LES YEUX DU ROBOT

OLIVIER GAILLARD

SEPTEMBRE 2004

Table des Matières

Introduction **3**

1. Les espaces de couleurs **3**

1.1 Définition 4

1.2 RGB (Red, Green, Blue) 4

1.3 YUV 5

1.4 HSI (Hue, Saturation, Intensity) 5

2. Les réseaux de neurones **7**

2.1 Présentation 7

2.2 La compétition 9

3. La segmentation **9**

4. La formation des groupes **10**

5. La reconnaissance de forme **11**

6. La conversion des coordonnées **13**

7. La carte virtuelle **14**

Introduction

Ce programme de traitement et d'analyse d'images a été conçu pour le concours Eurobot organisé par Planète Sciences.

Le programme est capable de capturer une image à partir d'une webcam, traiter l'image, analyser les couleurs, analyser les formes, détecter les balles et les poteaux, donner la position d'un objet par rapport au robot, construire une carte virtuelle contenant la position de toutes les balles vues.

Nous allons voir en détails chaque partie de ce programme.

1. Les espaces de couleurs

1.1 Définition

Les espaces de couleurs sont des référentiels permettant de coder une couleur. Par exemple, le vert va être représenté dans le mode RGB par : R=0 G=255 B=0. Chaque espace de couleurs possède des spécifités particulières, ainsi, le choix de l'espace de couleur utilisé dépends de l'application que l'on veut effectuer.

Nous allons voir en détails trois des espaces de couleurs les plus connus : RGB, YUV, HSI.

1.2 RGB (Red, Green, Blue)

Le RGB est le mode que l'on connaît le mieux. Chaque composante représente une couleur de base : rouge, vert et bleu. La valeur de chaque composante est proportionnelle à la quantité des couleurs de base à mélanger pour obtenir la couleur finale.

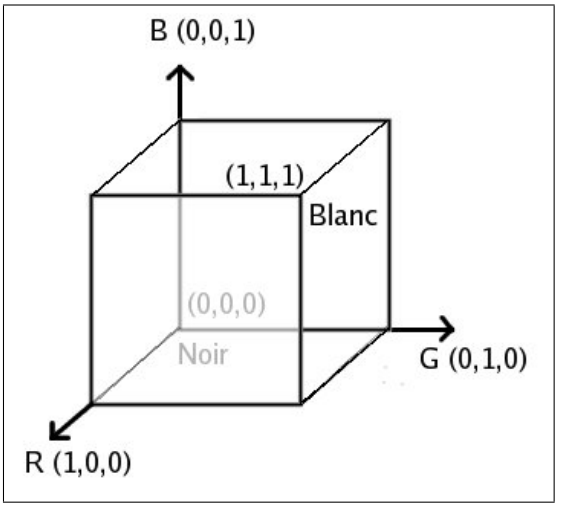

FIG. 1 – Cube RGB

L'avantage de ce mode de couleur est qu'il est très intuitif, ainsi, lorsque l'on change une des valeurs, on sait à peu près quel va être le résultat de ce changement sur la couleur

finale.

Par contre, l'inconvénient de ce mode est sa sensibilité à la lumière. En effet, en ajoutant de la lumière sur un objet, les trois composantes vont changer de valeurs.

1.3 YUV

Le YUV est un espace de couleur très utilisé, il est notamment utilisé pour le codage des couleurs de la télévision. On peut diviser cette espace en deux parties :

– la luminance (Y), qui représente la luminosité d'une couleur.
– la chrominance (UV), qui détermine la nature d'une couleur (marron, violet, ...).

Son principal avantage pour notre utilisation est qu'il est moins sensible aux variations de lumière. En effet, pratiquement seul la luminance est affecté par un changement de luminosité.

La conversion de RGB en YUV est très simple :

Y = 0,299.R + 0,587.G + 0,114.B
U = 0,147.R - 0,289.G + 0,437.B + 0,5
V = 0,615.R - 0,515.G - 0,100.B + 0,5

1.4 HSI (Hue, Saturation, Intensity)

Ce mode de couleur est très utilisé en robotique.

Son avantage est le même que le YUV, il n'est pas sensible aux variations de luminosité.

