Universita' di Pisa

Ingegneria Robotica

Sistemi di guida e navigazione

Algoritmo di tracking per la stima della posa di una camera monoculare

Studente:
Daniela Resasco

Professore:
Lorenzo Pollini

9 maggio 2016

Abstrac

In questo report verrà presentato l'algoritmo di visione implementato per la stima della posa di una camera monoculare. Queste informazioni verranno utilizzate in seguito per il controllo di un braccio robotico posto su di una carozzina al fine di premere un pulsante scelto dall'utente. Il pulsante viene premuto dall'utente tramite interfaccia tablet. Una volta identificato il pulsante premuto, la camera fornisce al robot le informazioni di posizione ad orientazione dello stesso, in modo da posizionarsi sul percorso corretto. La camera utilizzata è monoculare pertanto non si hanno informazioni sulla distanza. Per ovviare a questa mancanza si è deciso di utilizzare un algoritmo esterno capace di utilizzare le informazioni delle features trovate per stimare la posizione relativa. Questo algoritmo è affetto da un errore di scala e per ridurne gli effetti, si utilizza un criterio di convergenza della stima del fattore di scala utilizzando il metodo della massima verosimiglianza.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nella vita quotidiana ci troviamo spesso a dover premere dei pulsanti, i.e quelli del citofono, i pulsanti dell'ascensore etc. Le pulsantiere, si pensi a quelle degli ascensori, sono poste in modo da essere premute senza dover alzare il braccio più in alto della linea degli occhi. Se pensiamo ai bambini, per premere un pulsante si mettono sulle punte dei piedi. Cosa succede se una persona con una carrozzina non arriva a premere un pulsante?. Questo progetto si occupa di aiutare i diversamenti abili ad essere più indipendenti. Lo scopo del progetto è l'identificazione della posizione del pulsante desiderato e l'invio di tali informazioni al controllore del braccio robotico. L'utente con il supporto di un tablet, preme su di esso il bottone desiderato. Una volta identificato il pulsante premuto, la camera fornisce al robot le informazioni di posizione ad orientazione in modo da posizionarsi sul percorso corretto. La camera utilizzata è monoculare pertanto non si hanno informazioni sulla distanza. Per ovviare a questa mancanza si è deciso di utilizzare un algoritmo esterno chiamato *PTAM* [1] capace di utilizzare le informazioni delle features trovate per stimare una posizione relativa. Il codice si divide in tre fasi:

- Fase di plan
- Fase di tracking
- Interfacciamento con l'algoritmo esterno
- Fase di triangolazione e scalatura

Nella prima fase si utilizzano algoritmi di elaborazione dell'immagine, in modo da ottenere la posizione dell'oggetto in coordinate camera. Nella seconda fase si ricercano nella seconda immagine le features di interesse e si stima la posizione del bottone tramite algoritmi di matching. Nella terza fase si elaborano i dati ricevuti dall'algoritmo esterno, Ptam. Nella quarta e ultima fase i dati elaborati vengono triangolati e, opportunamente scalati per trovare la posa relativa.

I sistemi di acquisizione maggiormente impiegati possono essere suddivisi in due categorie. Una prima categoria fa riferimento al modo in cui viene usato il sistema di visione. Se la telecamera inquadra direttamente l'oggetto dal punto di vista del manipolatore, si parla di sistemi "eye-in-hand ". In questo caso la posizione e l'orientamento della telecamera sono solidali con l'end-effector del manipolatore. Un'altro tipo di configurazione consente di avere la telecamera che mantiene una posizione fissa nell'ambiente e al più con orientamento variabile, ed inquadra sia il manipolatore che l'oggetto target. Quest'ultima modalità viene denominata "eye-to-hand ".. È possible notare le differenze nelle capacità visive del manipolatore in questa due modalità di configurazione dalla figura 1.1

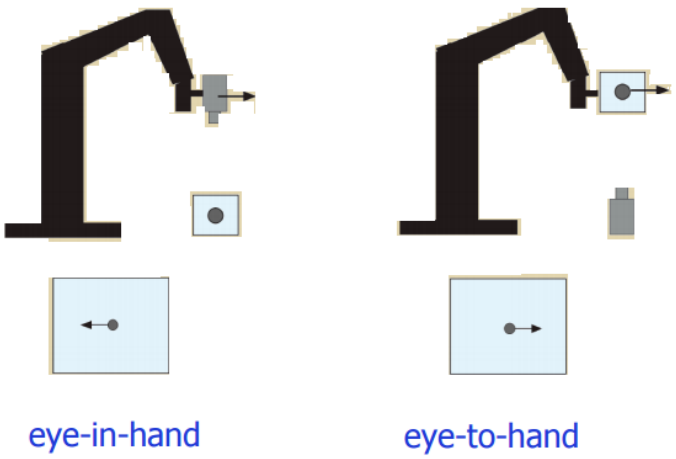


Figura 1.1: Differenza tra le due categorie di posizionamento della camera

Nelle tecniche tradizionali di asservimento visivo viene invece spesso effettuata la misura diretta della posizione del manipolatore fornita dalla telecamera, per migliorare l'accuratezza del posizionamento del end-effector. I dati forniti dalla telecamera, in questo caso, incrementano la precisione delle informazioni fornite dall'apparato meccanico del robot, in quanto non risultano affetti dai tipici errori dovuti ad un'imprecisa conoscenza dei parametri fisici del manipolatore, che risultano particolarmente evidenti nel caso in cui si debba considerare anche la flessibilità della struttura. L'ambiente, in cui si muove il robot, possiede alcune caratteristiche che inevitabilmente influenzano il processo di comprensione della scena. Nei sistemi robotici dotati di sensori di visione vengono spesso applicate le seguenti ipotesi:

- il robot deve essere un agente autonomo, per cui non è possibile alcun intervento esterno per controllarlo;

- gli oggetti da riconoscere presenti nell'immagine devono essere fortemente caratterizzati;
- le condizioni di illuminazione della scena sono in genere fortemente variabili, questo influisce molto sul riconoscimento degli oggetti;
- le occlusioni, rendono impossibile il riconoscimento degli oggetti basandosi sull'analisi della loro forma

In questo progetto è stato impiegato una configurazione del primo tipo.

1.0.1 Organizzazione del report

Nel capitolo 2 si spiega il modello della camera utilizzata e le sue caratteristiche. Nel capitolo 3 verrà spiegato il meccanismo di tracking e del matching mentre nel capitolo 4 viene introdotto l'algoritmo sviluppato. Nel capitolo 5 vengono esposti gli esperimenti e nel 6 le conclusioni. In appendice A vi è una panoramica del codice sviluppato.

Capitolo 2

Modello della camera

La camera utilizzata è una camera monoculare posta sul end-effector del braccio robotico. Questo tipo di telecamera ha un ingombro minimo, ma si perde la percenzione della distanza. Per ricostruire queste informazioni occorre utilizzare la *stereo visione*[1] utilizzando la *geometria epipolare* [2]. La camera utilizzata nel progetto è la camera monoculare della Playstation 3 mostrata in Fig 2.1 di produzione della Sony con una risoluzione di 640X480 pixel.

Figura 2.1: Camera utilizzata nel progetto

[1]Stima della profondità tramite l'utilizzo di due immagini della stessa scena prese da angolazioni differenti

[2]Essa descrive le relazioni e i vincoli geometrici che legano due immagini 2D della stessa scena 3D catturata da due fotocamere con posizione e orientamento distinto

2.0.2 Modello pinhole

La geometria di tale modello prevede che i raggi luminosi provenienti dal mondo che passano attraverso un foro di dimensione infinitesima, vadano a formare l'immagine su un piano disposto dietro il foro, chiamato *piano immagine*. In termini matematici consideriamo un punto nel mondo con cordinate $X_c = [X_c, Y_c, Z_c]^T$ il cui sistema di riferimento O è centrato nel foro. Questo sistema di riferimento prende il nome di *centro ottico*. L'asse z è l'asse ottico, ed è la retta perpendicolare al piano dell'immagine passante per il centro ottico. Il raggio luminoso che segue l'asse ottico è chiamato *raggio principale*. La distanza fra il piano immagine e il centro ottico è la *lunghezza focale* f. La figura 2.2 mostra lo schema di queste definizioni.

