

# Rapport de projet - DéTECTEUR de gradient avec OpenCV

Prénom et nom: Felipe Mateus Falcão Barreto

Cours: Analyse d'Images

## Résumé:

Ce projet vise à comparer l'utilisation de trois détecteurs de gradient d'image implémentés avec OpenCV sur un ensemble d'images comportant des contours de référence: Sobel, Laplace et Canny.

À cette fin, des paramètres et des métriques sont utilisés pour évaluer chaque modèle.

## Objectifs:

- Permettant de choisir le détecteur
- Permettant de régler les paramètres associés
- Capable de traiter « un lot d'images »

## Les détecteurs de gradient :

Sobel:

Sobel estime le gradient à partir des dérivées par rapport à x et y. La combinaison de ces dérivées permet d'obtenir le gradient final. Des techniques de lissage, comme le gaussian blur, peuvent être appliquées pour une meilleure robustesse.

De plus, un threshold est utilisé pour extraire un masque binaire de contours, facilitant ainsi la reconnaissance des bords. La valeur du threshold est inversement proportionnelle au nombre de contours, ce qui peut entraîner un excès ou une discontinuité de ces derniers.

Laplace:

Détecteur de contours basé sur la dérivée seconde de l'image, mettant en évidence les zones de changement de courbure. Il ne fournit pas l'orientation du gradient, mais signale les points de changement de courbure grâce à la méthode de zero-crossings.

Un filtre gaussien (DoG) a été appliqué pour atténuer le bruit et faciliter la localisation des contours.

Canny:

Basée sur l'opérateur de Sobel, cette méthode combine lissage, calcul de gradient et utilisation de thresholds pour produire des contours fins et continus. Son objectif est de trouver un threshold ni trop élevé (peu de courbes) ni trop bas (beaucoup de courbes).

Grandeurs:

Cinq grandeurs ont été utilisés pour évaluer les modèles :

- contours\_détectés : nombre de pixels contours dans l'image calculée
  - contours\_référence : nombre de pixels contours dans l'image de référence
  - contours\_corrects : nombre de pixels contours correctement détectés dans l'image calculée
- selon l'image de référence (aussi égal à contours\_détectés  $\cap$  contours\_référence)

- faux\_positifs : nombre de pixels détectés comme contours, mais non contours dans l'image de référence (aussi égal à contours\_détectés – contours\_corrects)
- faux\_négatifs : nombre de pixels contours non détectés dans l'image calculée, mais contour dans l'image de référence (aussi égal à contours\_référence – contours\_corrects)

Mesures:

En fonction des grandeurs, trois mesures ont été calculées pour évaluer la précision de la prédiction du gradient.

- La performance :

$$P = \text{contours\_corrects} / (\text{contours\_corrects} + \text{faux\_positifs} + \text{faux\_négatifs})$$

- Le taux de faux positifs:

$$\text{TFP} = \text{faux\_positifs} / (\text{contours\_corrects} + \text{faux\_positifs} + \text{faux\_négatifs})$$

- Le taux de faux négatifs:

$$\text{TFN} = \text{faux\_négatifs} / (\text{contours\_corrects} + \text{faux\_positifs} + \text{faux\_négatifs})$$

Fonctions auxiliaires:

- Permettant de choisir le détecteur (main)
- Permettant de régler les paramètres associés
- Capable de traiter « un lot d'images »

Explications des fonctions d'OpenCV utilisées:

- cvtColor: les fonctions de dégradé sont appliquées en niveaux de gris, donc la conversion des images à cette échelle devient également très utile.
- Sobel: elle applique une convolution dérivée à l'image à l'aide du filtre Kernel, vous permettant de choisir si elle sera horizontale, verticale ou mixte.
- GaussianBlur: elle réduit et lisse le bruit. Très utile à appliquer avant la modélisation pour une meilleure prédiction des contours.
- Canny: elle recherche un seuil qui ne soit ni trop élevé (trop peu de courbes) ni trop bas (trop de courbes) pour appliquer les courbes de l'image.
- Threshold: elle permet d'obtenir une carte binaire de l'image. Utile pour le traçage des contours.
- Laplacian: elle calcule le laplacien de l'image. Utilise les mêmes fonctions de base que Sobel. Les valeurs d'échelle peuvent aller de 0 à 255.
- getZeroCrossings: elle génère une carte binaire des passages par zéro. Un threshold de 0 est utilisé. Concrètement, toute valeur supérieure à 0 devient blanche, et toute valeur inférieure ou égale à 0 devient noire, créant ainsi des contrastes marqués.