Par contre, la conversion à partir de données RGB demande plus de ressources processeur.

La conversion de RGB et HSI se fait de la manière suivante :

$$I = \frac{1}{3}(R + G + B) \tag{1}$$

$$S = 1 - \frac{3}{R + G + B}[min(R, G, B)] \tag{2}$$

$$H = cos^{-1}\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \tag{3}$$

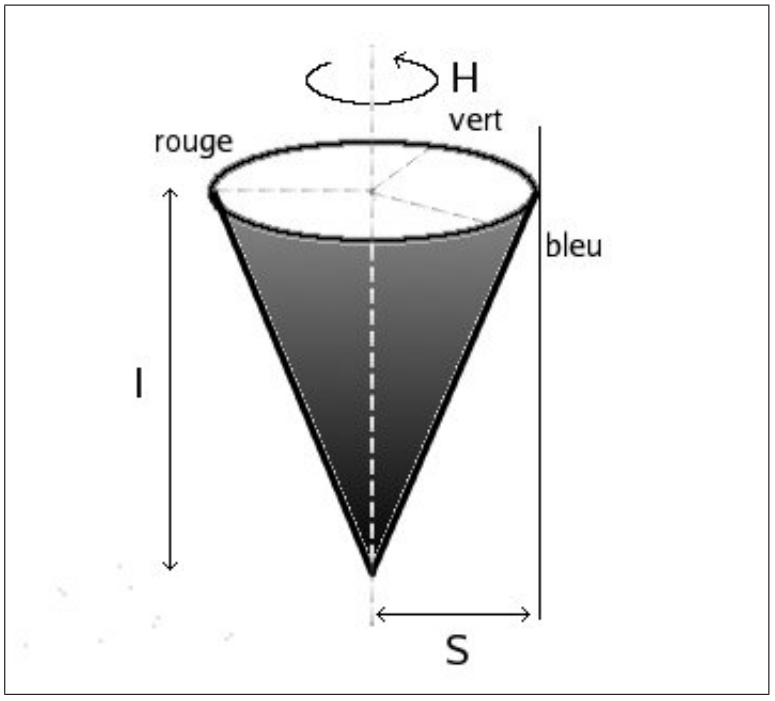

FIG. 2 – Cone HSI

2. Les réseaux de neurones

2.1 Présentation

Le réseau de neurones est l'un des plus grands outils de l'informatique. Il est utilisé dans des situations où l'algorithme nécessaire à la tâche demandée serait trop complexe à réaliser. Néanmois, le réseau de neurones n'est pas toujours facile à maîtriser. C'est une grande boîte noire dans laquelle des données sont rentrées et transformées par le réseau de neurones. Cette boîte noire doit cependant être calibrée pour la tâche à traiter.

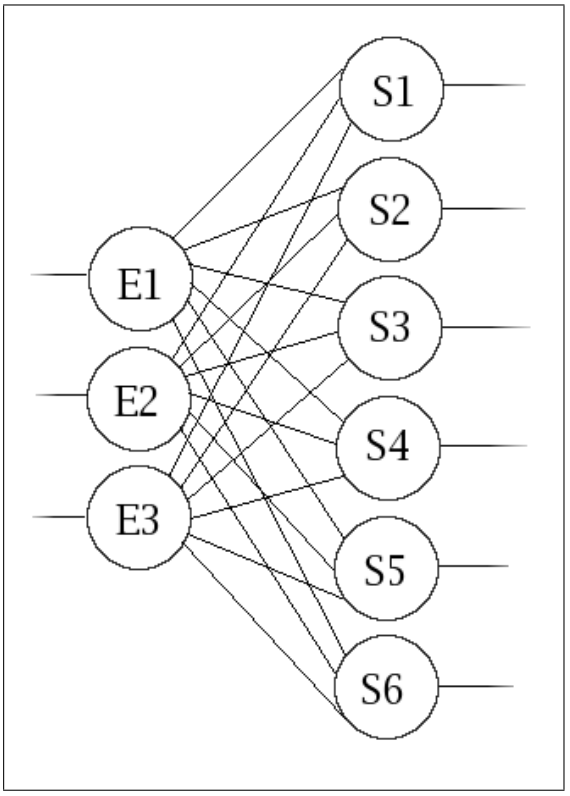

FIG. 3 – Exemple de réseau de neurones

Le réseau de neurones informatiques prennent exemple sur les neurones biologiques pour traiter les informations. Chaque neurone est relié aux autres par des connections pondérés. La sortie d'un neurone résulte de l'intégration de ces entrées. En général l'intégration utilisée est simplement la somme de ces entrées multiplié pour les poids associés à ces entrées.

