Détection de contours par approche du premier ordre

François Lepan

26 mars 2013

Introduction

Dans ce rapport nous allons voir comment mettre en oeuvre la détection de contours par approche du premier ordre. Nous verrons l'utilité du calcul de la norme d'un gradient et différents cueillages permettant d'aboutir à la détection de contour.

1 Seuillage de la norme d'un gradient

Tout d'abord pour détecter un contour il faut effectuer certains calculs. Le premier est le calcul de la norme d'une image.

Après avoir calculé sa norme nous obtenons des valeurs pour les pixels de la norme. Analysons ces valeurs.

L'image obtenue est de profondeur 32 bits. Comment expliquer sa valeur minimale et sa valeur maximale ?

La valeur minimale est égale à 0 et sa valeur maximale est égale à 904,896. Ces valeurs s'expliquent par le calcul de la norme. Sachant que les valeurs pour l'image convolutionY (*cf.* Fig. 1 à droite) varient entre -804 et 895 et pour l'image convolutionX (*cf.* Fig. 1 à gauche) elles varient entre -844 et 788.

Donc après calcul ($\sqrt{p_x^2 + p_y^2}$) on peut obtenir des valeurs qui oscillent entre 0 et 904,896 sur l'image de la norme (*cf.* Fig. 2).

Maintenant que nous avons obtenu l'image de la norme, on se rend bien compte qu'il y a beaucoup de pixels "bruit".

Un procédé pour les éliminer est le seuillage des valeurs de l'image de la norme du gradient.

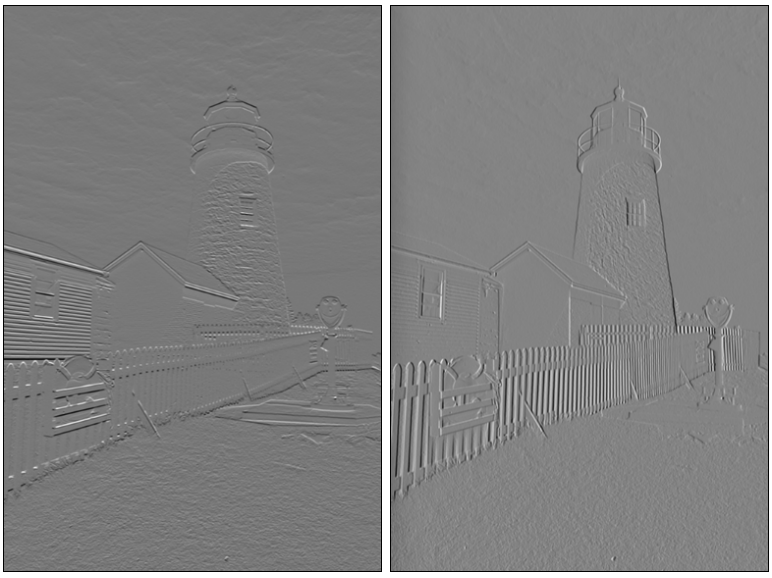

FIGURE 1 – Résultat de la convolution en X à gauche et Y à droite

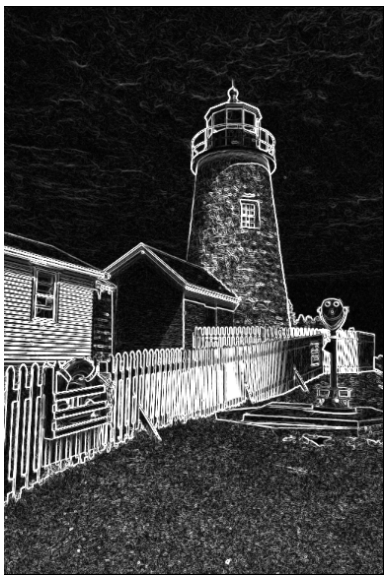

FIGURE 2 – Résultat de la norme du gradient

Seuillage de la norme du gradient précédemment calculée

Une première approche est de calculer un seuil global sur l'image.

Est-il possible de trouver ainsi un seuil global pour l'image qui mette en évidence les pixels contours de manière satisfaisante ?

Non ce n'est pas possible car si on met le seuil trop haut afin d'éliminer des pixels "bruit" on élimine par la même occasion des pixels de contour (*cf.* Fig. 3 à droite). Mais si le seuil est trop bas on garde trop de pixel "bruit" (*cf.* Fig. 3 à gauche).

