ENSSAT
LANNION

PROJET NOTÉ

Détection des contours d'une image : le filtre de Canny

colinleverger [at] gmail [dot] com

Colin LEVERGER - ENSSAT Informatique, Multimédia et Réseaux
Promotion 2017

Destinataire : Benoit VOZEL

15 avril 2016

Introduction

Dans le cadre de la formation Informatique, Multimédia et Réseaux dispensé à l'ENSSAT de Lannion, nous avons étudié le traitement d'image en seconde année. Cette matière, qui s'inscrit dans la composante "multimédia" de notre cursus, nous a permis d'appréhender les concepts clés de la manipulation d'images par programmation. Nous avons notamment travaillé à la création de filtres gaussiens, à la création de matrices de gradient, etc.

Pour pratiquer et appréhender au mieux les concepts évoqués en CM, nous avons bien évidemment travaillé sur un TP noté. Il s'agissait d'écrire avec Scilab un programme permettant de trouver et d'isoler les contours des objets présents sur une image, de manière automatique. J'ai choisi d'utiliser un grand classique du traitement d'image, à savoir Lena, qui n'est autre qu'une pinup des années 70 qui se trouvait au bon endroit au bon moment (à savoir, dans les mains d'un chercheur dans le domaine de l'imagerie numérique lorsqu'il cherchait une image pour ses traitements).

Ce compte rendu présente la démarche lors de la réalisation et du codage de ce TP. La structure du sujet sera suivie, car nous allons expliquer les méthodes et fonctions dans l'ordre chronologique de codage. Les fonctions et méthodes codées seront en effet très souvent réutilisées pour la suite du TP.

Le code associé à ce compte rendu pourra être trouvé dans l'archive jointe à ce rapport. Très peu de code sera directement exposé dans le rapport, mais les structures complexes et les choix d'implémentation des algorithmes y seront clairement explicités.

Table des matières

Partie 1

Préliminaires

Afin de travailler avec Scilab, quelques étapes préliminaires ont étés effectuées.

Tout d'abord, le choix de l'image était important pour pouvoir développer rapidement les premières fonctions. L'exécution des algorithmes prend un temps non négligeable, et il était important de prendre une petite image pour le début des opérations : pas plus de 64x64 pixels. Une image de 64x64 pixels représente par exemple une matrice de 64 par 64. Évidemment, plus l'image est précise et grande (plus le nombre de pixels est important), plus la taille de la matrice à traiter est grande, et plus le traitement prend du temps. [1]

Le choix s'est donc porté sur deux images pour ce TP : une petite et une moyenne. À terme, l'image en entrée du programme fera une taille d'environ 400x400 pixels.

La seconde chose à faire avant de se lancer dans le codage était l'installation du module Scilab nécessaire pour utiliser les fonctions basiques de chargement d'image. Il s'agit en effet de pouvoir transformer une image en entrée en matrice de valeurs. [2]

1. L'algorithme a été testé avec une image 6000x6000 afin de le benchmarquer. L'hypothèse émise était que cet algorithme est linéaire ; si traiter une image de 60x60 prend 20 secondes, traiter une image de 6000x6000 prendrait simplement 100 fois plus de temps, soit 2000 secondes = 33,3 minutes. Malheureusement, le test n'a pas pu se terminer : erreur de capacité pour l'ordinateur...

2. L'installation du module dans Scilab sur Mac OS est loin d'être triviale... L'utilisation du logiciel sous Windows et préférée.

Partie 2

Première étape : le rehaussement de Canny

La première étape de l'algorithme peut se décomposer en deux sous parties. Il s'agit tout d'abord de lisser l'image, afin d'en retirer les impuretés, et de calculer dans un second temps la norme du gradient et l'angle de la normale au gradient pour chaque pixel de l'image lissée.

2.1 Le filtrage de l'image

La présence d'impuretés sur une image est courante, et même très probable. Un exemple d'impuretés pourra être vu en figure 2.1 en page 2 [Phi]. Dans ce cas précis, il d'agit du bruit blanc gaussien, qui provoque une altération nettement visible de l'image.

On sent bien que la détection de contours demandée pourrait potentiellement être faussée par le bruit présent sur l'image : il est donc important de filtrer l'image, afin de supprimer le bruit et de pouvoir effectuer les traitements en limitant le biais au maximum.

FIGURE 2.1 – Le bruit blanc gaussien sur l'image Lena

Nous allons donc "lisser" l'image, en lui appliquant un filtre gaussien. Le filtre est lui même une matrice, et il faut l'appliquer en effectuant des produits de convolution entre notre image et le filtre. Typiquement, une boucle va parcourir toutes les valeurs de notre matrice image de base et effectuer un traitement (filtrage) pour chacune de ces valeurs. Nous pourrons trouver un schéma de l'application de ce filtre en figure 2.2 en page 3 [VOZ16].

Le principe est le suivant : nous allons sommer la valeur de tout les voisins de notre pixel en les multipliant par les facteurs présents dans la matrice de filtrage, puis normaliser avant d'affecter la valeur au nouveau pixel. Une nouvelle matrice de taille égale à l'image sera initialisée et remplie au fil du traitement.

La taille filtre que nous appliquons sur notre image peut être plus ou moins grande. Évidemment,

FIGURE 2.2 – Application du filtre par produit de convolution

plus elle est grande plus le traitement est coûteux et plus il prend du temps. On note aussi que plus la taille du filtre gaussien est importante, plus l'image est floutée par le traitement. Deux filtres ont étés utilisés pendant les manipulations (voir figure 2.3 en page 3). Le premier filtre va lisser l'image convenablement, mais moins efficacement que le second.

On note la présence de coefficients devant les filtres : ceux-ci représentent la normalisation des valeurs. On va en effet diviser chaque pixel par la somme totale des éléments présents dans le filtre.

1	2	1			
2	4	2	x 1/16		
1	2	1			
2	4	5	4	2	
4	9	12	9	4	
5	12	15	12	5	x 1/159
4	9	12	9	4	
2	4	5	4	2	

FIGURE 2.3 – Les filtres utilisés pour expérimenter

La fonction d'application du filtre sera utilisée plusieurs fois le long de ce TP : c'est pour ça qu'elle a été codée de la façon la plus générique possible.

Partie 3

Deuxième étape : Suppression des non-maximums

Partie 4

Troisième étape : seuillage par hystérésis

Partie 5

Dernière étape : trouver le seuil automatiquement

Partie 6

Conclusion

Table des figures

Bibliographie

[Phi] Philippeau, Xavier : *Filtre Mean Shift.* http://xphilipp.developpez.com/articles/meanshift/

[VOZ16] VOZEL, Benoit : *CM traitement image*, 2016