



Università degli Studi di Bergamo

SCUOLA DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Laboratorio di Robotica

Documentazione attività di laboratorio

Prof.
Davide Brugali

Candidati
Giulia Allievi
Matricola 1058231

Martina Fanton
Matricola 1059640

Indice

1	Introduzione	2
2	Librerie	3
2.1	Grasp Pose Generator, GPG	3
2.2	Grasp Pose Detector, GPD	3
3	Integrazione	4
3.1	Library	4
3.2	Plugin	5
3.3	Component	5
3.4	Messages	5
4	Risultati	6

Introduzione

L'obiettivo del nostro progetto è quello di fornire le coordinate e l'orientamento della pinza ad un robot, al fine di potersi posizionare per afferrare un oggetto. Le coordinate e l'orientamento vengono calcolate da un'opportuna libreria (nel nostro caso dalla libreria *Grasp Pose Generator*, *GPG*), la quale necessita delle informazioni dell'oggetto inquadrato sottoforma di una nuvola di punti, la point cloud, che verrà fornita attraverso la *Stereo Camera ZED*. Il software del progetto sarà realizzato utilizzando il *Framework STAR*.

La Stereo Camera ZED è dotata di vari sensori, per esempio l'accelerometro e il giroscopio, ma nella nostra applicazione serviranno solo i dati delle immagini, in particolare solo quelli della point cloud. La stereocamera ha un range di profondità compreso fra 30 cm e 20 m (fonte: [datasheet](#)), perciò gli oggetti dovranno essere inquadrati ad una distanza compresa in questo intervallo.

Di seguito, nel capitolo 2 verranno analizzate due diverse librerie che si possono utilizzare per ottenere i dati relativi alla posizione e all'orientamento della pinza del robot, mentre nel capitolo 3 verranno descritti i passi necessari per integrare, compilare ed eseguire il progetto. Infine, nel capitolo 4, verranno mostrati i risultati finali.

Librerie

Prima di costruire l'applicazione finale, sono state analizzate due diverse librerie *stand-alone*, la *Grasp Pose Generator*, GPG, la cui documentazione è reperibile al seguente [link](#), e la *Grasp Pose Detector*, GPD, la cui documentazione è invece disponibile al seguente [link](#).

2.1 Grasp Pose Generator, GPG

2.2 Grasp Pose Detector, GPD

Per entrambe, riportare i comandi di compilazione ed esecuzione per ottenere la versione stand alone. Mettere i link, spiegare GPG e per GPD descrivere solo le differenze con la libreria precedente. Sempre per entrambe, alla fine mettere un'immagine con la visualizzazione delle prese e delle info date dal terminale.

Integrazione

In questo capitolo, verranno descritti i passi seguiti per ottenere l'applicazione finale.

3.1 Library

Per prima cosa, andiamo a copiare e compilare la libreria GPG descritta nella sezione precedente nella cartella [guarda percorso] della cartella *Library*. Per fare ciò eseguiamo i seguenti comandi in un terminale:

```
cd <location_of_your_workspace>
cd grasp_candidates_generator
mkdir build && cd build
cmake ..
make
sudo make install
```

Per eseguire la libreria, digitare il comando seguente all'interno della cartella *build* e sostituire *some_cloud* con il nome del file contenente la point cloud:

```
./generate_candidates ../cfg/params.cfg ~/gpg/some_cloud.pcd
```

La libreria appena importata funziona fornendogli in ingresso un file con estensione .pcd che contiene le informazioni sui punti della point cloud, però nel nostro caso la point cloud non sarà salvata su un file, ma verrà fornita dalla stereocamera. Modifichiamo quindi la libreria per fare in modo che riceva in ingresso non un file, ma una struttura dati denominata *Cloudcamera* che contiene un puntatore alla struttura dati *pcl::PointCloud<pcl::PointXYZRGBA>::Ptr*. Quest'ultima variabile è un vettore contenente dei punti nel formato XYZRGBA: ogni punto è caratterizzato da 4 campi, i primi tre (X, Y, e Z) sono di tipo double e contengono le informazioni relative alle coordinate di quel punto in metri, invece l'ultimo campo, RGBA, è un intero senza segno a 32 bit in cui si codifica l'informazione sul colore, ogni parametro del colore (R, G, B e A) è memorizzato in 8 bit, questo significa che ognuno di questi valori può variare da 0 a 255, perciò possono essere rappresentati fino a $2^{32} = 4.294.967.296$ colori. Nello specifico, gli 8 bit meno significativi contengono l'informazione relativa al colore blu (B), i bit da 9 a 16 contengono l'informazione del colore verde (G), i bit da 17 a 24 codificano il valore del rosso (R), infine i bit da 25 a 32 contengono l'informazione del parametro alfa (A), che indica la

trasparenza del colore. Nella nostra applicazione non utilizzeremo quest'informazione perché la point cloud è formata solo da punti di colore nero, che viene codificato dal valore più basso, ovvero 0.

La seconda modifica che apportiamo riguarda il tipo restituito dalla funzione [copia nome], perché non sarà di tipo *Grasppose* ma *void*, il vettore *Grasppose* viene aggiornato passando alla funzione l'indirizzo della variabile da aggiornare. [da continuare]

Dopo aver modificato la libreria, è necessario ricompilarla andando ad eseguire l'ultimo comando della sequenza di compilazione illustrata in precedenza, che è:

```
sudo make install
```

È estremamente importante ripetere questo passaggio ogni volta che si desidera apportare delle modifiche alla libreria.

3.2 Plugin

nella sottocartella *plugin* della cartella *Functionality* di STAR. TODO
mettere parametri file cfg che è utile conoscere (numsample e dim pinze)

sistemare
dettagli im-
portazione,
come make-
file, elimina-
zione main
ecc...

3.3 Component

Abbiamo creato due diversi componenti, il componente che si occupa dell'utilizzo della libreria e della pubblicazione dei messaggi contenenti le informazioni sulla presa è *ObjectDetection*, mentre il componente *ImageVisualization* si limita a ricevere e stampare sul terminale il messaggio ricevuto dal primo componente.

3.4 Messages

3 messaggi creati con vari campi.

Risultati

Immagini con prese, eventualmente anche quelle con integrazione robot.