TP évalué

Intégration de Capteurs pour la robotique

ISIMA

2017–2018

Ce TP sera évalué et doit donc être rendu.

Chaque groupe doit envoyer les noms et prénoms de ses membres ainsi que l'url d'un dépôt *git* public contenant le *package* `sensor_training_result` conçu lors de ce TP à l'adresse mail `gerald.lelong@easymov.fr` avant le timestamp unix `1513002600`.

Description

Le but de ce TP est de simuler un dispositif semblable à une caméra 3D. Il s'agira de la combinaison d'un caméra monoculaire classique et d'un lidar plan tournant sur lui-même. Notre système publiera, au final, un nuage de points 3D en couleur grâce à la combinaison des informations des deux capteurs.

1. **Lancez le fichier de lancement `simple_turtlebot.launch` du *package* `simple_turtlebot`.**
   Les fichiers de lancement se lancent avec la commande `roslaunch`.
   La fenêtre qui s'ouvrira est le simulateur **Gazebo** dans lequel vous pouvez visualiser le robot simulé dans son environnement (actuellement vide).

2. **Créez un nouveau *package* ROS dans le *workspace* fournit.**
   Un *package* se crée avec la commande `catkin create pkg`.
   Le *package* doit s'appeler, dans le cadre de ce TP, `sensor_training_result`.

3. **Créez un fichier de lancement dans votre *package*.**
   Le fichier de lancement doit s'appeler, dans le cadre de ce TP, `default.launch`.

4. **Incluez `simple_turtlebot.launch` du *package* `simple_turtlebot` dans votre fichier de lancement.**
   Il s'agit d'utiliser la balise `include`.

5. **Spécifiez l'argument `world` de `simple_turtlebot.launch` pour simuler votre robot dans un environnement conçu par vos soins.**
   Il s'agit d'y préciser le chemin d'un fichier `.world`.

6. **Copiez le fichier `simple_turtlebot.urdf.xacro` dans le dossier `urdf` du package `simple_turtlebot` vers un dossier `urdf` situé à la racine de votre *package*.**
   C'est ce fichier que vous modifierez pour ajouter des capteurs au Turtlebot simulé.

7. **Spécifiez l'argument `xacro_file` de `simple_turtlebot.launch` pour charger le fichier *URDF* de votre *package*.**

8. **Utilisez la documentation disponible à l'adresse ci-dessous pour modifier votre fichier URDF de manière à ajouter une caméra (monoculaire) au Turtlebot.**
   `https://goo.gl/HqKmqd`

9. **Incluez le fichier `spinning_joint.urdf.xacro` du *package* `sensor_training` dans votre propre fichier *URDF*.**
   Il s'agit d'utiliser la balise `xacro:include`.
   Ce fichier va ajouter un membre (*link*) nommé `laser_link` représenté sous la forme d'un cube ainsi qu'une articulation (*joint*) capable de tourner de manière continue sur votre robot.

10. **Utilisez la documentation disponible à l'adresse ci-dessous pour modifier votre fichier URDF de manière à transformer le membre `laser_link` en un lidar (laser).**
    `https://goo.gl/HqKmqd`
    Vous devez choisir la version utilisant le CPU et non le GPU de votre machine.

11. **Incluez `control.launch` du *package* `sensor_training` dans votre fichier de lancement.**
    Il s'agit d'utiliser la balise `include`.
    Ce fichier permet de lancer les interfaces de contrôle de l'articulation rotative à laquelle votre lidar est attaché.

12. **Créez un noeud ROS mettant en rotation le lidar à une vitesse donnée.**
    L'inclusion du fichier `control.launch`, à l'étape précédente, a dû ajouter le *topic* `laser_velocity_controller/command` auquel *Gazebo* souscrit. Votre noeud doit y publier une commande en vitesse pour l'articulation du lidar.

13. **Ajoutez le lancement du noeud créé à l'étape précédente à votre fichier de lancement.**

14. **Ajoutez le lancement du noeud `camera_lidar.py` fournit par le *package* `sensor_training` à votre fichier de lancement.**
    Ce noeud écoute des données lidar et caméra. Pour chaque message lidar, il récupère la couleur de chacun des points 3D par projection dans l'espace caméra et les publie sous la forme d'un nuage de points en couleur.

15. **Connectez les *topic* auxquels `camera_lidar.py` souscrit à vos capteurs.**
    Il s'agit d'utiliser la balise `remap`.

