Rapport TP1 - Analyse d’images

Yanisse FERHAOUI - Amine ITJI

I. Calcul des filtres

Dans notre tp, la fonction **convFilter** est utilisée pour appliquer un filtre de convolution à une image. Elle prend en entrée une image ainsi qu'un noyau de convolution, le facteur de normalisation du noyau, et enfin le niveau de seuillage, puis effectue la convolution entre l'image et le noyau pour produire une nouvelle image résultante. Cette fonction est utilisée pour appliquer les filtres de Prewitt, Sobel et Kirsch dans notre cas. Elle peut être appelée avec le code suivant en C++ :

```
Mat gradient_prewitt = convFilter(input_image, prewitt_kernel, factor, threshold);
```

Cette fonction crée une image noire qui sera l’image de sortie, puis parcours cette dernière en appliquant le filtre en paramètres en faisant appel à **filterPixel** que nous avons également implémenté, prenant en paramètres l’image d’entrée, le noyau à utiliser, les coordonnées du pixel à filtrer et le facteur de normalisation du filtre. Cette fonction parcourt les 8 pixels autour de celui sélectionné et retourne le pixel filtré divisé par le facteur de normalisation.

Quant au facteur de normalisation, il est simplement calculé avant la convolution grâce à la fonction **normalizeFilter** qui prend en paramètre le noyau à normaliser. Nous avons fait le choix de réaliser cette opération avant le filtrage, puis d’inclure la variable *factor* dans les paramètres des deux autres fonction pour éviter de faire le calcul de la norme (nb de lignes*nb de colonnes) fois ce qui nous aurait fait perdre énormément de temps de calcul et de mémoire inutilement.

II. Cas Bidirectionnel

Le calcul du cas bidirectionnel se trouve donc dans la fonction **convFilter**, en effet nous avons utilisé **filterPixel** pour un filtre donné ainsi que sa rotation à 90° de manière à avoir une valeur verticale *f_x* ainsi qu’une valeur horizontale *f_y*. Nous avons obtenu la valeur du pixel en effectuant l’opération suivante : sqrt(f_x² + f_y²), et arcTan(f_y / f_x) pour sa pente.

III. Cas Multidirectionnel

Le cas multidirectionnel, quant à lui, se trouve dans la fonction **convFilterMulti**, et fait cette fois-ci appel quatre fois à la fonction **filterPixel** pour obtenir les valeurs g_1, g_2, g_3, et g_4, deux valeurs sont calculés de manière linéaire comme pour la méthode de convolution précédente et deux autres pour les différentes diagonales. Le calcul de la valeur

du pixel est le suivant max(g_1, g_2, g_3, g_4) pour le module, pour la pente le calcul dépend du résultat du module.

IV. Calcul du seuillage

Pour le calcul du seuillage, nous avons opté pour la méthode globale car nous n'avions pas le temps d'aborder la méthode par hystérésis, cependant la méthode globale donnait un indice de seuil trop fort, beaucoup de contours finissent par disparaître. Finalement, nous avons mis l'indice de seuillage à 40.0.

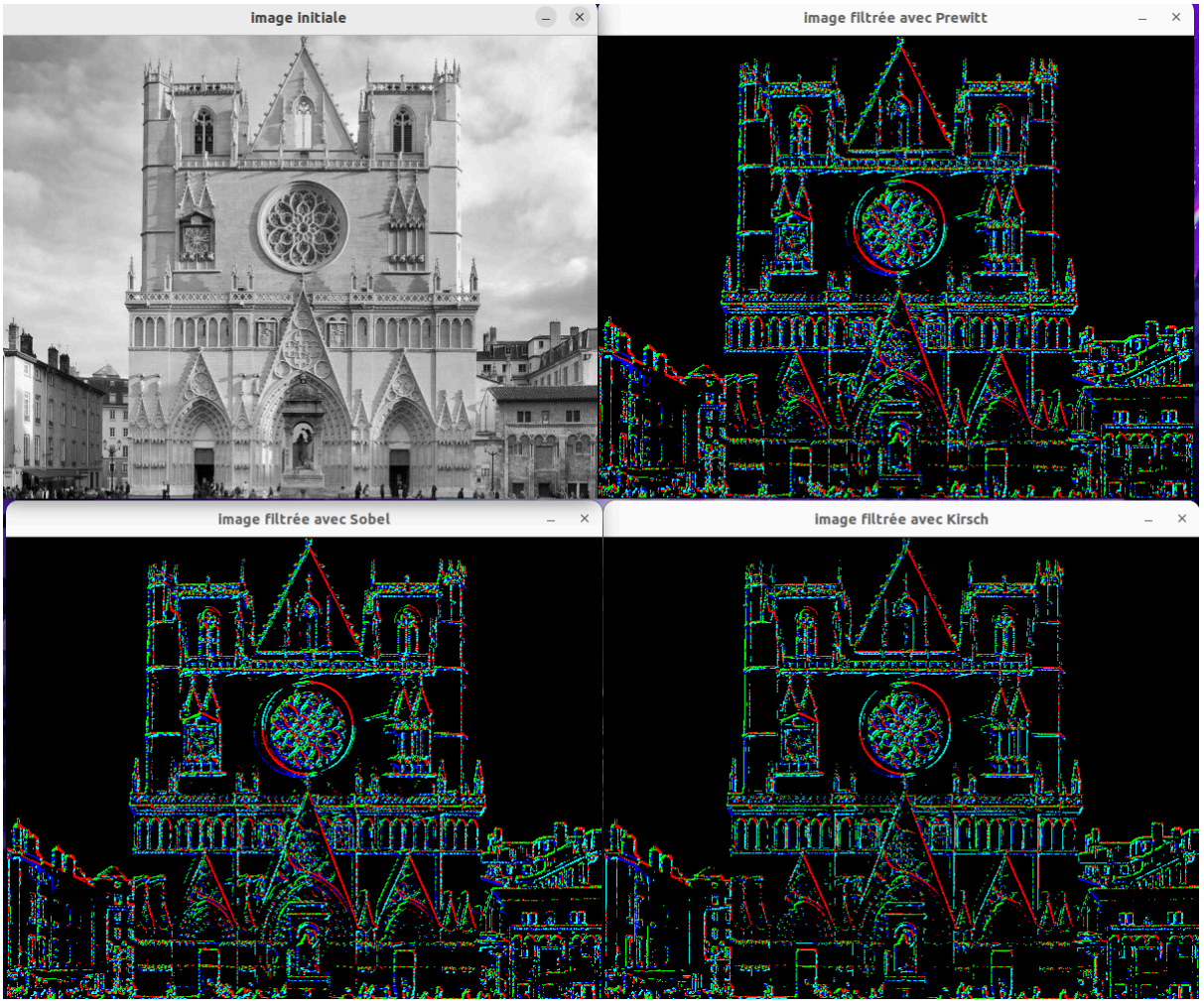

Figure 1 : Capture d'écran du filtrage de l'image Cathedrale-Lyon.jpg