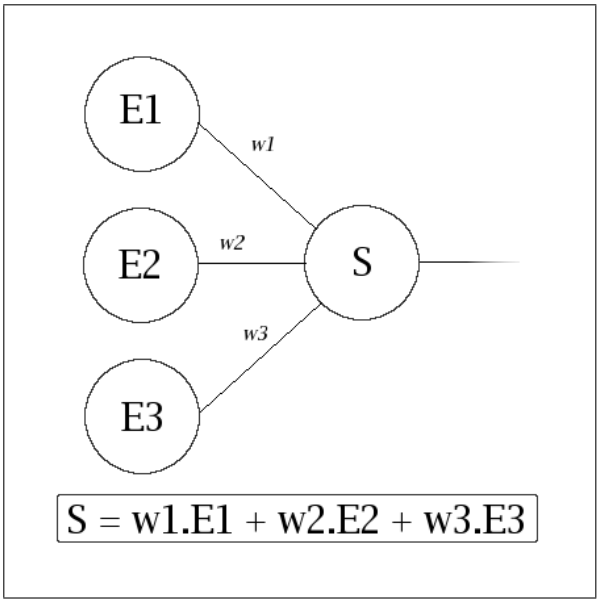

FIG. 4 – Détails d'une connection entre plusieurs noeuds

La calibration du réseau (ou apprentissage) va se faire en modifiant les connections entre les différents neurones et notamment leurs poids. La modification de ces poids se fait en utilisant des algorithmes de minimisation des erreurs. L'apprentissage se fait en appliquant un grand panel d'exemples en entrée du réseau pour que celui puisse s'adapter.

2.2 La compétition

Le réseau de neurones à compétition est l'un des plus simples mais il est très puissant. Il permet de différentier un set de données.

Pour la vision, nous allons l'utiliser pour différentier automatiquement les différentes couleurs présentes dans une image. Par exemple, on pourra lui demander de différentier dix couleurs différentes. Lors de l'apprentissage, le réseau va alors essayer de séparer les six couleurs les plus opposés possibles.

3. La segmentation

La segmentation est un processus qui permet de transformer l'image capturée par la caméra. Cette transformation a pour but de la rendre plus simple à traiter par la suite.