FIGURE 3 – Résultat d'un seuillage sur la norme du gradient. À gauche un seuillage bas et à droite un seuillage haut

On voit bien que cette approche n'est pas satisfaisante il faudrait avoir plusieurs seuils en fonction de la position du pixel. Il faut pour cela calculer des maxima locaux à l'image de la norme du gradient.

2 Détection des maxima locaux de la norme d'un gradient

Afin de calculer les maxima locaux il faut pour cela s'aider de la direction du gradient de Sobel et en fonction de la direction prendre le pixel qui a la valeur maximale (*cf.* Fig.4).

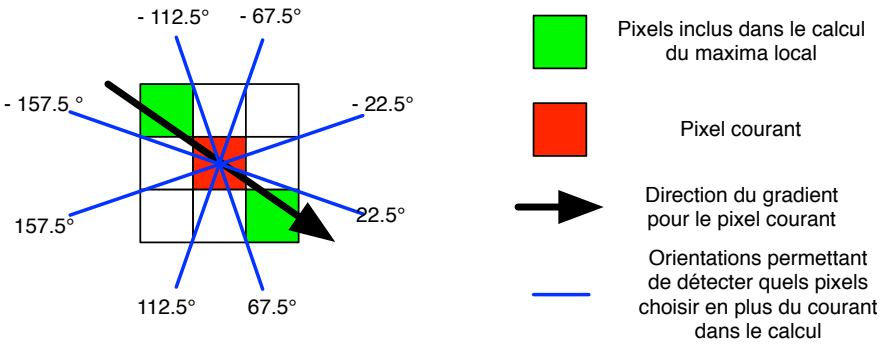

FIGURE 4 – Schéma de l'utilisation du gradient de Sobel pour le calcule des maxima locaux

Examinons l'image représentant la direction du gradient de Sobel et interprétons la

Prenons un pixel du bord droit du phare sur l'image orientation (*cf.* Fig. 5). Sur l'image de base le phare est plus clair que le fond. Donc après avoir calculé l'orientation on est censé obtenir une valeur proche de 180^o (l'orientation doit aller dans le sens de la valeur de pixel plus élevé). Après observation des valeurs grâce à l'outil *Pixel Inspector* on observe des valeurs de pixels comprises entre 170 et 180 degré ce qui est normal.

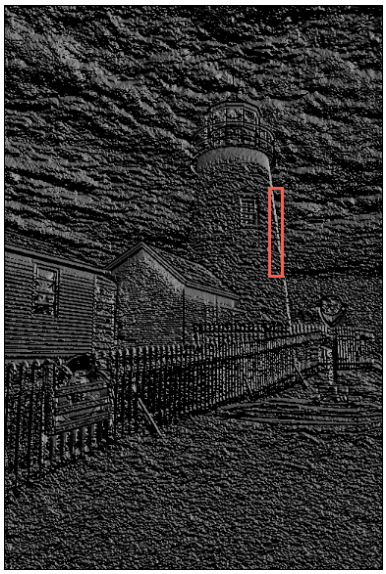

FIGURE 5 – Image orientation obtenue grâce aux images de la Fig. 1 et au calcul suivant : $atan2(p_y, p_x) * 180/\pi$ avec p_x le pixel de la convolution en X et p_y le pixel de la convolution en Y

Image obtenue après calcul des maxima locaux

On obtient l'image suivante Fig. 6 (à droite). En la comparant à l'image de la norme du gradient (Fig. 6 à gauche) on voit bien que la plupart des pixels "bruit" ont été supprimé. Mais il reste encore des pixels "bruit", afin de les supprimer il faut encore effectuer un seuillage particulier le seuillage des maxima locaux par hystérésis.

FIGURE 6 – de gauche à droite : résultat du calcul de la norme du gradient et résultat du calcul des maxima locaux de la norme du gradient

