Algorithmes pour l'autonomie

Cours correspondant : Robotique pratique (http://www.ensta-bretagne.fr/lebars/robotique_pratique.pdf).

Rendre sous Moodle dans un fichier de la forme DATE_NOM1_NOM2_..._NOMN.zip tous les documents utiles permettant de montrer le travail effectué : codes, fichiers de config, logs, données brutes de capteurs, infos précises sur les versions logicielles et matérielles utilisées, schémas en CAO ou manuels, photos et vidéos à différentes étapes, etc.

Dans ce TD, **plusieurs activités à se repartir en groupes** de quelques personnes et à faire **à tour de rôle**, sont proposées :

1. Suivi de waypoints avec un robot bateau à moteur, sous-marin, buggy, char ou quadrirotor en simulation
2. Suivi automatique d'objet coloré par webcam montée sur servomoteurs pan and tilt
3. Cartographie 3D avec une Kinect 2

Tout le monde ne pourra a priori pas faire toutes les activités à cause de limites de matériel, temps, etc. l'idée est que selon ses connaissances, chacun puisse faire ce qu'il ne connaissait pas.

Activité 1 : Suivi de waypoints avec un robot bateau à moteur, sous-marin, buggy, char ou quadrirotor en simulation

Lieu : salle info
Nb max personnes : illimité
Durée estimée : 2h30

Révision équation d'état, simulation avec Euler, MATLAB, coordonnées homogènes, dessin

1. Donner des équations d'état (juste l'équation d'évolution en 2D ou 2.5D) pour l'un des robots bateau à moteur, sous-marin, buggy, char ou quadrirotor (au choix) évoqués lors du 1er cours et les coder en MATLAB. Pour se donner une idée de leur comportement, vous pouvez consulter les vidéos suivantes :
   - Bateau à moteur (Motorboat) : https://www.youtube.com/watch?v=OvAK0KxSyt0
   - Sous-marin (SAUC'ISSE) : https://www.youtube.com/watch?v=uivKq5Ii_Go
   - Buggy : https://www.youtube.com/watch?v=3QkiayBLo1o
   - Char (ETAS WHEEL d'euRathlon 2013 ou futur char pour euRathlon 2017): https://www.youtube.com/watch?v=0j-9sdhdWXw
   - Quadrirotor: https://www.youtube.com/watch?v=lV-m9-O11hY
2. Dessiner le robot à un état donné.

3. Simuler le robot avec Euler en train de faire un cercle (avec une commande constante simple).

4. S'inspirer de http://www.ensta-bretagne.fr/lebars/Share/Ecrazor_game.zip pour pouvoir utiliser les touches du clavier pour contrôler en direct le robot simulé.

Régulation du robot

1. En supposant qu'on mesure la position x,y du robot à tout moment (e.g. avec un GPS, ou USBL pour le sous-marin) ainsi que le cap (avec une boussole), on va pouvoir faire un suivi de waypoints automatique. S'inspirer de http://www.ensta-bretagne.fr/lebars/Share/Ecrazor_game.zip pour rajouter la génération de waypoints automatique dans le code du robot simulé.

2. D'abord faire la régulation en cap (en utilisant juste la boussole) et tester son bon fonctionnement à plusieurs caps différents.

3. Pour aller dans la direction du waypoint, on va ensuite calculer le cap qu'il faudrait suivre et le passer en entrée de la régulation en cap, et on va rajouter une condition de validation qui fait passer au waypoint suivant quand on est dans un rayon de 2.5 m autour...

Activité 2 : Suivi automatique d'objet coloré par webcam montée sur servomoteurs pan and tilt

Lieu : salle info
Nb max personnes : environ 5-10
Durée estimée : 2h30

Le but de cette activité est de mettre au point un suivi automatique d'objet par une webcam montée sur servomoteurs pan and tilt. La caméra peut donc tourner suivant 2 axes, qu'il faudra commander pour que l'objet détecté reste au centre de l'image de la caméra.

Monter le kit pan and tilt et attacher une caméra dessus.

Vérifier le fonctionnement des servomoteurs à l'aide d'un testeur de servos.
Le testeur de servo permettra de faire tourner les moteurs à une position donnée. Il faudra bien veiller à choisir des éléments compatibles parmi ceux à votre disposition. Certains

connecteurs de servos peuvent avoir une forme incompatible avec le testeur, rogner la partie du connecteur gênante si nécessaire pour respecter le format JR.

