Projet d'imagerie - Traitement d'images dans l'embarqué

Étudiant : ROUSSEAU Sylvain
Encadrant : BÉRÉZIAT Dominique

Janvier - Mai 2015

1 Introduction

L'objectif du projet était de réaliser la partie vision d'un robot. Ce robot et son intelligence artificielle étaient réalisés par les M2 STL alternants, en vue de participer à la coupe de France de robotique organisée chaque année par Planète Science. [1] Lors de ce tournoi, les robots s'affrontent en duel lors d'épreuves d'une durée de 90 secondes sur un plateau de jeu spécialement conçu pour l'occasion. Chaque robot se voit attribuer à la fin de chaque partie des points en fonction de critères variants chaque année. Le robot ayant accumulé le plus de points est alors déclaré vainqueur. Cette année, il fallait entre autres réussir à capturer le maximum de pions, les ramenner dans son camp et les empiler. Le robot étant déjà équipé d'un système de positionnement et un système de détection de collisions à très courte portée, le but de la vision était de détecter les robots adverses à moyenne et à longue portée. Les robots ayant une vitesse importante (3 mètres par seconde dans notre cas), la détection se devait d'être rapide (10 images par seconde minimum) afin éviter les collisions et d'optimiser son chemin. L'algorithme devait s'éxecuter sur un Raspberry Pi 2 [2], possédant un processeur quadricore cadensé à 900 MHz, le capteur servant à l'aquisition des images étant le module vidéo du Raspberry Pi.

1. http://www.planete-sciences.org/robot/?section=pages&pageid=79
2. https://www.raspberrypi.org/products/raspberry-pi-2-model-b

2 De la détection de robot à la détection de balises

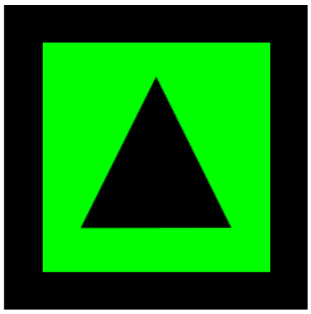


FIGURE 1 – Exemple d'une des faces de la balise

La première partie du projet consistait à passer d'une idée (détecter un robot adverse situé devant nous) à un algorithme. Le réglement du tournois autorise l'ajout d'une balise d'une taille maximale égale à un cube d'arêtes de longueur 8 cm que l'on peut placer sur le robot adverse. [3] Ceci permet alors de passer de la détection d'un objet d'une forme inconnue à la détection d'une sous image particulière. Une balise a alors été réalisée en parallèle avec un premier algorithme. L'image montrée en figure 1 est imprimée sur chacun des 4 cotés (on ne met rien au dessus et en dessous de la balise). On modifie la couleur en fonction de l'orientation de la face. Par exemple si la face est de couleur rouge cela nous indiquera que l'adversaire nous fait face tandis que si elle est verte, cela indiquera que l'on observe le flanc gauche du robot.

3 Premier algorithme

La première partie de l'algorithme consiste à détecter le carré noir qui défini le contour de notre balise. La première idée était alors de considérer un carré comme l'intersection d'un ensemble de quatre droites presque parallèles deux à deux (du fait de l'effet de perspective). Le premier algorithme utilise donc la transformée de Hough [4] pour détecter les cotés du carré. Cependant, lors du premier test en conditions réelles, nous avons pu observer que le plateau de jeu est trop grand pour qu'une telle technique puisse fonctionner efficacement. En effet, il fallait pouvoir détecter une balise de 8 centimètres de coté à plus de 3 mètres de distance. À cette distance la détection de droite ne fonctionne pas du fait de la faible taille et des parasites (le plateau de jeu, le fond, etc.). Nous avions également envisagé la possibilité d'affiner le positionnement du robot grâce à trois balises extérieures au plateau de jeu, ce qui s'avèrerait utile en cas de collision avec un autre robot, car, dans un tel cas, le robot perd son positionnement et doit retourner dans un coin de la table pour pouvoir se repositionner. Cette idée à finalement été abandonnée avec le deuxième algorithme. Il serait possible dans le futur d'adapter le deuxième algorithme pour ajouter

3. Page 25 du règlement : `http://www.planete-sciences.org/robot/data/file/coupe/2015/E2015_Rules_EU_EN_final.pdf`

4. Introduction à la transformée de Hough : `http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm`

de nouvelles balises. Dans l'infographie ci-dessous, voici un résumé détaillé du premier algorithme suivit d'une explication des étapes principales de la partie post-traitement de l'algorithme.

