Université Joseph Fourier

GRENOBLE

UFR IM²AG

Analyse et Traitement d'image

LEFEVRE Henry - M2 MIA
SEGURET Aymeric - M2 MIA

Rapport de Travaux Pratiques

Filtrage

Il existe différentes manières de réduire le bruit présent sur une image : Filtre médian, filtre moyenneur etc … Dans notre cas il nous a été demandé de réaliser les filtres suivant :

1. Le Filtre Gaussien

$$h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

L'application d'un filtre gaussien correspond au passage du noyau présenté ci-dessus. Ce noyau diminue le bruit mais applique un flou sur l'ensemble de l'image.

L'application d'un tel filtre sur l'image « house » ne présente que peu d'intérêt, nous avons donc décidé de présenter les résultats de l'algorithme sur une image bruitée.

2. Le Filtre médian

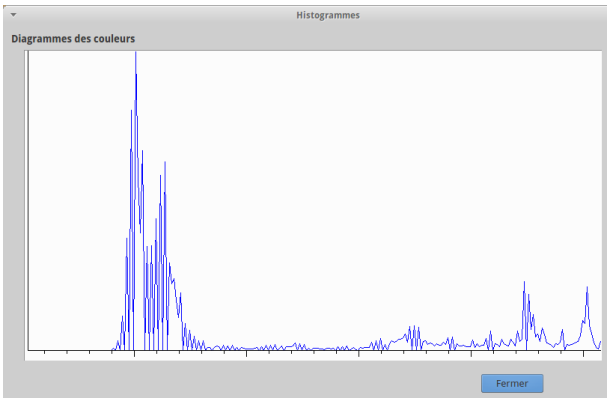
Comme le filtre Gaussien, le but d'un filtre médian est de supprimer le bruit présent sur l'image. Et comme précédemment nous présentons nos résultats sur l'image « house » et sur notre image bruitée.

3. Le Filtre adaptatif

Comme les deux filtres précédents, le filtre adaptatif « nettoie » l'image originale. Pour cela il floute l'image mais conserve les bords. Toujours présenté avec les mêmes images.

Calcul et étalement d'histogramme

L'histogramme permet de représenter la distribution des intensités lumineuses. Nous avons choisi de le représenter comme suit avec les trois composantes RGB, qui dans le cas de nos images en niveau de gris sont identiques.

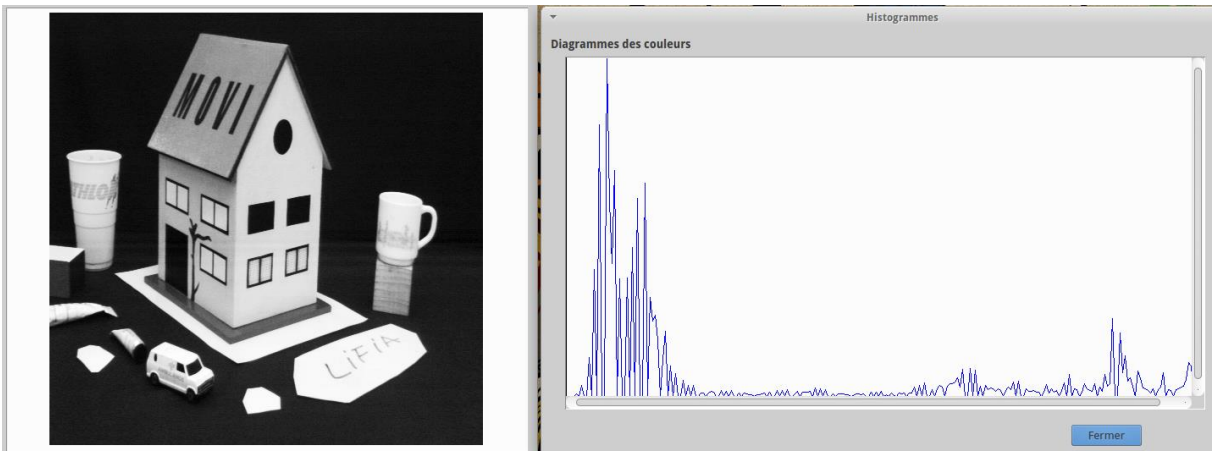


1. Etalement d'histogramme

Le recadrage d'un histogramme se fait entre 0 et 255. Chaque pixel subit une transformation égale à la formule suivante :

$$y = 255 * \frac{x - histo\_min}{histo\_\max - histo\_min}$$

On obtient les résultats suivants :



2. Inversion d'histogramme

Chaque pixel subit une transformation linéaire de façon à ce que la nouvelle intensité lumineuse soit égale à :

$$y = x - 255$$

Ce qui correspond à ce que l'on appelle traditionellement un « negatif ».

