Dugne Quentin
Glédel Clément

# Compte rendu : Détection de contours

Ce TP a pour objectif de détecter les contours présents dans une image à l'aide d'opérateurs dérivatifs et de comparer les résultats obtenus afin de juger leur efficacité.

## 1. Le gradient

Une première méthode nous permettant d'obtenir une carte binaire des contours est le gradient. Ce filtre dérivatif du premier ordre se base sur la détection des maximas et de la direction du gradient d'une image.

Afin d'étudier cette direction du gradient, nous avons utilisé les masques directionnels de Kirsh qui vont nous permettre de discrétiser l'espace des directions. Ainsi nous disposons de 8 masques, correspondant à chacune des directions, pour lesquels nous calculerons 8 gradients différents. La valeur maximum en chaque pixel sera retenu nous permettant ainsi d'obtenir la direction associée à cette valeur. Il existe bien entendu des méthodes plus avancées ou plus calculatoire mais le résultat obtenu reste similaire.

La seconde étape réside dans la suppression de non maximas, pour cela nous avons décidé de nous appuyer sur la discrétisation de la direction, cette méthode va nous permettre d'amincir les contours obtenus.

Cependant beaucoup de contours non existants apparaissent sur le résultat, nous devons donc appliquer une méthode de seuillage afin d'éliminer les contours en trop qui peuvent généralement être dû au bruit. Nous effectuons donc un seuillage basique, c'est à dire que nous n'acceptons que les valeurs du gradient au dessus d'un certain seuil. Nous obtenons donc le résultat suivant :

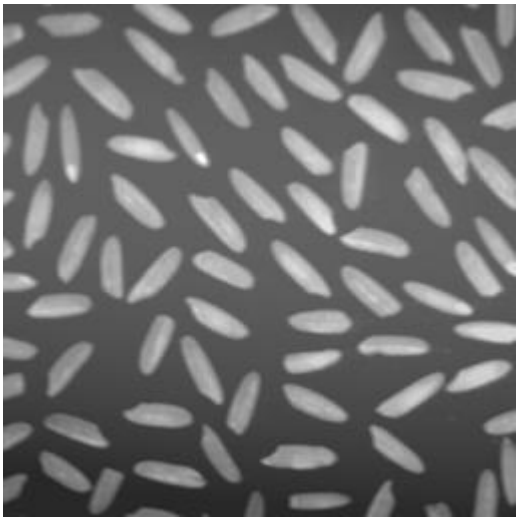
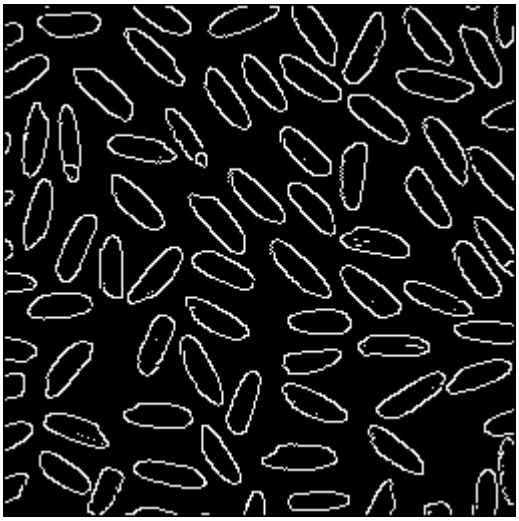


