

Rapport TP3 Analyse d'image

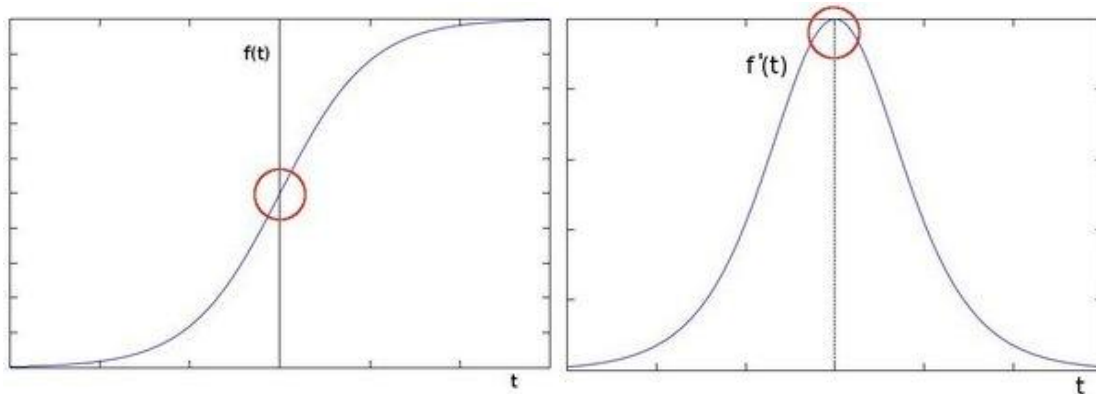
Contents

Présentation théorique des algorithmes de contours	2
Sobel.....	2
Laplace.....	3
Canny.....	4
Explication des fonctions d'OpenCV utilisé.....	4
GaussianBlur :.....	4
Sobel :.....	5
convertScaleAbs :.....	5
addWeighted :.....	5
Laplacian :.....	6
Canny :.....	6
Résultats	7
Test visual.....	7
Test sur golfcart.pgm	7
Test sur buffalo.pgm.....	9
Conclusion tiré.....	11
Test automatisé	11
Interprétation des résultats.....	13

Présentation théorique des algorithmes de contours

Sobel

Le filtre de Sobel est une méthode de détection de contours, qui est basée sur le calcul des gradients de l'intensité de chaque pixel. Elle recherche donc les extrêmes des dérivées selon les directions horizontales et verticales. Ci-dessous un exemple de contours détecté en utilisant la dérivée première.



Source : Site d'OpenCV

La méthode de Sobel calcule donc :

1. Deux dérivées :
 - a. Le changement horizontal : Qui est calculé avec le produit de convolution de l'image (noté I) avec le noyau G_x
- b. Le changement vertical : Qui est calculé de la même façon que les changements horizontaux.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

2. Et ensuite on combine les deux résultats afin de calculer une approximation du gradient.

$$G = \sqrt{G_x^2 + G_y^2}$$

La documentation précise aussi qu'elle peut utiliser une formule plus simple :

$$G = |G_x| + |G_y|$$

Le filtre de Sobel à plusieurs avantages :

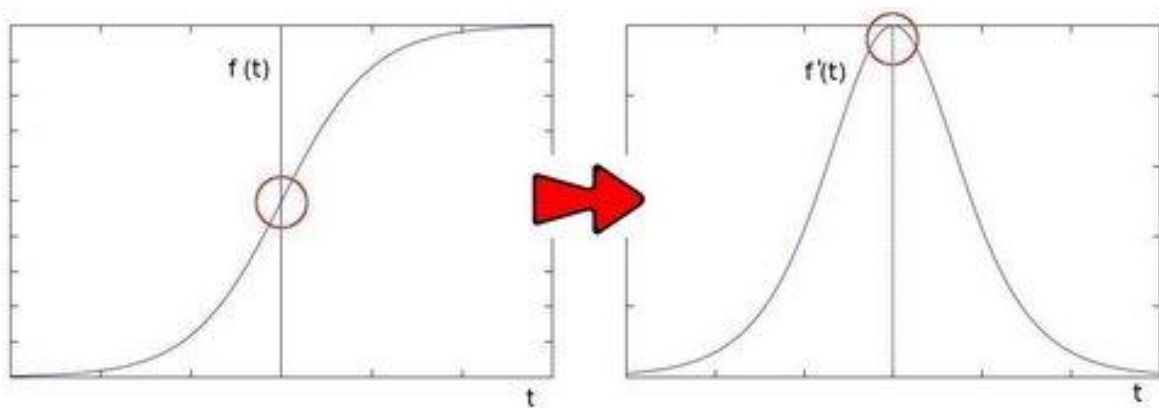
- Simple
- Calcul rapide car c'est une approximation