Attention : aller progressivement et vérifier avec vos encadrants, des multimètres, etc. avant de brancher la batterie la première fois ou si vous avez un doute !

Remplacer ensuite le testeur de servos par une carte d'interface pour pouvoir contrôler les servomoteurs depuis un programme exécuté sur un PC. Commencer à tester avec les programmes fournis par le constructeur de la carte d'interface, puis avec votre propre programme, en s'aidant notamment des infos données dans le TD de manipulation de ces cartes fait précédemment.

Combiner ensuite avec un code de détection d'objet pour que les servomoteurs bougent de manière cohérente par rapport à l'objet.

Activité 3 : Cartographie 3D avec une Kinect 2

Lieu : salle info
Nb max personnes : environ 10
Durée estimée : au moins 2h30

La Kinect 2 (fournie à l'origine avec les consoles Xbox One) est un regroupement de plusieurs capteurs :

- Une **caméra Full HD** dans le visible :
  - Résolution : 1920 x 1080 (nombre de pixels en largeur x nombre de pixels en hauteur)
  - Fréquence de rafraichissement : 30 Hz (15 Hz in low light)
- Une **caméra dans l'infrarouge** (avec éclairage actif) :
  - Résolution : 512 x 424 (nombre de pixels en largeur x nombre de pixels en hauteur)
  - Fréquence de rafraichissement : 30 Hz
- De nombreux micros

La Kinect a des prérequis matériels et logiciels assez contraignants et pas toujours très clairs, notamment dus à la grande quantité de données qu'elle peut fournir : port **USB Type A 3.0** (privilégier un port USB qui ne partage a priori pas sa bande passante avec d'autres éléments), **Windows 8-11/Ubuntu 14.04-24.04 64 bit**, processeur et carte graphique suffisamment

récents et puissants), Kinect SDK v2[1] / libfreenect2[2]. We will use also MeshLab and OpenCV to view and process some of its data.

Par divers traitements internes, elle est capable de nous retourner :

- Des **images couleurs**.
- Des **images infrarouges**.
- Des images en niveaux de gris, celui-ci étant plus ou moins foncé en fonction de la distance des objets face au capteur (profondeur). Cette image donne la **distance au premier obstacle pour chaque pixel**. Ces données sont calculées par la Kinect (ou son driver) en utilisant les données brutes de la caméra infrarouge. L'étendue de mesure est **0 à 8 m**.

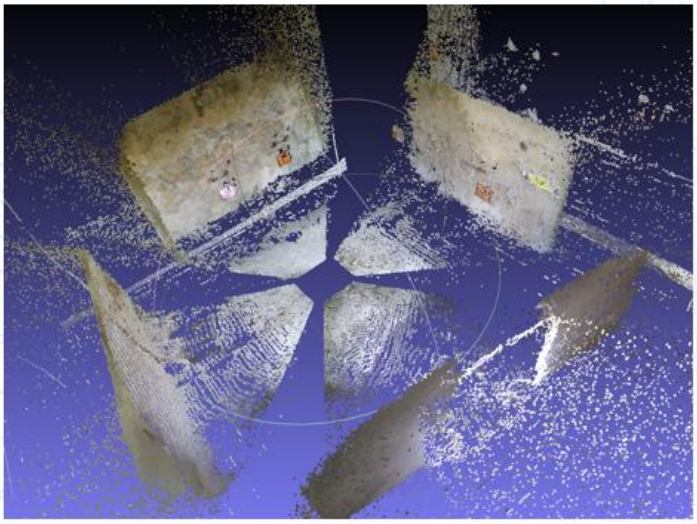
Comme ces images viennent de capteurs séparés, plusieurs problèmes sont à considérer :

- Point de vue légèrement différent -> zones visibles dans une image mais pas dans l'autre.
- Résolution et ouverture angulaire (FOV) différentes.
- Symétries, etc.

