3D Mapping with depth camera on Ros2 humble

Configuration

Ubuntu 22.04
Ros2 Humble
Motherboard intel i5
Realsense D435i

Realsense D435i

https://github.com/IntelRealSense/realsense-ros

Ubuntu 22.04: ROS2 Humble

Install librerealsense2

https://github.com/IntelRealSense/librealsense2-release

Utilisation D435i

librerealsense2 ~$ realsense-viewer

Démarrer le nœud de la caméra

$ ros2 launch realsense2_camera rs_launch.py

topics disponibles

/camera/color/camera_info
/camera/color/image_raw
/camera/color/metadata
/camera/depth/camera_info
/camera/depth/image_rect_raw
/camera/depth/metadata
/camera/extrinsics/depth_to_color
/camera/imu
/clicked_point
/goal_pose
/initialpose
/parameter_events
/rosout
/tf
/tf_static

Utilisation de paramètres

les filtres temporels et spatiaux sont activés :

$ ros2 run realsense2_camera realsense2_camera_node --ros-args -p enable_color:=false -p spatial_filter.enable:=true -p temporal_filter.enable:=true

avec un fichier de lancement :

$ ros2 launch realsense2_camera rs_launch.py depth_module.profile:=1280x720x30 pointcloud.enable:=true

Activation de l'accélération et du gyroscope

Elle s'effectue soit en ajoutant les paramètres suivants à la ligne de commande :

$ ros2 launch realsense2_camera rs_launch.py pointcloud.enable:=true enable_gyro:=true enable_accel:=true

soit en cours d'exécution à l'aide des commandes suivantes :

$ ros2 param set /camera/camera enable_accel true
$ ros2 param set /camera/camera enable_gyro true

topics avec gyro et accel

/camera/accel/imu_info
/camera/accel/metadata
/camera/accel/sample

/camera/depth/color/points

/camera/extrinsics/depth_to_accel
/camera/extrinsics/depth_to_gyro
/camera/gyro/imu_info
/camera/gyro/metadata
/camera/gyro/sample

Pour obtenir la valeur d'un paramètre (node : /*camera/*camera)

$ ros2 param get /camera/camera base_frame_id

Utilisation de l'IMU de D435i

Installation **$ sudo apt-get install ros-humble-imu-tools**

Lancer le nœud Realsense

$ ros2 launch realsense2_camera rs_launch.py enable_accel:=true enable_gyro:=true unite_imu_method:=1

ou

$ ros2 launch realsense2_camera rs_launch.py **pointcloud.enable:=true** enable_accel:=true enable_gyro:=true unite_imu_method:=1

Exécuter le nœud imu_filter_madgwick

Sus : /imu/data_raw
/imu/mag

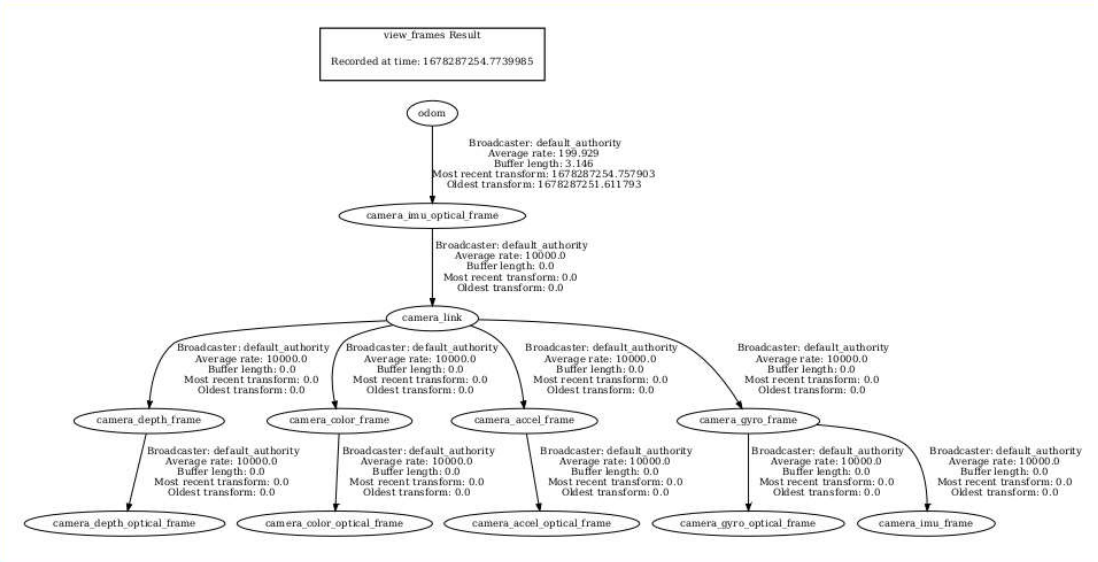
Pub : /imu/data frame_Parent : /odom
frame_Child : /camera_imu_optical_frame

