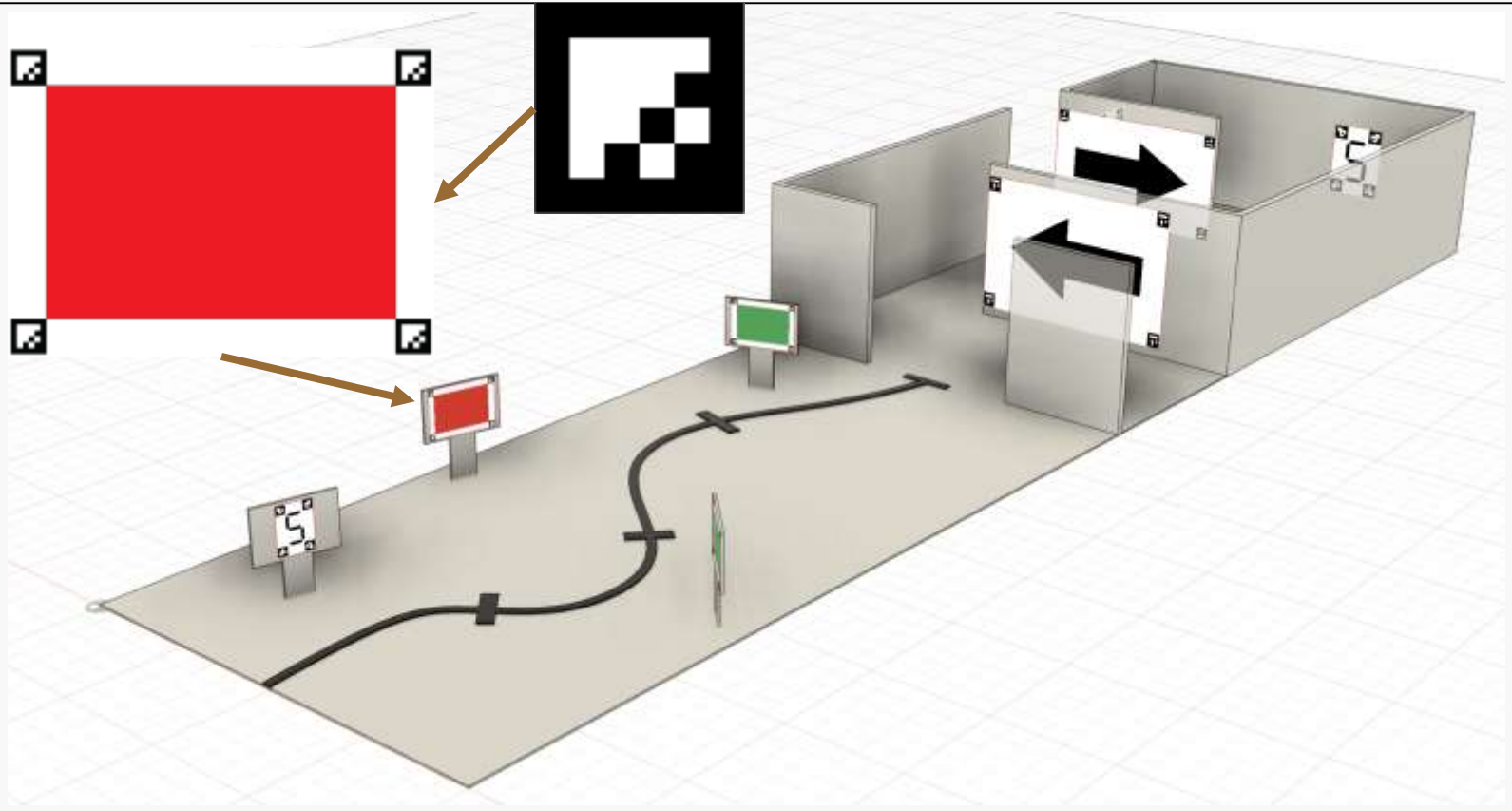
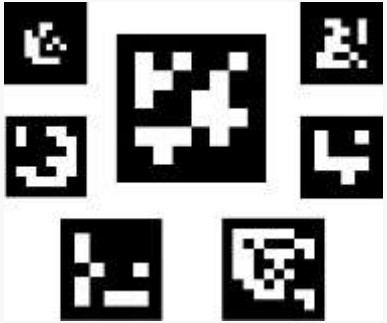
ING 2 – S2
OpenCV pour la robotique
Aruco et flèche

Romaric Sichler – 24/02/2024



Images d’illustrations non contractuelles

ARuco, pour quoi faire ?



