RAPPORT DE PROJET : LOCALISATION DE SOURCES SONORES

Quentin Leblanc

ENSIBS

Table des matières

Table des figures

I. Introduction

L'objectif de ce projet est de créer un programme permettant la localisation de sources sonores. Par la suite, il devra être implanté sur de petites bases robotiques, comme celles qui sont utilisées lors des travaux pratiques à l'ENSIBS.

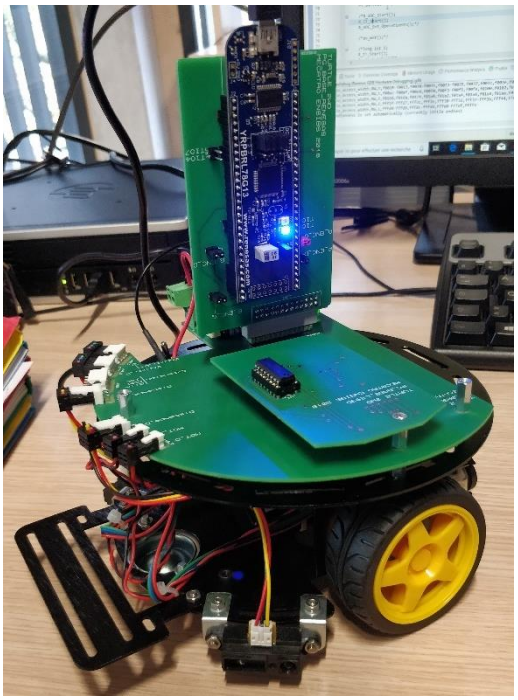

Figure 1 : Base robotique de l'ENSIBS

L'enjeu ici est d'obtenir un programme et une configuration technique assez performants pour la localisation de sources sonores, tout en restants assez légers pour être utilisés sur des bases aux ressources limitées.

II. Etat de l'art

1. Principes de captation

Il existe deux méthodes de captation des sons différentes :

- Biomimétique : deux à quatre microphones afin de se rapprocher du fonctionnement humain(bio-inspirée)
- Antennerie : Un grand nombre de microphones placés sur une antenne.

a. Biomimétique

Cette méthode utilise par exemple l'enregistrement binaural, technique de spatialisation sonore la plus proche de l'écoute naturelle. Cette méthode demande peu de microphones ce qui permet de l'implanter plus aisément sur de petites bases. Cependant, la programmation peut être longue et ce principe de fonctionnement trouve rapidement ses limites lorsqu'il se trouve dans un environnement bruité.

b. Antennerie

Cette méthode est basée sur des méthodes de traitement du signal prenant en compte le bruit. Pour obtenir une antenne capable d'écouter finement dans une direction, celle-ci doit être de grande taille (proche du mètre) et comporter un nombre élevé de microphones. Mais malgré cela, son diagramme évolue selon la fréquence et perd énormément en résolution pour les plus basses fréquences.

2. Traitements possibles des sons captés

Les méthodes de détermination de la position angulaire de sources se basent sur une observation du champ acoustique, avec au minimum deux capteurs. Ces méthodes se distinguent en plusieurs classes :

1. La mesure de la différence d'intensité sonore (l'amplitude des signaux) entre les capteurs. C'est la solution la plus simple. Elle souffre de plusieurs limitations : les micros doivent avoir la même sensibilité, les pré-amplis les mêmes gains, et surtout lorsque la source sonore est éloignée la différence de trajet et donc la différence d'atténuation entre le micro de gauche et de droite est négligeable.

2. Certaines méthodes sont basées sur une mesure de différences de temps de trajet des ondes pour atteindre les différents capteurs (TDOA : Time Delay Of Arrival), soit la mesure du déphasage entre les ondes captées par les différents microphones. La précision de mesure peut être bien meilleure, même pour une source sonore éloignée. Cependant, la fréquence maximale du signal sonore est donnée par la demi-longueur d'onde qui sépare les deux microphones. Si la différence du trajet sonore entre le micro de gauche et le micro de droite est de 1cm, on a $\frac{340}{2\times1\times10^{-2}} = 17\,000\ Hz$. La fréquence minimale est fixée par la résolution du convertisseur Analogique-numérique et le rapport S/N des micros avec leurs pré-amplis.

