Mathieu LENORMAND
Henri Junior YONGOUA-PAHO

2013

I.M.A. (Informatique Microélectronique Automatique)
POLYTECH' Lille
Ecole Polytechnique Universitaire de Lille

Mémoire de Projet

Robotino Joueur de Hockey

Tuteur enseignant :
Mme Claudine LECOCQ

Remerciements

Nous remercions Mme Claudine LECOCQ, de nous avoir encadrés pendant ce projet.

Nous remercions Mr Olivier SCRIVE, de nous avoir mis à disposition la salle Robotino et d'avoir répondu à quelques questions.

Introduction

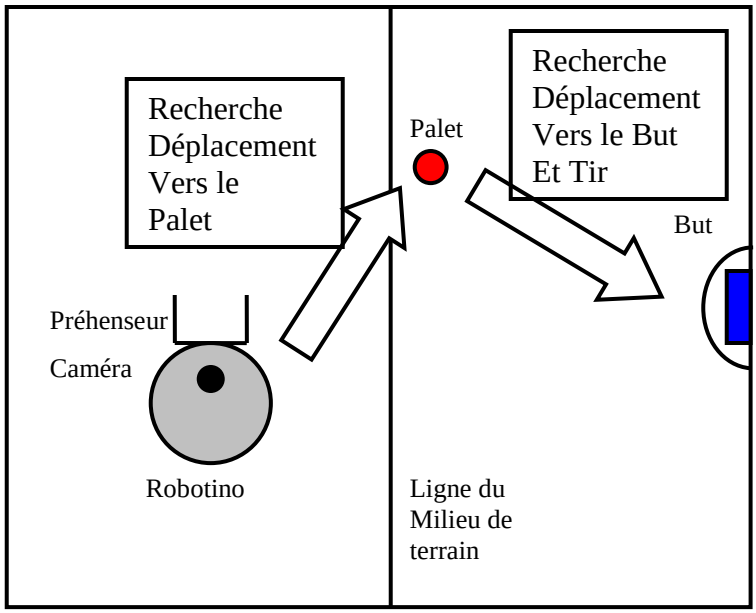
Objectif principal :

Tester les limites en traitement d'images de RobotinoView2, le logiciel de Festo pour programmer le Robotino, en s'inspirant de la Festo Robotino Hockey Challenge Cup ou RobotCup.

Il consiste à programmer un Robotino, pour :

- détecter un palet rouge avec des fonctions de traitement d'images
- diriger le Robotino vers ce palet pour que le préhenseur fixe l'entoure
- détecter une zone colorée qui représente un but
- diriger le Robotino vers le but
- lancer le palet dans le but

Ces étapes ont été les premières parties de notre programmation, ensuite notre objectif était de rendre ce programme plus robuste, c'est-à-dire prévoir des cas plus complexe, comme des problèmes de hors champ de la caméra, d'autres objets ou obstacles sur le terrain.



Développement

I) Description du contexte du projet

1) Présentation de Festo

Festo est un fournisseur mondial (car présent dans 176 pays) de technologies d'automatisation et un leader en matière de formation. Leur but est d'apporter à leurs clients une productivité et une compétitivité maximales.

La branche Festo France a été créée en 1958 et implantée en région parisienne (Bry-sur-Marne, 94), elle propose différentes gammes de services (tels que des bureaux d'études, une division didactique et un centre de relation clientèle) à but professionnel ou éducatif.

Festo développe des robots en s'inspirant de la nature, appelé « Bionic Learning Network », ce système d'étude utilise des principes naturels pour concevoir des applications technologiques et industrielles.

Par exemple : the Bionic Handling Assistant qui est un bras bionique qui s'inspire d'une trompe d'éléphant dont un prototype a été fourni à Polytech.

Dans un but pédagogique, Festo a aussi développé un robot mobile, le Robotino, sur lequel nous avons fait notre projet.

2) Description du Robotino

Principales caractéristiques du Robotino :

Robot mobile : Châssis rond en inox et trois unités de motorisation omnidirectionnelles
- Diamètre : 370 mm
- Hauteur (habillage compris) : 210 mm
- Poids total : environ 11 kg

Capteurs :

- 1 Structure de protection en caoutchouc intégrant un détecteur anticollision
- 9 capteurs de distance analogiques à infrarouge
- 1 capteur inductif analogique
- 2 capteurs optiques numériques
- 1 caméra Web couleur à interface USB et compression jpeg

Commande :

- 1 PC 104 plus embarqué avec OS Linux Ubuntu temps réel et plusieurs interfaces de communication :
- Ethernet
- 2 ports USB
- 2 liaisons RS232
- 1 port parallèle
- 1 connecteur VGA
- 1 Point d'accès Wi-Fi performant avec antenne, conforme à 802.11g et 802.11b, que l'on peut faire basculer en mode client et avoir un Cryptage WPA2 en option.

Platine de commande EA09 qui permet :
- Le pilotage de quatre moteurs à courant continu
- L'interface Ethernet pour accès externe direct à la régulation des moteurs
- L'intégration de composants électriques additionnels à l'aide de deux connecteurs d'E/S à 20 broches

RobotinoView2, le logiciel de programmation du Robotino conçu par Festo, dont la configuration requise est un PC avec Win 2000/XP SP2/VISTA/Windows 7. Le Robotino peut être programmé avec d'autres logiciels (tels que Matlab) et en utilisant d'autres langages de programmation comme le C++ et Java.

Ce robot peut être utilisé pour des Travaux Pratiques, des projets ou des concours comme le Festo Robotino Hockey Challenge Cup.

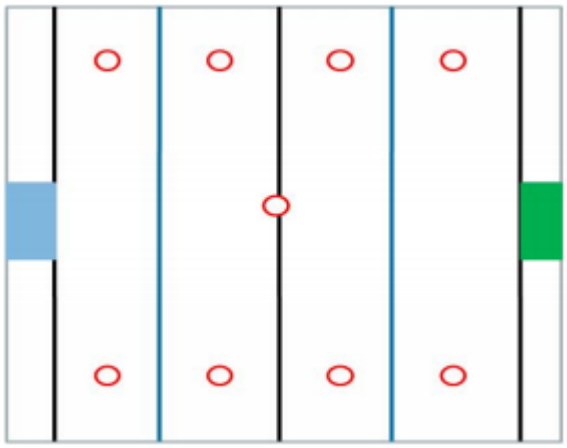
3) Présentation de la Festo Robotino Hockey Challenge Cup

Le but du jeu est de jouer au hockey. Il y a 2 équipes de 3 Robotinos chacune. Chaque robot possède le même dispositif fixe pour frapper une rondelle de hockey.

Les dimensions :

Les dimensions des robots sont délimitées par un cylindre de rayon 0,4m et une hauteur 0,3m. Les palets sont des disques circulaires de couleur rouge avec un diamètre de 10 cm. la compétition se déroule dans une arène constituée d'un champ de 4,5 m x 4,5 m. Le champ est délimité par des plaques ayant une hauteur de 0,5m, ce qui permet de s'assurer que les

caméras des robots ne seront pas gênées par des objets en dehors de l'arène. Le champ est divisé par 5 lignes. Il y a une ligne noire centrale qui divise le champ en deux parties égales.



