Navigazione basata su inseguimento di frecce

Relazione di progetto

Progetto del corso Robotica (principi e progetto)
Università degli Studi di Bergamo
A.A. 2019/2020

Calegari Andrea - 1041183
Paganessi Andrea - 1040658
Piffari Michele - xxxxxxx

December 28, 2019

Contents

List of Figures

1

Stato dell'arte

Obbiettivo: andare a implementare sistema di *visual navigation* per la base robotica in figura 1.1.

1.1 SAL - Stato avanzamento lavori

- Analizzato codice Out-Of-Box del progetto dello scorso anno
  - Il codice preso non aveva main: creato
  - Compresa struttura pub/sub
  - Analizzati topic/nodes pubblicati
- Cambio camera. Perchè? Prestazioni scarse al variare della luce
- Nuova camera –> ueye cam
- Fatta funzionare nuova camera
  - Demo (programma già fornito con la camera)
  - Ros –> utilizzato file debug-launch (inserire caratteristiche che la camera fornisce quando parte lo script).
- Scelta la posizione della camera: verticale inclinata e non orizzontale
- Progetto A.A. utilizza formule vecchie
- Problema CPU consuming (problema intrinseco della camera)
- Memory problem –> risolto con *free*
- Doppie maschere: frecce di due colori
- Aggiunte queste features:
  - Gaussian blur
  - Brightness
  - Erosion - Dilatation
- Fatta erosione solo sulle frecce vicine (quelle nella metà inferiore del frame), mentre invece, le frecce nella metà superiore non vengono erose ma solo dilatate.
- Problema inizializzazione che mostrava rettangoli bianchi su alcune immagini intermedie nelle maschere
- Aggiunta distanza tra centri con tracciamente linea

Figure 1.1: Base robotica addetta alla movimentazione

Figure 1.2: Base verticale sulla quale andare ad inserire la camera

- Prendiamo la freccia più vicina e analizziamo i dati relativi solo a questa freccia: supponendo che tutte le frecce siano uguali, è ovvio che l'area maggiore è quella della freccia più vicina (TODO: da mettere come giustificazione del codice che scriveremo)

Per creare grafi della struttura del codice ROS vedi e comando rqt.

2

Camera

2.1 Scelta della camera

Ad inizio del progetto siamo andati a lavorare con una camera *SpotLight Pro Webcam*, fornita dalla casa *Trust* (fig. 3.1): questa camera abbiamo però visto che non era in grado di dare sufficienti garanzie di funzionamento stabile in alcune delle più comuni condizioni luminose.

Si è deciso quindi di passare ad una camera di tipo industriale, in grado di fornire delle prestazioni più *stabili e affidabili*. La scelta è ricaduta sulla camera della casa produttrice *IDS (Imaging Development System)*: si tratta del modello *UI-1221LE-C-HQ* equipaggiata con la lente *BM2420* della casa *Lensagon*.

2.2 Vericale o orizzontale?

2.3 Come ottenere le immagini dalla camera?

Figure 2.1: Camera utilizzata inizialmente

3

Gestione delle maschere

3.1 HSV

Nella strutturazione del progetto ci è venuto molto naturale andare a lavorare con una scala di colori HSV. Ma perchè non applicare un filtraggio basato su RGB? Come noto nella letteratura, nell'ambito dell'*image recognition* è usuale il problema di andare a mascherare un colore piuttosto che un altro, come nel nostro caso. Potremmo voler trovare, sempre per esempio, oggetti rosso, scannerizzando nell'immagine solamente colori (255,0,0) nella scala RGB; con questo approccio andremmo ad applicare una condizione troppo stringente ai colori. Si potrebbe pensare, come soluzione a questa condizione parecchio stringente, di trovare colori in un range di rossi, come per esempio (130,0,0);(255,0,0): il problema comunque persisterebbe proprio per il fatto che il rosso è ottenuto come combinazione di più colori primari, e non come un solo singolo colore. Potremmo pensare a questo istante di andare a fondo del problema, applicando

So at this point we could continue going down this path, changing the RGB range for all three primary colour values; but it honestly requires a lot of effort to cover all our bases and even if somehow we do manage to get a reasonable range setup it's very likely we'll end up with a lot of noise in the fields we detect.