Néanmoins elle à plusieurs désavantages :

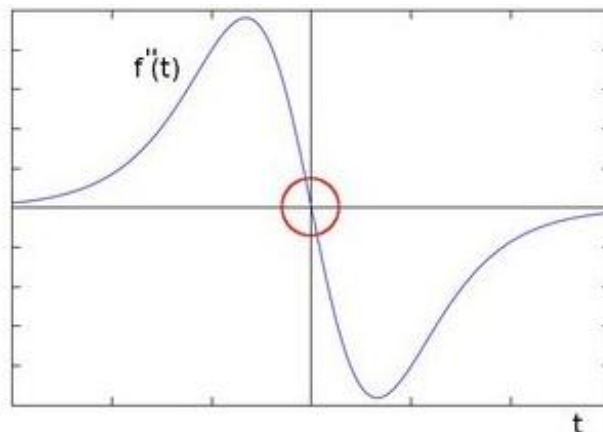
- L'approximation peut être fausse
- Si l'image est bruitée ce filtre fonctionne moins bien.

Laplace

Le filtre de Laplace est une méthode de détection de contours qui est basé sur le même principe que Sobel en terme calculatoire.



Mais ce coup-ci on va calculer la dérivée seconde



Lorsque la dérivée seconde passe par 0 cela met en évidence le contour. Le fait de passer par la dérivée seconde va rendre la méthode encore plus sensible au bruit.

D'après la documentation d'OpenCV l'opérateur de Laplace est basé sur :

$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Canny

Le filtre de Canny est une méthode de détection de contours qui est basé sur Sobel.

1. Tout comme Sobel :
 - a. On va calculer les changements horizontaux et verticaux

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- b. On trouve le gradient et la direction

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

2. On supprime les maximums pour garder que les pixels qui sont potentiellement un contour
3. Canny utilise ensuite deux seuils (haut et bas) :
 - a. Si le gradient du pixel est supérieur au seuil haut le pixel est accepté comme un contour.
 - b. Si le gradient du pixel est inférieur au seuil bas alors le pixel est rejeté.
 - c. Si le gradient du pixel est entre le seuil bas et haut il sera accepté que s'il est connecté à un pixel qui est supérieur au seuil haut.

Explication des fonctions d'OpenCV utilisé

GaussianBlur:

Cette fonction permet de réaliser un flou Gaussien sur une image, la variable qui permet de plus ou moins flouter une image est « ksize », par exemple si la valeur de cette variable est élevé plus l'intensité du flou sera conséquent.

Voici le prototype de la fonction :

```
void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT)
```

Ses différents paramètres sont :

- src : Image d'entrée
- dst : Image de sortie
- ksize : Taille du noyau.
- sigmaX : Déviation en X (je n'ai pas compris donc j'ai mis à 0).
- sigmaY : Déviation en Y (je n'ai pas compris donc j'ai mis à 0)

Sobel :

Cette fonction permet de calculer les images dérivées en utilisant l'opérateur de Sobel.

Voici le prototype de la fonction :

```
void Sobel(InputArray src, OutputArray dst, int ddepth, int dx, int dy, int ksize=3, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)
```

Ses différents paramètres sont :

- src : Image d'entrée
- dst : Image de sortie
- ddepth : C'est la profondeur de l'image de sortie, d'après la documentation d'OpenCV on va choisir CV_16S
- dx : Ordre de la dérivé en X
- dy : Ordre de la dérivé en Y
- ksize : Taille de la matrice (on utilisera comme valeur 3)
- scale : C'est le facteur qui est utilisé sur les valeurs des dérivées (c'est donc ce paramètre que l'on va faire varier)
- delta : Pour créer une image noir sur fond blanc nous allons mettre comme valeur 255 à ce paramètre

Afin de pouvoir faire complètement le filtre de Sobel il va donc falloir l'appliquer deux fois pour avoir le gradient en x et le gradient en y, puis utiliser une fonction pour les combiner (fonction `addWeighted()`). Cette fonction renvoie une image en niveau de gris il va falloir donc convertir l'image en image binaire par la suite.

convertScaleAbs :

Cette fonction permet de convertir une image en CV_8U d'après la documentation d'OpenCV

Voici le prototype de la fonction :

```
void cvConvertScaleAbs(const CvArr* src, CvArr* dst, double scale=1, double shift=0)
```

Ses différents paramètres sont :

- src : Image d'entrée
- dst : Image de sortie

addWeighted :

Cette fonction permet de créer une image à partir de la somme de deux autres images.