- Le jeu démarre au centre du terrain. Ce point central est caractérisé par un disque circulaire de diamètre de 15 cm.

- Aux extrémités du terrain, il y a une ligne noire parallèle à la ligne médiane. Cette ligne est appelée la ligne de but. La distance à l'extrémité est de 25 cm.

- Toutes les lignes noires ont une largeur de 38 mm.

- La surface de but est marquée en vert ou bleue sur les parois latérales. La largeur de l'objectif est de 80 cm.

- Devant les buts, il y a des demi-cercles de rayon = 40 cm. Ces lignes se composent de bandes métalliques qui peuvent être détectés par des capteurs inductifs.

- Il existe deux autres lignes bleues qui divisent le terrain en trois parties égales : une région d'attaque et deux zones défensives. Ces lignes sont en fait des bandes métalliques (couleur argent) qui peuvent être détectées par des capteurs inductifs.

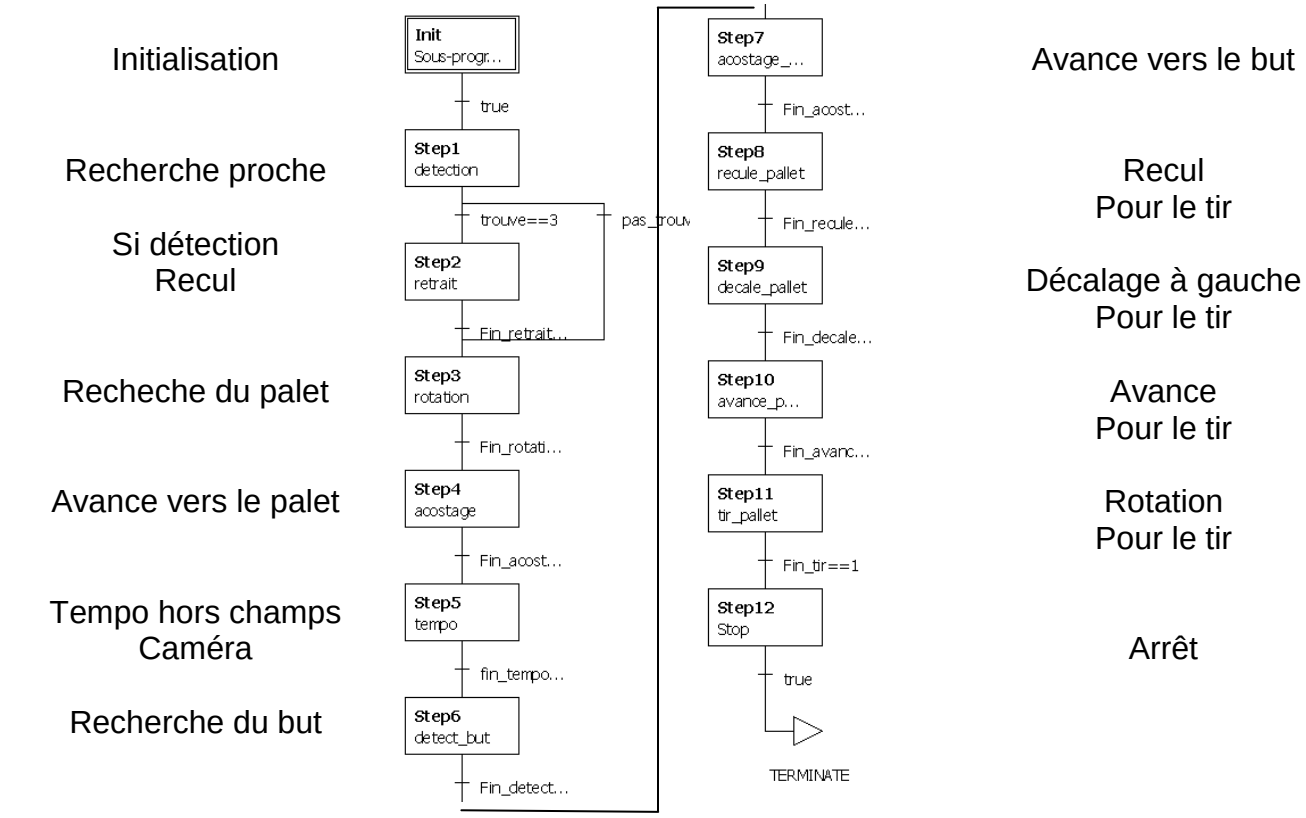
- La largeur sera de 50 mm sans les bandes métalliques.

- Il y a 4 points dans la zone d'attaque et 4 points dans les deux zones défensives. Ces points sont marqués par des disques circulaires noirs de 10 cm de diamètre.

II) Découpe du projet en différentes tâches

1) Programme principal du Robotino joueur de Hockey

Le programme principal sur RobotinoView2 est sous forme de Grafcet :



Pour la plus part des transition,

La plupart des transitions ne sont pas totalement visible à cause du copier/coller de l'image du Grafcet. Pour la plus part des transitions, c'est variable = 1, 2 ou 3.

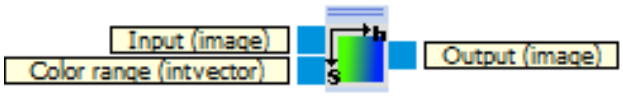
2) Traitement d'image : Reconnaissance de la rondelle et du but

Les détections du palet et du but sont sensiblement les mêmes d'un point de vu traitement d'image à la seule différence que le palet est rouge et le but bleu.

On obtient une image grâce à la caméra USB placée au-dessus du Robotino.
On peut visualiser les images issues de la caméra et les transmettre avec ce bloc :

Pour ensuite pouvoir effectuer un traitement de ce résultat.

Ensuite on l'a relié au bloc suivant :



Nommé Recherche de plage colorimétrique (Color Range Finder), Ce bloc nous permet de sélectionner sur l'image, une zone de couleur et en relâchant le bouton de la souris, le bloc garde en mémoire la valeur, la teinte et la saturation de la plage colorimétrique choisie.
Ce floc fait la recherche de la couleur sélectionnée, dans toutes les images de la vidéo prises par la camera. En sortie de ce bloc, on obtient une image binarisée avec en blanc toutes les parties de l'image dont la couleur appartient à la plage sélectionnée et le reste en noir. Une fois la plage colorimétrique sélectionnée. Ensuite on a relié le bloc précédent, à 2 fois le bloc suivant :

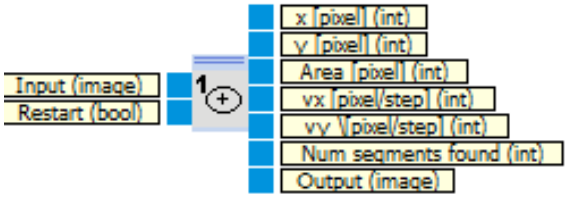
Nommé Filtre (Filter)
Dans lequel on peut choisir le type de filtre (Maximum ou Minimum) et la taille de masque (allant de 3 à l'infini (si on dépasse la taille de l'image cela devient inutile) par pas de 2 en 2)

Le premier filtre a un type Minimum et une taille de masque égale à 7, ayant pour but de réaliser une érosion.