Exécutez le nœud imu_filter_madgwick et remappez le sujet /camera/imu vers /imu/data_raw :**à la place de:imu/data_raw prendre camera/imu**

$ ros2 run imu_filter_madgwick imu_filter_madgwick_node --ros-args -p use_mag:=false -r /imu/data_raw:=/camera/imu

Créer une tf de la frame /camera_imu_optical_frame **à la frame** /camera_link

Dans launch file
tf_imu_camera = Node(package = "tf2_ros",
executable = "static_transform_publisher",
arguments = ["0.0" , "0.0",
"0.0","0.0","0.0","0.0" ,"camera_imu_optical_frame", "camera_link"])



Launch pour D435i avec IMU

$ ros2 launch scatcat_bringup D435i_imu_launch.py

Voir lien

rtabmap_ros

git clone https://github.com/introlab/rtabmap_ros/tree/humble-devel

Installation

RTA-Map Ros2 package

cd ~/ros2_ws

$ git clone https://github.com/introlab/rtabmap.git src/rtabma

$ git clone -b humble-devel https://github.com/introlab/rtabmap_ros.git src/rtabmap_ros

$ rosdep update && rosdep install --from-paths src --ignore-src -r -y

$ export MAKEFLAGS="-j6" # Can be ignored if you have a lot of RAM (>16GB)

$ rm -rf build/ install/ log/

$ colcon build --symlink-install --cmake-args -DCMAKE_BUILD_TYPE=Release

Visualisation

Rviz ou rtabmapviz

Utilisation de RtabMap

RTAB-Map fournit un outil pour parcourir les données dans la base de données :

$ rtabmap-databaseViewer ~/.ros/rtabmap.db

les données de /mapData sont stockées par défaut à ~/.ros/rtabmap.db

Dans rtabmap_ros/launch/ros2/rtabmap.launch.py
'database_path', default_value='~/.ros/rtabmap.db', description='Where is the map saved/loaded.'

Etapes
Lancer l'acquisition
Stopper l'acquisition
Ouvrir avec rtabmap

```
$ rtabmap ~/.ros/rtabmap.db
```

Raz data in rtabmap.db

```
$ ros2 launch rtabmap_ros rtabmap.launch.py \
rtabmap_args:="--delete_db_on_start"
```

Chargement des données de rtabmap.db

```
$ ros2 launch rtabmap_ros rtabmap.launch.py \
    rtabmap_args:="-d"
```

Chargement data for E115.db

```
$ rtabmap ~/Save/E115.db
```

Create a new database at the the specified path:

```
    $ roslaunch rtabmap_ros rtabmap.launch rtabmap_args:="--delete_db_on_start" database_path:="my/path/to/database"
```

```
$ ros2 launch rtabmap_ros rtabmap.launch.py \
    rtabmapviz:=false \
    rtabmap_args:="-d"
```

```
$ ros2 launch rtabmap_ros rtabmap.launch.py \
    rtabmapviz:=false \
    rtabmap_args:="-d" \
    database_path:="~/.ros/E115.db"
```

Odometrie perdue

Dans Rviz, une erreur apparaît sur le topic /odom, cela signifie que l'odométrie ne peut pas être calculée, c'est-à-dire que l'odométrie est perdue et la cartographie ne peut pas continuer.