We need a method that doesn't have to many parameters in order to simplify our detection space. This is where HSV comes into the picture (excuse the pun).

TODO: completare

Lo standard per

3.2 Frecce o cerchi?

3.3 Maschere

3.4 Erosione e dilatazione

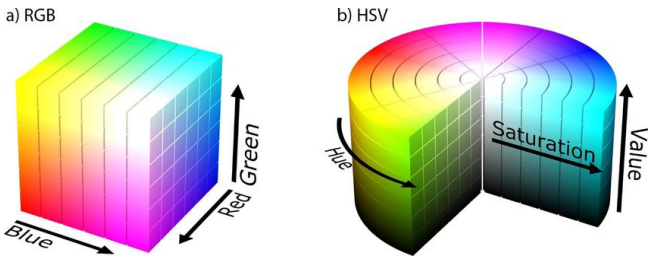

Figure 3.1: RBG vs HSV

4

Identificazione delle forme

4.1 Definizione del problema

IL capitolo precedente ha mostrato come sia possibile ottenere dei contorni, delle figure data una immagine a colori RGB. È ora necessario identificare la forma di ciascuno di questi contorni riconoscendo così i vari quadrati e rettangoli presenti nell'immagine che avessero, una volta acquisiti dalla camera, il colore specificato. Come ultimo passaggio va svolto il controllo per verificare che esista o meno una freccia; quest'ultima altro non è che un triangolo e un rettangolo sufficientemente vicini fra di loro.

4.2 Interpolazione

Per approssimare il contorno ottenuto e filtrato attraverso le mask, come spiegato nel Capitolo 3, è necessario usare una funzione che approssima un poligono con un altro poligono con meno vertici così che la distanza tra di essi sia inferiore ad una certa soglia. Tale funzione è così definita nella libreria OpenCV:

> *void approxPolyDP(InputArray curve, OutputArray approxCurve, double epsilon, bool closed)*

Si è reso necessario effettuare un tuning del parametro *double epsilon* in quanto, per frecce diverse, a distanza variabile e con orientazione non fissa erano ottimali diversi valori. Il valore che più si adattava a tutti i casi presi in considerazione è stato ottenuto sperimentalmente e corrisponde a *double epsilon=0.045*.

La funzione di cui sopra restituisce quindi una lista di poligoni ognuno dei quali ha una lista dei propri vertici. Il passo successivo è stato cercare nella lista dei poligoni un poligono che avesse 4 lati nel caso di un rettangolo e 3 in quello di un triangolo:

```
1 IF (result->total >= 3  && result->total <= 3 &&
2 fabs(cvContourArea(result, CV_WHOLE_SEQ))>lower_area_triang)
```

```
1 IF(result->total >= 4  && result->total <= 6 &&
2 fabs(cvContourArea(result, CV_WHOLE_SEQ))>lower_area_rect)
```

Sempre per via sperimentale è stato possibile scoprire che vincolando il poligono che approssima il quadrilatero cercato ad avere tra i 4 e i 6 lati, la probabilità di riconoscere correttamente un quadrato aumentava.Per il triangolo questo non si è reso necessario vista già gli ottimi risultati con la ricerca vincolata a 3 lati.

