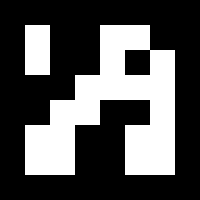
**DÉTECTION ARUCO**

1. **Élaboration de la vision**

Afin de permettre au robot de se repérer dans l’espace, nous avons mis en place une détection d’ArUco. La détection permet d’obtenir les coordonnées de la position de l’ArUco par rapport au robot placé à l’horizontal.



1. Calibration de la caméra

Dans un premier, nous avons réalisé un programme « Cam\_calibration.py» permettant de prendre une série de photos et de les placer dans un fichier. Ces photos seront utilisées pour calibrer la caméra, il suffit de placer un damier devant la caméra et ensuite de varier son inclinaison (voir photo dans le fichier projet sur la Raspberry). Puis appuyer sur la lettre q de votre clavier afin de stopper le programme une fois que vous avez le nombre de photos souhaitées.

ATTENTION : Si le nombre de photos est trop important, le programme suivant ne fonctionnera pas. Une quarantaine de photos suffira.

PRÉRECQUIS : Saisir le dimension du damier dans le programme « process.py» en suivant l’exemple suivant.

Une image contenant texte, clipart

Description générée automatiquement

cb\_width = 7 #Nombre d’intersection horizontal

cb\_height = 2 #Nombre d’intersection vertical

cb\_square\_size = 20,1 cm #Longueur damier

Ensuite, il faut exécuter le programme « process.py». Ce programme va réaliser automatiquement la calibration de la caméra et générer une matrice de transition. Ces données seront enregistrées dans un fichier « camera\_cal.npy». Ce dernier sera utilisé dans le programme principal afin d’obtenir les données de la position de l’ArUco par rapport au robot.

Cette série de vidéos vous permettra d’éclairer votre chemin :

https://www.youtube.com/watch?v=mBOfx-wRB2Q

https://www.youtube.com/watch?v=XFBKwme5HYk

https://www.youtube.com/watch?v=cIVZRuVdv1o

1. **Programme principal**
2. Élaboration programme principal **SANS** communication Arduino

Le programme principal « Detect\_aruco.py» utiliser les résultats obtenus de l’exécution des deux programmes précédents et affiche les axes sur l’image obtenue ainsi que les coordonnées de l’ArUco par rapport au robot.

Ce programme affiche uniquement les résultats de la détection mais ne transmet aucune donnée à l’Arduino qui pilote les actionneurs.

1. Élaboration programme principal **AVEC** communication Arduino

Le programme principal « Class\_detect.py» possède les mêmes caractéristiques que « Detect\_aruco.py» mais n’affiche pas le retour image afin de ne pas surcharger la Raspberry.

La communication entre la Raspberry et l’Arduino se fait via un port USB, c’est donc une communication série.

Sur la Raspberry, les noms des ports USB changent à chaque branchement. Pour palier cela, nous avons fixé le nom des ports USB de la Raspberry. Cela veut dire que la Raspberry reconnait le module branché automatiquement.

(voir le tutorial suivant : <https://www.freva.com/fr/attribuer-des-noms-de-port-usb-fixes-a-votre-raspberry-pi/>)

Cela nous permet de mettre, dans l’initialisation de la communication série, le paramètre suivant '/dev/ttyUSB\_arduino'. Ainsi, peu importe le nom de port USB, le programme reconnaitra l’Arduino.

1. Exécution du programme principal dans un environnement python

Nous avons créé un environnement python afin d’y installer les librairies nécessaires sans entrer en conflit avec d’autres projets présents ou futurs installés sur la raspberry. Pour exécuter un programme python dans cet environnement, il faut activer l’environnement à l’aide des commande suivantes dans le terminal :

cd cheminfichier/

source env/bin/activate

python nomscript.py

Le programme sera alors exécuté dans l’environnement souhaité.

À NOTER : Des informations complémentaires se trouvent en commentaires dans les codes eux-mêmes.