Détection de contours

Sous-échantillonage (Permet d'alléger les différents traitements, il reste léger pour éviter la perte de données).

Filtre médian, permet de supprimer le bruit pouvant géner la détection de contours.

Détection de contours. On utilise un filtre de Canny.

Filtrage des contours obtenu, on cherche sur l'image d'origine si les contours sont à coté de couleur correspondant à des faces.

Détection de zones d'intérêt

Détection des droites verticales et horizontales des contours de l'image. On utilise la transformée de Hough.

Les zones d'intérêt correspondent aux intersections des couples de droites parallèles verticales avec les couples de droites parallèles horizontales n'ayant pas une aire trop faible.

Classification des faces

Classification des faces en fonction des formes. Si on cherche un ennemi, on ne cherche qu'à savoir si la face possède un triangle. Si on cherche le positionnement du robot, on cherche seulement les 3 formes différentes correspondantes. Toute autre forme trouvée invalide la face potentielle.

Détection d'ennemie

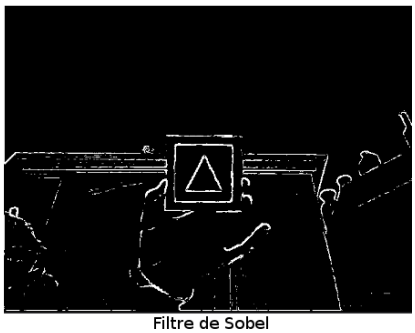
On retourne la position sur l'écran et la couleur de la face détéctée

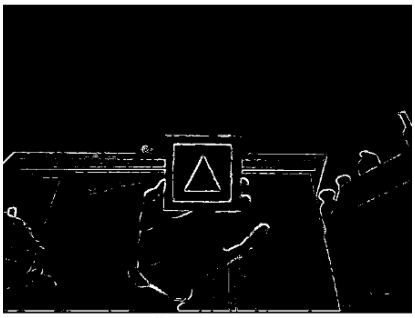
Positionnement du robot

On a besoin de détecter deux balises pour pouvoir calculer la position du robot. On doit éventuellement capturer une deuxième image.

La première partie consistant en une détection de contours, on a comparé l'efficacité de trois algorithmes courants dont on a optimisé les paramètres sur une image test.

FIGURE 2 – Image originale





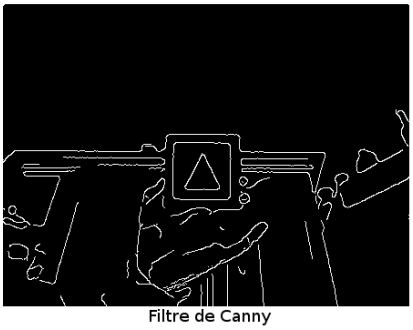


FIGURE 3 – Détection de contours

Le filtre de Canny permet d'obtenir des contours de taille 1 pixel et de meilleur qualité que les autres algorithmes. En contrepartie, celui-ci est plus lourd que les deux autres filtres (Sobel et son dérivé Scharr). L'étape la plus calculatoire de cet algorithme étant la détection de droites, moins le nombre de contours detectés est important, plus la détection de droites s'allège. On a donc choisi le filtre de Canny qui bien que plus lourd permet d'alléger en conséquence la partie suivante.
Pour finir, on filtre les contours obtenus en se rapportant à l'image d'origine. Pour ce faire, pour chaque pixel correspondant à un contour selon Canny, on regarde si un des pixels d'angle du carré de taille 5x5 centré sur le pixel correspond à la couleur d'une des faces recherchées (il faut alors défini les plages de couleur de chaque face).