16. **Ajoutez le lancement du noeud `point_cloud_assembler` fournit par le *package* `laser_assembler` à votre fichier de lancement.**
    Ce noeud écoute des nuages de points et les aggrège dans le repère souhaité. Le nuage de points généré n'est pas publié, mais doit être récupéré par l'appel au service `assemble_scans` de ce noeud.
    Le repère est spécifié à l'aide du paramètre `fixed_frame`.
    N'hésitez pas à visiter la documentation de ce noeud !

17. **Connectez le *topic* où `camera_lidar.py` publie ses nuages de points au noeud `laser_assembler`.**
    Il s'agit d'utiliser la balise `remap`.

18. **Créez un noeud qui appelle périodiquement (toutes les 5 secondes par exemple) le service `assemble_scans` du noeud `laser_assembler` et publie le nuage de points aggrégé.**

19. **Ajoutez le lancement du noeud créé à l'étape précédente à votre fichier de lancement.**

20. **Ajoutez *RViz* à votre fichier de lancement de manière à ce qu'il soit lancé avec une configuration permettant de visualiser le nuage de points couleur publié.**
    Il s'agit d'utiliser l'option `-d` du noeud `rviz` permettant de spécifier un fichier de configuration.

Résultat attendu

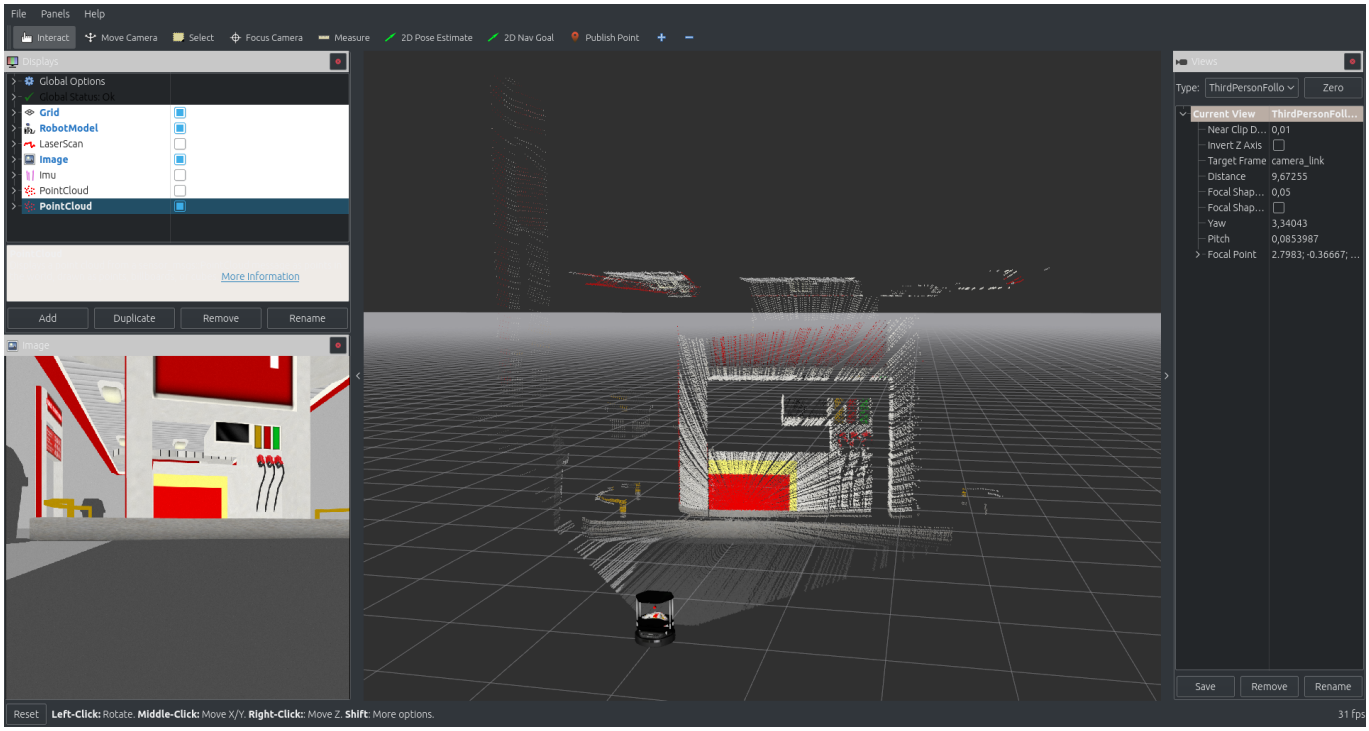

Figure 1 – Exemple de résultat attendu