3. Les méthodes basées sur la formation de voies estiment parmi plusieurs directions possibles de la source celle qui est la plus probable. La résolution se fait généralement dans le domaine fréquentiel, en utilisant des méthodes de type Espérance Maximisation ou des méthodes de sous-espace.

4. Enfin certaines méthodes s'aident de la présence dans le voisinage des capteurs d'un objet diffractant dont la fonction de transfert directionnelle est connue (HRTF : Head-Related Transfer Function) pour l'interprétation de la différence des signaux à plusieurs capteurs.

III. Solutions théoriques

1. Localisation à 180°

La localisation du son utilise la comparaison des signaux perçus par plusieurs microphones, que ce soit pour comparer l'intensité sonore ou la différence des temps d'arrivée. Cela implique donc une utilisation de deux microphones au minimum. Dans un premier temps, le choix s'est donc porté sur l'utilisation de seulement deux microphones.

La localisation du son se fait en trois étapes au sein du programme :

1. Acquisition des signaux S1 et S2 (S1 le son au micro de gauche et S2 le son au micro droit).
2. Intercorrélation de S1 et S2 : afin de retrouver le motif S2 au sein du signal S1, il faut calculer la distance entre les deux signaux, mais avec un paramètre de décalage τ.
3. Recherche temporelle du maximum de l'intercorrélation afin de déterminer τ.
4. Calcul de la direction d'arrivée grâce à la formule : $\theta = acos(\frac{v \times \tau}{D})$

a. Acquisition des signaux

Le son capté par les microphones doit être échantillonné. Pour cela, le critère de Shannon-Nyquist permet d'obtenir : $F_e > 2 \times (F_{max} - F_{min})$
En partant du principe que le robot doit localiser les sources sonores uniquement audibles par l'oreille humaine, $F_{max} = 20\ kHz$ et $F_{min} = 20\ Hz$.
Alors $F_e > 40\ kHz$
Dans la bibliographie étudiée, la fréquence choisie est généralement de 44 100 Hz. Cela permet d'étendre le spectre jusqu'à 22 050 Hz afin d'éviter que les fréquences ne se coupent. C'est donc cette valeur qui a été utilisée comme fréquence d'échantillonnage dans le programme.

b. Calcul de la position angulaire de la source sonore

En prenant la situation suivante :

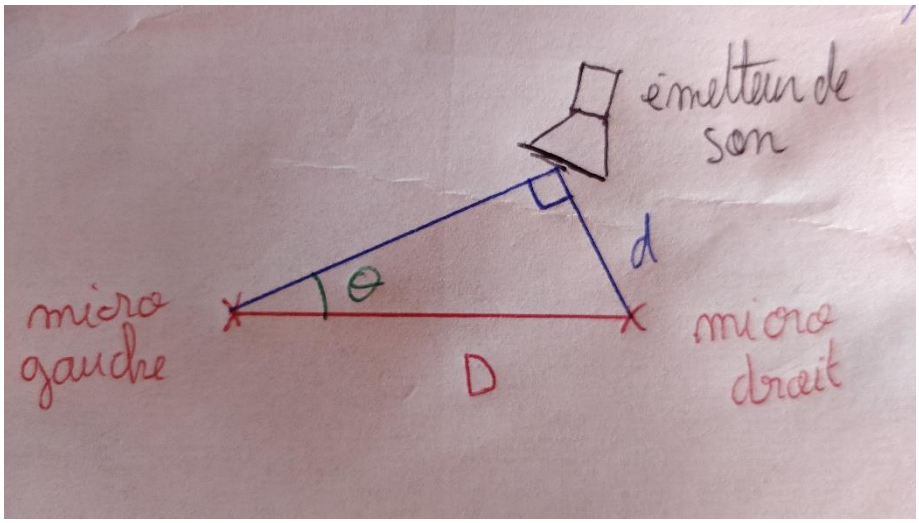

Figure 2 : Schéma d'une mise en situation pour le calcul de l'angle

Pour obtenir l'angle depuis lequel se situe la source sonore, on utilise deux formules :

$$\begin{cases} \sin\theta = \dfrac{d}{D} \\ v = \dfrac{d}{\tau} \end{cases} \rightarrow \begin{cases} \theta = \text{asin}(\dfrac{v \times \tau}{D}) \\ d = \tau \times v \end{cases}$$

On en déduit donc $\theta = asin(\frac{v \times \tau}{D})$. Il est donc nécessaire de connaître la vitesse du son (v), la distance entre les microphones (D) et le retard du son entre les deux microphones (τ).

