Dérive de cartes

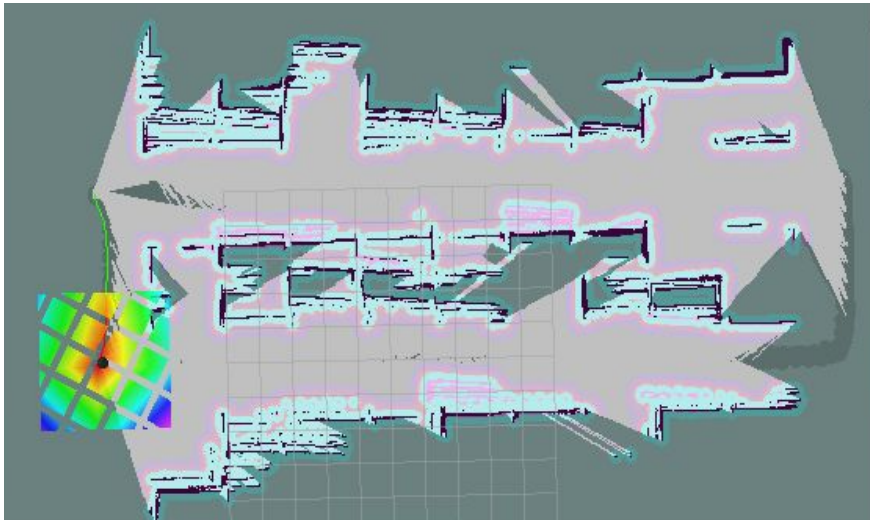
Alexis Dupuis et Corentin Chauvin-Hameau

1.

Présentation de la problématique

Introduction sur le SLAM

Simultaneous Localisation and Mapping



Combine :

- Données **proprioceptives** (ici : odométrie)
- Données **extéroceptives** (ici : scan laser)

Trois objectifs :

- **Mapping** : construire une map métrique grâce aux capteurs (nécessite une excellente localisation)
- **Localisation** : connaître la pose (nécessite une map précise)
- **Déplacement** : planificateurs globaux et locaux de trajectoire (ici : navfn et dwa_local_planner du package move_base)

Introduction sur le SLAM

3 catégories principales d’algorithmes

- **Corrélation des mesures :**

Construction incrémentale de la carte à chaque instant → aligner les éléments communs aux instants t-1 et t

- **Filtrage**

Utilisation alternative :

  - Des données proprioceptives pour estimer la pose
  - Des données externes pour la corriger

- **Optimisation**

Utilisation de toute l’information des instants précédents pour déterminer la map dans son ensemble → réduire un ensemble de contraintes portant sur les données capteur et odom.

Introduction sur le SLAM

Autres éléments importants

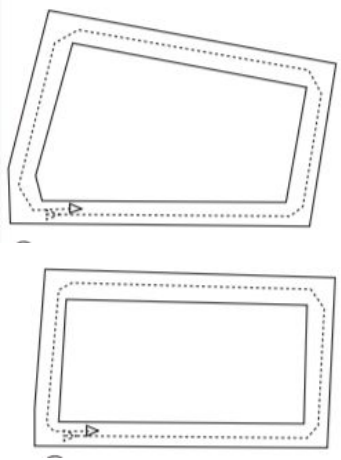
Représentation des incertitudes :

- Matrices de covariances
- Jauge la confiance en les sources d'informations

Détection de boucles :

Lorsqu'un robot revient à un endroit connu, fermeture de boucle

- Influence grandement la forme générale de la map
- Rattrape des erreurs commises
- Crucial !

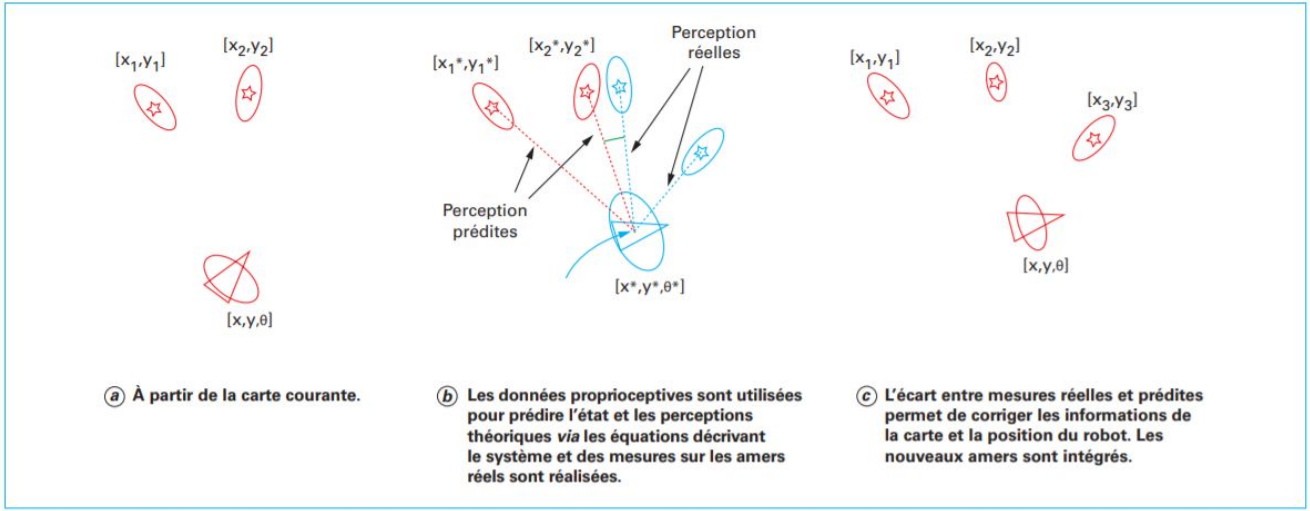


Source : Techniques de l'ingénieur

Filtre à particules Rao-Blackwell

SLAM par filtrage

Alternance des phases prédictives, correctives, et de màj de la map



ⓐ À partir de la carte courante.

ⓑ Les données proprioceptives sont utilisées pour prédire l'état et les perceptions théoriques *via* les équations décrivant le système et des mesures sur les amers réels sont réalisées.

ⓒ L'écart entre mesures réelles et prédites permet de corriger les informations de la carte et la position du robot. Les nouveaux amers sont intégrés.

Source : Techniques de l'ingénieur

Filtre à particules Rao-Blackwell

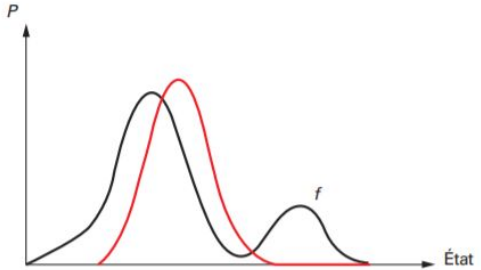
Filtres à particules- Kézako?

Représentation des incertitudes :

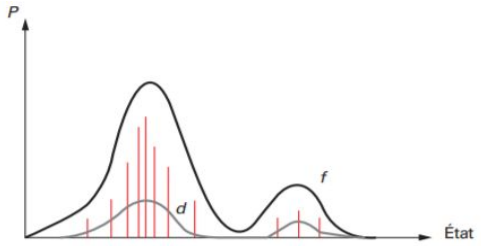
- ~~Approximation gaussienne de l'état~~ → ensemble de particules
- Distribution des particules suivant une loi *d*

Représentation des états possibles par les particules :

- Map reconstruite autour de chaque particule (trivial)
- La plausibilité de chaque map donne du poids *w* → closed-loop
- Ré-échantillonnage régulier des particules :
  - Duplication des plus lourdes
  - Oubli des plus légères



(a) Approximation par une gaussienne (en rouge).

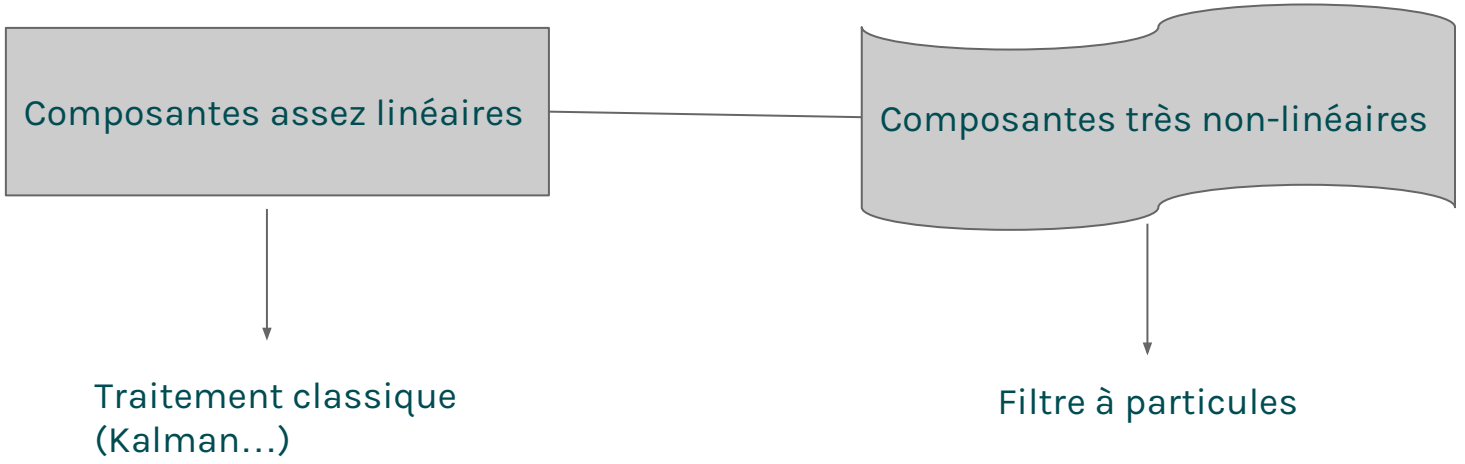


(b) Approximation par un ensemble de particules (en rouge) tirées selon la densité *d* dont la hauteur représente le poids associé.

