Introduction au traitement numérique des images

Humbert Florent

27 mai 2007

Table des matières

Chapitre 1

Avant-propos

1.1 Introduction

Ce cours vous introduira les outils mathématiques nécessaires aux traitements d'image, indépendamment d'une implémentation du type abstrait image. Dans une première partie, nous définirons les notations d'usage dans ce cours. Ensuite, nous formaliserons la notion générale d'image (indépendamment du nombre de couleurs...). Puis nous aborderons la notion de filtre, de masque et nous nous attarderons sur les possibilités d'applications des filtres.

Certaines parties, telles que la formalisation peuvent être ignorées à la première lecture, elles nécessitent de connaître certaines notions mathématiques comme les espaces vectoriels.

Afin de connaître l'origine des filtres, certaines preuves ont été réalisées. Mais pour l'application pure et simple en traitement d'image, la compréhension de ces preuves n'est pas nécessaire.

À noter que certaines preuves ont été totalement réalisées par moi-même (même si une preuve équivalente existe), l'erreur étant humaine jusqu'à dernier ordre, il se peut qu'une erreur se soit glissée par inadvertance. Veuillez m'en informer si vous avez le temps.

1.2 Remerciements

Je tiens à remercier Pierre Schwartz et Miles pour les corrections qu'ils ont su apporter.

Chapitre 2

Notations

Avant de commencer le cours, voici une petite liste de notations (assez classiques) utilisées dans ce cours.

2.1 Ensembles

Notation 1 (Ensemble de fonctions) *On notera $\mathcal{F}(E,F)$ l'ensemble des fonctions de l'ensemble E dans l'ensemble F.*

Notation 2 (Ensemble d'applications linéaires) *On notera $\mathcal{L}(E,F)$ l'ensemble des applications linéaires de E dans F.*

Notation 3 (Ensemble d'applications intégrables) *On notera $\mathcal{L}^1(E)$ l'ensemble des applications intégrables sur E à valeur dans $\mathbb{C}$ (E étant un ouvert ou un fermé).*

Notation 4 (Ensemble de séries sommables) *On notera l^1 l'ensemble des séries sommables sur $\mathbb{Z}^2$.*

Notation 5 (Ensembles classiques) *Les ensembles $\mathbb{C}$, $\mathbb{R}$, $\mathbb{N}$ et $\mathbb{Z}$ désignent respectivement l'ensemble des complexes, des réels, des entiers naturels et des entiers relatifs.*

L'ensemble $\mathbb{K}$ désignera $\mathbb{C}$ ou $\mathbb{R}$.

Notation 6 (Ensemble d'entier) *On note $[[n,m]]$ l'ensemble des entiers compris entre n et m. (Donc $[[n,m]] = [n,m] \cap \mathbb{Z}$)*

Notation 7 (Ensemble cercle) *Soit $r > 0$, $(x,y) \in \mathbb{R}^2$. On notera $\mathcal{C}_r(x,y)$ la partie de $\mathbb{R}^2$ définie par :*

$$\forall (a,b) \in \mathbb{R}^2, (a,b) \in \mathcal{C}_r(x,y) \Leftrightarrow (x-a)^2 + (y-b)^2 \leq r^2$$

Notation 8 (Frontière) *Soit F un ensemble fermé. On note ∂F la frontière de F.*

On peut éventuellement étendre cette définition à des ouverts.

2.2 Convolution deux dimensions

Notation 9 (Convolution sur un espace continu) *Soit* $(f,g) \in \mathcal{L}^1(\mathbb{R}^2)^2$, *on note* $f*g$ *le produit de convolution entre* f *et* g *défini par :*

$$\forall (x,y) \in \mathbb{R}^2, (f*g)(x,y) = \iint_{\mathbb{R}^2} f(u,v)\cdot g(x-u,y-v)\,du\,dv$$

Notation 10 (Convolution sur un espace discret) *Soit* $(f,g) \in l^1$, *on note* $f*g$ *le produit de convolution entre* f *et* g *défini par :*

$$\forall (x,y) \in \mathbb{Z}^2, (f*g)(x,y) = \sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty} f(i,j)\cdot g(x-i,y-j)$$

Notation 11 (Convolution sur un espace discret, cas simple) *En général, la première fonction est à support borné, dans ce cas, on peut définir le produit de convolution comme suit :*

Soit $f \in l^1$ *et* $h \in \mathcal{F}([[0,n-1]]\times[[0,m-1]],\mathbb{R})$ *avec* n *et* m *impair. On note* $h*f$ *le produit de convolution entre* f *et* h *défini par :*

$$\forall (x,y) \in \mathbb{Z}^2 (h*f)(x,y) = \sum_{i=-(n-1)/2}^{(n-1)/2}\sum_{j=-(m-1)/2}^{(m-1)/2} h(i+\frac{n-1}{2}, j+\frac{m-1}{2})\cdot f(x+i,y+j)$$

On peut également écrire avec une notation sous forme de suite (qui est exactement équivalente) :

$$\forall (x,y) \in \mathbb{Z}^2 (h*u)_{x,y} = \sum_{i=-(n-1)/2}^{(n-1)/2}\sum_{j=-(m-1)/2}^{(m-1)/2} h_{i+\frac{n-1}{2},j+\frac{m-1}{2}}\cdot f_{x+i,y+j}$$

Attention *Dans ce cas, on ne réalise plus la convolution centrée en* 0 *pour* h, *mais centrée en* $\left(\frac{n-1}{2},\frac{m-1}{2}\right)$

2.3 Petit et grand o

Notation 12 (Petit o) *Soit* $(f,g) \in \mathcal{F}(\mathbb{K},\mathbb{K})^2$. f *est négligeable devant* g *si et seulement si il existe une application* $\epsilon \in \mathcal{F}(\mathbb{K},\mathbb{K})$ *tel que :*

$$\begin{aligned} \forall x \in I, \qquad & f(x) = \epsilon(x)g(x) \\ & lim_{x\to a}\epsilon(x) = 0 \end{aligned}$$

On note $f = o_a(g)$ *où simplement* $f = o(g)$ *si il n'y a pas de matière à confusion.*

Notation 13 (Grand O) *Soit* $(f,g) \in \mathcal{F}(\mathbb{K},\mathbb{K})$. f *est dominée par* g *si et seulement si il existe* $c > 0$ *et* $x_0 \in \mathbb{K}$ *tel que*

$$\forall x > x_0, |f(x)| < c\cdot|g(x)|$$

On note $f = O(g)$.

2.4 Fonctions

Notation 14 (Conjugué) *On désigne par l'opérateur * le conjugué d'un nombre complexe défini par :*

$$\forall (a,b) \in \mathbb{R}^2, (a+ib)^* = a - ib$$

Notation 15 (Gradient) *Soit $f \in \mathcal{F}(\Omega, \mathbb{R})$ une application de classe C^1 où Ω est un ouvert dans $\mathbb{R}^2$, on note : $\nabla(f)$ le gradient de f défini par :*

$$\forall (x,y) \in \Omega, \nabla(f)(x,y) = \left(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y) \right)$$

Notation 16 (Rotation) *On désignera par r_θ, la rotation vectorielle d'angle θ de $\mathbb{R}^2$ dans $\mathbb{R}^2$ définie par :*

$$\forall (x,y) \in \mathbb{R}^2, r_\theta(x,y) = \begin{pmatrix} x \cdot \cos(\theta) - y \cdot \sin(\theta) \\ x \cdot \sin(\theta) + y \cdot \cos(\theta) \end{pmatrix}$$

Notation 17 (Translation) *L'application $\tau \in \mathcal{F}(\mathbb{K}^2, \mathbb{K}^2)$ est une translation si et seulement si il existe $(l,k) \in \mathbb{K}^2$ tel que :*

$$\forall (x,y) \in \mathbb{K}^2, \tau(x,y) = \begin{pmatrix} x+l \\ y+k \end{pmatrix}$$