Ottenuti ora tutti i quadrati e i rettangoli sufficientemente grandi nella figura va affrontato il problema del riconoscimento di ogni freccia presente nel seguente modo:

- per ciascun rettangolo identificato, si calcola la distanza che intercorre tra esso e ogni triangolo riconosciuto. Per calcolare la distanza tra due figure è necessario calcolarne il centro prima:
  - calcolo il centro del rettangolo

  - calcolo il baricentro del triangolo
  - caclolo la distanza cartesiana tra i due punto appena individuati

- si tiene in considerazione solamente la distanza minore calcolata.
- si confronta suddetto valore con un valore di soglia sperimentale; se questo è minore allora si può assumere che il triangolo e il quadrato presi in considerazione siano una freccia.
- la freccia appena rilevata viene aggiunta alla lista di frecce rilevate nell'immagine.

Per ogni freccia, che ora latro non è che una coppia di punti,

$$\begin{aligned} C_{triangolo} &= (x_t, y_t) \\ C_{rettangolo} &= (x_r, y_r) \end{aligned} \tag{4.1}$$

vanno identificati nell'ordine:

- il centro della freccia, ottenuto come il punto medio del segmento che collega i due centri che definivano la freccia precedentemente.

$$C_{freccia} = (\frac{x_t + x_r}{2}, \frac{y_t + y_r}{2})$$

- L'inclinazione della freccia nel piano, calcolata come:

$$\begin{aligned} \phi = atan(\frac{\Delta y}{\Delta x}), dove \\ \Delta y = y_t - y_r \\ \Delta x = x_t - x_r \end{aligned} \tag{4.2}$$

- L'area dell'oggetto freccia, ricavata come somma dell'area del triangolo e del quadrato.

5

Dalle pixel coordinates alle world coordinates

5.1 Definizione del problema

Nel capitolo precedente è stato illustrato un metodo atto all'identificazione delle frecce che rientrano nel campo visivo della camera. Viene ora trattato come sia possibile ottenere la posizione della freccia, precedentemente ottenuta, nelle coordinate 3D UVW rispetto alla base del robot.