Voici le prototype de la fonction :

```
void addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma, OutputArray dst, int dtype=-1)¶
```

Ses différents paramètres sont :

- src1 : Image d'entrée 1

- alpha : Poids de la première image
- src2 : Image d'entrée 2
- beta : Poids de la deuxième image
- gamma : Nombre à ajouter à chaque somme (on met ce paramètre à 0)
- dst : Image de sortie

Laplacian:

Cette fonction permet d'appliquer le filtre de Laplace sur une image

Voici le prototype de la fonction :

```
void Laplacian(InputArray src, OutputArray dst, int ddepth, int ksize=1, double scale=1, double delta=0, int borderType=BORDER_DEFAULT)
```

Ses différents paramètres sont :

- src : Image d'entrée
- dst : Image de sortie
- ddepth : C'est la profondeur de l'image de sortie, d'après la documentation d'OpenCV on va choisir CV_16S
- ksize : Taille de la matrice de la dérivée seconde
- scale : C'est le facteur qui est utilisé sur les valeurs des dérivées (c'est donc ce paramètre que l'on va faire varier)
- delta : Pour créer une image noir sur fond blanc nous allons mettre comme valeur 255 à ce paramètre

Comme pour Sobel, cette fonction renvoie une image en niveau de gris, Il va falloir aussi convertir l'image de sortie en CV_8U et il va aussi falloir convertir l'image de sortie en image binaire.

Canny:

Cette fonction permet d'appliquer sur une image le filtre de Canny

Voici le prototype de la fonction

```
void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3, bool L2gradient=false)
```

Ses différents paramètres sont :

- image : Image d'entrée
- edges : Image de sortie
- threshold1 : C'est le seuil bas qui va permettre de choisir on non si le pixel est un contour
- threshold2 : C'est le seuil haut qui va permettre de choisir on non si le pixel est un contour (afin de le faire varier aussi on va partir du seuil bas et on le multiplie par un ratio (on a choisi un ratio de 3))
- apertureSize : Taille de la matrice utilisée pour sa fonction Sobel interne.

Afin d'avoir une image en sortie avec des contours noir sur fond blanc on doit faire la négative sur l'image. Cette fonction renvoie directement une image binaire il y a donc pas besoin de convertir son image de sortie.