2. Localisation à 360°

Pour le moment, le programme réalisé n'est capable de localiser les sources sonores qu'à 180°. En effet, l'utilisation de deux microphones ne permet de déterminer que si le son est plus proche de la droite ou de la gauche. Mais pour acquérir l'information à 360°, il est nécessaire d'utiliser plus de deux microphones. Les idées qui suivent ont été imaginées pour parvenir à une localisation à 360°.

a. Quatre microphones en croix

La première idée consiste à utiliser les couples de microphones en deux temps et séparément :

1. Les microphones de devant et de derrière vérifient si le son vient de derrière ou de devant
2. Les microphones de droite et de gauche localisent alors la direction du son en prenant en compte l'information donnée par les deux premiers microphones.

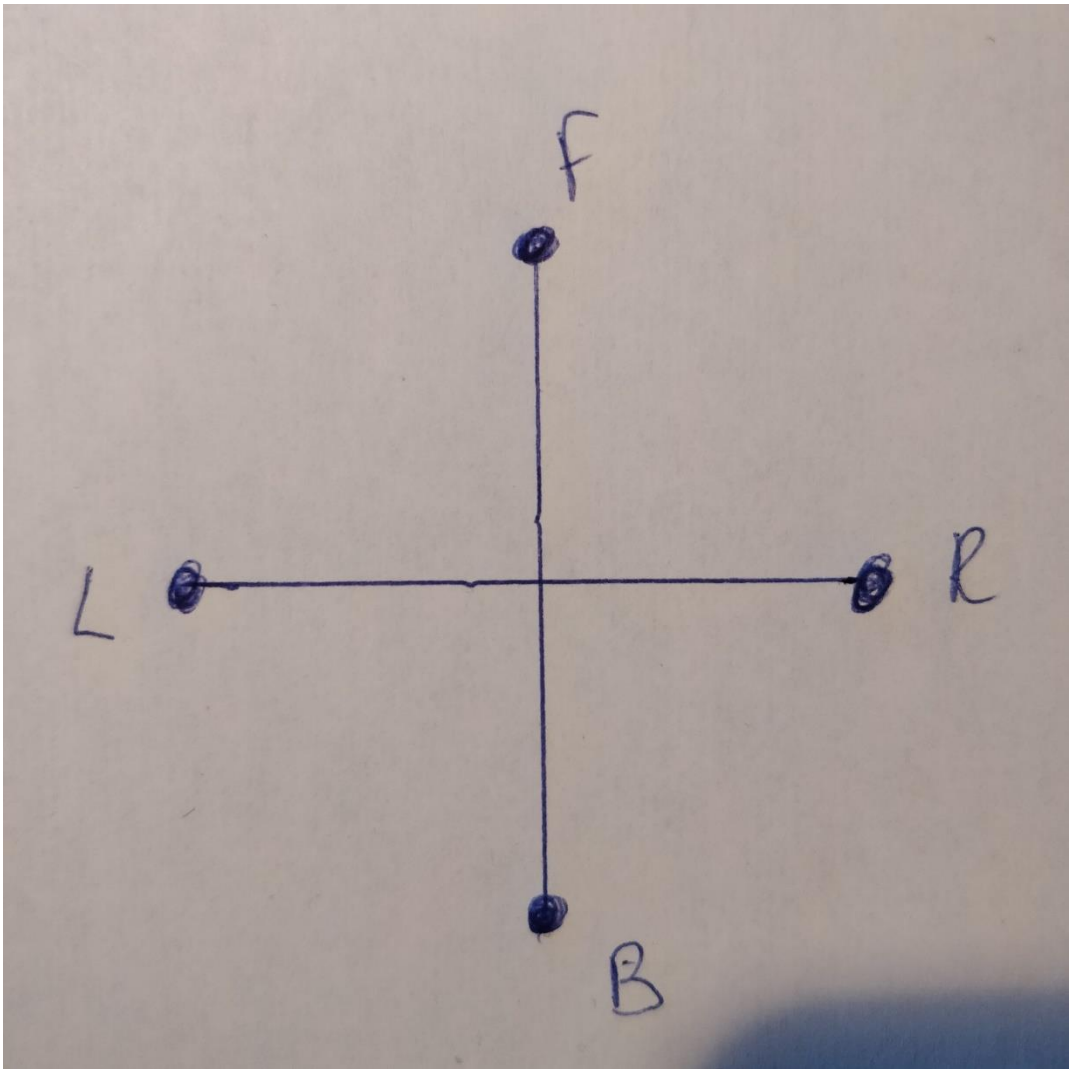

Figure 3: Solution en croix (L=Left, R=Right, F=front et B=Back)

b. Quatre microphones en losange

La seconde idée utilise les quatre microphones ensemble, et toujours en deux temps :

1. Le système repère les deux microphones les plus proches de la source sonore
2. Les deux microphones les plus proches localisent la direction de la source sonore par rapport à eux.

