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 25, 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. 5.1): 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

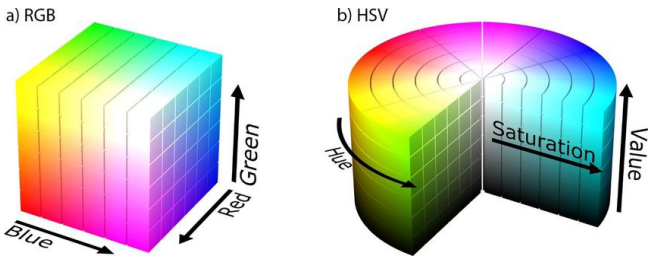

Figure 3.1: RBG vs HSV

4

Identificazione delle forme

5

Dalle camera coordinates alle world coordinates

5.1 Scelta della camera

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 XYZ rispetto alla base del robot.

5.2 Vericale o orizzontale?

5.3 Come ottenere le immagini dalla camera?

Figure 5.1: Camera utilizzata inizialmente

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