Le deuxième filtre a un type Maximum et une taille de masque égale à 7, ayant pour but de réaliser une dilatation.

L'ensemble réalisant une ouverture qui sert à retirer les défauts captés ou les objets non désirés.

Ensuite on a relié le bloc suivant :



Nommé Traqueur de segment (Segment tracker)
Ce bloc a pour but de labelliser l'image c'est-à-dire de répertorier les parties de l'image qui répondent aux critères rentrés dans ce bloc.
On peut choisir les critères suivants :

- Surface maximale : le segment traqué est celui dont la surface est la plus grande par rapport aux autres segments trouvés.
- Centre de l'image : le segment traqué est celui qui est le plus proche du centre de l'image.
- En bas : le segment traqué est celui qui est le plus proche du bord inférieur de l'image.
- En haut : le segment traqué est celui qui est le plus proche du bord supérieur de l'image.
- A gauche : le segment traqué est celui qui est le plus proche du bord gauche de l'image.

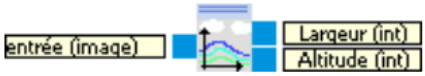
- A droite : le segment traqué est celui qui est le plus proche du bord droit de l'image.

Il est possible de combiner 2 caractéristiques, par exemple, si on met un traqueur de segments réglé en bas puis un autre réglé à droite, l'ensemble traquera le segment le plus proche du coin en bas à droite.

On peut modifier aussi le paramètre Surface minimale [en pixel], ce paramètre peut varier de 0 à l'infini (si on dépasse la taille de l'image cela devient inutile), et il sert à ne pas prendre en compte les segments dont la surface est plus grande que la surface minimale spécifiée

Nous avons choisi pour le traqueur de segment :
Initialisation : Surface maximale
Surface minimale [en pixel] : 100
Car c'était la distance la plus adaptée dans le périmètre dans lequel nous travaillons généralement.

En parallèle à la sortie du deuxième filtre, on a relié le bloc suivant :



Nommé Information d'image.
Ce bloc donne en sortie la largeur et la hauteur de l'image, dans notre cas :
Largeur : 300 pixels
Hauteur : 230 pixels

Dans la suite du projet, nous avons utilisé l'information de la largeur pour pouvoir centrer la caméra et par la même occasion l'axe du Robotino dans l'axe de l'objet recherché ou de celui du but.

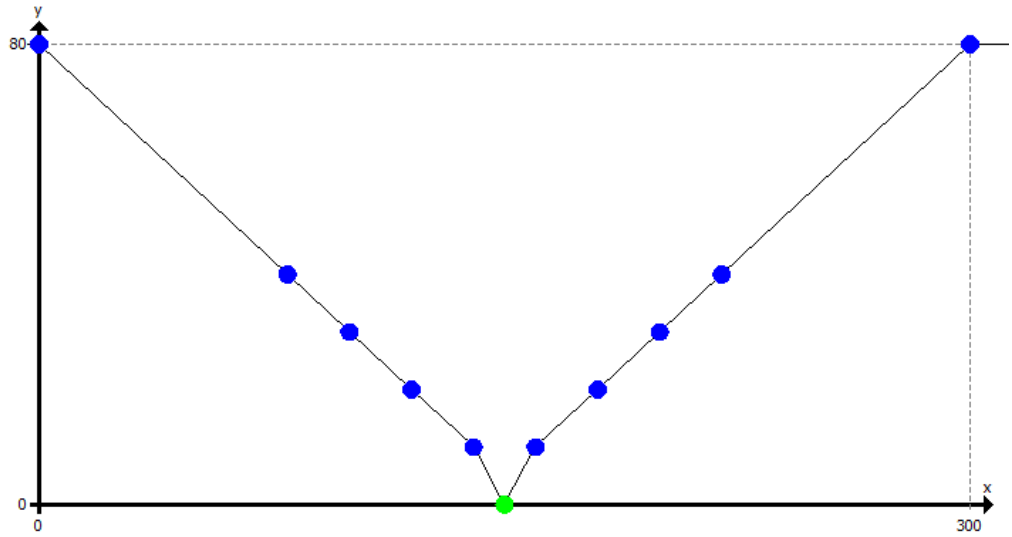
Pour cela, nous faisons tourner le robot sur lui-même avec une vitesse de 80 mm/s qui décroît progressivement dès-que le palet est dans les champs de la caméra pour s'arrêter quand le centre du palet se situe à + ou – 10 pixels de la droite qui forme le centre de la caméra, à ce moment l'axe du Robotino est dans l'axe du palet.

Pour cela nous avons utilisé un bloc appelé fonction de transfert :

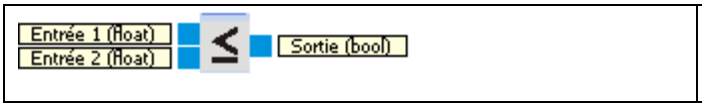
Ce bloc de fonction permet de transformer le signal d'entrée x en un signal de sortie y quelconque.

Comme exemple d'utilisation de ce bloc fonction, la transformation pixel-vitesse pour la détection du but ou du palet :

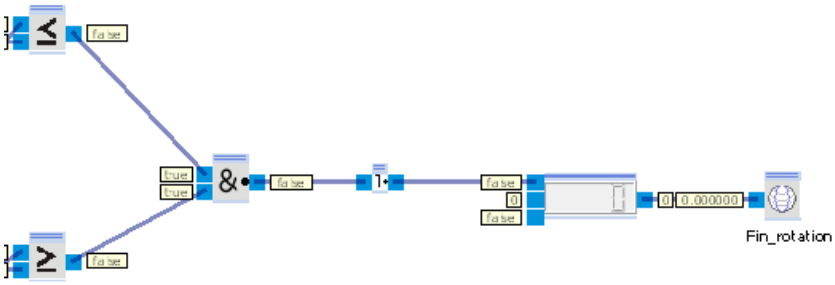
	x	y
0	0	80
1	80	40
2	100	30
3	120	20
4	140	10
5	150	0
6	160	10
7	180	20
8	200	30
9	220	40
10	300	80



0 et 300 sont les valeurs extrêmes(en pixel) de la caméra, donc plus le centre de l'objet ou du but est proche de l'axe de la caméra (c'est-à-dire 150), plus on le fait ralentir car au début nous faisions passer la vitesse de 80 mm/s à 0. Mais avec l'énergie cinétique du Robotino, lors de son arrêt, l'objet n'était plus exactement dans le centre de la camera.
A côté on utilise deux comparateurs : inférieur ou égal et supérieur ou égal :



On fait la différence et une sommation entre la valeur 150 (milieu de la caméra) et une variable égale à 10. Comme ça on s'assure que le centre du palet ou du but est centré entre 140 et 160 pixels sur la largeur de la caméra.



