TIPE

Partie 2 : Recherche par forme

Ma première approche dans la détection d'une balle sera de considérer celle-ci comme un objet sphérique, donc circulaire sur une image, ce qui donne une géométrie assez simple. Le point essentiel de la détection, et mon objectif, est de déterminer les contours des objets présents dans l'image, puis de déterminer quels contours de l'image sont circulaires. L'algorithme présenté est une version très simplifiée d'un algorithme connu dans la littérature sous le nom de transformée de Hough (*Hough transform*).

1. Détection de contours

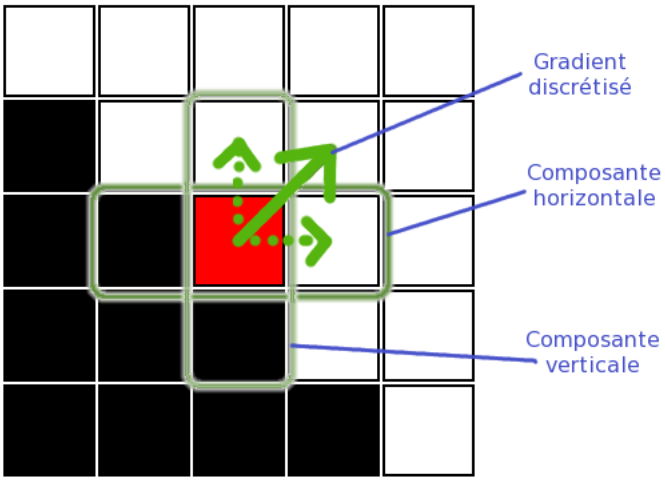
En premier lieu, qu'est-ce qu'un contour dans une image? Mathématiquement, c'est une zone de l'espace où les propriétés (luminosité, couleur, ...) de l'image changent beaucoup, comme on le voit sur l'image, où les contours sont représentés en rouge. Il s'agit dont une zone de **fort gradient pour une fonction (luminosité, couleur, ...) donnée**, qui délimite bien ce qui m'intéresse dans l'image, à savoir la position de la balle. Notons I la fonction (ici, l'intensité) qui représente l'image. Il nous faut donc calculer le gradient sur une image, qui se définit pour un espace "continu" comme suit :

$$\overrightarrow{\text{grad}}\, I(x,y) = \begin{cases} \frac{\mathrm{d}I}{\mathrm{d}x} \\ \\ \frac{\mathrm{d}I}{\mathrm{d}y} \end{cases}$$

Pour une image discrète, il ne reste plus qu'à l'adapter. On a $\mathrm{d}x = \mathrm{d}y = 1$: en effet, l'information importante dans la détection des contours est uniquement la valeur du gradient en un point comparée aux autres valeurs du gradient dans l'image, et on peut donc définir notre gradient à une constante multiplicative près. Quant à la valeur de $\mathrm{d}I$, la manière la plus simple de la calculer est la différence entre les valeurs de I aux deux points adjacents horizontalement puis verticalement.

On obtient donc une forme de **gradient discrétisé** :

$$\overrightarrow{\text{grad}_d} I(i,j) = \begin{cases} I(i+1,j) - I(i-1,j) \\ I(i,j+1) - I(i,j-1) \end{cases}$$