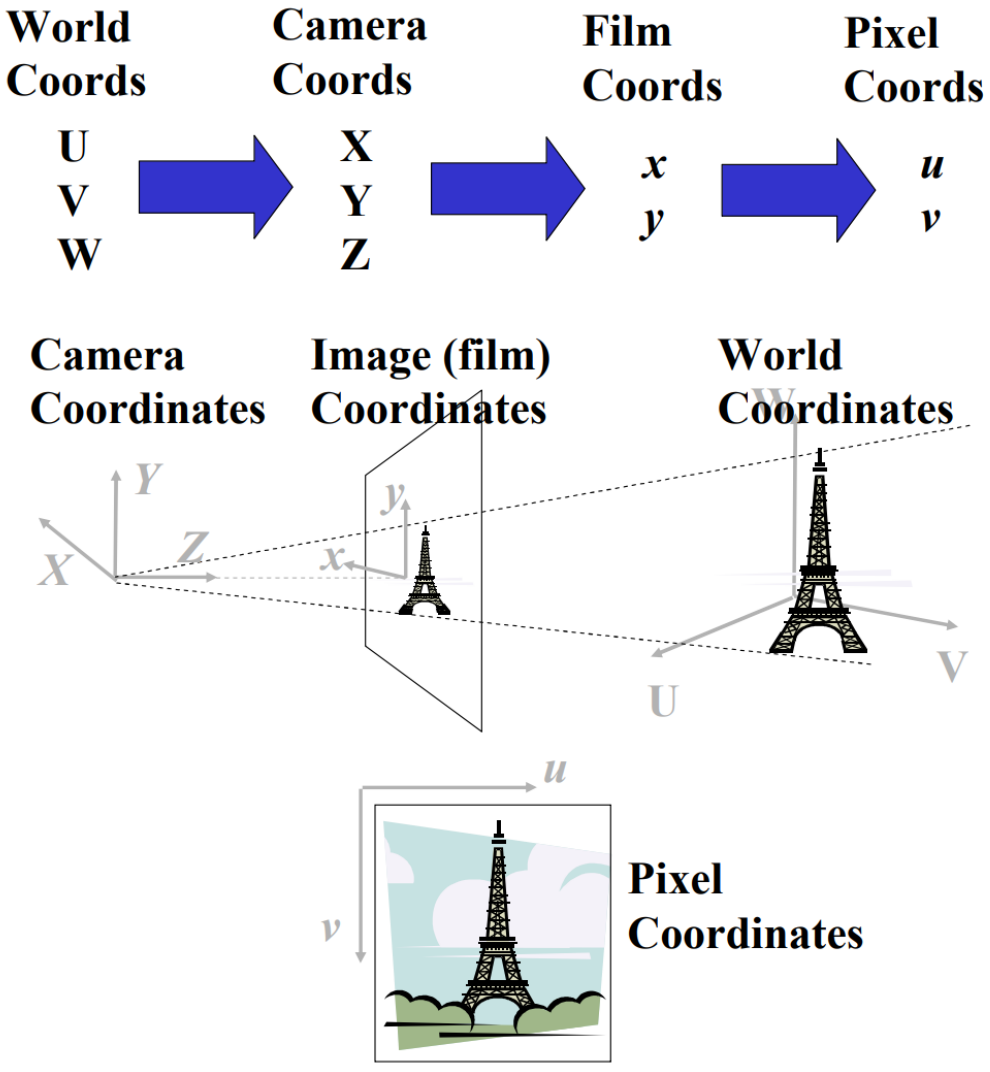

Figure 5.1: schema concettuale delle diverse coordinate

Come si vede in Fig.5.1 si devono effettuare tre trasformazioni per ottenere a partire dalle world coordinates le pixel coordinates. Il problema, nel caso specifico preso in considerazione, risulta essere l'opposto: dalle coordinate nella camera è necessario ottenere le coordinate globali dell'oggetto effettuando una trasformazione inversa. Si analizzeranno ora le singole trasformazioni che permetteranno alla fine di ottenere una trasformazione globale.

5.2 Da wolrd coordinates to camera coordinates

Figure 5.2: posizione del world frame

Partendo da un sistema di riferimento solidale al robot e all'altezza del pavimento, che identifichiamo come sistema globale, è possibile definire una matrice di rototraslazione per ottenere il sistema di riferimento solidale al centro della camera.

$$R_{worldcam} = R_{traslazione} * R_{rotazione} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h_{cam} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} cos(\frac{\pi}{2}+\alpha) & -sin(\frac{\pi}{2}+\alpha) & 0 & 0 \\ sin(\frac{\pi}{2}+\alpha) & cos(\frac{\pi}{2}+\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

È stata effettuata una traslazione lungo l'asse W in quanto la camera è posta esattamente sopra all'origine del sistema O_{UVW} ed una rotazione rispetto all'asse U di $\frac{\pi}{2}$ in quanto, per convenzione, si associa all'asse delle Z la profondità nel frame solidale alla camera. A questo punto è necessario effettuare un'altra rotazione di α gradi rispetto all'asse X a seconda dell'inclinazione alla quale si sceglie di far lavorare la camera INSERISCI LA FOTO DELLA CAMERA SUL PALO PER FAR CAPIRE IL CONCETTO!!!!!!!!!!!!!!!!!!!!!!!!!!!!