Et avec un et logique (&), un inverseur et un compteur qui permet de faire passer une variable à 1 pour changer de sous programme.

Remarque :

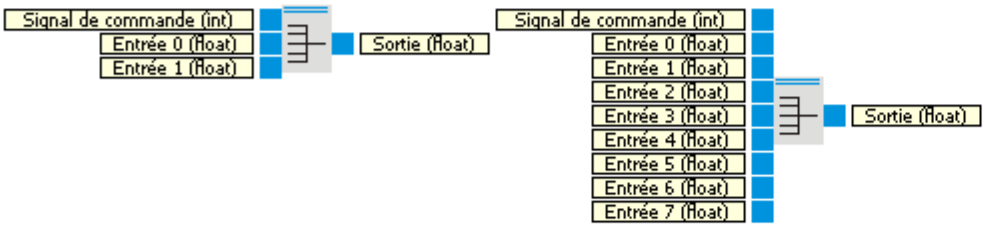
L'une des différences entre la recherche du palet et celle du but est que pour la recherche du but on a rajouté une fonction de transfert selon la vitesse en x du Robotino. Car à cette étape là, le robot a le palet dans son préhenseur ; et quand on ne mettait pas de vitesse selon x, le Robotino perdait le palet à cause de la force centrifuge.

3) Déplacements

On utilise aussi un bloc fonction de transfert qui permet de faire avancer le Robotino. Plus il est loin de son objectif, plus il va vite.

	x	y
0	0	500
1	300	0

C'est-à-dire 0 c'est que l'objet est tout en haut de la caméra, ce qui équivaut au plus loin dans le champ de vision, il roule à 500 mm/s. Et quand il est à 230 ce qui équivaut au plus près du champ de vision il est à 115 mm/s. Cependant cette vitesse n'est envoyée que si le robotino capte un objet grâce aux capteurs de distance. La vitesse de 115 mm/s est envoyée au robotino grâce au bloc Multiplexeur. Ce bloc est le suivant :



Le signal de commande est relié à la sortie du traqueur de segment qui correspond au nombre de segments trouvés.

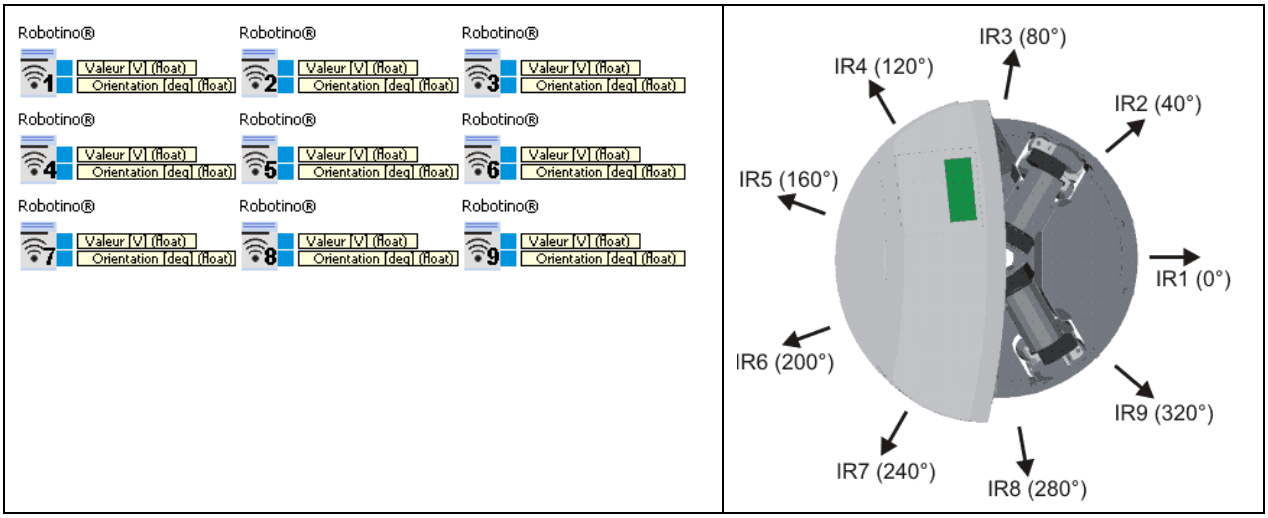
Quand le traqueur de segment ne détecte plus de segment on suppose que le palet n'est plus dans le champ de vision et on passe à la deuxième entrées du multiplexeur qui correspond à l'accostage de l'objet avec le préhenseur décrite dans la partie 3.

Remarque :

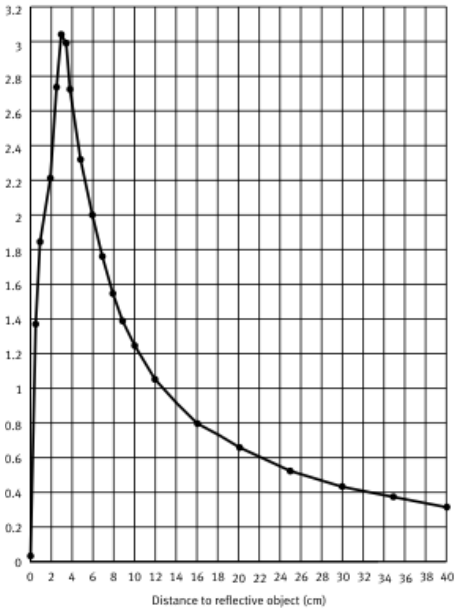
La différence entre la recherche du palet et celle du but est que le but n'est jamais hors du champ de vision de la caméra car le but en beaucoup plus grand que le palet (environ 50 x 30 cm). De plus, on fait arrêter le Robotino quand il est à environ 50 cm du centre du but pour pouvoir tirer, ce qui n'est pas le cas pour le palet, que l'on atteint complètement.