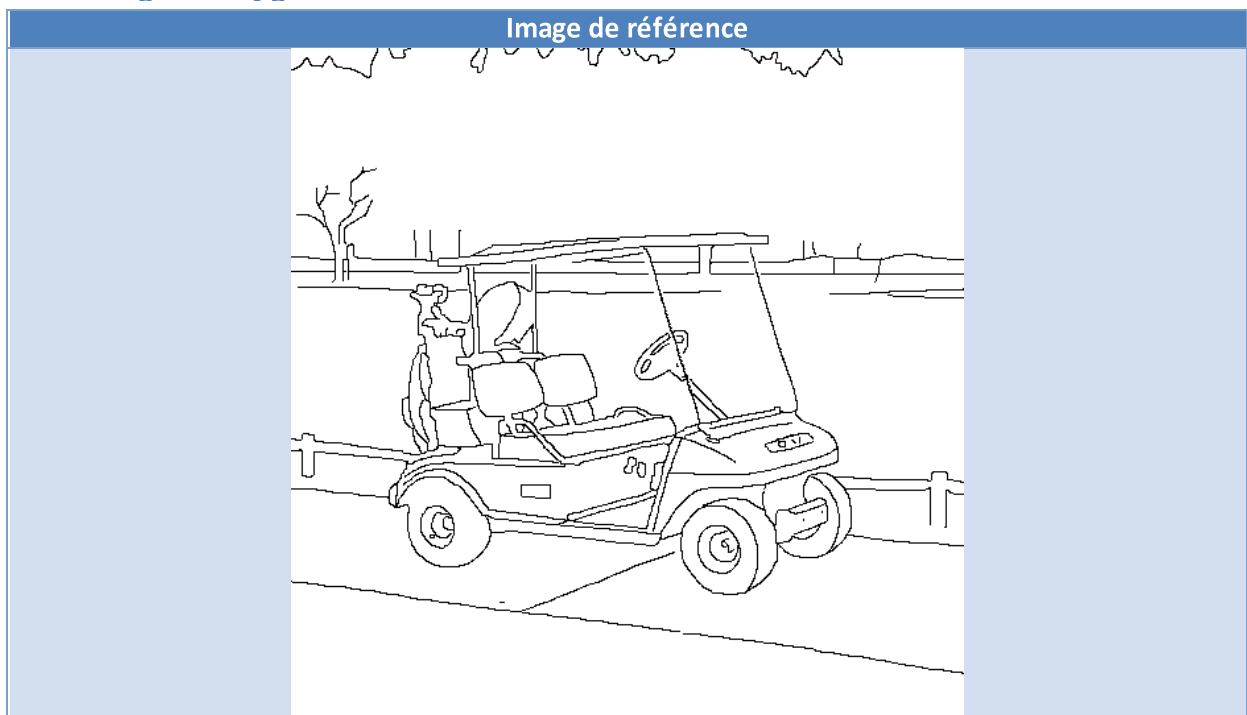
Résultats

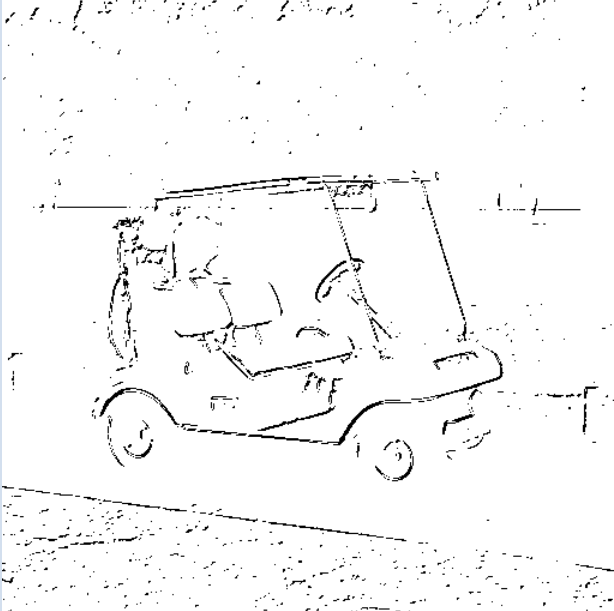

Note pour Canny le seuil max est défini par la formule suivante : $\text{seuil_min} * 3$

