DÉTECTION DES CONTOURS D'UNE IMAGE : FILTRE DE CANNY

Rapport de mini-projet

Année universitaire 2015-2016
ENSSAT, IMR2

NANTEL Maëlig

Introduction

Dans le cadre de la formation Informatique, Multimédia et Réseaux dispensé à l'ENSSAT, nous avons étudié des méthodes de traitement d'images. Il s'agissait essentiellement de comprendre la représentation informatique ainsi que les principaux modèles mathématiques permettant de traiter des images existantes. Durant la première partie de ce cours nous nous sommes intéressés à la problématique de détection des contours dans image. *Qu'est-ce qu'un contour? Comment les détecter de manière efficace?* Pour répondre à ces questions, nous avons implémenté l'algorithme du filtre de Canny dans un mini-projet encadré.

Ce rapport explique les différentes étapes du filtre de Canny et présente les démarches menées durant son implémentation sur le logiciel Scilab. L'image utilisée pour tester les différentes étapes est celle du célèbre nageur Michael Phelps lors de sa dernière médaille d'or aux Jeux Olympiques de 2012 à Londres.

Image initiale en 634*446 (soit 282 764 pixels)
(Source : news.nationalpost.com)

1) Notions générales

Durant ce mini-projet, on ne considère que des images en niveaux de gris. Une image est un ensemble de pixels dans un support à deux dimensions, on la modélise en mathématique et en informatique par une matrice où chacun des éléments est un entier pouvant varier de 0 (noir) à 255 (blanc). On peut alors considérer une image comme étant un ensemble de régions de niveaux de gris homogène. Un contour est de ce fait une ligne de séparaison entre deux régions. Cette ligne représente une variation significative des niveaux de gris. Compte tenu de ces définitions, il apparaît alors logique d'analyser chaque pixel de l'image pour permettre d'en détecter les contours.

Plusieurs algorithmes de détection de contours, dont l'algorithme de filtre de Canny, se basent sur l'étude du gradient pour chaque pixel de l'image. Un gradient est une grandeur vectorielle permettant de représenter l'évolution d'une valeur physique. Dans notre cas le gradient va modéliser la variation de la luminance dans l'image.

2) Algorithme du filtre de Canny

L'algorithme du filtre de Canny peut être décomposé en 3 étapes distinctes :

- Rehaussement de Canny
- Suppression des non-maximums
- Seuillage par hystérésis

2.1) Rehaussement de Canny

A) Réduction du bruit

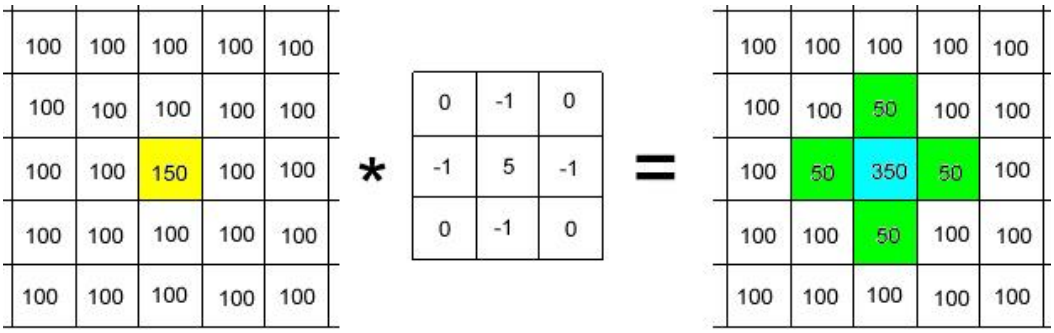
Nous avons vu dans 1) qu'un contour représentait une séparaison entre deux zones de niveaux de gris homogène dans une image. En réalité, peu d'images type « photographie » présente naturellement des zones de niveaux de gris homogène. En effet, lorsqu'on regarde une image, on apprécie souvent sa netteté et la présence de nombreux détails. En traitement d'image, ces détails trop nombreux se comportent comme du bruit sur l'image puisque certains pixels peuvent potentiellement causer de fortes variations d'intensité lors du calcul du gradient. Une conséquence serait la détection de faux positifs dans les contours.

