M1 E3A - Voie André Ampère

453

Traitement d'Image

Enseignant:
Emmanuel Aldea
Thomas Rodet

Rédigé par:
Pierre-Antoine Comby

école normale supérieure paris-saclay

université PARIS-SACLAY

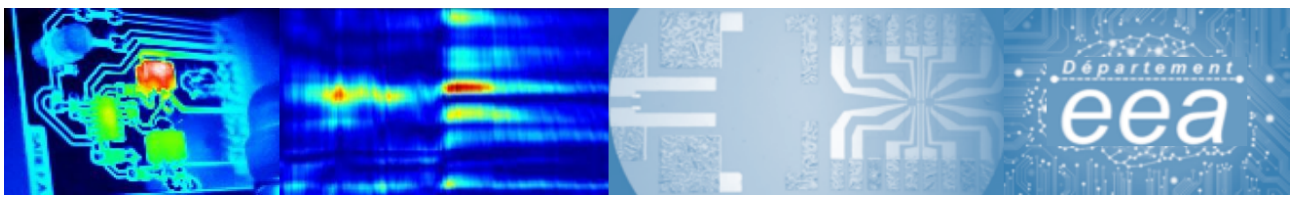

Table des matières

Chapitre 1

Introduction au traitement d'image

1 Introduction

1.1 Domaines d'applications

1.2 Définion

Définition

- Une *image* est une répresentaiton continue d'une fonction $f(x, y)$ qui relie f à l'intensité lumineuse du point (x, y).
- Une *Image numérique* Échantillonnage $I(x, y)$ dicret (Matrice 2D) de f qui relie $I(x, y)$ à l'intensité lumineuse d'une case (x, y) nommé *pixel*.
- Une Image numérique possède également une *quantification*: nombre de bits pour décrire une couleur /intensité d'un pixel.

FIGURE 1.1 – Echantillonage spatial

FIGURE 1.2 – Quantification

1.3 Notation et structure

Pour accéder aux pixel d'une image :

- w : nombre de colonne $i \in [0, w-1]$
- h : nombre de lignes $j \in [0, h-1]$

Ainsi $I(i,j)$ représente la valeur du pixel à la $i$ème colonne et $j$ème ligne.

Chaque pixel possède une valeur , quantifié qui peux correspondre à :

gris intensité comme scalaire en $[0, 2^n - 1]$ où n est la dynamique de l'image (la plupart du temps $n = 8$)

couleur Triplet R, G, B , chaque canal correspond à un 'niveau de couleur' codé de meme manière que le gris.

1.4 Amélioration d'image

Objectifs Réhausser le contraste d'une image pour faire apparaitre les objets distinctements. Pour cela on utilise des histogrammes

1.4.1 Histogrammes d'une image

Après quantification on peux compter les pixels de meme valeurs

Définition

- Un *histogramme* est une application :

$$H : \begin{cases} [0, N-1] \to [0, wh] \\ H(z) \mapsto card(\{(x,y) \in [0,w] \times [0,h] | I(x,y) = z\}) \end{cases}$$

- On a ensuite *l'histogramme normalisé* :

$$H_n : \begin{cases} [0, N-1] \to [01] \\ H_n(z) \mapsto H(z)/(wh) \end{cases}$$

  L'histogramme normalisé est une distribution de proba empirique.

- Et *l'histogramme cumulé normalisé*

$$H_{cn} : \begin{cases} [0, N-1] \to [0, 1] \\ H_{cn}(z) \mapsto \sum_{k=0}^{z} H_n(k) \end{cases}$$

  L'histogramme cumulé normalisé est une fonction croissante et une focntion de répartition empirique.

```
  int H[N]; // histogramme
float Hn[N]; // histogramme cumulé
float Hcn[N]; // histogramme cumulé normalisé
for (i = 0; i<N; i++){
  H[i] = 0;
  Hn[i] = 0;
  Hc[i] = 0;
}
// calcul de H
for (i = 0; i<w; i++)
  for (j = 0; j<h; j++){
  int val = I(i,j);
  H[val] = H[val] + 1;
}
// calcul de Hn,Hcn
Hc[0] = Hn[0] = H[0] / (w*h);
for (i = 1; i<N; i++){
  Hn[i] = H[i] / (w*h);
  Hcn[i] = Hcn[i-1] + Hn[i];
}
```

Listing 1 – Calcul des histogrammes

1.4.2 Transformation de l'histogramme

On souhaite conserver la relation d'ordre de l'histogramme pour $I(x_1, y_1) <= I(x_2, y_2) \implies I'(x_1, y_1) <= I'(x_2, y_2)$. Pour cela :

- Étalement de la dynamique via une transformation affine :

$$\boxed{z' = z'_{min} + (z'_{max} - z'_{min})\frac{z - z_{min}}{z_{max} - z_{min}}}$$

- Quantification $z' = round(z)$

Remarque: Généralement on fais entre 0 et 255 (8 bits).