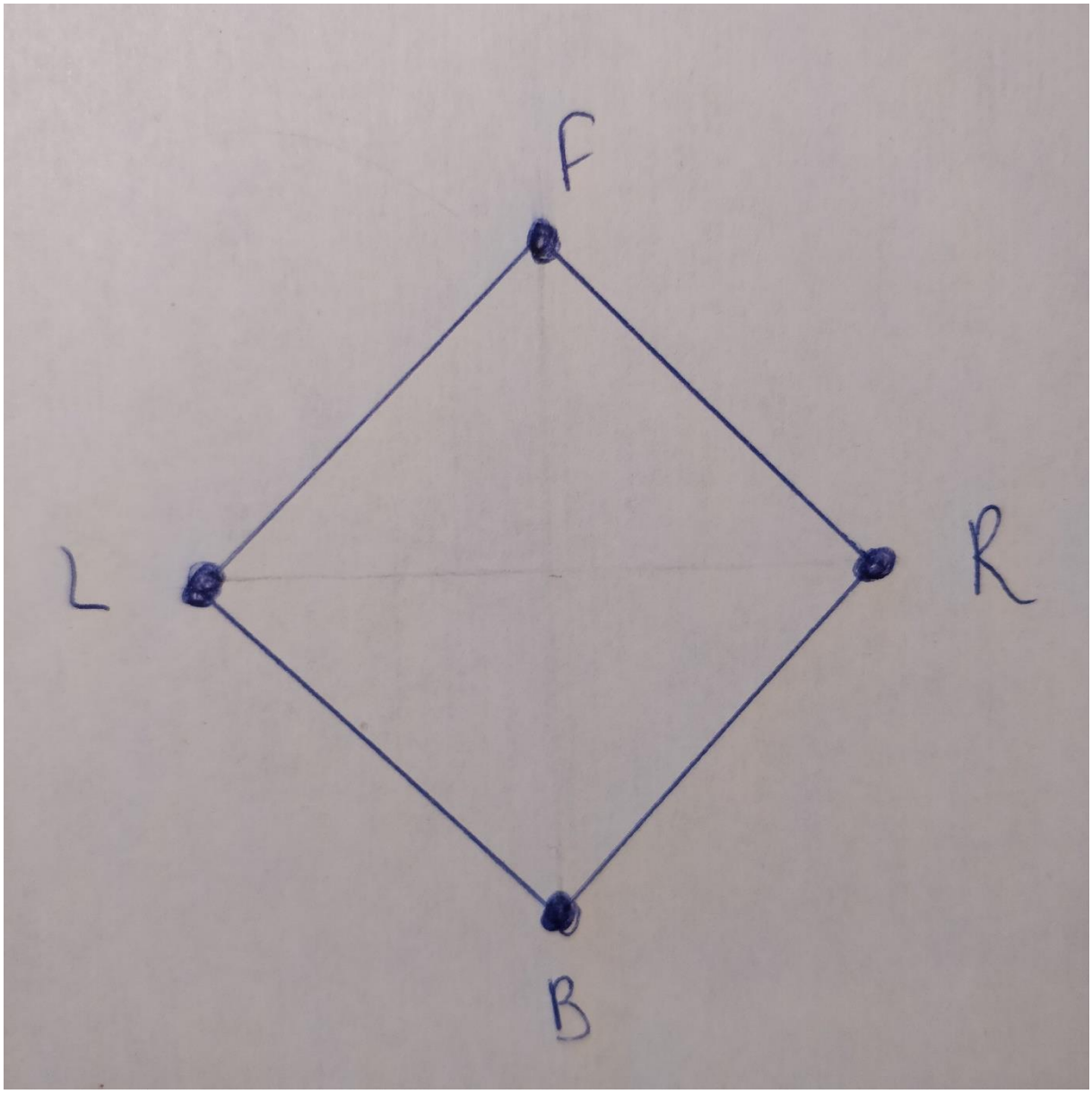

Figure 4 : Solution en losange (L=Left, R=Right, F=front et B=Back)

Les différences des temps d'arrivée entre les différents microphones sont comparées, et la plus faible différence est retenue. Cette solution découpe alors l'espace autour d'elle en quatre parties de 90°, ce qui permet de localiser une source sonore sur 360° autour du sujet.

c. Trois microphones équidistants

Ce concept reprend l'idée des quatre microphones en losange. Les deux microphones les plus proches sont utilisés pour déterminer la localisation de la source sonore. Une fois les deux micros déterminés, on cherche à déterminer la direction par rapport à ces deux micros. Toutefois, l'espace est divisé en trois parties de 120° autour du sujet (à la place des 90° possibles avec quatre microphones).