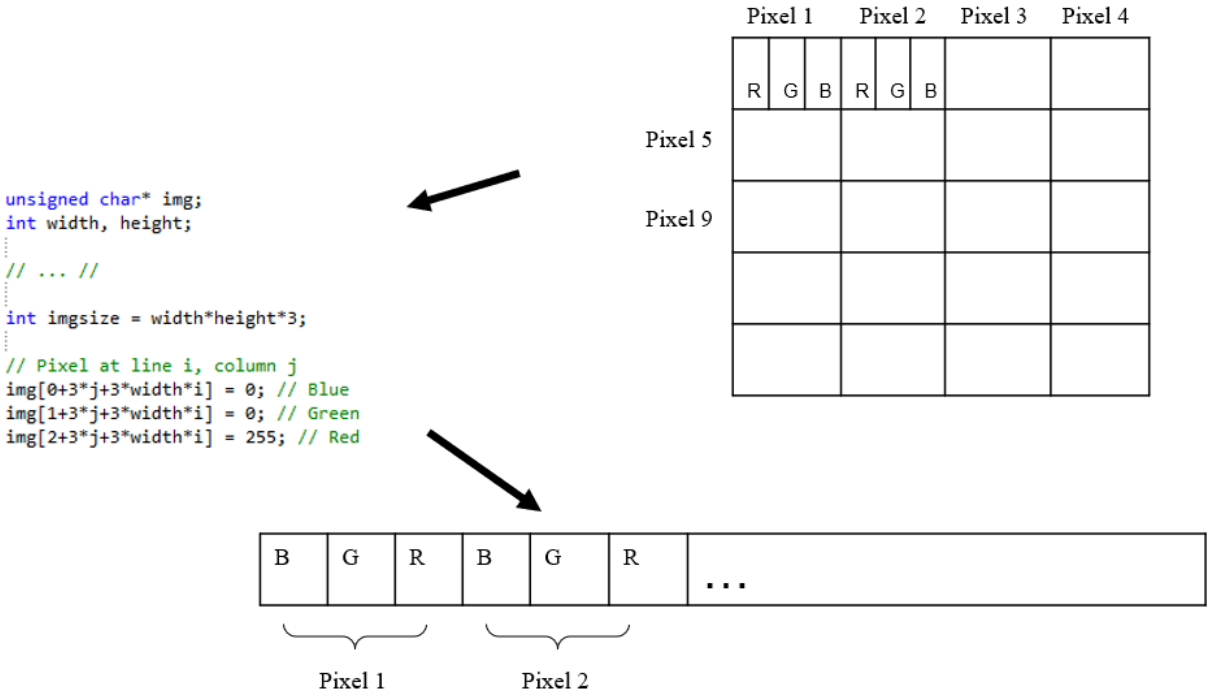
Le but de ce sujet est d'utiliser la Kinect pour faire une reconstruction 3D simple et rapide de l'environnement d'un robot. Celui-ci aurait juste comme capteurs une Kinect et une boussole. L'idée est que si le robot prend 4 fois des données en tournant à chaque fois de 90 degrés, on devrait pouvoir reconstituer l'ensemble de l'environnement qui l'entoure, comme dans l'exemple suivant :

[1] Already installed on the Windows PCs of the school, otherwise see https://www.microsoft.com/en-us/download/details.aspx?id=44561 and https://www.microsoft.com/en-us/download/details.aspx?id=100067, please read carefully the installation instructions.

[2] https://github.com/OpenKinect/libfreenect2, do not set **CMAKE_INSTALL_PREFIX**, so that the library is installed in **/usr/local** on Ubuntu (except for the Ubuntu PCs of the school (in room E006), where libfreenect2 is already installed in **/usr/local** but the default build instructions on https://github.com/OpenKinect/libfreenect2 still need to be followed to get **Protonect** test program). Note also that a reboot might be required to be sure that the suggested udev rules are active.



Pour info/rappel la représentation courante d'une image couleur (chaque pixel ayant sa couleur définie par un niveau de rouge, de vert et de bleu (3 canaux), en particulier R=G=B=0 est noir et R=G=B=255 est blanc) en informatique est de cette forme (e.g. avec l'API C d'OpenCV, et l'API C++) :




```
cv::Mat img;

// ... //

int nbpix = img.rows*img.cols;

// Pixel at line i, column j
img.at<cv::Vec<unsigned char, 3>>(i, j)[0] = 0; // Blue
img.at<cv::Vec<unsigned char, 3>>(i, j)[1] = 0; // Green
img.at<cv::Vec<unsigned char, 3>>(i, j)[2] = 255; // Red
```

Pour l'image en profondeur, il n'y a qu'un niveau de gris (1 canal) par pixel (ou 3 canaux avec R=G=B pour chaque pixel), ce niveau de gris (entre 0 et 255) étant proportionnel à la distance des obstacles dans le champ du pixel :

- Gris foncé : loin, noir étant 8 m (ou pas de donnée de distance)
- Gris clair : près, blanc étant 0 m (ou dans la zone morte, à vérifier…)

1. Assembler la Kinect et utiliser le Kinect SDK v2 (run Kinect Studio v2.0 app) pour Windows ou libfreenect2 (run Protonect app) pour Ubuntu pour bien comprendre ce que représente l'image en profondeur en observant avec des objets à diverses distances.

