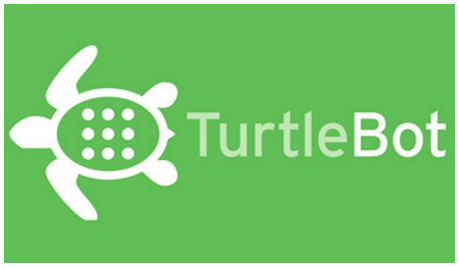
ARGS
-
Autonomous Robots Gaming System



1. Description du projet

ARGS est un système permettant à deux robots de faire une partie de morpion en réalité augmenté. Un des robots joue pour les deux joueurs en ce déplaçant sur la case jouée, pendant que l'autre, fixe, filme le terrain de jeu. Ainsi on pourra voir le terrain sur l'écran du robot fixe en réalité augmenté. On vous présentera ici le robot ce déplaçant.

2. Architecture du système

Le système, en plus de ROS, est composé de trois node supplémentaires. A savoir, "Grille_Jeu" permettant de créer le terrain de jeu en fonction de la position du repère (cf partie 3), du robot et de la taille des cases ; "Navigation" s'occupant du déplacement du turtlebot en fonction des cases jouées ; et "Jeu_IA" ayant en sortie contenant les cases à jouées.

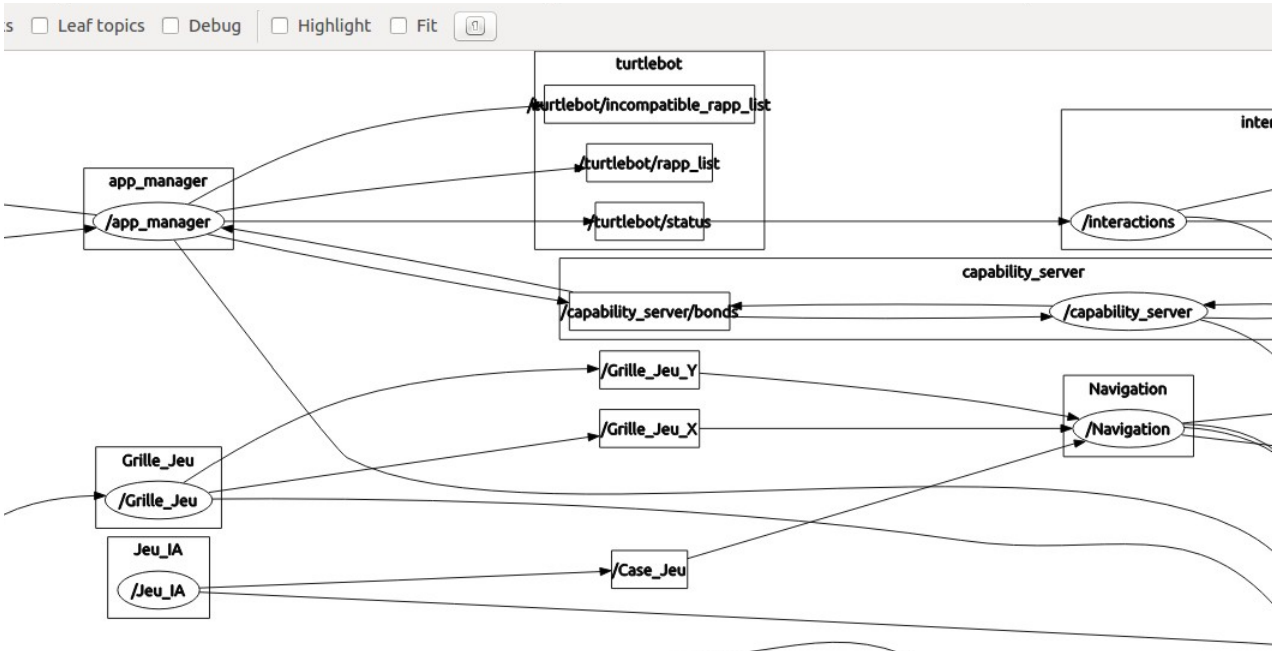


Illustration 1: Lien entre les nodes ajoutés

3. Repère et position du robot

Le repère permettant de créer le terrain virtuel est un damier (voir Illustration 2). On repère celui-ci en utilisant la librairie « *find_2d_objects* », qui permet de sélectionner manuellement un objet et de récupérer sa position dans un plan 3D. De cette façon, en fixant la position du robot à 0, on peut créer les positions de case de jeu. Pour cela, on ajoute à la position du repère une valeur sur l'axe X et Y (le plan du sol) différent pour les 9 cases.