Pour réduire ce problème, il est donc nécessaire de réduire ce « bruit ». Pour cela, l'algorithme de Canny se base sur l'application d'un filtre Gaussien (filtre moyenneur) sur l'image. C'est un filtrage spatial, ce qui signifie qu'on remplace la valeur d'un pixel par une fonction des valeurs des pixels voisins. Il est important de bien choisir la taille du filtre à appliquer ; en effet, plus le filtre utilisé sera grand, moins il sera sensible au bruit. Pour l'image choisie, un filtre symétrique 5x5 convient bien.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Masque utilisé (Source : Wikipédia)

Nous avons vu en cours que l'application d'un filtre sur une image revenait à calculer un produit de convolution entre la matrice image et un masque représentant le filtre. Appliquer un filtre sur une image revient donc à multiplier chacun des pixels de la matrice image par le filtre. Pour calculer la valeur d'un pixel x, y de l'image après l'application du filtre, on multiplie sa valeur initiale par celle du pixel central du filtre et on additionne ensuite la valeur des produits des pixels adjacents.



(Source : algojava.com/produit-de-convolution-des-matrices)

Dans l'exemple ci-dessus, on applique un produit de convolution sur le pixel central on obtient :

$$(5 * 150) + (-1 * 100) + (-1 * 100) + (-1 * 100) + (-1 * 100) + (0 * 100) + (0 * 100) + (0 * 100) + (0 * 100)$$

Afin de ne pas modifier la moyenne de l'image, il est nécessaire de normaliser la somme des éléments du filtre à 1. Ce n'est pas réellement une étape du produit de convolution, mais cela permet de conserver la dynamique de l'image.

Avec la représentation ci-dessus, on comprend bien que pour calculer la nouvelle valeur des pixels bordures de l'image, on va devoir « se positionner » en dehors de l'image. Dans l'implémentation de l'algorithme, il est donc nécessaire d'agrandir le cadre de l'image pour pouvoir effectuer les calculs.

Image après application du filtre Gaussien 5x5

B) Calcul des gradients

À partir de cette image, il est désormais possible de calculer les gradients pour chaque pixel de l'image. Le calcul des gradients sur une direction (x ou y) revient à appliquer un filtre de convolution.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad ; \quad G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Filtres de convolution pour les calculs de gradients (Source : Wikipédia)

Pour chaque gradient en x J_x et en y J_y on calcule sa norme et sa direction dans l'image. La direction du gradient correspond à l'angle (en radian) de sa normale.

$$norme = \sqrt{Jx(i,j)^2 + Jy(i,j)^2}$$

$$direction = arctan(\frac{-Jy}{Jx})$$

Il faut bien comprendre que la direction du gradient nous permettra d'obtenir la direction du contour que nous allons détecter. On comprend alors qu'il n'y aura que 4 directions possibles (rappelons que l'image est représentée dans un support à deux dimensions x et y) :

- Horizontal (0 °)
- Diagonale droite (45 °)
- Vertical (90 °)
- Diagonale gauche (135 °)

À partir de la direction précédemment calculée, nous devons donc approximer la direction du contour. La répartition est la suivante (en comparant avec un cercle trigonométrique) :

135 °	90 °	45 °
0 °		0 °
45 °	90 °	135 °

2.2) Suppression des non-maximums

À ce stade nous avons donc accès à deux matrices supplémentaires nous donnant la valeur de l'intensité des gradients (la norme) et sa direction dans l'image (angle de sa tangente). Nous avons vu précédemment qu'un contour représentant une forte variation dans les niveaux de gris, il sera donc caractérisé par une norme de gradient importante. Cependant, la norme seule ne suffit pas à déterminer si un pixel fait partie ou non d'un contour. Dans une image classique, un contour est rarement « brut », il est progressif. C'est encore plus vrai maintenant que nous avons appliqué un filtre Gaussien sur l'image. Avec la matrice actuelle, nous allons obtenir des gradients aux normes importantes sur toute la zone de changement entre deux zones, et donc un contour épais.

