

**UNIVERSITÉ CLERMONT AUVERGNE**

**RAPPORT**

pour obtenir le diplôme de

**MASTER Automatique, Robotique**

de : **École Universitaire de Physique et d'Ingénierie**

présentée et soutenue publiquement

par

**Gaëtan LEMBERT**

le 30 juin 2023

Encadré par **LENGAGNE Sébastien**

---

**Génération de trajectoire virtuelle à base de  
QR codes et localisation d'un robot mobile**

---



# Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au bon déroulement de ce projet et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à mon professeur encadrant, Mr Sébastien LENGAGNE, qui m'a beaucoup aidé tout au long de ce projet. Grâce à ses conseils, j'ai pu comprendre comment s'organiser pour le bon développement d'un projet. Il fut d'une aide précieuse dans les moments délicats.

Enfin, je tiens à remercier les personnes qui m'ont conseillées et relues lors de la rédaction de ce rapport de projet : mes parents et mon colocataire.

# Résumé

Ce rapport a pour but de présenter le travail fourni pour réaliser ce projet, qui s'inscrit dans le cadre du Master Automatique, Robotique de l'EUPI de l'Université Clermont Auvergne à Aubière.

L'objectif est de mettre en place un ensemble de paquets ROS destinés à un robot mobile qui sera utilisé lors de TPs de robotique mobile, portant sur le suivi de ligne, à Polytech.

Le robot devra détecter des QR codes placés le long du parcours, qui généreront chacun un point de passage virtuel autour d'eux. Ensuite, il générera une trajectoire passant par tous les points de passage récoltés, et enfin retournera sa position en translation et rotation par rapport à la trajectoire tracée.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Présentation . . . . .	6
1.2	Cahier des charges . . . . .	7
1.3	Protocole . . . . .	7
<b>2</b>	<b>Architecture de la programmation</b>	<b>10</b>
<b>3</b>	<b>Définition des points de passage</b>	<b>12</b>
3.1	Localisation des QR codes . . . . .	12
3.2	Localisation des points de passage . . . . .	14
<b>4</b>	<b>Calcul de la trajectoire</b>	<b>16</b>
4.1	Choix de la méthode . . . . .	16
4.2	Définition des paramètres . . . . .	17
4.3	Calcul des B-Splines . . . . .	17
4.3.1	Détermination des polynômes . . . . .	17
4.3.2	Détermination des coefficients . . . . .	18
4.4	Tracé de la trajectoire . . . . .	18
<b>5</b>	<b>Localisation du robot par rapport à la trajectoire</b>	<b>20</b>
5.1	Localisation du robot dans le repère monde . . . . .	20
5.2	Calcul de la distance entre le robot et la trajectoire . . . . .	20
5.3	Calcul de l'angle entre le robot et la trajectoire . . . . .	21
<b>6</b>	<b>Conclusion</b>	<b>22</b>
<b>7</b>	<b>Bibliographie</b>	<b>23</b>

# Chapitre 1

## Introduction

### 1.1 Présentation

Dans le cadre des TPs portant sur la robotique mobile, Mr LENGAGNE veut mettre en place un TP de suivi de ligne virtuelle. Pour cela, il a besoin que le robot utilisé soit programmé pour générer une trajectoire à suivre et informer de sa position par rapport à cette trajectoire. Le but du TP est donc de suivre la trajectoire générée en minimisant les différences de position entre le robot et la trajectoire.

Pour ne pas dégrader le lieu du TP, il est nécessaire qu'aucun tracé ne soit réalisé sur le parcours. Les points passage sont donc générés par des QR codes placés le long du parcours.

Toute la programmation est effectuée sous l'environnement ROS et le robot utilisé est le LIMO de l'entreprise AgileX ( Fig. 1.1).



FIGURE 1.1 – Robot LIMO de AgileX

Ce robot est équipé d'une caméra RGB qui lui permet de détecter des QR codes et retourner les informations de position et l'ID de ces derniers.

## 1.2 Cahier des charges

Pour le bon déroulement du TP, il est demandé que la trajectoire soit continue en termes de position, de vitesse et d'accélération. Il faut donc que la trajectoire soit construite par des fonctions pouvant être dérivées deux fois.

Pour réaliser ce projet, il est nécessaire de poser les deux hypothèses suivantes :

H1 : Le robot doit avoir la vision sur au moins deux QR codes successifs pour pouvoir placer un nouveau point de passage.

H2 : Le premier QR code aperçu par le robot est désigné comme l'origine du référentiel monde.

Dans le cas où le robot ne voie aucun ou un seul QR code, le programme ne doit pas s'arrêter.

Ces hypothèses ont pour but de trouver les points de passage l'un après l'autre pour que chaque point soit déclaré en fonction du point précédent.

## 1.3 Protocole

Le but de ce projet est de retourner la position en translation et en rotation du robot par rapport à la trajectoire générée ( Fig. 1.2). Pour cela, il faut définir un protocole pour mener à bien cette expérience.