4) Accostage de l'objet avec le préhenseur

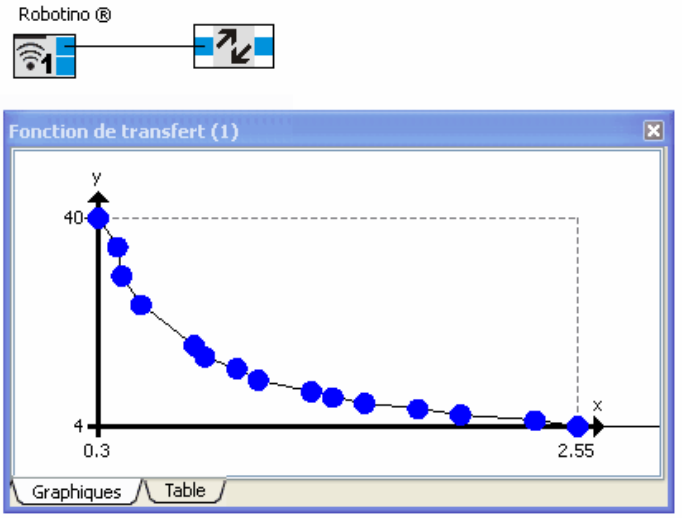
C'est la partie du sous programme qui consiste à aller vers l'objet, durant laquelle le palet est hors du champ de la caméra. Durant cette partie nous utilisons un des capteurs de distance analogiques à infrarouge celui numéroté 1 (celui qui est devant dans l'axe de la caméra et du préhenseur) :



Dans la documentation de RobotinoView2, on trouve la fiche technique du capteur de distance (un Sharp GP2D120), qui nous donne la courbe illustrant la relation entre la distance de l'objet en cm et le signal de sortie analogique en volts :



Nous avons donc copié ces valeurs dans un bloc fonction de transfert :



Le capteur détecte un objet jusqu'à un maximum de 40 cm. Pour notre projet, nous avons pris comme limite de distance, 30 cm.

Tension (en Volts)	0.3	0.39	0.41	0.5	0.75	0.8	0.95	1.05	1.3	1.4	1.55	1.8	2	2.35	2.55
Distance (en cm)	40	35	30	25	18	16	14	12	10	9	8	7	6	5	4

Sur la deuxième entrée du multiplexeur décrit dans la partie 2, cette entrée est sélectionnée quand le palet est hors champ**,** on envoie la distance multiplié par 10

Quand la valeur du capteur est égale à 6 cm, on déclenche un bloc de temporisation :

Qui nous permet de le faire avancer encore pendant une seconde car quand nous mettions une valeur plus petite que 6 cm, on a rencontré des problèmes pour des valeurs plus petites

Une fois que le bloc de temporisation a atteint une seconde une variable est incrémentée qui permet de passer au sous programme suivant qui est la recherche du but précédemment décrit.

5) Lancement du palet pour marquer un but

Une fois arrivé à environ 50 cm du but le Robotino entre dans le sous programme de lancement du palet peut être divisé en 4 phases :

- recul

Pour faire reculer on lui envoie une consigne de -150 mm/s tant que la distance détectée par le capteur est égale à 11 cm, pour que cela soit un juste supérieur à la longueur du préhenseur

- décalage à gauche

Pour faire ce décalage on utilise un générateur d'onde arbitraire :

Ce bloc nous permet de générer une consigne pendant un temps voulu, c'est pourquoi on envoie un créneau de 150 mm/s pendant 2 secondes qui nous permet avec l'inertie du robot de se décaler de la largeur du Robotino

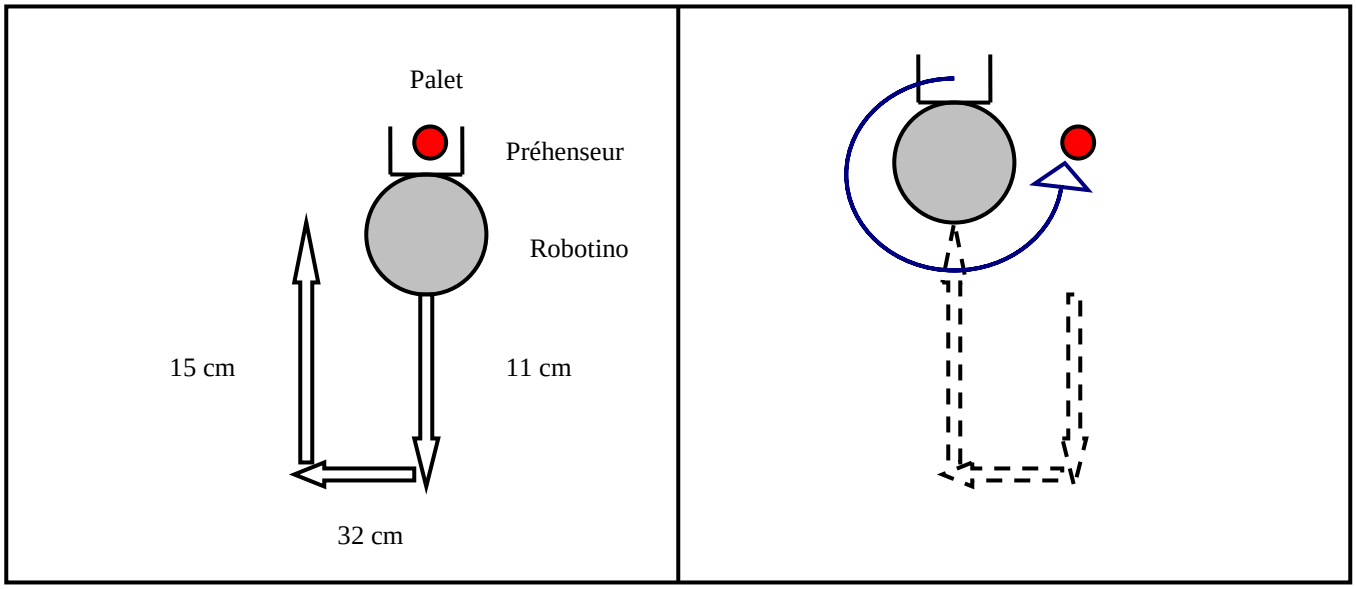
- avance

Pour faire l'avance, on réutilise un générateur d'onde arbitraire qui envoie un créneau de 150 mm/s pendant 1.1 seconde qui nous permet avec l'inertie du robot d'avancer d'une distance supérieure de celle du recul pour être à la hauteur du palet

- rotation

Pour faire le tir, on réutilise un générateur d'onde arbitraire qui envoie un créneau de 1000 deg/s pendant 1 seconde qui permet au Robotino de faire plus d'un tour sur lui-même

Illustration :



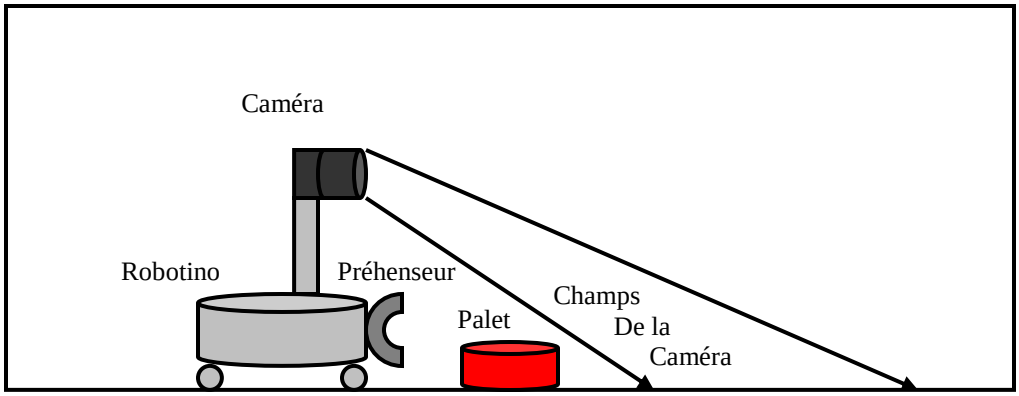
Déplacements du Robotino Pendant la phase de tir	Position finale Juste avant la rotation pour le tir