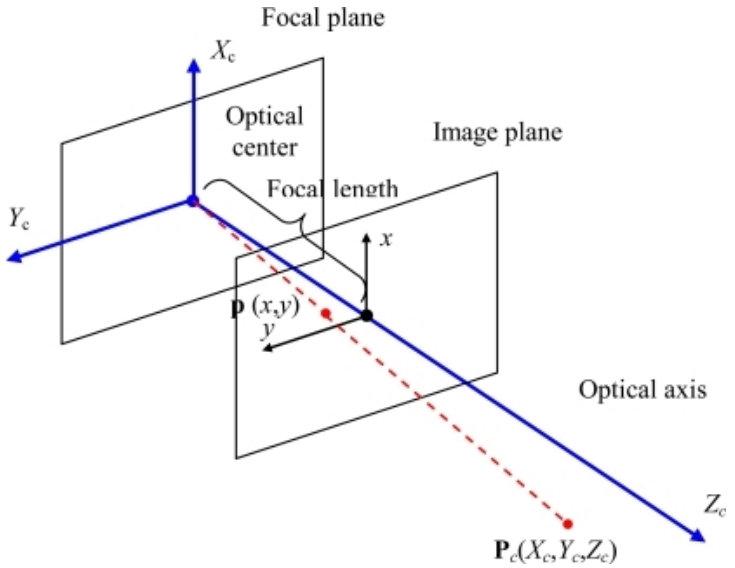


Figura 2.2: Modello schematico del modello pinhole

Con queste definizioni possiamo chiamare *parametri instrinseci* tutti quei parametri necessari a collegare le coordinate di un pixel dell'immagine con le coordinate corrispondenti nel sistema di riferimento della camera. In altre parole sono i parametri necessari a specificare le caratteristiche ottiche, geometriche e digitali della camera. Tali parametri sono:

- la lunghezza focale
- le coordinate in pixel del centro dell'immagine
- la distorsione geometrica
- la dimensione dei pixel

I *parametri estrinseci* sono tutti quei parametri che definiscono la posizione ed orientazione del sistema di riferimento della camera rispetto al riferimento mondo. Sono un insieme di parametri geometrici che identificano univocamente le trasformazioni tra il sistema di riferimento della camera e quello mondo, supposto noto. Tali parametri sono:

- vettore di traslazione
- matrice di rotazione 3x3 ortogonale

Il punto sul piano immagine $x = [x_i, y_i]^T$ corrispondente a X_c è definito come in figura 2.3

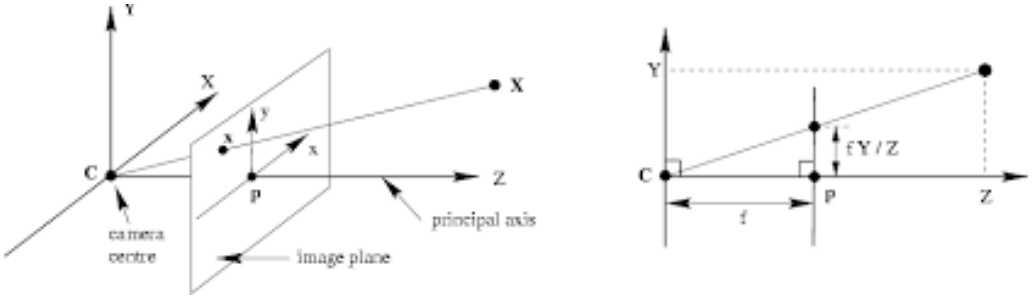


Figura 2.3: Punti nel piano immagine

$$\begin{cases} x_i = f_x * \frac{X_c}{Z_c} + c_x \\ y_i = f_y * \frac{Y_c}{Z_c} + c_y \end{cases} \tag{2.1}$$

Utilizzando il modello pinhole, il punto X_c proiettato nel piano immagine è esprimibile come:

$$s * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} * \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.2}$$

Dove c_x e c_y sono i centri ottici espressi in pixel e A è la matrice dei parametri intrinseci, il fattore s che moltiplica il vettore dei punti nel piano immagine è detto *fattore di scala*. La matrice di roto-traslazione è chiamata matrice dei parametri estrinseci, ed è usata per descrivere il moto della camera attorno ad una scena statica. Nella figura 2.4 si può vedere uno schema del modello pinhole con la descrizione dei parametri intrinseci ed estrinseci.

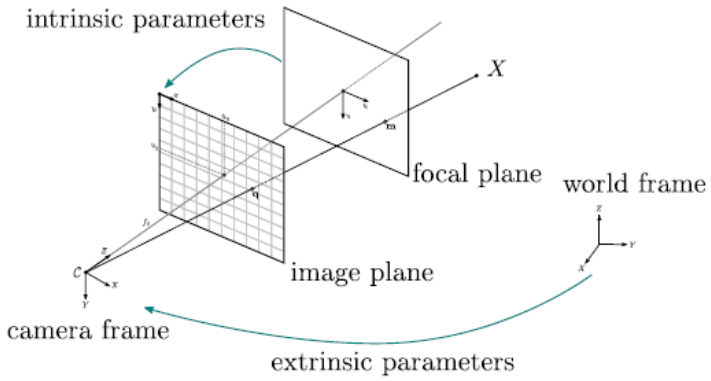


Figura 2.4: Schema del modello pinhole con parametri estrinseci ed intrinseci

Capitolo 3

Tracking and Matching tramite SIFT

Il tracking è l'operazione di seguire lo spostamento di un oggetto o di un insieme di punti in frame consegutivi. I punti usati per questa operazioni vengono chiamati *features*. La qualità del tracking influenza in maniera molto significativa gli algoritmi che lo utilizzano. Nel codice sviluppato, il tracking viene utilizzato per stimare la posizione delle features dato il loro moto. Per lo studio in esame si è deciso di usare il tracking con il *SIFT* acronimo di Scale-Invariant Feature Transform. Gli operatori utilizzati tradizionalmente in fotogrammetria (Forstner e Gulch 1988 [2]; Harris e Stephens 1988 [3]) ricercano punti omologhi (point detector) su spigoli o discontinuità radiometriche; al contrario l'operatore SIFT ricerca i punti (keypoint) su regioni più ampie dell'immagine (region detector) superando i problemi di occlusione e deformazione prospettiche. Questo metodo è stato sviluppato da D. G. Lowe [4] e permette di rilevare i punti caratteristici di una immagine e di associargli dei descrittori basati sui gradienti delle aree dell'immagine nell'intorno delle features. Questi descrittori sono utilizzati per ricercare il punto nelle immagini successive. I passi principali dell'algoritmo SIFT sono i seguenti:

- Individuazione degli estremi locali nello scale-space: si cercano punti interessanti su tutte le scale e posizioni dell'immagine utilizzando una funzione DoG -Difference of Gaussian-. L'approccio utilizzato è quello del filtraggio in cascata (cascade filtering approach), che consente di determinare le posizioni e la scala delle features candidate ad essere punti chiave e che, in un secondo momento, vengono caratterizzate con maggior dettaglio.

- Localizzazione dei keypoint: per ciascun punto candidato viene costruito un modello dettagliato per determinarne posizione e scala. I punti vengono inoltre selezionati secondo misure di stabilità.

- Generazione delle orientazioni: per ottenere l'invarianza rotazionale, ad ogni punto chiave (keypoint) vengono associate una o più orientazioni calcolate in base ai gradienti locali dell'immagine.
- Generazione del descrittore: a partire dai gradienti locali dell'immagine, alla scala selezionata e nell'intorno del punto chiave, viene costruito il descrittore.

Lo scale-space è una funzione che permette di calcolare punti dell'immagine che sono invarianti a cambiamenti di scala definito come una funzione $L(x, y, \sigma)$ data dalla convoluzione in x e y di una funzione Gaussiana, variabile in scala, $G(x, y, \sigma)$ con l'immagine $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y) \tag{3.1}$$

Ogni immagine è filtrata mediante convoluzioni di gaussiane, formando uno spazio delle "scale ". Per ogni scala sono quindi calcolate le differenze fra gaussiane adiacenti DoG, i cui massimi sono memorizzati come punti di interesse *keypoints* 3.1.

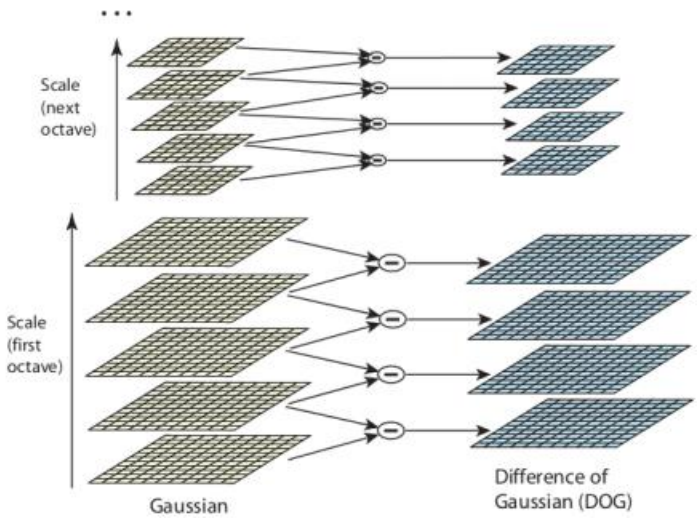


Figura 3.1: Difference of Gaussian

Per ogni keypoint individuato è quindi definito un descrittore capace di descrivere i gradienti radiometrici nell'intorno del punto di interesse indipendentemente da rotazioni, variazioni di scala e cambiamenti di illuminazione. In base alla distanza euclidea fra questi vettori n-dimensionali è infine possibile individuare keypoints omologhi fra le immagini. Un esempio di punti di interesse estratti è mostrato in figura 3.2