Notation 18 (Produit scalaire) *On notera $<,>$ le produit scalaire usuel entre deux vecteurs de $\mathbb{R}^n$ défini par :*

$$\forall u \in \mathbb{R}^n, \forall v \in \mathbb{R}^n, < u, v >= \sum_{i=1}^{n} u(n) \cdot v(n)$$

Notation 19 (Dérivée partielle) *Soit $\vec{n}$ un vecteur de norme 1 tourné d'un angle θ. Soit $f \in \mathcal{F}(\mathbb{R}^2, \mathbb{R}^2)$ une application C^1. On notera $\frac{\partial f}{\partial \vec{n}}$ l'application de $\mathbb{R}^2$ dans $\mathbb{R}^2$ définie par :*

$$\forall (x,y) \in \mathbb{R}^2, \frac{\partial f}{\partial \vec{n}}(x,y) =< \nabla(f), \vec{n} >= \left(\frac{\partial f}{\partial x} \cos(\theta) + \frac{\partial f}{\partial y} \sin(\theta) \right)(x,y)$$

En notant $\vec{n_\perp}$, le vecteur orthogonal à $\vec{n}$, on notera : $\frac{\partial f}{\partial \vec{n_\perp}}$ l'application définit par :

$$\forall (x,y) \in \mathbb{R}^2, \frac{\partial f}{\partial \vec{n_\perp}}(x,y) =< \nabla(f), \vec{n_\perp} >= \left(-\frac{\partial f}{\partial x} \sin(\theta) + \frac{\partial f}{\partial y} \cos(\theta) \right)(x,y)$$

Notation 20 (Opérateur Laplacien) *Soit f une application C^2 définit sur un ouvert Ω de $\mathbb{R}^2$ à valeur dans $\mathbb{R}^2$, on note Δ l'opérateur laplacien défini par :*

$$\forall (x,y) \in \Omega, \Delta(f)(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y)$$

2.5 Norme

Notation 21 (Norme de vecteur) *Soit v un vecteur de $\mathbb{R}^n$.*

On note :

–
$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v(i)^2}$$

–
$$\|v\|_1 = \sum_{i=1}^{n} |v(i)|$$

–
$$\|v\|_\infty = max_{i=1,n} v(i)$$

Notation 22 (Norme de fonction) *Dans le cas d'existence de ces expressions, on définit les normes suivantes pour une application f définie sur une partie U de $\mathbb{R}^2$ dans un espace vectoriel.*

–
$$\|f\|_1 = \iint_U |f(x,y)|\, dx\, dy$$

–
$$\|f\|_\infty = sup_{(x,y)\in U} f(x,y)$$

On peut noter que la norme infinie s'applique pour toute fonction bornée et que la norme 1 s'applique pour les fonctions intégrables sur leur espace de départ (à noter que si U est fermé, alors cette norme existe).

Notation 23 (Norme d'ensemble) *Soit E un fermé de $\mathbb{R}^2$. On note $\|E\|$ l'aire de E définie par :*

$$\|E\| = \iint_E dx\, dy$$

On peut étendre cette définition à d'autres ensembles. Soit F une partie de $\mathbb{R}^2$, si il existe un fermé E tel que $E\backslash\partial E \subset F \subset E$ Alors $\|F\| = \|E\|$

2.6 Notation relative au cours

Voici la liste des notations utilisées uniquement dans ce cours.

Notation 24 (Image) *On notera $\mathbb{I}$ l'ensemble contenant toutes les images si il n'y a pas confusion entre espace discret et espace continu. Sinon, on écrira :*

- *Dans le cas continu :*
$$\mathbb{I}_c = \mathcal{F}(\mathbb{R}^2, E)$$
Dans ce cas, on considérera éventuellement les restrictions des fonctions aux fonctions de classe quelconque. C'est à dire : $\mathbb{I}_c = C^p(\mathbb{R}^2, E)$.
- *Dans le cas discret :*
$$\mathbb{I}_d = \mathcal{F}(\mathbb{Z}^2, E)$$

E est ici un espace vectoriel de dimension finie.

Chapitre 3

Formalisation

3.1 Image sur un espace continu

Nous introduisons une formalisation des images sur un espace continu dans le but de définir la discrétisation et dans le but de pouvoir déterminer des approximations correctes dans un espace discret de certains opérateurs (comme le laplacien).

3.1.1 Définition

Pour pouvoir être le plus complet possible, il faudra introduire un $\mathbb{R}$-espace vectoriel E de **dimension finie** qui correspondra aux couleurs. Par exemple, une définition des couleurs en RGB[1] nécessitera un espace à trois dimensions tandis qu'une seule dimension sera nécessaire pour une image grise (ou en niveau de gris).

Définition 1 (Canal) *Un sous-espace vectoriel de dimension* 1 *de l'espace vectoriel E s'appelle un canal.*

Actuellement, en informatique, les couleurs sont souvent codées sur 24 bits (donc 256 possibités pour chaque canal), si l'on additionne une couleur blanche $(255, 255, 255)$ avec une autre couleur blanche, cela devrait donner également une couleur blanche[2]. Mais pour réaliser certains types de calculs, borner les résultats de ceux-ci n'est pas pratique (nous verrons plus tard pourquoi), c'est pourquoi nous réaliserons les calculs dans un espace vectoriel. Ensuite, juste avant l'affichage de l'image à l'écran, nous repasserons dans la bonne plage de couleurs (un entier entre 0 et 255 pour chaque canal dans cet exemple).

Définition 2 (Image dans un espace continu) *Soit $(E, +, .)$ un espace vectoriel de dimension finie. Une image à valeur dans E est une application de $\mathbb{R}^2$ dans E.*

On note $\mathbb{I}_c$ l'ensemble des images à valeur dans un espace vectoriel E.

[1] Red, Green, Blue

[2] Ce qui montre que cet espace n'est pas un espace vectoriel

3.1.2 Image à espace de départ borné

On pourrait se demander pourquoi l'espace de départ d'une image est $\mathbb{R}^2$ alors que dans la pratique, l'espace est borné. En réalité, cela permet de réaliser certains calculs de manières exactes (par exemple lors du calcul d'une convolution).

Pour se ramener dans un cas « réel », il suffit de considérer que l'image est à support borné.

Rappel 1 (Fonction à support bornée) *Dans le cas de fonction dont l'espace de départ est $\mathbb{R}^2$, une fonction $f \in \mathcal{F}(\mathbb{R}^2, E)$ où E est un espace vectoriel est à support borné si et seulement si*

$$\exists m > 0, \forall (x, y) \notin \mathcal{C}_m(0, 0), f(x, y) = 0$$

(Où 0 est l'élément neutre de la loi + de l'espace vectoriel E).

3.2 Image discrète

Pour le traitement d'image en informatique, on utilise des images discrétisées (que l'on peut obtenir à partir d'une image réelle de différentes manières).

3.2.1 Définition

De la même façon que pour une image dans un espace continu, on définit une image dans un espace discret comme suit :

Définition 3 (Image discrète) *Soit E un espace vectoriel, une image discrète est une application de $\mathbb{Z}^2$ dans E.*

3.2.2 Cas bornée

En règle générale, les images que l'on traite sont de dimensions finies, mais on peut se ramener à ce cas en considérant les fonctions de $\mathbb{Z}^2$ dans E à support borné.

3.3 Discrétisation d'une image réelle

Pour pouvoir définir certaines opérations sur les image discrètes, il nous faut définir une manière de discrétiser une image réelle. Pour cela, on définit un pas de discrétisation h assez petit.