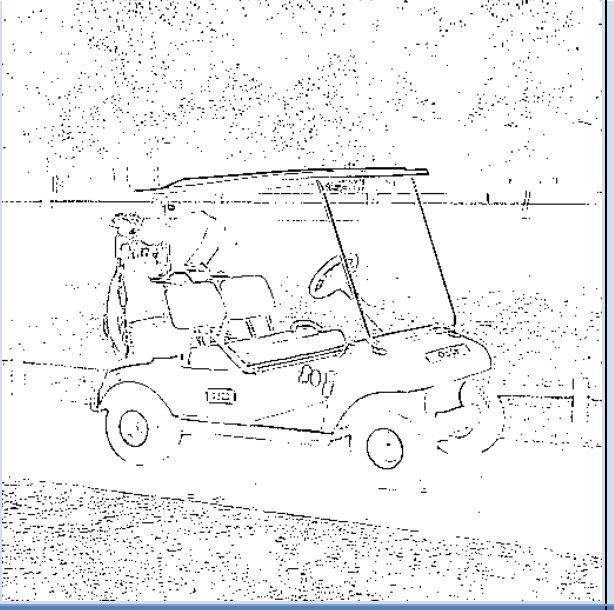

Test visual

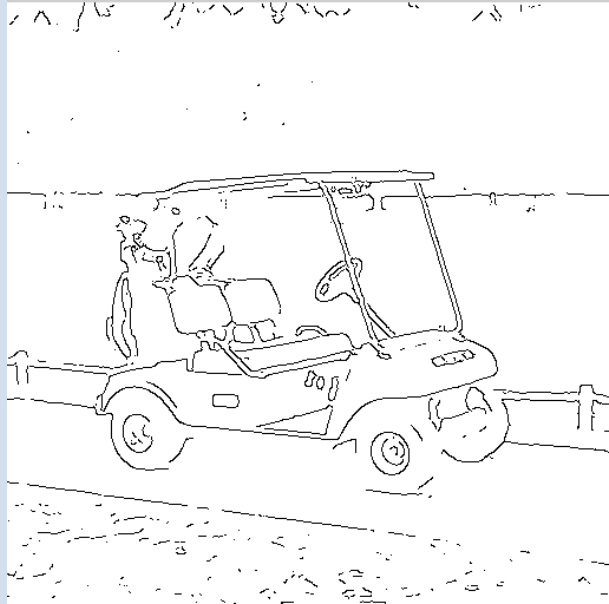
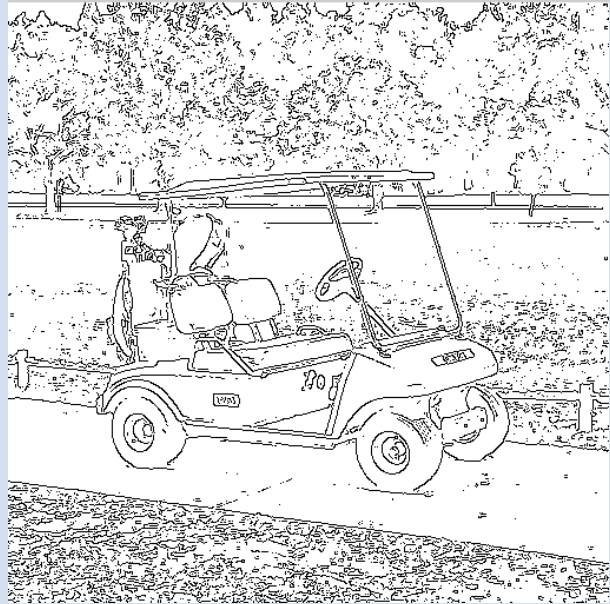
Ici on va essayer de voir l'importance du blur dans les différents algorithmes. On va prendre un cas optimal pour l'algorithme et un cas où l'algorithme est clairement moins efficace pour quasiment les mêmes paramètres.

