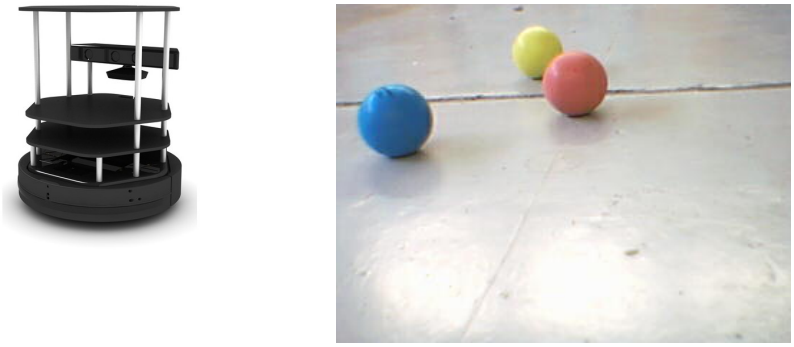
Traitement des images
– Travaux Pratiques –

Un questionnaire d'évaluation est fourni à la fin du sujet. Merci de le remplir au fur et à mesure de la séance et de le rendre en fin de séance.

1 Objectif du TP

Ces travaux pratiques illustrent, certes sur un exemple simple, les notions introduites en cours sur la chaîne perceptuelle depuis l'acquisition jusqu'à l'interprétation des images. Le but est la détection et reconnaissance d'objets à partir de la caméra couleur embarquée sur un robot mobile. Ces fonctions visuelles seront exploitées par ailleurs par le robot pour se positionner à une distance donnée de ces objets. Ces objets seront ici matérialisés par des balles de couleurs différentes pré-supposées connues (diamètre : D = 10.5 cm) Certaines balles sont assimilées aux consignes spatiales à atteindre par le robot e.g. le robot doit se positionner à une distance prédéfinie (1 m) de la balle rouge (figure 1)



(a) (b)

FIGURE 1 – Robot Turtlebot2 (a), image acquise depuis sa caméra embarquée (b).

Les travaux pratiques se divisent en deux parties :

1. le **traitement des images**, *i.e.* le filtrage et la segmentation des images couleurs afin d'en isoler les régions colorimétriques pertinentes dans le voisinage immédiat du robot,
2. l' **analyse des images** ainsi segmentées et la localisation, certes grossières, de ces balles pour en déduire le déplacement approprié du robot.

2 Manipulation du robot

Le travail s'effectue sur un PC sous Ubuntu, qui est une distribution Linux. Les turtlebots sont également sous Ubuntu et le robot est piloté au travers du *middleware* ROS (Robot Operating System).

Pour se connecter à votre PC, il faut choisir l'option *Ubuntu* au démarrage. Puis,
— login : `etudiant`
— mot de passe : `etudiant`
Il y a 4 turtlebots numérotés de 1 à 4. Tous les turtlebots utilisent les mêmes identifiants :
— login : `turtlebot`
— mot de passe : `turtlebot`
Sur chaque robot en début de séance, il faut installer la base de code du projet. Ceci doit être fait une unique fois même si plusieurs binômes partagent un même robot. La procédure est la suivante :

1. se connecter à un robot
2. se placer dans le répertoire (avec la commande *change directory* `cd`) :
   `/home/turtlebot/catkin_ws/src`
3. cloner le code depuis Github, avec la commande :
   `git clone https://github.com/BrunoDatoMeneses/TP-Traitement-Image`

Si l'opération se passe bien, l'ensemble du code du projet se trouve désormais sur le PC du robot dans le répertoire :
`/home/turtlebot/catkin_ws/src/TP-Traitement-Image/ballsearch/src`
Le travail sera effectué sur votre PC, mais pour tester votre code, il faudra l'installer sur le PC du robot. La procédure générale de développement est la suivante :

1. Sur votre PC, créer votre workspace. Sur un terminal, taper les commandes suivantes :
   `mkdir -p /catkin_ws_<NOM_GROUPE>/src`
   `cd /catkin_ws_<NOM_GROUPE>/src`
   `catkin_init_workspace`
   `cd ..`
   `catkin_make` (compilation)
   ATTENTION : compilation à toujours faire dans `catkin_ws_<NOM_GROUPE>`
2. Télécharger le code du projet :
   `cd /catkin_ws_<NOM_GROUPE>/src`
   `git clone https://github.com/BrunoDatoMeneses/TP-Traitement-Image`
   `cd ..`
   `catkin_make` (compilation)
3. Veiller à référencer les chemins d'utilisation de ROS et de votre workspace. Pour cela, sur un terminal, taper la commande suivante :
   `gedit ~/.bashrc`
   Vérifiez qu'à la fin du fichier se trouvent les 2 lignes suivantes sinon ajoutez-les :
   `source /opt/ros/indigo/setup.bash`
   `source ~/catkin_ws_<NOM_GROUPE>/devel/setup.bash`
   Ouvrez un nouveau terminal, s'il n'y a pas d'erreur, votre workspace est prêt.
4. Analyser/modifier/compléter/compiler (catkin) le code. Sur votre PC, placer vous dans le répertoire
   `/home/etudiant/catkin_ws_<NOM_GROUPE>/TP-Traitement-Image/ballsearch/src`
   Pour éditer un fichier `my_file.cpp`, utiliser la commande :
   `gedit my_file.cpp &`
5. Envoyer le code modifié sur le robot, avec la commande suivante, il y a deux chemins séparés d'un espace à spécifier, celui où se trouve dossier à copier sur l'ordinateur de travail et celui sur l'ordinateur du robot, X est le numéro du turtltebot choisi pour le

test (1, 2 ou 3) :
`scp -r /home/etudiant/catkin_ws_<NOM_GROUPE>/src/TP-Traitement-Image/ turtlebot@turtlebotX:/home/turtlebot/catkin_ws/src`
un mot de passe vous sera demandé, il faudra donner le mot de passe de connexion du robot : `turtlebot`