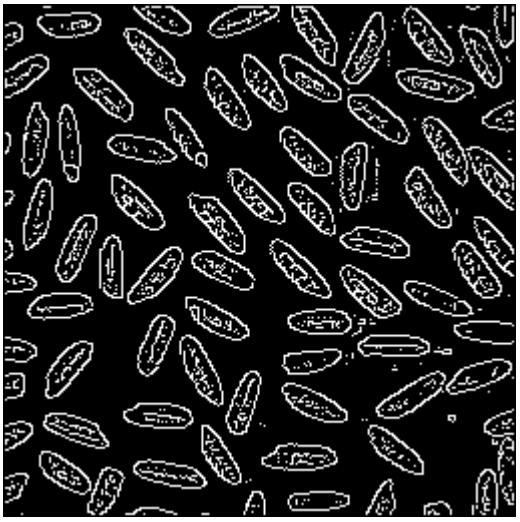
Image originale



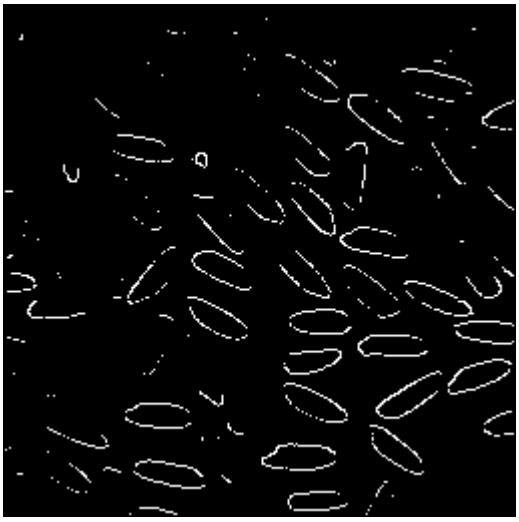
Gradient avec un seuil de 90

Nous avons ici une image résultante très correcte, même si quelques points qui ne sont pas des contours subsistent. Cela est dû à la variation importante dans les nuances de gris, puisque comme nous pouvons le voir sur l'image, plusieurs pixels sont blancs à certains endroits.

Néanmoins, le choix du seuil a une grande importance puisque si celui-ci est trop faible alors il est possible d'accepter de faux contours alors qu'à l'inverse s'il est trop grand, il y a des chances pour que les vrais contours ne soient pas détecter. C'est ce que l'on peut voir avec les deux images ci-dessous:



Gradient avec un seuil de 40



Gradient avec un seuil de 150

Pour l'image de gauche, le seuil choisi est trop faible, ainsi on s'aperçoit que les pixels blancs détectés dans l'image subsistent. Dans le cas contraire, un seuil trop grand va venir supprimer des contours corrects. C'est le cas de cette image où le contour de plusieurs grains est en partie effacé à cause du seuil élevé.

Afin de palier à cette solution il existe une seconde méthode de seuillage que nous n'avons pas implémenter au cours de ce TP. Celle ci se base sur un double seuillage.

Cet opérateur reste tout de même relativement sensible au bruit, il est donc préférable d'appliquer un filtre gaussien au préalable afin de lisser l'image comme c'est le cas dans le filtre de Canny.

## 2. Le filtre Laplacien

Le gradient n'est pas le seul opérateur dérivatif permettant d'obtenir une carte binaire des contours, en effet il est possible de l'obtenir en utilisant un filtre laplacien 8 voisins. Pour cette partie nous utiliserons le code implémenté lors des séances précédentes. Cette fois-ci, l'étape consistant à rechercher les non maximas n'est pas à effectuer, il faut néanmoins établir un seuillage. Les contours sont déterminés par un laplacien avoisinant la valeur nulle, il est néanmoins quasi impossible d'obtenir une valeur égale 0. Ainsi nous devons rechercher dans l'image le moment où, d'un pixel à un autre, nous avons un changement de signe dans les valeurs. Cette méthode se nomme Zero Crossing.

Cette méthode de seuillage vérifie que la différence entre les 2 pixels, entourant notre contour, dépasse un certain seuil. Si c'est le cas, alors nous l'acceptons. Nous obtenons donc le résultat suivant :

