

Filtre de Canny

Compte-rendu



Introduction

Le module d'analyse et traitement d'image s'inscrit dans la seconde année de la formation IMR (Informatique Multimédia et Réseaux) à l'ENSSAT. Ce module permet de nous faire découvrir, via une mise en pratique, le célèbre algorithme conçu par John Canny en 1986. L'algorithme de M. Canny, consiste à détecter les contours au sein d'une image.

Ce filtre de Canny se décompose en trois axes. Dans un premier temps nous verrons comment réduire le « bruit » dans une image puis nous aborderons la suppression des non-maximums avant de finir sur la dernière étape, le seuillage par hystérésis de façon semi-automatique.

Aussi pour la suite de ce rapport, le choix d'une image de petite taille (3x3 pixels) permet de mieux comprendre chacune des étapes de l'algorithme et valide également le fonctionnement de cet algorithme. J'analyserai donc à chaque fois, le résultat attendu dans un premier temps puis le résultat obtenu après traitement sur le logiciel Scilab (version 5.5.2) avec la bibliothèque SIVP qui permet de traiter une image.

I- Réduction du bruit

En 2018, une image n'est pas toujours nette. Malgré les différents progrès techniques dans une majorité des systèmes numériques et l'apparition récente des voitures complètement autonomes ou encore des derniers smartphones permettant de capturer des images de la plus haute qualité qu'il soit sur le marché, une image peut parfois être « bruitée ». Le bruit d'une image est la présence d'informations parasites qui s'ajoutent de façon aléatoire aux détails de la scène photographiée numériquement. Pour représenter ce bruit, voici un exemple :



Illustration 1: Image sans bruit



Illustration 2: Image avec bruit

A) Application du filtre gaussien

La première étape de l'algorithme va donc consister à supprimer un maximum de bruit sur l'image que l'on souhaite traiter. On appliquera pour cela le filtre gaussien. Ce filtre va parcourir l'ensemble de l'image et mettre en évidence le point central et réduire progressivement les autres valeurs voisines. Il existe plusieurs tailles différentes de ce filtre afin d'obtenir de meilleurs résultats. Plus le filtre est grand, plus le traitement sera long et efficace et vice versa. Dans le cadre du projet, j'ai utilisé le filtre 3x3 et 5x5 définis comme cela :

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Illustration 4: Filtre gaussien 3x3

$$\frac{1}{256} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

Illustration 3: Filtre gaussien 5x5

Puis l'application de ce masque se fait de la façon suivante. Le pixel centrale sera positionné sur le premier pixel de notre image à traiter puis l'ensemble du masque va parcourir l'ensemble des pixels à traiter sur l'image. Nous sommes sur un traitement linéaire et par conséquent, plus l'image sera grande, plus le temps de traitement sera fastidieux.

Prenons une image 3x3 afin d'avoir une représentation de sa matrice (en niveaux de gris allant de 0 à 255).

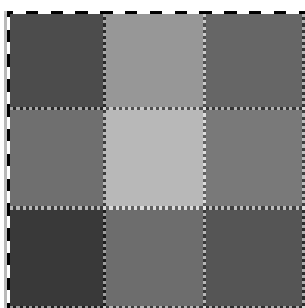


Illustration 5: Image
3x3 pixels

Matrice initiale

```
70.    146.   96.
105.   180.  115.
51.    103.   79.
```

La seule constatation que nous pouvons faire ici est que le niveau de gris le plus haut est 180 et que le plus bas (proche du noir) est le 51 parmi ces neuf pixels. Appliquons manuellement le filtre gaussien 3x3 pour le pixel central.

Pour cela, il s'agira de superposer le filtre sur l'image et de multiplier chacun des points superposés entre eux avant de les sommer et de diviser le tout par la somme des coefficients du filtre. Pour calculer le pixel central voici le calcul attendu :

$$Image_{filtrée}(2,2) = (1*70 + 2*146 + 1*96 + 2*105 + 4*180 + 2*115 + 1*51 + 2*103 + 1*79) / 16 = 122,125$$

Quand le filtre dépasse de l'image, la valeur par défaut choisie est le noir (0). En appliquant ce filtre sur l'ensemble de la matrice on obtient ceci :

Application d'un filtre : Gaussien 3x3

60.	93.	67.
79.	122.	88.
50.	78.	58.

Illustration 6: Image filtrée par filtre gaussien 3x3

Ce résultat semble cohérent avec le résultat trouvé précédemment.