Marqueurs délimitants les zones d'intérêts

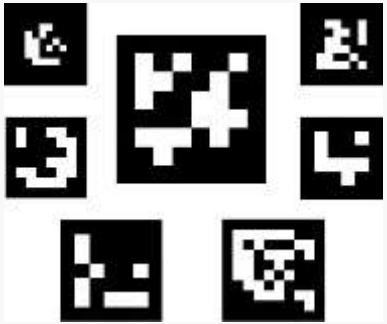
Détection simple de position, d'identifiant et d'orientation

Bibliothèque incluse dans openCV

Fonctionne par dictionnaire : usages multiples et adaptés

- Choisir un dictionnaire
- Ici 4 pixels par 4 pixels : petit nombre de variations nécessaire
- Pour d'autres applications : plus grand nombre de marqueurs possible

ARuco, à vous de jouer



Vérifiez que openCV est installé sur python3

```
pip3 install opencv-contrib-python
```

Essayez d'importer aruco depuis opencv (depuis python)

```
from cv2 import aruco
```

Générons des ARuco depuis le dictionnaire 4x4

```
cv2.aruco.generateImageMarker(dict,id,size)
```

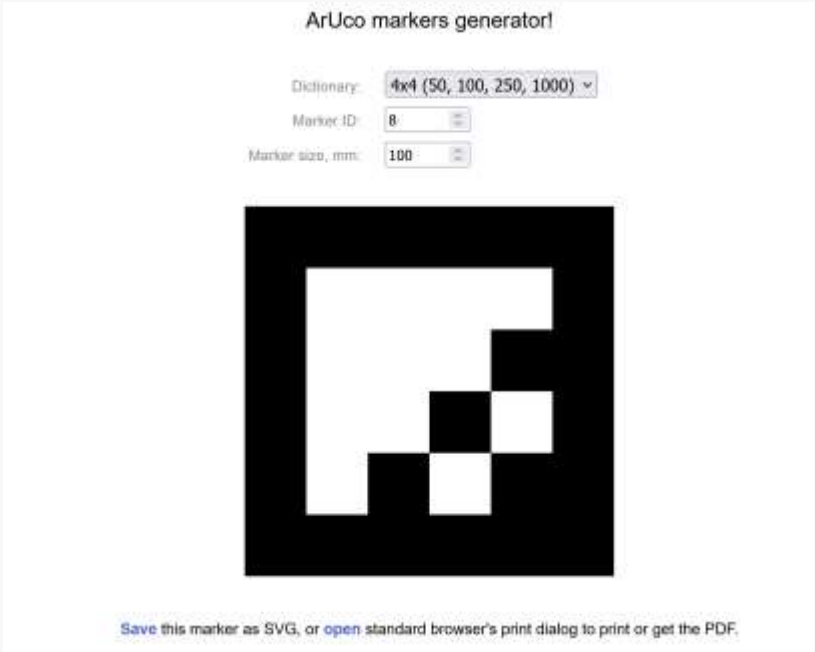
où dict est le dictionnaire utilisé, id l'identifiant du marqueur et size la largeur et hauteur de l'image générée