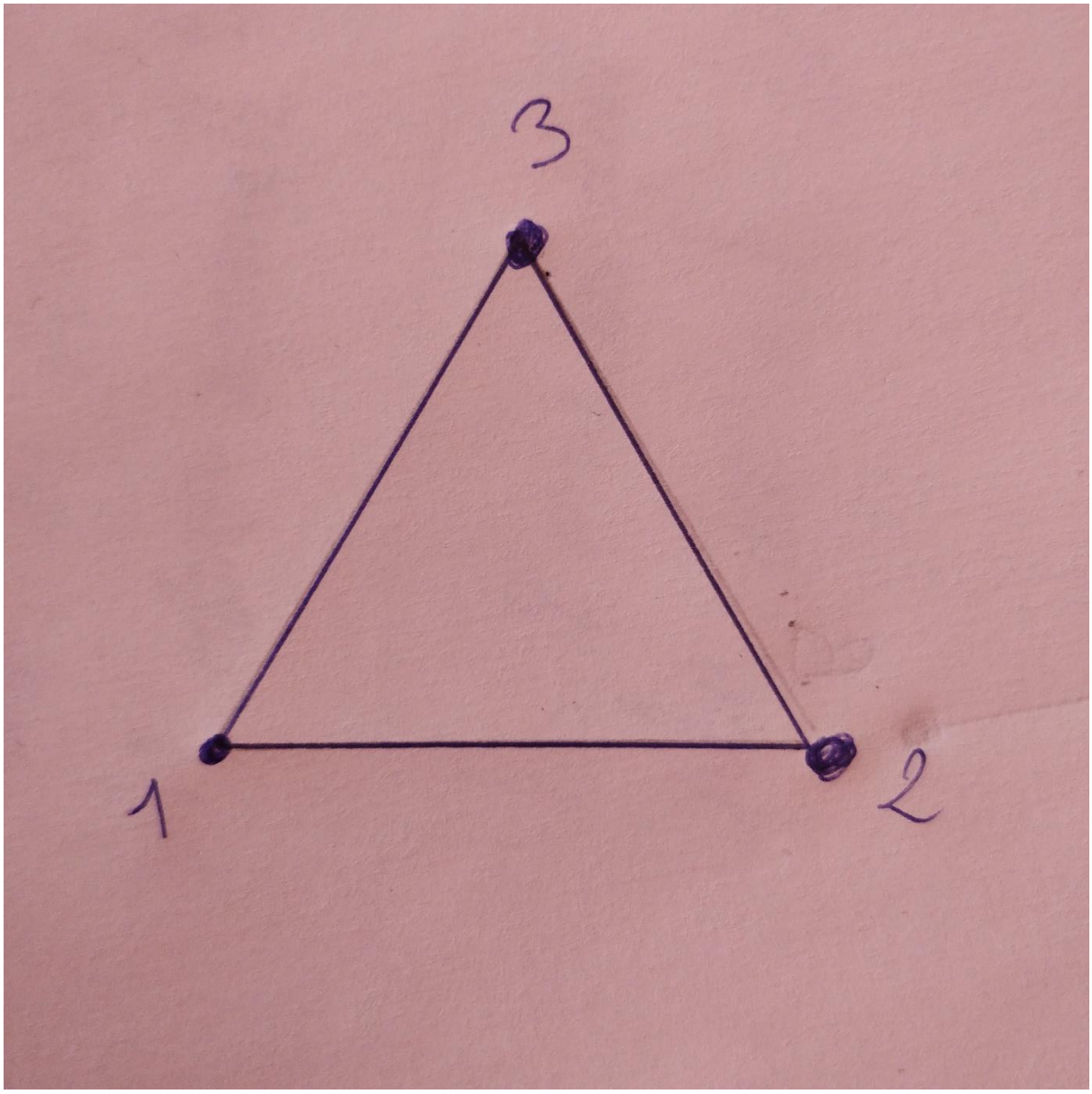

Figure 5 : Schéma de la solution à 3 microphones

L'avantage de cette solution est qu'elle demande moins de microphones que les deux premières. C'est donc cette dernière qui sera mise en œuvre.

IV. Mise en œuvre et résultats

1. Outils utilisés

Afin d'utiliser ce programme en robotique, il sera nécessaire d'utiliser ROS (Robot Operating System), un ensemble d'outils informatiques open source permettant de développer des logiciels pour la robotique. ROS ne fonctionnant que sous les environnements Linux ou MAC OS, le système d'exploitation utilisé sera Linux. Les besoins de l'école ont alors mené à l'utilisation de ROS Kinetic sur Ubuntu Xenial (16.04 LTS).

Les trois langages de programmation principaux de ROS sont Python, C++ et Lisp : C++ est plus rapide que Python et a l'avantage de prendre en charge les pointeurs avec notamment une bonne gestion de la mémoire. De plus, le C++ a l'avantage de pouvoir être développé sur des micro-processeurs. Ne connaissant pas le Lisp, le choix du langage s'est penché sur le C++.
Pour le bon fonctionnement, la bibliothèque *ALSA (libasound2-dev)* est nécessaire, ainsi que g++ version 9 car le code sera en C++ 11. Pour la compilation du programme, catkin_make sera utilisé.

2. Mise en œuvre d'une localisation à 180°

a. Acquisition des signaux

Pour acquérir les sons qui arrivent à la carte son, il existe deux possibilités :

- hw:x;y (*x=1 → carte 1, y=0 → périphérique 0)*
- plughw:x;y

Si *hw:x;y* est plus léger et permet d'accéder directement au hardware, *plughw:x;y* qui est plus "gourmand" car il traite le son, semble pouvoir fonctionner sans problème sur une Raspberry Pi.