Ensuite, il suffit de réaliser le lien entre l'image f réelle et l'image u discrète par :

$$\forall (i, j) \in \mathbb{Z}^2, u(i, j) = u_{i,j} = f(h \cdot i, h \cdot j)$$

3.3.1 Exemple d'application

Soit $f \in \mathbb{I}_c$, une image C^1.
Comme pour tout $h > 0$ on a :

$$\forall (x,y) \in \mathbb{R}^2, \frac{\partial f}{\partial x}(x,y) = \frac{f(x+h,y) - f(x,y)}{h} + o(1)$$

D'où :

$$\forall (x,y) \in \mathbb{R}^2, \frac{\partial f}{\partial x}(x,y) \cdot h = u(\frac{x}{h} + 1, y) - u(\frac{x}{h}, y) + o(h)$$

En passant en entier, on obtient :

$$\forall (i,j) \in \mathbb{Z}^2, \frac{\partial f}{\partial x}(h \cdot i, h \cdot j) \cdot h = u(i+1,j) - u(i,j) + o(h)$$

On peut ensuite rentrer le réel h dans la dérivée et on obtient une approximation de la dérivée selon l'axe x de f (à un facteur près de décalage).

Ainsi, nous pouvons définir une approximation de la dérivée $v \in \mathbb{I}_d$ selon l'axe x d'une image $u \in \mathbb{I}_d$ par :

$$\forall (i,j) \in \mathbb{Z}^2, v(i,j) = u(i+1,j) - u(i,j)$$

Chapitre 4

Filtres

La plupart des opérations que l'on peut réaliser sur une image utilisent la notion de filtre. Ils peuvent servir à corriger une image, à détecter des contours ou encore à améliorer la netteté. Nous allons définir ce qu'ils sont et préciser certaines propriétés les concernant.

4.1 Définition générale

Définition 4 (Filtre) *Un filtre est une application de* $\mathbb{I}$ *dans* $\mathbb{I}$.

4.2 Filtres linéaires

Pour de nombreuses transformations d'images, on utilise uniquement des filtres linéaires. Nous allons donc surtout nous intéresser à ce type de filtre.

4.2.1 Définition

Un filtre linéaire n'est pas seulement une application linéaire, mais il dispose également d'une propriété d'invariance par translation.

En effet, si l'on dispose d'une image, on souhaiterait que le résultat par le filtre soit le même suivant que l'image soit disposée à un endroit ou à un autre.

Définition 5 (Filtre linéaire) *T est un filtre linéaire si et seulement si $T \in \mathcal{L}(\mathbb{I}, \mathbb{I})$ [1] et si pour toute translation τ (de $\mathbb{R}^2$ dans $\mathbb{R}^2$ dans le cas continu, de $\mathbb{Z}^2$ dans $\mathbb{Z}^2$ dans le cas discret) et pour toute image $f \in \mathbb{I}$,*

$$\forall (x, y), T(f \circ \tau)(x, y) = T(f)(\tau(x, y))$$

Où $T \circ (f \circ \tau) = (T \circ f) \circ \tau$

L'intervention d'une application linéaire provient du fait que l'on souhaite disposer d'opérateur que l'on puisse appliquer sur des parties d'image que l'on regroupe ensuite.

Par exemple, si une image i_1 est nulle pour $x > 0$ et si une image i_2 est nulle pour $x < 0$. Il serait *a priori* normal que : $T(i_1 + i_2) = T(i_1) + T(i_2)$.

[1] T est un endomorphisme dans l'ensemble des images

Avec des mots, appliquer le filtre sur les deux images puis regrouper les images résultantes est équivalent à appliquer directement le filtre sur l'ensemble des deux images.

4.2.2 Cas discret

On peut trouver une propriété très intéressante sur les filtres linéaires dans le cas discret.

Théorème 1 *Pour tout filtre linéaire T, il existe une image $h \in \mathbb{I}_d$ tel que :*

$$\forall (i,j) \in \mathbb{Z}^2, \forall u \in \mathbb{I}_d, T(u)(i,j) = (h * u)(i,j)$$

Définition 6 (Masque) *On appelle h un noyau de convolution, ou un masque de convolution ou simplement masque. Il est parfois appelé filtre par abus de langage.*

Preuve

Chaque fonction (ou suite) u de $\mathbb{Z}^2$ dans E peut être décomposée dans une base de $\mathcal{F}(\mathbb{Z}^2, E)$.

On note : $\delta^{x,y} \in \mathbb{I}_d$ l'image définie par :

$$\forall (i,j) \in \mathbb{Z}^2, \delta^{x,y}(i,j) = \left\{ \begin{array}{cc} 1 & si\ i = x\ et\ j = y \\ 0 & sinon \end{array} \right\}$$

À noter que les fonctions $\delta^(x,y)$ sont assez similaires au symbole δ de Kronecker (à l'exception qu'elles sont définies pour un espace de deux dimensions alors que le symbole de Kronecker n'est défini que pour un espace de dimension 1).

Ainsi, comme la famille $(\delta^{i,j})_{i,j}$ est une base de $\mathbb{I}_d$, on a :

$$u = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} u(i,j) \cdot \delta^{i,j}$$

En appliquant le filtre à u et en utilisant la linéarité de T, on obtient :

$$T(u) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} u(i,j) \cdot T(\delta^{i,j})$$

Comme T est à valeur dans $\mathbb{I}_d$, $T(\delta^{i,j})$ est décomposable dans la base, d'où :

$$\exists \alpha^{i',j'} \in \mathbb{I}_d, T(\delta^{i,j}) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \alpha^{i',j'}(i,j) \cdot \delta^{i,j}$$

En regroupant les termes, on trouve que :

$$\forall (n,m) \in \mathbb{Z}^2, \exists \alpha^{n,m}, T(u)(n,m) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \alpha^{n,m}(i,j) \cdot u(i,j)$$

Ensuite, en utilisant la propriété de la translation, on obtient la propriété :

$$\forall (l,k) \in \mathbb{Z}^2, \alpha^{n+l,m+k}(i+l,j+k) = \alpha^{n,m}(i,j)$$

En définissant l'image $h \in \mathbb{I}_d$ par :

$$\forall (i,j) \in \mathbb{Z}^2, h(i,j) = \alpha^{i,j}(0,0)$$

On obtient :

$$\begin{aligned}
\forall (n,m) \in \mathbb{Z}^2, T(u)(n,m) &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \alpha^{n,m}(n-i,m-j) \cdot u(n-i,m-j) \\
&= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \alpha^{i,j}(0,0) \cdot u(n-i,m-j) \\
&= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} h(i,j) \cdot u(n-i,m-j) \\
&= (h * u)(n,m)
\end{aligned}$$

4.2.3 Cas continu

Le cas continu n'est pas aussi simple, en effet, on ne peut pas trouver une base dans laquelle décomposer le signal (même si on peut montrer que cette base existe). On peut juste formuler le théorème suivant :

Théorème 2 *Pour toute image $h \in \mathbb{I}_c$ et $f \in \mathbb{I}_c$, le filtre T de $\mathbb{I}_c$ dans $\mathbb{I}_c$ défini par : $T(f) = h * f$ est un filtre linéaire.*

Preuve

La linéarité de T est claire à cause de la linéarité de l'intégrale. La propriété de translation se montre comme suit :

$$\forall (x,y) \in \mathbb{R}^2, (h*f)(x+k,y+l) = \iint_{\mathbb{R}^2} h(u,v) \cdot f(x+k-u,y+l-v)\, du\, dv$$

Comme k et l n'intervient que dans f, on peut poser la translation τ définie par : $\forall (x,y) \in \mathbb{R}^2, \tau(x,y) = (x+k,y+l)$.