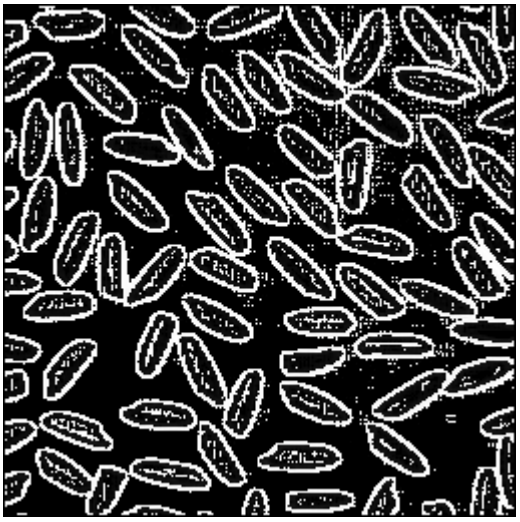
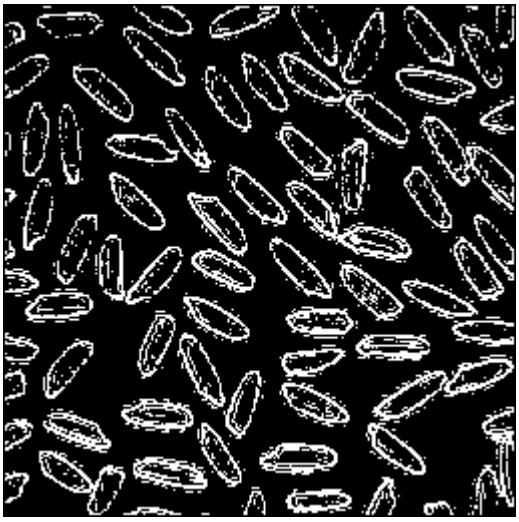


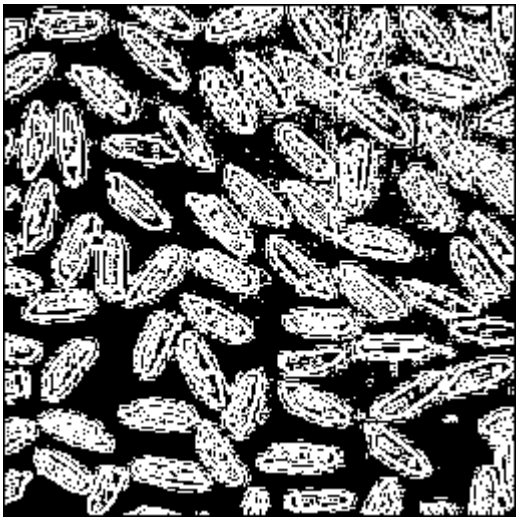
Image Laplacien



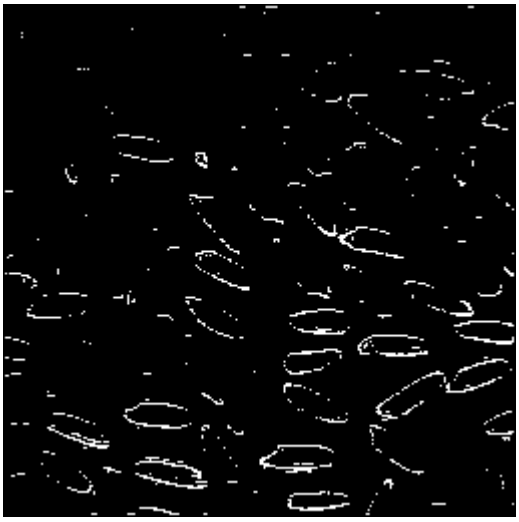
Laplacien seuillé à 25% du max

Comme on peut le voir sur l'image de gauche, le résultat obtenu avec le Laplacien est moins optimal que l'image obtenue avec le gradient. Plusieurs points blancs sont considérés comme étant des contours dans l'image. Ces points blancs persistent même après un seuillage à 25 % du max. Ainsi les contours des grains sont nettement moins lisse que l'image avec le gradient.