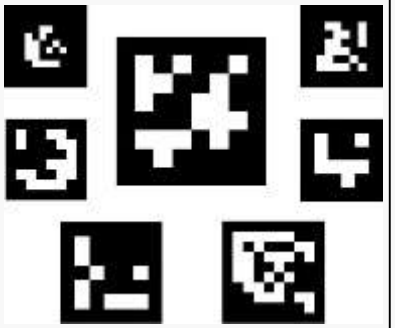
Voir **arucomaker.py** sur bootscamp

Génération en ligne possible depuis chev.me/arucogen/

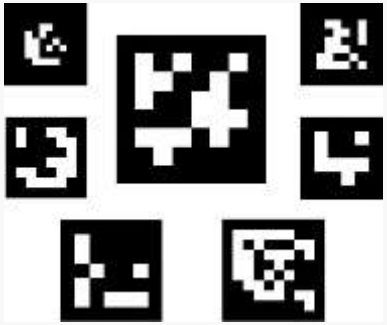
ARuco, à vous de jouer

Démo sur https://chev.me/arucogen/





ARuco, détection



Utilisation de la classe

`cv2.aruco.ArucoDetector(dict,params)`

où dict est le dictionnaire utilisé

et params est généré par la fonction `DetectorParameters()`

Ensuite utilisation de `detectMarkers(img)`

où img est une image

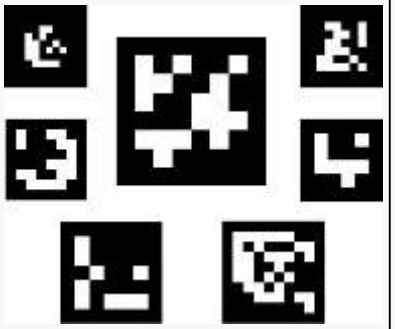
et qui renvoie : markerCorners, markerIds, reject

où markerCorners est un array de chacun des angles de chacun des marqueurs

et où markerIds est la liste des marqueurs trouvés

Voir **arucodetect.py** sur bootscamp

ARuco, détection



La détection des positions et des identifiants des marqueurs est facile et fonctionne largement en variation de luminosité

Essayez le programme **arucodetect.py** avec différentes images capturées

Testez avec 1 puis 2 puis 4 marqueurs sur la même image

Meilleure appréhension des zones d'intérêts de vos captures

Déformer un image

Permet de mieux appréhender une zone d'intérêt

Utilise `getPerspectiveTransform(coins1,coins2)`

où coin1 est une matrice de quatres points sur l'image d'origine

et coin2 est une matrice des positions de ces mêmes points sur l'image déformée

Utilise `warpPerspective(img,vectTrans,(x,y))`

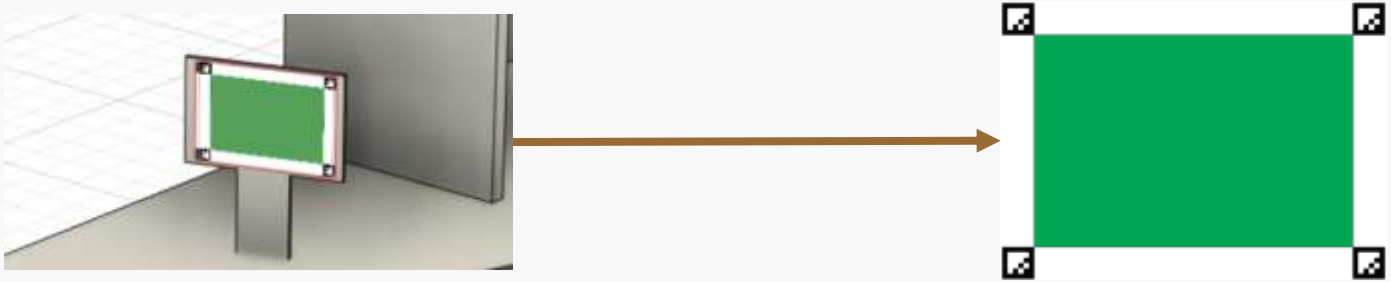
où `img est l'image d'origine, vectTrans le vecteur obtenu ci-dessus et x,y la taille de l'image deformée`

Déformer un image, à vous !