5.3 Da camera coordinates a film coordinates

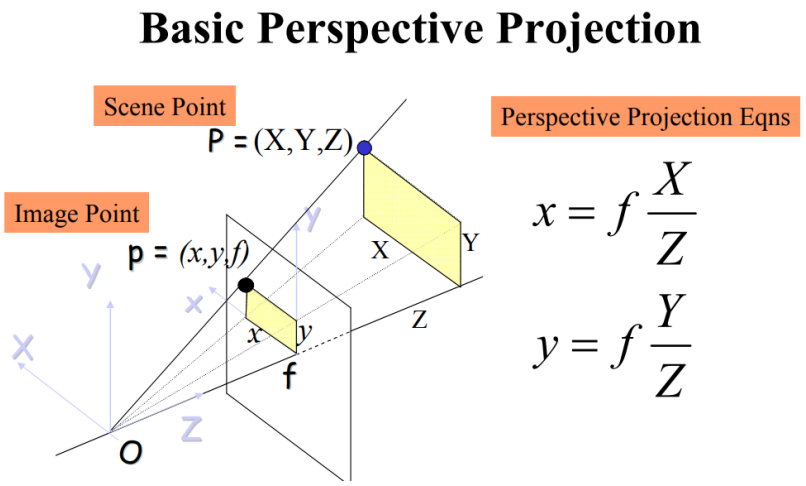

Figure 5.3: descrizione del problema

In Fig.5.3 f rappresenta il fuoco della camera, parametro ottenibile attraverso una procedura di calibrazione.

5.4 Da film coordinates a pixel coordinates

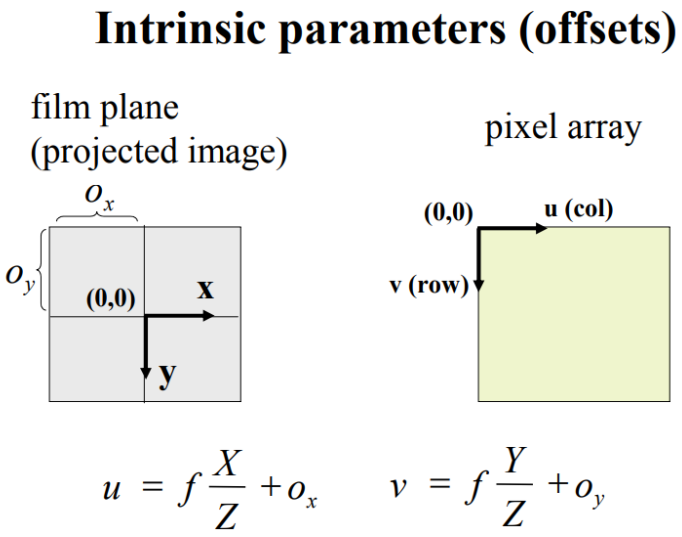

Figure 5.4: descrizione dell'ultima trasformazione

In Fig.5.4 O_x e O_y sono anch'esse ricavabili tramite la procedura di calibrazione della camera

5.5 Problema al rovescio

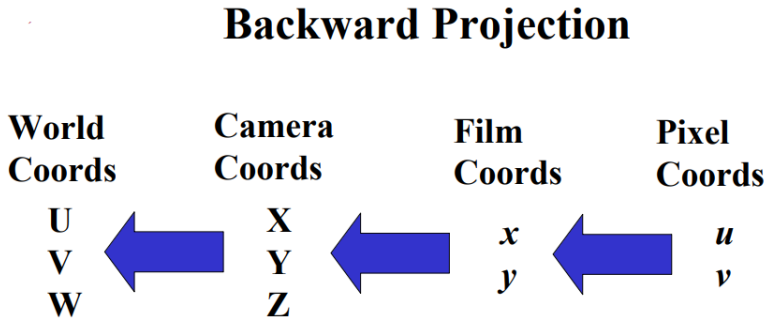

Figure 5.5: Problema inverso

Come si vede in Fig.5.5 si deve ora affrontare il processo inverso siccome, nel caso trattato, si hanno a disposizione u e v e si vogliono ottenere U, V, W. Come già detto prima O_x e O_y e f sono ottenibili tramite calibrazione;dunque:

$$\begin{aligned} x &= u - O_x \\ y &= v - O_y \end{aligned} \tag{5.1}$$

Sono state ottenute le equazione del film coordinates. È ora necessario ottenere le camera coordinates. Per fare ciò è necessario conoscere il valore di Z, cosa ottenibile in due modi:

- utilizzando una camera con sensore di profondità
- assumendo che gli oggetti siano sempre posti su un piano di cui si conosce l'equazione.