Encore une fois le choix du seuil est important comme le montre les images ci-dessous:



Laplacien seuillé à 5 % du max

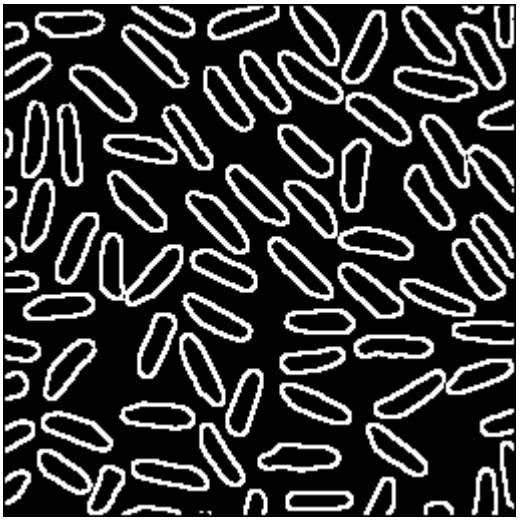


Laplacien seuillé à 50% du max

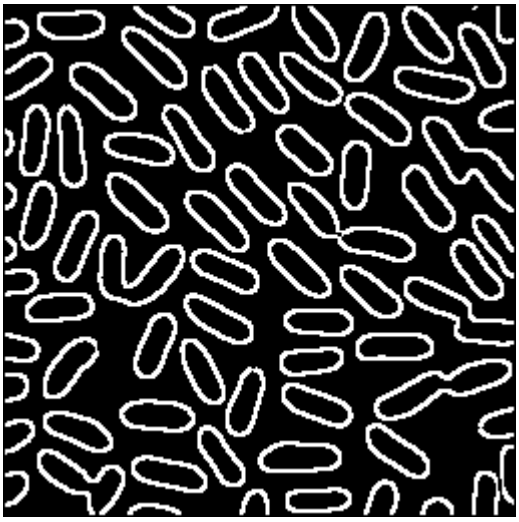
L'inconvénient de ce type de seuillage est que si celui-ci est trop faible, nous allons accepter plusieurs faux contours, ainsi nous rajoutons des erreurs dans l'image. C'est que l'on peut voir avec l'image de gauche, les points blancs à l'intérieur des grains sont considérés comme des contours ainsi que les valeurs qui les entourent. A l'inverse un seuil trop élevé va venir supprimer des contours corrects comme pour le seuillage du gradient.

## 3. LoG approximé par DoG

Enfin, nous avons implémenté une 3ème méthode, un peu plus avancée que la précédente. Pour des raisons de simplicité, nous implémenterons la méthode de DOG qui est une approximation de la méthode de LOG. Celle-ci se base sur un calcul de Gaussien. Nous effectuons une différence entre 2 gaussienne de sigmas différents (en respectant un facteur de 1,6 entre les 2 sigmas (2 Gaussiens voisins)) afin d'obtenir une approximation de la valeur du Laplacien. La méthode de seuillage est la même qu'auparavant. Nous obtenons un résultat de la forme suivante :



DoG(sig1=3.68,sig2=2.3) seuillé à 20% du max

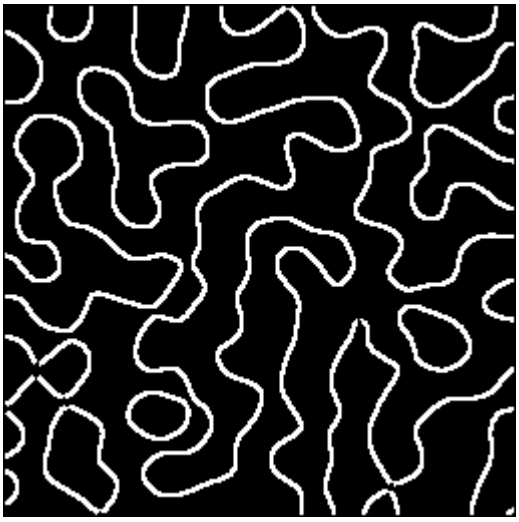


DoG(sig1=6.4,sig2=4) seuillé à 20% du max

Le sigma va tout d'abord avoir une influence sur l'épaisseur du contour, en effet comme le montre ces 2 exemples, le contour semble plus épais pour l'image de droite. Les résultats employés ici sont très correctes, mais quelques difficultés peuvent être rencontrées lors du choix du sigma. En effet il est impératif de choisir une taille de gaussienne impair, il faut donc faire attention à la valeur du sigma que l'on choisit. En ce qui concerne le seuillage, celui-ci fonctionne de la même manière que le précédent. Mais l'utilisation d'une gaussienne améliore considérablement le résultat.

