cerchiamo delle approssimazioni che ci permettono di semplificarla. Viene assunto allora che tutti i raggi della telecamera siano paralleli a quello principale identificato dall'asse di puntamento z del riferimento della camera. Questo porta ad identificare il fascio di raggi con un unico vettore a norma unitaria x che rappresenta la loro direzione e che ora è l'incognita del problema.

$$\underset{\|x\|=1}{\operatorname{argmin}} \sum_{i=0}^{N} [n_i^T x - 1]^2 + \gamma \frac{N}{M} \sum_{k=0}^{M} [c_k^T x - A^*]^2 \tag{6.2}$$

In questo caso la funzione deve essere minimizzata nel vicolo di vettore unitario, per far questo parametrizziamo il vettore incognito come tutti i punti appartenei alla sfera unitaria con centro l'origine:

$$x = \begin{bmatrix} cos\varphi \cdot cos\theta \\ cos\varphi \cdot sin\theta \\ sin\varphi \end{bmatrix} \tag{6.3}$$

Data questa parametrizzazione è possibile calcolare in modo non complesso la derivata prima e la derivata seconda rispetto ai due parametri φ e θ e minimizzare la funzione con l'algoritmo iterativo Newton-Raphson.

Trovati i due parametri e dopo essersi ricondotti al vettore unitario x si può calcolare la posizione finale della camera come:

$$X = m + \rho \cdot x \tag{6.4}$$

dove m è la stima della posizione dell'oggetto oppure la media di tutti i punti da visualizzare e ρ è la distanza a cui la telecamera è stata messa a fuoco.

Infine la rotazione della camera è trovata come quella rotazione che ha asse z che punta dal centro focale della camera ad m e con rotazione attorno al proprio asse z libera.

Quest'ultimo grado di libertà può essere scelto per allineare la direzione dell'immagine con dimensione maggiore (larghezza di solito) con la massima dispersione dei punti da visualizzare individuata dall'asse maggiore dell'ellisse equivalente trovata con gli 'Hu moments'. Un altro metodo per scegliere questo grado di libertà è individuare la rotazione attorno all'asse z che permette al robot di raggiungere in modo più agevole questa posizione.

6.2 Algoritmo per incrementare la copertura

Il secondo algoritmo invece ha l'obiettivo di calcolare una nuova coppia di posizioni telecamera in grado di visualizzare al meglio una serie di punti scelti e ottenendo anche una buona triangolazione.

Questi punti da inquadrare sono scelti come tutti quei punti del modello 3D che non sono stati individuati da una precedente nuvola di punti. Possono essere calcolati in modo simile a quello descritto in precedenza: traslazione del modello nella stima di posizione, inserimento in una griglia 3D e ricerca dei punti non comuni tra la nuvola di punti iniziale e il modello 3D.

Anche in questo caso il problema viene formalizzato nella ricerca del minimo della funzione di perdita sotto riportata nella quale è stata utilizza la medesima approssimazione dell'algoritmo precedente e dove le incognite sono x_1 e x_2 rappresentati le direzioni principali dei raggi della telecamera 1 e 2.

$$\underset{\|x_{1/2}\|=1}{\operatorname{argmin}} \sum_{i=0}^{N} \Big\{ \overbrace{[n_i^T x_1 - 1]^2}^{①} + \overbrace{[n_i^T x_2 - 1]^2}^{②} + \gamma_1 \cdot 2 \overbrace{[n_i^T x_1 - n_i^T x_2]}^{③} \Big\} + \gamma_2 \cdot 2N \overbrace{[x_1^T x_2 - A^*]^2}^{④}$$

In questa funzione i primi due addendi ① e ② penalizzano posizioni della prima e della seconda telecamera che non inquadrano bene l'oggetto (o meglio i punti da visualizzare), il terzo ③ invece richiede che entrambe le telecamere inquadrino l'oggetto con la stessa inclinazione.

Il quarto elemento ④ invece impone la condizione per avere una buona triangolazione tra le due nuove telecamere.

I parametri γ_1 e γ_2 sono scelti per dar maggiore o minore importanza a vari elementi appena elencati.