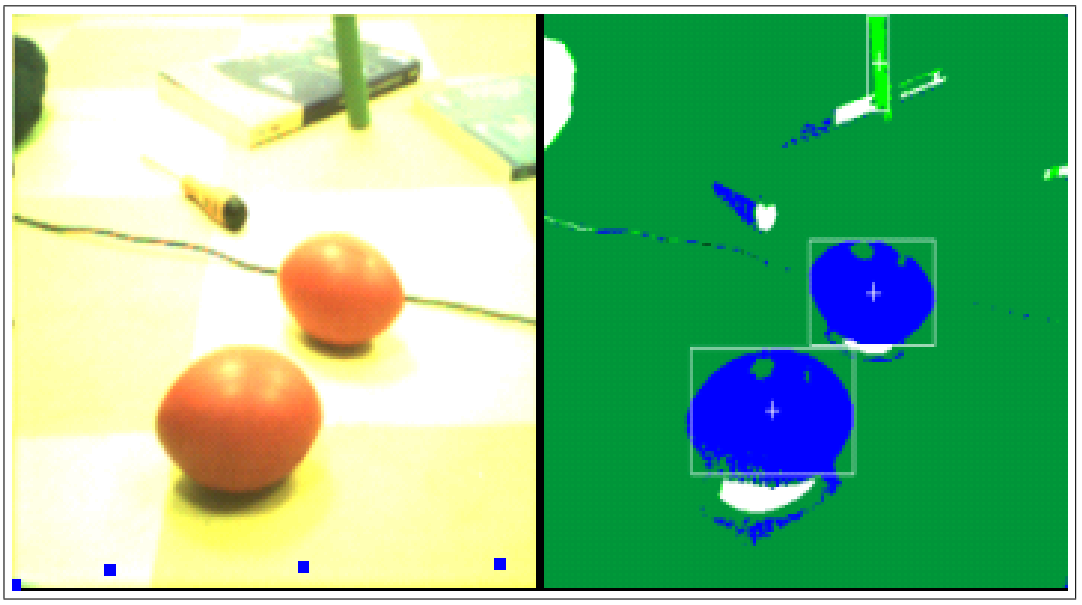

FIG. 5 – Exemple de segmentation

Dans notre cas, la segmentation va être une diminution du nombre de couleurs de l'image. On va ainsi passer de 256 couleurs à moins de 10 couleurs. Cette segmentation est très délicate puisqu'elle va produire l'image de base pour les traitements suivants.

C'est dans ce traitement qu'intervient le réseau de neurones. Il aura en entrée les trois composantes de chaque pixel et en sortie la numéro de la couleur segmentée. Le réseau devra être préalablement calibré. Cette opération est en grande partie automatique mais nécessite quelques réglages manuels pour une meilleure efficacité.

On aura donc en sortie de cette segmentation une image contenant une dizaines de couleurs partagées par le réseau de neurones.

4. La formation des groupes

Pour trouver un object dans l'image, on la parcourt en effectuant des tests de couleur sur une partie des pixels de l'image. On va ainsi tester, pour commencer, seulement 1% de l'image, en faisant des tests uniquement tous les 10 pixels. En effet, si un object est situé sur l'image, on considère que celui-ci à une largeur supérieur à 10 pixels, ainsi, même en effectuant ces sauts, il sera trouvé.

Une fois qu'un pixel de la couleur cherchée est trouvé, quelques tests supplémentaires vont être effectués pour avoir une meilleure précision de la position et de la taille de l'object. Pour délimiter l'object, on utilise l'algorithme suivant :
- On prends comme point de départ le pixel trouvé.
- On monte tant que la couleur est la même.
- On descend en faisant le même.
- Puis on teste la gauche et la droite.

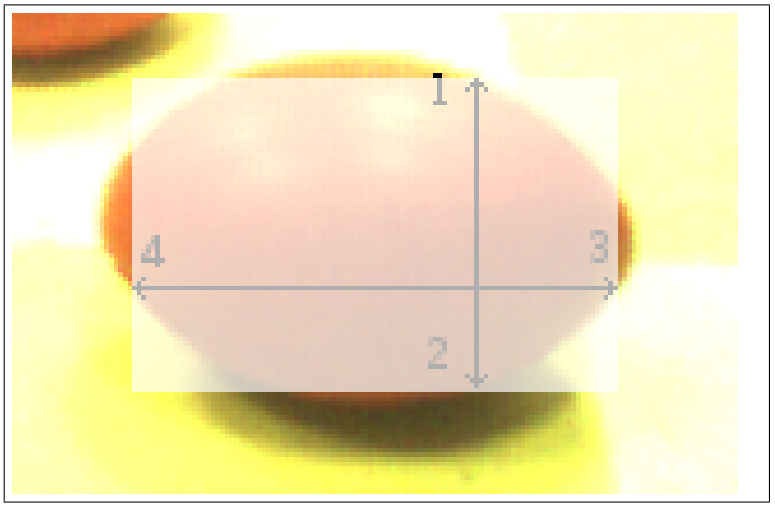

FIG. 6 – Exemple de l'algorithme sur une balle

On a alors les coordonnées d'une boîte qui entoure l'object. Mais un problème subsiste, si l'object est plus grand que 10 pixels, il peut être retrouvé plusieurs fois. On va alors utilisé le centre de l'object que l'on peut calculé grâce aux coordonnées obtenues. Si le centre est le même à une certaine erreur près, on considère que c'est le même object.

5. La reconnaissance de forme

La reconnaissance de forme utilise la représentation ıPhi-s code. Cette représentation stocke les distances entre le centre d'un object et les points de son contour. On obtient alors une fonction des distances en fonction de l'orientation avec laquelle on calcule la distance.