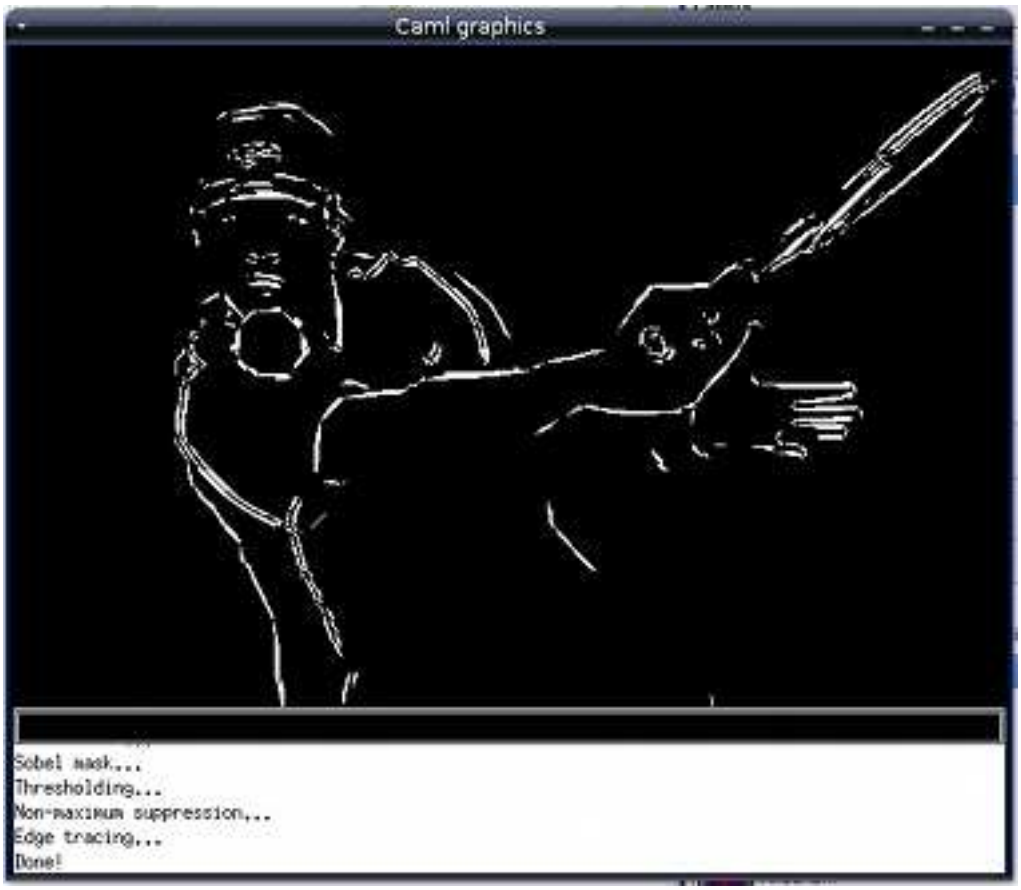
Exemple de calcul de gradient discrétisé

Dans la pratique, on pourra utiliser des formes de gradients discrétisés légèrement plus évoluées, mais le principe de base reste le même.

Il est alors possible de définir **un contour comme un point où le gradient est grand en norme**. Plus exactement, avec g_0 donné, on définira un contour par :

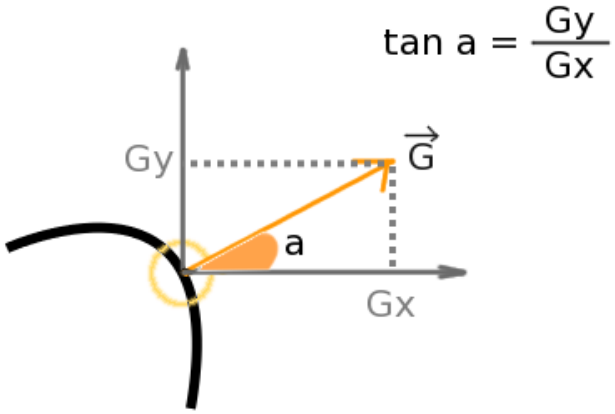
P est un contour $\Leftrightarrow \|\overrightarrow{\text{grad}_d} I(P)\| \geqslant g_0$

Cela m'a permis d'obtenir des cartes de contours d'une image, qui pour chaque point indiquent s'il est un contour ou non.



Pour savoir plus facilement si les contours détectés sont circulaires, on aura besoin de connaître la direction de ceux-ci, donc la direction, par rapport à l'axe horizontal $\vec{e_x}$, du vecteur gradient. On a aisément, à l'aide d'un peu de trigonométrie et des composantes du gradient dans la base orthonormée canonique :

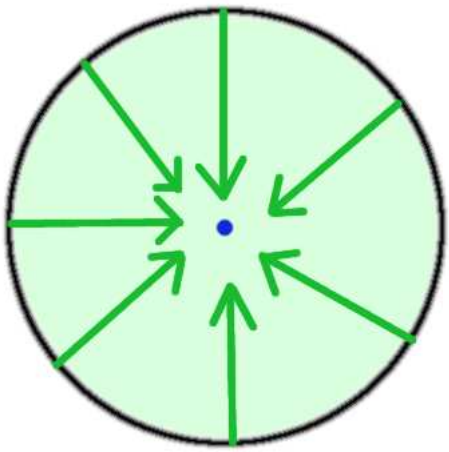
$$(\vec{e_x}, \overrightarrow{\text{grad}_d}I(P)) = \arctan \frac{\overrightarrow{\text{grad}_d I(P)}.\vec{e_y}}{\overrightarrow{\text{grad}_d I(P)}.\vec{e_x}}$$



De plus, comme on le voit sur l'exemple, **le vecteur gradient est toujours normal à la tangente** au contour, on a donc l'angle entre l'axe horizontal et la normale.