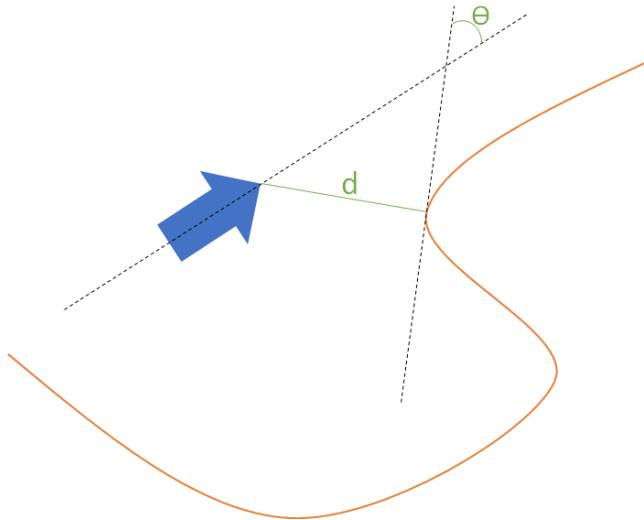


FIGURE 1.2 – Schéma montrant les objectifs du projet

Tout d'abord, le robot est commandé par un joystick pour se déplacer le long du parcours. Il faut que le robot se déplace de sorte à ce qu'il ai toujours la vision sur au moins deux QR codes sucééssifs.

Au départ, la première fois que le robot détecte un QR code, il considère ce dernier comme l'origine du référentiel monde.

Les QR codes étant toujours aperçus par paire (Fig. 1.3), il est possible d'exprimer la position de l'un par rapport à l'autre. Mais il est également possible de l'exprimer dans le repère monde par une formule de récurrence.



FIGURE 1.3 – Détection des QR codes

Une fois que la position du QR code est déterminée, il faut en déduire la position du point de passage donné par celui-ci. Selon le QR code, un ID est transmis et est associé à une translation dans le repère du QR code. Ce qui permet de calculer la position du point dans le repère monde (Fig. 1.4).

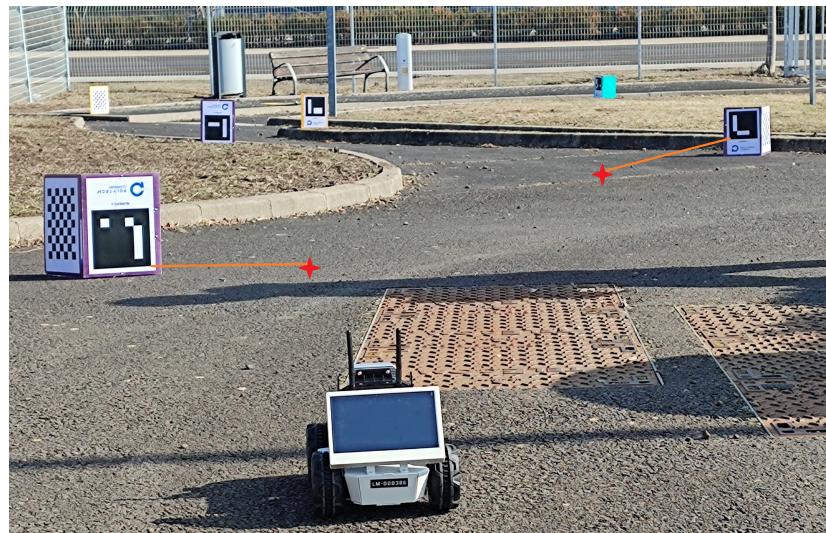


FIGURE 1.4 – Localisation des points de passage

Puis à partir de ces points, la trajectoire est générée par le biais de fonctions polynômiales d'ordre supérieur ou égal à 3 pour respecter la condition de continuité (Fig. 1.5).

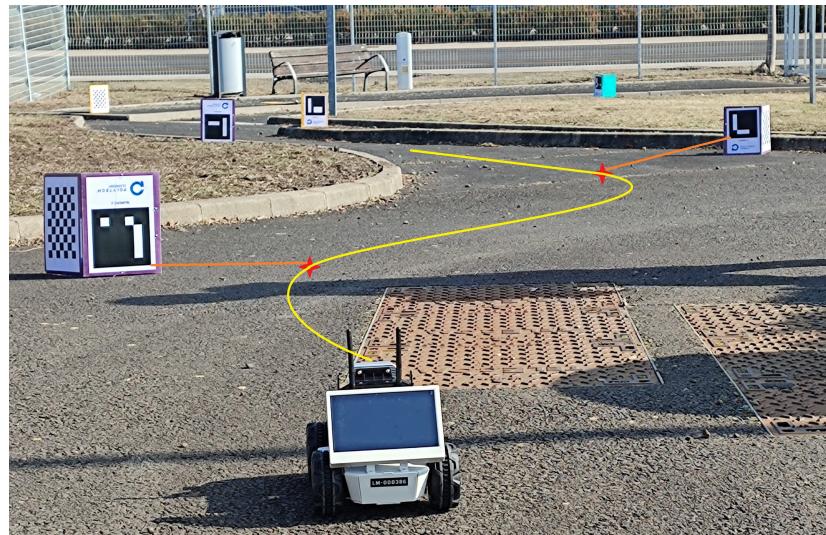


FIGURE 1.5 – Tracé de la trajectoire

Donc, la trajectoire se met à jour à chaque nouvelle détection d'un QR code inconnu.

Enfin le robot analyse chaque point de la courbe générée et renvoie sa position par rapport à la trajectoire.

## Chapitre 2

# Architecture de la programmation

La programmation sous l'environnement ROS nécessite de mettre en place des nœuds (programmes) qui souscrivent et publient sur différents topics. L'architecture choisie est composée de quatre nœuds(Fig. 2.1)

Le premier nœud est nommé filter-frame, il a comme topic d'entrée les messages que le robot envoie à chaque fois qu'il aperçoit un QR code. Ces messages sont composés de la position du QR code dans le repère de la caméra et de son ID. Le nœud renvoie en sortie la position du QR code et celle du robot dans le repère monde.