Norme des gradients avant la suppression des non-maximums

La suppression des non-maximums consiste donc à garder uniquement les maximums locaux. On garde uniquement les points qui dont la norme du gradient n'est pas plus petite qu'au moins un de ses deux voisins le long du gradient. Pour savoir avec quels pixels il faut effectuer la comparaison, il faut se référer au tableau précédent, en fonction de l'angle de la normale :

- 0 ° : comparer aux pixels de droite et gauche
- 90 ° : comparer aux pixels du dessus et du dessous
- 45 ° : comparer aux pixels en haut à droite et en bas à gauche
- 135 ° : comparer aux pixels en haut à gauche et en bas à droite

Après la suppression des non-maximums

2.3) Seuillage par hystérésis

À la fin de la seconde étape, nous avons déjà une vision assez claire de l'emplacement des contours dans l'image. Cependant, il est toujours possible d'avoir détecté des faux positifs liés aux bruits ou variations très locales dans l'image. La dernière étape de l'algorithme de Canny, le seuillage par hystérésis va permettre de décider pour chaque point du contour s'il faut le conserver ou non. Pour cela, on utilise deux seuils de décisions (un seuil bas et un seuil haut) et on les compare à la norme du gradient des pixels détectés comme contours :

- Si la norme est inférieure au seuil bas, on retire le pixel du contour ;
- Si la norme est supérieure au seuil haut, on conserve le contour ;

- Si la norme se trouve entre les deux seuils, on ne conserve le contour que s'il est connecté à d'autres points déjà conservés dans le contour. Un contour est rarement isolé dans une image, puisqu'il permet de marquer la différenciation entre deux zones.

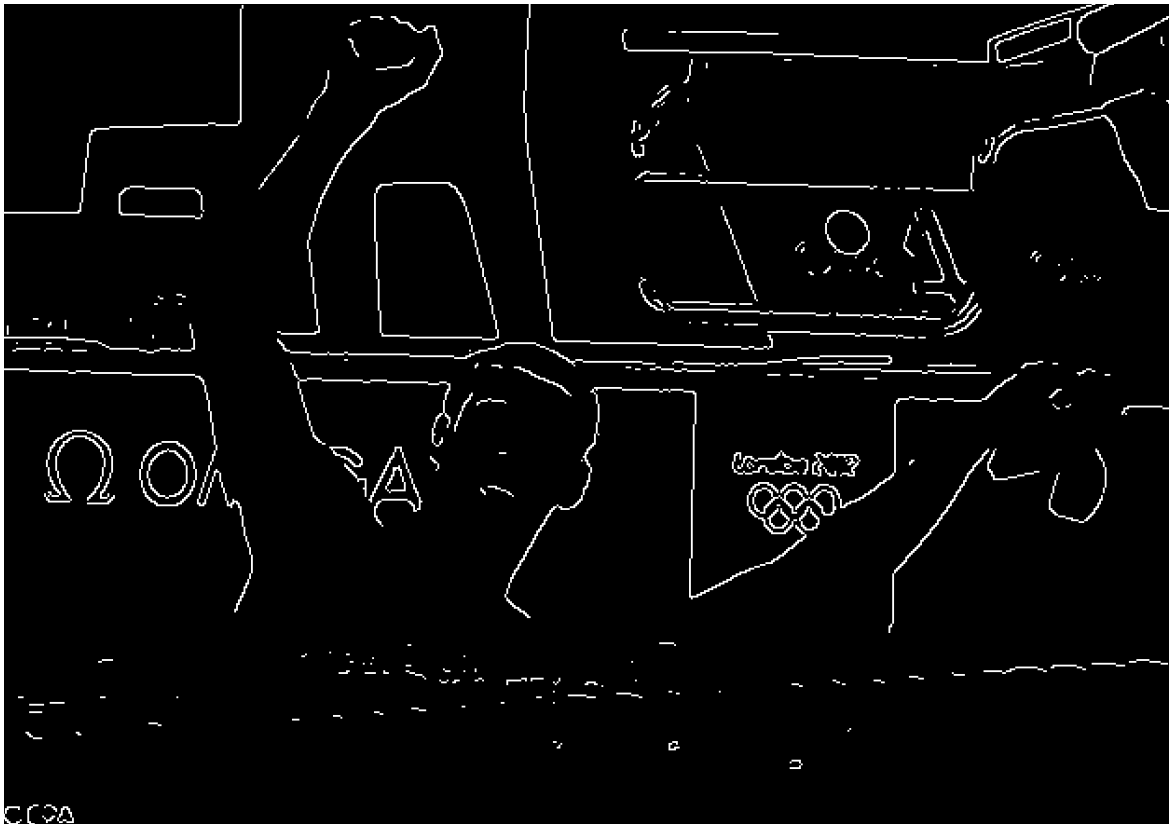
Concernant l'implémentation, il est nécessaire de diviser cette étape en deux. On commence par comparer uniquement aux seuils haut ou seuil bas afin de conserver ou rejeter les pixels. Ensuite, nous devons étudier le cas des pixels se trouvant entre les deux seuils. Étant donné que nous souhaitons les conserver que s'ils sont connectés à un point

