Notes de cours de IMA201a-b interpolation, transformation géométriques et restauration

Saïd Ladjal (said.ladjal@telecom-paris.fr)

Table des matières

Présentation de ce polycopié

Ce polycopié couvre les séances enseignées par Saïd Ladjal en IMA201 : Interpolation, transformations géométriques, filtrage et restauration. Ils complètent le cours et le résument. On peut s'y référer pendant les TPs.

Chapitre 1

Interpolation et transformations géométriques et filtrage

Introduction

Les images numériques peuvent être modélisées comme des tableaux de nombres (ou de vecteurs à trois dimension si l'on considère une image couleur). Faisant abstraction du caractère limité d'une image ont peut regarder une image à niveaux de gris comme une fonction allant de $\mathbb{Z}^2$ vers $\mathbb{R}$. Ainsi, un couple de coordonnées $(i, j) \in \mathbb{Z}^2$ représente une position dans le monde physique valant $ie_1 + je_2$, où e_1 et e_2 sont (le plus souvent) deux vecteurs d'un plan, de même norme et orthogonaux (voir échantillonnage bi-dimensionnel pour des cas différents pour le couple e_1, e_2).

Dans beaucoup de situations on peut être amené à devoir estimer la valeur de l'image en un point de coordonnées non entières. Le cas le plus courant étant de devoir afficher sur un écran ne possédant que 2 millions de pixels, une photographie possédant 10+ millions de pixels. Il y a peu de chances que la résolution horizontale et verticale de l'écran soient des diviseurs des résolutions de la photographie.

Ce chapitre se décompose en deux parties. La première traite la question du calcul de la valeur d'une image en un point non entier. Différentes méthodes sont proposées. Elles sont toutes présentées dans un cadre général unificateur. La seconde partie, très succincte, présente la manière d'appliquer une transformation géométrique à une image. Cette dernière partie s'appuie sur la première en disant que l'application d'une transformation géométrique se réduit à savoir calculer la valeur d'une image en des points non entiers. Un seul cas échappe à cette simplification, c'est le dézoom qui exige un filtrage des hautes fréquence avant calcul de la valeur interpolée afin d'éviter le repliement de spectre.

ATTENTION : Ceci est une version préliminaire. Notamment, pour les figures d'illustration, il vous suffit de vous reporter aux transparents des cours.

1.1 Interpolation

1.1.1 Position du problème et cadre générique de l'interpolation

Le problème que l'on se pose est le suivant [1]. On connaît une image numérique, notée I_e à valeurs réelles et définie sur $\mathbb{Z}^2$ (ou un sous-ensemble de type $[0 \dots N-1] \times [0 \dots N-1]$) et on veut donner une estimation de ce que devrait être la version définie sur $\mathbb{R}^2$ que l'on nomme I_c.

Le cadre général que nous adoptons est que l'on suppose que l'image I_c définie sur $\mathbb{R}^2$ a une certaine régularité et en déduire la valeur de $I_c(x, y)$ sous cette contrainte de régularité et connaissance les valeurs des $I_e(i, j)$ pour i et j entiers. Chaque régularité, donne lieu à un type d'interpolation. Nous étudions les types de régularité les plus utilisés.

Nous notons I_d la "fonction" [2] définie sur $\mathbb{R}^2$ et représentant I_e :

$$I_d = \sum_{(i,j)} I_e(i, j)\delta_{(i,j)}$$

où $\delta_{(i,j)}$ est le dirac centré en (i, j). Cette fonction I_d nous sera utile pour exprimer la fonction interpolée I_c comme une convolution entre I_d et un certain noyau interpolant.

1.1.2 Différentes régularités et les interpolations qui en découlent

Constante par morceau, interpolation au plus proche voisin

Si l'on impose que l'image soit constante par morceaux sur des pavés (de $\mathbb{R}^2$) du type $[i - \frac{1}{2}, i + \frac{1}{2}[\times[j - \frac{1}{2}, j + \frac{1}{2}[$ alors il est clair que

$$I_c(x, y) = I_e(\lfloor x + \frac{1}{2} \rfloor, \lfloor y + \frac{1}{2} \rfloor)$$

où $\lfloor \gamma \rfloor$ est la fonction partie entière de γ (qui à un réel associe le plus grand entier $\leq \gamma$).

L'interpolation ainsi construite s'appelle l'interpolation au plus proche voisin car le couple $(\lfloor x + \frac{1}{2} \rfloor, \lfloor y + \frac{1}{2} \rfloor)$ est le couple d'entiers le plus proche du couple (x, y) (on néglige ce qui se passe pour les demi-entiers).

On peut exprimer I_c comme une convolution par :

$$I_c = I_d * T_0$$

où T_0 est la fonction pavé :

$$T_0(x, y) = \begin{cases} 1 \text{ si } |x|, |y| < \frac{1}{2} \\ 0 \text{ sinon} \end{cases}$$

Linéaire par morceaux, interpolation bilinéaire

Si l'on impose que l'image soit linéaire par morceaux dans le sens suivant :

$\forall t \in \mathbb{R}$ les fonction $s \mapsto I(t, s)$ et $s \mapsto I(s, t)$ sont affines sur les intervalles $[i, i + 1]$

1. Les illustrations de cette parties sont dans les pages 1-20 des transparents
2. qui est plutôt une distribution.

Autrement dit, l'image est affine par morceaux sur les intervalles de bornes entières et ce en colonne comme en ligne.

Alors, si on note $i_0 = \lfloor x \rfloor$, $j_0 = \lfloor y \rfloor$, $x_F = x - i_0$ et $y_F = y - j_0$ les parties entières et fractionnaires de x et y, d'une part I_c doit être définie par

$$\begin{array}{rcccl} I_c(x,y) & = & x_F & y_F & \times I_e(i_0+1, j_0+1) \\ & + & (1-x_F) & y_F & \times I_e(i_0, j_0+1) \\ & + & x_F & (1-y_F) & \times I_e(i_0+1, j_0) \\ & + & (1-x_F) & (1-y_F) & \times I_e(i_0, j_0) \end{array}$$

Pour voir cela il suffit de faire une interpolation linéaire sur les lignes j_0 et j_0+1 (pour l'ordonnée x), puis une interpolation linéaire sur la colonne x (pour l'abscisse y). On obtient le même résultat si on commence par une interpolation linéaire sur les colonnes i_0 et i_0+1 puis une interpolation sur la ligne y.

On vérifie facilement que sur toute colonne et sur toute ligne, la fonction ainsi définie est encore affine par morceaux.

De plus, on peut facilement exprimer la fonction I_c à partir de la fonction I_e par

$$I_c = I_d * T_1 \tag{1.1}$$

où T_1 est une fonction définie par

$$T_1(x,y) = \begin{cases} (1-|x|)(1-|y|) \text{ si } |x|, |y| < 1 \\ 0 \text{ sinon} \end{cases}$$

($T_1(x,y) = t(x) \times t(y)$ où t est la fonction triangle telle que $t(-1) = t(1) = 0$ et $t(0) = 1$).