Une autre possibilité et de faire de l'égalisation d'histogramme pour se rapprocher d'un histogramme cumulé uniformément croissant, cela peux passer par une transformation non linéaire.

$$z' = \frac{N-1}{wh}\sum_{i=0}^{z} H(i) = (N-1)H_{cn}(z)$$

1.5 Bruit

Apparition

- erreurs générant dans les pixels des valeurs différentes des valeurs réelles
- additif, multiplicatif, impulsionnel
- sources : capteur, transmission, interférences

Bruit gaussien

- bon modèle pour bruit capteurs :

$$\rho(\eta) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(\eta-\mu)^2}{2\sigma^2})$$

- exemple en astronomie : sur l'exemple on a $\sigma = 64$ (très élevé, pour des valeurs comprises entre 0 et 255). On moyenne donc sur plusieurs acquisitions pour augmenter la qualité (bruit gaussien de moyenne nulle). Pour 128 mesures, on divise l'écart-type par $\sqrt{128}$.

Bruit multiplicatif

- images radar, laser
- effets photochimiques (bruit "grain")
- $I(i,j) = I_0(i,j)\eta(i,j)$ avec $E[\eta] = 1$

Bruit convolutif

- effet de flou
- défaut de mise au point
- mouvement rapide de la caméra
- $I(i,j) = I_0(i,j) * g + \eta(i,j)$
- déconvolution (Fergus et al.) : il faut avoir le mouvement ("masque") pour pouvoir déconvoluer. Par minimisation de l'énergie, on essaye de trouver cette trajectoire de l'échec.

2 Filtrage Linéaire

Caractéristiques

- processus qui élimine une composante indésirable d'un signal
- parfois utilisé pour créer un effet artistique etc.
- en général associé à une perte d'information
- utilise le voisinage du pixel pour calculer sa nouvelle valeur classifications très variées :
  - filtrage linéaire et non-linéaire
  - filtrage passe-bas, passe-bande et passe-haut
  - etc.

2.1 Généralités

2.1.1 Formulation de Base (1D)

Définition

Un filtre linéaire réalise une opération de convolution en le signal $x(t)$ et la réponse impulsionnelle $h(t)$

$$(x \star h)(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau$$

Proposition

Le produit de convolution est :

- commutatif
- distributif
- associatif
- Équivalent à un produit classique dans le domaine fréquentiel :

$$x \star h = \mathcal{F}^{-1}[\mathcal{F}(x)\mathcal{F}(h)]$$

2.1.2 Utilisation en TI (2D discret)

Définition

Pour une image I et un filtre g:

$$(I \star g)(i,j) = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} I(i-n, j-m) g(n,m)$$

En général le support de g est compact, de dimension impaire.

2.2 Filtres courants pour réduire le bruit

Proposition *(Filtre moyenneur)*

Le niveau de gris du pixel central est remplacé apr la moyenne des niveaux de gris des pixels environnants.

$$w = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Proposition *(Filtre Gaussien)*

le niveau de gris du pixel central est remplacé par la moyenne des niveaux de gris des pixels environnants, pondérée par une Gaussienne 2D centrée dans ce pixel.

- filtre lisseur (donc passe-bas)
- taille du support en fonction du paramètre σ : $l = Int + (3\sigma)$
- élimine moins brutalement les hautes fréquences et préserve mieux les détails

Exemple: *pour* $\sigma = 0.625, l = 2$

$$w = 0.4 \times 10^{-2} \times \begin{pmatrix} 0.03 & 0.16 & 5.98 & 0.16 & 0.03 \\ 0.16 & 7.7 & 27.8 & 7.7 & 0.16 \\ 5.98 & 27.8 & 100 & 27.8 & 5.98 \\ 0.16 & 7.7 & 27.8 & 7.7 & 0.16 \\ 0.03 & 0.16 & 5.98 & 0.16 & 0.03 \end{pmatrix}$$