Le deuxième noeud est appelé waypoint. Il a pour entrée la position du QR code dans le repère monde. Il renvoie en sortie la position du point de passage associé au QR code.

Le troisième nœud appelé trajectory a pour entrée les points de passage, et génère une trajectoire passant par tous ces points. Il renvoie les coordonnées de chaque points composants la trajectoire en fonction du déplacement s le long de cette dernière.

Le dernier nœud est nommé localisation. Il prend comme entrées les points composants la trajectoire mais aussi la position du robot dans le repère monde. Il renvoie en sortie la localisation du robot par rapport à la trajectoire, c'est-à-dire la distance et l'angle entre eux.

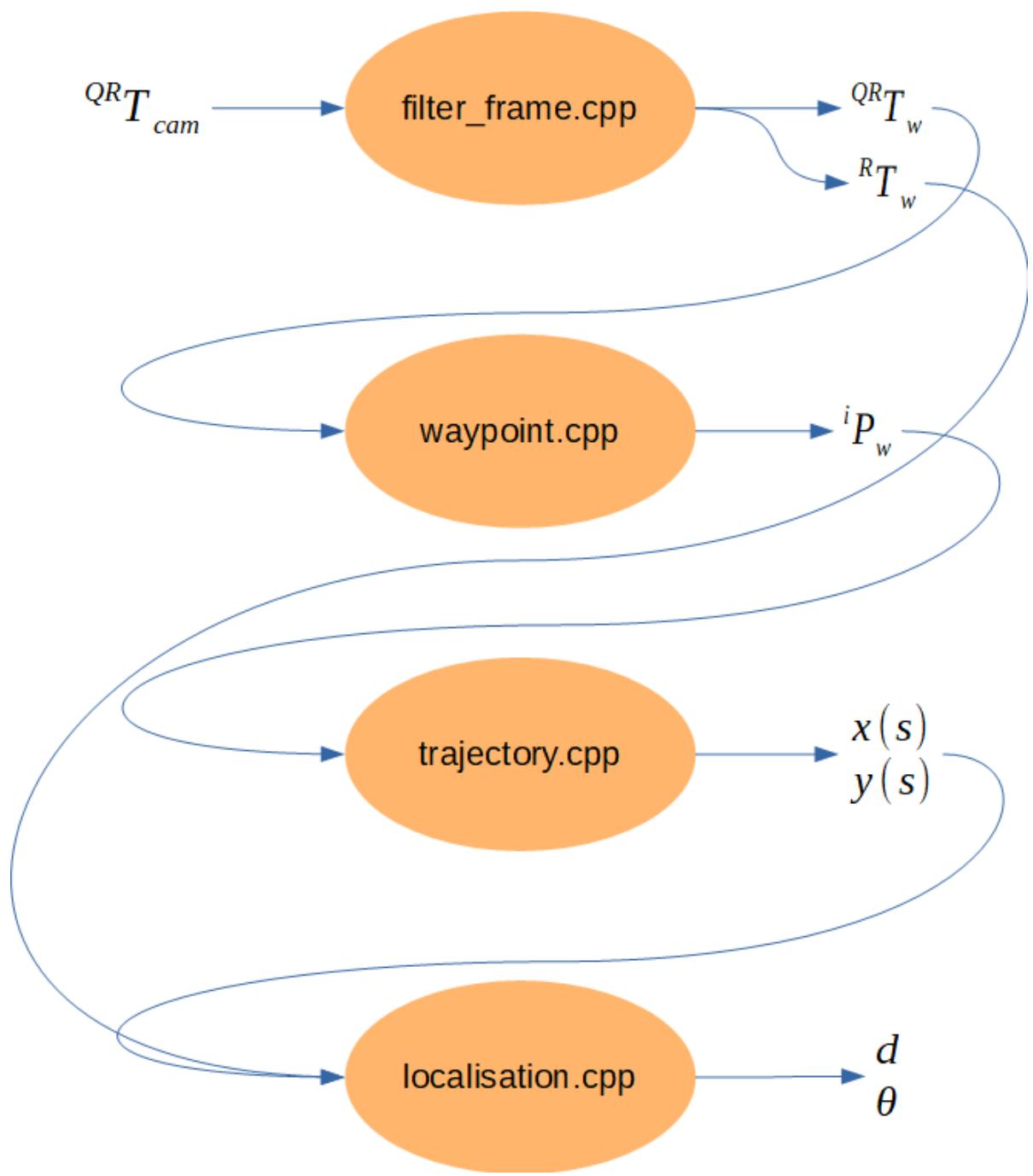


FIGURE 2.1 – Architecture de la programmation

# Chapitre 3

## Définition des points de passage

En se déplaçant le long de la piste, le robot renvoie la position des QR codes qu'il aperçoit ainsi que leur ID. Cette position est donnée dans le repère de la caméra du robot.

Quand plusieurs QR codes sont aperçus, le robot envoie un message sous la forme d'un vecteur. Seules les positions des deux QR codes les plus proches sont collectées. Le but est d'exprimer la position du QR code d'indice  $i$  dans le repère d'indice  $i-1$ . Si ce résultat est égal à un résultat déjà obtenu, cela signifie que la position de ce QR code a déjà été définie, il n'est pas accepté dans la base de données. Sinon cela veut dire que c'est un nouveau QR code et sa position est entrée dans la base de données.