Polynomiale de degré trois, interpolation bicubique

Comme pour le cas de l'interpolation linéaire nous fixons la contrainte sur I_c en la présentant sous la forme de contrainte sur les lignes et les colonnes de l'image. Le cas bicubique étant plus difficile que le cas linéaire par morceaux nous traitons complètement le cas monodimensionnel.

Cas d'un signal Pour un signal monodimensionnel on demande à ce que les échantillons $f(n)$ soient interpolés par une fonction $f(x)$ qui est polynomiale de degré 3 sur les intervalles entiers $([n, n+1])$, continue et ses deux premières dérivées continues également (sur tout $\mathbb{R}$).

On construit $x \mapsto f(x)$ de la manière suivante :

— On constate que la fonction

$$h(x) = \begin{cases} \frac{1}{2}|x|^3 - x^2 + \frac{2}{3} \text{ si } |x| \leqslant 1 \\ -\frac{1}{6}|x|^3 + x^2 - 2|x| + \frac{4}{3} \text{ si } 1 \leqslant |x| \leqslant 2 \\ 0 \text{ si } |x| > 2 \end{cases} \tag{1.2}$$

vérifie bien les hypothèses de régularité.

— On en déduit que toute fonction $g(x) = \sum c_k h(x-k)$ vérifie les conditions de régularité. En effet, g est une combinaison linéaire de translatées de h (d'une translation entière). Elle vérifie donc les conditions et est toujours bien définie car, pour x fixé, il n'y a qu'un nombre fini de $h(x-n)$ qui soient non nuls (h est à support compact).

— On cherche une suite de coefficients c_k tels que

$$\forall n \in \mathbb{Z}, \quad f(n) = \sum_k c_k h(n-k) \tag{1.3}$$

et il suffira alors de prendre

$$f(x) = \sum_k c_k h(x-k)$$

pour obtenir une fonction qui vérifie les hypothèses.

Calcul des c_k :
On doit avoir, d'après l'équation 1.3,

$$(c * u)_n = f(n)$$

où u est la suite des échantillons de h aux points entiers. C'est-à-dire $u_0 = \frac{2}{3}$, $u_{-1} = u_1 = \frac{1}{6}$ et les autres valeurs nulles.

La suite (c_n) est donc le résultat du filtrage de la suite $f_n = f(n)$ par le filtre inverse de u. Le filtre u est à réponse impulsionnelle finie et son inverse est à réponse impulsionnelle infinie. Le calcul de chaque c_k nécessite donc un temps de calcul infini si on veut l'obtenir par une convolution. Une méthode bien connue pour inverser rapidement un filtre RIF est d'utiliser une équation de récurrence.

Si on note $F(z), U(z)$ et $C(z)$ les transformées en z de f_n, u_n et c_n respectivement, alors on doit avoir

$$C(z) = \frac{F(z)}{U(z)}$$

Avec

$$U(z) = \frac{1}{6}z + \frac{2}{3} + \frac{1}{6}z^{-1} = -\frac{1}{6a_0}(1 - a_0 z)(1 - a_0 z^{-1})$$

où $a_0 = -2 + \sqrt{3} \approx -0,26$ est l'une des deux racines de $x^2 + 4x + 1$, l'autre étant $1/a_0$.

L'application de l'inverse du filtre dont la TZ est $(1 - a_0 z^{-1})$ se fait simplement par application de la récurrence

$$w_n = f_n + a_0 w_{n-1} \text{ en faisant croitre } n$$

et produit, si f_n est causale et que l'on pose $w_{-1} = 0$ le résultat exacte pour tous les w_n calculés.

Pour appliquer l'inverse du filtre $(1 - a_0 z)$ il faut appliquer l'inverse du filtre $(1 - a_0 z^{-1})$ à la suite w_n renversée (car l'inverse n'est pas causal stable). Il faut appliquer la récurrence suivante

$$r_{n-1} = w_n + a_0 r_n \text{ en faisant décroitre } n$$

Il n'y a plus qu'à multiplier r_n par $-6a_0$ pour obtenir les c_n.

Il faut noter qu'il subsistera une petite erreur car, la suite w_n devrait être, même si f_n est à support fini, de support infini. Cependant on peut rendre cette erreur exponentiellement petite (voir les cours de première année).

Interprétation sous une forme de convolution :
Contrairement aux cas précédents, il n'est plus possible d'exprimer simplement le résultat final comme une convolution des échantillons par une certaine fonction simple. Cependant, si l'on rassemble tous les morceaux on arrive à l'expression suivante

$$f_c(x) = \sum_n f(n)\varphi(x-n)$$

où φ est la fonction définie par

$$\varphi(y) = \sum_k v_k h(y-k)$$

et v est l'unique suite (bornée) qui vérifie $v*u = \delta_0$ (le filtre inverse de $u = \ldots, \frac{1}{6}, \frac{2}{3}, \frac{1}{6}, 0, \ldots$)

Retour aux deux dimensions :

D'après l'étude qui précède, si nous voulons que les lignes et les colonnes de I_c (pour toute colonne et ligne d'indexe $t \in \mathbb{R}$) vérifient la condition d'être polynomiale de degré 3 sur les intervalles entiers et avoir une dérivée seconde continue, alors il faut au moins que cela soit vrai pour les lignes et colonnes entières. Il vient que pour n fixé on a

$$\forall x \in \mathbb{R}, \quad I_c(x,n) = \sum_k I_e(k,n)\varphi(x-k)$$

On fixe maintenant x et y varie, on doit avoir

$$I_c(x,y) = \sum_n I_c(x,n)\varphi(y-n) = \sum_n \sum_k I_e(k,n)\varphi(y-n)\varphi(x-k)$$

Soit,

$$I_c = I_d * \varphi_2 \tag{1.4}$$

où $\varphi_2(x,y) = \varphi(x)\varphi(y)$.

On vérifie réciproquement que la fonction définie par l'équation précédente vérifie les contraintes de régularité imposées.

Algorithme effectif :
Pour appliquer une interpolation bicubique il suffit de calculer les coefficients c_k pour chaque ligne. On obtient alors un tableau de la même taille que l'image où chaque ligne a été remplacée par les coefficients c_k. Sur ce tableau, on calcule pour chaque colonne k les coefficients mono-dimensionnels d'interpolation bicubique $c_{k,n}$. On obtient la valeur de l'interpolée bicubique au point non entier (x,y) par

$$I_c(x,y) = \sum_n \sum_k c_{k,n} h(y-n)h(x-k).$$

Comme la fonction h est de support 4, la formule précédente implique un maximum de 16 opérations pour obtenir une interpolation.

Remarquons qu'il est inutile de calculer tous les $c_{k,n}$ si on ne veut calculer que quelques interpolations. On peut par exemple, avec une bonne précision, ne calculer que les coefficients $c_{k,n}$ correspondant à une partie de l'image autour des points (x, y) qui nous intéressent (un fenêtre de taille 10 pixels entourant (x, y) suffit). Le calcul de tous les $c_{k,n}$ sur l'image n'est utile que comme un "investissement" si l'on veut calculer beaucoup de valeurs de l'interpolée.