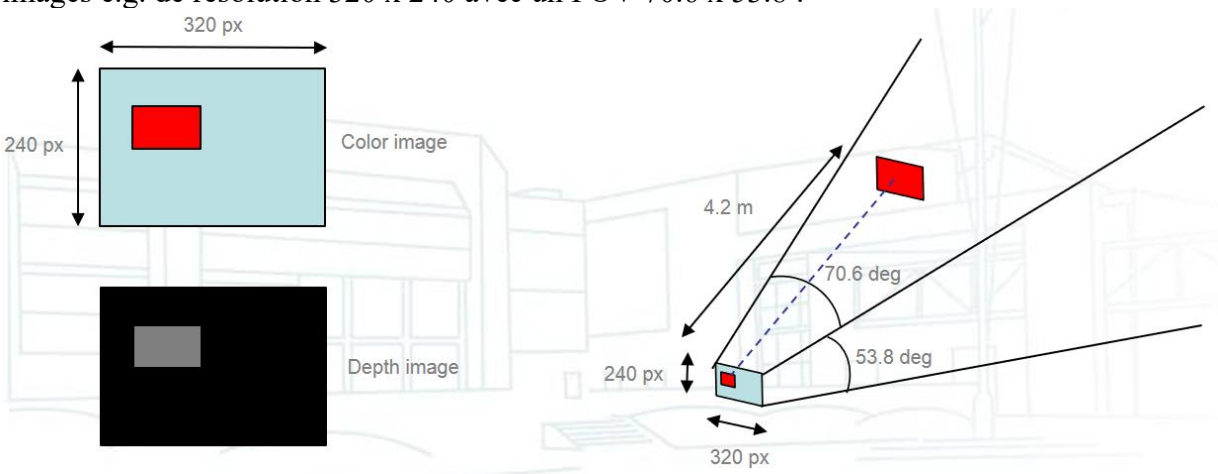
2. Télécharger et lancer ensuite UxVCtrl[3][4] et appuyer sur **SHIFT+P** avec le focus sur l'une de ses fenêtres OpenCV[5] (ou taper la commande **snapshot** dans le terminal

[3] https://www.ensta-bretagne.fr/lebars/Share/UxVCtrl_Kinect2.zip for Windows (you might need to install also Visual Studio 2010 VC++ Redistributable).
[4] https://www.ensta-bretagne.fr/lebars/Share/UxVCtrl_Freenect2.zip for Ubuntu: run **sudo apt-get install -y libopencv-dev libmodbus-dev libgtk2.0-dev cmake ; chmod +x ./UxVCtrl_ubuntu_noble** ; **LD_LIBRARY_PATH=. ./UxVCtrl_ubuntu_noble** (**noble** is for Ubuntu 24.04, replace with **jammy** for Ubuntu 22.04), if it does not work use **cmake-gui** (**sudo apt-get install cmake-gui**) to rebuild from source on https://github.com/ENSTABretagneRobotics/UxVCtrl/releases/tag/Fall2024r2 with

UxVCtrl si la touche SHIFT n'est pas reconnue) pour enregistrer des images déjà approximativement corrigées dans son sous-dossier **pic** : FOV (Field Of View) : 70.6 x 53.8 (angle de vue en degrés en largeur x angle de vue en degrés en hauteur) en 320 x 240 après corrections d'échelle (sans déformation des images, mais avec perte de données).

Le FOV, la résolution, et la distance pour chaque pixel vont pouvoir être utilisés pour placer un point coloré dans l'espace.