déjà accepté comme contour, il est nécessaire de faire la comparaison sur la matrice résultante des premiers tests afin d'être sûr d'avoir déjà conservé ou rejeté les pixels adjacents à celui que l'on analyse.

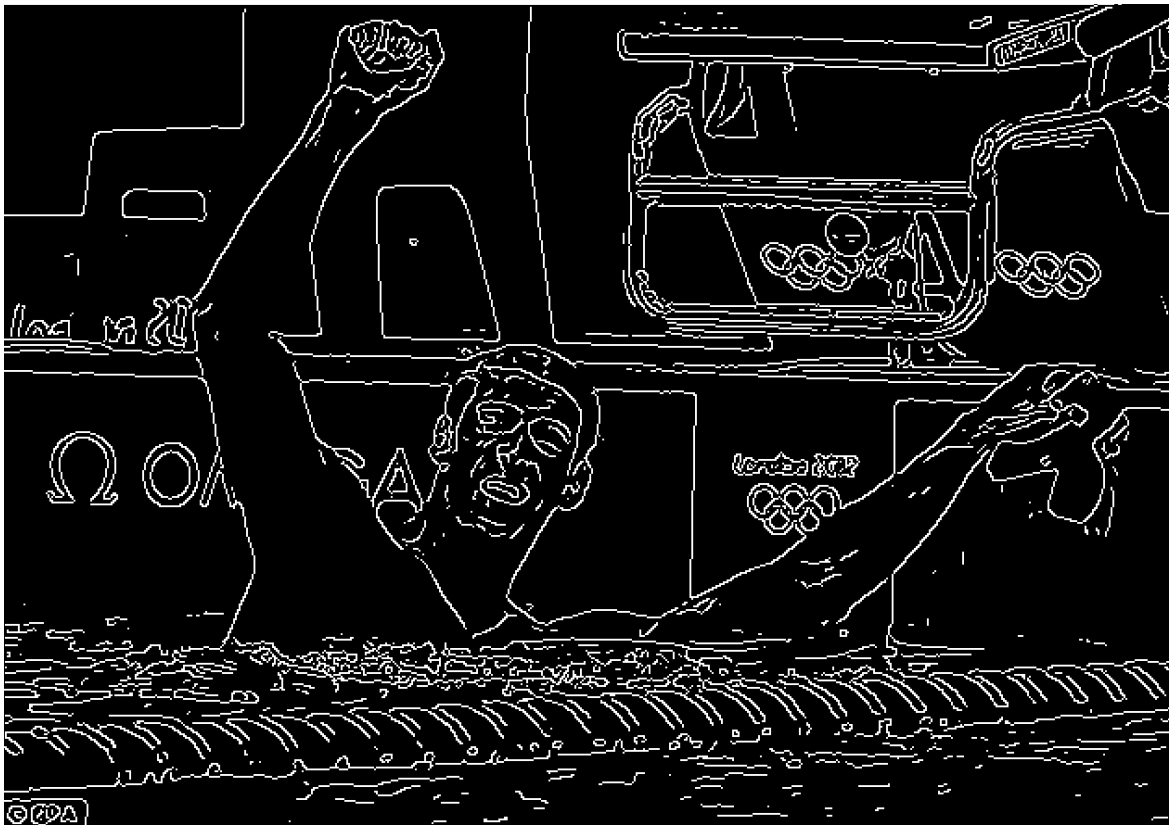
À la fin de l'algorithme, nous souhaitons uniquement obtenir les contours en blanc sur un fond noir. Lors de l'étape de seuillage, on remplace alors la valeur de l'intensité du gradient par 255 (blanc) si le pixel est considéré comme contours et tout le reste de la matrice est mis à 0 (noir).