Source : Techniques de l'ingénieur

Filtre à particules Rao-Blackwellisé

Rao-Blackwellisation



\+ Gmapping : resampling seulement si nécessaire

SLAM avec un environnement large et changeant

Rao-Blackwellisation

Méthode basique (gmapping) : facteur d'oubli

Environnements dynamiques :

- En recherche : (presque) toujours pour applications intérieures (portes, humains, chaises…)
- Objectifs trop différents pour s'inspirer des papiers lus

Problématique des grands environnements :

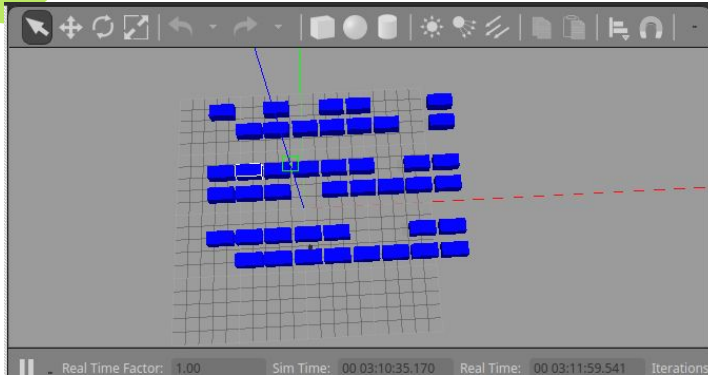
- Nécessité d'optimiser les calculs et le stockage
- Recherches intéressantes menées (optimisation de la mémoire RBPF)
- Mise de côté pour ce projet

Contexte et objectifs du projet

Contexte

SLAM pour un robot autonome sur une aire de stockage de conteneurs :

- Environnement très fortement dynamique : de plein à vide en quelques jours
- Waypoints donnés en absolu : le contrôle de la dérive est critique !



Besoin de données de simulation :

- Simulation : création d'un environnement Gazebo réaliste
- Accessibilité et maîtrise des paramètres utiles sous ROS
- Malléabilité : création automatisée de nouveaux environnements
- Durée de simulation : automatisation des consignes

Contexte et objectifs du projet

Objectifs :

- Fournir un environnement de test clé-en-main
- Travailler la documentation et le code
- Observer la dérive sur quelques séries de tests

2.
Présentation du package ROS

Organisation du package

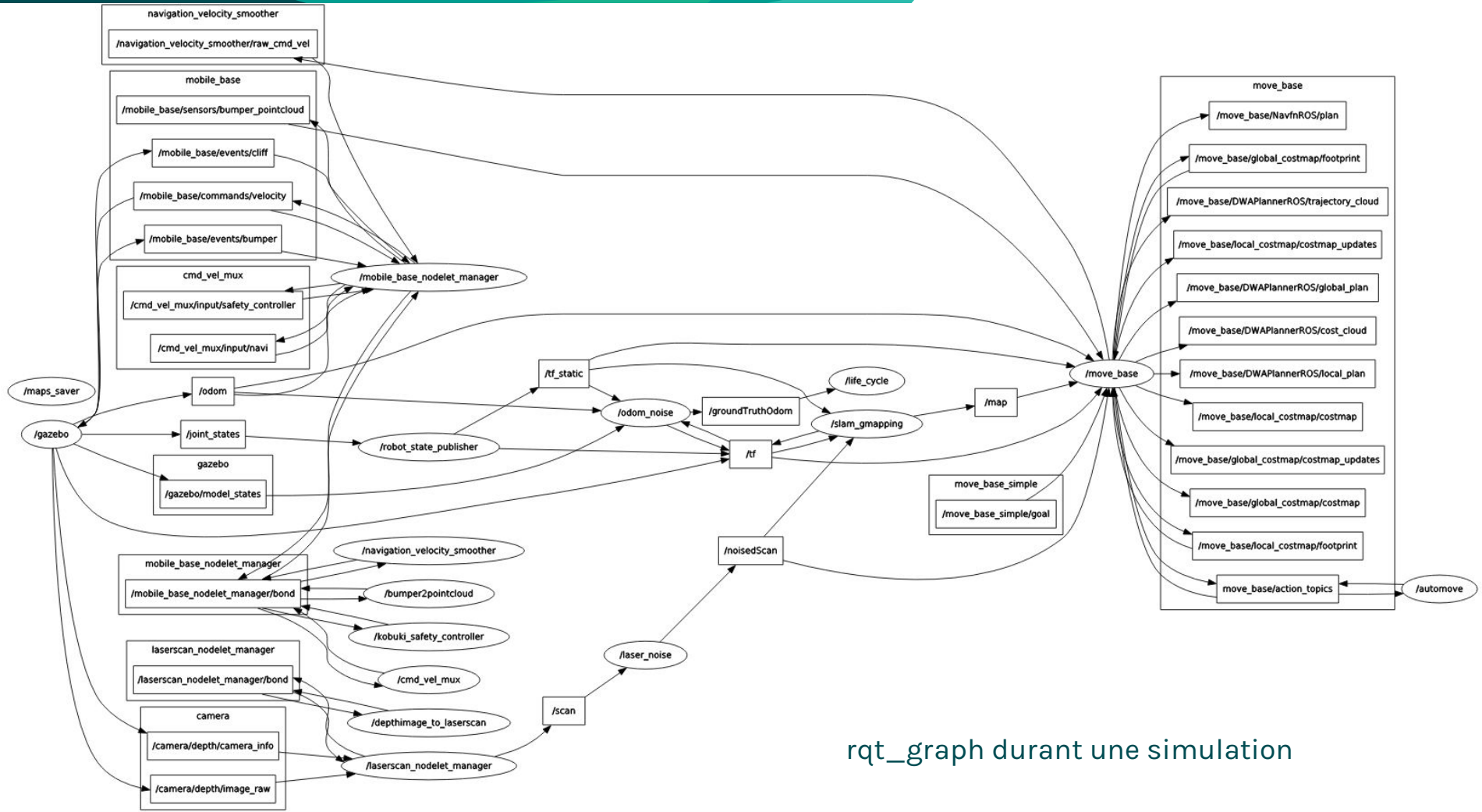
De nombreux nodes déjà implémentés :

- Simulation du Turtlebot sous Gazebo
- Navigation : move_base
- SLAM : gmapping

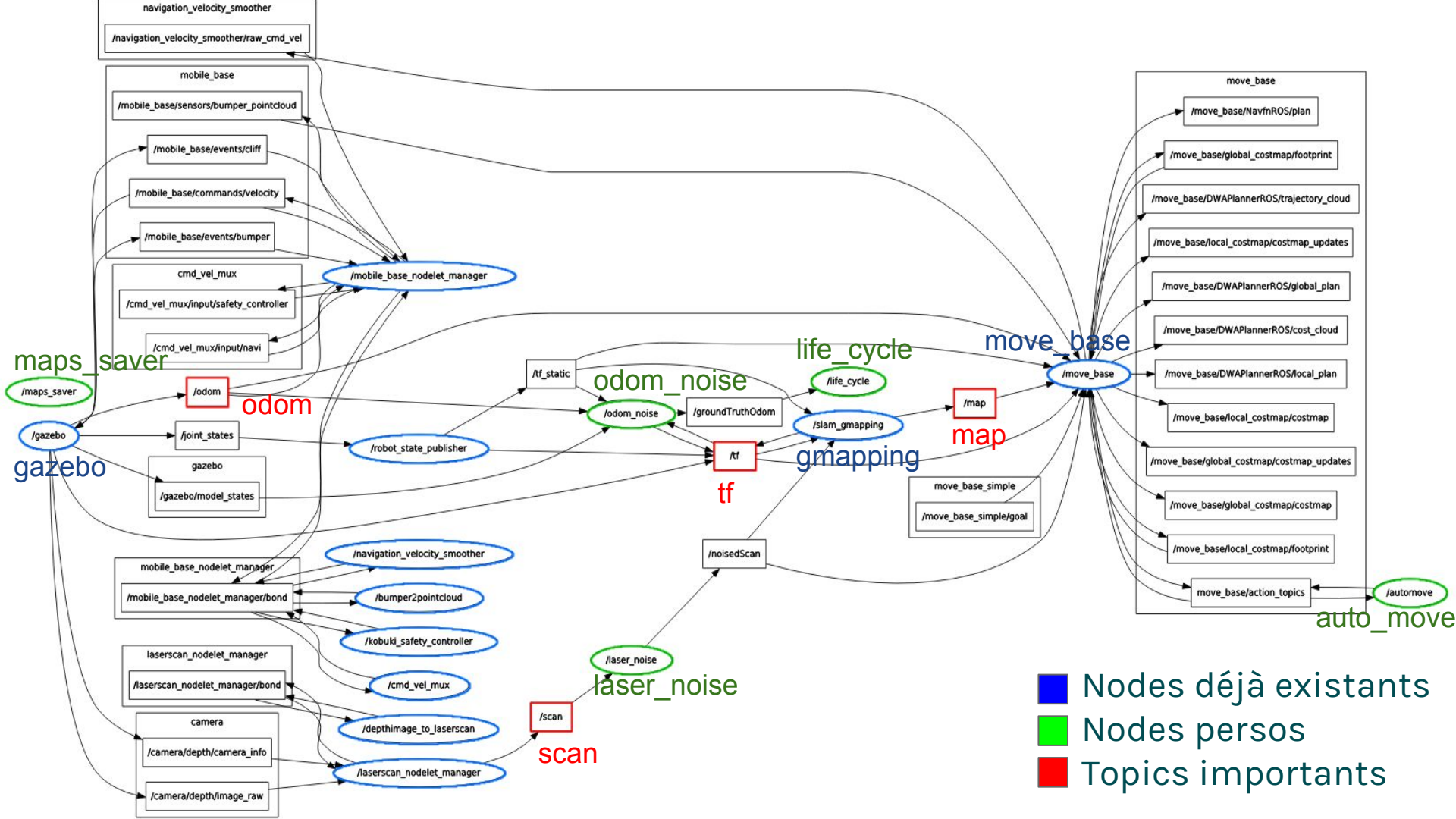
Organisation du package

Des nodes développés par nous :