Représentation dans l'espace d'un groupe de pixels rouges de l'image couleur, dont le niveau de gris correspondant dans l'image en profondeur se traduit par une distance de 4.2 m, si images e.g. de résolution 320 x 240 avec un FOV 70.6 x 53.8 :



Pour pouvoir traiter des images, on a besoin de la bibliothèque OpenCV (en C/C++ pour pouvoir être compatible avec Kinect SDK v2 / libfreenect2 par la suite). Si la version sur les PC utilisés ne convient pas, voir https://www.ensta-bretagne.fr/lebars/Share/setup_qt_opencv.pdf[6]. Vérifier le bon fonctionnement d'OpenCV avec CMake comme suggéré dans le document précédent. Un autre exemple de code faisant quelques traitements de seuillage et détection de contours sur des images de webcam standard est aussi disponible à titre indicatif : https://www.ensta-bretagne.fr/lebars/Share/thresh_contours.zip. Pour des infos supplémentaires sur OpenCV, voir http://docs.opencv.org/4.6.0/.

3. Charger et afficher 1 image color et l'image depth correspondante chacune dans sa fenêtre.

   imread namedWindow imshow waitKey destroyAllWindows release

4. Afficher maintenant l'image couleur modifiée avec uniquement les pixels qui ont une profondeur > 0, i.e. mettre en noir ceux avec une profondeur de 0 (pour cela il faudra parcourir l'image pixel par pixel).

DISABLE_OPENCV_SUPPORT=OFF and **ENABLE_CVKINECT2SDKHOOK=ON** CMake options and copy the built program to the original folder to ensure the configuration files used are the same.

[5] For UxVCtrl on Ubuntu, you can switch between cameras by pressing ENTER, 6, and then 7 multiple times, note also that numeric pads and SHIFT key might not work in some configurations.

[6] OpenCV, CMake and Qt Creator are already installed on the Windows PCs of the school, but if you want to use Visual Studio Code, you might need to install its extension **ms-vscode.cpptools-extension-pack**, see also https://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf.

5. Calculer les coordonnées x,y,z de chaque pixel dans l'espace et les sauvegarder dans un fichier texte de la forme :
   z0 x0 y0 r0 g0 b0
   z1 x1 y1 r1 g1 b1
   …

   Il y a normalement un effet de perspective à prendre en considération pour que les positions soient réalistes…

   ofstream

Il faut trouver un moyen de dessiner des nuages de points en 3D (la plupart des formats de fichiers 3D étant plutôt optimisés pour dessiner des mesh formées de vertex, arêtes, faces et textures, voir https://en.wikipedia.org/wiki/Polygon_mesh et e.g. le format OBJ). Le format de fichier PLY, pouvant être facilement ouvert avec e.g. le logiciel MeshLab[7][8], peut être utilisé.

6. Télécharger, extraire et ouvrir dans MeshLab puis dans un éditeur de texte le fichier https://www.ensta-bretagne.fr/lebars/Share/points_sample_PLY.zip et essayer de comprendre le format.

7. Rajouter l'écriture de l'en-tête correcte dans le fichier créé avec les points 3D précédemment (noter qu'il y a besoin d'indiquer le nombre de points dans l'en-tête), changer l'extension du fichier généré en .ply et ouvrir le résultat dans MeshLab.

8. Gérer la rotation de 4 paires d'images (color+depth) prises à 90 deg pour qu'elles ne soient pas superposées si on ouvre les 4 fichiers .ply correspondants dans MeshLab. L'idée est d'obtenir une reconstruction 3D de l'environnement comme mentionné plus haut.

9. Avec par exemple le code d'adaptation Kinect vers OpenCV pour que ce soit comme une webcam (télécharger https://www.ensta-bretagne.fr/lebars/Share/Kinect2WebcamOpenCV460.zip pour Windows ou https://www.ensta-bretagne.fr/lebars/Share/Freenect2WebcamOpenCV460.zip pour Ubuntu), sauvegarder les images brutes de la Kinect et faire soi-même (éventuellement en mieux…) le code faisant tous les redimensionnements et autres adaptations à la place de UxVCtrl…

[7] If it is not already installed, for Windows download and extract https://www.ensta-bretagne.fr/lebars/Share/MeshLab.zip, for Ubuntu try **sudo apt install meshlab** (avoid the version from **snap**). For non-English versions of the OS, the decimal separator might need to be changed (comma instead of dot), on Linux try to run **export LANG=C; meshlab** to force the use of the dot… Check https://stackoverflow.com/questions/41734443/mesh-lab-2016-where-is-the-light-icon-gone if lighting looks off.
[8] If MeshLab does not display correctly on Ubuntu 22.04 and later, try to uncomment **WaylandEnable=false** in **/etc/gdm3/custom.conf** and reboot...