Images bien échantillonnées : Interpolation de Shannon

Si une image numérique a été bien échantillonnée, le théorème de Shannon nous dit qu'elle peut être retrouvée de manière exacte en tout point (x, y) à partir de ses valeurs aux points entiers $I(k, n)$ par la formule :

$$I(x, y) = \sum_{k,n} I(k, n) Sinc(x - k) Sinc(y - n) \tag{1.5}$$

avec $Sinc(x) = \frac{\sin(\pi x)}{\pi x}$.

Cette formule pose un problème d'efficacité de calcul, car elle fait intervenir pour chaque (x, y) toutes les valeurs $I(k, n)$. Elle peut néanmoins être mise en pratique dans certains cas particuliers. Par exemple si les (x, y) qui nous intéressent sont de la forme $(m/2^d, l/2^d)$ où m et l sont des entiers, on peut utiliser la transformée de Fourier rapide pour calculer en l'ordre de $dN^2 \log(N^2)$ opérations toutes les valeurs des interpolées (où N est la taille de l'image). Pour cela il suffit de calculer la TFD de l'image. Ajouter des zéros autour pour atteindre une taille $2^d N$ et calculer la TFD inverse (voir illustration dans les transparents).

1.1.3 Conclusion

On a vu les principales méthodes d'interpolation. La qualité de l'interpolation augmente avec la régularité (constante par morceau, linéaire, cubique, Shannon). Dans la pratique, on choisit la méthode d'interpolation suivant la qualité voulue mais aussi le temps de calcul dont on dispose. Un bon compromis est l'utilisation de l'interpolation bilinéaire. L'interpolation bicubique donne des résultats très suffisants en un temps de calcul légèrement plus long et plus gourmand en usage mémoire du fait de la nécessité de stocker les coefficients $c_{n,k}$ (sous forme flottante).

1.1.4 Récapitulatif des temps de calculs

Pour l'interpolation au plus proche voisins, il faut une opération par pixel de la nouvelle image (copier la valeur spatiale la plus proche).

Pour l'interpolation bi-linéaire il faut quatre opérations par pixel.

Pour l'interpolation bicubique : Si l'image d'origine est de taille carrée $N \times N$ il faut créer le tableau des $c_{n,k}$ pour un temps de calcul d'à peu près $4N^2$ opérations. Puis pour chaque pixel de l'image il faut 16 opérations par pixel (on va chercher les 16 pixels les

plus proches et on les moyenne en utilisant le $c_{n,k}$. On cherche les 16 plus proches car la fonction h de l'équation (1.2) a un support de taille 4).

Pour l'interpolation Shannon le seul moyen de ne pas avoir un nombre d'opérations infini est de faire une transformée de Fourier discrète et il faut que l'image d'arrivée ait une taille multiple de la taille de l'image d'origine. Bien sûr on peut approximer le calcul de l'équation (1.5) en prenant quelques centaines de points autour de (x, y).

1.2 Transformations géométriques

Pour diverses raisons on peut vouloir appliquer une transformation géométrique [3] à une image. Par exemple, on voudrait l'afficher en plus grand sur un écran pour que l'utilisateur voit mieux les détails. On peut vouloir calculer sa projection sur un plan pour simuler une perspective dans le cadre d'un calcul de "texture" en synthèse d'image...

Nous résumons le problème sous la forme :

— On se donne une image numérique $I(k,n)$ pour k et n entre 0 et N et une transformation géométrique T.

— On veut calculer une nouvelle image numérique J telle que les objets dans J apparaissent comme la transformation de ceux de I par T. Ceci s'écrit : $J(k,n) = I(T^{-1}(k,n))$ où k et n sont entiers et parcourent l'ensemble des entiers tels que $I(T^{-1}(k,n))$ ait un sens.

Ainsi, on voit que si l'on sait calculer $I(T^{-1}(k,n))$ alors on aura résolu le problème. Nous avons ramené le problème à un **problème d'interpolation** :

Pour chaque (k,n) calculer $(x,y) = T^{-1}(k,n)$ et utiliser une méthode d'interpellation pour évaluer $I(x,y)$.

Il y a simplement une remarque et le cas particulier de la rotation que nous allons voir par la suite.

Cas de la diminution de la taille (dézoom)

Si la transformation T fait diminuer la taille de l'image alors un pixel sur la nouvelle image représente une surface plus grande qu'un pixel sur l'ancienne image. Utiliser une seule valeur de I pour représenter toute cette zone est une mauvaise approximation (penser au théorème d'échantillonnage de Shannon). Le résultat de cette mauvaise approche peut être vue sur la figure de la page 24 des transparents.

Comme l'approche naïve décrite ci-dessus s'apparente à un mauvais échantillonnage (l'image d'origine contient beaucoup d'informations et elle est sous-échantillonnée), il faut dans le cas du dézoom appliquer un filtre passe-bas à l'image I avant de procéder au calcul de $J(k,n) = I(T^{-1}(k,n))$. Ainsi la bonne approche pour appliquer une transformation qui réduit la taille de l'image est la suivante :

$$\tilde{I} = I * g$$
$$J(k,n) = \tilde{I}(T^{-1}(k,n))$$

L'image $\tilde{I}$ est obtenue par convolution de l'image I avec un masque passe-bas qui peut être, par exemple, un masque gaussien dont la taille est proportionnelle au facteur de dézoom qu'effectue T. C'est cette image $\tilde{I}$ que l'on interpole pour obtenir la valeur de la nouvelle image J.

Cas de la rotation

Une méthode particulièrement adaptée à la rotation a été proposée par L. Yaroslavsky [4]. Elle est basée sur la remarque suivante : Toute matrice de rotation d'un angle

3. Les illustrations sont aux pages 21-40 des transparents

4. M. Unser, P. Thévenaz, and L. Yaroslavsky, ?Convolution-based interpolation for fast, high- quality rotation of images, ? IEEE Trans. Image Process. 4, pp. 1371 ?1381, Oct. 1995

FIGURE 1.1 – comparaison entre 8 rotations de 45 degrés classiques utilisant l'interpolation bilinéaire (gauche) et 8 rotations de 45 degrés utilisant la méthode de rotation par transvections.

θ peut s'écrire sous la forme :

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan\frac{\theta}{2} \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -\tan\frac{\theta}{2} \\ 0 & 1 \end{bmatrix}$$

L'application d'une transformation dont la matrice est de la forme

$$\begin{bmatrix} 1 & -\lambda \\ 0 & 1 \end{bmatrix}$$

est très simple. En effet $T^{-1}(k,n) = (k + \lambda n, n)$. Ainsi chaque ligne n de l'image J ne dépend que de la même ligne de l'image I. Par ailleurs, lorsque n est fixé, l'application $k \mapsto k + \lambda n$ n'est qu'une simple translation. L'idée de L. Yaroslavsky est d'utiliser la transformée de Fourier pour appliquer la translation. Cela s'écrit :

$$J(.,n) = \mathcal{F}^{-1}\left[\mathcal{F}(I(.,n))(\xi).e^{-2i\pi\delta\xi}\right]$$

Avec $\delta = \lambda n$ le déplacement de la ligne numéro n. La formule précédente vient du fait qu'une translation se traduit dans le domaine de Fourier par la multiplication par une onde $\xi \mapsto e^{-2i\pi\delta\xi}$.