En revanche le sigma est un paramètre qu'il ne faut absolument pas négligé. En effet nous obtenons des résultats totalement différents en fonction du sigma choisi.

DoG(sig1=1,sig2=0.625) seuillé à 50% du max



DoG(sig1=16,sig2=10) seuillé à 4% du max

Comme nous pouvons le voir un seuil très petit va amincir les contours et aura une qualité similaire que le filtre laplacien. A l'inverse un sigma trop important aura de mauvais résultats puisque celui-ci prendra un filtre de taille trop imposante lors de la convolution, changeant ainsi les valeurs de l'image.

De premier abord, le résultat obtenu avec le gradient semble prometteur à l'inverse du filtre laplacien qui semble extrêmement sensible au bruit. Le DOG quant à lui semble aussi obtenir de bon résultats mais beaucoup de paramètres reste à fixer contrairement au gradient.

Par une étude visuelle on s'aperçoit que chaque opérateur dispose de ces petits défauts. Aucun d'entre eux ne permet parfaitement de détecter les contours, on se retrouve souvent avec des pixels manquants ou superflus.

4. Comparaison quantitative

Afin d'approfondir notre étude nous allons analyser les images de manières quantitatives. Pour cela nous disposons d'un large panel d'image avec leur contours correspondants. Ainsi nous allons comparer nos résultats avec le contour original de l'image. Pour cela nous utiliserons 5 grandeurs qui nous permettrons d'établir les performances de nos algorithmes. Nous travaillerons sur 3 images.

La première image est un ours sur un fond majoritairement uniforme mais disposant tout de même de quelques contours. Son pelage dispose de peu de variations sauf au niveau du visage. Les contours doivent donc se situer au niveau du visage, de la forme de l'ours mais aussi au niveau des caillou en haut à gauche.

Image originale : bear_2

1) Gradient seuillé à 83

2) Laplacien seuillé à 14% du max

3) DoG(sig1=3.68,sig2=2.3) seuillé à 25% du max

Certains contours présents dans l'image de référence ne sont détectés dans aucune des 3 images résultantes. En effet cela reste normal puisqu'il n'y a pas de grande variations de couleurs, ainsi aucun des algorithmes n'est capable de détecter les contours (narines, pierre, pelage..). Ainsi le nombre de faux négatifs sera conséquent.

| Images | Contours détectés | Contours référence | Contours corrects | faux positif | faux négatifs | Performance | Taux de FP | Taux de FN |
|---|---|---|---|---|---|---|---|---|
| 1 | 5412 | 5978 | 2675 | 2737 | 3303 | 30.69% | 31.41% | 37.90% |
| 2 | 8912 | 5978 | 2536 | 6376 | 3442 | 20.53% | 51.61% | 27.86% |
| 3 | 7553 | 5978 | 3706 | 3847 | 2272 | 37.72% | 39.15% | 23.12% |

Les observations effectuées auparavant semblent être correct, le laplacien est l'opérateur dérivatif disposant des moins bons résultats, contrairement au DoG qui dispose d'une performance plus importante. Le laplacien quant à lui semble très sensible aux variations et dispose d'un nombre conséquent de faux positifs. Le gradient dispose de moins de faux positifs que le DoG. En effet si l'on regarde l'image, on aperçoit un nuage de points non reliés entre eux réparti équitablement le long des contours. A l'inverse, le DoG dispose de traits continus sur des contours spécifiques, il est donc normal que le gradient est un nombre de faux négatifs plus importants que son homologue.

La deuxième image représente un lion et une lionne sur une pierre avec diverses variations de niveaux de gris. Le fond de l'image est très uniforme ce qui ne devrait poser aucun problème pour les algorithmes. En revanche les variations au niveau de la crinière ne sont pas très contrastées, les contours ne seront sûrement pas détectés.