III) Robustesse du programme

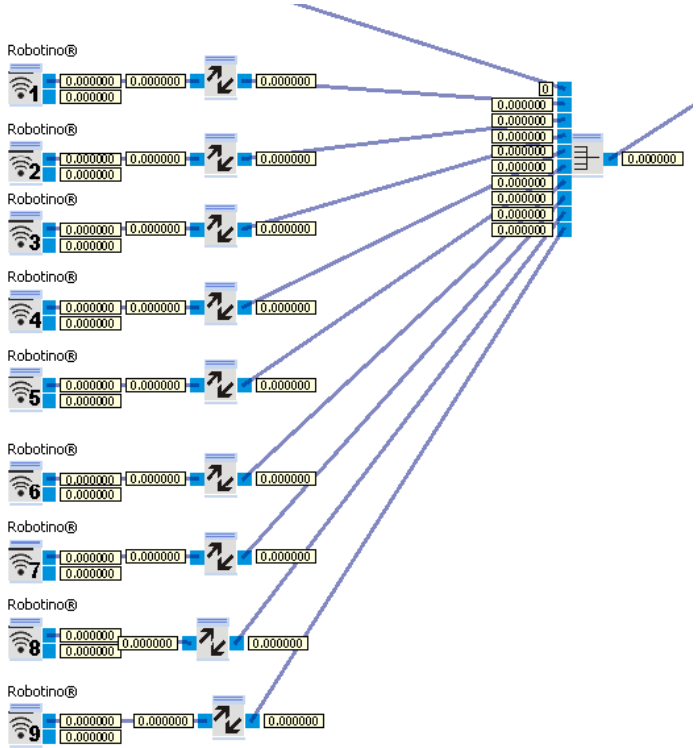
1) Traitement d'image : Problèmes possibles

1-a) Problèmes de hors champ : objet trop prêt

Un problème était si le palet était hors champ de la caméra car trop près du Robotino.



Pour cela au début, on commence par regarder si un des capteurs détecte un objet, en reliant les capteurs à un multiplexeur, comme ceci :



Pour vérifier, on regarde qu'un autre capteur adjacent détecte aussi l'objet.

Si c'est le cas on fait reculer le Robotino d'environ 50 cm dans la direction opposée au capteur qui a la distance la plus petite de l'objet. Cette étape peut prendre jusqu'à 6 secondes.

1-b) Problèmes de hors champ : objet trop loin

Evidement comme toutes caméras, celle que nous utilisons a une distance maximale de vision et quand l'objet est trop loin, la détection n'est pas possible.
Pour palier à ce problème, nous avons pensé à déplacer le Robotino sur le terrain. Pour cela, on pensait utiliser soit les capteurs pour détecter les murs, soit une méthode endométrique qui consiste à faire un parcours avec le Robotino en mémorisant le nombre pas fait par chaque moteur pour pouvoir refaire le même parcours ensuite. Nous avons utilisé cette méthode en TP de Robot 1, pour faire aller le Robotino de la salle « Robotino » à la salle « Kuka ».
Cependant pour cela il faut partir du même point à chaque fois ce qui ne convenait pas à notre projet.

Nous avons donc laissé ce problème de côté pour plus nous intéresser aux problèmes de traitement d'image.

1-c) Problèmes avec d'autres objets

Un autre problème serait que le palet soit caché par un objet. Pour cela il faudrait déplacer le robot, ce qui nous ramène au problème précédent.

En laissant cette partie de côté, il resterait à distinguer plusieurs objets différents, pour cela il y a plusieurs paramètres que l'on peut utiliser tel que :

- La couleur
- La taille
- La forme

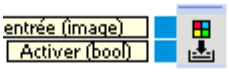
Le problème la détection de couleur est le suivant : Si on est en présence de 2 objets de même couleur, le problème n'a pas évolué. Et cette technique est très influencée par la luminosité et l'éclairage.

Pour la taille, le traqueur de segment de RobotinoView2 donne une taille, mais en pixels, ce qui n'est pas facile à utiliser. Nous avons donc pensé à utiliser un autre logiciel pour le traitement d'image.

2) Possibilités envisagées de l'utilisation d'un autre logiciel

Pour utiliser un autre logiciel, on devait vérifier que l'on pouvait extraire et introduire des images dans RobotionView2.

Pour cela il y a deux blocs sur RobotinoView2 dans la partie échange de données :
- un bloc : enregistreur d'image (qui permet d'enregistré une image capturée par la caméra dans un dossier choisi)
- un bloc : lecteur d'image (qui permet de lire une image dans un dossier choisi)

L'enregistreur d'image	Le lecteur d'image
	

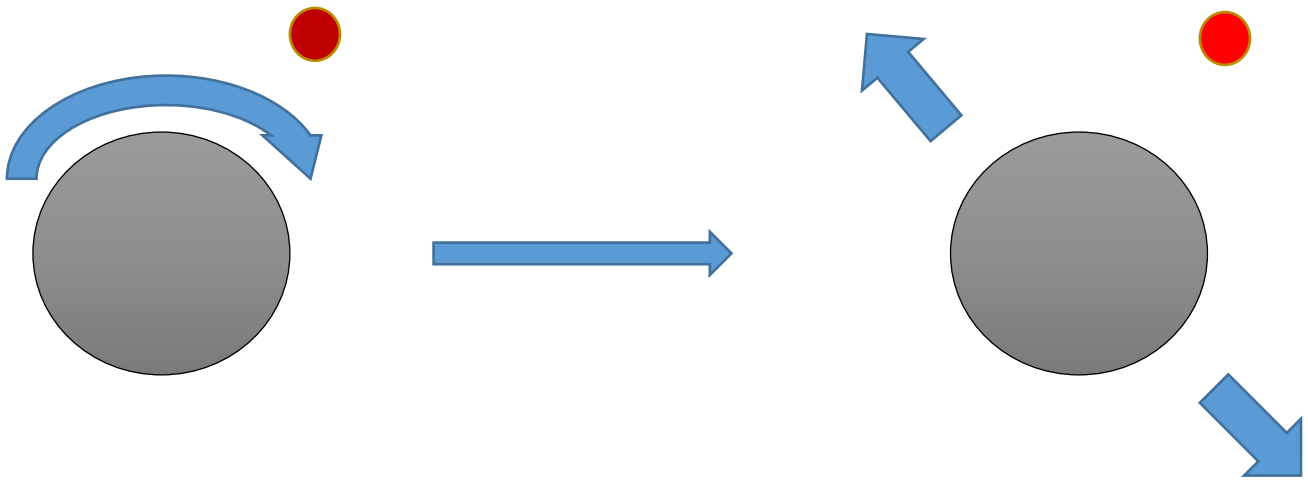
Ces blocs nous permettent d'extraire une image de RobotinoView2 issue de la caméra, de traiter par Matlab et que RobotinoView2 récupère le résultat pour l'utiliser

Nous avons écrit un programme sur le quel nous avons fait des essaies de traitement d'image sur Matlab, en particulier pour obtenir une image binarisée et en tirer des informations (comme la taille, la largeur et la hauteur)

A notre stade on arrive a enregistré une image prise par la caméra dans un dossier. On peut traiter l'image avec Matlab. Et enfin faire lire le résultat par RobotinoView2 dans un autre dossier où sont stockées les images résultats de Matlab. Le problème est que chaque étape est lancée manuellement et qu'à ce stade, nous n'arrivons pas à utiliserRobotinoView2 et Matlab simultanément en temps réel.