Test sur golfcart.pgm

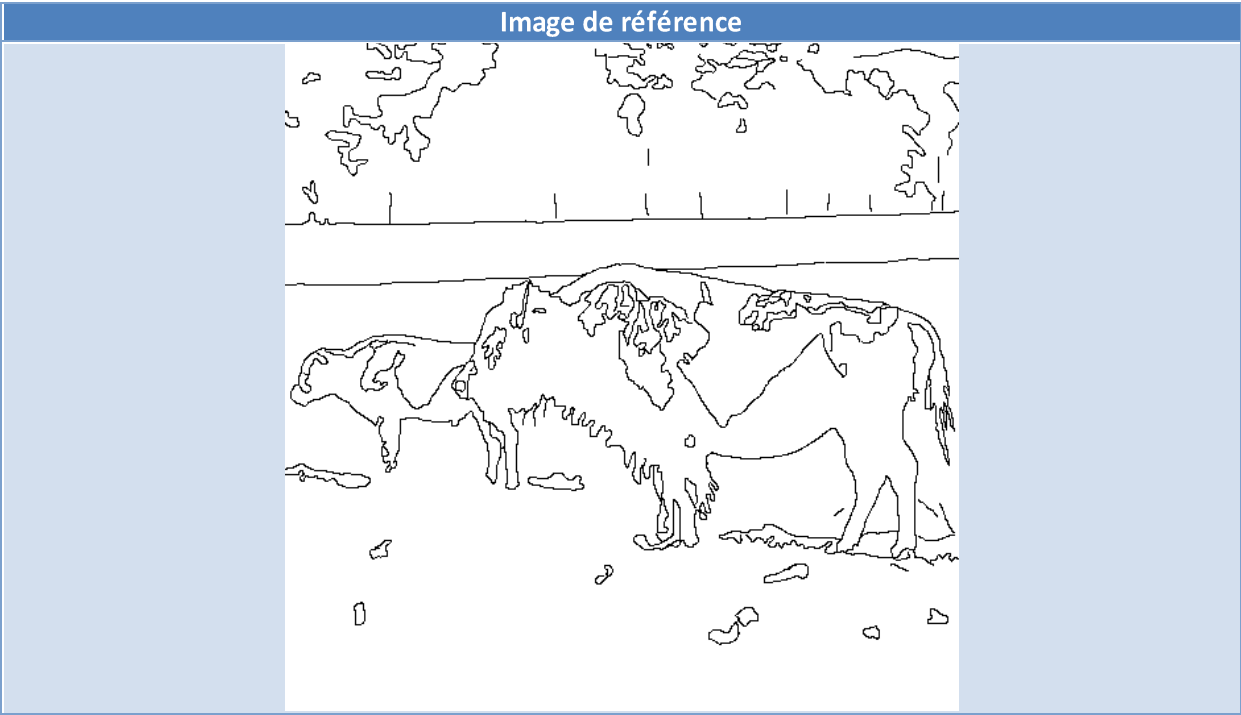



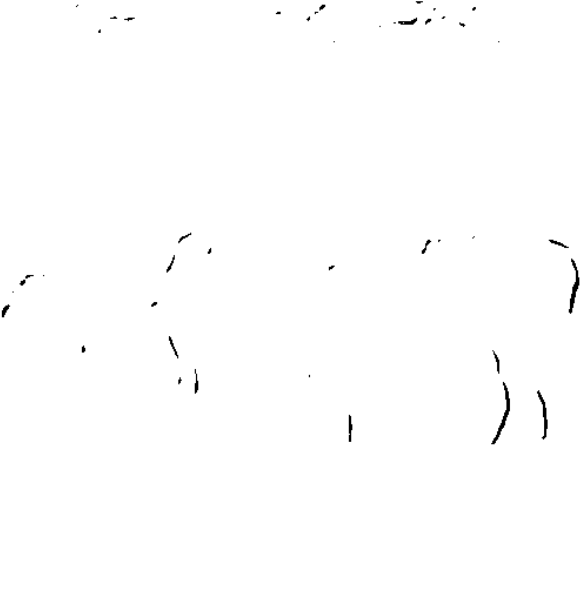
Sobel			
			
Parametre		Parametre	
Blur	1	Blur	7
Scale	1	Scale	1
Seuil	167	Seuil	167
Optimal		Non optimal	



Laplace			
			
Parametre		Parametre	
Blur	1	Blur	7
Scale	1	Scale	1
Seuil	147	Seuil	147
Optimal		Non optimal	

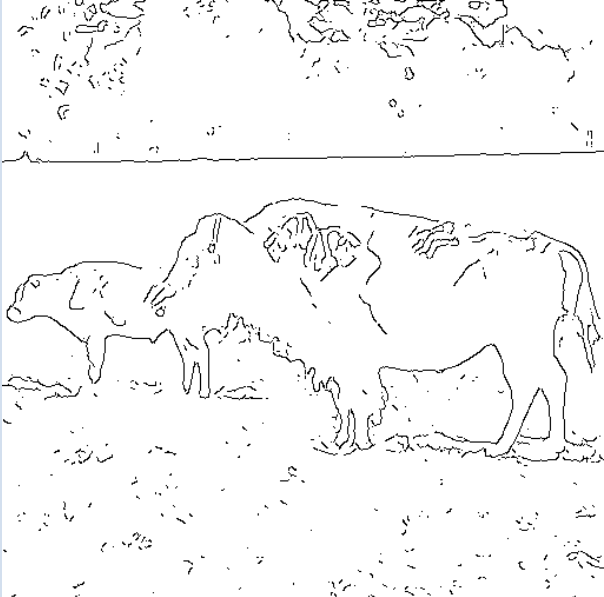

Canny			
			
Paramètre		Paramètre	
Blur	7	Blur	1
Seuil minimal	96	Seuil minimal	96
Optimal		Non optimal	

Test sur buffalo.pgm



Sobel			
			
Paramètre		Paramètre	
Blur	1	Blur	9
Scale	1	Scale	1
Seuil	148	Seuil	148
Optimal		Non optimal	

Laplace			
			
Parametre		Parametre	
Blur	3	Blur	9
Scale	2	Scale	2
Seuil	79	Seuil	79
Optimal		Non optimal	

Canny			
			
Paramètre		Paramètre	
Blur	9	Blur	15
Seuil minimal	96	Seuil minimal	96
Optimal		Non optimal	

Conclusion tiré

En comparaison avec l'image de référence on tire les conclusions sur les algorithmes suivants :