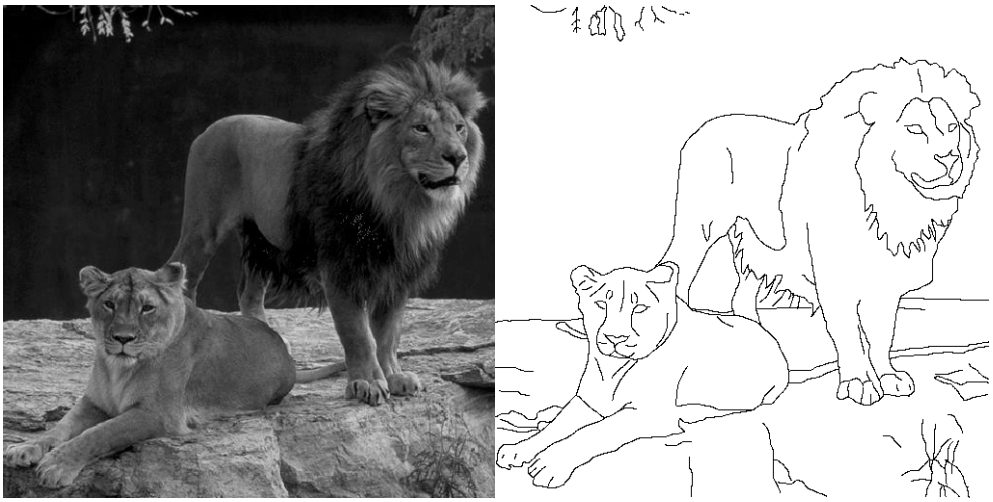
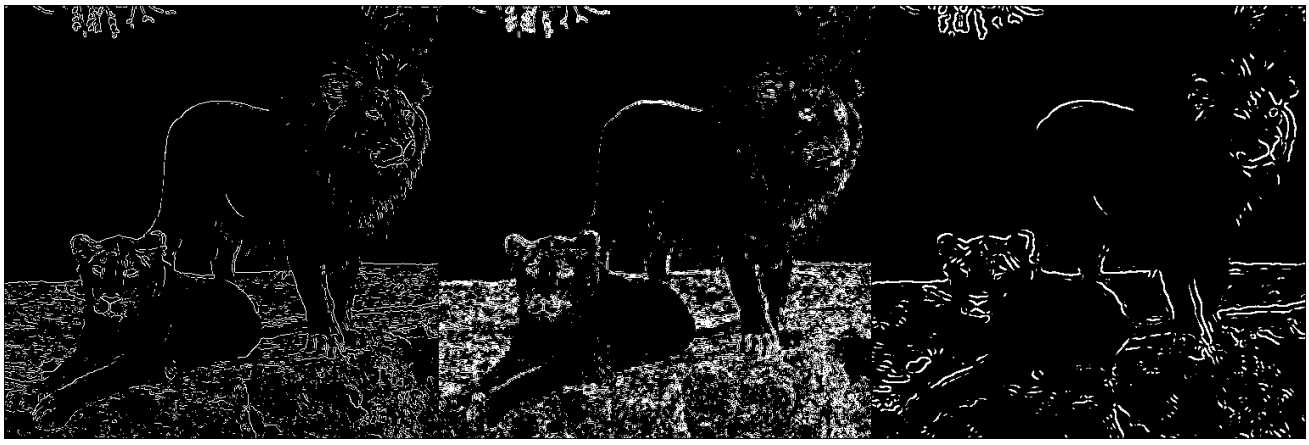