### 3.1 Localisation des QR codes

Une fois que le robot a aperçu et déclaré le premier QR code, il faut ensuite qu'il aperçoive le second QR code tout en ayant la vision sur le premier.

Pour chaque QR code détecté, la position renvoyée par le robot contient deux types de variables :

La position du repère du QR code dans le repère de la caméra sous forme coordonnées cartésiennes :

- x
- y
- z

La rotation du repère du QR code par rapport au repère de la caméra sous forme de quaternions :

- roll : noté  $\alpha$
- pitch : noté  $\beta$
- yaw : noté  $\theta$

Avec ces six variables, il est possible de construire la matrice de transformation du repère du QR code dans le repère de la caméra.

$${}^{QR}T_{camera} = \begin{pmatrix} R_{3 \times 3} & t_{3 \times 1} \\ 0 & 1 \end{pmatrix} \quad (3.1)$$

Il est fait ainsi pour les deux QR codes. Il est donc obtenu :

$${}^{QR_i}T_{camera} \text{ et } {}^{QR_{i+1}}T_{camera} \quad (3.2)$$

Pour faciliter les calculs par la suite, le problème sera posé dans un plan pour que seule la rotation en z soit valable.

La matrice de rotation s'exprime donc :

$$R_{3 \times 3} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.3)$$

La matrice de translation est donnée par :

$$t_{3 \times 1} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \quad (3.4)$$

Il est maintenant nécessaire de définir la matrice de transformation inverse du repère du QR code 1 :

$${}^{QR_i}T_{camera}^{-1} = {}^{camera}T_{QR_i} = \begin{pmatrix} R^T & -R^T * t \\ 0 & 1 \end{pmatrix} \quad (3.5)$$

Grâce à c'est deux matrices de transformation, il est possible de déterminer la position de tous les QR codes dans le référentiel monde.

$${}^{QR_{i+1}}T_{monde} = {}^{QR_{i+1}}T_{camera} \times {}^{camera}T_{QR_i} \times {}^{QR_i}T_{monde} \quad (3.6)$$

Cela est possible grâce à l'hypothèse du premier QR code comme origine du repère. La matrice de transformation de ce dernier est :

$${}^{QR_0}T_{monde} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.7)$$

Les positions de tous les QR codes peuvent maintenant être déduites de toutes ces matrices de transformation. Ils ainsi un nuage de points (Fig. 3.1)