3) Autre Robotino en position de gardien

Nous avons aussi rédigé un autre programme nommé Gardien qui tourne d'un angle de 180 degrés devant le but tant qu'il n'a pas trouvé le palet.
Une fois le palet détecté, le Robotino oscille de façons latérales selon l'axe perpendiculaire à la direction du palet devant le but pour contrer le tire : comme le fait le joueur dans les but au Babyfoot.



Conclusion

➔ Etat d'avancement du projet :

Nous pensons avoir répondu aux objectifs du projet, c'est-à-dire l'utilisation du Robotino pour aller chercher un palet l'amener à une zone représentant le but et à effectuer un tir.

Cependant il reste quelques améliorations à apporter pour les cas particuliers (comme les cas de hors champs de la caméra, avoir d'autres objets sur le terrain…) et la possibilité de faire jouer deux Robotino à la fois. Nous avons commencé à mettre en place des réponses plus ou moins avancées pour ces différents cas. Mais même si nous n'avons pas un programme qui répond à tous cas particulier, nous pouvons dire que nous y avons réfléchis et que nous avons des pistes.

➔ Bilan des compétences techniques acquises :

Ce projet nous a permis d'utiliser un Robotino et de découvrir un nouveau logiciel de programmation avec RobotinoView2, dont le développement se fait avec des blocs qui représente des fonctions plus ou moins évoluées (allant de la simple adition à une fonction de transfert ou une fonction de traitement d'images). Et dont le programme principal s'écrit comme un Grafcet.

Au niveau logiciel, nous avons pu commencer à voir les possibilités de traitement d'images avec Matlab.

➔ Bilan des compétences acquises en gestion de projet :

Ce projet nous a permis d'acquérir une meilleure gestion de projet, en particulier, à mieux gérer les imprévus, comme avec les disponibilités de matériels, de salles…

Ce sujet, nous a permis aussi d'avoir un recul sur nos connaissances en traitement d'images. Car au début du projet, en découvrant le logiciel RobotinoView2, nous avons vu les fonctions de traitements d'images qu'il contient (qui sont listés en Annexes) et comme nous n'avions pas encore commencé les Travaux Pratiques de traitement d'images, cela nous paraissait beaucoup. Cependant arrivé au stade du projet où nous avons rajouté des problèmes liés aux traitements d'images et que dans le même temps nous faisions nos T.P. sur Aphelion qui est un logiciel de traitement d'images. Et grâce à ce logiciel et au recul, on a vu que l'on peut tirer beaucoup plus d'informations sur une image avec certains logiciels qu'avec les fonctions du RobotinoView2.

Ce problème nous a montré encore l'intérêt de ne pas partir tête baissée dans un sujet et de bien définir les objectifs. Car l'objectif principal était le traitement d'image et que nous avons un peu privilégié la prise en main du Robotino avec RobotinoView2

Annexe

Document décrivant :
Un exemple de code Matlab

```
%clear all
%close all
%lecture de l'image
image = imread('palet.bmp');
seuil = graythresh(image);
seuil=seuil-0.185
im_bw = im2bw(image);%,seuil);
%bu = bwareaopen(im_bw,26);
bu1 = imclearborder(im_bw);
imshow(im_bw),figure,imshow(bu1);
cc = bwconncomp(~im_bw,26);
L = labelmatrix(cc);
clc
figure('Name', 'Génération des résultats');
%detection des formes
for object_label = 1:1:cc.NumObjects

    [row,col] = find(L==object_label);
    object = bwselect(~im_bw,col,row,8);
    area_state=regionprops(object,'Area');
    perimetre_state=regionprops(object,'Perimeter');
    P_sur_S_pratiq = perimetre_state.Perimeter/area_state.Area;
    cote = perimetre_state.Perimeter / 4;
    P_sur_S_theo_carre = (4 * cote) /(cote * cote);
    rayon = perimetre_state.Perimeter / (2*pi) ;

    P_sur_S_theo_cercle = (2*pi*rayon)/(pi*rayon*rayon);
    carre = P_sur_S_pratiq - P_sur_S_theo_carre;
    cercle = P_sur_S_pratiq - P_sur_S_theo_cercle;
%classification
    if abs(carre) < abs(cercle)
        disp='un carr';
    else
        disp='un disque';
    end
%affichage des formes avec les descriptions
    subplot(3,4,object_label)


    imshow(label2rgb(object,'bone','b','shuffle'));
    xlabel(disp);

end
```

Document décrivant :
La liste des fonctions de traitement d'Image sur RobotinoView2

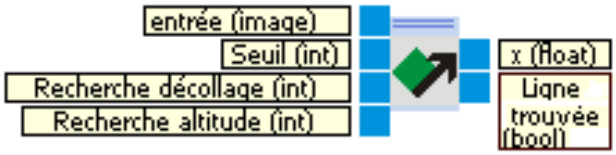
Pise de vue : Caméra

On obtient une image grâce à la camera USB placé au dessus du Robotino.
On peut visualiser et transmettre le résultat avec ce bloc :

Pour ensuite pouvoir effectuer un traitement de ce résultat.

Traitement d'image :

1) Détecteur de ligne :

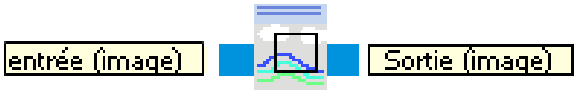


Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Seuil	int	0	Définit la sensibilité de l'algorithme au bruit dans l'image. Pour atténuer le bruit, il faut choisir une valeur de seuil plus élevée. Plage de valeurs : [0,255]
Début de recherche	int	20	La recherche d'une ligne débute au bord inférieur de l'image à la valeur indiquée.
Hauteur de recherche	int	20	L'image est analysée de bas en haut à la recherche de lignes. La hauteur de recherche définit le nombre de lignes d'image à prendre en compte pour une détection de ligne.
Sorties			
x	int		Position x de la ligne trouvée au sein de la fenêtre de recherche.
Ligne trouvée	bool		Vrai (true) si une ligne a été trouvée, sinon faux (false)

Ce bloc permet de trouver des lignes dans une image.