B) Application d'un gradient

Cette étape va permettre de retourner l'intensité d'un contour grâce à la formule suivante : $|(G)| = \sqrt{(G_x^2 + G_y^2)}$

Cette formule peut être approximée en appliquant deux filtres consécutivement sur l'image G_x et G_y , c'est-à-dire : $|(G)| = |(G_x)| + |(G_y)|$

où : $G_x = [1, 0, -1]$ et $G_y = [1; 0; -1]$

G_x permettra de détecter l'ensemble des contours perpendiculaires à l'image et G_y quant à lui indiquera l'ensemble des contours horizontaux. Voici l'application de ces filtres :

Application d'un filtre : Gradient X

- 93.	- 7.	93.
- 122.	- 9.	122.
- 78.	- 8.	78.

Application d'un filtre : Gradient Y

```
- 79.  - 122.  - 88.  
 10.    15.    9.  
 79.    122.   88.
```

Enfin, on souhaite à présent récupérer $|G|$. Donc on va maintenant appliquer cette formule $|G| = \sqrt{G_x^2 + G_y^2}$ afin d'obtenir une matrice qui informe sur l'intensité des contours. Nous obtenons alors le résultat suivant :

Norme du gradient

```
122.02459    122.20065    128.03515  
122.40915    17.492856    122.33152  
111.01802    122.26201    117.59252
```

C) Direction du contour

Enfin pour terminer la première étape de l'algorithme, il faut retourner une matrice qui informe sur la direction du contour de chacun des pixels. Cela permettra par la suite de corréler un contour avec la norme et la direction du pixel.

Pour ce faire, appliquons la formule suivante $\theta = \pm \arctan(G_y/G_x)$. Ce résultat retourne une valeur entre $-3,14$ et $3,14$. L'idée va donc être d'approximer une direction $0, 45, 90, 135^\circ$ en fonction de la valeur retournée. Pour être plus clair, convertissons θ en degrés (ex : $2,08\text{rad} \times 180^\circ / 3,14\text{rad}$). Nous aurons ainsi une valeur comprise entre -180° et 180° . Voici comment ci-dessous, l'illustration du choix des quatre directions possibles.

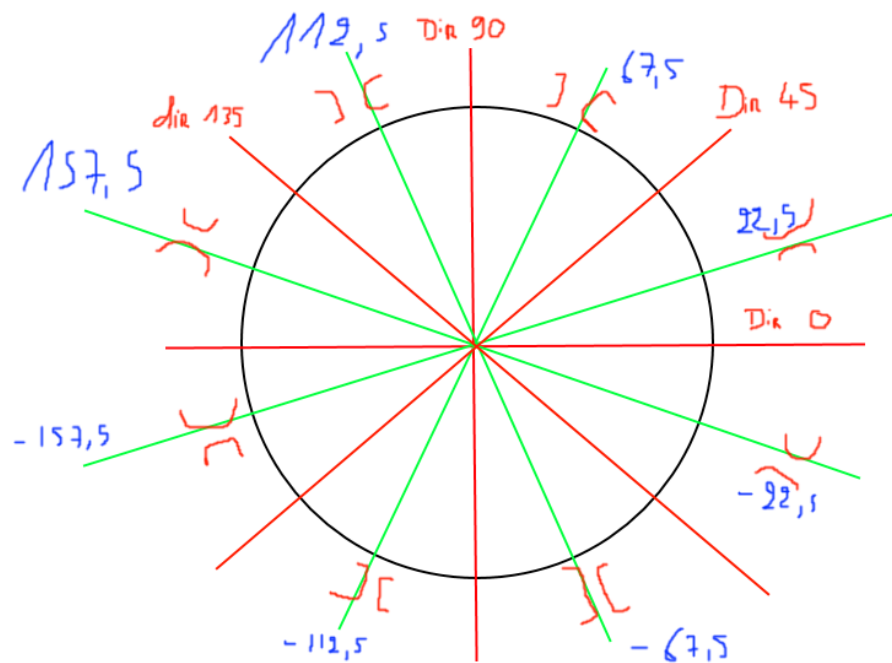


Illustration 7: Direction des contours

Une fois cette étape réalisée, nous obtenons la matrice contenant l'ensemble des directions de chaque pixel de notre image. Voici le résultat obtenu pour notre matrice initiale :

Angle de la normale du gradient

135.	0.	45.
90.	45.	90.
45.	0.	135.

Illustration 8: Orientations des contours

II- Suppression des non-maximums

Maintenant que nous avons l'intensité de chaque pixel dans notre image, l'idée va être de supprimer tous les pixels qui ne sont pas maximum en fonction de leur gradient. Les deux matrices calculées à l'étape précédente vont être utilisées ici afin de déterminer la nouvelle image. Pour donner un exemple, si le pixel $\text{img}(1,2) = 240$ et a un gradient de 45° , nous chercherons à récupérer ses deux voisins sur cet angle puis comparer si le pixel sur le quel nous nous situons est le plus intense par rapport à ses deux voisins. Si c'est le cas, nous garderons cette valeur, dans le cas contraire, on passera la valeur du pixel à 0.