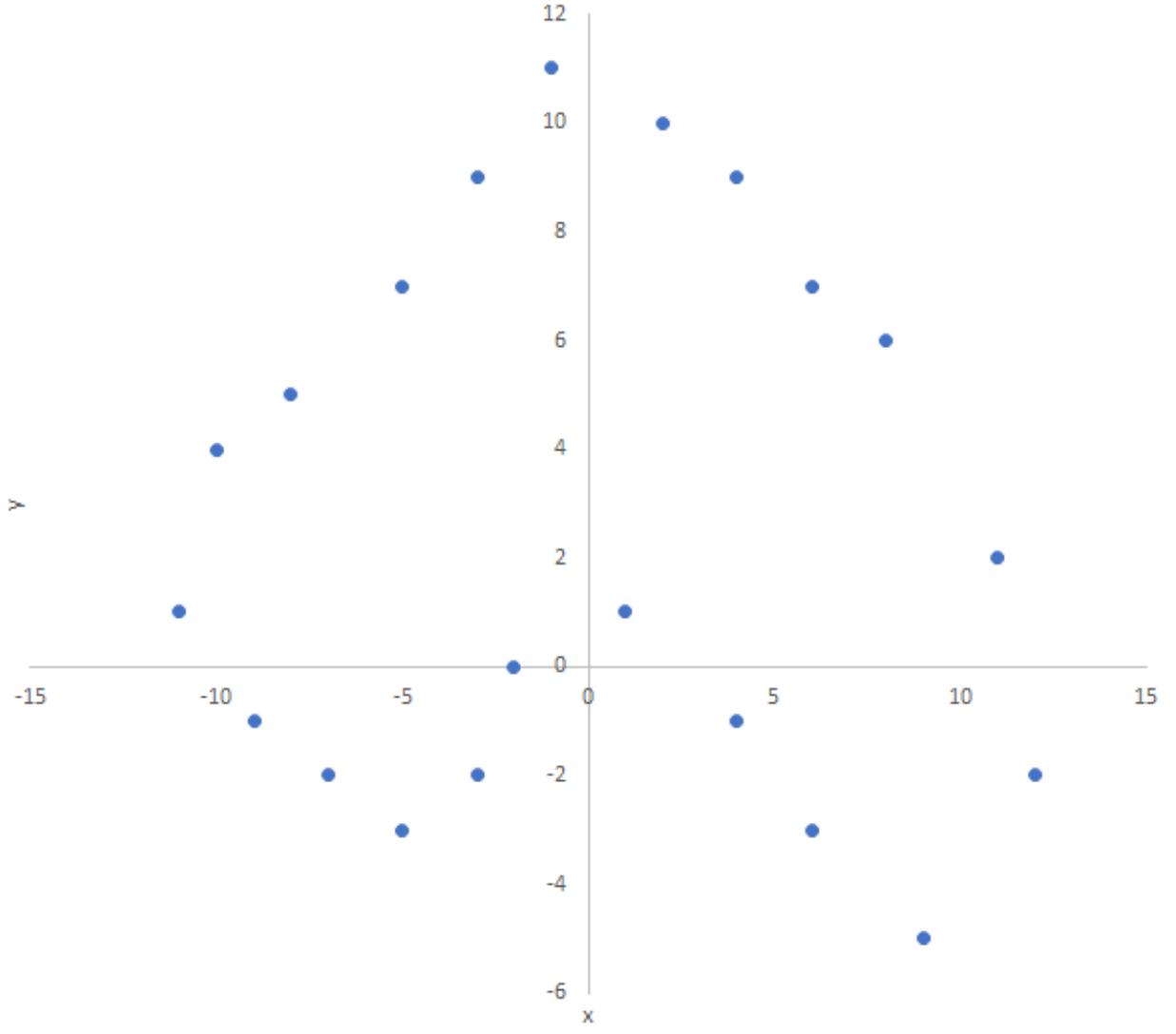


FIGURE 3.1 – Localisation des QR code

## 3.2 Localisation des points de passage

Une fois que la position d'un QR code est calculée, il faut maintenant trouver la position du point de passage donné par le QR code. Pour cela, chaque QR code possède un ID qui est associée à une matrice de transformation en translation pure, du point dans le repère du QR code :

$${}^{point}T_{QRcode} = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.8)$$

Il est maintenant possible de calculer la matrice de transformation du point dans le repère monde :

$${}^{point}T_{monde} = {}^{point}T_{QRcode} \times {}^{QRcode}T_{monde} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \quad (3.9)$$

Cette matrice donne les coordonnées du point dans sa partie translation :

$$t = \begin{pmatrix} x_{pt} \\ y_{pt} \end{pmatrix} \quad (3.10)$$

Les points de passage forment un nuage de points différent de celui des QR codes (Fig. 3.2)

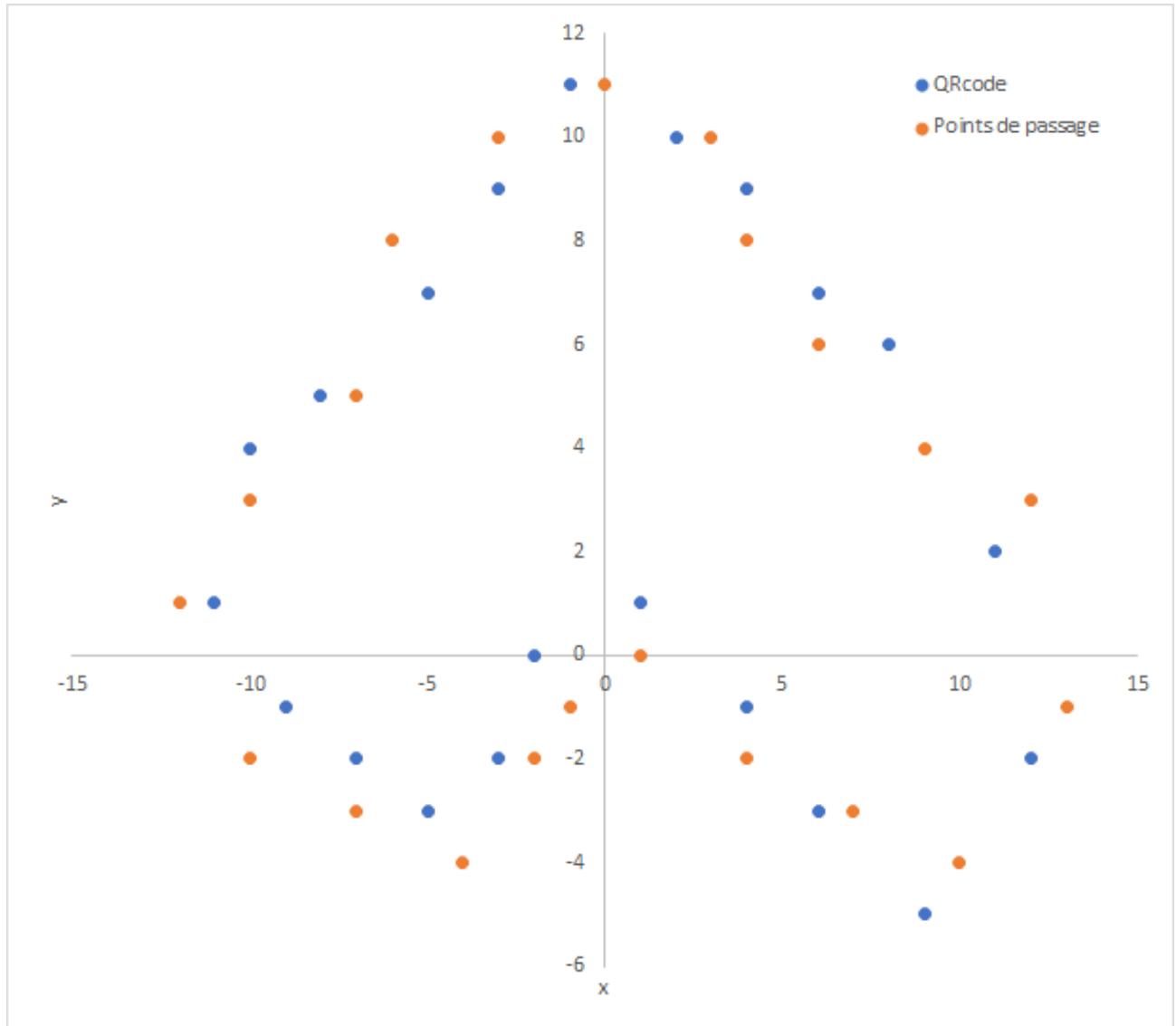


FIGURE 3.2 – Tracé des points de passages

# Chapitre 4

## Calcul de la trajectoire

### 4.1 Choix de la méthode

D'après les indications données, la trajectoire doit être continue en position, en vitesse et en accélération. La fonction définissant cette trajectoire doit donc être dérivable deux fois avec la dérivée seconde d'ordre 1 ou plus. La fonction choisie est donc un polynôme d'ordre 3 ou plus.

