

Analyse et Synthèse d'Images

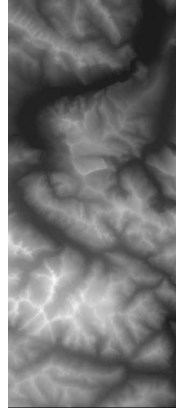
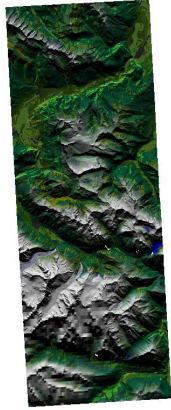
Jérôme Martin

TDa3

Détection de points d'intérêt

1. Objectif.

L'objectif de ce TDa est la détection de points d'intérêt. Nous nous plaçons dans une application d'observation de topographie de terrain. Nous cherchons à obtenir les caractéristiques du terrain, c'est-à-dire, les lignes de crêtes, les vallées, les rivières... Pour ceci, nous utilisons une approche multi-résolutions permettant de faire "disparaître" les détails du terrain, un détecteur de contraste permettant l'extraction des contours et un seuillage par hystérésis permettant de détecter ces points d'intérêt.



Vue du terrain avec végétation Topographie du terrain

2. Exercices.

1. Pyramide de gaussienne

- Calculer une pyramide à 3 niveaux sans lissage sur l'image de topographie.
- Idem en lissant les images trois fois par un filtre binomial de niveau 2.
- Calculer les images de différence entre les images des questions *a)* et *b)*. Calculer l'énergie des images de différence.

Rappel : Energie d'une image

L'énergie \mathcal{E}_I de l'image I est calculée par la formule :

$$\mathcal{E}_I = \sqrt{\sum_{i=0}^{width} \sum_{j=0}^{height} I^2(i, j)}$$

2. Détecteur de Sobel.

- Ecrire une fonction `sobel` calculant la détection de Sobel sur une image donnée en paramètre. On pourra prendre comme prototype :

```
void sobel (TPixmapFloat & img,
           TPixmapFloat & E, TPixmapFloat & Phi);
```

Appliquer le détecteur de Sobel sur l'image de topographie.

- b) Calculer la pyramide des modules et des orientations de Sobel sur les images de la pyramide de la question 1b).

Rappel : Détecteur de Sobel

Le détecteur de Sobel est un détecteur de contraste très populaire consistant à calculer des images E_0 et E_{90} en convoluant l'image d'origine avec les filtres m_0 et m_{90} définis par :

$$m_0 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad m_{90} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Le module du contraste E (magnitude) et la direction du contraste (phase) ϕ sont alors calculés pour le pixel (i, j) par :

$$E(i, j) = \sqrt{E_0^2(i, j) + E_{90}^2(i, j)}$$

et

$$\phi(i, j) = \tan^{-1} \left(\frac{E_{90}(i, j)}{E_0(i, j)} \right)$$

Rappel : fatan2

La fonction `fatana2` est définie par :
`float fatana2 (float x, float y)`
 et équivaut à $\tan^{-1} \left(\frac{y}{x} \right)$

3. Détection des extréma.

- a) Ecrire une fonction `seuillage` calculant l'image des extréma d'une image de contraste en utilisant un seuillage par hystérésis. On pourra prendre comme prototype de la fonction :

```
void seuillage (TPixampFloat & contraste,
               TPixmapFloat & seuillee,
               float Tl, float Th);
```

Appliquer le seuillage à l'image de contraste de la question 1a) en variant les valeurs des seuils. On pourra essayer les jeux de valeurs suivantes :

\mathcal{T}_l	\mathcal{T}_h
10	20

- b) Appliquer la fonction à tous les niveaux de la pyramide de la question 1b) et comparer les résultats sur l'image de magnitude et l'image d'orientation.

Rappel : Seuillage par hystérésis

Le principe du seuillage par hystérésis est de marquer les points en utilisant deux seuils : un seuil bas (\mathcal{T}_l) sensible au bruit et générant des candidats et un seuil haut (\mathcal{T}_h) donnant une bonne indication d'un vrai contraste.

La marquage d'un point (i, j) est donné par l'algorithme :

$$F(i, j) := \begin{cases} 0 & \text{si } \mathcal{T}_h < |E(i, j)|, \\ 1 & \text{si } \mathcal{T}_l < |E(i, j)| < \mathcal{T}_h, \\ 2 & \text{sinon} \end{cases}$$