La minimizzazione della funzione può essere eseguita nello stesso modo dell'algoritmo precedente con alcune modifiche: parametrizzazione di entrambi i vettori incogniti x_1 e x_2, calcolo della derivata prima e seconda rispetto ai 4 parametri, minimizzazione iterativa con l'algoritmo di Newton-Raphson e infine calcolo delle posizioni e delle rotazioni delle telecamere.

7

Misure e Analisi delle Prestazioni

Vengono ora analizzate le prestazioni degli algoritmi implementati rispetto ad alcuni parametri di configurazione per favorirne poi la scelta ottimale.

Il primo parametro di configurazione che viene analizzato riguarda le posizioni di acquisizione delle immagini. Quando il sistema viene configurato, ad esempio, impostando delle posizioni fisse di acquisizione delle immagini, è importante conoscere gli effetti che possono avere sul risultato finale.

Queste posizioni devono essere scelte innanzitutto in modo da posizionare la telecamera alla distanza di messa a fuoco dalla zona da inquadrare per ottenere delle immagini nitide. Per quanto riguarda la ricostruzione della nuvola di punti invece è bene sapere che la triangolazione con raggi di proiezione molto paralleli genera più errore rispetto ad una triangolazione con raggi che tendono ad essere ortogonali in quanto piccoli errori di ricerca delle corrispondenze o di calibrazione inducono più facilmente l'intersezione tra raggi in posizione errata. La direzione di ogni raggio, in particolare per distanze elevate dall'oggetto, può essere considerata simile alla direzione del raggio principale della fotocamera che parte dal centro focale e passa per il centro immagine che corrisponde alla direzione dell'asse z della matrice di rotazione della telecamera. Quindi l'aumento dell'ortogonalità tra raggi di triangolazioni ha una diretta corrispondenza con la scelta delle rototraslazione delle telecamere.

Viceversa però scegliere rototraslazioni con alta ortogonalità tra gli assi z comporta che la zona comune venga inquadrata da posizioni molto diverse e conseguentemente minori performance dell'algoritmo di ricerca delle corrispondenze. Immagini diverse infatti presentano più pixel, crescenti con l'aumento dell'ortogonalità, che non hanno una corrispondenza nell'immagine opposta portando quindi a mappe di disparità meno dense. Inoltre se è presente una zona vasta che concentra pixel non presenti nell'altra immagine l'algoritmo per le corrispondenze avrà anche difficoltà nella minimizzazione della funzione globale generando mappe di disparità con pochissime corrispondenze. Le corrispondenze sono direttamente

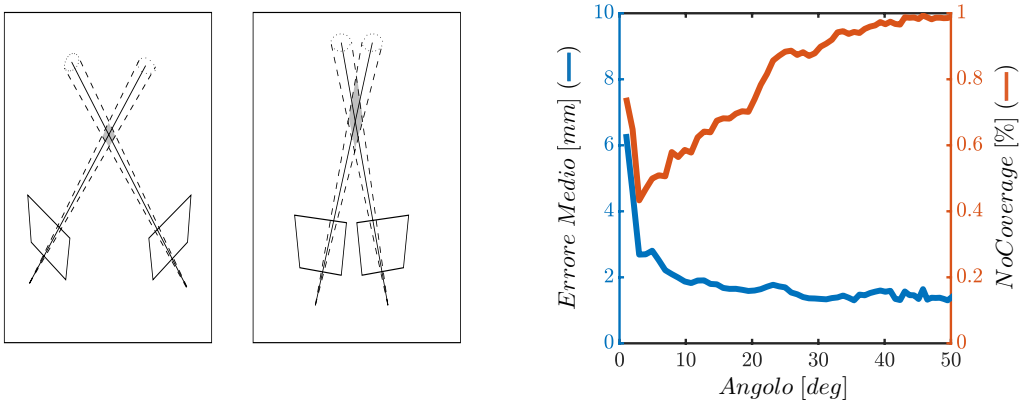
collegate alla generazione della nuvola di punti che quindi all'aumento dell'ortogonalità tra le posizioni avrà una riduzione dei punti triangolati.

Queste considerazioni vengono analizzate nel grafico seguente che mette a confronto l'angolo tra viste con la qualità della nuvola di punti in termini di errore medio in mm e di quantità di area coperta rispetto alla nuvola dei punti reale ottenuta campionando il modello 3D ma prendendo solo i punti visibili dall'alto.

