Vision par Ordinateur - Compte-rendu TP2

##### Léo HOUMMADY – leo.hoummady@student.ecp.fr

# 1 – Détection de contour

Dans cette partie, nous essayerons d’implémenter différentes méthodes de détection de contour, de les comparer entre elles, mais également de les comparer avec des détections de contour réalisées par des humains.

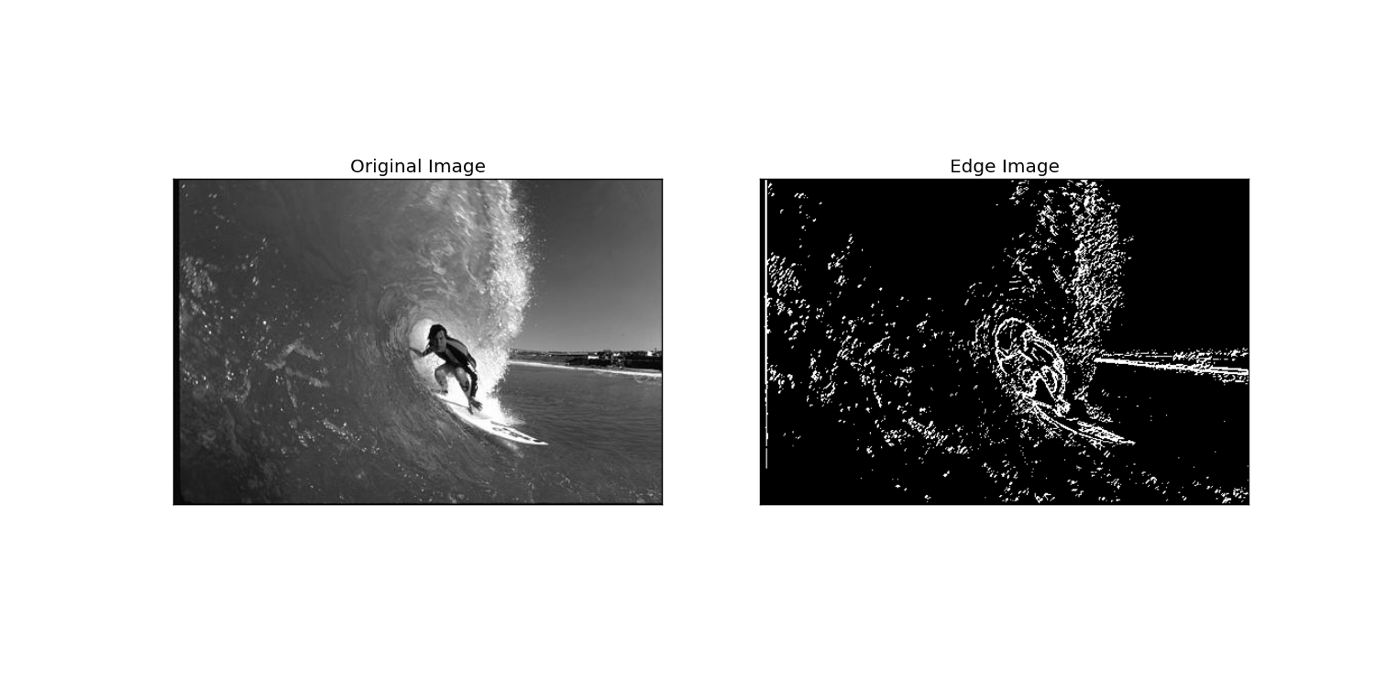
Dans un premier temps, voici un rapide retour sur les différentes méthodes utilisées.

### Détecteur de Sobel

L’une des méthodes vues en cours est le détecteur de Sobel. Comme la plupart des détecteurs, celui-ci est basé sur le calcul des gradients de l’image en chaque point. La méthode discrète de Sobel se base sur la multiplication de la matrice de l’intensité autour du pixel voulu par les matrices suivantes, représentant le « mélange » entre des filtres dérivés selon x et y, et des filtres gaussiens qui rajoutent de l’importance aux pixels les plus proches. Notons que la somme des coefficients des approximations vaut 1 de façon à ne pas introduire de biais sur l’intensité. Ces transformations vont en quelque sorte « concentrer » l’intensité sur les contours.



Concrètement, pour réaliser un détecteur de contour, nous avons ajouté les filtres selon les axes x et y. Ce détecteur a été implémenté en utilisant les fonctions d’origines de la librairie OpenCV.



### Détecteur Laplacien

Une autre méthode étudiée est l’utilisation du Laplacien. Le principe est assez similaire et se base cette fois-ci sur la dérivée seconde de l’intensité. De façon discrète, la matrice de Laplacien est implémentée par le produit des matrices d’intensité du contour du pixel avec les matrices suivantes :



Encore une fois, ce détecteur a été implémenté avec les fonctions d’OpenCV



### Détecteur de Canny

La méthode de Canny a été également utilisée ici. Concrètement, elle utilise un filtre gaussien, puis les matrices de dérivation selon les deux axes pour déterminer la force et l’angle du gradient. Enfin, une hystérésis est appliquée afin de lisser les contours les plus importants.



Là encore, nous avons utilisé le détecteur implémenté d’origine dans la librairie OpenCV.