- GaussianBlur: dépend du paramètre sigma, qui contrôle la largeur du filtre. La soustraction de deux filtres gaussiens de bandes passantes différentes sera composée des hautes fréquences préservées par l'un des filtres et non par l'autre. Cette opération est appelée différence de gaussiennes (DoG).

Résultats:

Sobel

**SOBEL\_THRESH = 225**

Mesures par contour:

contours\_detectes = 246092

contours\_reference = 255206

contours\_corrects = 244125

faux\_positifs = 1967

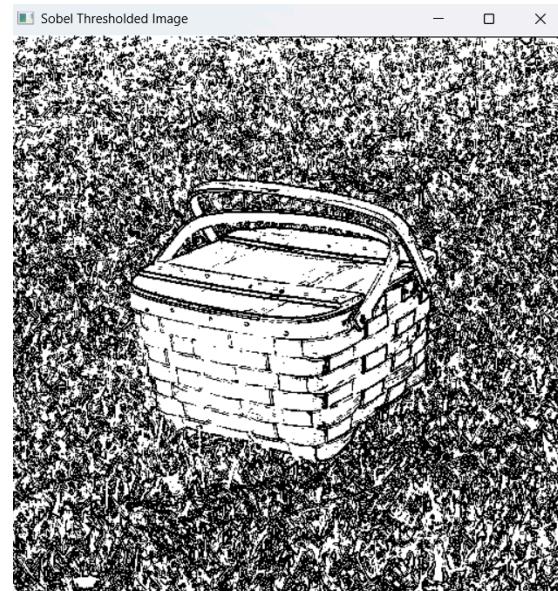
faux\_negatifs = 11081

Mesures de performances:

Performance: 0.949264

taux de faux positifs: 0.00764855

taux de faux negatifs: 0.0430877



Laplace:

**LAPLACE\_GAUSS\_MIN = 2.0**

**LAPLACE\_GAUSS\_MAX = 2.2**

Mesures par contour:

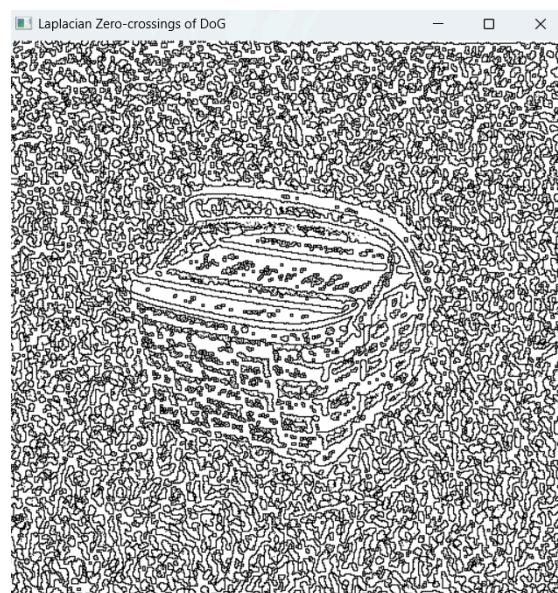
contours\_detectes = 217392

contours\_reference = 255206

contours\_corrects = 214641

faux\_positifs = 2751

faux\_negatifs = 40565

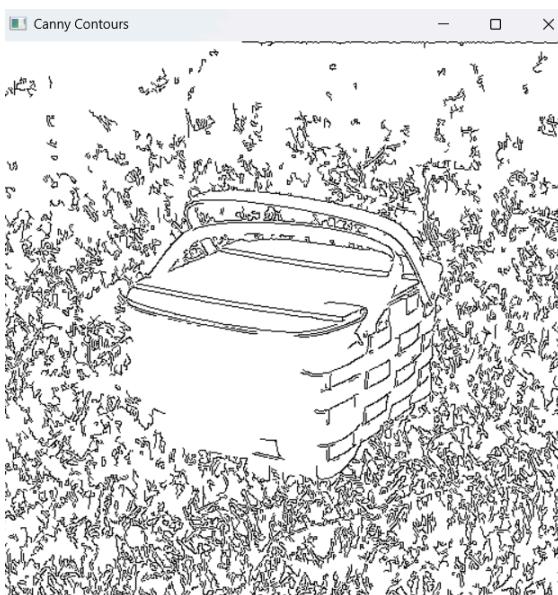


Mesures de performances:

Performance: 0.832081

taux de faux positifs: 0.0106646

taux de faux negatifs: 0.157255



Canny:

Cas 1:

**CANNY\_LOW\_THRESH = 125**

**CANNY\_HIGH\_THRESH = 350**

Mesures par contour:

contours\_detectes = 115740

contours\_reference = 255206

contours\_corrects = 113297

faux\_positifs = 2443

faux\_négatifs = 141909

Mesures de performances:

Performance: 0.439734

taux de faux positifs: 0.00948189

taux de faux negatifs: 0.550784

Cas 2:

CANNY\_LOW\_THRESH = 0

CANNY\_HIGH\_THRESH = 350

Mesures par contour:

contours\_detectes = 188003

contours\_reference = 255206

contours\_corrects = 184873

faux\_positifs = 3130

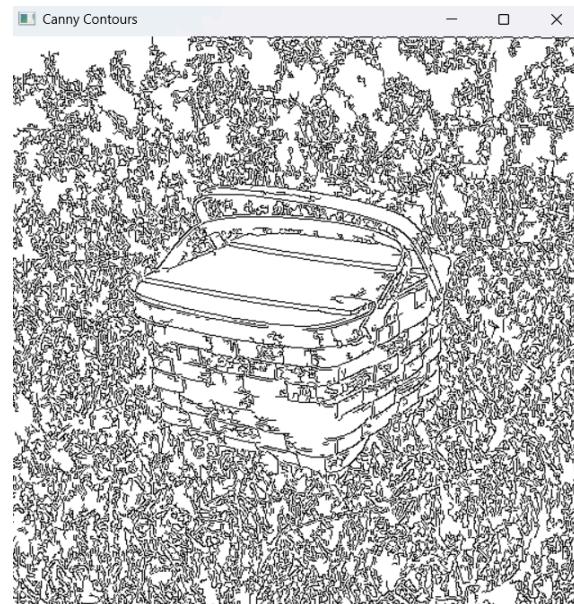
faux\_négatifs = 70333

Mesures de performances:

Performance: 0.71563

taux de faux positifs: 0.012116

taux de faux negatifs: 0.272254



Détecteurs	Performance	Taux de Faux Positifs	Taux de Faux Négatifs
Sobel	94,93%	0,76%	4,31%
Laplace	83,21%	1,07%	15,72%
Canny	71,56%	1,21%	27,23%

Conclusion:

Grâce à l'utilisation d'un seul threshold et à l'absence de dérivées secondes, l'algorithme de Sobel présente une implémentation plus simple et des performances supérieures (94,93%). L'algorithme de Laplace s'en approche, mais son image, plus complexe en raison d'un excès de contours, est plus difficile à visualiser. Ses performances restent toutefois acceptables (83,21%).

Enfin, l'algorithme de Canny, du fait de la différence de seuils, affiche des performances (43,97%) et des taux de faux négatifs (55,08%) très faibles. Même en fixant le threshold minimal à 0, sa précision demeure la plus faible (71,56% de performance et 27,23% des taux de faux négatifs).

Par conséquent, l'algorithme de Sobel est considéré comme le plus performant.