In questa analisi vengono utilizzate solo due viste dove la prima vista è fissata mentre la seconda varia aumentando l'angolo tra le due viste che è calcolato con la formula inversa del prodotto scalare tra le due direzioni degli assi z delle viste. Per ogni angolo è calcolata la nuvola di punti e le relative performance.

Dal grafico in figura (7.1) si nota che aumentando l'angolo l'errore medio diminuisce in quando si ha una migliore triangolazione ma viceversa aumenta la percentuale di area non coperta perché l'algoritmo di ricerca delle corrispondenze ha una mappa di disparità meno densa quindi vengono generati molti meno punti.

Il minimo della somma delle due curve corrisponde all'angolo ottimale il quale però dipende anche dalla forma dell'oggetto che si inquadra; la scelta consigliata è di imporre viste che tra loro hanno $15 - 20$ gradi.



(a) Volume di incertezza per angoli molto grandi o molto piccoli

(b) Qualità della nuvola di punti al variare dell'angolo

Figura 7.1: Analisi dell'errore medio e della copertura rispetto all'angolo di triangolazione tra 2 immagini

Un altro importante parametro di configurazione è il numero di immagini usate per generare la nuvola di punti.

Al crescere del numero di immagini la qualità della nuvola di punti dovrebbe avere miglioramenti in entrambi i parametri perché aumentando le viste che triangolano un singolo punto sicuramente l'errore di tale punto diminuisce, in particolare si nota che migliora la

precisione in profondità del punto, e inoltre con molte viste si riesce a coprire zone occluse aumentando l'area coperta.

Queste affermazioni in un analisi più profonda non sono del tutto vere e vengono analizzate nei tre grafici seguenti in figura (7.2) congiuntamente ad un altro parametro di configurazione introdotto che è il numero minimo di immagini di triangolazioni per generare un punto. L'imposizione di questo parametro afferma che se nella fase di esplorazione del grafo per un singolo punto sono state individuate un numero di immagini maggiore o uguale al parametro imposto esso sarà triangolato ed inserito nella nuvola di punti finale.

Analizzando il primo grafico in cui questo parametro è stato imposto a 2, ovvero il numero minimo di immagini per poter triangolare, notiamo che al crescere del numero di immagini la percentuale dell'area non coperta giustamente diminuisce ma l'errore medio, che in questo grafico è stato normalizzato a $10mm$, ha una leggera flessione ma è pressoché costante; questo si verifica perché molti punti verranno triangolati con più immagini generando punti molto precisi e con errore medio basso mentre altri verranno triangolati ancora con il numero minimo di immagini perché la loro corrispondenza è stata individuata in solo due immagini e avranno errore maggiore.

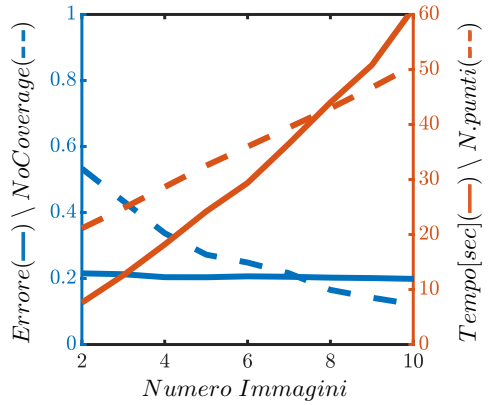
Viceversa nel secondo grafico il numero minimo di immagini di triangolazione è imposto uguale al numero massimo di immagini. In questo caso l'errore diminuisce perché ogni punto della nuvola di punti sarà triangolato con tutte le immagini disponibili quindi ottenendo la massima precisione, ma la percentuale di area non coperta aumenterà in quanto alla nuvola di punti finale apparterranno soltanto i punti per i quali è stata individuata corrispondenza in tutte le immagini e questo peggioramento è confermato dalla diminuzione drastica dei punti generati rispetto anche al grafico precedente che con 10 immagini generava attorno ai 3 milioni di punti mentre in questo sempre con 10 immagini attorno ai 200 mila punti.