Après inversion :

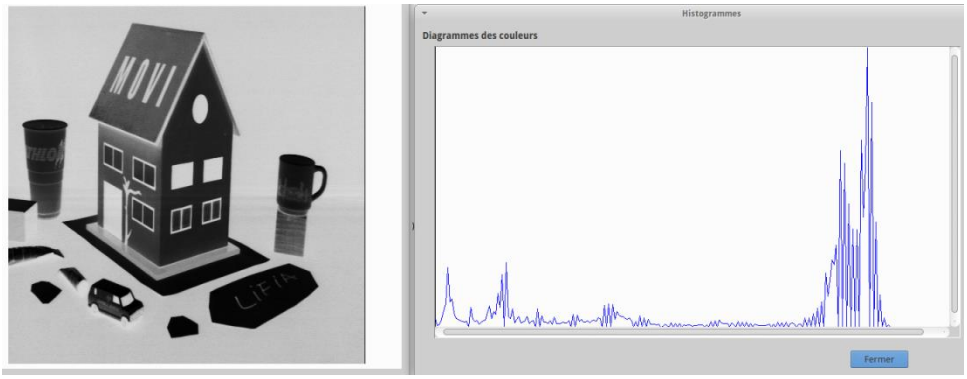


Image du gradient

A partir d'une image nous cherchons à calculer le gradient en X, le gradient en Y et à afficher le module du gradient. L'image calculée représente l'intensité de la dérivée en chaque point de l'image. Les opérateurs de Sobel sont utilisés pour calculer les 2 gradients.

gradient X $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ **gradient Y** $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$

Puis la norme euclidienne est utilisée pour calculer la norme du gradient.

$G = Gx^2 + Gy^2$

L'information extraite avec cet algorithme permet de détecter les bords d'une image. C'est-à-dire les zones des transitions lors du passage d'un objet à un autre.

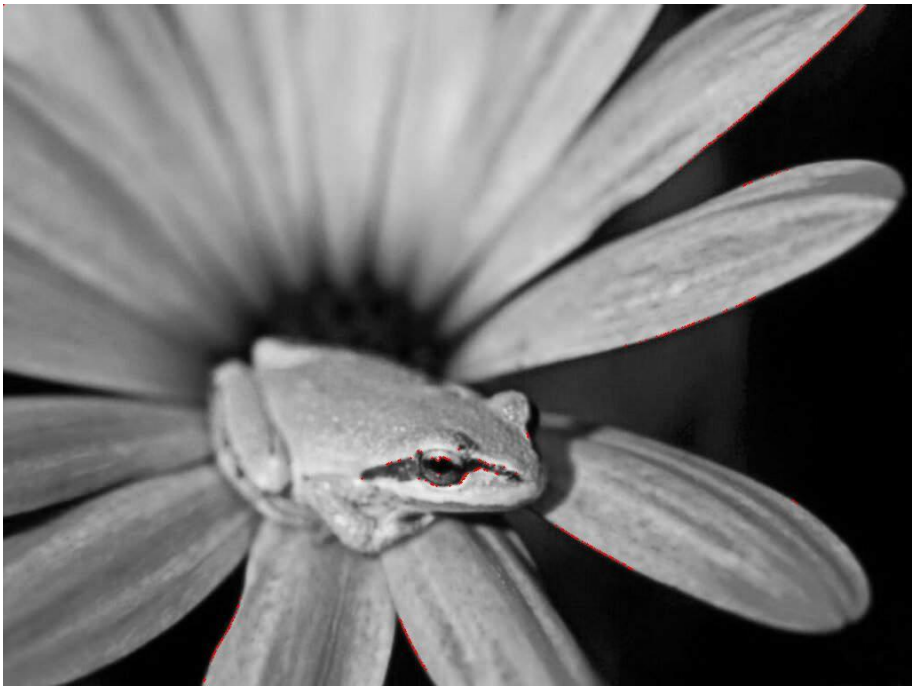
Remarque : pour le calcul de Gx et Gy, le noyau est accumulé deux fois sur chaque pixel. Une fois avec le noyau normal et une seconde fois avec le noyau retourné par rapport à « l'axe des 0 ». De cette manière, pour une forme donnée, les bords horizontaux et verticaux ne dépendront pas de la direction du gradient (i.e. un bord n'est pas privilégié par rapport à un autre).

Détection des points d'intérêt

La détection de zones d'intérêt d'une image consiste à mettre en évidence des zones de cette image jugées « intéressantes » pour l'analyse, c'est-à-dire présentant des propriétés locales remarquables. De telles zones peuvent apparaître, selon la méthode utilisée, sous la forme de points, de courbes continues, ou encore de régions connexes rectangulaires ou non et qui constituent le résultat de la détection. *Wikipédia*

Dans le cadre de notre TP il nous a été demandé d'implémenter le détecteur de Harris. Celui-ci repose sur un principe simple : les coins sont les points de l'image où le contour change brutalement de direction. Il s'agit de points particulièrement stables, et donc intéressants pour la répétabilité.

Voici quelques résultats obtenus :



Les images ci-dessus présentent nos résultats après sélection des 250 meilleurs points des images. L'algorithme du détecteur de Harris que nous avons implémenté est légèrement différent de celui qui nous a été donné dans le sujet du TP.

