Aquisition et analyse d'image

Hadrien Croubois Nicolas Lourdeau

Philosophie d'utilisation

Dans le cadre de ce projet, les outils developpés ont toujours été pévu pour etre reutilisés. Ainsi le code se decompose tout naturelllement en deux parties, une librairie facilement compliable au format `.so` ou `.dll` fournit avec les fichier entête correspondant d'un coté et d'autre part un programme simple permettant d'appeler facillement les fonctionnalitées de la librairie.

Afin de donner differents exemples d'utilisation de la bibliothèque *lenactions*, deux applications sont fournis et donne ainsi differents exemple d'utilisation plus ou moins aux niveaux des fonctionnalités de la bibliotheque.

— **Lenaction** : programme simple, effectuant des appels a la bibliotheque a partir des arguments hard-codé dans le code du programme.
— **LenaSH** : un shell minimaliste interprettant des commandes ecrites dans un langage ad-hoc et proposant ainsi une interface haut niveau pour l'utilisateur.

Outils utlisés

Afin d'optimiser la portabilité de la bibliothèque, cette derniere ne depend d'aucun autre outil. Ecrite en C/C++ elle gere toutes les etapes du traitement d'images, du chargement de fichier au calcul de contours, en passant par la transition dynamique entre les espaces de couleur RGB et HSV.

Le programme **LenaSH** utilse les outils flex et bison pour la reconnaissance du langage ad-hoc developpé parallement a la librairie.

Afin de simplifier l'etape de compilation des differents composants du programme sur differentes architectures, nous utilisons l'outil CMake ainsi qu'un script `./generator`.

Fonctionnalités

De nombreux algorithmes sont actuellement deployés dans la bibliotheque a differents niveau :

— Pixel :
  — Convertion d'espace de couleur
  — Operateurs de fusion (quadratique, angle)
— Image :
  — Chargement/Sauvegarde au format `.ppm` / `.pgm`
  — Calcul de seuil
  — Composition avec un filtre
  — Assemblage de deux images
  — seuillage (local, global, histerisis)
  — affinage de contour
  — calcul de contour fermés

La mise en place de filtre de convolution standart permet par ailleur de calculer simplement les contours via les filtres de Prewitt, Sobel et Kirsch ainsi que d'appliquer un filtre moyenneur gaussien pour lisser l'image.

Algorithme

seuillage par histerisis

Cette operation de seuillage revient a calcul des composantes connexes pour le critere de luminosité ($> low$) dans l'image. Pour cela on utilise une structure de union find qui garanti un resultat rapide $\mathcal{O}(n.Ack^{-1}(n))$.

L'ajout d'un drapeau au niveau des composantes connexes permet de marquer les composantes dont un des elements verifie le critere de luminosité ($> high$).

On ne garde ensuite que les pixels appartenants a une composante connexe marqué.

Affinage des contours

Pour cette operation, on se base sur une image de contour obtenue a partir de l'operateur de melange `angle` appliqué a deux gradients.

L'angle decrivant localement le contour (azimut du gradient) est discrétisé et permet de parcourir localement la largeur du contour. En ne gardant que le pixel au centre de ce contour on arrive a garder un contour fidéle, peu bruité et limitant les trous.

Fermeture des contours

Pour cette operation, nous avons developpé un algorithme a vague proche de ceux utilisés en systeme distribué pour la communication sur des grilles de processeurs.

Ici chaque pixel est une entité pouvant etre dans 4 etats different :

— *Vide*
— *Champ*
— *Contour*
— *Ancre*

Au debut de l'algorithme, on part d'un contour affiné dont les pixels sont dans l'etat *Contour* tandit que le fond est dans l'etat *Vide*.

La premiere passe se charge de detecter les Ancres parmit des pixels du Contours, pour cela on detecte tout ceux qui ont soit aucun voisin (ancres d'adjacence 2) ainsi que ceux qui sont en bord de contour (soit un unique voisin, soit deux voisin collés) et qui sont des ancres d'adjacence 1.

Les ancres sont les points d'interet qu'il sagit de relié. Leur adjacence correspond au nombre de liaison à former pour faire partie d'un contour fermé.