In questo grafico si può notare un altro aspetto di imporre il numero di immagini di triangolazione elevato: si nota che il tempo di generazione delle nuvole di punti con questo parametro posto al massimo ha tempo di elaborazione decisamente inferiore rispetto a quella con parametro imposto al minimo. La selezione dei punti che rispettano il numero minimo di immagini di triangolazione può essere fatto postumo la generazione della nuvola di punti ma se viene fatto durante la generazione si ha questo tipo di velocizzazione per due motivi. Il primo è sicuramente la diminuzione della quantità di punti quindi dell'effettivo tempo impiegato nella triangolazione. Il secondo invece riguarda il tempo impiegato nell'esplorazione del grafo, conoscendo il numero minimo di immagini di triangolazione e la parte di grafo già esplorata con analisi preventiva si può evitare di processare alcune mappe di disparità risparmiando notevolmente tempo.

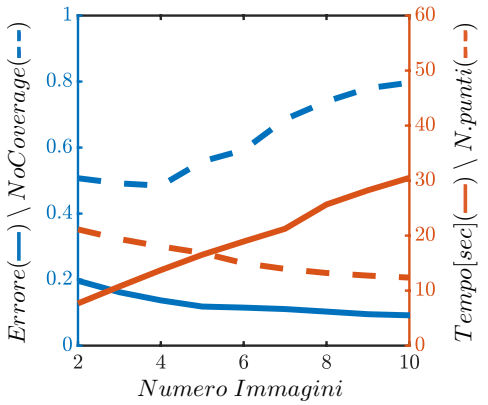
La scelta di imporre il numero di immagini di triangolazione al massimo non è una scelta soddisfacente ma si preferisce ad esempio porre tale parametro uguale alla metà del numero di immagini a disposizione come fatto nel terzo grafico, ottenendo buone prestazioni sia in errore medio che in percentuale di area coperta.

Per quanto riguarda il numero di immagini è consigliato utilizzare dalle 3 alle 5 immagini

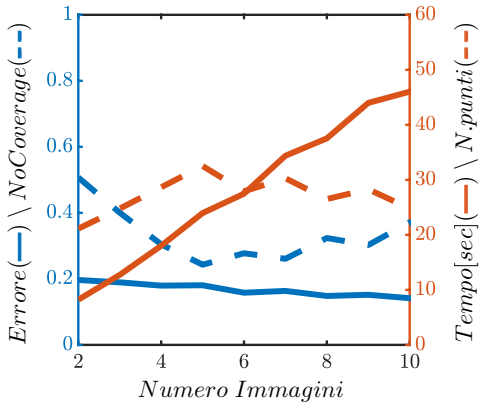
che permettono di ottenere buone nuvole di punti con tempo di elaborazione tra i $10 \div 30$ secondi.



(a) Immagini di triangolazione minime sempre uguale a 2



(b) Immagini di triangolazione uguale al numero di immagini



(c) Immagini di triangolazione minime uguale alla metà del numero di immagini

Figura 7.2: Analisi delle prestazioni rispetto al numero di immagini utilizzate e con diverse scelte del numero minimi di immagini di triangolazione per ogni punto

Infine un ultima configurazione da analizzare riguarda il come collegare le immagini acquisite ovvero tra quali immagini ricercare le corrispondenze. Questa analisi è fatta in figura (7.3) nella quale è stato fissato il numero di immagini a 5 e il numero minimo di immagini di triangolazione alla meta del numero immagini, che dopo la quinta disparità, le immagini totali saranno sempre 5 con quindi numero minimo di immagini di triangolazione 3 ed è stata valutata la qualità della nuvola di punti variando il numero di mappe di disparità generate.

In quest'analisi le prime 4 disparità sono servite per collegare tra loro tutte le immagini mentre le successive portano collegamenti aggiuntivi tra le immagini fino ad arrivare a 10 disparità ovvero tutte le combinazioni non ordinate e senza ripetizioni di 2 oggetti presi in un insieme di 5 oggetti, $\binom{5}{2} = 10$, che permettono di collegare tutte le immagini tra di loro.