Si seul un polynôme est choisi, l'ordre de celui dépend du nombre de points de passage, qui peut être très grand, ce qui rendrait la programmation très fastidieuse. Pour résoudre ce problème, il faut utiliser les courbes B-Splines (Fig. 4.1). Elles permettent de définir chaque point de la courbe par rapport à des points de contrôle.

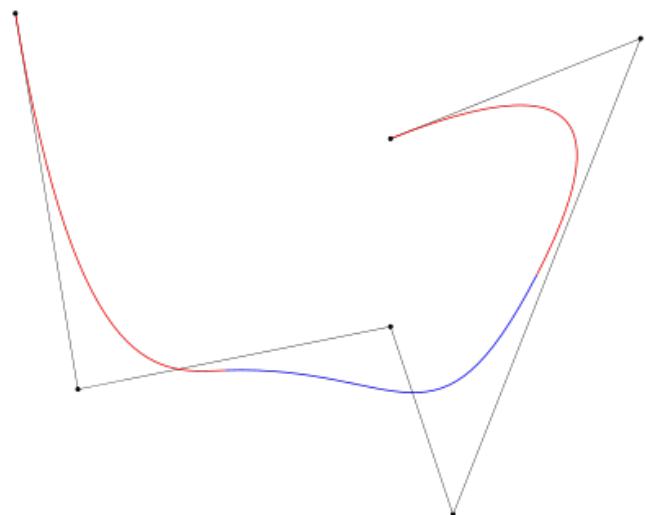


FIGURE 4.1 – exemple de courbe construite à partir de B-Splines

Les B-Splines permettent de définir une courbe devant passer par un grand nombre de points à l'aide de plusieurs polynômes de faibles degrés. Cela permet donc de rendre les calculs moins fastidieux et plus rapides que pour un seul polynôme d'ordre très grand.

Dans le cas présent, les expressions des courbes B-Splines sont définies à l'aide de l'algorithme de Cox-De-Boor.

## 4.2 Définition des paramètres

Pour utiliser l'algorithme de Cox-De-Boor, il faut définir plusieurs paramètres préalablement.

Le nombre de points de passage est noté N et chacun de ces points se note :

$$t_i = \begin{pmatrix} x_i \\ y_i \\ 0 \end{pmatrix} \text{ avec } i \text{ allant de } 0 \text{ à } N - 1 \quad (4.1)$$

L'algorithme de Cox-De-Boor travail sur des intervalles construits entre des points de contrôle, noté Pi. Pour simplifier les calculs, le nombre de points de contrôle est fixé à N, et comme la trajectoire est fermé, le nombre d'intervalle est également N.

Il faut également définir l'ordre de B-Splines, il est choisi à p=3, qui est le plus petit ordre possible dans ce cas, pour répondre au critère de continuité.

## 4.3 Calcul des B-Splines

Les B-Splines sont définies par l'équation suivante :

$$S(x) = \sum_{i=k-p}^k P_i B_{i,p}(x) \text{ avec } k \text{ l'indice de l'intervalle} \quad (4.2)$$

D'après cette équation, la courbe est définie entre chaque intervalle par le polynôme de même indice que l'intervalle ainsi que les p polynômes d'ordre p précédents.

La trajectoire étant une boucle, la courbe est définie par des polynômes d'indice négatif. Ces derniers sont alors en N-i pour que la matrice M soit construite correctement.

Il faut donc déterminer les coefficients P ainsi que les polynômes B(x).

### 4.3.1 Détermination des polynômes

Il faut d'abord calculer les polynômes B(x) avec l'algorithme de Cox-De-Boor. Il s'agit d'une formule de récurrence :

$$B_{i,0}(x) = \begin{cases} 1 & \text{si } x \in [t_i, t_{i+1}] \\ 0 & \text{sinon} \end{cases} \quad (4.3)$$

Puis :

$$B_{i,p}(x) = \frac{x - t_i}{t_{i+p} - t_i} \times B_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} \times B_{i+1,p-1}(x) \quad (4.4)$$

Ces polynômes sont ensuite rangés dans une matrice M, sous la forme :

$$M = \begin{pmatrix} B_{0,p}(x_0) & 0 & 0 & \cdots & 0 & B_{N-3,p}(x_0) & B_{N-2,p}(x_0) & B_{N-1,p}(x_0) \\ B_{0,p}(x_1) & B_{1,p}(x_1) & 0 & \cdots & 0 & 0 & B_{N-2,p}(x_1) & B_{N-1,p}(x_1) \\ B_{0,p}(x_2) & B_{1,p}(x_2) & B_{2,p}(x_2) & \cdots & 0 & 0 & 0 & B_{N-1,p}(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & B_{N-4,p}(x_{-1}) & B_{N-3,p}(x_{N-1}) & B_{N-2,p}(x_{N-1}) & B_{N-1,p}(x_{N-1}) \end{pmatrix} \quad (4.5)$$

$$M \in \mathbb{R}^{N \times N} \quad (4.6)$$

### 4.3.2 Détermination des coefficients

Une fois les polynômes définis, il maintenant possible de déterminer les coefficients P. Pour cela, il faut résoudre l'équation qui définit la courbe :

$$Y = M \times \theta \Rightarrow \theta = Y \times M^{-1} \quad \text{avec} \quad \theta = \begin{pmatrix} P_0 \\ P_1 \\ \vdots \\ P_i \\ \vdots \\ P_{N-1} \end{pmatrix} \quad (4.7)$$

Le vecteur Y est de taille N et contient les ordonnées des points de passage.