Le résultat de cette rotation (8 fois de suite de 45 degrés) est montré figure 1.1. Il est comparé à 8 rotations de 45 degrés utilisant l'interpolation bilinéaire. On voit bien que l'interpolation bilinéaire a introduit un flou que n'a pas introduit la méthode de rotation décrite ci-dessus.

1.3 Filtrage

Ce qui suit doit être lu accompagné des transparents : Le filtrage est un terme qui désigne diverses opérations sur les images. Nous allons différentes classes de filtrage. Le filtrage linéaire, le filtrage médian, ainsi que quelques filtres destinés au débruitage.

1.3.1 Filtrage Linéaire

On appelle filtrage linéaire toute transformation de l'image qui est linéaire et invariante par translation. C'est-à-dire que l'image résultante J s'écrit sous la forme $J = AI$ où I est l'image d'origine et A une opération linéaire (que l'on peut modéliser par une matrice carrée. De plus, comme on impose l'invariance par translation (si I subit une translation alors l'image J est translatée de la même quantité) la transformation linéaire A est une convolution.

Ainsi, tous les filtrages numériques s'écrivent sous la forme :

$$J = K * I$$

où K est le noyau de convolution.

Le choix d'un filtre linéaire ne dépend que du choix du noyau K. Voici quelques exemples :

— Filtre moyenneur dont le noyau est constant :

$$K = \begin{bmatrix} \frac{1}{k^2} & \cdots & \frac{1}{k^2} \\ \cdots & \cdots & \cdots \\ \frac{1}{k^2} & \cdots & \frac{1}{k^2} \end{bmatrix} \quad k \text{ est la taille du filtre}$$

il s'agit de l'exemple le plus simple de passe-bas.

— Le gradient : Il s'agit en fait de deux filtres dérivateurs. Le dérivateur en x et celui en y. (voir figure 1.2).

1.3.2 Les filtres pour le débruitage

Dans cette section nous présentons différents filtres en les regardant comme des outils de débruitage.

Le filtre médian

Ce filtre fait partie d'une plus grande théorie appelée "morphologie mathématique". Il ressemble au filtrage moyenneur. Pour chaque pixel de I, on considère l'ensemble des valeurs prises par I dans un voisinage du pixel. Au lieu de renvoyer la moyenne, on renvoie la médiane. C'est-à-dire la valeur telle qu'il y ait autant de pixels plus brillants que de pixels plus sombre que cette valeur dans le voisinage. Il est particulièrement adapté au filtrage du bruit "impulsionnel" (voir figure 1.3).

On peut en donner la formule suivant, où l'on note I l'image d'entrée, B est un ensemble de coordonnées proche de $(0, 0)$ (typiquement tous les couples $(-N, -N) \ldots (N, N)$ donnera un médian sur une fenêtre de taille $2N \times 2N$ autour de chaque pixel) et O est la sortie du filtre :

$$g_x(i,j) = f(i,j) - f(i-1,j) = f * [1, -1]$$

$$g_y(i,j) = f(i,j) - f(i,j-1) = f * \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

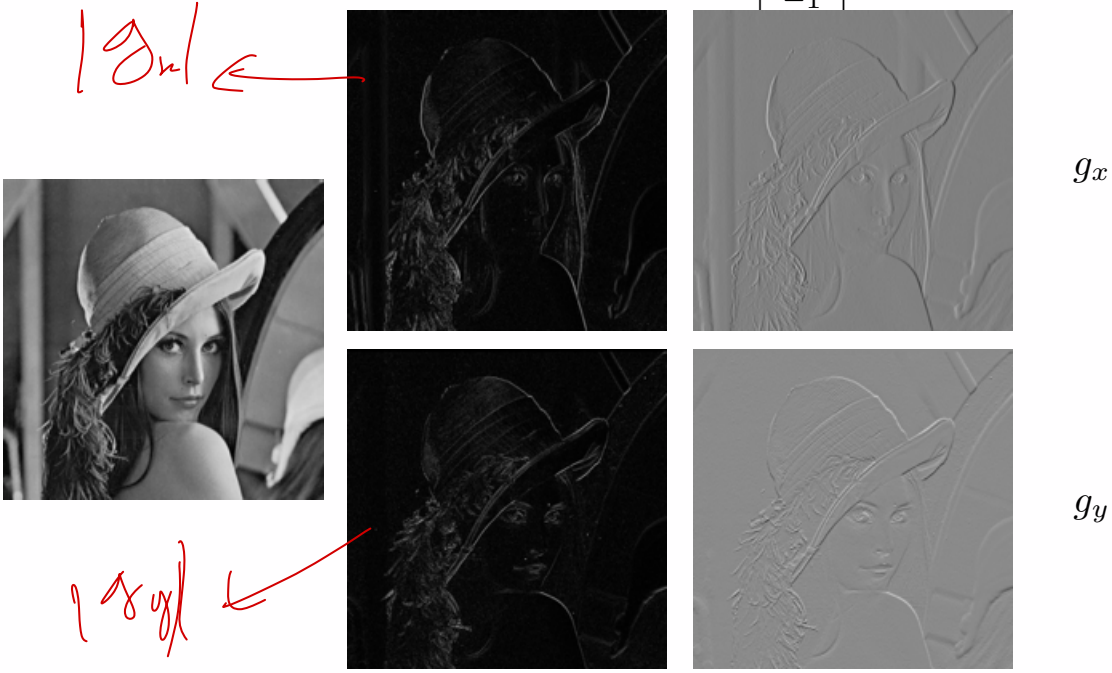

FIGURE 1.2 – Le gradient d'une image est un vecteur à deux dimension : la dérivée suivant la direction x et la dérivée suivant la direction y.