Pour récupérer de l'odométrie perdue, vous pouvez également réinitialiser l'odométrie à l'aide de "Détection -> Réinitialiser l'odométrie"

Command line $ ros2 service call /reset_odom std_srvs/srv/Empty

N'oubliez pas de replacer la caméra dans une scène avec beaucoup de texture avant de réinitialiser, sinon l'odométrie ne pourra toujours pas démarrer du tout.
La réinitialisation de l'odométrie obligera RTAB-Map à créer une nouvelle carte, la carte devrait donc disparaître et une nouvelle s'afficher.

Les services

```
$ rosservice call rtabmap/set_mode_localization
$ rosservice call rtabmap/set_mode_mapping
```

Carte d'obstacles 2D (Projection au sol)

Dans rviz Display : Map /map
/grid_prob_map
/octomap_grid

Cartographie extérieure stéréo

Il est préferable d'utilisation de la caméra en mode stéréo pour la cartographie en extérieur.

Utilisation de la camera

Il existe 3 modes de fonctionnement

Mode RGB

rtabmap_ros/launch/ros2/realsense_d435i_color.launch.py

params : ‘subscribe_depth’:True
‘subscribe_odom_info’:True
Node : rgbd_odometry

Realsense-camera
$ ros2 launch realsense2_camera rs_launch.py enable_gyro:=true enable_accel:=true unite_imu_method:=1 enable_sync:=true

Rtabmap
$ ros2 launch rtabmap_ros realsense_d435i_color.launch.py

Rviz
$ ros2 launch scatcat_bringup rviz_rtabmap_launch.py

ou passage des parametres de la camera dans le launch parent

$ ros2 launch scatcat_bringup D435i_rtabmap_color_launch.py

Voir lien

Mode infra rouge

rtabmap_ros/launch/ros2/realsense_d435i_infra.launch.py

params : ‘subscribe_depth’:True
‘subscribe_odom_info’:True
Node : rgbd_odometry

$ ros2 launch realsense2_camera rs_launch.py enable_gyro:=true enable_accel:=true unite_imu_method:=1 enable_infra1:=true enable_infra2:=true enable_sync:=true

$ ros2 param set /camera/camera depth_module.emitter_enabled 0

$ ros2 launch rtabmap_ros realsense_d435i_infra.launch.py

Mode Stéréo rtabmap_ros/launch/ros2/realsense_d435i_stereo.launch.py

params : ‘subscribe_stereo’:True
‘subscribe_odom_info’:True
Node : stereo_odometry

$ ros2 launch realsense2_camera rs_launch.py enable_gyro:=true enable_accel:=true unite_imu_method:=1 enable_infra1:=true enable_infra2:=true enable_sync:=true

$ ros2 param set /camera/camera depth_module.emitter_enabled 0

$ ros2 launch rtabmap_ros realsense_d435i_stereo.launch.py

Save 3D Map for publishing in rviz2

Save au format .ply ou .pcd run Rtabmap $ rtabmap ~/.ros/rtabmap.db

File Exporter .ply ou .pcd

Visualisation d’un fichier pcd en utilisant 'pcd_viewer'.
$ pcl_viewer (your_pcd_file.pcd)

Publish a sensor_msgs/pointcloud2 use Perception_pcl package

https://github.com/ros-perception/perception_pcl

With ros2_ws/src/perception_pcl/pcl_ros/tools/pcd_to_poincloud.cpp

the topic name is /cloud_pcd

parameters file_name (where to read)
tf_frame (where to publish)
publishing_period_ms (in ms)

Run
create a tf for the frame
$ ros2 run tf2_ros static_transform_publisher 0 0 0 0 0 0 world_frame odometry

$ ros2 run pcl_ros pcd_to_pointcloud --ros-args -p "file_name:=/home/bertrand/ros2_ws/src/scatcat_bringup/bag_files/cloud_V3.pcd" -p "tf_frame:=odometry" -p "publishing_period_ms:=3000"

$ rviz2