Voici les deux matrices d'entrées :

Norme du gradient

122.02459	122.20065	128.03515
122.40915	17.492856	122.33152
111.01802	122.26201	117.59252

Angle de la normale du gradient

135.	0.	45.
90.	45.	90.
45.	0.	135.

Si l'on le pixel à la position 2,2 de la norme, nous avons la valeur 17, sa direction correspond à 45° . On récupère par conséquent ces deux voisins, $\text{pixel1} = (1,3)$ et $\text{pixel2} = (3,1)$. Puis nous comparons le 17 à ses deux voisins, dans notre cas 128 et 111 et s'il est un maximum on gardera sa valeur. Ici nous supprimons la valeur car 17 est inférieur à pixel1 et pixel2 . Voici la matrice finale pour cette image :

Suppression des non-maximums

122.02459	0.	128.03515
122.40915	0.	0.
111.01802	122.26201	117.59252

III- Seuillage par hystérésis

A) Sans calcul des seuils automatique

Enfin la dernière étape de l'algorithme consiste à reprendre la matrice des non-maximums puis à parcourir deux fois l'image. Une première fois afin de déterminer en fonction d'un seuil haut, les contours (255) de l'image de façon certaine et également en fonction d'un seuil bas afin de déterminer si ce n'est pas un contour (0). Une fois cette étape réalisée, on effectuera un autre passage sur l'image afin de déterminer si les valeurs entre ces deux seuils sont des contours ou non.

Dans notre exemple, reprenons la matrice des non-maximums suivante :

Suppression des non-maximums

122.02459	0.	128.03515
122.40915	0.	0.
111.01802	122.26201	117.59252

et prenons un seuil haut défini manuellement à 120 et un seuil bas à 112. On s'attend à avoir 4 pixels passer à 255, 1 pixel à 0 (le pixel à 111 car inférieur à 112) et 1 pixel à déterminer en fonction de ses voisins perpendiculaire au gradient. On obtient la matrice suivante :

Matrice partiellement traitée

255.	0.	255.
255.	0.	0.
0.	255.	117.59252

Pour déterminer cette valeur, on récupère la direction de ce pixel (ici 135°), on applique une rotation de 90° au gradient afin d'être perpendiculaire à cette direction. Puis on regarde si les deux voisins de la direction à 45° sont déjà des contours. Dans ce cas précis nos deux voisins sont hors de la matrice (4 ; 2 et 3 ; 4) et ne sont par conséquent pas des contours.

Donc la matrice finale ressemble à ceci :

Image finale


255.	0.	255.
255.	0.	0.
0.	255.	0.

B) Avec calcul des seuils automatique

Comme nous avons pu le constater, la valeur des seuils choisis font varier le résultat final. Plus le seuil haut sera élevé et plus le nombre de contours dans l'image diminuera mais plus ces contours seront fiables. L'idée est qu'un utilisateur puisse lui même choisir un ratio dans l'image afin d'obtenir un résultat de contours plus ou moins fiable. De façon général le ratio sera compris entre 0,7 et 0,9.

Pour réaliser ce calcul, il va tout d'abord falloir récupérer l'histogramme de notre image, ce qui nous retourne le nombre de valeurs de gris dans l'image. Une fois cette histogramme récupéré, on normalisera celui-ci avant de le cumuler ce qui nous donnera un ratio de 0 à 100 % avec une correspondance en niveau de gris. Pour reprendre l'exemple de notre matrice 3x3, voici le résultat attendu :

i	51	70	79	96	103	105	115	146	180
H(i)	1	1	1	1	1	1	1	1	1
Hn(i)	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9
C(i)	1/9	2/9	3/9	4/9	5/9	6/9	7/9	8/9	9/9


 70 % 80 %

Toutes les valeurs ne sont pas représentées ici mais l'histogramme contient bien les 256 niveaux de gris. Seuls les niveaux utilisés dans notre exemple ont été ajoutés à la table. On peut donc voir ci-dessus qu'avec un ratio de 70 %, notre seuil haut sera de 115 et par conséquent le seuil bas sera de 57,5 (seuil bas = seuil haut / 2). Nous obtenons le résultat matricielle suivant :

Matrice partiellement traitée

```

255.      0.    255.
255.      0.     0.
111.01802 255.  255.

```

Image finale

```

255.  0.    255.
255.  0.     0.
0.    255.  255.