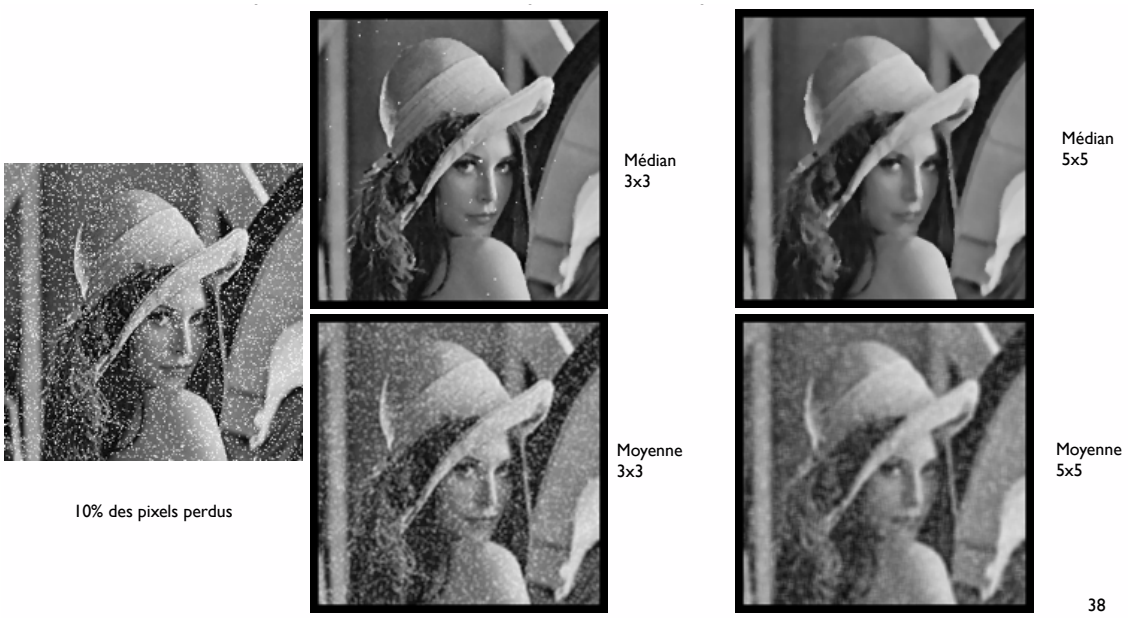

FIGURE 1.3 – comparaison du filtrage moyenneur et du filtrage médian sur du bruit impulsionnel

$$O(i,j) = median\left(\{f(i+n, j+m) | (n,m) \in B\}\right)$$

Et l'opérateur médian renvoie la valeur médianne expliquée ci-dessus.

L'ensemble B est appelé "élément structurant". C'est généralement une boule autour de $(0,0)$.

Le filtrage bilatéral

Ce filtrage consiste à renvoyer une moyenne des pixels de l'image pour chaque pixel de I examiné. Cependant cette moyenne est faite avec des poids qui dépendent du pixel examiné. Le poids de chaque pixel est d'autant plus grand que ce pixel est proche spatialement du pixel examiné. Il est aussi d'autant plus fort que le pixel a un niveau de gris qui est proche de celui du pixel examiné. (voir page 34 pour la formule et page 35 pour un exemple).

Ce filtre a deux paramètres : σ_I qui détermine ce que "proche" en niveau de gris signifie. Il doit être réglé en fonction de la force du bruit qui affecte l'image. L'idée est que plus le bruit est fort, plus des pixels qui étaient égaux dans l'image parfaite deviennent différents après ajout du bruit. Il faut donc augmenter σ_I pour les faire intervenir dans la moyenne.

Le second paramètre est σ_s il correspond à la taille de la fenêtre de recherche autour du pixel d'intérêt. Il doit être assez grand pour faire participer le plus de pixels à la moyenne (pour réduire le bruit) et être assez petit pour ne pas mélanger des zones de l'image qui ne sont pas semblables.

On peut en donner la formule suivante (I entrée, O sortie) :

FIGURE 1.4 – Le filtre bilatéral. Remarquer comme les bords sont bien conservés malgré le fait qu'il s'agisse d'une moyenne. Mais comme cette moyenne ne mélange que des pixels de valeur proche de celle de la position (i, j) il n'y a pas de flou à travers les bords bien contrastés.

$$O(i,j) = \sum_{(n,m)} e^{-\frac{((i-n)^2+(j-m)^2)}{2\sigma_s^2}} e^{-\frac{(I(i,j)-I(n,m))^2}{2\sigma_I^2}} \frac{1}{w(i,j)} I(n,m)$$

avec $w(i, j)$ un facteur de poids qui ramène la somme de tous les poids à 1 pour le calcul de la moyenne

$$w(i,j) = \sum_{(n,m)} e^{-\frac{((i-n)^2+(j-m)^2)}{2\sigma_s^2}} e^{-\frac{(I(i,j)-I(n,m))^2}{2\sigma_I^2}}$$

Le filtre des moyennes non locales

Le filtre bilatéral a le défaut de ne comparer que la luminosité des pixels pour décider de les faire collaborer l'un avec l'autre dans la moyenne pour obtenir une valeur plus fiable du pixel filtré. Ce défaut explique, par exemple, qu'une texture dont l'amplitude est comparable à celle du bruit sera totalement effacée par cet algorithme.

Pour résoudre ce problème, le filtre moyennes non locales a été inventé. Sa formule est très proche de celle du filtre bilatéral. La seule chose qui change est que la fenêtre de recherche est simplement un carré autour du pixel d'intérêt (pour réduire le temps de calcul). Mais surtout, le critère de ressemblance (pour donner plus de poids à un pixel dans la moyenne) n'est plus la différence de niveau de gris mais la différence entre deux voisinages de l'image autour des deux pixels (celui qui est examiné et celui qui va participer à la moyenne). Ce critère est bien plus fiable que la simple différence de niveaux de gris et permet d'être sûr de ne donner de l'importance qu'aux pixels réellement ressemblants. Le défaut étant qu'il ne débruite pas très efficacement les zones de l'image qui n'ont pas beaucoup de zones ressemblantes dans l'image elle-même (typiquement les bords). (voir pages 36 et suivantes pour la formule est des exemples).

Chapitre 2

Restauration

2.1 Position du problème

Lors de l'acquisition d'une image numérique plusieurs types d'erreurs se produisent qui font que l'image acquise n'est pas une parfaite réplique de la réalité. Le but de la restauration [1] est de réduire les défauts d'acquisition au minimum. Les principaux défauts que subit l'image sont les suivants :

- Le bruit : Comme toute mesure physique, la mesure de l'énergie lumineuse est entachée d'un bruit.
- Le flou : Il peut avoir différentes causes (optique, atmosphérique, bougé, diffraction). Il y a toujours, au moins, le flou du au fait que l'on intègre la lumière qui tombe sur toute la surface du pixel du capteur. Ce flou n'est pas a priori visible car précisément de la taille du pixel. Cependant on s'en rend compte quand on zoom une image ; L'image zoomée a toujours l'air plus lisse qu'une image prise à la résolution du zoom. (voir page 5)
- Le mauvais échantillonnage. L'optique de l'appareil photographique a une résolution théorique qui dépend des phénomènes de diffractions qui ont lieu dans l'appareil. Cependant, dans la plupart des situations la résolution du capteur numérique est inférieur à celle nécessaire pour bien échantillonner l'image continue (et passe-bas) qui se forme dans le plan du capteur. Un exemple de sous-échantillonnage est donné par la matrice de Bayer, qui n'échantillonne les canaux couleur que dans un rapport deux (le canal vert) ou quatre fois (bleu et rouge) inférieur à la résolution en pixels de l'appareil photographique.

2.1.1 Le flou

Le flou est le fait qu'un point de la scène se transforme en une tâche de lumière sur le capteur.