Image originale : lions



1) Gradient seuillé à 89 2) Laplacien seuillé à 31% du max 3) DoG(sig1=3.68,sig2=2.3) seuillé à 33% du max

| Images | Contours | Contours | Contours | faux | faux | Performance | Taux | Taux de |
|---|---|---|---|---|---|---|---|---|

| | détectés | référence | corrects | positif | négatifs | | de FP | FN |
|---|---|---|---|---|---|---|---|---|
| 1 | 12292 | 9005 | 5289 | 7003 | 3716 | 33.04% | 43.75% | 23.21% |
| 2 | 24501 | 9005 | 7812 | 16689 | 1193 | 30.40% | 64.95% | 4.64% |
| 3 | 10017 | 9005 | 5539 | 4478 | 3466 | 41.08% | 33.51% | 25.71% |

Dans cet exemple, le nombre de faux positifs sera plus important dû aux grandes variations de niveaux de gris au niveau de la pierre. Cela implique donc une baisse des faux négatifs. Ainsi cela induit grandement en erreur le gradient et le Laplacien. Comme on s'y attendait certain contours ne sont pas détectés. Les mêmes observations que précédemment sont à signaler. D'un point de vue visuel, le gradient semble plus appréciable que le DoG. L'ordre hiérarchique en terme de performance est une nouvelle fois vérifié mais elles sont plus élevé que le cas précédent.

La dernière image représente un troupeau de gnous dans un pré. On peut donc supposer que la détection des contours au niveau de l'herbe va poser problème, mais aussi au niveau des animaux rassemblés entre-eux puisque cela laisse une faible variations de couleurs.

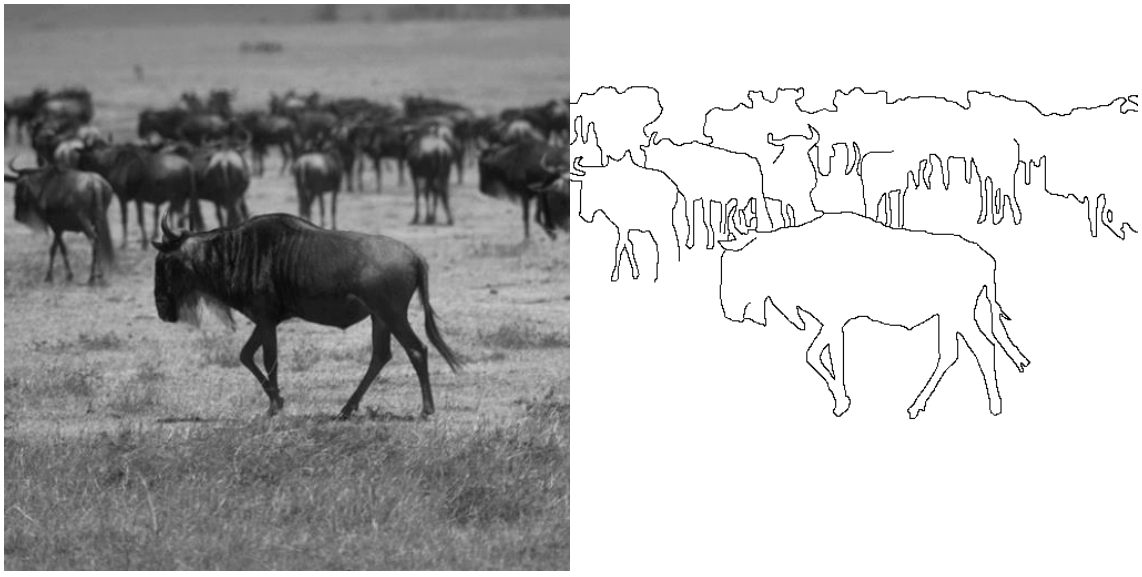
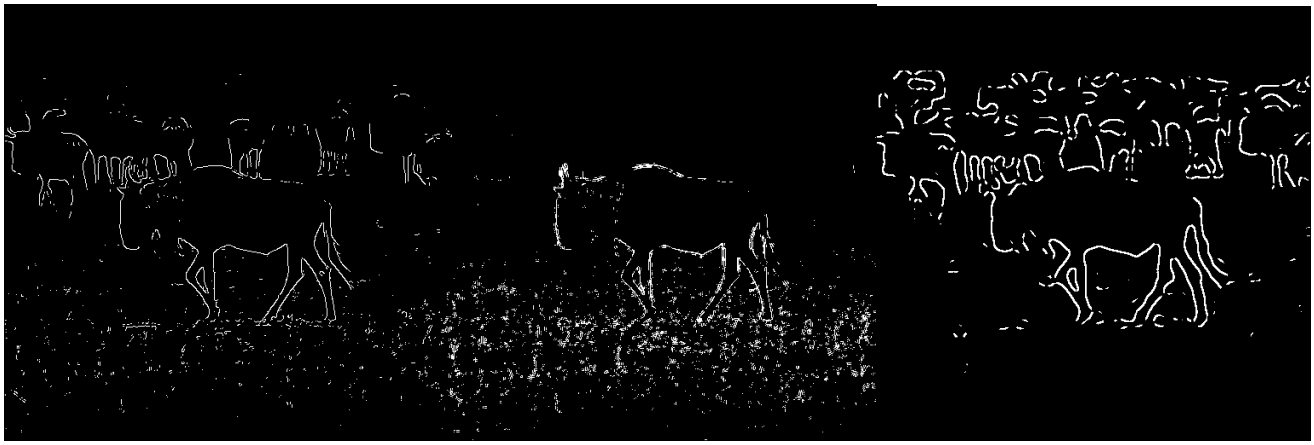