Arrivati alle 4 disparità la nuvola di punti generata ha già ottime prestazioni ed è la scelta adottata nei grafici (7.2) ed è anche la scelta consigliata ovvero quella di acquisire immagini in posizioni sequenziali e poi cercare le corrispondenze solo tra immagini vicine senza doversi preoccupare del problema che tra immagini distanti la ricerca delle corrispondenze non avrà mappa di disparità densa e quindi quasi inutile. Questa scelta ha anche tempi computazionali inferiori a nuvole di punti con più mappe di disparità perché il tempo di calcolo totale della nuvola di punti corrisponde a circa $3 \div 4$ secondi per ogni mappa di disparità più il tempo di esplorazione e di triangolazione che dipende dal numero minimo di immagini per triangolare e dalla quantità di punti da generare.

Ad ogni modo l'aggiunta di collegamenti tra i nodi porta informazione aggiuntiva che migliora la nuvola di punti generata sia per quanto riguarda l'errore medio e la copertura come si vede in figura (7.3).

I collegamenti aggiuntivi infatti possono creare corrispondenze tra immagini che con le mappe di disparità precedenti non venivano evidenziati aumentando il numero di immagini di triangolazione per ogni punto con il conseguente miglioramento dell'errore medio e aumento del numero di punti generati migliorando di conseguenza la copertura.

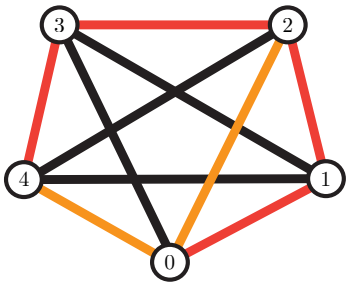
I collegamenti aggiuntivi portano anche delle informazioni che permettono un controllo di consistenza per ogni punto. Infatti aumentando i collegamenti si creano più grafi circolari che per uno stesso punto 3D generano due osservazioni nell'immagine di partenza del grafo cioè il punto di partenza dell'esplorazione e quello di arrivo. Avendo queste due osservazioni di uno stesso punto è possibile controllare se ci sono stati errori nella ricerca delle corrispondenze e non generare tale punto; ovvero se la distanza tra le due osservazioni è inferiore ad 1 pixel si considerano corrette tutte le corrispondenze nelle varie immagini all'interno della catena di esplorazione del grafo, si fa una media tra le due osservazioni per creare un unica osservazione per tale immagine e si procede con la triangolazione se invece la distanza è superiore ad 1 pixel nella catena di esplorazione può esserci una corrispondenza errata e tale catena non verrà triangolata.

Esiste un altro controllo di consistenza che può essere eseguito ma che in quest'analisi non è stato fatto. Si ricorda che nell'approccio di triangolazione spiegato nella relativa sessione una mappa di disparità da immagine 1 ad immagine 2 viene considerata uguale ad una mappa di disparità da immagine 2 a 1 in quanto entrambe legano punti tra le stesse immagini. Avere però entrambe le disparità ci permette di fare un controllo aggiuntivo per migliorare una delle due mappe che poi verrà utilizzata nella triangolazione.

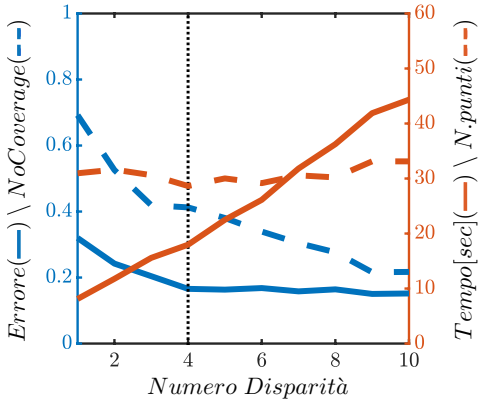
Il controllo è eseguito verificando per ogni punto la mappa di disparità da immagine 1 a immagine 2 e viceversa ottengano la stessa corrispondenza entro un intervallo di 1 pixel altrimenti tale punto è invalidato. Con questo metodo è possibile eliminare le false dispa-

rità generate dall'algoritmo alle corrispondenze in particolare a bordo oggetto ovvero dove si hanno salti netti di disparità nei quali la funzione globale da minimizzare potrebbe non identificare subito il cambio netto di disparità.

Si ricorda che anche in questo caso si ha un aumento del tempo computazionale in quanto la ricerca delle disparità aggiuntiva richiede $3 \div 4$ secondi come ogni disparità più il tempo del controllo di consistenza ma effettivamente non porta molta informazione aggiuntiva in quanto già con solo una disparità avremmo collegato i punti nelle due immagini.