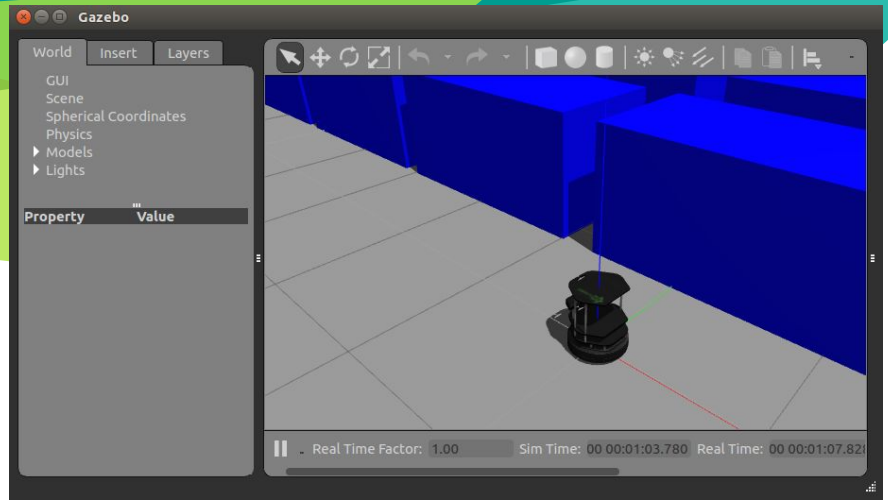
- Bruits de mesures sur l'odométrie et les scans
- Génération du monde de test
- Consignes de déplacement du Turtlebot
- Sauvegarde régulière de la carte



rqt_graph durant une simulation


navigation_velocity_smoother
/navigation_velocity_smoother/raw_cmd_vel
mobile_base
/mobile_base/sensors/bumper_pointcloud
/mobile_base/events/cliff
/mobile_base/commands/velocity
/mobile_base/events/bumper
cmd_vel_mux
/cmd_vel_mux/input/safety_controller
/cmd_vel_mux/input/navi
/mobile_base_nodelet_manager
maps_saver
/maps_saver
/odom
odom
/gazebo
gazebo
/joint_states
gazebo
/gazebo/model_states
/robot_state_publisher
/tf_static
/odom_noise
odom_noise
/groundTruthOdom
/tf
tf
life_cycle
/life_cycle
/slam_gmapping
gmapping
/map
map
move_base_simple
/move_base_simple/goal
move_base
/move_base
move_base
/move_base/NavfnROS/plan
/move_base/global_costmap/footprint
/move_base/DWAPlannerROS/trajectory_cloud
/move_base/local_costmap/costmap_updates
/move_base/DWAPlannerROS/global_plan
/move_base/DWAPlannerROS/cost_cloud
/move_base/DWAPlannerROS/local_plan
/move_base/local_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/global_costmap/costmap
/move_base/local_costmap/footprint
move_base/action_topics
/automove
auto_move
mobile_base_nodelet_manager
/mobile_base_nodelet_manager/bond
/navigation_velocity_smoother
/bumper2pointcloud
/kobuki_safety_controller
/cmd_vel_mux
laserscan_nodelet_manager
/laserscan_nodelet_manager/bond
/depthimage_to_laserscan
/laserscan_nodelet_manager
camera
/camera/depth/camera_info
/camera/depth/image_raw
/noisedScan
/laser_noise
laser_noise
/scan
scan
Nodes déjà existants
Nodes persos
Topics importants

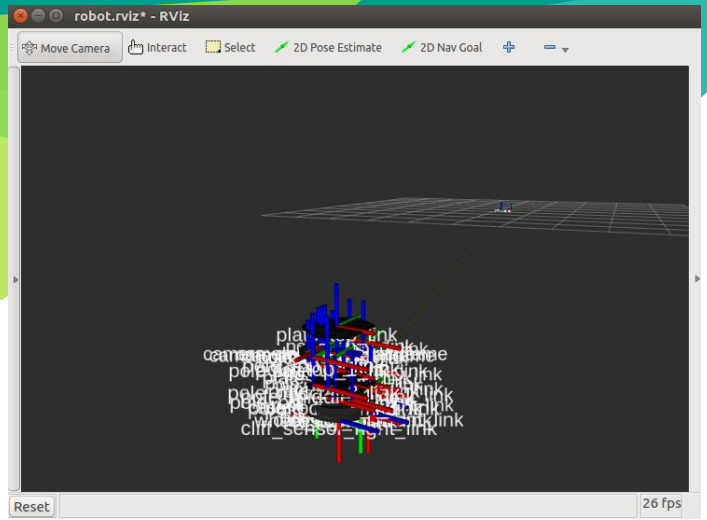
Détails des nodes



Nodes simulant le Turtlebot

- /gazebo : simulateur physique

Détails des nodes



Nodes simulant le Turtlebot

- /gazebo : simulateur physique
- /robot_state_publisher : publie les tf transforms du robot

Détails des nodes

Nodes simulant le Turtlebot

- /gazebo : simulateur physique
- /robot_state_publisher : publie les tf transforms du robot
- /mobile_base_nodelet manager : gère la base mobile du turtlebot (commandes, capteurs…)

Détails des nodes

Nodes simulant le Turtlebot

- /gazebo : simulateur physique
- /robot_state_publisher : publie les tf transforms du robot
- /mobile_base_nodelet manager : gère la base mobile du turtlebot (commandes, capteurs…)
- /kobuki_safety_controller : comportements de sécurité du robot

Détails des nodes

Nodes simulant le Turtlebot

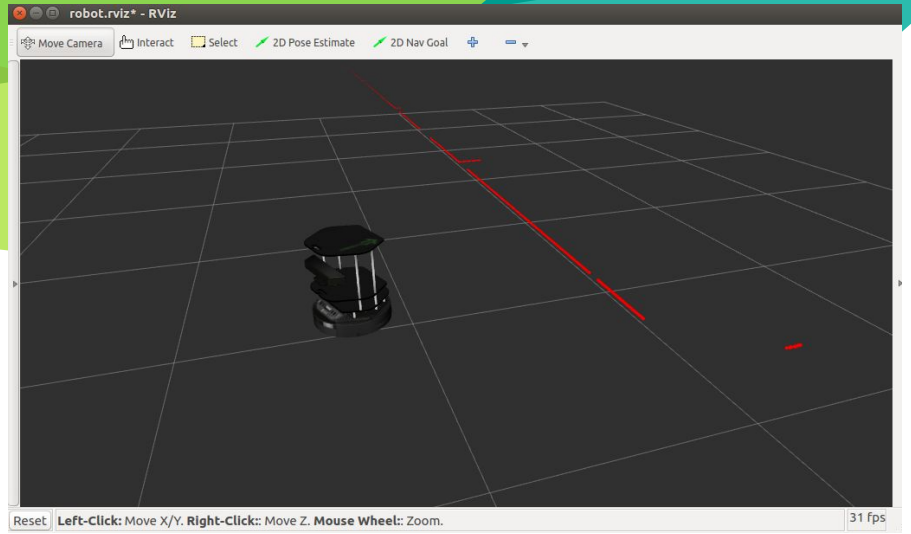
- /cmd_vel_mux : multiplexeur de commandes

Détails des nodes

Nodes simulant le Turtlebot

- /cmd_vel_mux : multiplexeur de commandes
- /navigation_velocity_smoother : filtre les commandes pour les rendre réalisables

Détails des nodes



Nodes simulant le Turtlebot

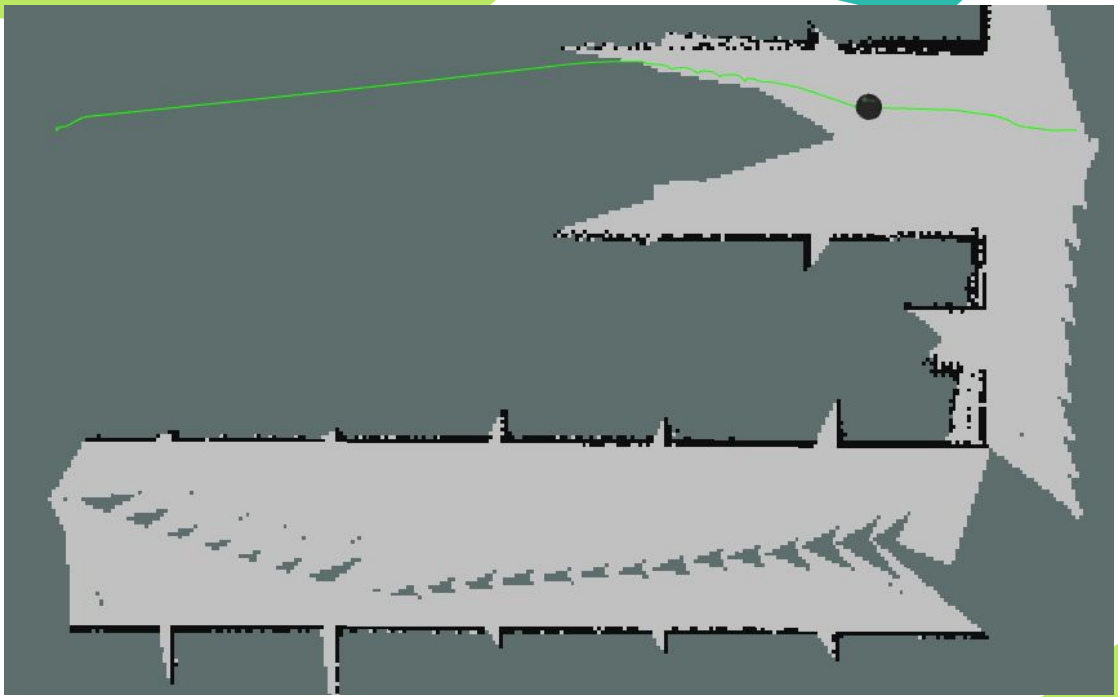
- /cmd_vel_mux : multiplexeur de commandes
- /navigation_velocity_smoother : filtre les commandes pour les rendre réalisables
- /depthimage_to_laserscan et /laserscan_nodelet_manager : simule le capteur laser