Le flou peut être du à plusieurs phénomènes :

- Flou optique : La focalisation de la lumière par la lentille n'est pas parfaite. La tâche de flou a alors une forme proportionnelle à la forme de l'ouverture.
- Flou de bougé : Du au mouvement des objets dans la scène ou alors le mouvement de l'appareil lui-même. Surtout si le temps de pause est long.

1. À lire accompagné des transparents du cours de restauration

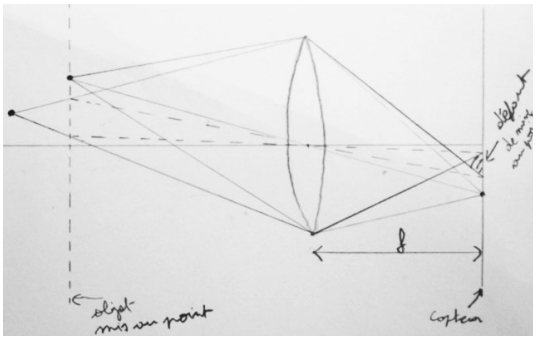

FIGURE 2.1 – Le flou optique : Si un point lumineux n'est pas dans le plan de mise au point (pointillés) son image est une tâche dont la taille est proportionnelle à l'ouverture.

— Flou de diffraction : Lorsque la lumière passe par un trou, la diffraction (caractère ondulatoire de la lumière) fait que l'image d'un point est une tâche correspondant à la transformée de Fourier de la forme de l'ouverture.
— Flou atmosphérique : Les variations de température de l'atmosphère entraine des chemins de optiques variables qui modifient le front d'onde. Cette modification du front d'onde (qui n'est plus un plan) cause un flou. On modélise le flou atmosphérique par une gaussienne. Dans les dispositifs optiques de haut niveau (téléscopes) des miroirs déformables permettent de compenser la distorsion du front d'onde `https://www.researchgate.net/figure/Wavefront-correctors-Top-left-Photo-sh fig5_279170359`.
— Flou du pixel : Le capteur intègre la lumière sur tout le pixel. Cela peut s'interpréter comme un flou car toute la lumière du pixel est confondue en une seule mesure.

Il faut remarquer que le flou de pixel est le flou de diffraction peuvent avoir un effet bénéfique : En effet, il permettent que l'échantillonnage (grille des pixels) se fasse sans trop de repliement spectral. Néanmoins on peut calculer que le flou de diffraction peut réduire la résolution d'un appareil. Par exemple, un appareil de 5000 par 4000 pixels photographie un tableau de 5m sur 4 m situé à 10m de l'appareil. L'angle de ce que voit un pixel est donc de $1mm/10m = 10^{-4}\, radians$. Si la longueur d'onde est $500nm = 5.10^{-7}m$ et que l'ouverture est de $2mm$ alors l'angle de la tâche de diffraction est de l'ordre de $5.10^{-7}/(2.10^{-3}) = 2,5.10^{-4} radians$. Ainsi l'appareil photo a perdu en résolution par rapport à son maximum théorique (dans un rapport 2,5).

Modélisation du flou Dans les cas les plus simples, le flou est le même dans toute l'image et tout point lumineux devient une fonction $h(x, y)$ (la tâche de flou). Si bien que l'image qui se forme sur le capteur g se calcule à partir de l'image parfaite f par la formule.

$$g = h * f, \text{ soit } g(x, y) = \iint_{u,v} f(x - u, y - v)h(u, v)dudv$$

où g est la fonction qui se forme sur le capteur, f l'image parfaite qui aurait du se former sans flou et h la tâche de flou.

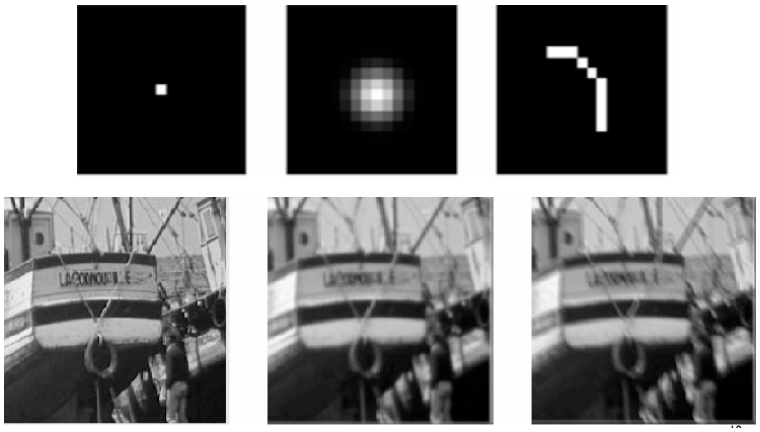

FIGURE 2.2 – différentes tâche de flou est leur effet sur une image. Le flou pixel n'est pas visible car, justement l'image (finie) qui est affichée a une taille de pixel qui est égale à la largeur du flou. Flou pixel, puis flou gaussien, puis flou de bougé.

2.2 Modélisation et résolution du problème

Dans la suite nous allons modéliser le processus d'acquisition de l'image par l'équation suivante :

$$g = Af + b$$
$f \in \mathbb{R}^{N^2}$: image de taille $N \times N$
$g \in \mathbb{R}^{N^2}$: image de taille $N \times N$
A : matrice de convolution carrée de taille $N^2 \times N^2$
$b \in \mathbb{R}^{N^2}$: réalisation d'un bruit gaussien

où f est l'image parfaite, g l'image acquise et b le bruit.

2.2.1 Restauration naïve

On peut penser que pour retrouver f à partir de g, il suffit d'inverser la matrice A. Cependant, en présence de bruit, l'inversion rehausse fortement le bruit. En effet, les opérateurs de flou sont des passe-bas. Et les images naturelles elles-mêmes contiennent beaucoup de basses fréquence et très peu d'énergie dans les hautes fréquences.

Ainsi, dans les hautes fréquences, l'information de l'image est faible et le flou la réduit encore plus. Le bruit, lui, a autant d'énergie à toutes les fréquences. L'application de l'inverse A multiplie par beaucoup la composante haute fréquence de g. Dans cette composante, l'énergie du bruit prédomine.

Dans la suite nous allons donner une description précise de ces phénomènes et nous dirons comment restaurer les images sans rehausser le bruit.

Aux figures 2.3 et 2.4 vous verrez le résultat de l'inversion A sur une image où $b = 0$ et un autre quand un bruit très faible est ajouté ($\|b\|^2 = 0.1$. Le bruit est pourtant très faible, mais le résultat de $A^{-1}g$ est une image visiblement fausse.