Image originale : gnu



1) Gradient seuillé à 89 2) Laplacien seuillé à 50% du max 3) DoG(sig1=6.4,sig2=4) seuillé à 21% du max

| Images | Contours détectés | Contours référence | Contours corrects | faux positif | faux négatifs | Performance | Taux de FP | Taux de FN |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 4488 | 6056 | 2800 | 1688 | 3256 | 36.16% | 21.80% | 42.05% |
| 2 | 7293 | 6056 | 1703 | 559 | 4353 | 14.62% | 48.00% | 37.38% |
| 3 | 9343 | 6056 | 5765 | 3578 | 291 | 59.84% | 37.14% | 3.02% |
| 4 | 8961 | 6056 | 6355 | 2606 | -299 | 73.37% | 30,08% | -3,45% |

Comme on s'y attendait la partie avec de l'herbe a posé des soucis conséquent au laplacien qui essaye de détecter des contours qui ne seront pas représentés sur l'image de référence. La performance s'en voit ainsi drastiquement réduite, puisque par la même occasion le laplacien n'arrive pas à détecter les contours au niveau du troupeau de gnous. Il en est de même pour le gradient. A contrario, la performance du DoG est épatante, celle ci coïncide parfaitement avec le contour dessiné à la main.

4) DoG(sig1=3.68,sig2=2.3)
seuillé à 38% du max

L'outil de mesure utilisé tout au long de cette partie présente un défaut majeur. En effet, dû à la non exactitude des positions des contours, nous ne regardons pas la présence d'un pixel à une position donnée mais dans un filtre 3x3. Cela peut donc être une source d'erreur dans les mesures de performances si plusieurs contours se retrouvent à proximité. C'est le cas de ce dernier exemple présenté. Nous détectons plus de contours corrects que de contours référence. Ainsi nous obtenons un taux de faux négatifs inférieur à 0 augmentant ainsi par la même occasion la performance.

## Conclusion

Les opérateurs permettant la détection de contours ne sont pas optimaux puisqu'ils sont sensibles aux bruits, ils dépendent grandement des paramètres mis en entrée. Les résultats obtenus donne des contours incomplets. Qui plus est la position des points de contours n'est pas toujours correct (obligation de chercher le contour dans un élément de taille 3x3). Cependant l'opérateur de contour n'est qu'une étape dans la détection. Diverses méthodes viennent compléter le résultat comme la liaison des contours. Mis à part ces quelques défauts, on s'aperçoit tout de même d'une hiérarchie qui se dessine d'un point de vue des performances même si le gradient devrait donner un bon résultat après la liaison des contours. Par la suite il est aussi possible d'établir la détection des points d'intérêts par la méthode de Harris pour obtenir des résultats optimaux.