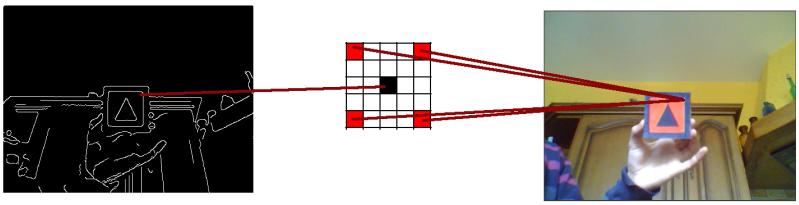


FIGURE 4 – Description du filtrage

L'utilisation d'un voisinage dans un carré de taille 5x5 a été faite après des test. En effet, la réponse dans un carré de taille 5x5 est meilleur sans alourdir les traitements (on reste à 4 pixels testés pour chaque contours possible). Près des bords l'imprécision du capteur fausse la couleur ce qui n'est pas totalement effacé par le filtre médian appliqué au début. Le résultat sur l'image de test est le suivant :

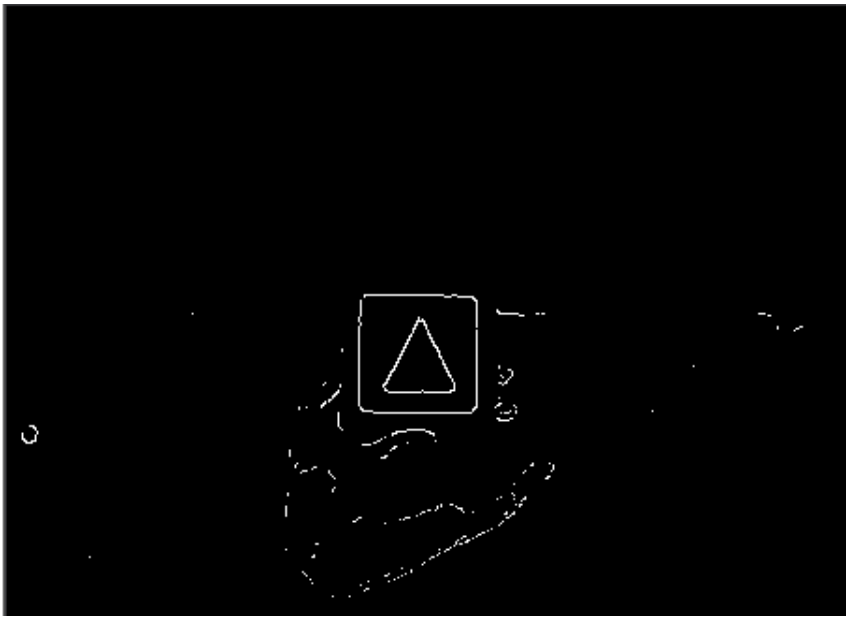


FIGURE 5 – Résultat du filtrage

Sur l'image de test, le filtre permet d'éliminer la majeure partie des contours inintéressants sur une image expérimentale. On trouvera plus tard en faisant des tests en réel, que les couleurs de la table de jeu correspondent parfaitement à celles des balises, et qu'elles sont très présentes ailleurs (dans le fond, sur les autres robots ...) ce qui génère énormément de bruit. De plus ce filtre est très sensible aux conditions de lumière qui sont dures à prévoir avant le concours. Voici un résumé de la chaine de pré-traitement du premier algorithme.

Image originale

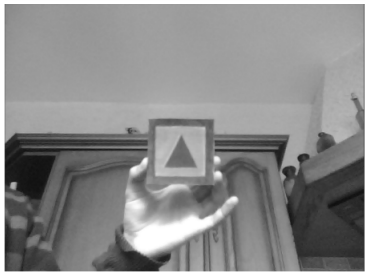
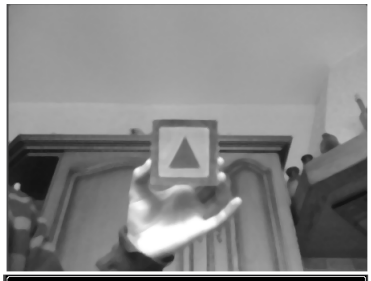
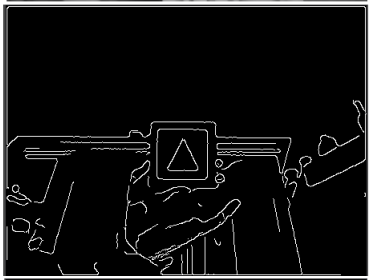


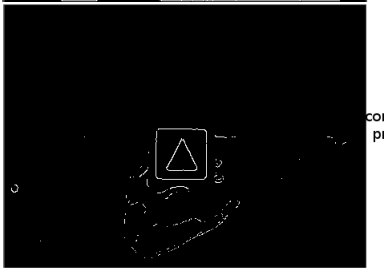
Image en niveau de gris



Filtre médiant



Détection de contours (Filtre de Canny)



Filtrage des contours obtenu précédemment

Lors de l'implémentation de la partie sur la transformée de Hough, les tests sur le terrain ont permis de se rendre compte que malgrè le pré-traitement, la détection de droite n'était pas possible au delà de quelques centimètres du capteur.