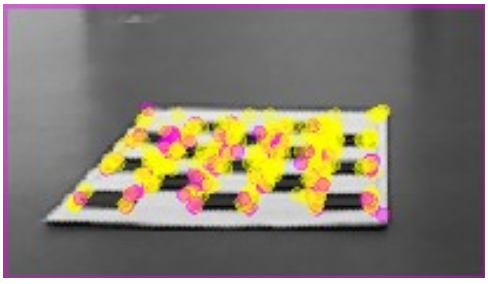


Illustration 2: Repère du terrain virtuel

C1	C2	C3
B1	B2	B3
A1	A2	A3

Ainsi le terrain de jeu (voir ci dessus) est créer à partir du coin en bas à gauche. Les cases sont ainsi d'une taille de 45 cm, ce qui correspond au diamètre du robot, à savoir 30 cm, auquel on rajoute 7,5 cm de marge.

A l'initialisation on publish sur le topic /InitialPose des zéros, pour initialiser la pose du robot (poistion et orientation) face au repère. De cette façon le turtlebot peut jouer sans connaître la carte au préalable.

Ensuite, l'utilisateur doit sélectionner le damier sur l'interface graphique du package find_object_2d (voir le README en annexe). De cette façon le damier sera positionné sur la carte.

Initialpose : http://docs.ros.org/api/geometry_msgs/html/msg/PoseWithCovarianceStamped.html

File : geometry_msgs/PoseWithCovarianceStamped.msg

Contenu :
```
std_msgs/Header header
        uint32 seq                  // Consecutively increasing ID
        time stamp                  // Timestamp
        string frame_id             // 0 : no frame ; 1 : global frame
geometry_msgs/PoseWithCovariance pose
        geometry_msgs/Pose pose
                geometry_msgs/Point position
                        float64 x
```

float64 y
float64 z
geometry_msgs/Quaternion orientation
float64 x
float64 y
float64 z
float64 w
float64[36] covariance

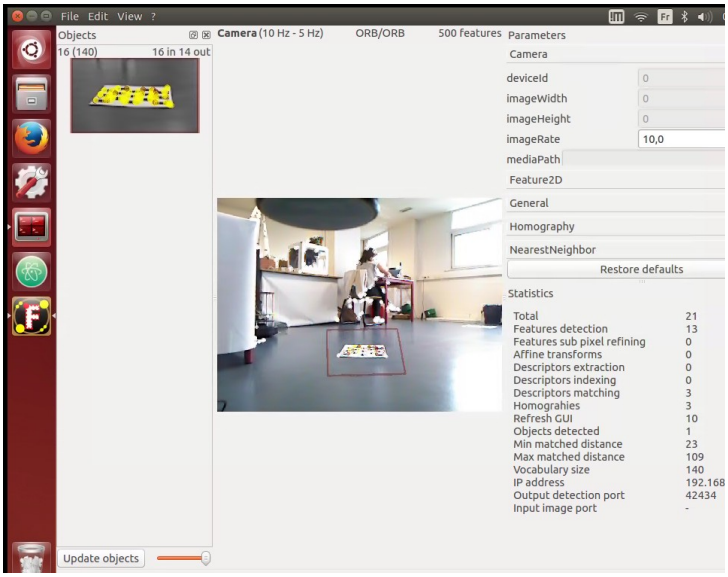


Illustration 3: Interface graphique find_object_2d

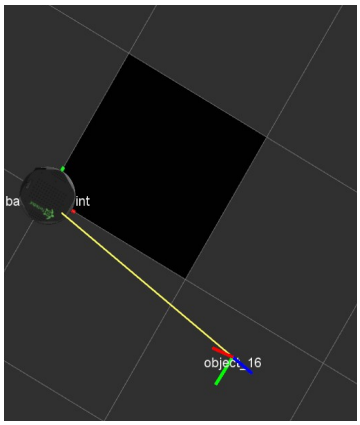


Illustration 4: Détail de Rviz

Dans l'illustration 4, on peut voir un détail de l'interface Rviz, qui permet de visualiser le robot dans l'espace et sa vision par exemple. Ici, on peut voir le repere de l'objet et celle du robot. Toujours dans cette interface on peut visualiser les topics de position.