Ouvrez le fichier **warp.py** sur bootscamp

Modifiez celui-ci pour déformer une image capturée avec une élément de jeu encadré par 4 ARuco

L'objectif est d'obtenir une image où la surface délimitée par les quatres ARuco rempli entièrement la nouvelle image



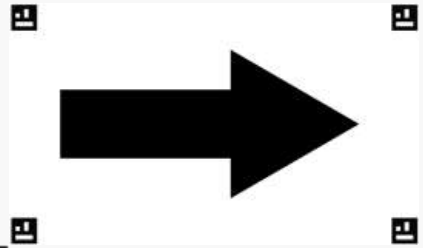
ARuco dans le terrain

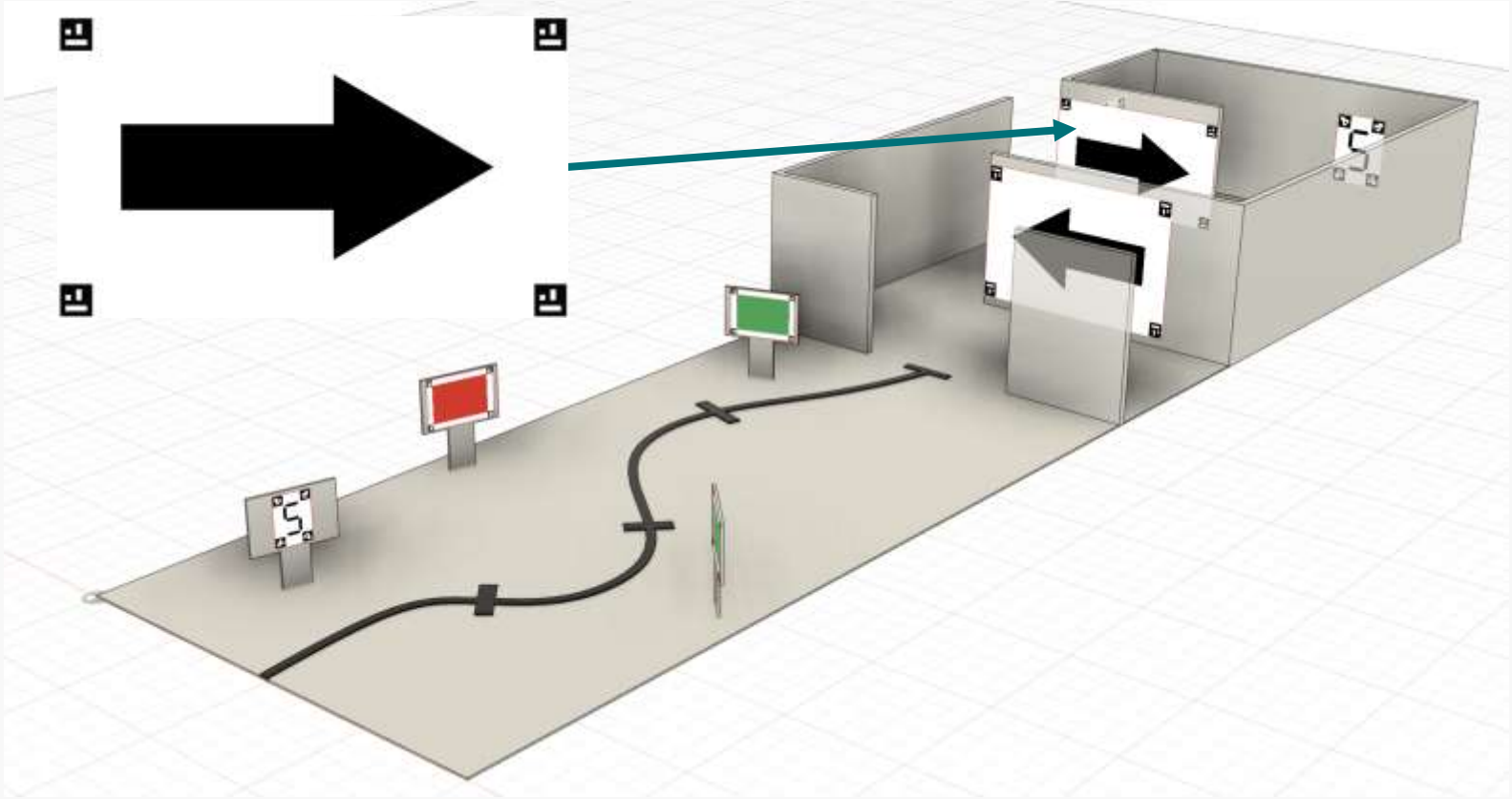
Les marqueurs délimitants les zones d'intérêts sont spécifique au type de la zone.