Détails des nodes

SLAM

- /slam_gmapping :
  - → Localisation
  - → Mapping

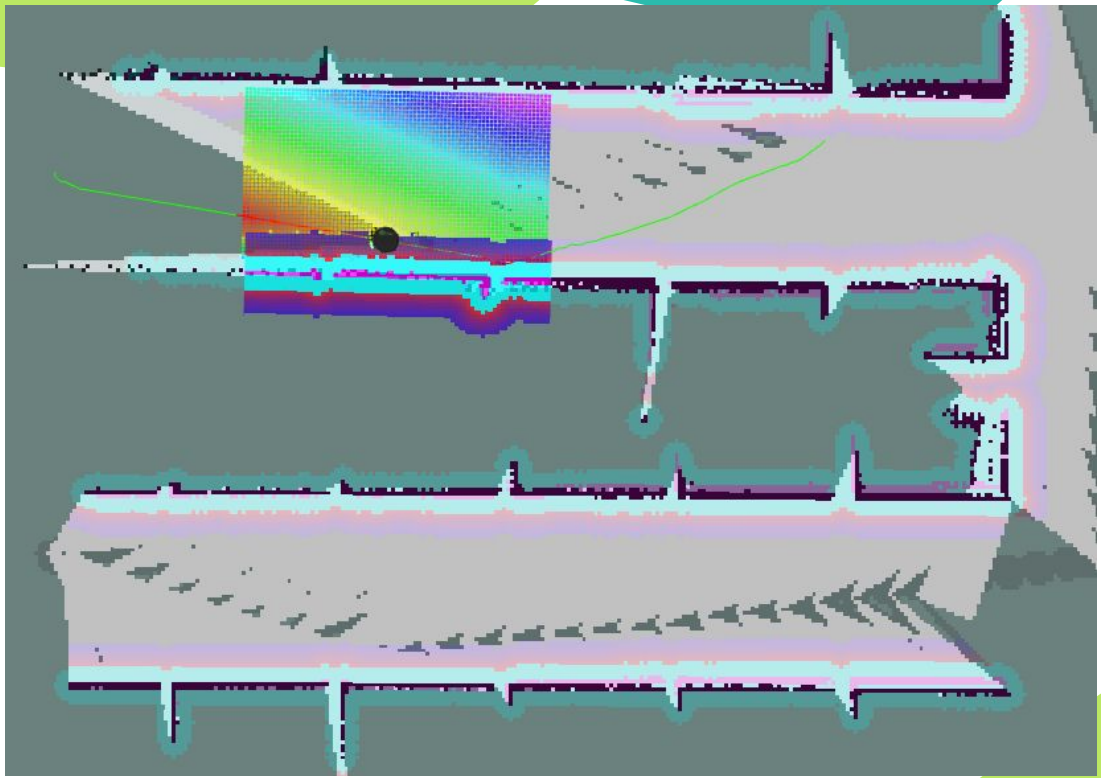
De nombreux prérequis



Détails des nodes

Navigation

- /move_base :
  - → Global planning
  - → Local planning
  - → Commande



De nombreux prérequis

Très modulaire

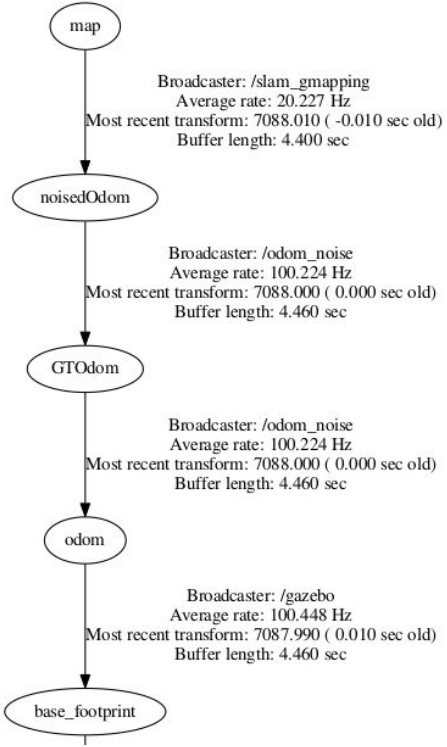
Détails des nodes

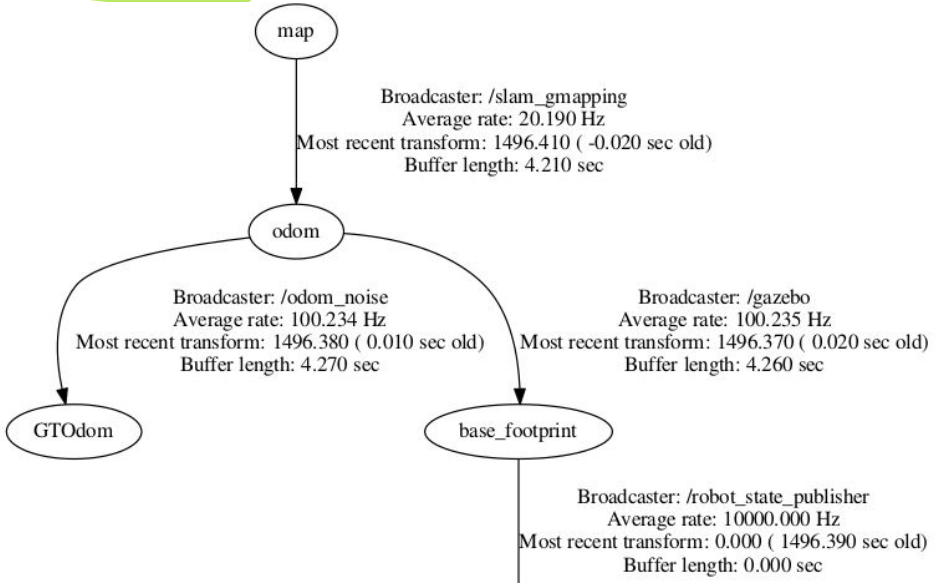
Bruitage de l’odométrie

- /odom_noise :
  - → Récupère la pose réelle donnée par Gazebo
  - → Génère une odométrie bruitée
  - → Publie les nouvelles transformations

Problème : l’odométrie de base n’est pas complètement fiable

Détails des nodes

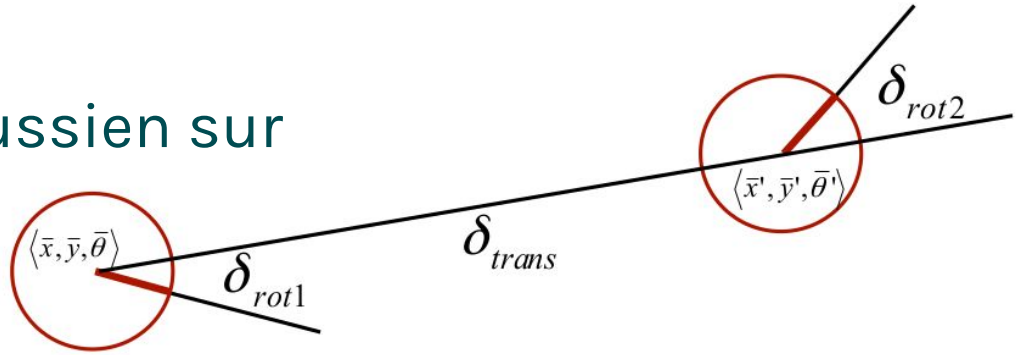




Détails des nodes

Bruitage de l'odométrie

- Modèle de bruit : bruit gaussien sur ces deltas



$$\hat{\delta}_{rot1} = \delta_{rot1} + \varepsilon_{\alpha_1 |\delta_{rot1}| + \alpha_2 |\delta_{trans}|}$$

$$\hat{\delta}_{trans} = \delta_{trans} + \varepsilon_{\alpha_3 |\delta_{trans}| + \alpha_4 |\delta_{rot1} + \delta_{rot2}|}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} + \varepsilon_{\alpha_1 |\delta_{rot2}| + \alpha_2 |\delta_{trans}|}$$

Détails des nodes

Source : acroname.com

Bruitage des mesures laser

- /laser_noise :