D'où :

$$\forall (x,y) \in \mathbb{R}^2, (h*f)(\tau(x,y)) = \iint_{\mathbb{R}^2} h(u,v) \cdot (f \circ \tau)(x-u,y-v)\, du\, dv = (h*(f\circ\tau))(x,y)$$

Ce qui est équivalent à : $(T \circ f) \circ \tau = T \circ (f \circ \tau)$

4.2.4 Propriété du produit de convolution

Le produit de convolution (discret ou continu) a de nombreuses propriétés intéressantes :

- il est bilinéaire :

$$\forall (f,g,h) \in \mathbb{I}^3, \forall (a_1,a_2) \in \mathbb{R}^2, (a_1 \cdot f + a_2 \cdot g) * h = a_1(f*h) + a_2(g*h)$$

- il est commutatif :

$$\forall (f,g) \in \mathbb{I}^2, (f*g) = (g*f)$$

- il est associatif :

$$\forall (f,g,h) \in \mathbb{I}^3, (f*g)*h = f*(g*h)$$

  Cette propriété est importante. Si l'on dispose d'une image f et de deux masques h_1 et h_2. Alors calculer $(h_1 * h_2) * f$ est équivalent à calculer $h_1 * (h_2 * f)$. Donc si on connaît déjà les masques h_1 et h_2 (ce qui est en général vrai), on peut calculer une fois pour toute la convolution $h_1 * h_2$, ce qui permet d'éviter de nombreux calculs.
- il commute avec les translations

$$(h*f) \circ \tau = h * (f \circ \tau)$$

4.2.5 Astuce pour la programmation

Nous avons vu que f est à valeurs dans un espace vectoriel E, qui dispose donc d'une loi interne $+$ et d'une loi externe $\cdot$. Mais ce n'est pas toujours le cas lorsque l'on programme. Pour pallier à ce problème, on peut déterminer toutes les projections de f dans une base de E afin que ces fonctions soient à valeurs dans $\mathbb{R}$.

Soit $(e_k)_{k \in [[1,n]]}$ une base de l'espace vectoriel E. On détermine la projection de f dans chaque dimension. On définit ainsi une sous-image f_k à valeur dans $\mathbb{R}$ définie par :

$$\forall k \in [[1,n]], f_k = < f, e_k >$$

Et les projections h_k par :

$$\forall k \in [[1,n]], h_k = < h, e_k >$$

Ainsi, la convolution $h_k * f_k$ utilise des sommes et des produits dans $\mathbb{R}$, ce qui est plus facile à utiliser.

On reconstruit ensuite la convolution de f par un masque h par la formule :

$$h * f = \sum_{k=1}^{n} (h_k * f_k) \cdot e_k$$

On peut noter qu'en général, h_k est constant selon k car on applique le même masque à chaque canal de l'image.

Avec des mots, cela revient à appliquer la convolution à chaque canal, puis à regrouper ensuite tous les canaux.

Chapitre 5

Application de filtre (cas discret)

Nous allons dans ce chapitre définir certains filtres servant très fréquemment en traitement d'image.

Chaque filtre sera testé sur l'image suivante (figure 5.1).

FIG. 5.1 – Photographie de papillon réalisée par mes soins

5.1 Application de filtre

5.1.1 Généralités

Supposons dans un premier temps que les images ne soient pas à support borné mais que les masques de convolution le soient.

Pour simplifier les notations, l'image $u \in \mathbb{I}_d$ sera à valeur dans $\mathbb{R}$ (mais cela fonctionnerait de la même manière dans un autre espace vectoriel).

En général, on écrit le masque h sous forme de matrice, par exemple :

$$h = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Attention, pour simplifier, $h(x, y)$ correspondra au coefficient de h de la ligne y et de la colonne x contrairement à la notation avec les matrices.

Si une partie de l'image discrète s'écrit :

$$u = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & 1 & 2 & 3 & \dots \\ \dots & 4 & 5 & 6 & \dots \\ \dots & 1 & 7 & 2 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

Soient x et y les valeurs des paramètres de u correspondant au coefficient 5 (donc u(x,y) = 5).

Alors dans notre cas,

$$\begin{aligned} (h * u)(x, y) &= h(0,0) \cdot u(x-1, y-1) + h(1,0) \cdot u(x, y-1) \\ &\quad + h(2,0) \cdot u(x+1, y-1) + h(0,1) \cdot u(x-1, y) \\ &\quad + h(1,1) \cdot u(x, y) + h(2,1) \cdot u(x+1, y) \\ &\quad + h(0,2) \cdot u(x-1, y+1) + h(1,2) \cdot u(x, y+1) \\ &\quad + h(2,2) \cdot u(x+1, y+1) \\ &= -2 - 4 - 6 - 7 + 4 \cdot 5 \\ &= 1 \end{aligned}$$

On peut ainsi déterminer la valeur de la transformation en chaque point de cette manière.

Si h est de taille impaire (en largeur l et en hauteur h), on peut écrire l'application du masque comme cela :

$$\forall (x, y) \in \mathbb{Z}^2 (h*u)(x, y) = \sum_{i=-(l-1)/2}^{(l-1)/2} \sum_{j=-(h-1)/2}^{(h-1)/2} h(i+\frac{l-1}{2}, j+\frac{h-1}{2}) \cdot f(x+i, y+j)$$

5.1.2 Cas borné

Dans la « vraie vie », on travaille sur des images à taille limitée. Ainsi, lorsque l'on applique le masque, on peut chercher à accéder à des endroits qui ne sont pas définis. Par exemple, avec le même masque h et l'image suivante :

$$u = \begin{pmatrix} 2 & 3 & \dots \\ 6 & 2 & \dots \\ \dots & \dots & \dots \end{pmatrix}$$

Pour calculer $(h * u)(0, 0)$, on cherche à accéder au point : $u(-1, -1)$ qui n'est pas défini. Il y a plusieurs méthodes pour parer à ce problème, nous allons en voir deux.

NB : Dans ces deux méthodes, en convoluant une image de taille $n \times n$ par un noyau de taille $m \times m$, on obtiendra une image de taille $(n+m-1) \times (n+m-1)$.

Première méthode

La première méthode la plus simple consiste à renvoyer 0 lorsque l'on cherche à accéder à une zone hors borne, ou dans le cas général, au neutre 0 pour la loi $+$ de l'espace vectoriel E (en général le noir). Cela permet ainsi de se ramener directement au cas général en considérant des images de taille infinie mais à support borné.

Le problème majeur à cette méthode étant que cela peut avoir un effet indésirable sur les bords, les couleurs auront tendances à s'assombrir, mais pour de petits masques (ici, taille 3*3), cela est négligeable.

Deuxième méthode

La deuxième méthode consiste à déterminer un point (x, y) se trouvant dans les bornes de l'image de distance minimale par rapport au point que l'on souhaite déterminer la couleur.

Si l'on veut accéder au point (x', y') hors bornes. On détermine les entiers $(x, y) \in [[0, l]] \times [[0, h]]$ minimisant la distance $\|(x' - x, y' - y)\|$

On en déduit une extension de l'image $u \in \mathbb{I}_d$ au point (x', y') valant : $u(x', y') = u(x, y)$

Conclusion

On montre ainsi que même pour une image bornée, on peut se ramener au cas non bornée mais en réduisant les calculs dans les intervalles que l'on souhaite (en général les mêmes que pour l'image d'origine).

Attention Il faut bien noter qu'après application du filtre, il faut retomber dans le domaine visible. Par exemple si l'on travaille en niveau de gris de 0 à 255, si après application du filtre, on trouve 267, il faudra redescendre à 255, et si l'on trouve un nombre négatif, il faudra remonter à 0.

5.2 Détection de contours

Une application intéressante du produit de convolution est de permettre la détection des contours d'une image. Cela correspond à chercher un filtre qui permet de déterminer les fortes variations de couleurs.

Pour cela, il y a de très nombreux moyens, plus ou moyen efficaces selon la qualité ou le type d'image sur lequel on travaille.