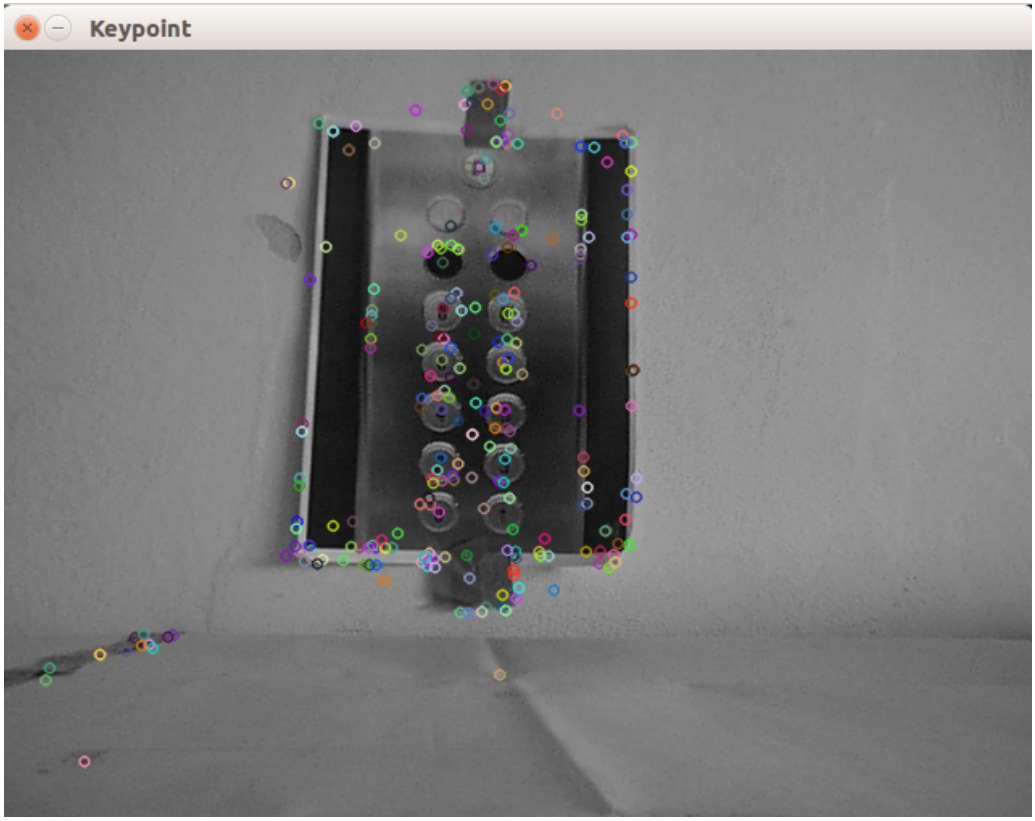


Figura 3.2: Keypoint

Il keypoint è quindi costituito dalle seguenti informazioni:

- posizione nel piano immagine
- scala alla quale è stato rilevato il punto caratteristico
- orientamento ossia la direzione principale del gradiente della regione nell'intorno del punto
- descrittore che descrive in maniera univoca il punto caratteristico

Ad ogni frame vengono calcolati i keypoint e i relativi descrittori. Per ogni feature della prima immagine viene confrontato il suo descrittore con tutti i descrittori dei keypoint della seconda immagine. Il punto caratteristico con il descrittore che gli assomiglierà di più e che supera una soglia di somiglianza verrà definito come il punto corrispondente della prima immagine. In figura 3.3 si può vedere un esempio di matching per il compito in esame.

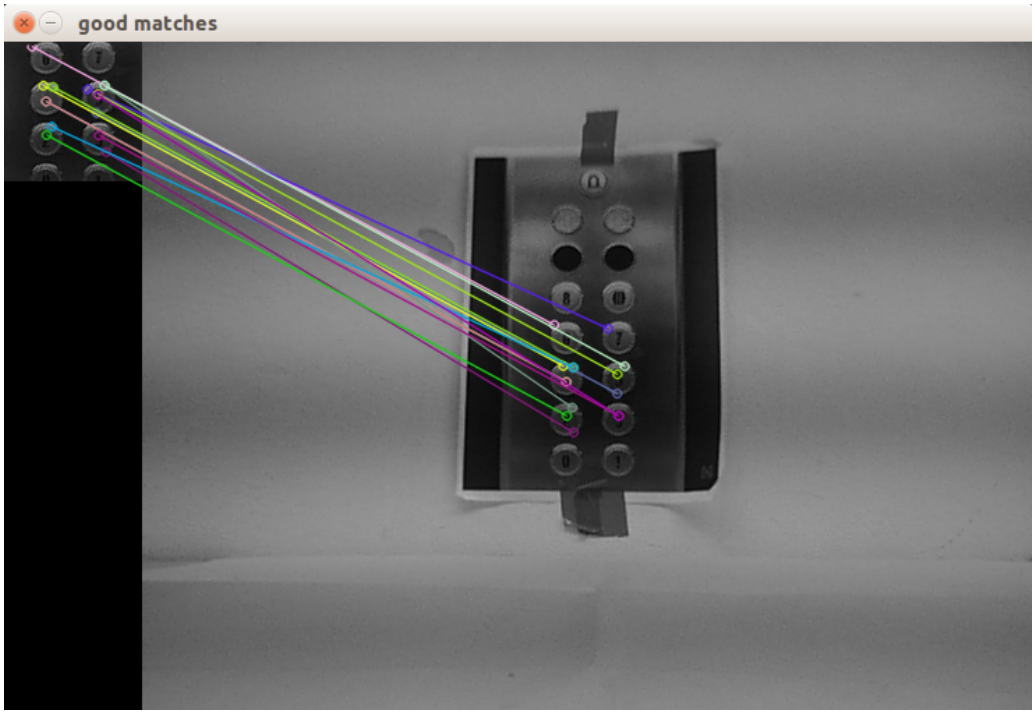


Figura 3.3: Match tra due frame consegutivi

Capitolo 4

Algoritmo

Lo scopo del progetto è di tracciare e stimare la posizione 3D del pulsante desiderato durante il moto del braccio utilizzando una camera monoculare. Come definito nel capitolo 1 la camera non fornisce informazioni sulla profondità. Per ottenere la posizione 3D date le informazioni 2D del pulsante si utilizza un algoritmo esterno detto PTAM. Per gestire i due algoritmi si utilizza il framework ROS atto a far comunicare più algoritmi contemporaneamente tramite l'invio di messaggi. In figura 4.1 si è schematizzata la struttura dell'algoritmo.

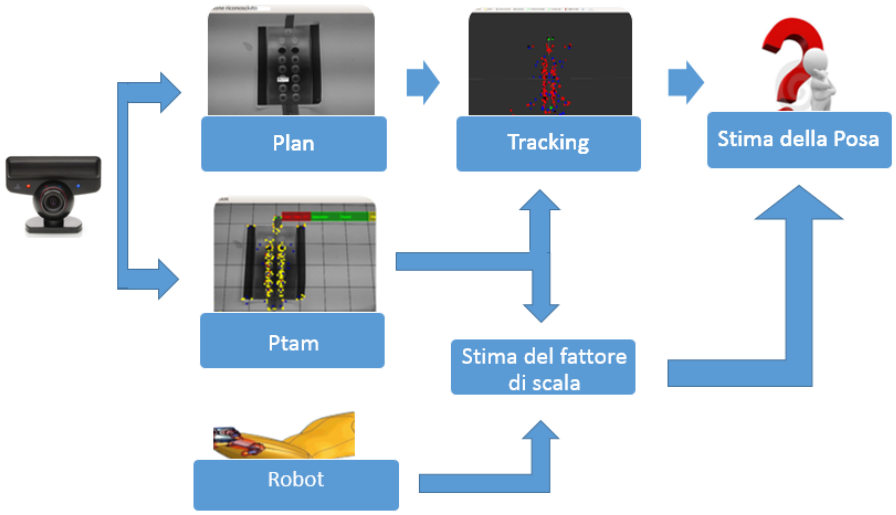


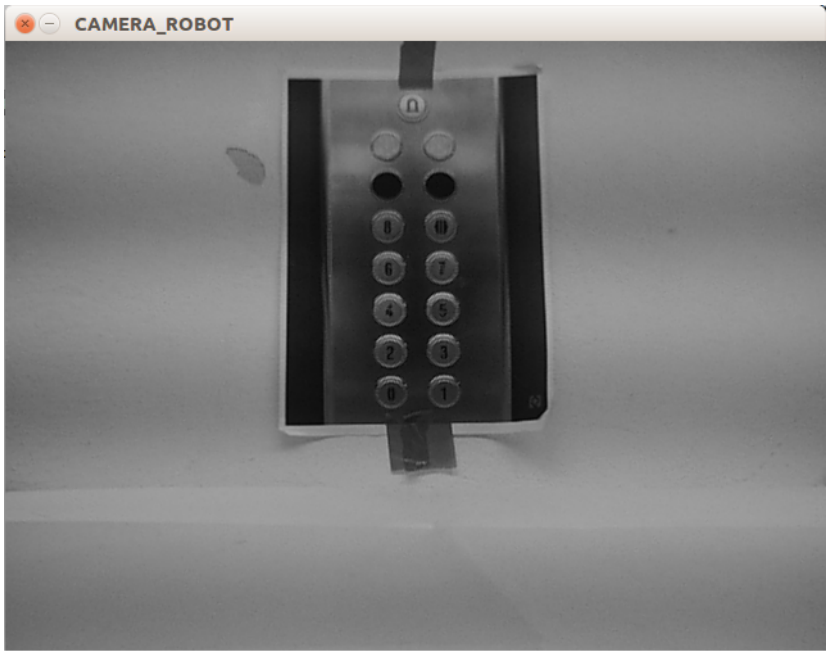
Figura 4.1: Struttura dell'algoritmo sviluppato

La prima parte è dedicata al riconoscimento delle forme geometriche nella scena e all'interazione con l'utente. La seconda parte è dedicata al tracking del-