Pour minimiser le bruit, on fusionne des valeurs consécutifs de notre courbe et on calcule leur moyenne.

De plus, l'object recherché pouvant avoir plusieurs tailles, fonction de la distance auquel la webcam prends l'image, on normalise notre courbe. Pour la normalisation, on divise par la plus grande distance trouvée.

Un autre problème est l'orientation de l'object, en effet celui-ci peut être vu avec des angles de vue différents. On va déphaser notre courbe pour la faire commencer à l'index qui à la plus grande distance.

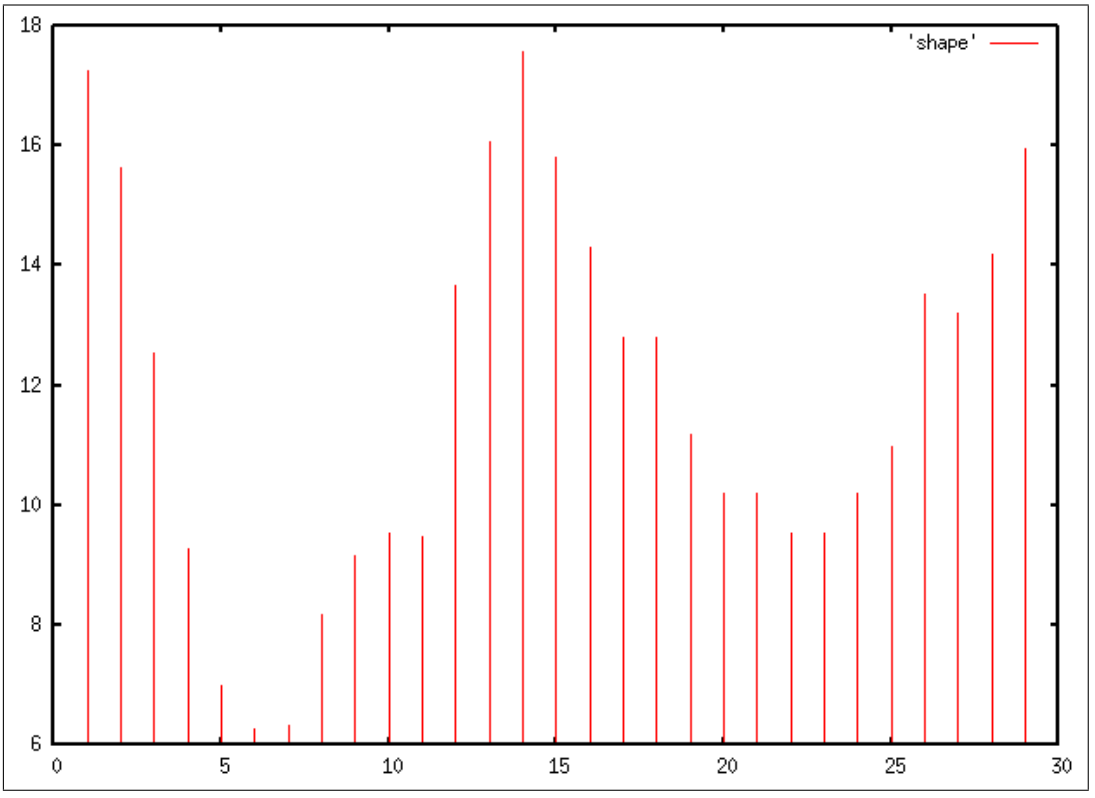

FIG. 7 – Graphe des distances centre-contour en fonction de l'angle

Il ne reste alors plus qu'à comparer une courbe de référence, obtenue lors d'un calibrage, et la courbe des objects trouvés. Si le décalage entre les deux courbes est en dessous d'un certain seuil, on admettra alors que les objects sont identiques.

6. La conversion des coordonnées

Nous disposons maintenant de l'emplacement sur l'image des objects trouvées. Mais, ce qui nous intéresse est leur position réelle par rapport au robot. Cette position réelle dépend évidemment de l'orientation et de la position de la caméra. Il va donc falloir faire une calibration pour trouver une correspondance.