Ces coefficients sont appelés les points de contrôle des B-Splines. Ce sont eux qui définissent les courbes B-Splines composant la trajectoire.

## 4.4 Tracé de la trajectoire

Il est maintenant possible de tracer la trajectoire( Fig. 4.2).

Ainsi, chaque points de la trajectoire est défini par ces coordonnées en x et en y :

$$P(s) = \begin{pmatrix} x(s) \\ y(s) \end{pmatrix} \quad (4.8)$$

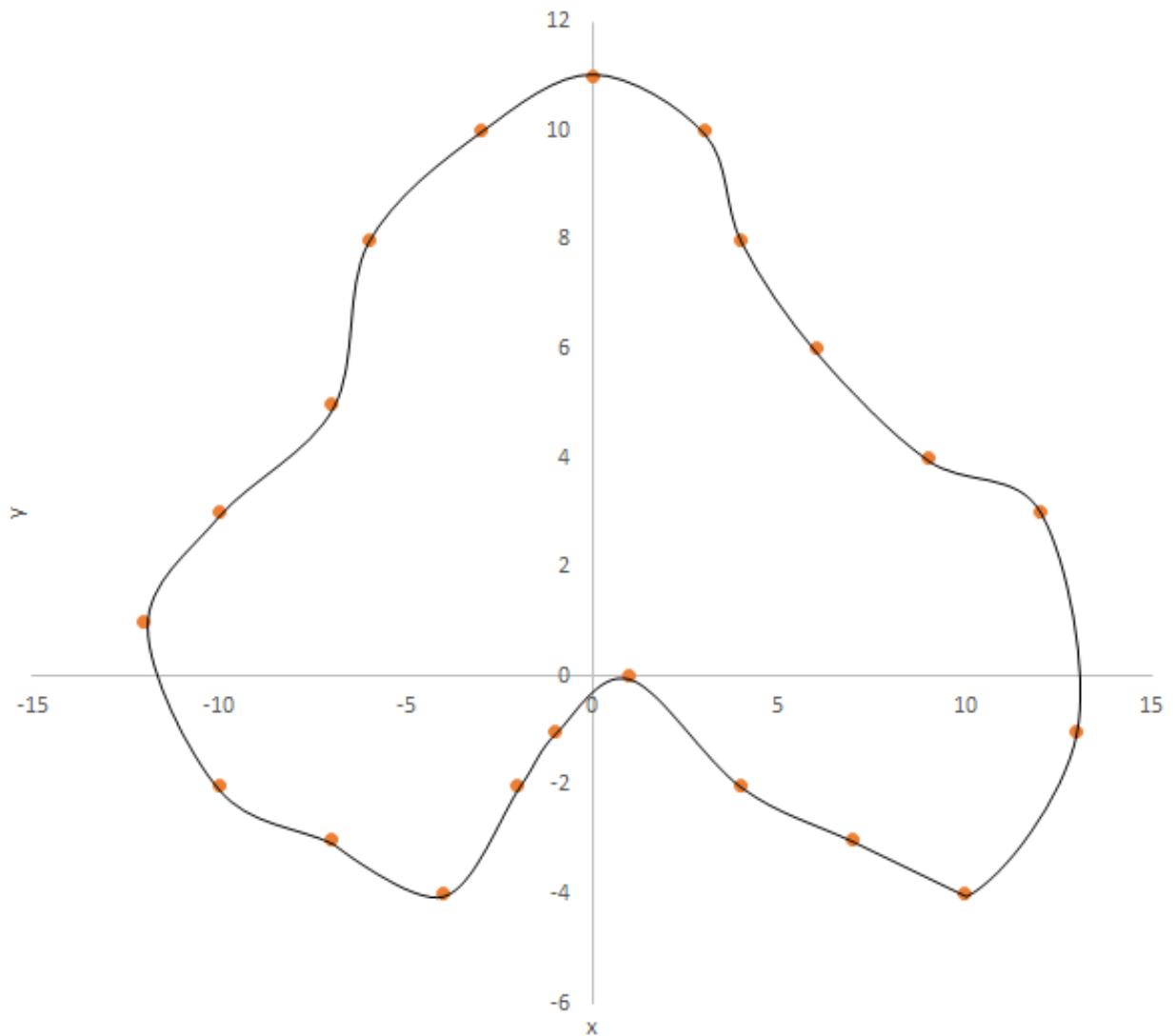


FIGURE 4.2 – Tracé de la trajectoire

# Chapitre 5

## Localisation du robot par rapport à la trajectoire

### 5.1 Localisation du robot dans le repère monde

La trajectoire étant définie dans le repère monde, il faut donc que le robot se situe lui aussi dans ce repère.

Pour cela, le robot doit avoir la vision sur un QR code dont les coordonnées sont connues, il peut donc retourner la position du QR code dans le repère de la caméra.