4. Jeu de morpion automatique

Le jeu de morpion a été créer à partir d'un code trouvé sur github permettant à un utilisateur de jouer contre l'ordinateur. Il a ensuite été modifié pour faire jouer l'ordinateur en autonomie compléte.
Le jeu est réalisé entièrement, les cases jouées stockées dans un tableau qui est ensuite transmis au node s'occupant du déplacement, facilitant ainsi la communication entre les deux nodes, puisque qu'un seul message est transmis, et la vérification se faisant directement dans le node de déplacement (via /move_bas_simple/goal).

5. Déplacement

Pour réaliser le déplacement du turtlebot, chaque case de jeu correspond à une coordonné. On publish ainsi ces coordonnées sur le topic /move_base_simple/goal. Le robot sera ainsi mobile et peut s'adapter à son evironnement. En effet si un obstacle surgit, le robot essayera de contourner celui-ci pour atteindre son objectif.

Move_base_simple/goal : http://docs.ros.org/api/geometry_msgs/html/msg/PoseStamped.html

File: geometry_msgs/PoseStamped.msg

Contenu :
```
std_msgs/Header header
        uint32 seq                  // Consecutively increasing ID
        time stamp                  // Timestamp
        string frame_id             // 0 : no frame ; 1 : global frame
geometry_msgs/Pose pose
        geometry_msgs/Point position
                float64 x
                float64 y
                float64 z
        geometry_msgs/Quaternion orientation
                float64 x
                float64 y
                float64 z
                float64 w
```

6. Launchfile

Un launchfile permet de lancer plusieurs node à la suite en une seule ligne de commande.
Ici, le launchfile appele "JeuMorpion.launch" permet de lancer la création de trois nodes (Navigation, Grille_Jeu, Jeu_IA).

7. Problème

Un des premiers problème a été la connection avec le robot, en effet de nombreuses déconnections ont été constaté. Des changements d'IP en ont été la cause principale.

Il a été utilisé un topic debug de pour trouver et résoudre les erreurs. Il reste un probleme de synchronisation entre les différents nodes.

Annexe 1 : README

Launch :

- Environnement sur le PC :
  - *export ROS_MASTER_URI=http://IP_ROBOT:PORT_ROBOT*
  - *export ROS_HOSTNAME=IP_PC*

- Environnement sur le robot :
  - *export ROS_MASTER_URI=http://IP_ROBOT:PORT_ROBOT*
  - *export ROS_HOSTNAME=IP_ROBOT*

Démarrage :

- Placer le robot face au repère, à côté de la caméra filmant le terrain de jeu
- Sur le robot :
  - *roslaunch turtlebot_bringup minimal.launch*
  - *roslaunch turtlebot_bringup 3dsensor.launch depth_registered:=true*
  - *roslaunch find_object_2d find_object_3d.launch* //Interface de reconnaissance d'objet
  - Edit/« Add object from scene »/« Take picture »
  - Choisir « Select keypoints » au lieu de « Select region », puis sélectionner le repère avec un cliquer-glisser
  - Next/End
- Sur le PC :
  - *roslaunch turtlebot_rviz_launchers view_navigation.launch –screen*
  - *roslaunch JeuMorpion.launch*

Références :

- Librarie find_object_2d :
  - https://github.com/introlab/find-object
  - http://wiki.ros.org/find_object_2d

- Tutoriel ROS :
  - http://wiki.ros.org/fr/ROS/Tutorials

- Jeu de morpion :
  - https://gist.github.com/rpip/5608979

- Package :
  - https://github.com/Inika29/ARGS