A partir de la on applique un algorithme de diffusion pour repandre un champ autour des ancres, les pixels passant ainsi de l'etat *Vide* à l'etat *Champ*.

La jonction de champs provoquant la liaison des deux ancres et la diminution de leur adjacence. Si un champ rencontre un pixel d'un contour qui n'est pas dans la composante connexe de l'ancre associé au champ, il s'y accrochera, faisant par la meme occasion diminué l'adjacence de l'ancre dont il provient. Pour cette diffusion on utilise une file de priorité implementé a partir de tas ainsi qu'une distance basé sur BLUE (best linear unbiased evaluation) et prenant aussi en compte les variations successives du gradient le long du contour a retrouver

Cet algorithme donne de bon de resultats des lors que les la taille des trous est inferieur à la taille des composantes connexes à fermer.

Voir la section .

Conclusion

Apres de longue seances de recherche, nous sommes satisfait de la forme que prend notre bibliotheque. Ne nombreux outils precedements developpé dans le cadre de `LibImAMXX` (librairie graphique developpé par Hadrien Croubois en M1 et qui a servit de support) sont actuellement en cours d'ajout. Parmit ces outils on compte notament l'affichage des histogrammes avec `gnuplot`, la normalisation des espaces de couleurs ainsi que le tatouage.

Pour ce qui est des algorithmes developpés, nous sommes relativement content des resultats renvoyés par l'algorithme d'affinage.

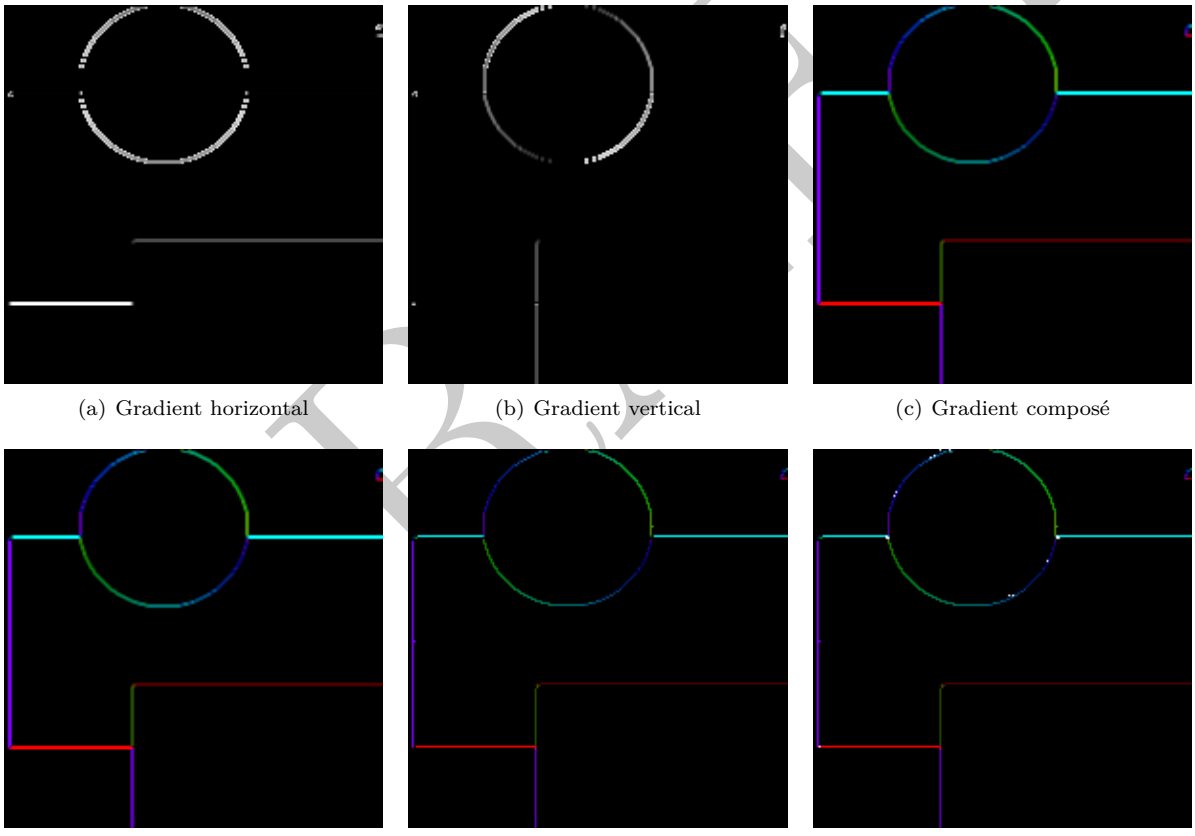
L'utilisation de champ dans un algorithme a vague nous semble une piste interressante pour la fermeture des contours. Beaucoup de travail reste cependant a faire, aussi bien pour ce qui est du choix de la fonction de distance que pour les heuristiques de creation des liens d'ancrage. Nous esperons pouvoir continuer ammeliorer notre algorithme afin de le rendre plus efficace, notament en s'adaptant dynamiquement aux propriété de l'image.