La seconda assunzione è, nella realtà dei fatti, una ipotesi corretta in quanto gli oggetti e le frecce giaceranno sempre sul pavimento. È quindi richiesto di calcolare l'equazione del pavimento nel camera frame:

- $$z = 0 \tag{5.2}$$

  rappresenta l'equazione del piano se fosse nel world frame

- $$z * R_{worldcam} = 0 \tag{5.3}$$

  il piano così calcolato è la descrizione dal punto di vista matematico del pavimento dal punto di vista della camera

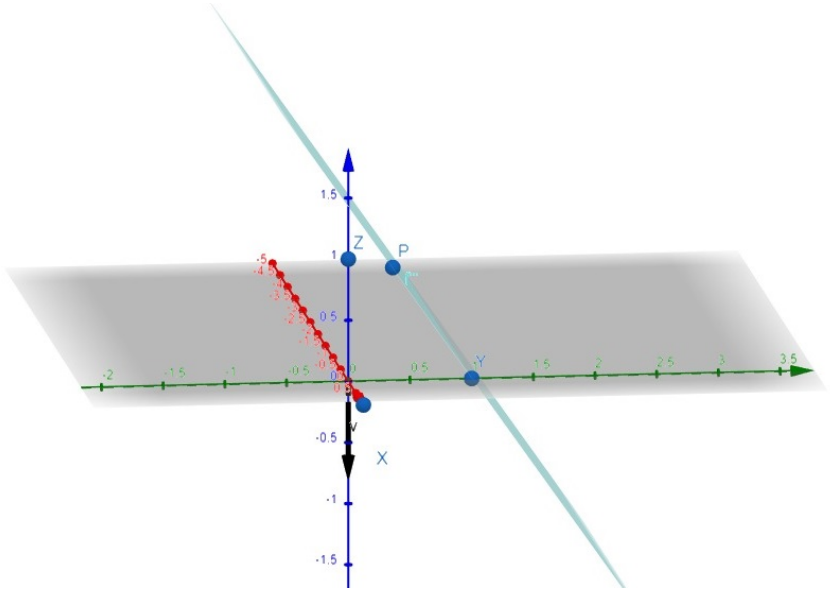

Figure 5.6: piano del pavimento nel camera frame

- si può dunque calcolare il fattore di scala s dato un generico piano $ax + by + cz + d = 0$

$$\frac{-d}{aX' + bY' + c} \tag{5.4}$$

dove

$$\begin{aligned} X' &= \frac{x}{f} = \frac{X}{Z} \\ Y' &= \frac{y}{f} = \frac{Y}{Z} \end{aligned} \tag{5.5}$$

che sono le coordinate normalizzate rispetto a Z.

- Dunque, come ultimo passaggio si moltiplica tutto per il fattore di scala:

$$\begin{aligned} X &= X' * s \\ Y &= Y' * s \\ Z &= S \end{aligned} \tag{5.6}$$

Per ottenere le equazione nel world frame si deve dunque utilizzare la matrice inversa:

$$\begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} = R^{-1}_{worldcam} * \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Bibliography

[1] *Descrizione della camera* https://en.ids-imaging.com/store/ui-1221le-rev-2.html

[2] *Manuale della camera* https://en.ids-imaging.com/IDS/datasheet_pdf.php?sku=AB02422

[3] *Manuale della lente* https://www.lensation.de/product/BM2420/

[4] *Ueye cam e ROS* http://wiki.ros.org/ueye

[5] *HSV vs RGB* https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c

[6] *HSV vs RGB* https://handmap.github.io/hsv-vs-rgb/

[7] *Camera Projection I* http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf

[8] *Camera Projection II* http://www.cse.psu.edu/~rtc12/CSE486/lecture13.pdf

[9] *Coordinate omogenee* http://robotics.unibg.it/teaching/robotics/pdf/14_Geometria3D.pdf