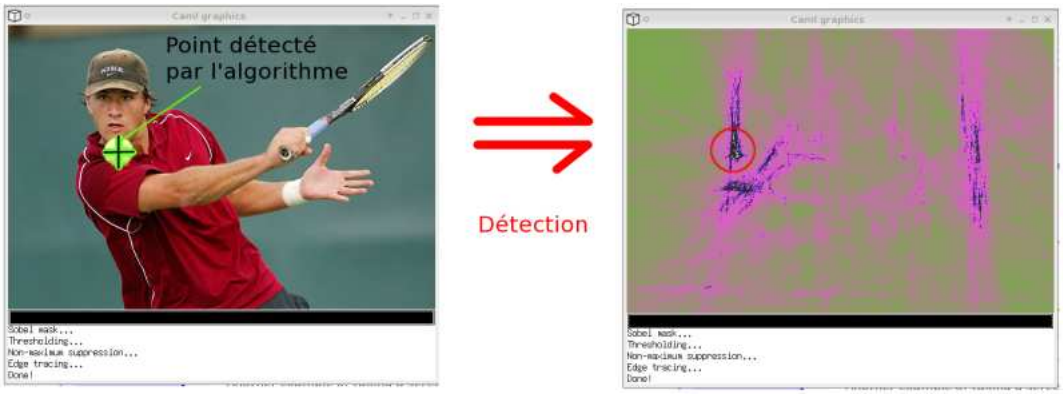
2. Détection de formes circulaires

A présent, j'ai pu pu commencer à déterminer où était **la forme circulaire la plus visible** dans une image. Cette méthode pourrait être étendue à la détection de plusieurs formes circulaires, mais le principe en reste le même. Tout d'abord, qu'est-ce qu'un cercle? Plus exactement, qu'est-ce que les points qui sont *sur* le cercle ont en commun, en terme de gradient? La réponse est simple : la normale aux contour circulaire, donc **le gradient calculé au niveau des points du contour circulaire sont dirigés vers le centre du cercle**. On conviendra alors qu'un point susceptible d'être le centre d'un cercle est un point où de nombreuses droites, perpendiculaires à la direction des contours, s'intersectent.



Contour circulaire

Algorithmiquement parlant, pour détecter ces points d'intersection, il est possible de tracer les droites issues de tous les points de l'image dirigées par le gradient en ce point, avec cette condition particulière : le tracé se fera sur un tableau (aussi appelé "accumulateur") où chaque point par lequel passe une droite verra sa valeur augmentée de 1. Le point qui aura le plus grand nombre de droites passant par lui, donc dont la valeur dans l'accumulateur sera la plus grande, sera le centre du cercle le plus apparent de l'image (ici, les couleurs les plus sombres sont celles où l'intensité est maximale).



3. Performances et limites

Des tests sur des images en situation réelle m'ont permis de constater que l'algorithme échoue souvent à cause de deux problèmes :

- Le premier problème observé est un problème de dimension. Sur un terrain en terre battue, de couleur radicalement différente de la balle, la détection des contours se fait très bien, et on obtient dans la plupart des cas la position de la balle. Le problème est alors, dans ces conditions, souvent la **taille de la balle** : une balle trop petite, comme c'est souvent le cas quand des photos sont prises d'une longue distance sur des terrains, échappe presque systématiquement à la détection et n'est même plus reconnaissable quand on affiche les contours détectés, même s'il elle est très visible sur le fond de l'image.

- Le second problème observé est d'une autre nature : **dans une image où il y a beaucoup de bruit** (par exemple, spectateurs dans les tribunes, balle située sur un fond peu uniforme, etc...), les contours sont imprécis et la balle n'est souvent pas détectée. Les couleurs du décor n'étant souvent pas proches de celles de la balle, on peut penser à effectuer un filtrage des couleurs de l'image.

- Enfin, si la **vitesse de la caméra** qui enregistre les images en entrée de l'algorithme est trop grande, la balle, même si elle conserve une couleur plutôt jaune, voit sa forme étirée ; si elle n'est plus suffisamment circulaire, l'algorithme ne la détectera pas.

Au niveau des performances, sans rentrer dans les détails, l'algorithme s'éxécute en quelques secondes pour une image de 800x600 pixels.

Il est à noter qu'une version nettement améliorée a été utilisée avec succès lors de la RoboCup, un tournoi de football entre robots, mais les causes d'échec de l'algorithme précédent ont été partiellement résolues en utilisant des balles relativement grosses par rapport aux balles de tennis et un environnement lumineux très contrôlé pour ne pas qu'il y ait trop de bruit visuel : ici, l'algorithme était très efficace. Dans le cadre de la trajectoire d'une balle de tennis, sur l'image produite par ma caméra, la balle est parfois si petite et tellement déformée par la vitesse qu'il est à la fois illusoire et désastreux en terme de performances de vouloir détecter sa position à l'aide de l'algorithme précédent.

C'est pourquoi j'ai pensé qu'il valait mieux se baser sur la couleur de la balle pour espérer pouvoir la détecter.