Si l'on travaillait en dimension 1, une manière de déterminer les fortes variations serait simplement de dériver la fonction (en supposant qu'elle est dérivable). Les images étant en deux dimensions, on ne peut pas réaliser directement ce type de dérivation. On peut utiliser des dérivées partielles, mais malheureusement, elles dépendent de l'orientation de l'image.

Cela dit, on peut par cette manière, déterminer "assez facilement" les variations selon un axe particulier.

5.2.1 Illustration

Voici une illustration du principe énoncé précédemment en une dimension. On peut par exemple disposer de contour de ce type (figure 5.2)[1] :

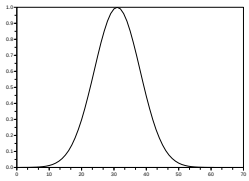

FIG. 5.2 – Pic

L'application de la dérivée donne la figure 5.3 :

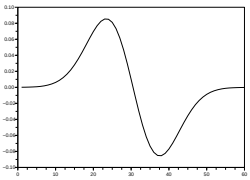

FIG. 5.3 – Dérivée du pic

On peut également avoir un contour en "escalier" (figure 5.4) dont la dérivée donne la figure 5.5.

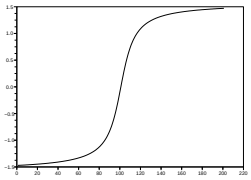

FIG. 5.4 – Escalier

On remarque ici que l'opération de dérivation rend bien compte de la forte variation.

[1] Ces graphiques ont été obtenues avec le logiciel libre Scilab

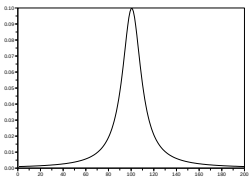

FIG. 5.5 – Dérivée de l'escalier

5.2.2 Détection suivant un axe

De la même manière que l'exemple d'application du chapitre formalisation, on peut disposer d'une approximation de la dérivée partielle selon un axe particulier.

– u(x+1,y) - u(x,y) selon l'axe x ;
– u(x,y+1) - u(x,y) selon l'axe y.

On peut ainsi déterminer facilement la matrice de convolution à appliquer à u pour réaliser cela.

Selon l'axe x, il faut appliquer :

$$h_x = \begin{pmatrix} 0 & -1 & 1 \end{pmatrix}$$

Exemple d'application On obtient ce type d'image :

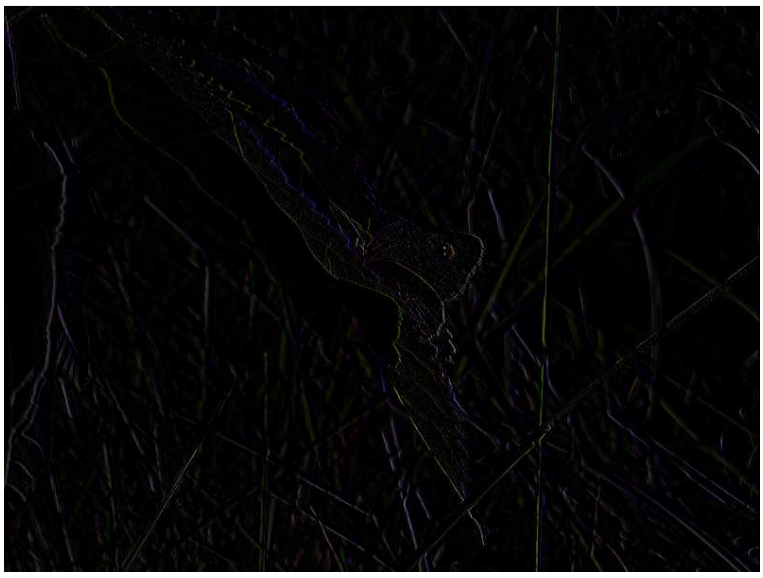

FIG. 5.6 – Détection de base selon x

Selon l'axe y, il faut appliquer la matrice de convolution :

$$h_y = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

On obtient l'image 5.7 :

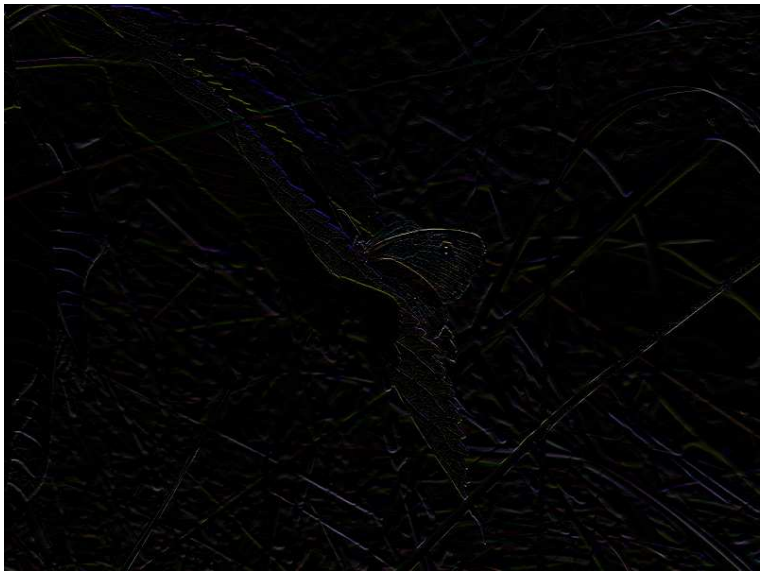

FIG. 5.7 – Détection de base selon y

On peut constater que la tige verte est quasiment invisible dans la détection selon y.

Un dernier exemple d'application sur une matrice composée uniquement de 0 et de 1 :

$$u = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Donne selon l'axe x (en transformant directement -1 en 0)

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Inconvénient

Cette méthode a deux inconvénients majeurs :
- la détection se fait selon un seul axe ;
- même selon un seul axe, il peut y avoir une détection que sur un bord (par exemple gauche sur l'exemple précèdent) ;
- la détection est très sensible au bruit et aux interférences.

Pour réduire les problèmes liés au bruit, nous allons présenter les filtres de Sobel et de Prewitt.

5.2.3 Opérateurs de Sobel et de Prewitt

Ces opérateurs utilisent une autre approximation de l'opérateur différentiel ∂.
Pour une image f de classe C^1 et pour tout $h > 0$, on a l'égalité :

$$\forall (x,y) \in \mathbb{R}^2, \frac{\partial f}{\partial x}(x,y) = \frac{f(x+h) - f(x)}{h} + o(1)$$

Mais on peut obtenir une meilleure approximation :

$$\frac{\partial f}{\partial x}(x,y) = \frac{1}{2}\frac{f(x+h) - f(x-h)}{h} + o(h)$$

On peut ainsi en déduire la matrice de convolution suivante :

$$h_x = \begin{pmatrix} -1/2 & 0 & 1/2 \end{pmatrix}$$

En appliquant directement cette matrice, la luminosité peut beaucoup diminuer (de deux fois par rapport aux matrices précédentes).

Pour permettre d'éliminer les bruits ou les parasites, ces opérateurs réalisent une moyenne de ces opérateurs sur le pixel du dessus et sur le pixel du dessous.

On réalise donc une succession de deux filtres :
- le filtre $(1,1,1)$ (Prewitt) ou $(1,2,1)$ (Sobel) selon l'autre axe pour lisser ;
- le filtre $(-1,0,1)$ selon l'axe pour dériver.