Résultats du seuillage

Le résultat obtenu après l'étape de seuillage par hystérésis est le résultat final de l'algorithme de Canny. Ce résultat est donc totalement dépendant des seuils choisis pour cette dernière étape.



Seuillage par hystérésis : rendu final avec seuil haut = 50 seuil bas = 10



Seuillage par hystérésis : rendu final avec seuil haut = 15 seuil bas = 10

Cette comparaison permet de conclure sur l'effet du seuil haut. On remarque qu'un seuil haut trop haut empêche de détecter certains contours peu marqués, mais ayant pourtant du sens dans l'image (moins de détails sur le visage par exemple). En revanche, un seuil haut trop bas va provoquer une détection trop importante liée au bruit (ici une trop forte détection des mouvements de l'eau).

Avec l'implémentation réalisée, je n'observe pas de différences lors de la modification du seuil bas. Je n'arrive cependant pas à identifier d'erreurs dans la mise en œuvre de l'algorithme.

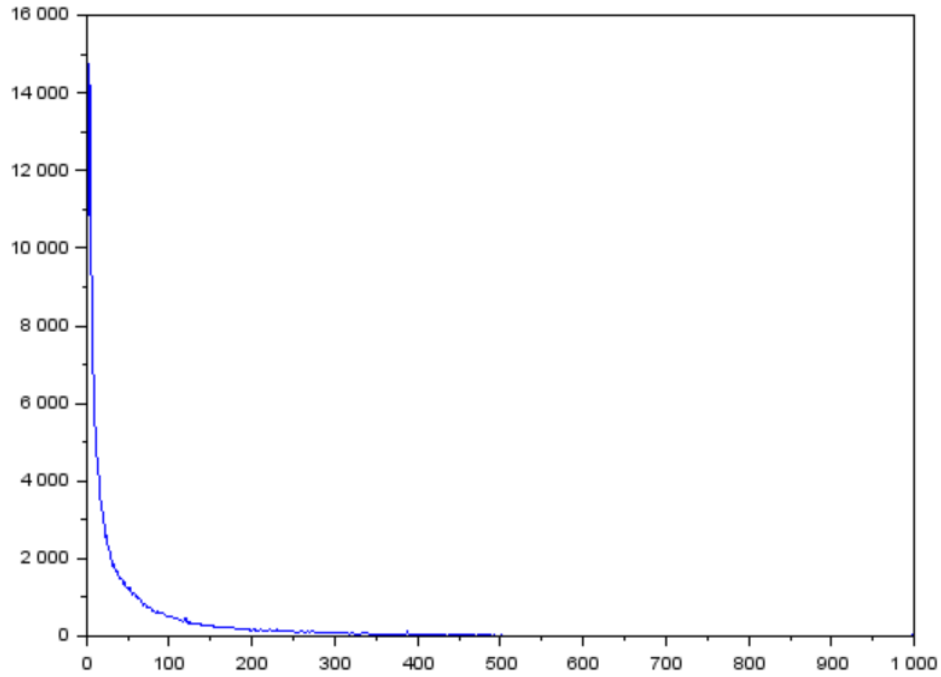
Le principal problème de l'algorithme de Canny est la détermination des seuils. Il n'existe pas de seuils produisant de bons résultats sur toutes les images. Il faut les déterminer avec de nombreux essais pour chaque image passée à l'algorithme. C'est plus généralement un problème récurrent dans tous les algorithmes utilisant des seuils.