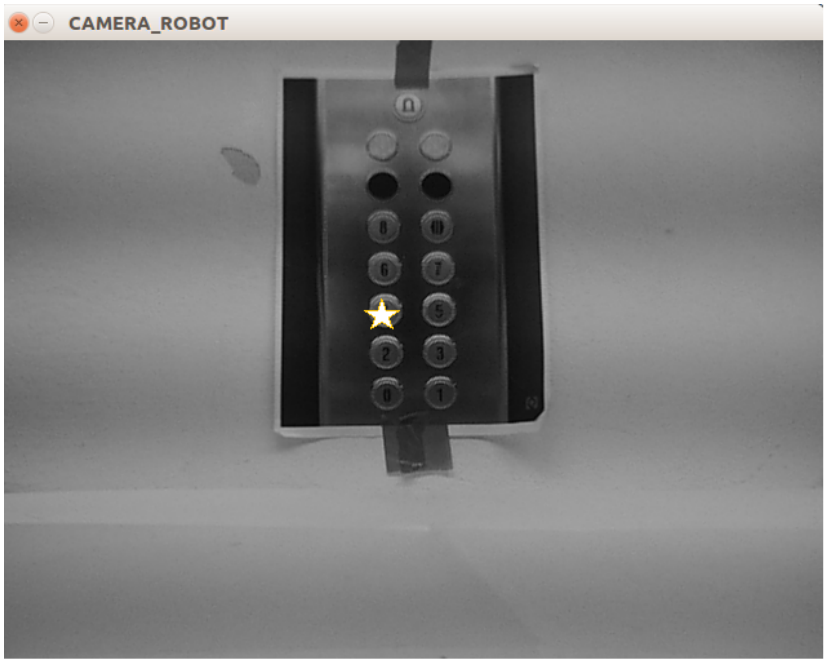
l'oggetto desiderato nei successivi frame. L'ultima parte ricerca la posizione 3D del oggetto utilzzando sia le informazioni provenienti da PTAM, sia le informazioni sullo spostamento reale del robot. Nell'allegato A.0.11 vengono mostrati le strutture degli algoritmi implementati.

4.0.3 Fase di plan

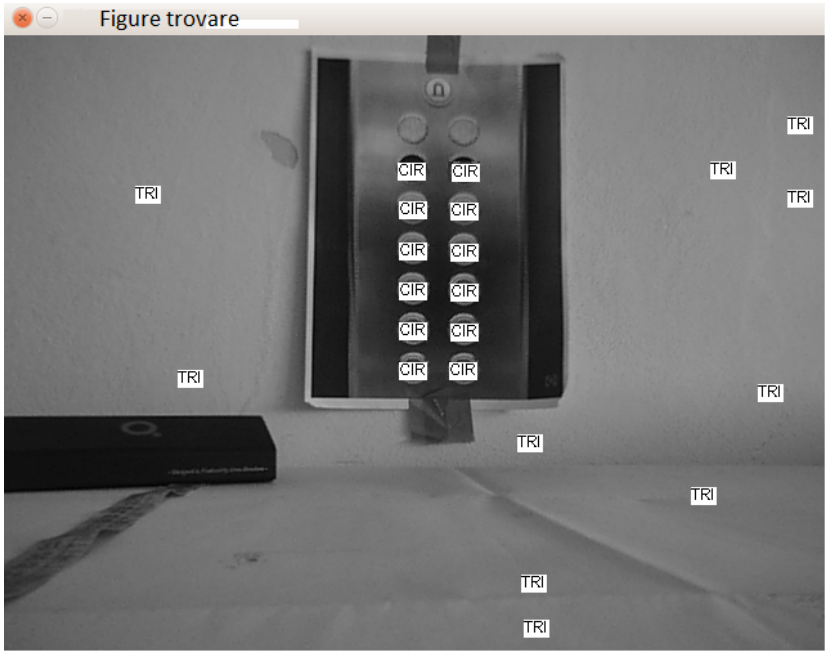
La telecamera invia lo streaming della scena al nodo Ros di PTAM e contemporaneamente invia un immagine statica, della stessa scena, al nodo da noi creato. Con questa immagine statica siamo in grado di riconoscere i contorni delle immagini tramite il filtro di Harris [3]. Questo filtro è basato sull'approssimazione del auto-correlazione del gradiente in diverse direzioni. Per ogni contorno trovato si calcola i centri di massa e si verifica quale tra questi centri ha la minor distanza euclidea dal punto premuto dal utente. Dall'immagine si estrapola la zona dell'oggetto interessato e si calcolano le features e i relativi descrittori utilizzando il filtro Sift. Nella figura 4.2 si può vedere un esempio di questo algoritmo.



(a) *Scena_iniziale*



(b) *Bottone_scelto*



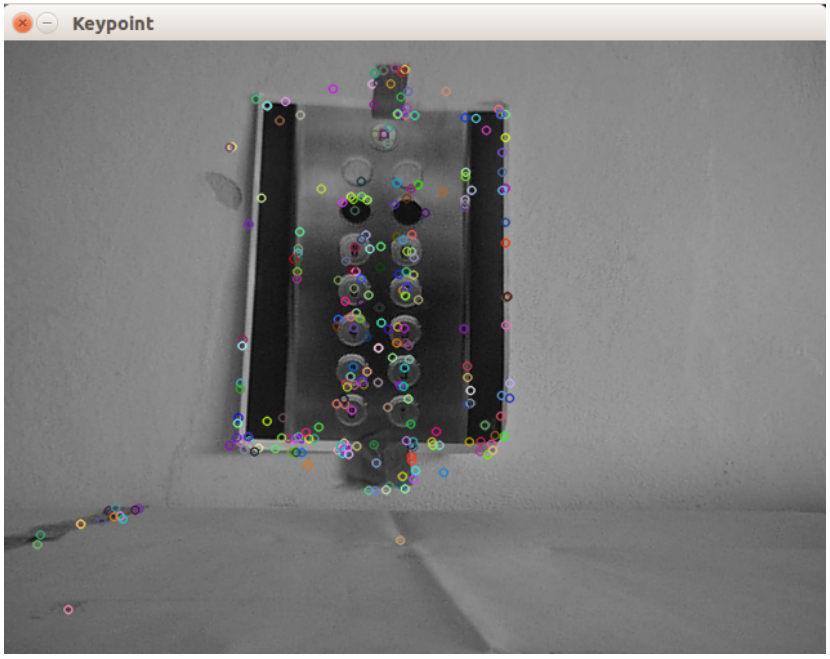
(c) *Figure_trovate*

(d) *Bottone_riconosciuto*

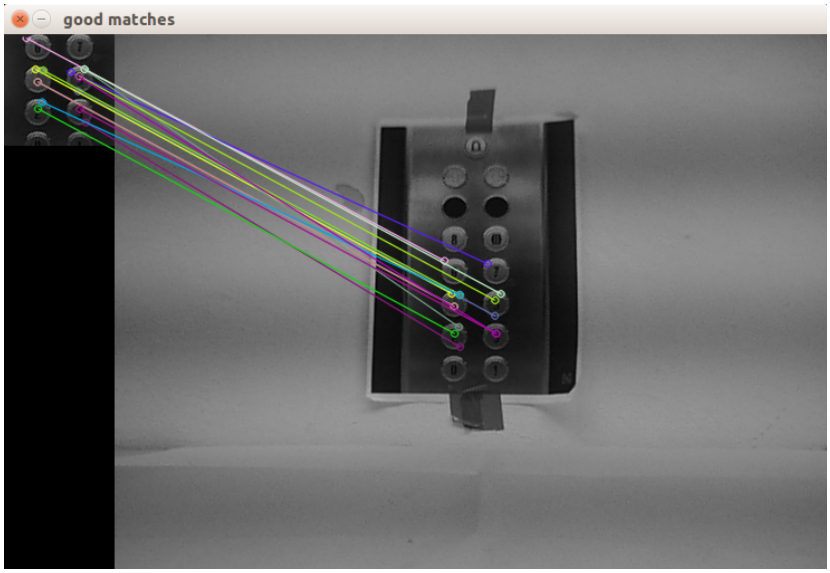
Figura 4.2: Esempio di ricerca del bottone selezionato

4.0.4 Fase di tracking

Quando il robot si muove invia un messaggio con la posa rispetto al frame precedente. In questo modo si tiene traccia dei reali spostamenti del robot. Nel nuovo frame si ricalcolano i descrittori dell'intera immagine e si calcola il match con le features del pulsante riconosciuto precedentemente. Il match è stato calcolato tramite il metodo FLANN acronimo di *fast approximate nearest neighbor* [5]. Questo metodo consiste nel trovare i punti vicini tramite una ricerca randomica dell'albero creato. Utilizzando i migliori descrittori, che realizzano il matching, si calcola il centroide di questi punti. Questo nuovo punto rappresenta la posizione del pulsante nel nuovo frame. Ad ogni iterazione si esegue il matching con il pulsante trovato nella prima immagine. In figura 4.3 si può vedere un esempio del matching per il compito in esame. Come si può notare dalla figura 4.3(b) i descrittori trovati possono essere molti, anche distanti dalla zona di interesse. Per questa ragione si utilizza un controllo aggiuntivo basato sulle distanze per minimizzare questo numero. In figura 4.3(c) si mostrano i punti trovati utilizzando il controllo aggiuntivo. Questo controllo aggiuntivo fa in modo che la distanza euclidea dei punti del match non superi mai una soglia imposta. In questo modo riusciamo ad eliminare possibili outlier e a ridurre il campo di ricerca.