On calcule directement le produit de convolution entre les deux matrices (ici selon l'axe y) :

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & x & 1 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -x & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & x & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Comme les coefficients autour de la matrice sont égales à 0, on peut les enlever et juste garder la matrice de convolution (ce qui évite des calculs supplémentaires) :

$$\begin{pmatrix} -1 & -x & -1 \\ 0 & 0 & 0 \\ 1 & x & 1 \end{pmatrix}$$

Ainsi, les deux opérateurs de Sobel s'écrivent :
- Selon l'axe x :

$$h_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

– Selon l'axe y :

$$h_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Cette matrice donne plus d'importance à la ligne centrale, tandis que l'opérateur de Prewitt, défini comme suit, donne autant d'importance aux 3 lignes :

– Selon l'axe x :

$$h_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

– Selon l'axe y :

$$h_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Exemple d'application : Filtre de Sobel selon x (figure 5.8) :

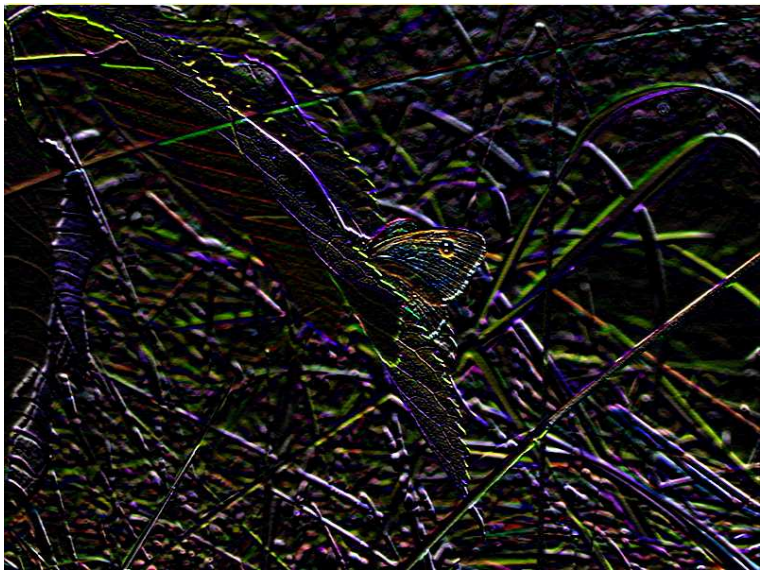

FIG. 5.8 – Filtre de Sobel selon x

Filtre de Sobel selon y (figure 5.9) :
On note une nette amélioration de la qualité par rapport aux filtres précédents.

5.2.4 Amplitude et norme

Pour un peu corriger le problème de la direction selon les axes, nous faisons intervenir la norme du gradient. On rappelle que le gradient est défini pour une fonction f de classeC^1 défini sur un ouvert Ω de $\mathbb{R}^2$ à valeur dans $\mathbb{R}$.

$$\forall (x,y) \in \Omega, \nabla(f)(x,y) = \left(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y) \right)$$

FIG. 5.9 – Filtre de Sobel selon y

Suivant la norme que l'on prend ($\|.\|_2$, $\|.\|_\infty$), on peut obtenir les transformations non linéaires suivantes :

- $\|\nabla(f)(x,y)\|_2 = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$
- $\|\nabla(f)(x,y)\|_1 = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$
- $\|\nabla(f)(x,y)\|_\infty = max(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$

Exemple Voici un exemple avec la norme 1 (figure 5.10) :

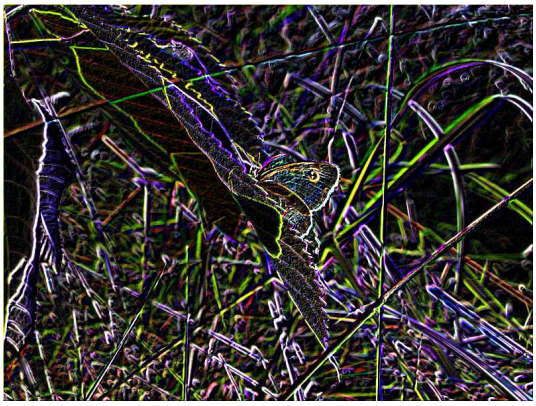

FIG. 5.10 – Norme des filtres de Sobel

Illustration du gradient Si l'on dispose d'une image (en vue 3d) de ce type (figure 5.11) :

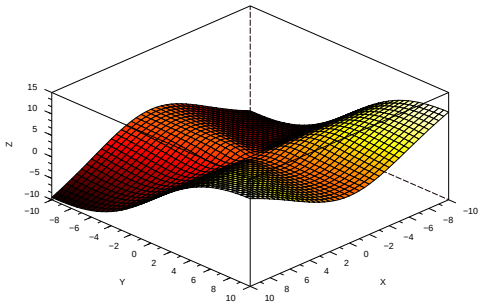

FIG. 5.11 – Image 3d

Nous pouvons réaliser une courbe de niveau comme suit (figure 5.12) :

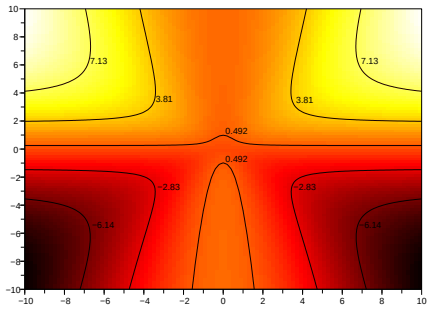

FIG. 5.12 – Contour 3d

Le gradient à la position (x, y) est normal aux courbes de niveaux.

5.2.5 Opérateur Laplacien

L'opérateur laplacien Δ peut servir dans la détection des contours, notamment grâce à une propriété intéressante qui le rend constant même après un changement de base orthonormée.

Théorème 3 *Soit $(\vec{n}, \vec{n_\perp})$ une base orthonormée de $\mathbb{R}^2$. Soit f une application définie de $\mathbb{R}^2$ dans $\mathbb{R}^2$ de classe C^2. Alors :*

$$\forall (x,y) \in \mathbb{R}^2, \Delta(f)(x,y) = \frac{\partial^2 f}{\partial \vec{n}^2}(x,y) + \frac{\partial^2 f}{\partial \vec{n_\perp}^2}(x,y)$$

Preuve

Soit θ l'angle de rotation du vecteur $\vec{n}$.

$$\frac{\partial f}{\partial \vec{n}}(x,y) = \frac{\partial f}{\partial x}(x,y)\cdot\cos(\theta) + \frac{\partial f}{\partial y}(x,y)\cdot\sin(\theta)$$

D'où :

$$\frac{\partial^2 f}{\partial \vec{n}^2}(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y)\cdot cos^2(\theta) + \frac{\partial^2 f}{\partial y^2}(x,y)\cdot sin^2(\theta) + 2\cdot\frac{\partial^2 f}{\partial x\partial y}(x,y)\cdot\cos(\theta)\cdot\sin(\theta)$$

La dérivée selon $\vec{n_\perp}$ peut simplement s'obtenir en utilisant le résultat d'avant et en ajoutant $\pi/2$ à θ.

On obtient ainsi :

$$\begin{aligned}\frac{\partial^2 f}{\partial \vec{n}^2}(x,y) + \frac{\partial^2 f}{\partial \vec{n_\perp}^2}(x,y) &= \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y)\\ &= \Delta(f)(x,y)\end{aligned}$$

Utilisation

On dispose d'un opérateur qui ne dépend pas de l'orientation de l'image, nous allons maintenant voir en quoi il peut être utile.

La fonction $\frac{\partial f}{\partial \vec{n}}$ désigne la dérivée selon l'axe du gradient n. De plus, plus le gradient est élevé, plus on est susceptible de se trouver sur une forte variation de couleur. On cherche donc à déterminer les extrêmes de la fonction précédente, cela se faisant en cherchant les passages par 0 de $\frac{\partial f^2}{\partial \vec{n}^2}$.

De manière classique, on utilise la nullité du laplacien et non de cette expression. Cela était discutable.