3) Version « semi-automatique » de l'algorithme

Nous avons terminé la mise en œuvre de l'algorithme du filtre de Canny sur Scilab. Cependant, nous avons mis en avant le problème de détermination des seuils pour réaliser l'étape de seuillage par hystérésis. Il existe une solution permettant de n'être dépendant que d'un seul seuil et non de deux comme c'était le cas jusqu'à présent.

Pour cela, on fixe $seuil\ bas = \frac{seuil\ haut}{2}$ et on détermine le seuil haut comme étant la valeur d'un certain centile de la norme des gradients. Le seuil dont cette méthode est dépendante est donc en réalité la valeur du centile désiré.

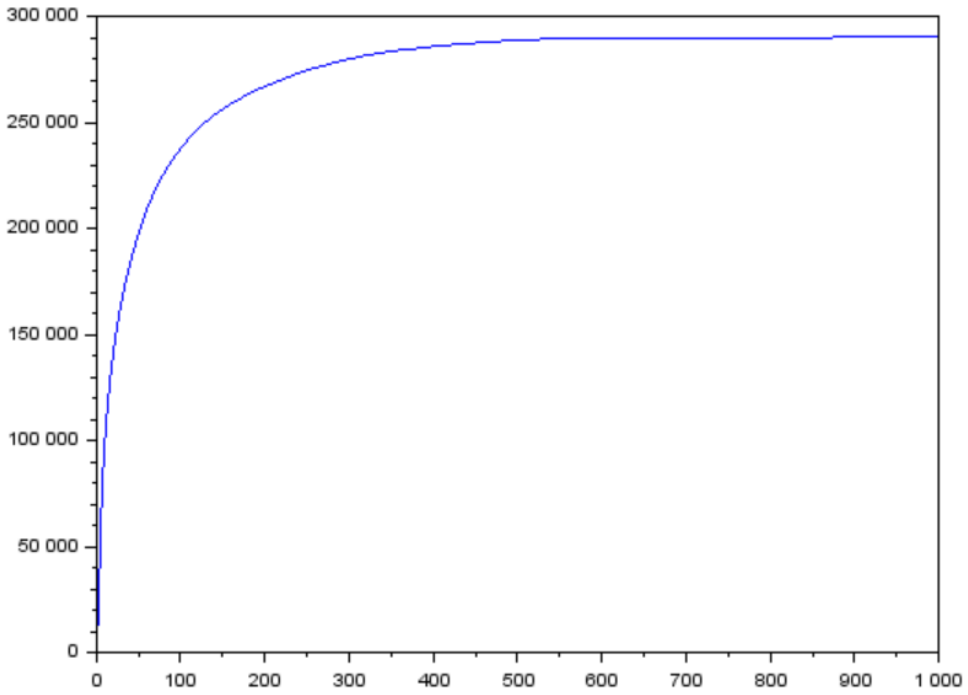
Pour pouvoir calculer le centile, il est nécessaire de répartir la norme des gradients dans un histogramme représentant la répartition des pixels par intensité.