  Basé sur les données réelles de l’Hokuyo URG-04LX

Détails des nodes

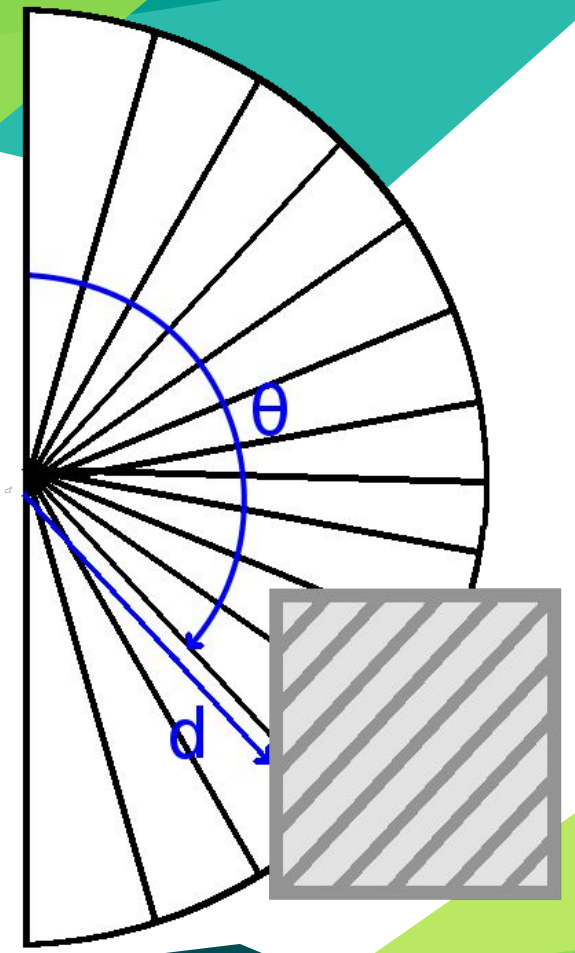
Bruitage des mesures laser

- Modèle de bruit : bruit gaussien sur d

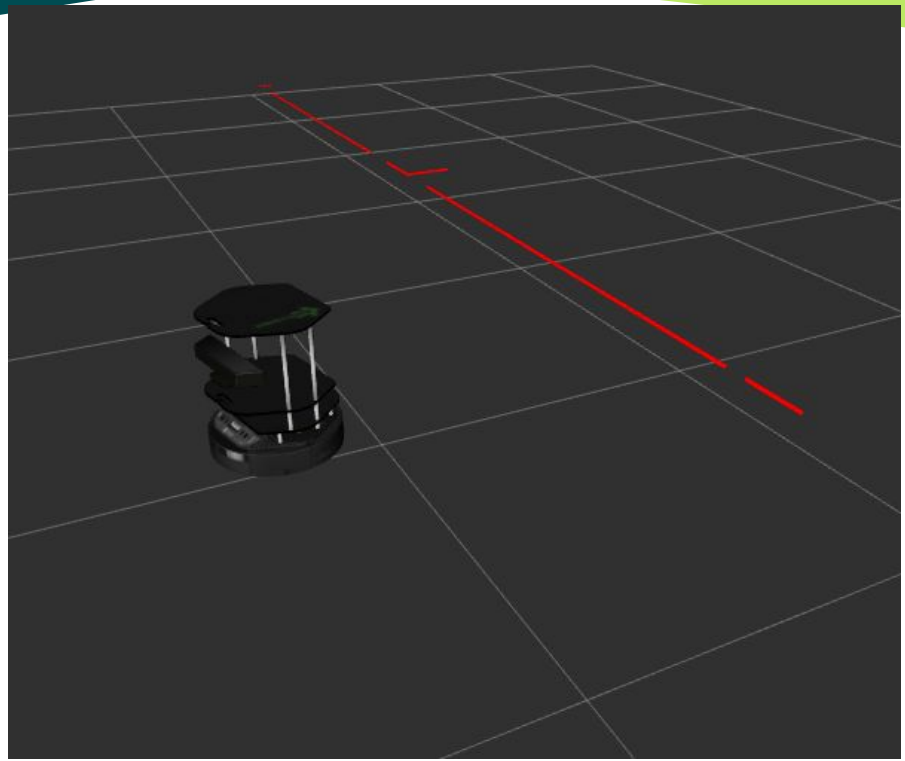
  → Si d < 1 m : σ = 0.01

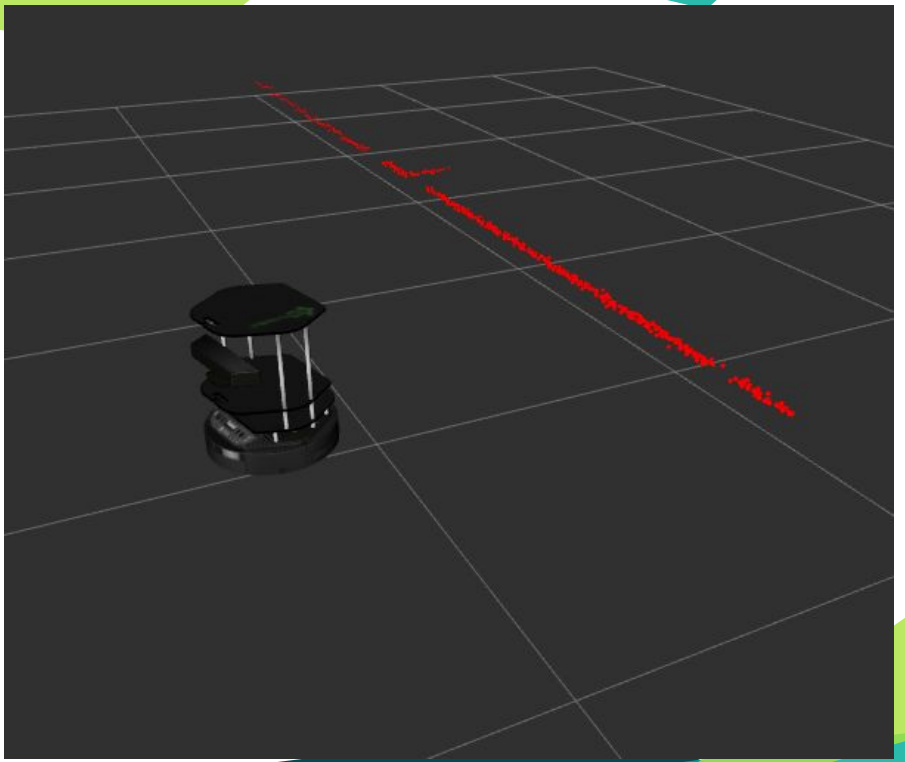
  → Si d >= 1 m : σ = 0.01*d

Le facteur 0.01 est paramétrable dans le launchfile.

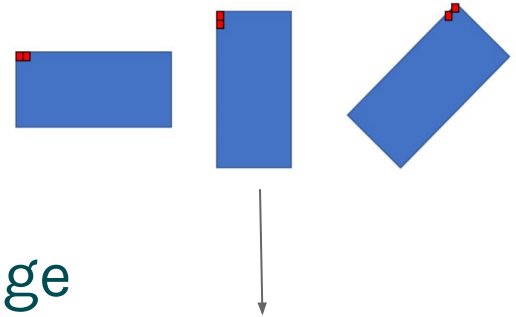


Détails des nodes





Détails des nodes

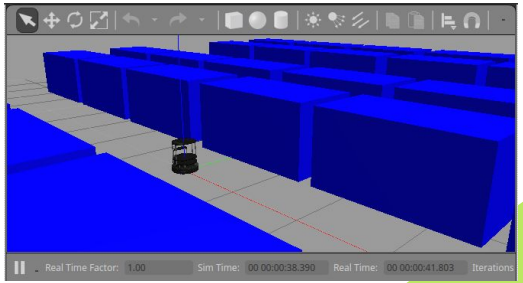


Construction de la map

- /life_cycle :

→ Génère un environnement à partir d'une image

→ Création express d'un nouvel environnement avec MS Paint



→ Gère les cycles d'apparition/disparition

→ Paramétrable : minsleeptime, randsleeptime, leftoversRatio, clearingFactor, securitySpawningDistance…

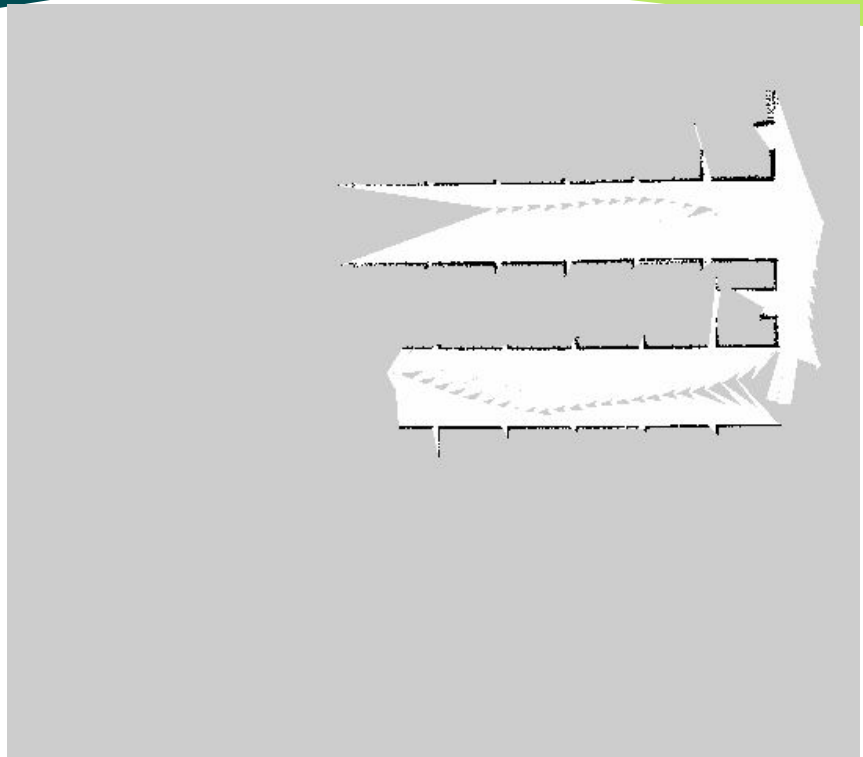
Détails des nodes

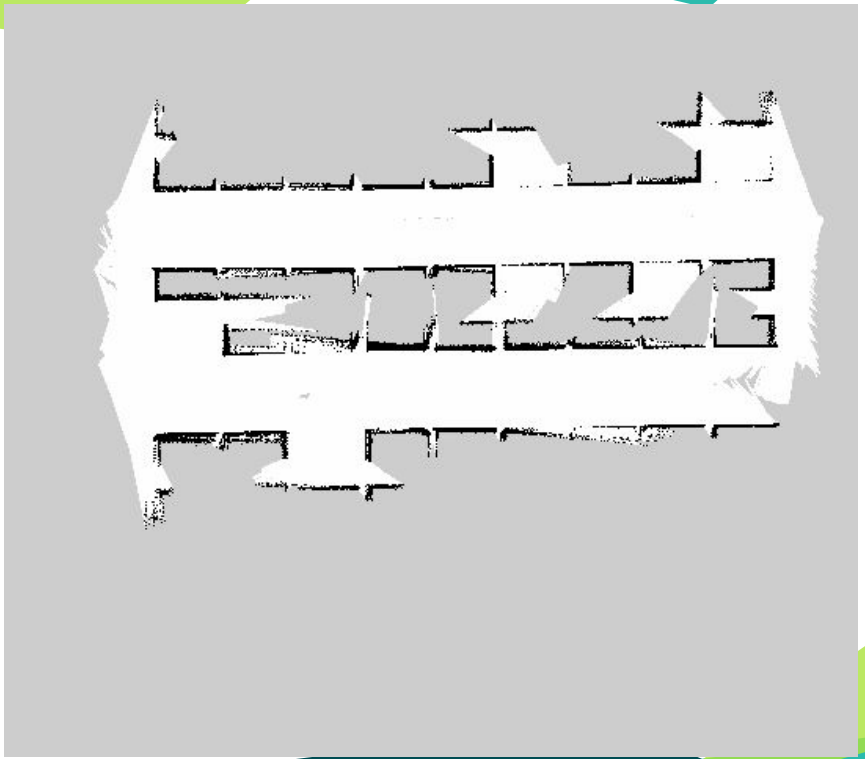
Sauvegarde de la map

- /maps_saver :

  Sauvegarde la map à intervalles réguliers

Détails des nodes



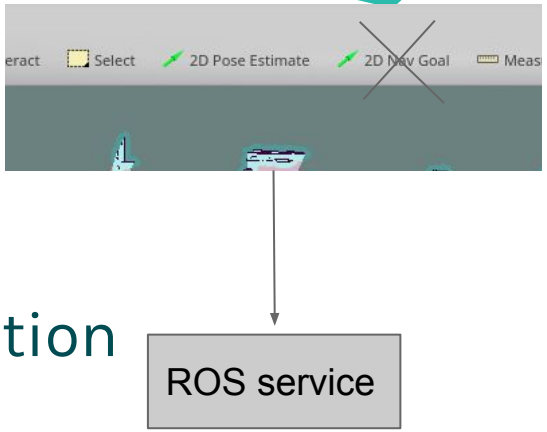


Détails des nodes

Envoi automatique de waypoints

- /automove :

  Automatise l'envoi de consignes en position au robot



Détails sur les launch files et paramètres

Très brièvement…

global_costmap_params.yaml :

- static_map ou rolling_window ?
- Dimensions

local_costmap_params.yaml :

- Fréquence d'update et de publication

costmap_commons_params.yaml :

- robot_radius
- inflation_radius

gmapping.launch.xml :

- Bruits d'odométrie et du laser considérés
- Portée du laser

Mais aussi…

world.launch

navigation.launch

3.