N.B. : Dans la majorité des cas, x= 0 et y=0, mais il peut être nécessaire de modifier ces valeurs en fonction du matériel utilisé. Pour connaître ces valeurs il suffit d'utiliser la commande « aplay -l » et de trouver la carte et le périphérique à utiliser.

b. Calcul du niveau sonore

Le programme doit calculer l'angle de la source sonore par rapport au robot, mais pour contrer les bruits de fond non désirés, le niveau sonore est calculé pour mettre un seuil d'utilisation en dessous duquel il ne faut pas descendre.

c. Détermination du décalage temporelle

Afin de simplifier la programmation et d'alléger au maximum les ressources nécessaires au bon fonctionnement du programme, l'intercorrélation n'a pas été directement calculée : il a donc fallu trouver un autre moyen de déterminer le décalage temporel. L'intercorrélation permettant de déterminer le décalage τ pour lequel la ressemblance est maximum, le problème a été pensé différemment : trouver le décalage τ pour lequel la différence entre les deux signaux S1 et S2 est minimum. Alors on augmente le décalage et on calcule la valeur absolue de la différence entre le signal S1 reçu à gauche et le signal S2 reçu à droite. Tant que la différence diminue, on continue d'augmenter le décalage, sinon cela veut dire que le décalage τ permettant la meilleure ressemblance possible a été trouvée.

d. Calcul de l'angle

Le décalage temporel ayant été obtenu, la formule $\theta = asin(\frac{v \times \tau}{D})$ est utilisée pour obtenir l'angle.

3. Mise en œuvre d'une localisation à 360°

a. Utilisation des décalages temporels

Cette solution fonctionne selon le même principe que la précédente. La différence résidant dans l'utilisation des décalages temporels, les décalages entre les couples de microphones (1-2), (1-3) et (2-3) sont calculés en parallèle, puis ils sont comparés. Le plus faible décalage est alors utilisé pour calculer l'angle et donc la position de la source.

b. Calcul de l'angle

A partir du décalage temporel, l'angle calculé correspond à l'angle de la source par rapport aux deux microphones les plus proches. Afin d'obtenir une localisation de la source sonore autour d'un robot, il n'est donc pas possible d'utiliser directement l'angle calculé. Il faut donc repérer le couple qui est utilisé pour le calcul de l'angle :

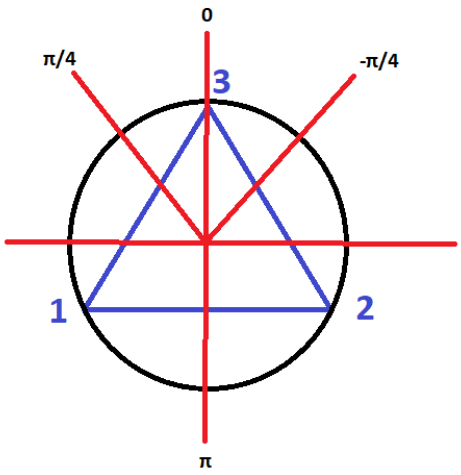

Figure 6 : Découpage de l'espace autour des microphones

Pour obtenir l'angle final θ_f en fonction de l'angle par rapport au couple le plus proche θ, il faut donc faire le calcul suivant :

- Couple 1-3 le plus proche : $\theta_f = \theta + \frac{\pi}{4}$
- Couple 1-2 le plus proche : $\theta_f = \theta + \pi$
- Couple 3-2 le plus proche : $\theta_f = \theta - \frac{\pi}{4}$

4. Incorporation à ROS

Si le programme fonctionne lorsqu'il est compilé avec Cmake puis exécuté, il ne sera pas reconnu par ROS s'il est laissé en l'état. En effet, il faut initialiser le nœud (ou node) avec la ligne :

```
ros::init(argc, argv, [nom du node]) ;
```

De plus, il est nécessaire de définir un objet représentant le node ROS, avec la ligne :

```
ros::NodeHandle nh ;
```

Après cela, le programme sera donc reconnu comme un nœud et pourra être exécuté par ROS.

L'objectif étant d'utiliser ce programme sur un robot, il convient de permettre au nœud ROS de localisation de communiquer avec un autre, via ce qui s'appelle un « topic ». Il faut donc définir notamment la nature de l'information à publier sur le topic (float, int, string, etc), ainsi que le topic sur lequel publier :

```
ros::Publisher cmd_velo = nh.advertise<geometry_msgs::Twist>([nom du topic]
, [taille de la file d'attente de publication]);
```

Avec la ligne cmd_velo.publish([variable à partager]), l'information à publier est donc envoyée sur le topic.