Histogramme (norme des gradients) (précision de 1 000)

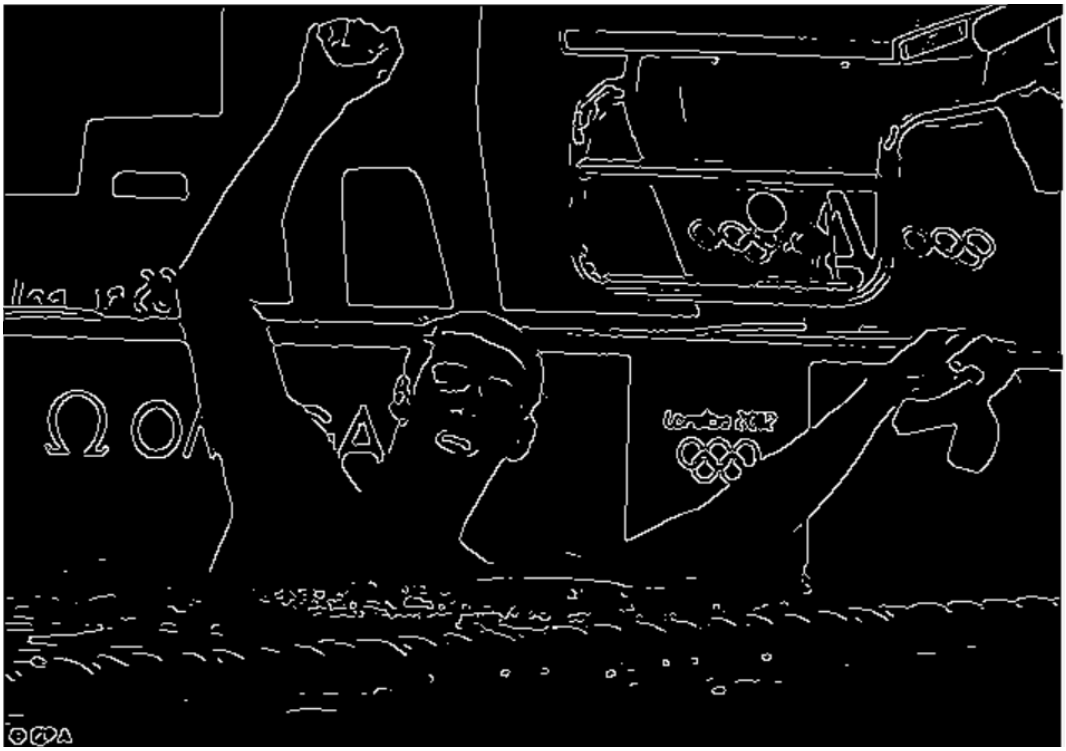
Affichage sous forme d'une courbe pour des raisons pratiques avec Scilab.

À partir de cette représentation des valeurs des gradients, on peut calculer la fonction de répartition et ainsi déterminer la valeur du centile souhaité.



Fonction de répartition (précision de l'histogramme à 1 000)

Rendu final avec seuils calculés automatiquement et le centile 75 seuil_haut calculé = 14



Rendu final avec seuils calculés automatiquement et le centile 90 seuil_haut calculé = 32

On peut donc faire les mêmes conclusions que lors de la détermination manuelle des seuils. Plus le centile est faible, plus l'algorithme va détecter de contours puisqu'il sera moins sélectif sur la norme des gradients à considérer comme contours.

Il est possible de vérifier la valeur calculée du centile (donc du seuil haut) à l'aide d'une fonction déjà existante dans Scilab :

- Pour le centile 90 et précision de 10000 pour l'histogramme
  - Valeur Scilab = 16,1166
  - Valeur calculée = 15,6242

- Pour le centile 75 et précision de 1000 pour l'histogramme
  - Valeur Sscilab = 16,116675
  - Valeur calculée = 14,1337

Les résultats obtenus pour le calcul du seuil haut sont donc satisfaisants.

Conclusion

Ce mini projet nous permit de bien comprendre la problématique de la détection de contours en traitement d'images. En mettant en œuvre l'algorithme de Canny, nous avons pu apprendre à manipuler et à interpréter des modélisations physiques et mathématiques tels que les gradients appliqués à une image. Ce projet nous a également confrontés à la difficile problématique de détection de seuils dans des algorithmes.