- Sobel :
 - Très sensible au niveau de flou mais il reste encore des contours détecté dans les tests fait.
 - Ne pas mettre de flou sur cet algorithme donne des résultats satisfaisant avec un bon seuil et un bon scale.
- Laplace :
 - Encore plus sensible que Sobel sur le flou, on perd rapidement les contours détecté
 - De même que Sobel ne pas mettre de flou et mettre des bons paramètres donne des résultats plutôt satisfaisant.
- Canny :
 - Mettre un blur sur cet algorithme accroît son résultat, cela permet de supprimer les formes inutiles comme l'herbe par exemple.
 - Mettre un blur trop élevé peut être en défaveur du résultat. En effet on commence à perdre des contours détectés si on accroît trop le blur.

Test automatisé

Du fait du grand nombre de valeur (car les tests ont été réalisé sur les 40 images données en exemple) tous les résultats brut ne seront pas mis dans ce rapport, ils sont disponible dans le fichier « log.csv ». On se contentera des quelques images en exemple (sans les paramètres utilisé pour chaque méthode) ainsi que des quelques moyennes.

Notation :

- FP : Faux positif
- FN : Faux négatif
- P : Performance
- TFP : Taux de faux positif
- TFN : Taux de faux négatif

Méthode	Déecté	Réf	Correct	FP	FN	P	TFP	TFN
goat_7.pgm								
SOBEL	5185	4638	1194	3991	3444	0.138371	0.46251	0.399119
LAPLACE	3922	4638	1643	2279	2995	0.237531	0.329478	0.432991
CANNY	3542	4638	2285	1257	2353	0.387617	0.213232	0.399152
bear_7.pgm								
SOBEL	5994	3302	635	5359	2667	0.0733172	0.618751	0.307932
LAPLACE	10184	3302	1336	8848	1966	0.109959	0.72823	0.161811
CANNY	4151	3302	1365	2786	1937	0.224212	0.457622	0.318167
turtle.pgm								
SOBEL	4662	4217	1927	2735	2290	0.277186	0.393412	0.329402
LAPLACE	2941	4217	1804	1137	2413	0.336944	0.212365	0.450691
CANNY	5299	4217	2749	2550	1468	0.406236	0.376829	0.216935
bear_5.pgm								
SOBEL	1867	2846	695	1172	2151	0.172972	0.291687	0.535341
LAPLACE	3207	2846	744	2463	2102	0.140139	0.463929	0.395931
CANNY	2331	2846	1269	1062	1577	0.324719	0.27175	0.403531
elephants.pgm								
SOBEL	5573	6221	2873	2700	3348	0.322049	0.302657	0.375294
LAPLACE	5125	6221	3419	1706	2802	0.431311	0.215214	0.353475
CANNY	6469	6221	4346	2123	1875	0.520853	0.254434	0.224712
golfcart.pgm								
SOBEL	6900	11377	4493	2407	6884	0.325958	0.174623	0.49942
LAPLACE	9836	11377	6639	3197	4738	0.455537	0.219363	0.325099
CANNY	7933	11377	7002	931	4375	0.568898	0.0756419	0.35546

Si on prend toutes les données on peut sortir les moyennes suivantes :

Moyenne de performance :

SOBEL	21%
LAPLACE	24,5%
CANNY	39,999%

Moyenne de taux de faux positifs :

SOBEL	39,12%
LAPLACE	40,01%
CANNY	25,45%

Moyenne de taux de faux négatifs :

SOBEL	39,63%
LAPLACE	35,35%
CANNY	34,54%

Interprétation des résultats

D'après les résultats, on peut déduire :

Pour Sobel, celui-ci est assez correct en termes de performance mais il est le moins performant parmi les 3 méthodes. Le taux de performance chute rapidement lorsqu'on applique un blur (surtout quand il est élevé) car le taux de faux positif monte très vite.

Pour Laplace, on remarque que si on doit augmenter le blur pour avoir un meilleur résultat (ce qui est arrivé dans les tests) il aura un moins bon résultat que Sobel. Notamment dû à la forte hausse du taux de faux négative (environ 70% parfois). En moyenne Laplace sera quasiment équivalent Sobel.

Pour Canny est la méthode la plus rentable parmi ces trois méthodes, il a des performances supérieures aux deux autres, de plus il a le taux de faux positifs et le taux de faux négatif le plus bas.