Le code du programme est disponible sous licence GNU GPL v3 sur GitHub : `https://github.com/Amxx/Lenactions`

Annexes - Résultats

(a) Forme geometriques (b) Lena

FIGURE 1 – Images utilisés lors des tests



(a) Gradient horizontal (b) Gradient vertical (c) Gradient composé

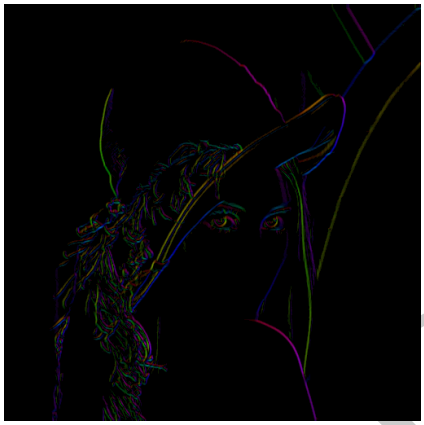
(d) Contour seuillé (e) Contour affiné (f) Contour fermé

FIGURE 2 – Résultats intermediaire pour des formes simples

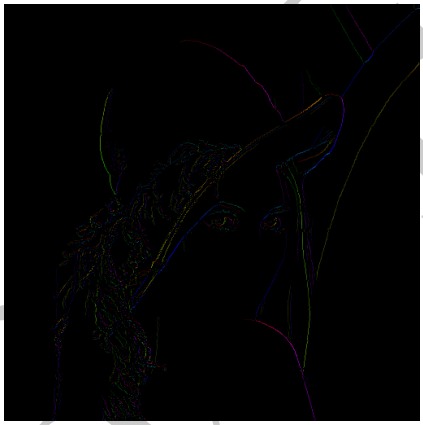
(a) Gradient horizontal

(b) Gradient vertical

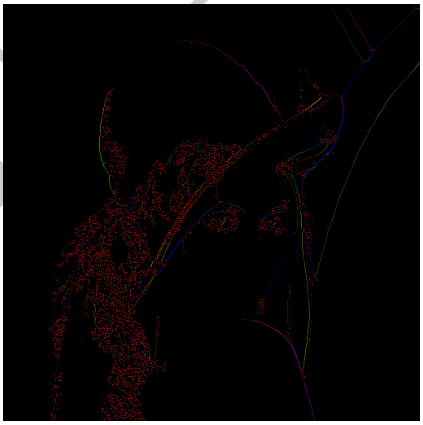
(c) Gradient composé



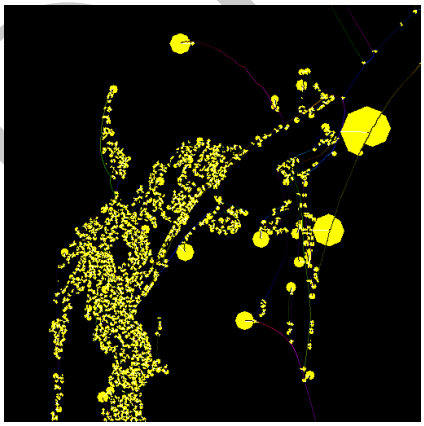
(d) Contour seuillé



(e) Contour affiné



(f) Points d'ancrage



(g) Limites de la diffusion

(h) Apres fermeture

FIGURE 3 – Résultats intermediaire pour Lena