Il faut ensuite construire la matrice de transformation de la caméra dans le repère du QR code et résoudre l'équation suivante :

$${}^{camera}T_{monde} = {}^{camera}T_{QRi} \times {}^{QRi}T_{monde} \quad (5.1)$$

La position du robot peut donc être déduite de cette matrice et est définie selon trois paramètres :

- x : l'abscisse du robot dans le repère monde ;
- y : l'ordonnée du robot dans le repère monde ;
- θ : l'angle de rotation autour de l'axe z du robot dans le repère monde.

Maintenant que la trajectoire et la position du robot sont définies dans le même référentiel, il est possible de situer le robot par rapport à chaque point de la trajectoire.

### 5.2 Calcul de la distance entre le robot et la trajectoire

Une fois la position du robot connue, il faut trouver quel est le point D, situé sur la courbe, le plus proche du robot ( Fig. 5.1). Pour cela, Le robot passe chaque point compris dans l'intervalle où se trouve le robot dans un algorithme de minimisation de la distance. L'intervalle est donné par le QR code détecté.

L'équation qui permet cela est donnée par :

$$d = \sqrt{(x_{robot} - x_{point})^2 + (y_{robot} - y_{point})^2} \quad (5.2)$$

Au point où d est minimisé, les coordonnées du point sont copiées dans le point D.

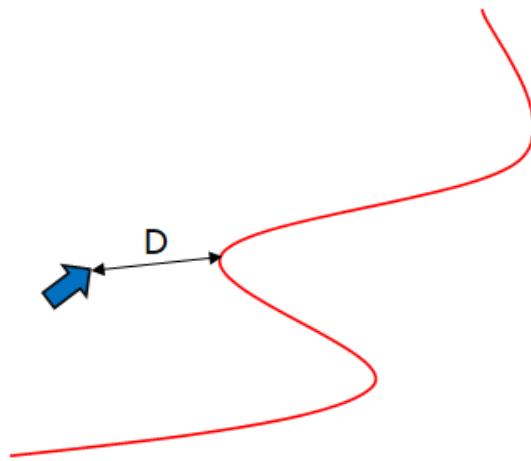


FIGURE 5.1 – Distance minimum entre le robot et la trajectoire

### 5.3 Calcul de l'angle entre le robot et la trajectoire

Le point le plus proche étant maintenant connu, il faut déterminer la différence de rotation entre le robot et l'angle qu'il doit avoir en suivant la trajectoire ( Fig. 5.2).

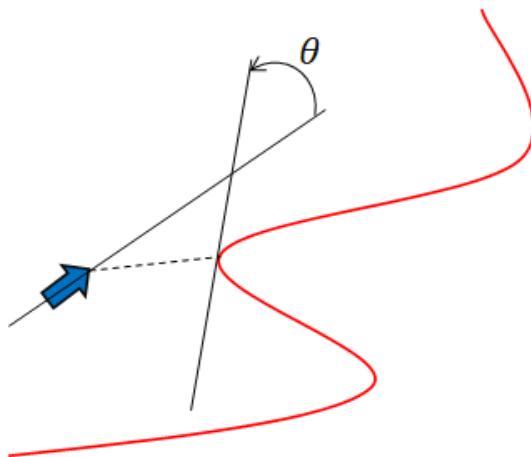


FIGURE 5.2 – Différence d'angle entre le robot et la trajectoire

Pour cela, il faut calculer la tangente à la courbe en le point le plus proche :

$$T_D(x) = f'(x_D) \times (x - x_D) + f(x_D) \quad (5.3)$$

L'angle de la droite formée par la tangente est ensuite calculé :

$$\theta_D = \arctan \frac{T_D(x_1) - T_D(x_2)}{x_1 - x_2} \quad (5.4)$$

Pour finir, il faut calculer la différence d'angle :

$$\theta = \theta_{robot} - \theta_D \quad (5.5)$$

En combinant  $d$  et  $\theta$ , il est obtenu la localisation du robot par rapport à la trajectoire.

# Chapitre 6

## Conclusion

Dans le cadre de ce projet, j'ai mené un travail de recherche pour mettre en place un ensemble de paquet ROS destinés à être utilisé lors d'un TP sur le suivi de trajectoire. J'ai également pu commencer à codés les différents nœuds de ces paquets.

À l'heure actuelle, le robot est capable de générer une trajectoire à partir des QR codes aperçus. Ce qui équivaut à environ 65 pourcents du travail de programmation.

Pour l'avenir, il reste certains codes à programmer pour finir ce projet. Notamment la partie sur la localisation du robot par rapport à la trajectoire générée. Il faut également traiter le cas où la trajectoire contient une boucle. Pour finir, il faut codé la position du point de passage dans le repère du QR code selon l'ID de ce dernier car actuellement cette position est la même pour tous les QR codes. Il est estimé un travail de 40 à 50 heures pour réalisé cela.

A titre personnel, ce projet m'a permis de revoir et d'approfondir des notions acquises lors de ma formation, puis de les implémenter dans cadre plus visuel. Cela m'a également fait comprendre comment mener à bien un projet.

Je tiens encore à remercier les personnes qui m'ont aidé durant ce projet.

# Chapitre 7

## Bibliographie

Lien vers le dépôt github :

<https://github.com/gaetanlembert/Projet-planification-de-trajectoire>

Lien vers le projet initial :

<https://github.com/loyzjve/Suivi-de-ligne-virtuelle-par-QR-Code>

Algorithme de Cox-De-Boor :

<https://pages.mtu.edu/shene/COURSES/cs3621/NOTES/spline/de-Boor.html>

Robot LIMO AgileX :

<https://global.agilex.ai/products/limo>