(a) Grafo con disparità consigliate in (—) ed esempi di grafo circolare in (—)



(b) Qualità e tempistiche al variare del numero di mappe di disparità

Figura 7.3: Analisi delle prestazioni con 5 immagini ma variando il numero di mappe di disparità che collegano le immagini.

Concludendo si specifica che questi tempi di cui si è parlato riguardano soltanto la fase di elaborazione dell'immagini e non contano i tempi di spostamento del mezzo robotico ma è da notare che gran parte dell'elaborazione può essere eseguita durante la movimentazione e quindi inserire una buona parte del tempo di calcolo all'interno delle imprescindibili tempistiche di movimentazione.

Inoltre tutti i tempi sono stati registrati su un laptop non particolarmente performante quindi si potrebbero avere tempistiche migliori su computer di livello superiore o scegliendo architetture di sviluppo diverse.

8 Conclusioni

Il sistema sviluppato è un sistema dinamico che può adattarsi ad innumerevoli applicazioni in quanto è completamente configurabile in base alle esigenze.

Gestisce configurazioni con numero di foto qualsiasi acquisite sia in posizioni fisse che in posizioni variabili, dispone di algoritmi per calcolare in modo autonomo le nuove posizioni di acquisizione e può gestire anche sistemi di più telecamere montate fisse su una struttura o una combinazioni di telecamere fisse e telecamere su robot.

La esile struttura fisica del sistema composta solamente da una piccola telecamera 2D con la relativa ottica gli permette di poter essere montato in qualsiasi sistema robotico o di movimentazione. Ha costi ed ingombri nettamente inferiori a sistemi più complessi che richiedono laser o ultrasuoni in quanto la telecamera utilizzata deve avere una buona risoluzione ma non necessità di alti frame rate.

Rispetto a questi sistemi genera nuvole di punti di qualità paragonabile anche se leggermente inferiore in quanto non usa sorgenti di luce esterna che facilitano la ricerca delle corrispondenze. Ha tempi di generazione superiori perché innanzitutto richiede l'attesa dello spostamento del mezzo robotico e poi nell'attuale implementazione ha tempi di elaborazione superiori legati sopratutto all'architettura hardware sul quale gli algoritmi sono stati implementati.

Sempre rispetto a questi sistemi ha però il vantaggio di avere meno problemi di calibrazione nel momento di prelievo del pezzo identificato nella nuvola di punti in quanto i sistemi appena menzionati devono essere posti lontani dalla zona di prelievo per non intralciare il robot che effettuerà il prelievo mentre il sistema sviluppato montato a bordo robot genera nuvole di punti da posizione molto più vicina affievolendo di molto questo problema.

Rispetto a sistemi stereoscopici classici con due telecamere affiancate, invece, ottiene prestazioni nettamente superiori in quanto è possibile aumentare a piacimento la baseline e quindi l'angolo di triangolazione tra ogni singolo raggio di proiezione aumentato così la precisione di ogni singolo punto e sopperendo a piccoli errori di identificazione dell'algoritmo

di ricerca delle corrispondenze. Inoltre ottiene prestazioni superiori anche perché riesce a legare molte immagini in un unica triangolazione a differenza della stereoscopia classica che utilizza solo due immagini per triangolazione.

Inoltre la struttura fisica con telecamera montata su robot permette di coprire zone ceche, di sopperire ad immagini saturate per riflessione sugli oggetti e permette di servire con un unica apparecchiatura più stazioni di prelievo degli oggetti.

Gli oggetti ai quali può essere applicato questo sistema sono oggetti di dimensione non troppo piccola e per i quali viene richiesto un tempo ciclo di qualche decina di secondo.

Gli sviluppi futuri per questo sistema riguardano in particolare l'implementazione in architetture hardware più preformanti come può essere la programmazione con CUDA su scheda video che permette la massiva parallelizzazione riducendo drasticamente i tempi di calcolo. Si vuole anche sviluppare l'intero sistema su una scheda embedded per renderlo un sistema di visione indipendente applicabile anche senza la presenza di un computer.

Elenco delle figure

Elenco delle tabelle