05/12/2016

Déjà fait : détection des plans (la frame du plan le plus proche est retournée) + extraction des éléments situés au dessus du plan (en donnant des paramètres > exemple : on cherche un objet situé a 20 cm au dessus du plan) => le but est d’améliorer le système de détection (pour l’instant utilisation de la forme) => réfléchir à différentes possibilités (utilisation des couleur) + remplir la base de donnée (=phase d’apprentissage)

Ensuite, lorsque l’objet est détecté et reconnu, alors une frame liée à l’objet doit permettre de le localiser par rapport à la frame de la table/plan => grip

A corriger : erreur code > gérer le cas ou aucun plan n’est trouvé (test pointeur != null pas fait …)

Detection objet ok (cluster)mais pas l’équivalent RGB couleur

06/12/2016

1. Object extraction : sous quelle forme (format de donnée) sont stockés les objets extraits
2. Node tabletopfilter : fonctionnement
3. Node : Object\_recognition\_server
4. Node : object\_discrimination
5. Node : tabletransformation
6. Node : Objectextract

Object extraction = extrait l’information correspondant à un objet et renvoi des cluster et du rgb

Src > filesystemedao.cpp > methode utilisé pour comparaison

Remplacer info.txt par un JSON (dans dossier db)

Modifier db : phase d’apprentissage + discrimination (avec méthodes)

07/12/2016

* Cluster = ensemble de points représentant un objet évoluant de manière dynamique =segmentation d’objet
* Dans notre projet : hypothèse > objets pas collé
* Objectif : rendre l’ensemble des briques du projet courant robuste
* Faire plusieurs essais/demo pour comprendee comment les briques marchent
* Optimisation code
* Etapes reconnaissance (variation pour apprentissage=storage dans bdd) : extraction plans > extraction plan le plus proche > extraction objets sur ce plan > cluster > balayage bdd avec cluster > extraction image rgb avec coordonnée > methode de matching (surf …)

13/12/2016

* Trouver des outils pour accélérer la discrimination
* Multithreading pour optimiser calculs
* (apres lancement de tous les launch) Service /pcl\_rgb\_extract > renvoi un tableau de cluster ainsi que sa position et autres informations
* Node : Objectdiscrimination > récupère données observées et données bdd pour faire la comparaison
* Experience : 1st : training = service call /object\_manager\_srv train

2nd : check = service call /object\_manager\_srv check

* Creation d’un catkin > faire git clone du projet

A FAIRE

* Catkin sur pc ok > recupération git ok > catkin make du nouveau catkin KO > prblm inclusion fichier / organisation des dossiers

03/01/2017

Gitlab : astrostudent astro4student

1ere erreur remarquée : étapes 10 et 11 du quickstart\_guide KO => seg fault

Modifications(code) :

1. Src / object\_discrimination/objectdiscrimination.cpp
2. Si tableau cluster vide -> throw (done) à faire éventuellement le catch

04/01/2017

Objectif update : créer plusieurs nœuds à part qui permettent de faire de la détection d’objet :

1. Etape 1 : 1) on donne un label 2) le prgramme retrieve tous les élements de la db qui ont se label (penser à utiliser catégorie) 3) pour chaque éléments retrieved de la db > comparer avec tous les élements détectés sur la tabletop pour voir si l’objet est présent sur la table
2. Etape 2 : 1) on demande au robot ce qu’il voit sur la table 2) il compare les éléments retrieved sur la table avec les éléments de la base de donnée 3) si correspondance > OK !