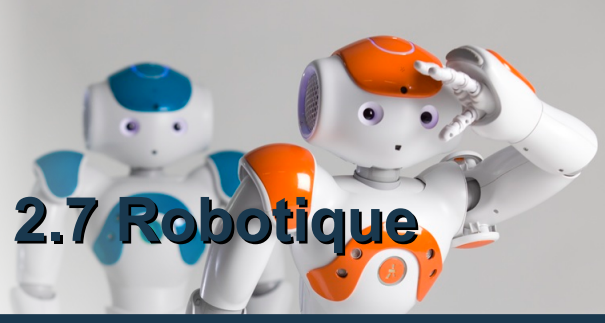
Rapport de synthèse – UV 2.7 Robotique

Cyril Rouvière - ENSI 2015 - 13/06/2013



Mots-clés

Robotique – NAO – Vision par ordinateur – Tracking – 42

Présentation du projet

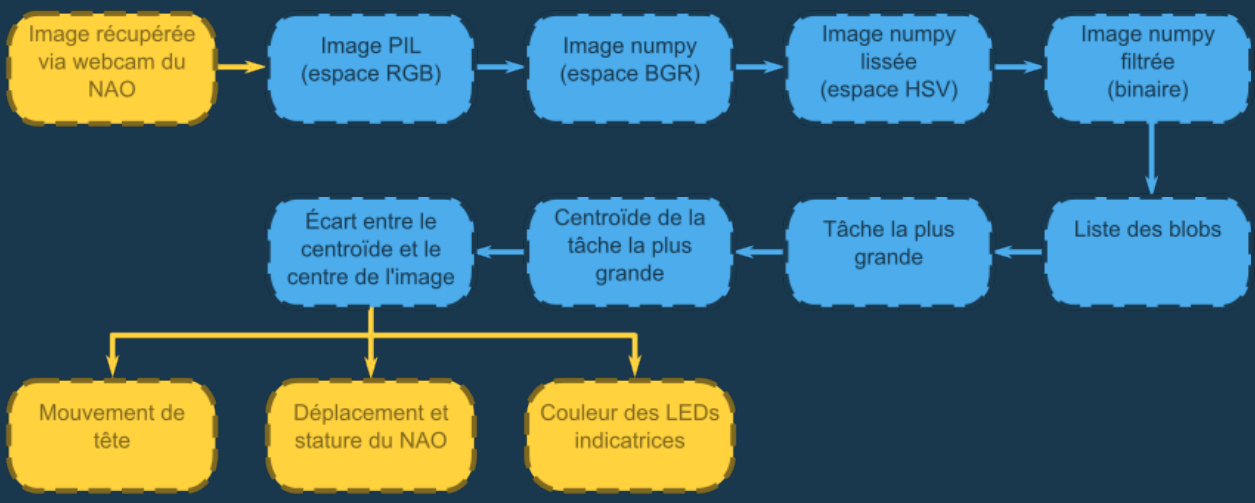
Dans le cadre de l'UV 2.7 Robotique, nous avons essayé de munir les NAOs d'un système de suivi et d'évitement des JOGs ; permettant les fonctionnalités suivantes :

- détecter les JOGs par leur couleur via webcam
- tourner la tête pour ne pas les perdre de vue
- avancer vers eux en gardant une distance de sécurité raisonnable
- reculer (voire se mettre en position de repli) si les JOGs s'approchent trop dangereusement

Proposition de solution

Le langage Python a été choisi pour sa simplicité et rapidité d'emploi. De plus, afin de ne pas démarrer la programmation à partir de zéro, nous nous sommes servis de librairies déjà existantes :

- le SDK « naoqi » fourni par *Aldebaran Robotics* pour récupérer facilement les images capturées par webcam, et agir sur les actionneurs
- le module OpenCV « cv2 » pour ses algorithmes efficaces de traitement d'image (et c'est plus facile que de travailler sur les pixels à la main !)



Démarche du programme

L'image est d'abord capturée par la caméra supérieure : se baser sur celle du bas donne un mauvais résultat, et coupler les deux à la fois est difficile. Elle est ensuite convertie en matrice numpy lissée (pour réduire le bruit) dans l'espace HSV afin d'appliquer un filtre de couleur plus efficacement qu'en BGR. La matrice binaire ainsi obtenue permet de segmenter différentes tâches, dont la plus grande (en aire) est sélectionnée et localisée via son centroïde. Sa position par rapport au centre de l'image détermine le mouvement de tête à opérer pour recentrer la tâche et « suivre du regard » la cible, ainsi que les déplacements du NAO à opérer pour s'en rapprocher, s'en éloigner et ou se tourner vers elle ; le tout signalé par des LEDs indicatrices.

Les différents cas possibles se résument à :

- si la tâche est beaucoup trop petite, ne rien faire (c'est peut-être un bruit)
- si la tâche est petite, s'en approcher (asservissement bangbang en vitesse et cap) et ajuster la position de la tête (asservissement proportionnel)
- si la tâche est de bonne taille, simplement suivre avec la tête
- si la tâche est trop grande, reculer tout en suivant du regard

Résultats obtenus

Un NAO ainsi programmé est bien capable de suivre les JOGs à la fois du regard, mais aussi en se déplaçant. Cependant, il les perd parfois de vue, surtout si la cible se déplace trop vite. En effet, la connexion ssh limite le nombre d'images traitées par secondes : elles doivent passer par le réseau, ce qui oblige de choisir un compromis entre résolution temporelle et définition d'image (on a ici opté pour QQVGA 160x120). De plus, les JOGs étant vus de dessus, on ne détecte par forcément sa couleur orange. On doit l'aider en rajouter un objet de cette couleur sur les JOGs.

Néanmoins, le programme semble mieux fonctionner pour détecter des NAOs ! Comme leurs couleurs sont similaires à celles des JOGs et que leur hauteur correspond à celle de la caméra, ils sont plus facilement détectables. On peut ainsi imaginer un train de NAOs : le premier en tête de fil suit une cible, et les autres suivent derrière en fil indienne …

Liens utiles

http://www.aldebaran-robotics.com/documentation/dev/python/index.html

http://www.opencv.org.cn/opencvdoc/2.3.2/html/index.html