Problèmes rencontrés, résultats de simulation

Principaux problèmes rencontrés

- Navigation en zone inconnue à débloquer
- Navigation en dehors de la global costmap à régler
- Déplacements proches d’obstacles, ou dans des passages étroits
- Publication des transformations pour l’odométrie bruitée
- Mise en place de l’Hokuyo

Principaux problèmes rencontrés

- Accélérer la simulation sous Gazebo

Et enfin…

De nombreux bugs plus ou moins connus :

- Problèmes graphiques de Gazebo et Rviz
- L’API Gazebo de suppression de modèles qui se lance une fois sur quatre

Principaux problèmes rencontrés

```
Pixel found :9.4 ; -4 ; 0
[ INFO] [1521710619.168382012]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1521710619.169171250]: waitForService: Service [/gazebo/set_physics_properties] has not been
 advertised, waiting...
libGL error: failed to create drawable
X Error of failed request:  0
  Major opcode of failed request:  155 (GLX)
  Minor opcode of failed request:  26 (X_GLXMakeContextCurrent)
  Serial number of failed request:  33
  Current serial number in output stream:  33
terminate called after throwing an instance of 'boost::exception_detail::clone_impl<boost::exception_
detail::error_info_injector<boost::lock_error> >'
  what():  boost: mutex lock failed in pthread_mutex_lock: Invalid argument
Aborted (core dumped)
[gazebo-2] process has died [pid 30977, exit code 134, cmd /opt/ros/kinetic/lib/gazebo_ros/gzserver -
e ode /home/corentin/Documents/ROS/ecn_projet_ws/src/evolutive_map/worlds/empty.world __name:=gazebo
__log:=/home/corentin/.ros/log/b3bed6ce-2db2-11e8-898e-5800e33a6476/gazebo-2.log].
log file: /home/corentin/.ros/log/b3bed6ce-2db2-11e8-898e-5800e33a6476/gazebo-2*.log
```

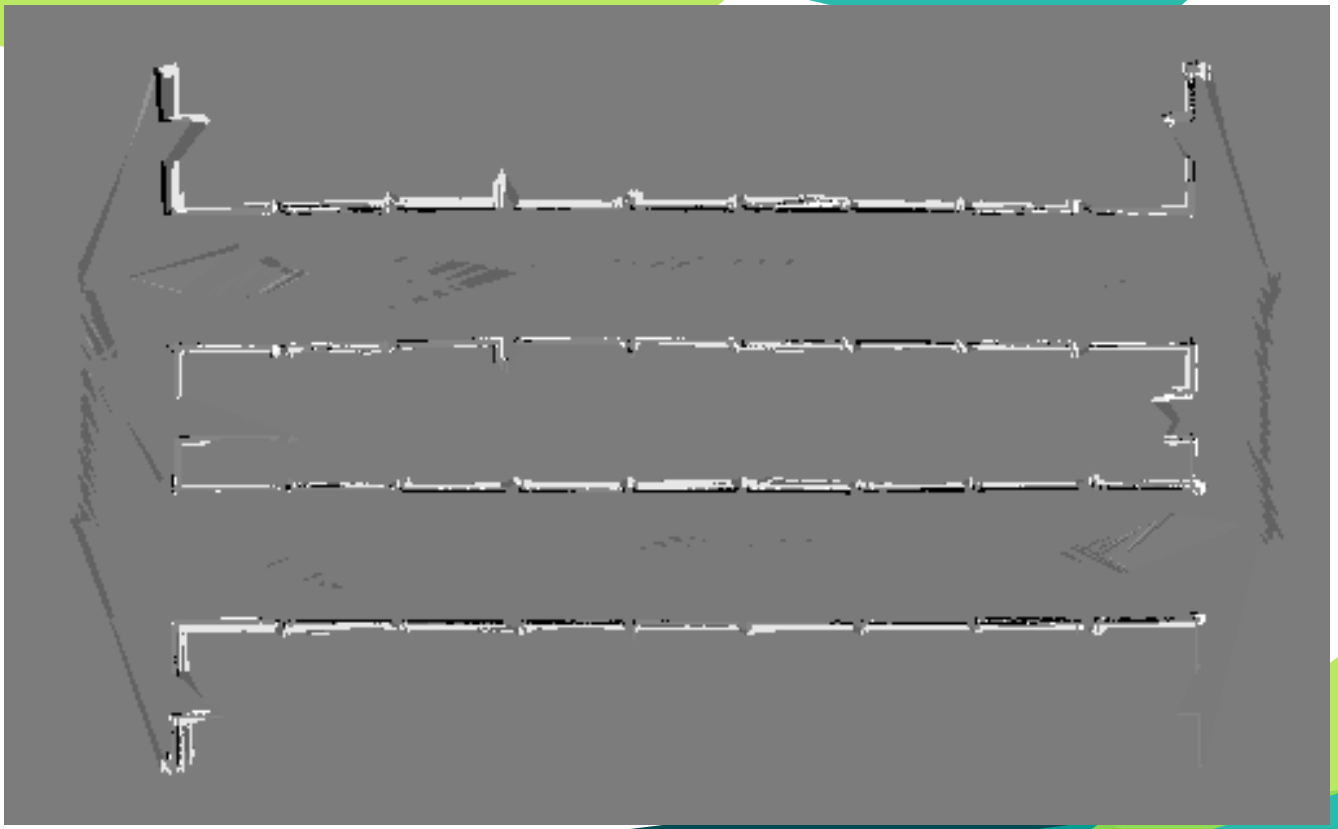
Le hasard, c’est Gazebo qui se promène incognito, Einstein

Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser non bruité

- Pas d’évolution de la map

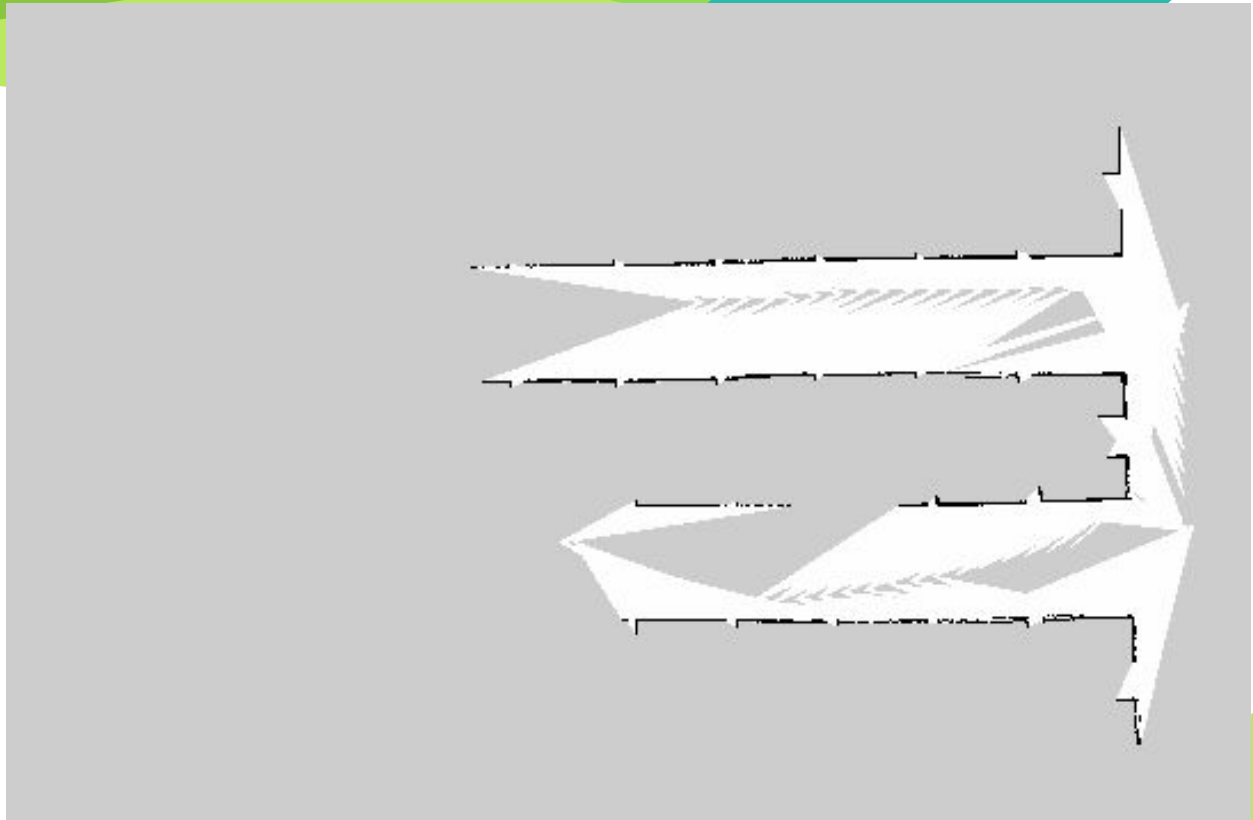


Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser non bruité

- Pas d’évolution de la map

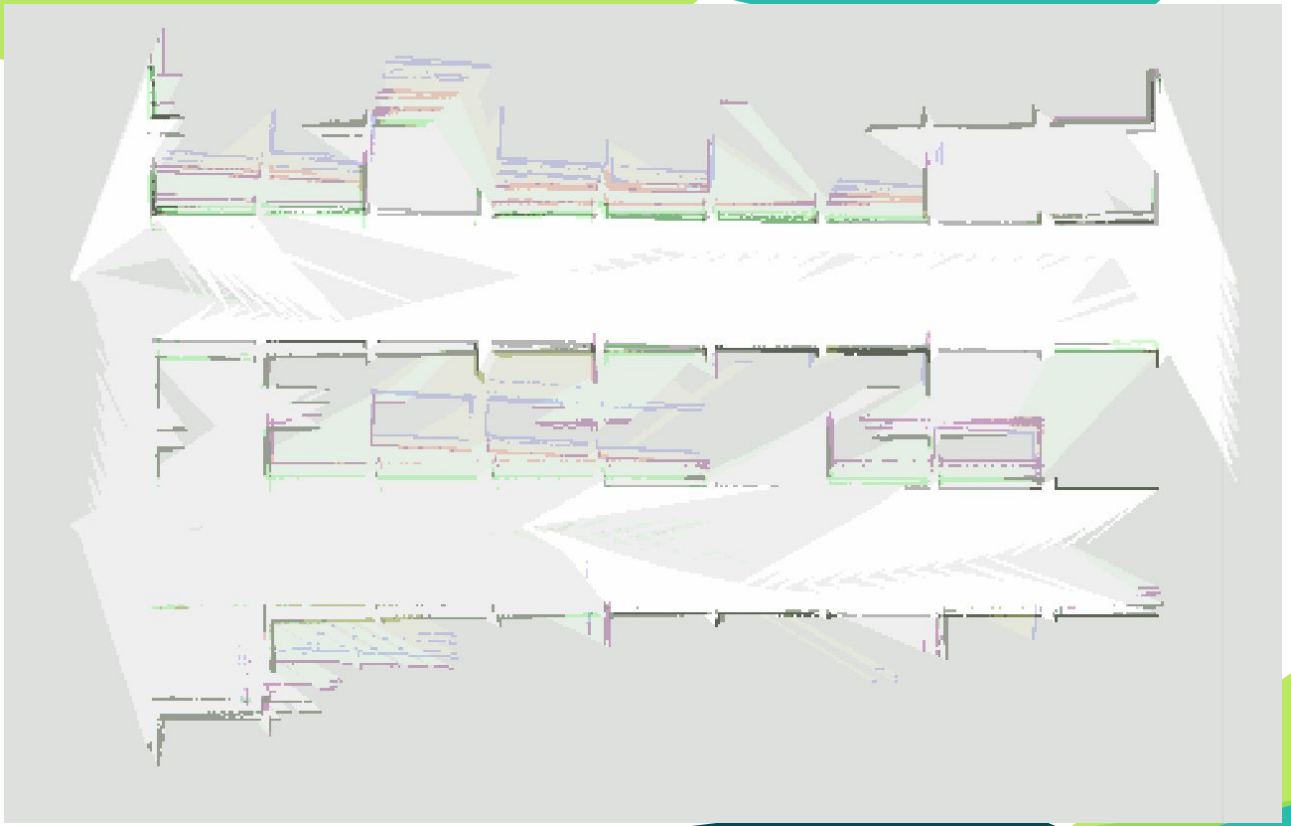


Simulations

Paramétrisation :

- Odométrie bruitée
- Laser peu bruité

- Évolution durant la phase de **disparition** des conteneurs
- Disparition toutes les 1 à 3 min
- 3h de simulation

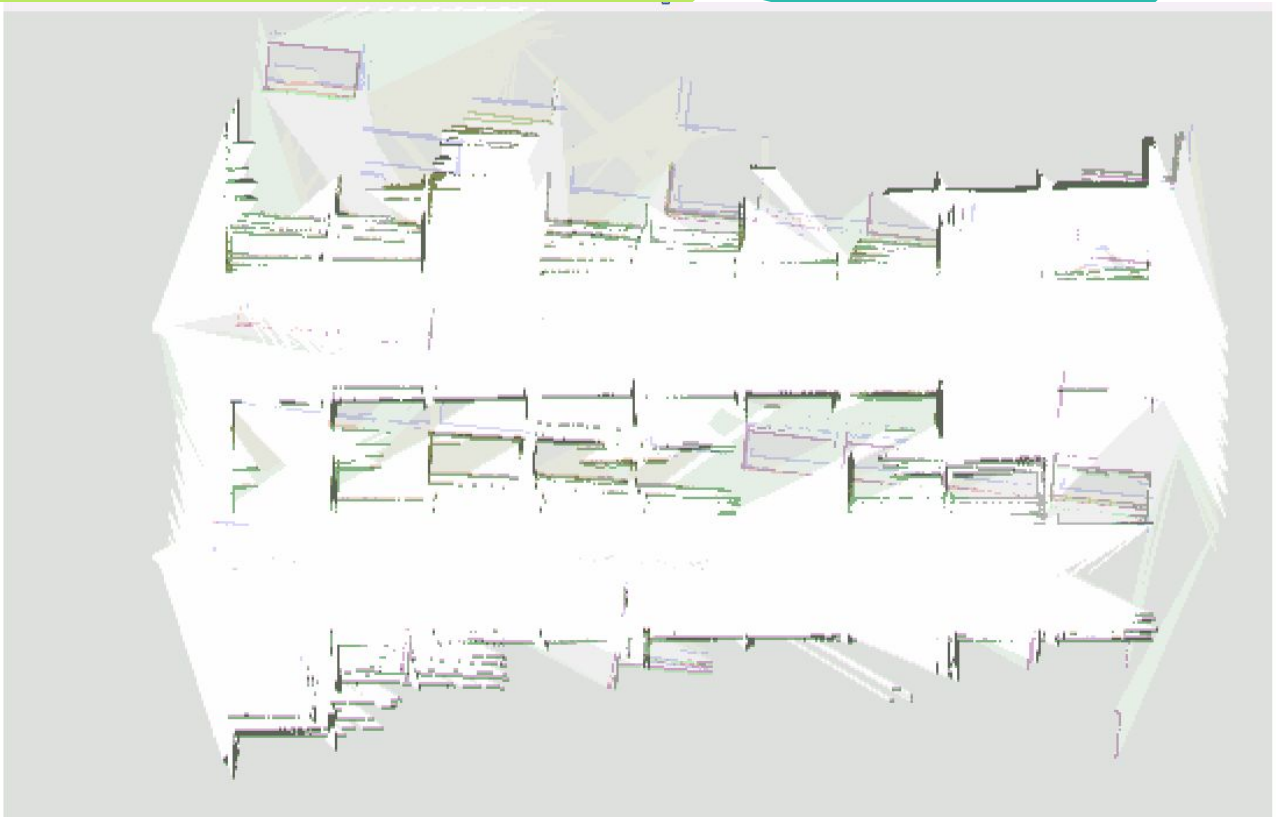


Simulations

Paramétrisation :

- Odométrie bruitée
- Laser peu bruité

- Évolution durant la phase d'**apparition** des conteneurs
- Apparition toutes les 1 à 3 min

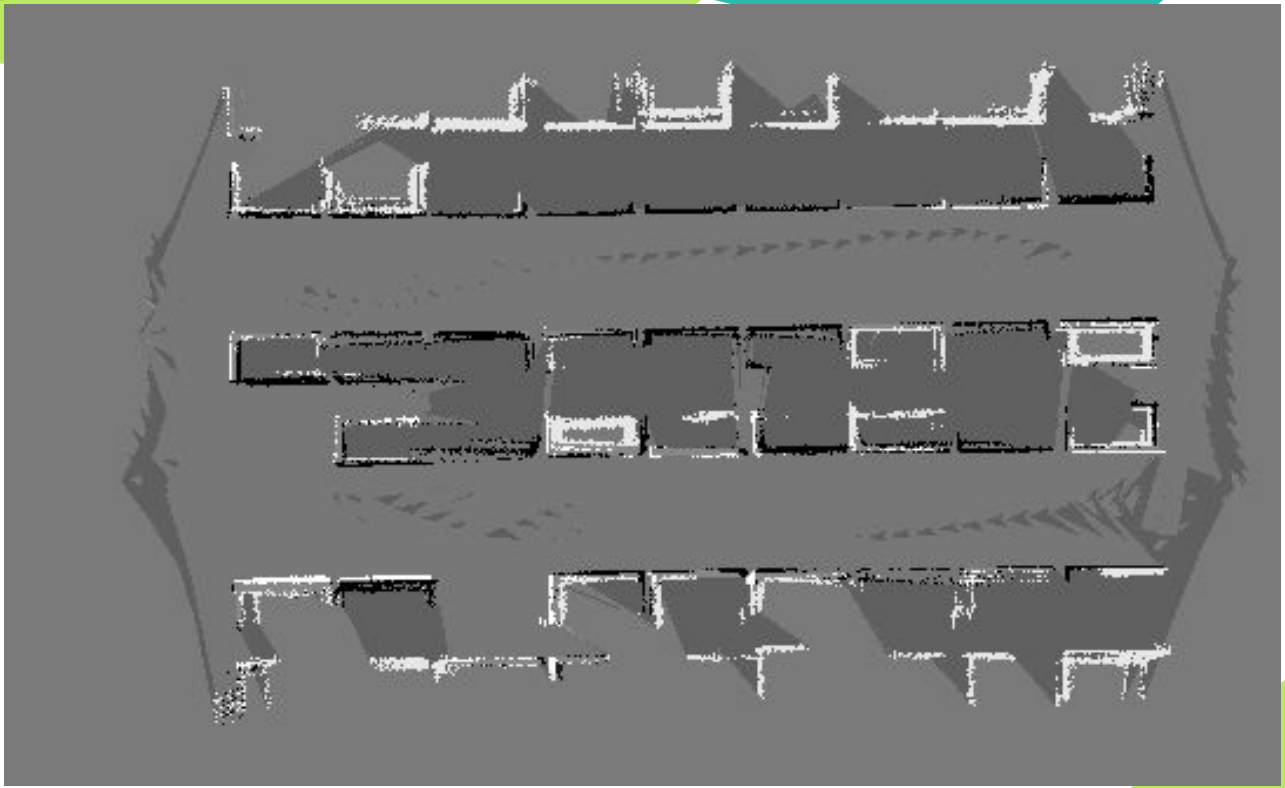


Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser peu bruité

- Évolution durant un cycle complet apparition / disparition.
- Apparition/disparition toutes les 1 à 3 min