Cette conversion contient deux composantes qui vont avoir chacun un traitement différent, étant donné que la nature de leur variation est différente. En effet, pour les ordonnées, on peut trouver une équation du second degré en faisant une régression par rapport à des points dont on aura préalablement enregistré la correspondance. On peut utilisé Gnuplot pour réaliser cette regression.

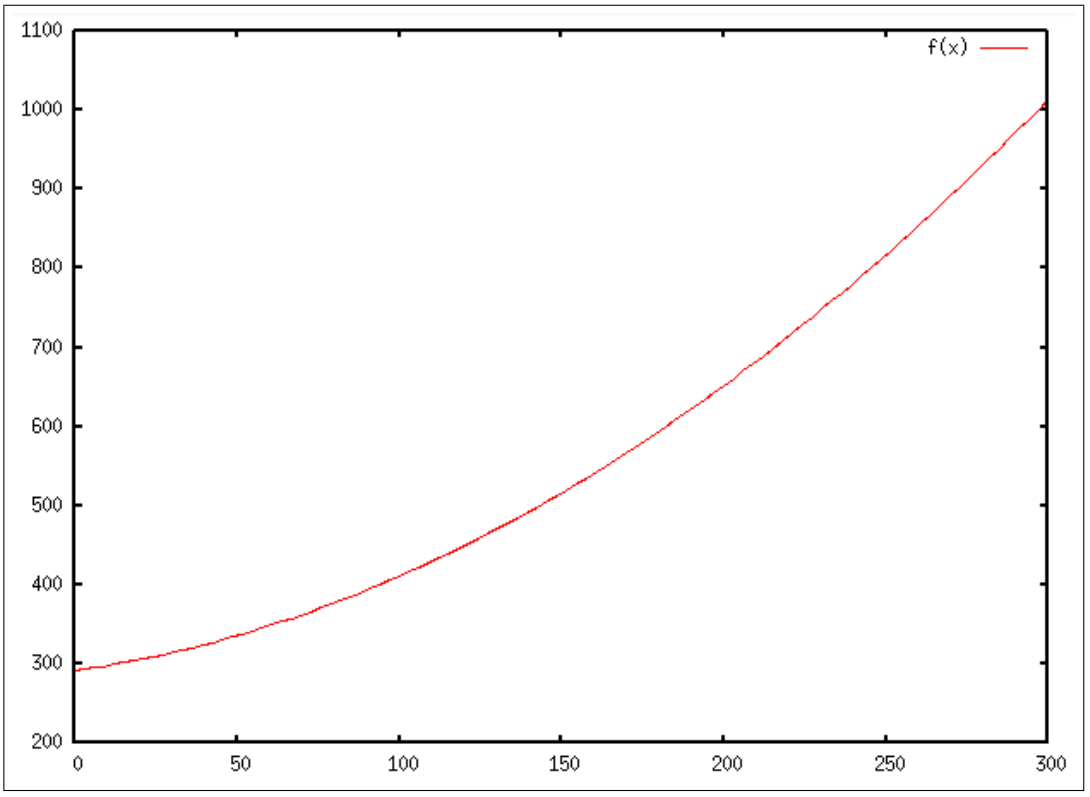

FIG. 8 – Courbe de la distance d'un objet en fonction la valeur du pixel

Pour les abscisses, c'est encore plus simple, il s'agit d'une équation du premier degré qui est en fonction de l'ordonné. Il faut donc trouvé pour chaque points de l'ordonnée les coefficients de cette équation.

7. La carte virtuelle

La carte virtuelle permet de mémoriser la position des objects trouvés et de faire des recoupements entre les diverses informations obtenues par la caméra. Le rôle de cette carte est d'analyser toutes les informations pour donner le plus précisément possible la localisation des objects, et notamment, celle de l'object le plus proche. Cette carte calcule aussi la probrabilité que l'object soit encore à la même position en fonction des évènements qui ont lieu.

Un système de score permet aussi de choisir un object en fonction de différentes options, comme l'orientation de l'object par rapport au robot, la probabilité de sa présence à l'endroit mémorisé, ...

Cette partie permet une interface facile avec un autre programme car elle donne des informations simples et fiables qui ne demandent aucun autre traitement.