Id=8 -> zone de couleur

Id=13 -> Flèche

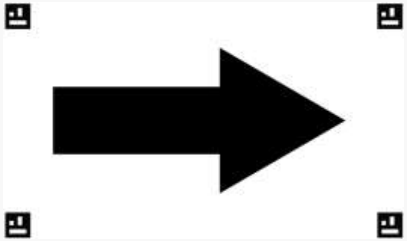
Id=9 -> Chiffre





Images d’illustrations non contractuelles

Labyrinthe et flèches



Évitement de bord basé sur le capteur ultrason ou caméra

Caméra plus difficile

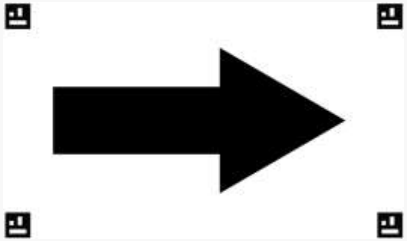
Les codes sont inclus dans adeept

Principe : prendre plusieurs mesures de distance ultrason à des angles différents en faisant bouger la tête du robot, orienter les roues avant vers la mesure de distance la plus longue

Problème : si il y a deux côtés ouverts, comment choisir

Suivre les flèches

Labyrinthe et flèches



Utilisation d'openCV pour suivre des flèches directionnelles

Nous cherchons à savoir si la flèche pointe vers la droite ou la gauche

Nombreuses technique possible, une simple est de compter les sommet

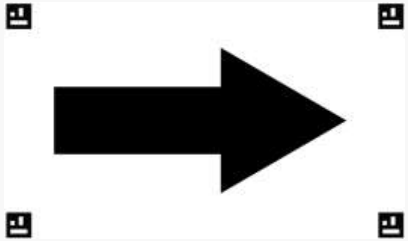
Utilisation de goodFeatureToTrack() comme au premier semestre

Obtention des positions des sommets de la flèche

Comptons le nombre de sommets à droite et à gauche de l'image

Le côté avec le nombre supérieur de sommet indique le sens de la flèche

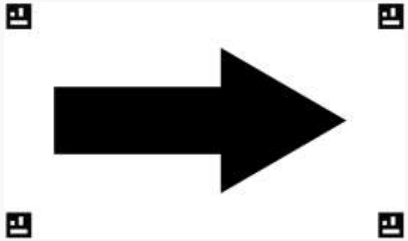
Labyrinthe et flèches



Voir le fichier **OrentationFleche.py** sur bootscamp

Comprendre son fonctionnement

Travail à rendre



Concevez un algorythme qui :

- prend une photo depuis la webcam de l'ordinateur
- trouve 4 ARuco avec le même identifiant dans l'image, sinon renvoie une erreur
- déforme l'image pour ne travailler que dans la zone d'intérêt définie par ces 4 marqueurs
- trouve le sens d'une contenue dans la zone définie et l'affiche en console
- vous pouvez utiliser des affichages graphiques pour que les éventuelles erreurs du programme soient transparentes

Travail à rendre

- rendez votre code dans un fichier .py ainsi qu'un diagramme commenté de l'algorithme
- La nomenclature sera NomPrénom.py et NomPrénom.pdf
- La date buttoire de dépôt est le 17 mars à 23:59

La suite du programme

-Point technique le jeudi 7 aprem

-Dernier cours concernant la lecture de chiffres et la continuité la semaine du 13 mai

-Rattrapages du semestre 1 la semaine prochaine