4 Deuxième algorithme

FIGURE 6 – détection de carrés (rouge) et de triangles (mauve)

Un deuxième algorithme a été créé , testé puis retenu. Il exploite principalement trois différents algorithmes. La détection de contours avec un filtre de Canny [5], l'algorithme d'analyse topologique par suivi de contours [suzuky85] [6] pour détecter des contours fermés sous forme de polygones et l'algorithme de Ramer–Douglas–Peucker [7] pour l'approximation de polygones (simplification des polygones précédemments calculés). L'idée est alors de réaliser une première détection de contours. On effectue ensuite une analyse topologique de ceux-ci qui nous permet d'extraire tous les contours fermés. Ensuite, on finit en simplifiant nos contours ainsi obtenus et on ne récupère que les contours correspondants à des carrés et à des triangles dans l'image. Les faces de nos balises correspondent alors aux zones de l'image où un triangle est contenu dans un carré tels que leurs aires et leurs rapports hauteur/largeur soient proportionnels. Pour obtenir l'orientation à laquelle correspond notre face, on détermine la couleur dominante dans notre carré. Cet algorithme se révèle suffisamment rapide pour s'éxecuter en temps réel sur un netbook de puissance équivalente à celle du Raspberry Pi. Des problèmes sur le robot on empéché les tests en conditions réelles. Lors de la réalisation de l'algorithme, les tests réalisés en première partie concernant la détection de contours ont pu entièrement être réutilisés. Bien paramétré, l'algorithme permet de détecter sans problème la balise à plus de 2 mètres du capteur.

5. [Canny86] J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

6. [*suzuki*85] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

7. `http://fr.wikipedia.org/wiki/Algorithme_de_Douglas-Peucker`

5 Implémentation

Pour l'implémentation du projet, pour des raisons de légèreté et de performance le choix du language c'est porté sur le C++. Ce code utilise la librairie OpenCV [8] qui implémente les trois algorithmes précédemments cités de manière optimale. Un outil permettant de configurer les variables du projet a été créé. Le robot n'ayant finalement pas été finalisé, aucun protocole d'échange des données n'a été mis en place. Cependant, un exemple d'utilisation est fait dans le "main" pour permettre aux étudiants de l'année prochaine de pouvoir réutiliser facilement ce projet.

6 Améliorations

Si l'année prochaine le code est réutilisé sur le futur robot, il y aura éventuellement des problèmes liés aux conditions réelles. Lors du mouvement, en fonction du capteur, il peut y avoir un flou cinétique. Il faudra alors essayer de le compenser en modifiant le filtre de Canny en pondérant la dérivée de l'image par rapport à x en fonction de la valeur du flou cinétique. Ceci nécessitera évidemment des tests préalables. Comme indiqué précédemment dans ce rapport il est envisageable d'ajouter d'autres balises. Il faudra alors définir des nouvelles formes pour les faces. On pourra remplacer la forme du centre par n'importe quel polygone (moyennant une petite modification du code).

7 Bibliographie

7.1 Détection de contours

Filtre de Sobel
Détection de contours basé sur les gradients de l'image (filtre d'ordre 1)

Filtre de Scharr
Filtre de Sobel optimisé

Filtre de Canny
Filtre ayant la meilleur réponse dans le cadre de notre projet, article d'origine : [Canny86] J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

7.2 Détection de formes

Détection de droites

Transformée de Hough
`http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm`

Transformée de Hough probabiliste
[*matas*00] Matas, J. and Galambos, C. and Kittler, J.V., Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. CVIU 78 1, pp 119-137 (2000)

8. `http://opencv.org/`

Détection de polygones

Suzuki et Abe

[*suzuki*85] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

Approximation de polygones

Algorithme de Ramer–Douglas–Peucker

`http://fr.wikipedia.org/wiki/Algorithme_de_Douglas-Peucker`

Librairies

La librairie utilisée pour l'ensemble du projet est OpenCV, le langage de programmation choisi est le C++.