(a) $Sift\_frame2$



(b) $Matching$

(c) *Good_matching*

(d) *Botton_frame2*

Figura 4.3: Tracking del pulsante

4.0.5 Fase di interazione

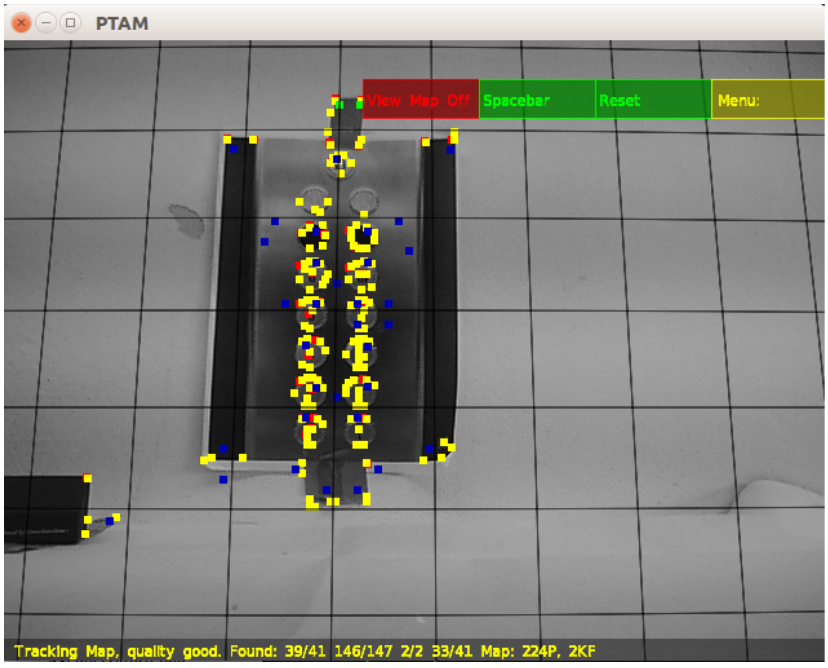
4.0.5.1 Ptam

Per spiegare come si è ottenuta la posa del pulsante rispetto alla telecamera, occorre spiegare come funziona l'algoritmo PTAM. Questo algoritmo è usato per la realtà aumentata e può essere riassunto da:

- Tracking delle features in ogni frame
- Mapping delle features

Una mappa è un insieme di M features rispetto al frame "mondo"W. Ogni features rappresenta un punto nel mondo 3D con coordinate $p_{jw} = (x_{jw}, y_{jw}, z_{jw}, 1)^t$ rispetto al frame W. La mappa è creata in fase di inizializzazione tramite la stereo visione. Come spiegato nel capitolo 2 la stereo visione è la stima della profondità tramite l'utilizzo di due immagini della stessa scena prese da angolazioni differenti. Durante la fase di inizializzazione la camera viene mossa sull'asse orizzontale, così facendo si tracciano le features durante lo spostamento e si stima la posizione 3D tramite algoritmi randomici. Un esempio di questa inizializzazione è mostrato in figura 4.4.

(a) *Fase_init*



(b) *Ptam_grid*

Figura 4.4: Fsse di inizializzazione di PTAM

Oltre alla posizione 3D delle features, figura 4.5(c), rispetto al frame word questo algoritmo fornisce la posa della camera rispetto al frame W 4.5(a). Questa posa è calcolata iterativamente minimizzando una funzione obbiettivo dato dall'errore di riproiezione nel piano immagine:

$$\mu = \underset{\mu}{\operatorname{argmin}} \sum_{j \in S} Obj(\frac{|e_j|}{\sigma}, \sigma_t) \tag{4.1}$$

Dove $|e_t|$ è l'errore di riproiezione dato da:

$$e_t = \begin{pmatrix} u_j \\ v_j \end{pmatrix} - CamProj(exp(\mu) * E_{cw} * p_j) \tag{4.2}$$

Il termine $CamProj(\exp(\mu) * E_{cw} * p_j)$ rappresenta il movimento della camera espresso come un vettore di μ usando una mappa esponenziale.

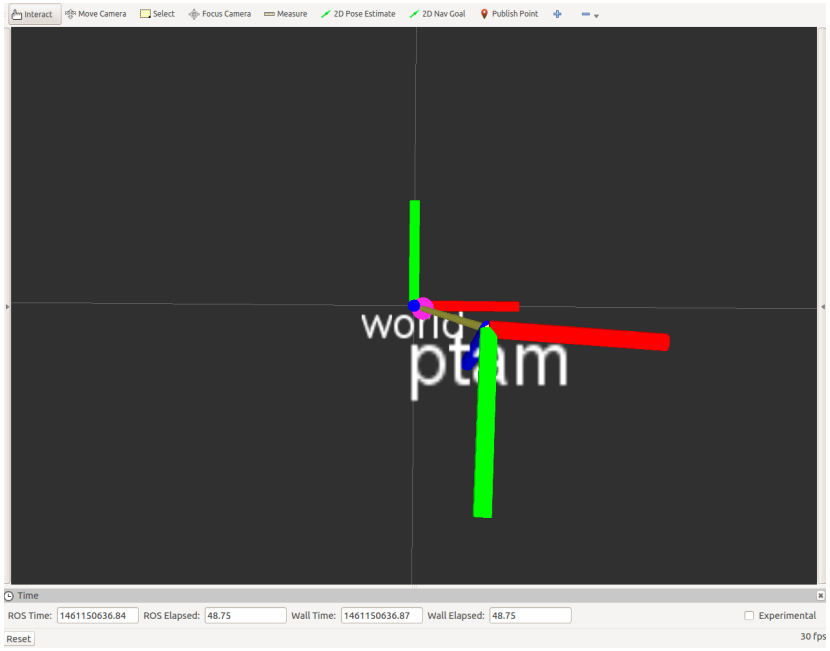
4.0.5.2 Stima del fattore di scala

Come detto nel capitolo 1 questo algoritmo è soggetto ad un errore di scala. Se non correggessimo questo errore ci troveremmo in una posizione diversa rispetto alla posizione vera. Per questa ragione per correggere la scala si sono scritti due differenti metodi. Il primo metodo consente in un iterazione di stimare il valore della scala. Per questo metodo si utilizza il rapporto tra il reale spostamento del robot tra due sistemi di riferimento, e lo scostamento dato da Ptam.

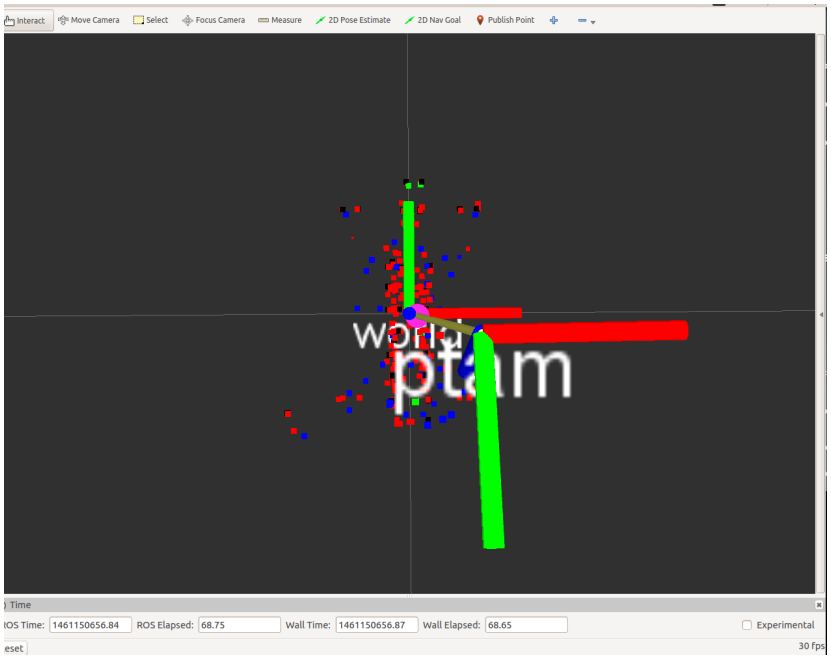
$$s = \frac{Pr_{t-1} - Pr_t}{Pp_{t-1} - Pp_t} \tag{4.3}$$

Dove $Pr_{t-1} - Pr_t$ è lo spostamento reale del end-effector, mentre $Pp_{t-1} - Pp_t$ è lo spostamento secondo Ptam. Come vedremo nel capito 5 questo metodo è soggetto ad un errore di stima. Per ovviare a questo errore, si utilizzano dei metodi che permettono la convergenza della stima della scala al suo valore reale. Nel suo articolo Jacob Engel [6] fornisce una stima dela fattore di scala utilizzando lo stimatore a massima verosimiglianza. Essenzialmente tale principio stabilisce di scegliere per un campione $\underline{x}$ dato, quel valore di θ per cui massima era la probabilita' di estrarre proprio quel $\underline{x}$. Poiche' il logaritmo e' una funzione monotona, al posto del massimo di $L(\underline{x}, \theta)$ si preferisce cercare il massimo di $logL(\underline{x}, \theta)$. Dato una coppia di punti (x_i, y_i), dove x_i è la posizione affetta dal fattore di scala e y_i è il valore reale, questi sono in relazione $x_i \approx \lambda y_i$. Se si ipotizza che il rumore di misura sia di tipo Gaussiano questi campioni possono essere scritti:

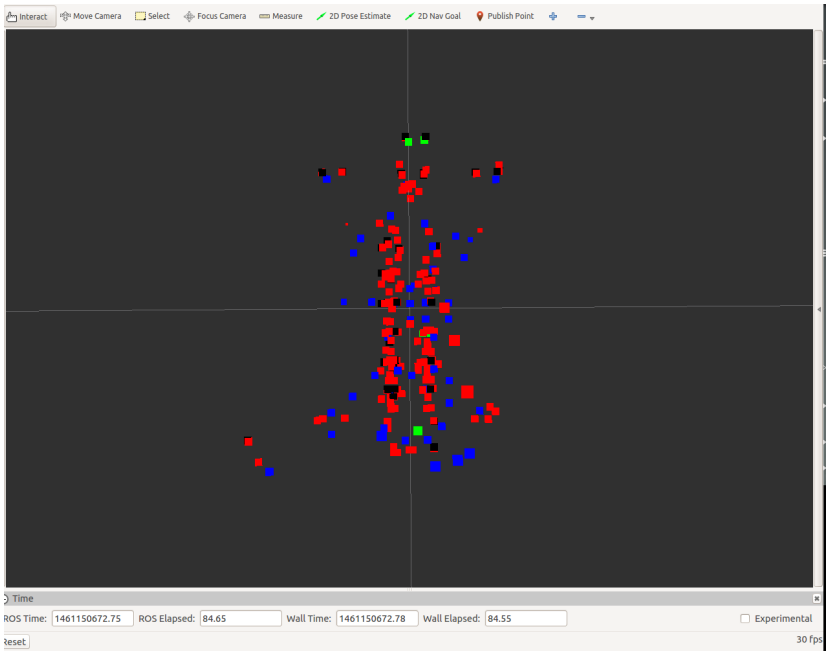
$$\begin{cases} x_i \sim \Gamma(\lambda\mu_i, \sigma_x^2 I_{d*d}) \\ y_i \sim \Gamma(\mu_i, \sigma_y^2 I_{d*d}) \end{cases} \tag{4.4}$$



(a) *Frame*



(b) *Point_W_frame*



(c) $Point\_W\_frame$

Figura 4.5: Posizione delle features rispetto al word frame

Dove $\mu_1....\mu_i \epsilon \Re^d$ rappresentano le vere ma ignote distanze, σ_x^2 e σ_y^2 rappresentano le varianze degli errori di misura. Assumendo che le varianze siano note è possiile trovare una soluzione in forma chiusa dello stimatore minimizzando la funzione:

$$L(\mu_1..\mu_n, \lambda) \propto \frac{1}{2}\sum_{i=1}^{n}(\frac{||x_i - \lambda\mu_i||^2}{\sigma_x^2} + \frac{||y_i - \mu_i||^2}{\sigma_y^2}) \tag{4.5}$$

Minimizzando per λ è possibile trovare una soluzione unica

$$\lambda^* = \frac{s_{xx} - s_{yy} + sign(s_{xy} * \sqrt{(s_{xx} - s_{yy})^2 + 4 * s_{xy}^2})}{2 * \sigma_x^- 1 * \sigma_y * s_{xy}} \tag{4.6}$$

Dove:

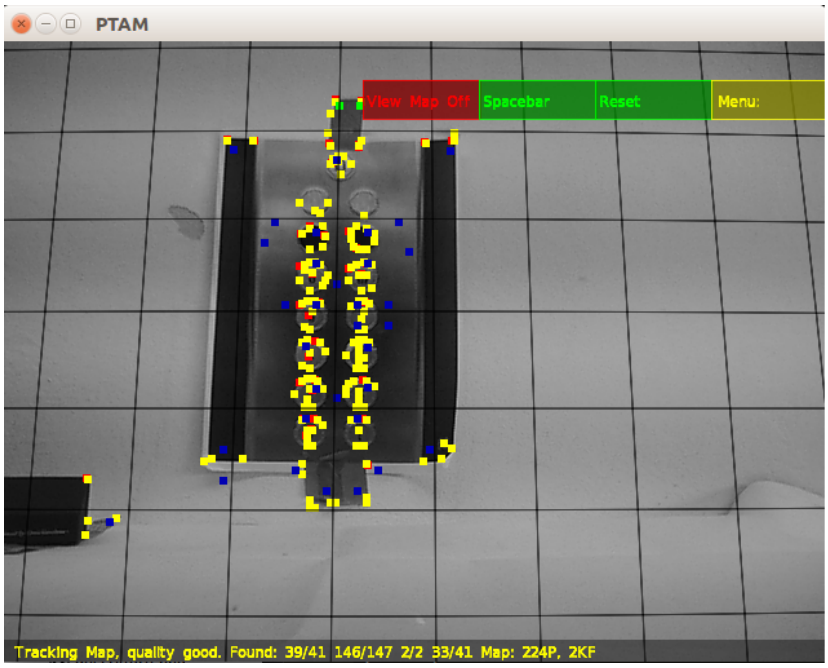
$$s_{xx} = \sigma_y^2 * \sum_{i=1}^{n} x_i^t x_i \qquad s_{yy} = \sigma_x^2 * \sum_{i=1}^{n} y_i^t y_i \qquad s_{xy} = \sigma_y^2 * \sigma_x^2 * \sum_{i=1}^{n} x_i^t y_i^2 \tag{4.7}$$

Questo metodo assicura la convergenza per un numero finito di passi. Nel capitolo 5 analizzeremo durante un esperimento i due metodi in esame.

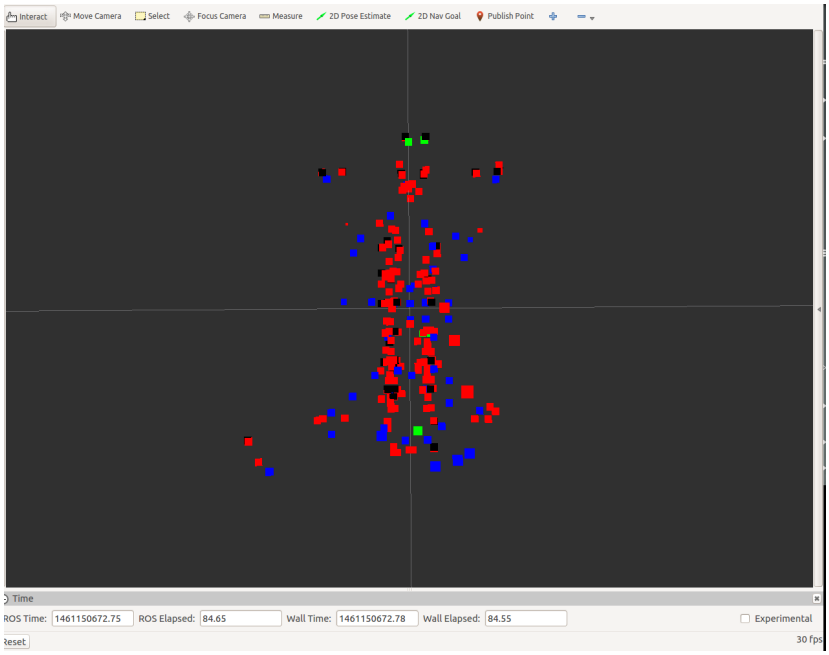
4.0.6 Posizione 3D del pulsante

Per quanto detto finora PTAM tramite l'inizializzazione della mappa crea una collezione di punti 3D riferiti al frame mondo. La posizione delle features di interesse può essere acquisita da un utente esterno tramite la lettura del topic di riferimento. Questi punti sono inviati tramite un messaggio ROS in forma di point cloud[1] e, come si puo vedere da figura 4.6 rappresentano il 3D delle features trovate nella immagine.