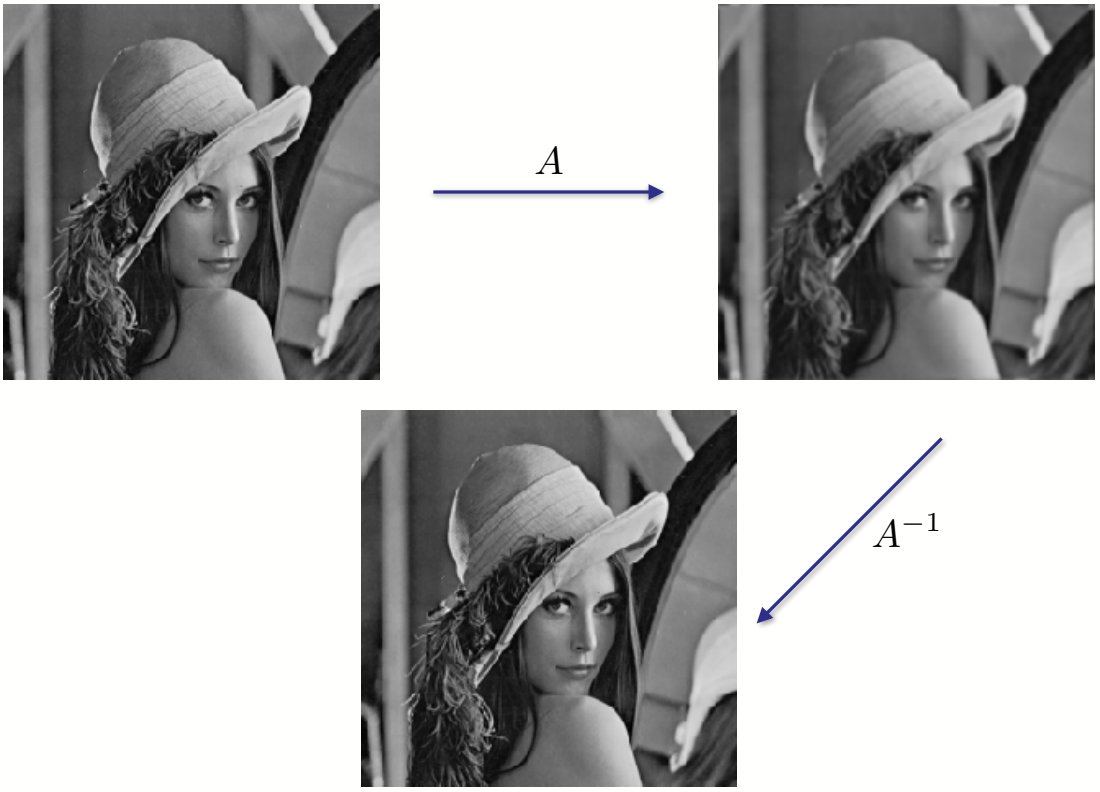

FIGURE 2.3 – Une image (haut gauche) subit un flou (haut droite), puis on lui applique l'opérateur inverse. Cela a l'air de fonctionner.

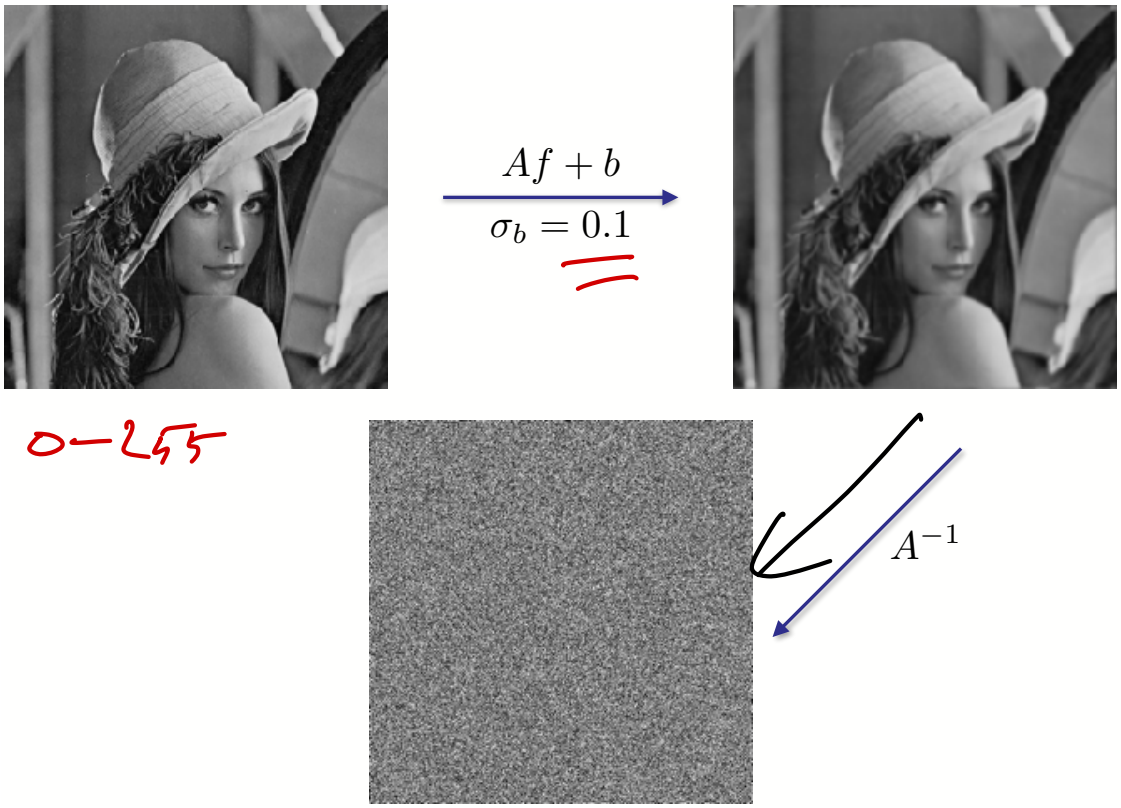

FIGURE 2.4 – Même chose que la figure 2.3, mais cette fois-ci du bruit a été ajouté juste après le flou (bruit de mesure). Le résultat est totalement inutilisable.

2.2.2 Restauration par Wiener

Le filtrage de Wiener consiste à trouver une méthode de restauration lorsque l'on connaît les propriétés statistiques du signal et du bruit ainsi que l'opérateur de dégradation (A) subi par le signal.

Afin d'introduire le filtrage de Wiener général, nous résolvons un problème de dimension 1. On suppose que X est une variable gaussienne de moyenne nulle et de variance σ_s^2. On observe la variable Y suivant le schéma

$$Y = \alpha X + B$$

où B est un bruit de variance σ_b^2. Le but est de trouver $\tilde{X}$ qui soit une approximation de X à partir de Y sous la forme :

$$\tilde{X} = \beta X \text{ t.q.}$$

et cela en minimisant l'erreur quadratique

$$\mathbb{E}(|\tilde{X} - X|^2).$$

α représente l'opérateur A, X représente f, Y l'image acquise g et β sera le meilleur "inverse" de A qui permet de ne pas trop rehausser le bruit.

Il est évident que si le bruit est nul, alors le mieux à faire est de prendre $\beta = 1/\alpha$.

