

UNIVERSITÉ PAUL SABATIER



MASTER INTELLIGENCE ARTIFICIELLE ET
RECONNAISSANCE DES FORMES
MASTER ROBOTIQUE : DÉCISION ET COMMANDE

Rapport

Navigation Autonome de Robot Mobile

Auteur :
Thibaut AGHNATIOS
Marine BOUCHET
Bruno DATO
Tristan KLEMPLE
Thibault LAGOUTE

Tuteur :
Frédéric LERASLE
Michaël LAUER
Michel TAIX

LAAS-CNRS

13 mars 2017

Suivi du document

Nom du document	Version Majeure	Version Majeure	Date de création	Dernière version
Rapport	A	0	13/03/2017	13/03/2017

Auteurs du document

Rédaction	Intégration	Relecture	Validation Interne
Equipe	??	??	??

Validation du document

Validation	Nom	Date	Visa

Liste de diffusion

Le rapport du projet est diffusé à l'ensemble des clients et des intervenants externes aux projets.

Historiques de révision

Version	Modification apportée	Auteur	Date
A.0	Création du document	Bruno Dato	13/03/2017

Table des matières

1	Présentation du projet	3
1.1	Contexte	3
1.2	Problématiques	3
2	Recherche balle	4
3	Navigation avec amers 2D dans un environnement connu	5
3.1	Solution mise en place	5
3.2	Commande haut niveau	5
3.3	Détection et localisation	5
3.3.1	Détection	5
3.3.2	Localisation	5
3.4	Commande	6
3.4.1	Odométrie	6
4	Conclusion	7

1 Présentation du projet

1.1 Contexte

1.2 Problématiques

2 Recherche balle

3 Navigation avec amers 2D dans un environnement connu

3.1 Solution mise en place

3.2 Commande haut niveau

3.3 Détection et localisation

Avant toute chose, il est important de rappeler que quelque soit la situation :

- `/map` → `/odom` → `/base_link` (cf. <http://www.ros.org/reps/rep-0105.html>)
- `/odom` est le repère relatif, assimilé certain, du robot et est placé au départ sur `/map`
- chaque frame peut avoir plusieurs fils mais qu'un seul parent
- la transformé entre deux frames est décrite par deux attributs :
 - `m_origin` : Vector3 de translation
 - `m_basis` : Matrix3x3 de rotation

3.3.1 Détection

Pour la localisation, nous utilisons la librairie *ar_track_alvar* qui utilise des marqueurs AR tags. Cette librairie permet de détecter et de suivre ces markers. Elle retourne l'identifiant, la position et l'orientation du marker.

Mise en place physique :

On a utilisé des AR tags de 16×16 cms. On place leur milieu à 31 cm du sol.

Initialisation :

Performance

- tracking
- x ar
- renvoie l'id du marker
- renvoie la position et l'orientation d'un amer dans le repère de la caméra `/camera_rgb_optical_frame`

Dans notre projets

3.3.2 Localisation

`localisation_node.cpp` :

Ce nœud permet d'envoyer la nouvelle localisation du robot si il détecte un marqueur.

La recherche se déclanche uniquement quand la commande haut niveau publie un `<std_msgs::Empty>` sur le topic `/nav/HLC/askForMarker`.

Dans le cas ou un marqueur est visible :

init `/map` → `/odom` → `/baselink` → `/camera_rgb_frame` → `/camera_rgb_optical_frame`

arrivée d'un marker `/camera_rgb_optical_frame` → `/ar_marker_0`

traitement

`/baselink` → `/camera_rgb_frame` → `/camera_rgb_optical_frame` → `/ar_marker_0`

on sait donc `/ar_marker_0` → `/baselink` = `/marker_0` → `/baselink`

sachant `/map` → `/marker_0`, on sait `/map` → `/baselink` = `/map` → `/odom`

Cette transformation est envoyé sur `/new_odom` que l'on publie avec le publisher `odom_pub`.

On publie également l'id du marqueur vu sur `/nav/loca/markerSeen` sous un `<std_msgs::Int16>`.

Dans le cas où on ne voit rien on publie : -1

localisation_broadcaster_node.cpp :

Ce nœud permet de tout le temps broadcaster la transform entre /map et /odom. Il souscrit à la <geometry_msgs : :Transform> /new_odom, qui contient la "nouvelle" position certaine du robot. Dès que celle-ci est publiée, /odom actualise sa position et on réinitialise le nav_msgs/Odometry : il devient notre nouveau référentiel.

3.4 Commande

3.4.1 Odométrie

4 Conclusion

Table des figures

1 texte de la légende 5

ANNEXE