[1] Point cloud o nuvola di punti è un insieme di punti caratterizzati dalla loro posizione in un sistema di coordinate e da eventuali valori di intensità (colore, profondità, ecc.) ad essi associati. Esse servono solitamente come rappresentazione di strutture tridimensionali come oggetti o superfici



(a) *Features_2d*



(b) *Point_cloud*

Figura 4.6: Point cloud della mappa 3D

Dalle informazioni della point cloud è possibile trovare le informazioni 2D riproiettando i punti nel piano immagine. Una volta che i punti vengono riproiettati si scelgono quelli più vicini al pulsante desiderato. Questa discriminazione viene fatta semplicemente calcolando tutte le distanze euclide dal pulsante. I punti scelti saranno quelli la cui distanza è minore di una soglia scelta, ci assicuriamo in questo modo di eliminare eventuali outlier presenti nel set di dati utilizzato. Grazie a questi punti è possibile trovare la posizione 3D del pulsante. Per ottenere questa infomazione si è deciso di fittare, tramite i minimi quadrati, un piano passante per questi punti. Utilizzando i parametri del piano assieme all'equazione 2.1 è possibile trovare le coordinate 3D:

$$\begin{cases} a * X_c + b * Y_c + c * Z_c + d = 0 \\ x_i = f_x * \frac{X_c}{Z_c} + c_x \\ y_i = f_y * \frac{Y_c}{Z_c} + c_y \end{cases} \tag{4.8}$$

Si hanno tre equazioni in tre incognite.

$$\begin{cases} a * \frac{x_i - c_x}{f_x} * Z_c + b * \frac{y_i - c_y}{f_y} * Z_c + c * Z_c - d = 0 \\ X_c = \frac{x_i - c_x}{f_x} * Z_c \\ Y_c = \frac{y_i - c_y}{f_y} * Z_c \end{cases} \tag{4.9}$$

Da cui risolvendo per Z_c si ottiene.

$$Z_c = \frac{d}{a * \frac{x_i - c_x}{f_x} + b * \frac{y_i - c_y}{f_y} + c} \tag{4.10}$$

La posizione delle features che fornisce PTAM è rispetto al frame mondo, per i metodi implementati bisogna trasformarle nel sistema di riferimento camera tramite una trasformazione omogenea 4.11. Dove T_c^w rappresenta le cordinate del frame camera rispetto il frame mondo.

$$P_c = T_c^w * P_w \tag{4.11}$$

Capitolo 5

Esperimenti

Nel capitolo 4 abbiamo parlato di due metodi per stimare il fattore di scala. Il primo è un metodo semplificato, dove la stima è fatta solo su di un campione e non c'è modo di verificare se questo valore è corretto. Se questa stima sbaglia, l'errore che genera viene portato avanti fino alla fine dell'esperimento. Come si vede in figura 5.1 e 5.2 la stima della scala con questo metodo, porti ad un errore massimo di stima della posizione di 1 cm per ogni spostamento effettuato.

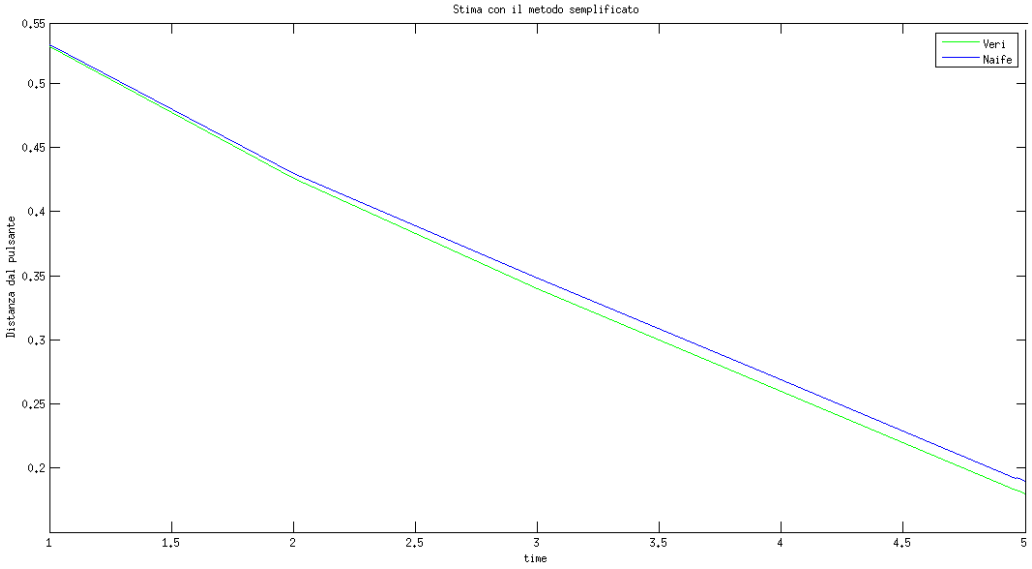


Figura 5.1: Grafico dell'errore di stima della posa. In blu i dati stimati mentre in verde quelli reali

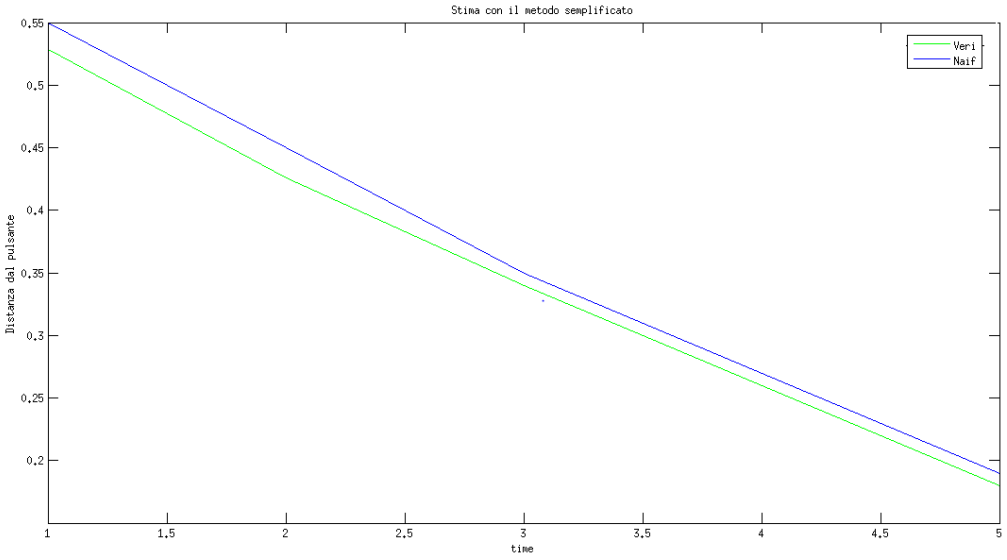


Figura 5.2: Grafico dell'errore di stima della posa. In blu i dati stimati mentre in verde quelli reali

Nel secondo metodo proposto invece, con l'utilizzo dello stimatore a massima verosomiglianza è possibile, utilizzando molti campioni, riuscire a far convergere l'errore di stima del fattore di scala al volore reale. Per questo metodo occorro molti campioni. Per verificare la validità del metodo prendiamo un numero di campioni sufficientemente grande (maggiore di 500) di uno o più spostamenti. Per gli esperimenti si è scelto di utilizzare 1000 campioni di un unico spostamento, la convergenza è raggiunta se la differenza tra due campioni successivi è minore di 0.0001. Nelle figure 5.3 e 5.4 possiamo vedere la convergenza della stima della scala per due esperimenti effettuati.

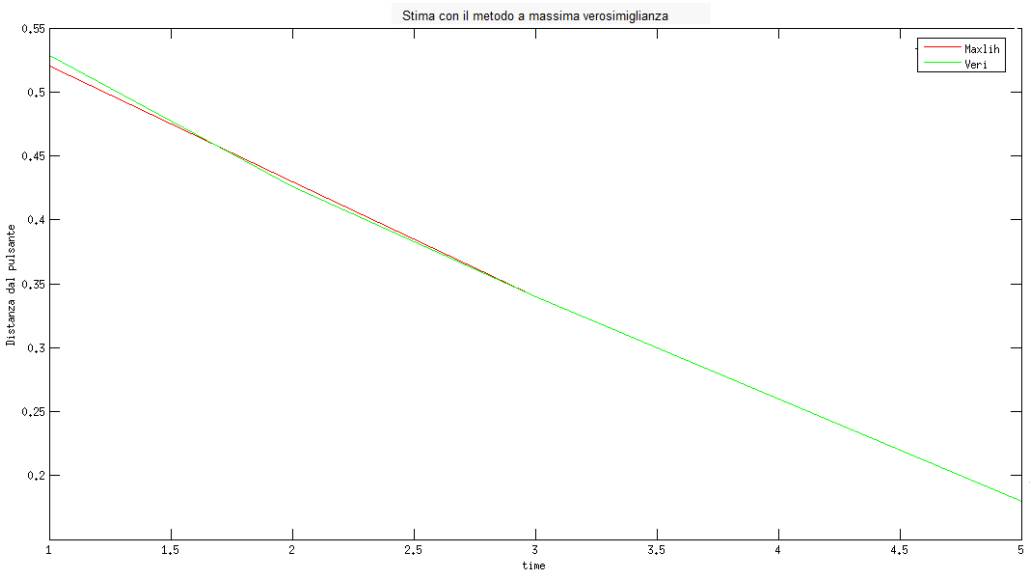


Figura 5.3: Grafico dell'errore di stima della posa. In rosso i dati stimati mentre in verde quelli reali