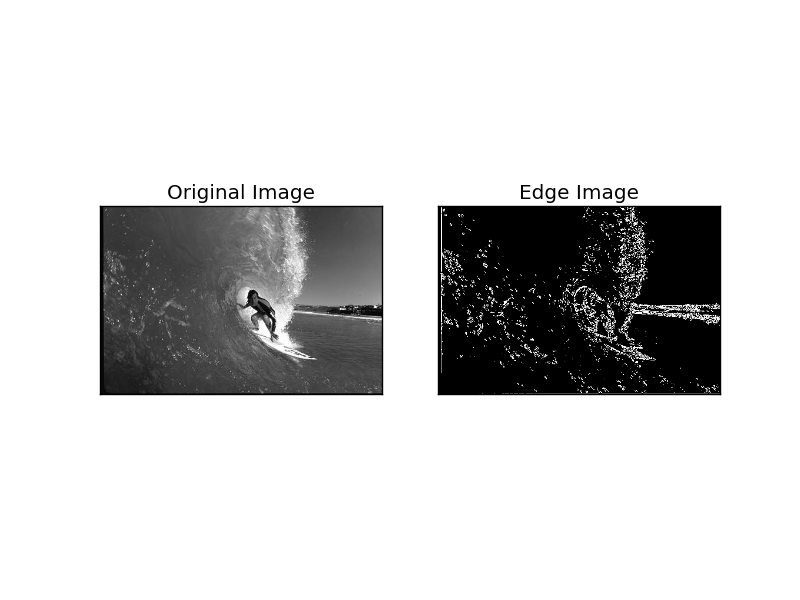


#### Détecteur de Prewitt :

Ce détecteur est assez proche de celui de Sobel. Concrètement, il fonctionne sur le principe de la détection du gradient selon les deux axes majeurs, en combinaison avec un filtre moyenneur. D’un point de vue discret, cela correspond à la multiplication des matrices d’intensité autour du pixel avec les matrices suivantes :



Nous avons conçu un algorithme qui permet d’implémenter cet algorithme. Les résultats sont a priori assez satisfaisants.



## – Mesure de performance des détecteurs

Nous avons suivi les indications qui nous encourageaient à mesurer la performance des différents détecteurs de façon quantitative. Pour toutes les 10 images de la base de donnée Berkeley, nous avons déterminé des valeurs de faux positifs, de faux négatifs, de vrais négatifs et de vrais positifs pour en déduire 3 indicateurs :

* P = contours-corrects / (contours-corrects + faux-positifs)
* TFP = faux-positifs / (faux-positifs + vrai-négatifs)
* TFN = faux-négatifs / (contours-corrects +faux-négatifs)

La valeur de chaque indicateur nous donnera des informations sur la façon dont un détecteur d’arrête comprend l’image.

|  |  |  |
| --- | --- | --- |
|  | Faible | Elevé |
| P | Trop de faux positifs, les contours ne sont pas détectés au bon endroit | Parmi les pixels que notre détecteur comprend comme contour, la majorité font en effet parti de contours sur l’image référence. |
| TFP | Beaucoup plus de vrais négatifs que de faux positifs. On détecte bien les non contours au bon endroit. | Beaucoup de faux positifs. Notre détecteur détecte trop de contours par rapport à l’image vérité. |
| TFN | Beaucoup de contours corrects par rapport aux faux négatifs. Notre détecteur détecte bien les contours au bon endroit, et fait peu d’erreur sur les non contours. | Beaucoup de faux négatifs par rapport au contours corrects. Notre détecteur détecte trop peu de contours lorsqu’il le devrait. |

Nous avons mesuré la performance des différents détecteurs sur l’ensemble des images de la base de donnée. En voici les résultats :

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Canny |  |  | Sobel |  |  | Laplacian |  |  | Prewitt |  |  |
|  | P1 | TFP1 | TFN1 | P2 | TFP2 | TFN2 | P3 | TFP3 | TFN3 | P4 | TFP4 | TFN4 |
| contour/101085.jpg | 36% | 8% | 82% | 28% | 21% | 68% | 8% | 31% | 90% | 29% | 21% | 66% |
| contour/119082.jpg | 21% | 11% | 95% | 25% | 20% | 88% | 12% | 34% | 92% | 28% | 26% | 83% |
| contour/156065.jpg | 21% | 7% | 87% | 22% | 18% | 63% | 4% | 24% | 93% | 21% | 18% | 66% |
| contour/175032.jpg | 6% | 22% | 86% | 9% | 38% | 59% | 0% | 38% | 99% | 9% | 38% | 60% |
| contour/220075.jpg | 39% | 4% | 83% | 34% | 8% | 70% | 5% | 18% | 94% | 33% | 11% | 62% |
| contour/296007.jpg | 7% | 1% | 100% | 9% | 7% | 99% | 1% | 14% | 100% | 7% | 7% | 99% |
| contour/300091.jpg | 35% | 2% | 87% | 32% | 5% | 71% | 9% | 11% | 87% | 30% | 5% | 75% |
| contour/43074.jpg | 1% | 2% | 92% | 2% | 7% | 66% | 0% | 10% | 99% | 2% | 7% | 66% |
| contour/76053.jpg | 21% | 2% | 97% | 12% | 9% | 93% | 2% | 20% | 98% | 11% | 12% | 92% |
| contour/78004.jpg | 18% | 8% | 96% | 20% | 14% | 92% | 9% | 23% | 95% | 20% | 18% | 90% |

Certaines des images sont excessivement difficiles à prévoir. Il serait nécessaire de changer quelques réglages dans les détecteurs afin d’améliorer la précision. Par exemple, pari les pistes d’amélioration, il serait possible

* De réduire les faux négatifs du détecteur de Canny. Il faudrait par exemple épaissir le trait ou seuiller les images références avec un seuil plus faible
* De même pour les détecteurs de Sobel ou Prewitt. Le détecteur de Laplace a un fort taux de faux positifs, et il serait intéressant de diminuer sa valeur de seuillage.