

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

<b>1</b>	<b>Présentation du projet</b>	<b>3</b>
1.1	Contexte . . . . .	3
1.2	Problématiques . . . . .	3
<b>2</b>	<b>Recherche balle</b>	<b>4</b>
<b>3</b>	<b>Navigation avec amers 2D dans un environnement connu</b>	<b>5</b>
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 . . . . .	6
3.4	Commande . . . . .	6
3.4.1	Odométrie . . . . .	6
<b>4</b>	<b>Conclusion</b>	<b>7</b>

# 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

#### CALIBRAGE

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

- `/map`  $\rightarrow$  `/odom`  $\rightarrow$  `/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 détection, nous utilisons la librairie `ar_track_alvar` qui utilise des marqueurs AR tags. Cette librairie permet de détecter et de suivre ces markers en utilisant la [?]. Elle retourne l'identifiant, la position et l'orientation du marker dans la position de la `/camera_rgb_optical_frame`, dans notre cas. La librairie fournit 55 AR tags et la possibilité d'entendre cette liste facilement. Dans notre cas, nous utilisons uniquement le noeud qui ne permet de lire plusieurs AR tags à la fois dans la même capture du flux vidéos.

#### Mise en place physique :

On a utilisé des AR tags de  $16 \times 16$  cm. On place leur milieu à 31 cm du sol.

#### Initialisation :

Le noeud `ar_track_alvar` s'initialise dans `localisation.launch` avec les paramètres suivant :

```
<arg name="marker_size" default="16" />
<arg name="max_new_marker_error" default="0.08" />
<arg name="max_track_error" default="0.2" />

<arg name="cam_image_topic"
      default="/camera/depth_registered/points" />
<arg name="cam_info_topic"
      default="/camera/depth_registered/camera_info" />
<arg name="output_frame" default="/camera_rgb_optical_frame" />
```

#### Lecture du contenu [? ] :

On peut écouter les `ar_track_alvar_msgs : :AlvarMarkers` que publie le noeud avec la commande :

```
rostopic echo /ar_marker_pose
```

#### Performances et évolutions :

L'orientation du marqueur trouvé par `ar_track_alvar` doit être :  $\vec{z}$  la normale et  $\vec{x}$  vers le haut. Il arrive, pour des problèmes inconnus (le réseau?) que ce repère ne soit pas correctement capturé, avec par exemple,  $\vec{y}$  vers le haut. Le robot, à la fin de la localisation se trouve alors couché. Il faudrait alors comprendre d'où vient le problème ou, sinon, ne pas prendre en compte les orientations aberrantes. La détection des marqueurs de  $16 \times 16$  cm s'effectue jusqu'à 2m 25 avec une précision de 2 cm et jusqu'à  $45^\circ$  d'angle à  $5^\circ$  près. Il serait intéressant, à l'avenir, d'approfondir les résultats obtenus avec d'autres tailles si l'erreur est trop aléatoire, ou alors, intégrer un filtre de Kalman pour fusionner les données odométriques et les observations.

FIGURE 1 – texte de la légende

### 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 . . . . . 6

## ANNEXE