La solution au problème que nous venons de poser est la suivante :

$$\beta = \frac{\overline{\alpha}}{|\alpha|^2 + \frac{\sigma_b^2}{\sigma_s^2}}$$

En effet,

$$\mathbb{E}(|\tilde{X} - X|^2) = |\alpha\beta - 1|^2\sigma_s^2 + |\beta|^2\sigma_b^2$$

SI on note $\gamma = \frac{\sigma_s^2}{\sigma_b^2}$ cette quantité est proportionnelle à

$$(1 + \gamma|\alpha|^2)(|\beta|^2 - 2Re(\gamma\alpha\beta/(1 + \gamma|\alpha|^2)) + CST = (1 + \gamma|\alpha|^2)\left|\beta - \frac{\overline{\alpha}\gamma}{1 + \gamma|\beta|^2}\right|^2 + CST$$

Donc le minimum est atteint pour

$$\beta = \frac{\overline{\alpha}\gamma}{|\alpha|^2\gamma + 1} = \frac{\overline{\alpha}}{|\alpha|^2 + \frac{\sigma_b^2}{\sigma_s^2}}$$

(nous supposons les variables complexes en prévision de la suite)

Cette équation s'interprète de la manière suivante : Plus le signal est faible par rapport au bruit, plus β diffère de $1/\alpha$. À la limite, quand σ_b devient très grand $\beta = 0$. C'est-à-dire que lorsque le signal est faible par rapport au bruit, la meilleure restauration est de renvoyer 0. Il n'y a plus assez d'informations dans Y pour retrouver X.

Cas général de la restauration de Wiener

Le cas général du filtrage de Wiener est donné par le modèle d'observation suivant :

$$Y = AX + B$$

où X et B sont des vecteurs gaussiens, c'est-à-dire, des variables aléatoires à valeurs dans $\mathbb{R}^N$. On suppose connue les choses suivantes

X v.a. $\in \mathbb{R}^N$
$\mathbb{P}(X) \sim e^{-\frac{1}{2}X^T C^{-1} X}$
$C = E(XX^T)$ matrice de covariance déf. pos.
$\mathbb{P}(B) \sim e^{-\frac{1}{2}\frac{\|B\|^2}{\sigma_b^2}}$ Cela signifie que le bruit est blanc (iid gaussien) de variance σ_b^2

On observe Y et on veut trouver $\tilde{X} = DY$ où D et est une opération linéaire. On souhaite que $\mathbb{E}(\|\tilde{X} - X\|^2)$ soit minimale. On pourrait montrer que le $\tilde{X}$ qui minimise cette quantité est aussi celui qui maximise la probabilité de X sachant l'observation Y. Cela mène à la formule suivante :

$$\tilde{X} = \underbrace{(A^T A + \sigma_b^2 C^{-1})^{-1} A^T}_{D} Y$$

Interprétation pour les images Pour appliquer cette approches aux images il est nécessaire de connaître la matrice C et l'énergie du bruit σ_b^2. Or, si une image est de taille 10^6 pixels (image de taille 1000×1000) alors la matrice C a 10^{12} coefficients ! Une simplification possible est de supposer que, ce qui est naturel, les images sont un processus invariant par translation. C'est-à-dire que la translatée d'une image naturelle est une image naturelle. On peut alors appliquer la théorie des processus SSL et il vient que la distribution des images, est gouvernées par la définition d'une densité spectrale de puissance, notée $\sigma_s^2(\omega)$ où ω parcourt les fréquences de Fourier.

Par ailleurs, si on suppose que A est une convolution alors A est diagonalisable en base de Fourier. De plus la transformée de Fourier d'un bruit blanc (B) est aussi un bruit blanc. Tout cela permet de réinterpréter la formule de D dans le domaine de Fourier et on arrive à la formule :

$$\hat{\tilde{f}}(\omega) = \frac{\overline{\hat{K}(\omega)}}{|\hat{K}(\omega)|^2 + \frac{\sigma_b^2}{\sigma_s^2(\omega)}} \hat{g}(\omega)$$
ω parcourt les fréquences de Fourier
$\sigma_s^2(\omega)$ puiss. du signal à la fréq.ω

(K est le noyau de convolution, c'est-à-dire que l'on suppose que $Af = K * f$) Qui donne une implémentation immédiate : Calculer la TF de g (observation), la multiplier point à point (parcourir) ω par $\frac{\overline{\hat{K}(\omega)}}{|\hat{K}(\omega)|^2 + \frac{\sigma_b^2}{\sigma_s^2(\omega)}}$. Dans la pratique on peut choisir $\sigma_s^2(\omega) = |\hat{g}(\omega)|^2$, ce qui revient à approximer la densité spectrale de puissance par la puissance de l'observation. Bien que $\hat{g}$ soit perturbé par le noyau de convolution, pratiquement, cette approximation donne de bons résultats.

Une autre possibilité est de poser $\sigma_s^2(\omega) = \frac{1}{\omega^\beta}$ où β est un exposant de décroissance souvent proche de 2.

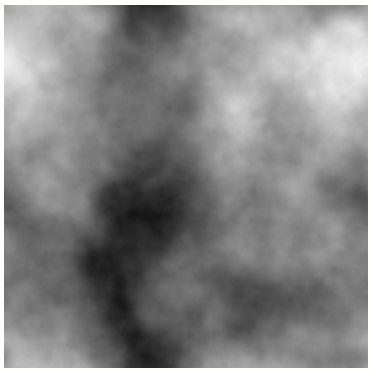

FIGURE 2.5 – Une image dont la densité de puissance est en $1/\omega^2$. On voit que cela ne correspond pas à une image naturelle (sauf une image de nuages).

Approche par minimisation d'énergie

Une autre approche possible de la restauration est de dire que l'image restaurée doit minimiser une énergie du type

$$E(\tilde{f}) = \underbrace{\|A\tilde{f} - g\|^2}_{\text{attache aux données}} + \lambda \underbrace{\int \|\nabla \tilde{f}\|^2}_{\text{régularité}} \tag{2.1}$$

Qui signifie que l'image doit bien expliquer l'observation (attache aux données) et qu'elle doit aussi être "naturelle" (régularité). La régularité choisie ici est de dire que la norme du gradient des images est faible.

La solution de ce problème de minimisation est

$$\hat{\tilde{f}}(\omega) = \frac{\overline{\hat{K}(\omega)}}{|\hat{K}(\omega)|^2 + \lambda\omega^2\sigma_b^2}\hat{g}(\omega)$$

Ce qui se réinterpète en Wiener comme : Nous avons choisi de dire que les images ont une densité spectrale de puissance en $1/\omega^2$.

Ainsi, chercher les image restaurées en minimisant l'énergie de l'équation (2.1) revient à supposer que les images sont un processus gaussien invariant par translation et dont la puissance spectrale est en $1/\omega^2$. Il est très facile de générer de telles images et on peut en voir une dans la figure 2.5. On voit que ce modèle est trop faible car il prend un modèle trop simpliste pour les images. C'est pourquoi il est utile de trouver des énergies plus adaptées aux images. Dans les prochains cours de la filière, de tels modèles plus avancés seront présentés.