Approximation du laplacien

On utilise en général une approximation des dérivées partielles secondes suivantes :

$$\forall(x,y), \frac{\partial^2 f}{\partial x^2}(x,y) = \frac{1}{h}\left(f(x+h,y) + f(x-h,y) - 2\cdot f(x,y)\right) + o(h)$$

D'où :

$$\forall(x,y), \Delta(f)(x,y) = \frac{1}{h}\left(f(x+h,y) + f(x-h,y) + f(x,y+h) + f(x,y-h) - 4\cdot f(x,y)\right)$$

Filtre laplacien

De la même manière que l'exemple d'application, on peut déterminer à partir de l'égalité précédente un masque approchant le laplacien :

$$h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Illustration

Voici un exemple d'utilisation (figure 5.13) :

FIG. 5.13 – Filtre laplacien

5.3 Flou et débruitage

Lorsque l'on prend en photographie quelque chose, il peut y avoir plusieurs phénomènes qui "bruitent" l'image, qui la rend de moins bonne qualité que ce que l'on peut voir. Il y a une différence entre l'image mesurée et la grandeur physique réelle.

Il peut y avoir plusieurs raisons à cela, par exemple, une mauvaise stabilité de l'appareil photo pouvant rendre l'image un peu flou. Une luminosité insuffisante ou encore des poussières sur l'objectif.

Il existe plusieurs modélisations mathématiques de la notion de bruit. On utilise en général un bruit additif qui peut être de plusieurs types :

- bruit gaussien ;
- bruit laplacien ;
- bruit impulsionnel et d'autres.

5.3.1 Bruit, définition utilisée pour les preuves

Nous n'allons pas définir le bruit de manière classique (qui utilise souvent des notions de probabilités), cette définition nous permettra de faire quelques preuves de manière plus simple.

Soit $f \in \mathbb{I}_c$ une image originale de classe C^0, on dit que $g \in \mathbb{I}_c$ est bruitée

par rapport à f si il existe $r > 0$ et $\epsilon > 0$ tel que :

$$\forall (u,v) \in \mathbb{R}^2, \left| \iint_{\mathcal{C}_r(u,v)} (g(x,y) - f(x,y))\, dx\, dy \right| < \epsilon$$

5.3.2 Brève introduction au débruitage

Soit $f \in \mathbb{I}_c$ l'image originale et $g \in \mathbb{I}_c$ l'image bruitée par rapport à f toutes deux de classe C^0. Soit $(u,v) \in \mathbb{R}^2$ le point que l'on souhaite corriger.

Comme f est continue, alors :

$$\forall (x,y) \in \mathbb{R}^2, f(x,y) = \frac{1}{\|\mathcal{C}_r(u,v)\|} \iint_{\mathcal{C}_r(u,v)} f(x,y)\, dx\, dy + o_{r->0}(1)$$

(avec $\|\mathcal{C}_r(u,v)\| = \iint_{\mathcal{C}_r(u,v)} dx\, dy$)

Ainsi, on peut avoir l'approximation :

$$-\epsilon + o(r) < -\|\mathcal{C}_r(u,v)\| \cdot f(x,y) + \iint_{\mathcal{C}_r(u,v)} g(x,y)\, dx\, dy < \epsilon + o(r)$$

Soit :

$$\left| \frac{1}{\|\mathcal{C}_r(u,v)\|} \iint_{\mathcal{C}_r(u,v)} g(x,y)\, dx\, dy - f(x,y) \right| < \frac{\epsilon}{\|\mathcal{C}_r(u,v)\|} + o(r)$$

Donc, intégrer sur une petite boule a tendance à diminuer l'erreur mais à égaliser un peu les pixels dans une même boule. Lorsque l'on intègre, on donne autant d'importance à chaque pixel, mais on peut très bien changer les importances de chaque pixel (par exemple, le point central plus important...)

Définition 7 (Filtre flou) *h est un masque de convolution de flou si et seulement si il existe $r > 0$ tel que :*

$$\forall (u,v) \in \mathbb{R}^2, \frac{1}{\|\mathcal{C}_r(u,v)\|} \|(h * g) - f\|_\infty < \frac{\epsilon}{\|\mathcal{C}_r(u,v)\|} + o(r)$$

Tel que :

$$\iint_{\mathcal{C}_r(x,y)} h(x,y)\, dx\, dy = \|\mathcal{C}_r(u,v)\|$$

et tel qu'il soit invariant par rotation autour du point $(0,0)$.

$$\forall \theta \in \mathbb{R}, h(r_\theta) = h$$

5.3.3 Formule générale

De manière générale, on cherche un masque h tel que :

$$\sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} h(i,j) = 1$$

Et étant invariant par rotation : $h(i,j) = h(i,-j) = h(-i,j) = h(-i,-j)$

Et en régle générale de centre plus important que le reste :

$$\forall (i,j) \in \mathbb{Z}^2 h(0,0) > h(i,j)$$

L'image corrigée sera donc simplement $h * f$.

5.3.4 Filtre moyenneur

Le masque classique est défini par :

$$h = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

Ce filtre est souvent utilisé pour faire des lissages. On peut également en définir de taille $n \cdot n$ et de valeur constante $1/n^2$.

Exemple 1 On a appliqué le filtre sur le papillon (figure 5.14).

FIG. 5.14 – Lissage du papillon

Exemple 2 Si on applique sur une image fortement bruitée (figure 5.15), nous obtenons la figure 5.16.

Comme la figure initiale est fortement bruitée, le résultat n'est donc pas vraiment de bonne qualité.

Exemple 3 Nous avons appliqué le filtre sur la figure bruitée 5.17, nous obtenons en retour la figure 5.18 (on a appliqué le filtre deux fois) :

5.3.5 Filtre moyenne pondérée

Un autre masque parfois utilisé donnant plus d'importance au pixel central est :

$$h = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

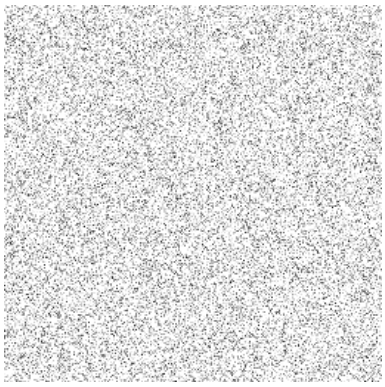

FIG. 5.15 – Un bruit additif

FIG. 5.16 – Lissage du bruit

5.3.6 Flou gaussien

On utilise souvent comme filtre de flou la fonction gaussienne définie par :

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Vous pouvez obtenir plus d'information sur cette fonction page 37 section 7.1.

Ensuite, on peut définir pour tout σ une matrice de convolution $h \in \mathbb{I}_d$ (centrée en 0) de taille impaire $2k+1$.

$$\forall (x,y) \in [[-k,k]], h(x,y) = g_\sigma(x,y)$$

NB Il faut noter que plus la taille de la matrice est importante, plus le nombre de calcul sera important (il existe de nombreuses techniques plus complexes permettant de diminuer considérablement le nombre de calcul, nous ne les présenterons pas dans ce cours).

FIG. 5.17 – Papillon bruité

FIG. 5.18 – Lissage du papillon bruité

En général, la somme des coefficients de la matrice n'est pas égale à 1, ce qui fait que l'image résultat sera sombre. En effet, vous pouvez constater que si la somme des coefficients est inférieur à 1, l'image aura tendance à être sombre (comme avec la détection des contours), par contre, si elle est supérieur à 1, l'image aura au contraire tendance à être claire.

Il faut donc savoir gérer entre la taille du noyau[2] (ou la taille de la matrice) et σ afin d'obtenir un m qui ne change pas la luminosité. Cela pouvant se faire simplement par dichotomie.

Exemple d'application Voici avec une matrice de taille 17 et σ adaptée pour la luminosité (figure 5.19) :

Ici, les calculs hors borne retournent la couleur noire, on peut constater que les bords de l'image ont tendance à s'assombrir à cause de cela.