3 Seuillage des maxima locaux par hystérésis

Dans cette partie nous allons expliquer l'algorithme du seuillage des maxima locaux par hystérésis grâce à l'algorithme suivant :

```
public ByteProcessor hystIter(ImageProcessor imNormeG, int seuilBas, int seuilHaut) {

        // On récupère la taille de l'image
        int width = imNormeG.getWidth();
        int height = imNormeG.getHeight();

        // On va créer une liste de pixels contours pour lesquels la valeur
        // est supérieure au seuil haut.
        // cette liste servira à savoir si il y a encore des pixels voisins
        // aux pixels contours compris entre les deux seuils
        ByteProcessor maxLoc = new ByteProcessor(width,height);
        List<int[]> highpixels = new ArrayList<int[]>();

        for (int y=0; y<height; y++) {
```

```
        for (int x=0; x<width; x++) {

                // On récupère le pixel courant de l'image issue
                // du calcul des maxima locaux
                int g = imNormeG.getPixel(x, y)&0xFF;

                // Si le pixel est inférieur au seuil bas on laisse le pixel à 0.
                // Cela permet d'éliminer les pixels non contour (non candidat).
                if (g<seuilBas) continue;

                // Si le pixel est supérieur au seuil haut on
                // le garde comme pixel contour.
                if (g>seuilHaut) {
                        maxLoc.set(x,y,255);
                        highpixels.add(new int[]{x,y});
                        continue;
                }

                // On met le pixel dont la valeur est comprise entre le seuil bas
                // et le seuil haut dans la nouvelle image. Ce seront les pixels
                candidats.
                maxLoc.set(x,y,128);
        }
}


// on sauvegarde les pas pour accéder aux pixels voisins
int[] dx8 = new int[] {-1, 0, 1,-1, 1,-1, 0, 1};
int[] dy8 = new int[] {-1,-1,-1, 0, 0, 1, 1, 1};

// La nouvelle liste de pixels contours
List<int[]> newhighpixels = new ArrayList<int[]>();

// tant qu'il reste des candidats potentiels
while(!highpixels.isEmpty()) {

        // on vide la liste de nouveaux pixels contours
        newhighpixels.clear();

        // Pour chaque pixel coutour
        for(int[] pixel : highpixels) {
                int x=pixel[0], y=pixel[1];

                // On cherche à savoir lesquels de ses voisins
                // sont des pixels candidats.
                // Si ce pixel est un pixel candidat alors on l'ajoute
                // dans les nouveaux pixels contours.
                for(int k=0;k<8;k++) {
                        // On récupère le voisin en fonction du pas
```

```
                                int xk=x+dx8[k], yk=y+dy8[k];

                                // on évite les effets de bord
                                if (xk<0 || xk>=width) continue;
                                if (yk<0 || yk>=height) continue;

                                // si le pixel est un candidat on le sauvegarde en tant que
                                // nouveau pixel contour et on met la valeur à l'indice de ce
                                // pixel à 255
                                if (maxLoc.get(xk, yk)==128) {
                                        maxLoc.set(xk, yk, 255);
                                newhighpixels.add(new int[]{xk, yk});
                                }
                        }
                }

                // On met dans la nouvelle liste de pixels contours les nouveaux pixels
                // précédemment calculé
                List<int[]> swap = highpixels;
                highpixels = newhighpixels;
                newhighpixels = swap;
        }

        // On élimine les pixels isolés
        for (int y=0; y<height; y++) {
                for (int x=0; x<width; x++) {
                        if (maxLoc.get(x, y)!=255) maxLoc.set(x,y,0);
                }
        }

        // On retourne l'image
        return maxLoc;
}
```

Lorsque cet algorithme est appelé sur l'image résultant du calcul des maxima locaux (*cf.* Fig. 6 à droite) on obtient la Fig. 7

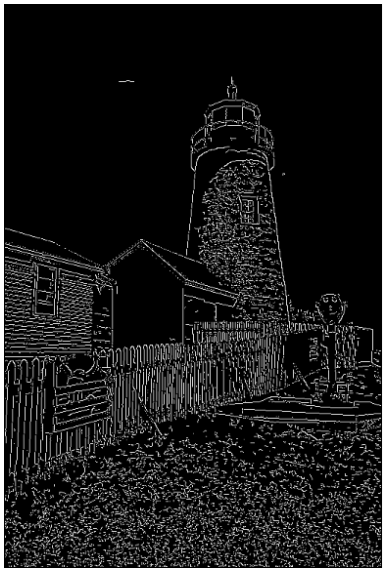

FIGURE 7 – Image résultant du seuillage des maxima locaux par hystérésis avec pour valeur de seuil min 30 et seuil max 100

Conclusion

On se rend bien compte que la détection de contour n'est pas une chose simple. Il faut calculer la norme de gradient de l'image pour ensuite en tirer une image résultant du calcul des maxima locaux et enfin appliquer un seuillage par hystérésis. Après tous ces calculs on obtient une image de contour satisfaisante mais malgré tout il reste des contours indésirables. Il faudra peut être appliquer de nouveaux calculs sur l'image afin d'enlever tout contour indésirable.