La zone dans laquelle une ligne est recherchée, est repérée par les deux lignes rouges horizontales. La ligne inférieure définit par la variable « début de la recherche ». La ligne supérieure a pour coordonnées en y la somme de la valeur de la variable « début de la recherche » + la valeur de la variable « Hauteur de recherche ». La zone délimitée par les lignes constitue la fenêtre de recherche.

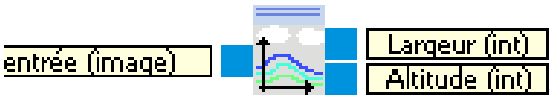
2) ROI (Region Of Interest) :



Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Sorties			
Sortie	image		L'image d'entrée complétée par la ROI. Les opérations de traitement d'image ci-après s'appliquent à la région sélectionnée

Ce bloc permet de sélectionner dans l'image d'entrée une région d'intérêt avec la souris.

3) Information d'image :



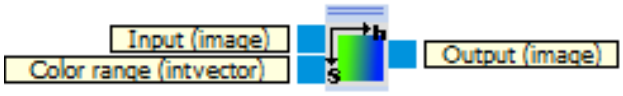
Entrées	Type	Standard	Description
Entrée	image		Image d'entrée
Sorties			
Largeur	int		La largeur de l'image en pixels
Hauteur	int		La hauteur de l'image en pixels

4) Conversion d'espace de couleurs :

Entrées	Type	Standard	Description
Entrée	image		Image d'entrée

Sorties			
Sortie	image		Image convertie

5) Recherche de plage colorimétrique (Color Range Finder) :



Entrées	Type	Par défaut	Description
Entrée	image		Image d'entrée
Plage colorimétrique	int vecteur	(0 0 0 0 0 0)	Définit la plage colorimétrique recherchée dans l'image. Les paramétrages effectués dans le dialogue seront écrasés.
Sorties			
Sortie	image		Image de niveaux de gris

Ce bloc permet de sélectionner sur l'image une zone de couleur et en relâchant le bouton de la souris, le bloc garde en mémoire la plage colorimétrique choisie au format TSV.

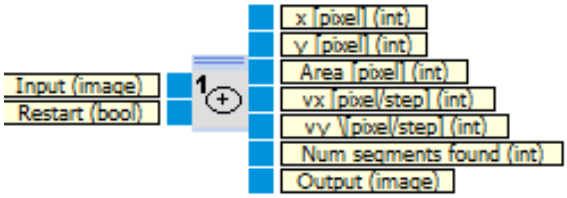
TSV signifie « teinte, saturation, valeur » ; on utilise aussi les termes anglais **HSV** *(hue, saturation, value)* ou **HSB** *(hue, saturation, brightness)*.
C'est un espace colorimétrique, défini en fonction de ses trois composantes :

- teinte : on code la teinte suivant l'angle qui lui correspond sur le *cercle des couleurs* :
  - 0° ou 360° : rouge ;
  - 60° : jaune ;
  - 120° : vert ;
  - 180° : cyan ;
  - 240° : bleu ;
  - 300° : magenta.
  - la teinte varie entre 0 et 360, mais est parfois normalisée en 0–100 % ;
- saturation : l'« intensité » de la couleur :
  - elle varie entre 0 et 100 % ;
  - elle est parfois appelée « pureté » ;
  - plus la saturation d'une couleur est faible, plus l'image sera « grisée » et plus elle apparaitra fade, il est courant de définir la « désaturation » comme l'inverse de la saturation ;
- valeur : la « brillance » de la couleur :
  - elle varie entre 0 et 100 % ;
  - plus la *valeur* d'une couleur est faible, plus la couleur est sombre. Une *valeur* de 0 correspond au noir.

On peut aussi directement entrer un vecteur à 6 composantes (teinte : minimum et maximum, saturation : minimum et maximum, valeur : minimum et maximum)

En sortie de ce bloc on obtient une image binarisée avec en blanc tous les parties de l'image dont la couleur appartient à la plage sélectionnée et le reste en noir.

6) Traqueur de segment (Segment Tracker) :



Entrées	**Type**	**Unité**	**Par défaut**	**Description**
Entrée	image			Image d'entrée
Redémarrage	bool		false	La traque de segment est réinitialisée si l'entrée est vraie. Ceci permet de passer, en cours d'exécution du programme, au segment le plus grand ou le plus proche du centre de l'image. Si l'on ne souhaite pas traquer de segment mais simplement trouver le segment le plus grand, le plus proche du milieu de l'image ... il suffit de mettre cette entrée en permanence à vrai.
Sorties				
x	int	pixels		Coordonnées x en pixels du centre de gravité du segment traqué
y	int	pixels		Coordonnées y en pixels du centre de gravité du segment traqué
Surface	int	pixels		Nombre de pixels dont le segment traqué est constitué
vx	int	pixels/pas		Indique de combien de pixels le segment traqué s'est déplacé depuis le dernier pas. Nombre positif : mouvement vers la droite. Nombre négatif : mouvement vers la gauche.
vy	int	pixels/pas		Indique de combien de pixels le segment traqué s'est déplacé depuis le dernier pas. Nombre positif : mouvement vers le bas. Nombre négatif : mouvement vers le haut.
Nombre de segments trouvés	int			Nombre de segments trouvés
Sortie	image			Image en noir et blanc avec les informations sur les segments trouvés. Un traqueur de segment connecté en

				aval peut traquer un autre segment

Ce bloc à pour but de labelliser l'image c'est-à-dire de répertorier chaque objet.
Pour cela on choisit initialisation (Surface maximale ; image centre ; en bas ; en haut ; sur la gauche ; sur la droite)

- Surface maximale : le segment traqué est celui dont la surface est la plus grande par rapport aux autres segments trouvés.
- Centre de l'image : le segment traqué est celui qui est le plus proche du centre de l'image.
- En bas : le segment traqué est celui qui est le plus proche du bord inférieur de l'image.
- En haut : le segment traqué est celui qui est le plus proche du bord supérieur de l'image.
- A gauche : le segment traqué est celui qui est le plus proche du bord gauche de l'image.
- A droite : le segment traqué est celui qui est le plus proche du bord droit de l'image.

Il est possible de combiner 2 caractéristiques, par exemple, si on met un traqueur de segments réglé en bas puis un autre réglé à droite, l'ensemble traquera le segment le plus proche du coin en bas à droite.

On peut modifier aussi le paramètre Surface minimale [en pixel], ce paramètre peut varier de 0 à l'infini (si on dépasse la taille de l'image cela devient inutile), et il sert à ne pas prendre en compte les segments dont la surface est plus grande que la surface minimale spécifiée

7) Filtre (Filter) :

Entrées	**Type**	**Par défaut**	**Description**
Entrée	image		Image d'entrée
Sorties			
Sortie	image		Image filtrée

Ce bloc permet de filtrer une image. Pour cela on choisit le type de filtre (Maximum ou Minimum) et la taille de masque (allant de 3 à l'infini par pas de 2 en 2)