Le programme initial localise le son dans une boucle infinie « while(true) » dans laquelle se situe l'exécution de la fonction « TraitementduSon() ». Cependant, si cette configuration est conservée lors de l'utilisation avec ROS, il devient impossible d'arrêter le nœud ROS. C'est pourquoi la fonction est exécuter par une boucle « while(ros ::ok) » directement dans la fonction main(), ros ::ok permettant d'interrompre l'exécution du programme lorsque l'entrée clavier ctrl+c est détectée.

Le programme ayant été modifier pour fonctionner avec ROS, il ne reste plus qu'à créer le package contenant le nœud localisation. Les détails pour la création du package de localisation sont disponibles à Liens utiles : (Voir « Création d'un package »).

5. Utilisation du programme avec turtlesim

Afin de tester le programme, ce dernier a été utilisé avec le package « turtlesim » incorporé à ROS, et notamment avec le nœud « turtlesim_node ». Ce nœud affiche une tortue qui se déplace en fonction des commandes qui lui sont envoyées sur le topic /turtle1/cmd_vel.

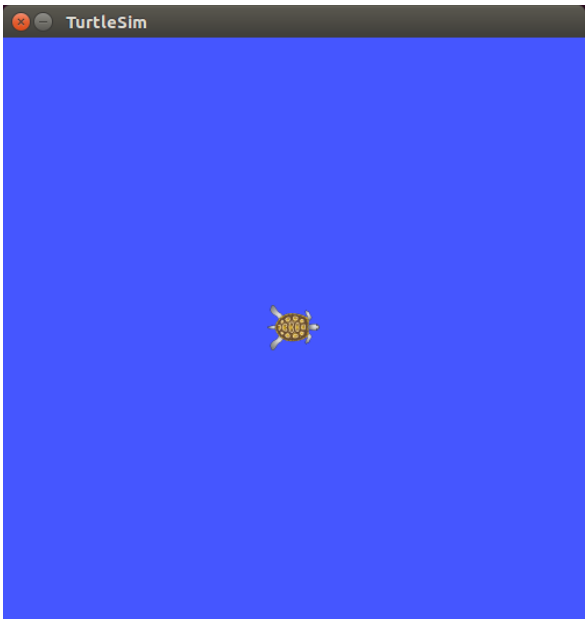

Figure 7 : Fenêtre de turtlesim_node

Pour la localisation à 180° la tortue tourne bien du côté duquel provient le son.

V. Conclusion

Pour conclure, la réalisation de ce projet a permis d'arriver à un système fonctionnel pour une localisation à 180°, même s'il reste toutefois perfectible notamment sur la précision de la localisation. En effet, la distance entre les microphones gagnerait à être augmentée, la limitation technique ayant imposé une distance de 5 cm pour ce projet. Quant à la localisation à 360°, il serait plus pertinent de tester le programme avec trois microphones comportant les mêmes caractéristiques.
Le projet a été réalisé sur une machine virtuelle, ce qui a pu ralentir l'avancée du projet, la gestion de plusieurs cartes sons étant compliquée voire impossible et les contrôleurs audios étant généralement instables.
Enfin, ce projet m'a permis de progresser dans le développement de programmes en C++, et j'ai pu approfondir mes connaissances sur ROS, qui étaient jusqu'à présent assez limitées. De plus, il a été très intéressant et enrichissant de se voir confronté à diverses complications et d'apprendre à les surmonter.

Liens utiles :

- Code pour une localisation à 180° dans ROS: https://gitcdr.univ-ubs.fr/ericsenn/localisation_audio/src/branch/master/Code_Final/Localisation180.cpp
- Code pour une localisation à 360° dans ROS : https://gitcdr.univ-ubs.fr/ericsenn/localisation_audio/src/branch/master/Code_Final/Localisation360.cpp
- Le CMakeLists.txt pour la compilation des deux programmes dans ROS : https://gitcdr.univ-ubs.fr/ericsenn/localisation_audio/src/branch/master/Code_Final/CMakeLists.txt
- Création d'un package ROS : https://gitcdr.univ-ubs.fr/ericsenn/localisation_audio/src/branch/master/Creation_package_v1.pdf