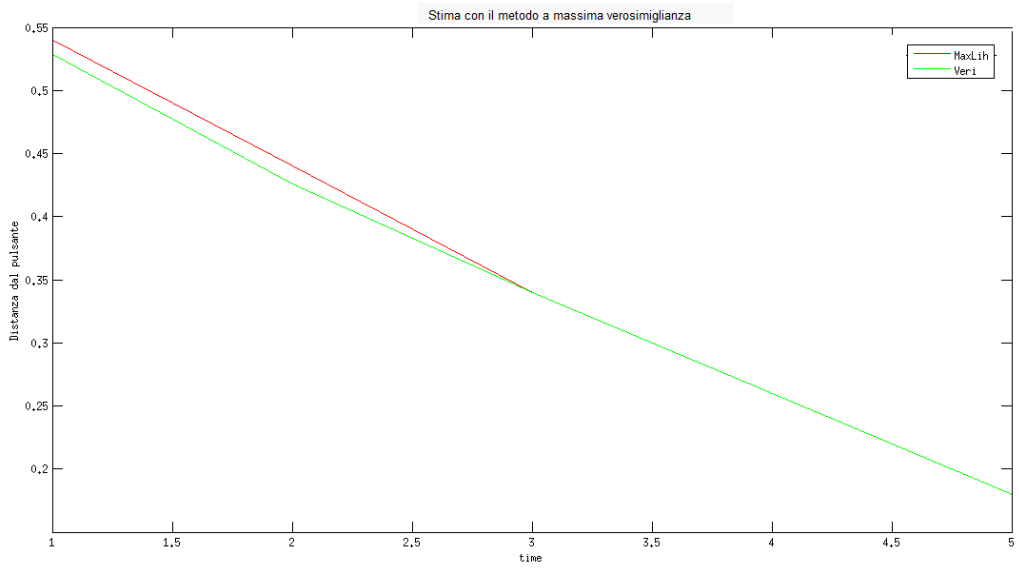


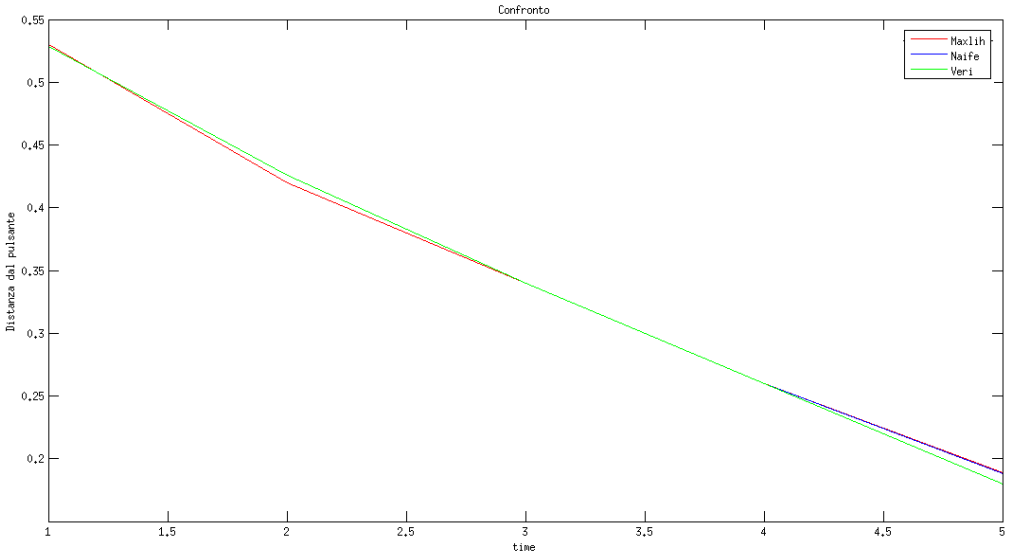
Figura 5.4: Grafico della stima del valore di scala. In rosso i dati stimati mentre in verde quelli reali

Nei due grafici sopra esposti si può vedere come la stima della scala con questo

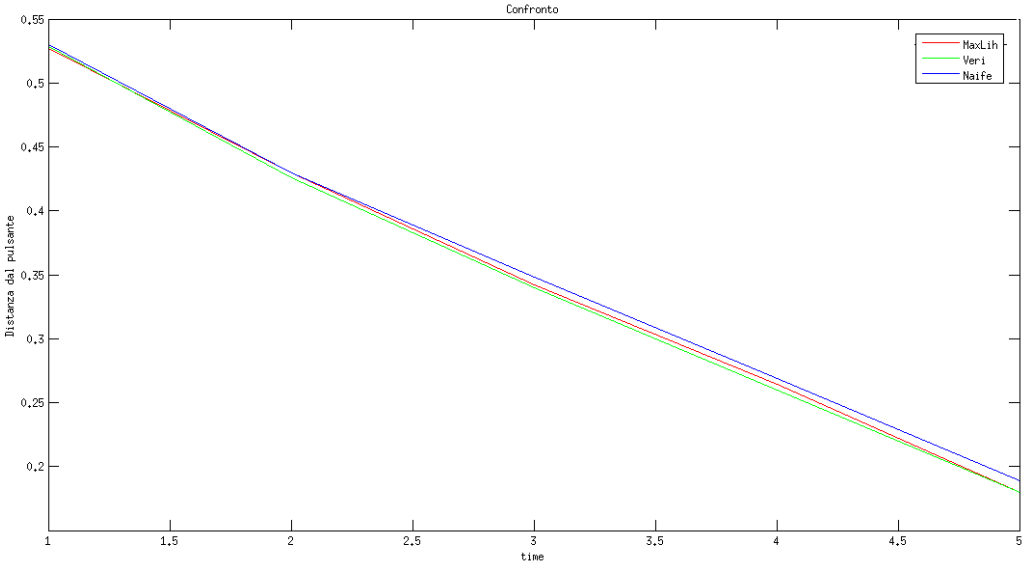
metodo, porti ad un errore massimo di qualche mm.

5.0.7 Confronto

In figura 5.5 mostriamo i confronti tra i due metodi per diversi set di dati. Come possiamo vedere in figura 5.5 il metodo della massima verosimiglianza riesce a stimare meglio, nel lungo periodo, la posizione corretta della camera ottenendo un errore pari a 0.003 m.



(a) $confronto\_set1$



(b) $confronto\_set2$

Figura 5.5: Confronto tra i due metodi con la distanza effettiva

Capitolo 6

Conclusioni

Come visto nel capitolo 5 entrambi i metodi comportano un minimo errore di stima della posizione. Questo errore di può essere causato da:

- nella fase di inizializzazione di Ptam. Non si raggiungono abbastanza features per creare una griglia stabile
- nello spostamento per la stima della scala. Non si raggiungono o si superano i cm richiesti
- nella fase di riproiezione dei punti.
- ambiente circostante.

Per rendere più stabile l'algoritmo si potrebbe utilizzare la dinamica del manipolare e un ulteriore controllo durante il percorso per correggere l'eventuale errore di posa.

Appendice A

Pseudo Codice

A.0.8 Fase di plan

Algorithm 1 Plan

1: $Scena \leftarrow TrovaContorni(p^{2D})$
2: $PerogniContorno \leftarrow CalcolaCentroDiMassa$
3: **if** $(PosizioneCentroDiMassa - PosizioneBottonePremuto < Soglia$ **then** ▷ Trovato pulsante desiderato
4: $Scena \leftarrow RitagliareImmagine(PosizioneBottonePremuto)$
5: $ScenaTagliata \leftarrow Sift$
6: **else** ▷ Premere nuovamente il pulsante e far ripartire la procedura
7: **end if**

A.0.9 Fase di Tracking

Algorithm 2 Tracking

1: $Frame \leftarrow Sift$
2: $Match(FrameEBottonePremuto) \leftarrow Flann$
3: $PerOgniKeypoint \leftarrow CalcoloCentroide$
4: **if** $(PosizioneCentroide - PosizioneBottonePremuto < Soglia$ **then** ▷ Feature valida
5: $FeaturesValide \leftarrow CalcoloCentroide$
6: $BottoneNellaSecondaImmagine = Centroide$
7: **else** ▷ Feature non valida
8: **end if**

A.0.10 Fase di Interazione

Algorithm 3 Interazione

1: $Punti3DFrameMondo \leftarrow Ptam$
2: $Punti3DFrameCamera \leftarrow Trasformo(Punti3DFrameMondo)$
3: $Scala \leftarrow MessaggioDalNodoDiStima(Scala)$
4: $Punti2D \leftarrow Riproietto(Punti3D * Scala)$
5: **if** $(PosizionePunti2D - PosizioneBottonePremuto < Soglia$ **then** ▷ Punti validi
6: **if** $(PuntiValidi > 3$ **then** ▷ Servono 3 punti per fittare il piano
7: $ParametriDelPiano \leftarrow FitPiano(PuntiValidi)$
8: $PosizioneBottone3D \leftarrow ParametriDelPiano$
9: **else** ▷ Impossibile trovare un piano
10: **end if**
11: **else** ▷ Punti non validi
12: **end if**

A.0.11 Stima del fattore di scala

Algorithm 4 Scala

1: $PosizioneRobotPtamAttuale \leftarrow Ptam$
2: $MovimentoRobotReale \leftarrow Robot$
3: $PosizioneRobotPtamPrecedente$ ▷ Salvata ad ogni spostamento
4: $MovimentoRobotPtam \leftarrow PosizioneRobotPtamPrecedente * PosizioneRobotPtamAttuale$
5: **if** $Vect(MovimentoRobotPtam).size() > 1000$ **then** ▷ Se si è preso abbastanza campioni
6: $Scala \leftarrow MassimaVerossimiglianza(MovimentoRobotPtam, MovimentoRobotReale)$
7: $InvioMessaggio \leftarrow scala$
8: **else** ▷ wait
9: **end if**

Bibliografia

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," 2007.

[2] G. E. Forstner W., "A fast operator for detectionand precise location of distinct points, corners and circular features," 1987.

[3] C. H. . M. Stephens, "A combined corner and edge detector," 1988.

[4] D. C. Jacob Engel, *Jürgen* Sturm, "Object recognition from local scale-invariant features," 1999.

[5] D. G. L. Marius Muja, "Fast approximate nearest neighbors with automatic algorithm configuration," 1988.

[6] D. G. Lowe, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," 2013.