```

IV- Jeux de tests

Enfin j'ai appliqué l'algorithme sur une image plus importante afin d'obtenir un résultat plus visuel. J'ai donc essayé avec trois valeurs de ratio différentes sur une image d'Andy Murray et voici les résultats ci-dessous. On remarquera que plus on augmente le ratio et plus le visage d'Andy se dessine sur l'image résultante :

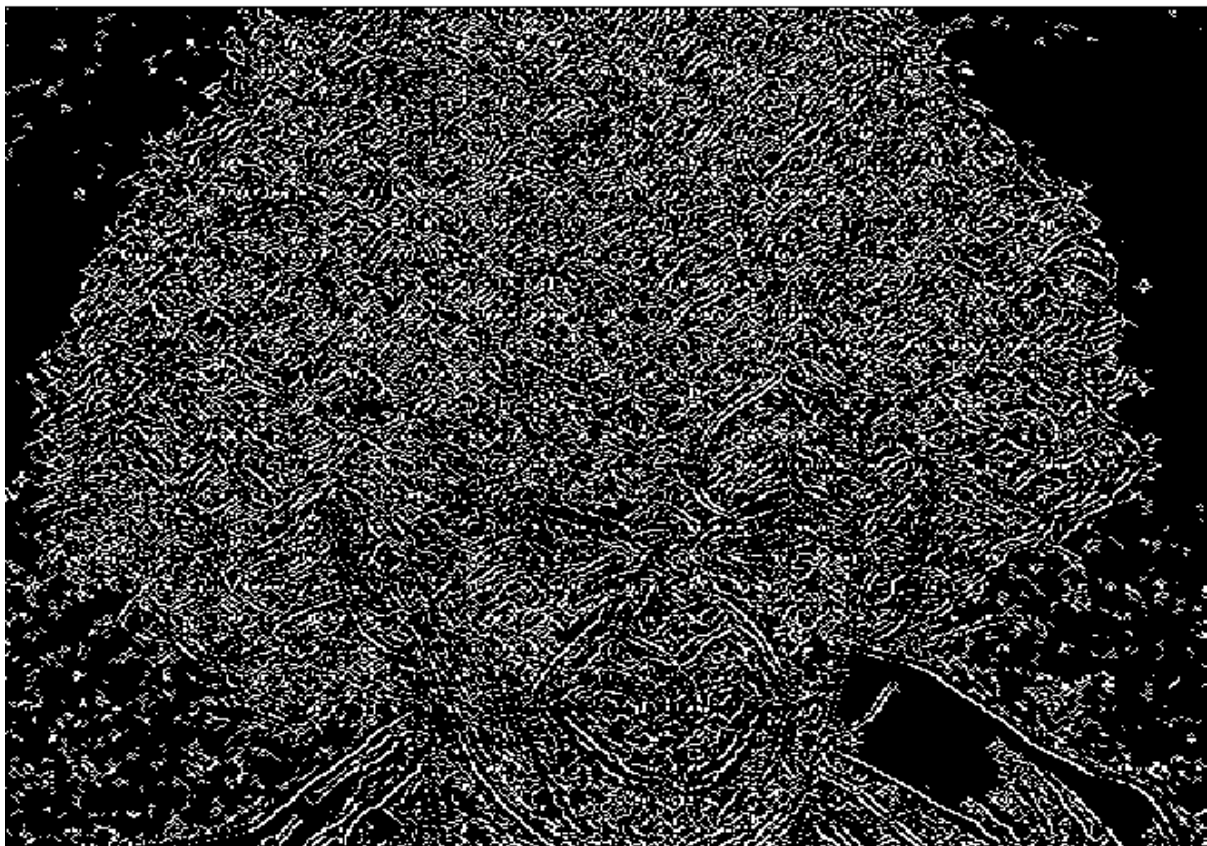


Illustration 9: Murray ratio 70



Illustration 10: Murray ratio 80



Illustration 11: Murray ratio 90

Conclusion

Ce projet basé sur l'application du filtre de Canny, m'a permis d'approfondir mes connaissances au niveau du langage Scilab et m'a également permis de me créer une référence quant au fonctionnement du traitement d'une image de façon générale.

J'ai apprécié la mise en application de cet algorithme. J'ai préféré choisir une image extrêmement petite pour valider étape par étape le bon fonctionnement de l'algorithme avant de l'appliquer sur image plus importante afin d'obtenir un résultat concluant. Cependant, malgré ces précautions, les jeux de tests sur Andy Murray montrent effectivement la délimitation de différents contours dans l'image mais le résultat obtenu ne semble pas idéal.

Aussi, j'ai pu remarquer que le temps de traitement d'une image était excessivement long plus l'image était grande. J'imagine que dans le domaine du traitement d'images, les ordinateurs qui s'occupent de ces tâches sont optimisés exclusivement pour ça et que des astuces au niveau algorithmique existent comme par exemple la séparation d'un masque 5x5 en deux masques 1x5 (ligne et colonne).