Voici le papillon flou corrigé, figure 5.20

[2]Dans la littérature de traitement d'image, on parle parfois de noyau pour parler de la matrice de convolution

FIG. 5.19 – Flou gaussien du papillon

FIG. 5.20 – Flou gaussien du papillon bruité

5.3.7 Application à d'autres filtres

Certains filtres comme nous l'avons vu dans la détection des bords ne sont pas efficaces lorsqu'il y a du bruit. C'est pourquoi on réalise parfois un débruitage avant d'appliquer ce filtre.

Le laplacien

Au lieu d'utiliser l'opérateur laplacien que nous avons vu, on en utilise parfois qui est composé avec une opération de débruitage.

$$h = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

On voit parfois également le masque suivant :

$$h = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

Sobel et Prewitt

Les opérateurs de détections de contours de Sobel et de Prewitt utilisent également un débruitage comme nous avons pu le voir dans la partie précédente.

5.4 Quelques filtres supplémentaires

Voici plusieurs filtres qui peuvent réaliser certains effets.

5.4.1 Rotation

On suppose que l'on souhaite effectuer une rotation d'une image d'un angle θ et centrée en 0.

Si f est une image continue, alors l'image tournée g est :

$$\forall (x,y) \in \mathbb{R}^2, g(x,y) = f(r_{-\theta}(x,y))$$

Soit :

$$\forall (x,y) \in \mathbb{R}^2, g(x,y) = f\begin{pmatrix} x \cdot \cos(\theta) + y \cdot \sin(\theta) \\ -x \cdot \sin(\theta) + y \cdot \cos(\theta) \end{pmatrix}$$

Dans le cas discret, il faut définir quel point de $\mathbb{Z}^2$ est le plus proche du point

$$\begin{pmatrix} x \cdot \cos(\theta) + y \cdot \sin(\theta) \\ -x \cdot \sin(\theta) + y \cdot \cos(\theta) \end{pmatrix}$$

On peut par exemple utiliser la fonction partie entière E. Dans ce cas, on peut définir l'image tournée par :

$$\forall (x,y) \in \mathbb{Z}^2, g(x,y) = f\begin{pmatrix} E(x \cdot \cos(\theta) + y \cdot \sin(\theta)) \\ E(-x \cdot \sin(\theta) + y \cdot \cos(\theta)) \end{pmatrix}$$

5.4.2 Amélioration des bords

Pour améliorer un peu la netteté des images selon l'axe x, on peut utiliser ce masque :

$$h = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Selon l'axe y, on peut utiliser le masque :

$$h = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

FIG. 5.21 – Amélioration des bords

Exemple Cela donne la figure 5.21

5.4.3 Gaufrage

Pour donner un effet de gaufrage, on peut utiliser un filtre linéaire.
– On applique premièrement le masque :

$$h = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

– Puis, pour une image où chaque canal est codé sur un entier de 0 à 255. On ajoute ensuite 128 à chaque canal.

En appliquant ce filtre sur le papillon, nous obtenons la figure 5.22 :

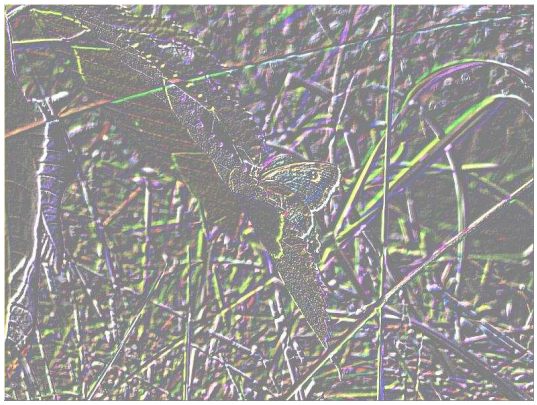

FIG. 5.22 – Effet gaufrage

5.4.4 Effet peinture

On peut obtenir un effet style peinture en appliquant le filtre suivant :

$$h = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

On peut voir un exemple d'application à la figure 5.23.

FIG. 5.23 – Effet peinture

5.4.5 Filtre de MDIF

Le filtre de MDIF est une composition du filtre de Prewitt et d'un lissage. Le masque est défini par :

$$h = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & -1 & 0 \\ 1 & 2 & 0 & -2 & -1 \\ 1 & 3 & 0 & -3 & -1 \\ 1 & 2 & 0 & -2 & -1 \\ 0 & 1 & 0 & -1 & 0 \end{pmatrix}$$

Vous pouvez voir une application de ce filtre à la figure 5.24.

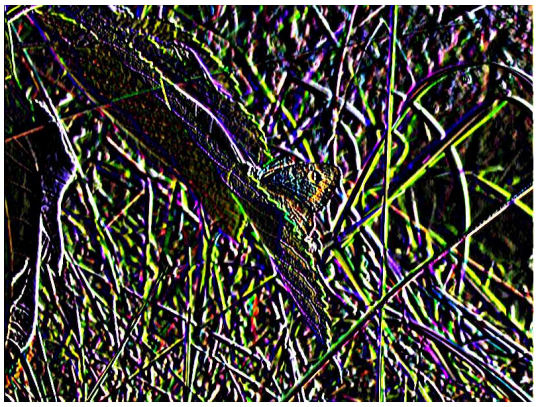

FIG. 5.24 – Filtre MDIF

Chapitre 6

Conclusion

J'espère que ce cours vous a permis d'en connaître un peu plus sur le traitement d'image même si certaines applications classiques n'ont pas été traitées.

Mais le cours évoluera certainement au cours du temps et j'essayerai de corriger et d'intégrer le plus d'éléments possibles.

Chapitre 7

Annexe

7.1 Fonctions gaussiennes

Les fonctions gaussiennes sont très utilisées en mathématiques et en physique.

Dans ce cours, nous utilisons particulièrement la fonction gaussienne définie en deux dimensions par :

$$\forall (x,y) \in \mathbb{R}^2, g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Vous pouvez voir une courbe de cette fonction à la figure 7.1. On la caractérise souvent par l'expression « courbe en cloche ».

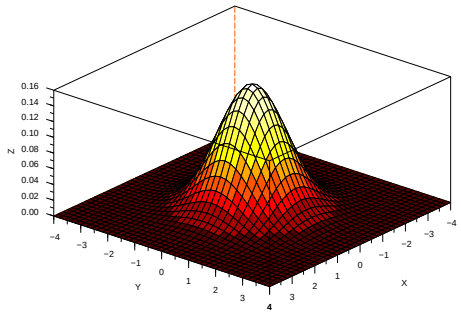

FIG. 7.1 – Fonction gaussienne

Le facteur correcteur $\frac{1}{2\pi\sigma^2}$ est utile pour que l'intégrale sur $\mathbb{R}^2$ de g_σ soit égale à 1.

L'intégrale de Gauss étant définie par :

$$\forall x \in \mathbb{R}, \int_{-\infty}^{\infty} \exp(-ax^2)\,dx = \sqrt{\frac{\pi}{a}}$$

On en déduit :

$$\begin{aligned}
\int_{-\infty}^{\infty} g_\sigma(x,y)\,dx\,dy &= \frac{1}{2\pi\sigma^2}\int_{-\infty}^{\infty}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)dx\,dy \\
&= \frac{1}{2\pi\sigma^2}\int_{-\infty}^{\infty}\exp(-\frac{x^2}{2\sigma^2})\,dx\cdot\int_{-\infty}^{\infty}\exp(-\frac{y^2}{2\sigma^2})\,dy \\
&= \frac{1}{2\pi\sigma^2}\cdot\sqrt{\pi\cdot 2\sigma^2}\cdot\sqrt{\pi\cdot 2\sigma^2} \\
&= 1
\end{aligned}$$

Table des figures