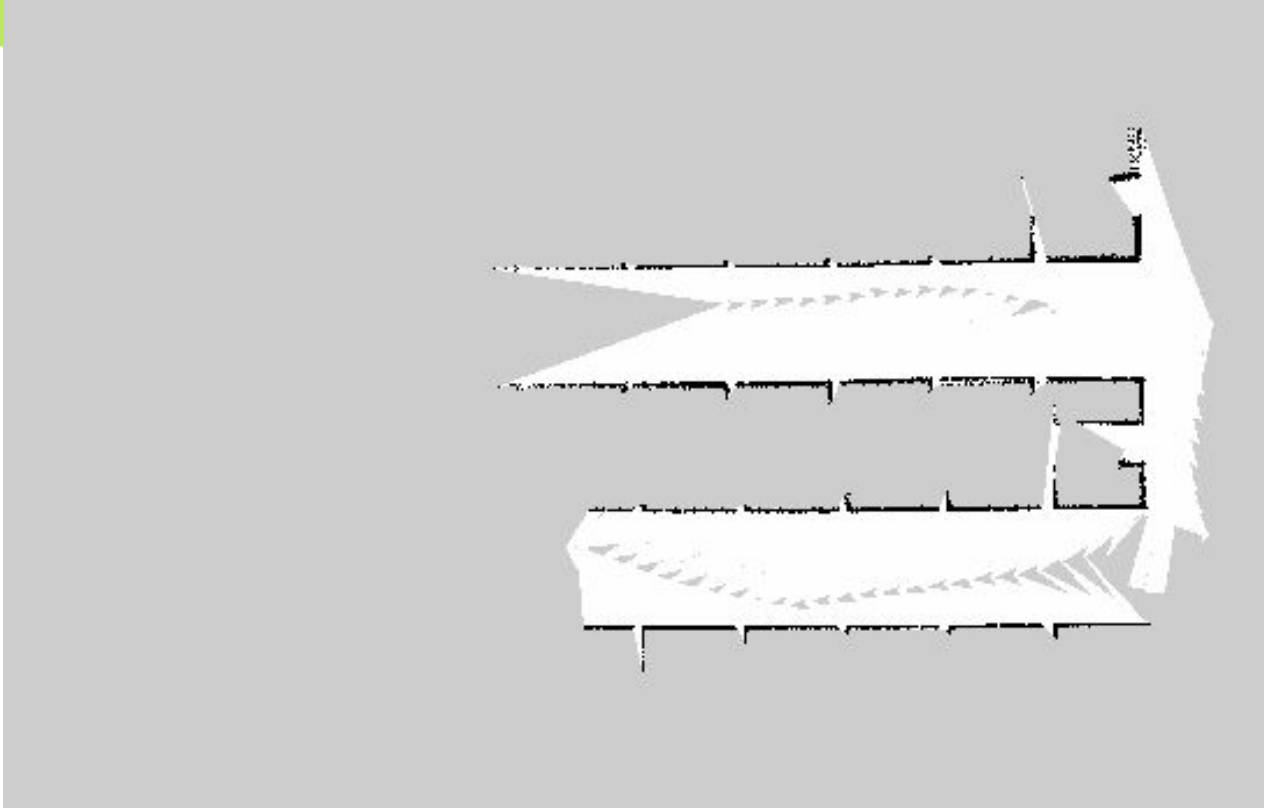


Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser peu bruité

- Évolution durant un cycle complet apparition / disparition.
- Apparition/disparition toutes les 1 à 3 min

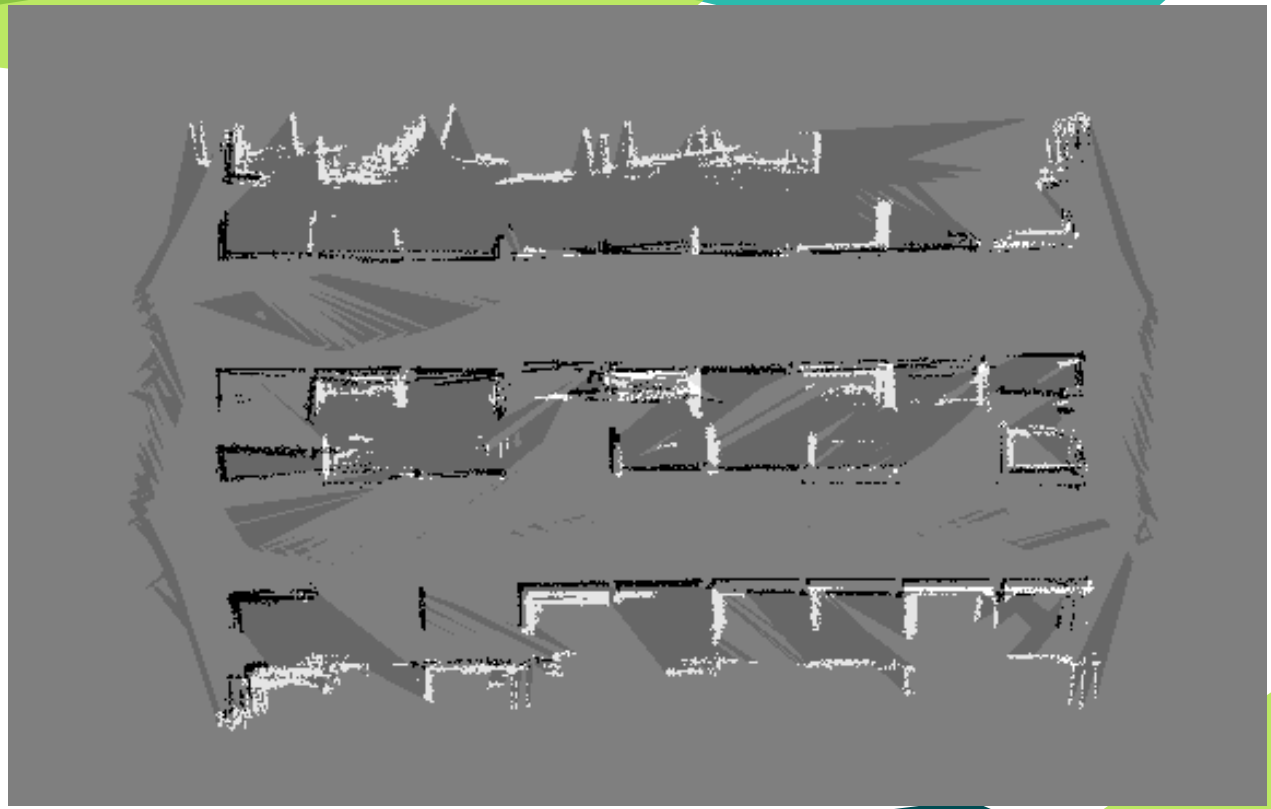


Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser bruité

- Évolution durant un cycle complet apparition / disparition
- Apparition toutes les 1 à 3 min

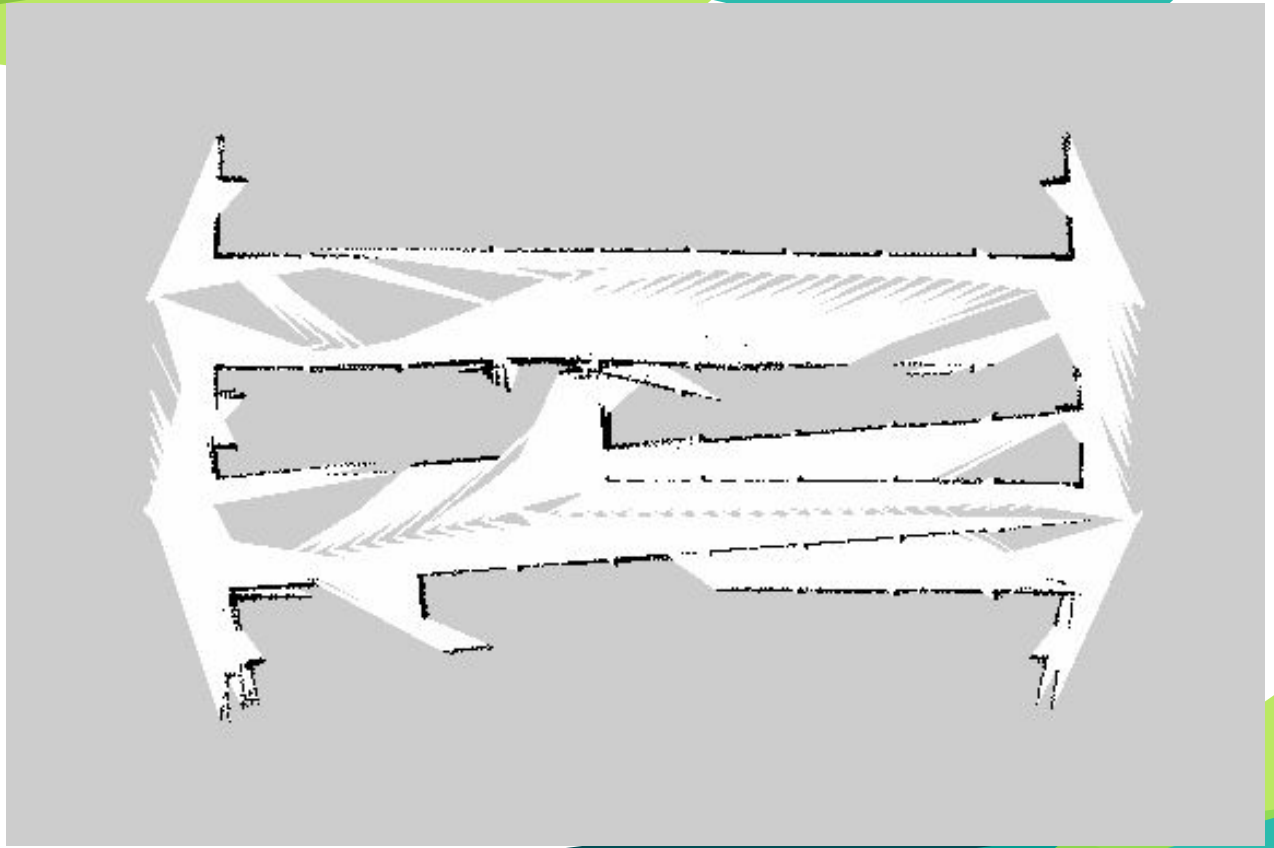


Simulations

Paramétrisation :

- Odométrie non bruitée
- Laser bruité

- Évolution durant un cycle complet apparition / disparition
- Apparition toutes les 1 à 3 min



Problèmes soulevés

- Mauvais oubli des conteneurs disparus par gmapping
- Angle du laser pas assez grand
- Incertitudes sur la validité des tests avec bruits d’odométrie

Perspectives

Tests prêts à être faits :

- Utiliser une map statique
- Jouer sur les différents paramètres : fréquence d'apparition/disparition, bruits, répartition des waypoints...

Perspectives

Tests utilisant notre environnement

- Utiliser l'Hokuyo plutôt que la Kinect
- Utiliser d'autres méthodes de SLAM
- Trouver et modifier le "facteur d'oubli" de gmapping

Isn't it a pleasure to study and practice what you have learned ?

Confucius