6. Compiler le code. Se placer dans le répertoire du robot :
`cd /home/turtlebot/catkin_ws`
effacer les exécutables compilés précédemment par d'autres groupes :
`rm -r build/`
et compiler :
`catkin_make`
7. tester la modification. Sur le robot, ouvrez trois terminaux, et entrez les commandes suivantes dans l'ordre (entrez une commande par terminal, attention à ne pas relancer les deux premières si c'est déjà fait) :
   (a) `roslaunch turtlebot_bringup minimal.launch`
   (b) `roslaunch turtlebot_bringup 3dsensor.launch`
   (c) `roslaunch ballSearch ballSearch.launch`

   A partir du moment où la commande (3) est entrée, le robot part à la recherche de la balle (attention, il est vif !). Par défaut, les lignes de codes responsables du déplacement sont commentées dans le fichier `ballSearch_node.cpp`, Il y en 3 commençant par `ballSearch.sendBallReference...`. Pour arrêter l'exécution, il faut sélectionner le terminal où `ballSearch` a été lancée et faire un `ctrl^C` (touches contrôle + C pressées simultanément). Pour relancer le test il suffira de relancer `ballSearch` dans ce même terminal.

Les étapes 4 à 7 sont à répéter en fonction du résultat des tests et du travail demandé dans le sujet.

Attention : plusieurs binômes partageront un même turtlebot, faites attention à ne pas télécharger votre code modifié sur le robot si un autre binôme est en train d'utiliser le robot.

3 Travail à réaliser

Dans cette première partie, l'objectif et d'observer le fonctionnement de la chaîne de traitement du robot, en particulier la segmentation de l'image prise par la caméra. Ensuite, il s'agira de comprendre l'implémentation d'une fonction d'*erosion*, en déduire et réaliser une implémentation d'une fonction de *dilatation*. Finalement, il faudra combiner ces deux fonctions pour réaliser une fonction d'*ouverture* et une fonction de *fermeture*.

3.1 Traitement d'images – Segmentation et filtrage de l'image

La segmentation repose sur (1) un seuillage colorimétrique puis (2) un filtrage morphologique de l'image pour filtrer les pixels parasites.

— Positionner les balles dans le champ de vue de la caméra à moins de 1,80cm (veillez à ce qu'il n'y ait pas d'obstacle entre le robot et la balle), puis exécuter sur le terminal du robot le programme `ballSearch` avec la commande :
`roslaunch ballSearch ballSearch.launch`

Ce programme permet de segmenter les balles rouge ou bleue sur les trois canaux colorimétriques (R,V,B). Analyser les images ainsi obtenues.

— On souhaite filtrer tout ou partie des pixels correspondant à des régions parasites. La démarche est d'appliquer une *ouverture* (Erosion $\mathcal{E}$ puis Dilation $\mathcal{D}$) sur une image binaire puis une *fermeture* (Dilation $\mathcal{D}$ puis Erosion $\mathcal{E}$). En s'inspirant de la fonction `erosion()`, compléter les fonctions `dilatation()`, `ouverture()`, et `fermeture()` du fichier `traitement.cpp`. On rappelle que pour tout pixel I de coordonnées (x, y) :

$$\mathcal{E} = \min\{I(x,y)|(x,y) \in \mathcal{H}\}, \mathcal{D} = \max\{I(x,y)|(x,y) \in \mathcal{H}\} \tag{1}$$

où $\mathcal{H}$ est la taille du masque (c.f. support de cours).

3.2 Analyse d'images – reconnaissance et localisation des objets

Le but est ici de regrouper les pixels en régions (voir fonction `Etiqueter_Region` du fichier `analyse.cpp`, calculer des attributs de forme pour filtrer les régions paraxites (fonction `Extract_attributs`), enfin de calculer le centre de gravité des régions subsistantes pour estimer leurs positions dans la scène observée.

— La fonction `Extract_attributs` du fichier `analyse.cpp` permet de calculer pour chaque région le barycentre, la surface, la compacité, le périmètre, et la boite englobante. Ces paramètres sont listés dans la structure `liste_objet` définie dans la fichier `objet.h`. Le principe est de filtrer les régions R parasites par des règles sur l'aire $A(R)$, la compacité $F_c(R)$, la dimension de la boite englobante,...En utilisant ces différents attributs, compléter le programme pour le calcul de la compacité. On rappelle :

$$F_c(R) = \frac{4\pi A(R)}{P(R)^2} \tag{2}$$

où $A(R)$ et $P(R)$ sont respectivement l'aire et le périmètre de la région R

— Le centre de gravité de chaque région R est représenté par les variables (U_{cg}, V_{cg}). La position réelle de l'objet est notée (X, Y, Z) où Z est la distance entre la caméra et l'objet telle que :

$$Z = f \cdot \frac{D}{k_u \cdot d} \tag{3}$$

où :

— k_u (resp. k_v) sont les dimensions ligne (resp. colonne) d'un pixel,

— D est le diamètre réel de l'objet,

— d est le diamètre dans l'image de l'objet (en nombre de pixels). Le diamètre d sera estimé par la boite englobante de la région de l'objet.

Calculer le profondeur Z, à partir des données de la caméra et des attributs de l'objet. Mesurer les distances relatives réelles caméra/balles.