Nous commençons par calculer les gradients en x et en y, puis x^2, y^2 et xy. Une fois ces calculs effectués on construit la matrice suivante :

$$C = \begin{matrix} Ix^2 & Ixy \\ Ixy & Iy^2 \end{matrix}$$

Nous appliquons la formule suivante: déterminant(C) ^{2} – (alpha*trace(C))^2

Après un seuillage sur les résultats, on extrait les maxima locaux, nous stockons les valeurs ainsi obtenues dans un vecteur que l'on trie à l'aide d'un quicksort. Et enfin nous affichons les meilleurs points (ici nous avons fixé un maximum de 250 points, une valeur totalement arbitraire).

Nous avons ici choisi de comparer les deux images analysées précédemment.
Pour l'affichage nous mettons en relation le meilleur point de l'image de gauche avec le point obtenant le meilleur score dans l'image de droite.
Nous pouvons voir ici que les points correspondent bien (ceci apparait avec le segment rouge).

Cependant si on choisit d'afficher les 25 meilleurs points de l'image de gauche, les concordances ne sont pas parfaites. En effet on voit les limites de la mise en correspondance SSD. Certains points mis en relation sont de notre point de vu totalement différents et cela est dû à la transformation effectuée sur l'image.

Faisons donc la même comparaison que ci-dessus en inversant les images, on cherche donc dans l'image de droite la meilleure correspondance avec le meilleur point de l'image de gauche.

On voit que l'on obtient pour l'image de gauche le même point que précédemment mais pas la même correspondance.

Cela prouve les limites de l'algorithme, on peut cependant imaginer un nouvel algorithme à partir de ces résultats. On peut penser que si on effectue les mises en relations des points dans les deux sens (a vers b et b vers a), on peut considérer que si les mêmes points sont mis en correspondance dans les deux cas, on les considère comme des points corrects.

On peut aussi utiliser les scores obtenus dans les deux sens, et dire que le point correspondant est le score qui obtient le meilleur cumul.

C'est à dire effectuer les comparaisons de a vers b puis de b vers a, d'additionner les « scores » obtenus et de garder le plus intéressant. Mais comme précédemment on obtient juste une chance plus importante que la relation soit correcte. Mais une transformation importante de l'image initiale limite énormément cette concordance.
Notre algorithme ne peut donc pas être parfait, il peut juste présenter une chance de relation.

Les préliminaires de cette partie ont été fructueux. Cependant la mise en relations comme demandé à la fin du TP, avec l'algorithme présenté ci-dessus n'a pu être totalement effectué. Cela n'a été fait que pour un point de l'image, on obtient donc le même point dans les deux sens.

Le résultat obtenus dans l'image ci-dessus se retrouve alors dans les deux sens.

Segmentation avec les K-moyennes

Le principe ici est de trouver l'appartenance de chaque pixel d'une image à son cluster le plus proche. Un cluster représente un groupe sous lequel sera identifié chaque pixel qui y appartient. Et ceux-ci sont définis par un centroïde (i.e. une couleur qui qualifie son cluster) initialisé aléatoirement au début.

L'espace de calcul s'effectue en milieu RGB. L'algorithme se déroule en 4 passes :

1. initialisation de K centroïdes
2. parcours de chaque pixel de l'image et on lui étiquette le cluster pour lequel il est le plus proche (distance calculée par rapport au centroïde du cluster)
3. déplacement des centroïdes en moyennant sur l'ensemble des pixels qui lui sont rattaché
4. on itère jusqu'à ce que la distance de chaque nouveau centroïde avec les anciens est en dessous d'un certain seuil

Les images ci-dessus présentent les résultats obtenus avec 4 clusters de segmentations. Les 4 centroïdes initiaux sont représentés par une croix rouge sur la première image. Après segmentation, on obtient l'image résultante.

Par conséquent on faisant varier le nombre de centroïdes on obtient des résultats différents, ci-dessous quelques exemples que l'on peut obtenir.

A noter que le choix des centroïdes dès la première itération change le résultat final. La répartition de chaque pixel dans les différents cluster se trouvant affecté dès le début.

Choix des centroïdes : L'initialisation se fait ici de manière aléatoire parmi l'ensemble des couleurs de l'espace RGB. C'est-à-dire que les K couleurs ne sont pas choisies aléatoirement parmi les pixels de l'image. De cette manière on pensait éliminer le cas où 2 centroïdes se trouvent très proches à l'initialisation. Dans le cas où ça arriverait, on avait imaginé que le résultat final allait devenir instable (i.e. le résultat varie grandement pour des faibles décalages des centroïdes proches). En effet un voisinage de couleur se serait potentiellement trouvé dans deux clusters différents.

Plus généralement, est-il alors possible de mesurer qualitativement cette initialisation ? Une idée serait par exemple de faire en sorte qu'il y ait une distance minimale au-delà de laquelle les centroïdes sont tous distants 2 à 2 deux.
Une autre idée serait que pour chaque centroïde *ci*, on définit une distance *di* qui est égale à la distance minimale entre *ci* et toutes les couleurs de l'image. Et on s'assure que *di* soit inférieure à une constante. On assurait ainsi que les centroïdes correspondent un minimum aux pixels de l'image.

Cependant sur quelques jeux de test effectués avec une initialisation manuelle et des centroïdes volontairement proches les uns des autres on s'est aperçu que les résultats étaient équivalents.