Proposition *(Filtre médian)*

- remplace par la valeur médiane de tous les pixels de la fenêtre d'analyse centrée sur le pixe
- filtre non-linéaire, plus coûteux
- très bien adapté au bruit impulsionnel

2.3 Filtre de sobel

Définition

On défini les dérivées discrètes:

$$\begin{cases} I_x[x,y] = I[x+1,y] - I[x-1,y] \\ I_y[x,y] = I[x,y+1] - I[x,y-1] \end{cases}$$

Remarque:

- Le calcul de la dérivé est equivalent a une convolution avec $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ resp. $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$
- Le filtre est sensible est bruit et lisse dans la direction orthogonale.

Proposition *(Sobel)*

On Utilise les deux filtres suivants pour déterminer les dérivées dans chacune des deux directions :

$$H_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad H_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

On en déduit alors la *magnitude du gradient* :

$$\|\nabla I\| = \sqrt{(I \star H_x)^2 + (I \star H_y)^2}$$

et son *orientation* :

$$\theta = \arctan(\frac{I \star H_y}{I \star H_x})$$

FIGURE 1.3 – Echantillonage spatial

2.4 Implémentation du filtrage

- complexité d'un algorithme : très importante en systèmes embarqués
- classes d'algorithmes :
  sub-linéaires en général $O(log(n))$: optimal (par exemple dichotomie)
  linéaires en $O(n)$. À noter que $O(nlog(n))$ est aussi considéré comme "rapide"
  autres plus longs
- séparabilité : permet de réduire le nombre d'opérations nécessaires.
- image intégrale : calcul linéaire en nombre de pixels (en calculant à l'aide des termes précédents)

$$\sum_{n=a}^{b}\sum_{m=c}^{d} I(n,m) = IN(b,d) - IN(a,d) - IN(b,c) + IN(a,c)$$

  La somme d'un rectangle : c'est donc seulement 4 opérations si on a fait le calcul de l'image intégrale auparavant.

Petit exo : quelle est la condition pour qu'un filtre 2D soit séparable ?

3 Filtrage pour la détection de contours

3.1 Filtre passe-haut : Laplacien

Généralité

Définition

On appelle laplacien l'opérateur différentiel :

$$\Delta f = \text{div}(\text{grad}(f)) = \nabla \cdot \nabla f = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

Le laplacien est étroitement lié aux phénomènes de diffusion, il représente le taux de variation moyen de $f(x)$ sur une sphère centrée en x quand la sphère varie.

Proposition ***(Laplacien d'une image)***

$$\nabla^2 I = \frac{\partial^2 I}{\partial x_i^2} + \frac{\partial^2 I}{\partial y_i^2} = I_{xx} + I_{yy}$$

Le masque qui correspond au calcul de I_{xx} est

$$L_{xx} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

On a donc le masque suivant

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

Calcul du masque

Développement de Taylor :

$$f(x+1) = f(x) + f'(x) + \frac{1}{2} f''(x)$$

Or, on a $f'(x) = f(x) - f(x-1)$ donc

$$f''(x) = 2(f(x+1) - 2f(x) + f(x-1)$$

Remarque:

- Ce masque est à symétrie centrale, ne dépend pas (en théorie) de la direction de référence. Ce filtre est invariant par rotation.
- Détection de contour : passage par zéro du Laplacien. Plus le contour est net, plus l'amplitude autour de zéro du laplacien est grande.
- Mais le Laplacien est *très* sensible au bruit.

Proposition

Pour limiter les effet du bruits on peux filtrer avec une gaussienne avant d'appliquer le laplacien.
Avec les propriétés de la convolution on peux directement convolué avec la dérivé seconde du laplacien (LoG) fonction "chapeau mexicain"

3.2 Diffusion dans une image

3.2.1 Diffusion isotrope

Le filtre Gaussien est équivalente au processus de diffusion **isotrope linéaire** suivant :

$$\frac{\partial u}{\partial t} = \mathrm{div}(d \nabla u)$$

Le processus de diffusion **non linéaire isotrope** s'écrit :

$$\frac{\partial u}{\partial t} = \text{div}(D(x,y)\nabla u)$$

Cependant, le flux de diffusion est toujours parallèle au gradient !
Choix du coefficient de diffusion :

$$D(x,y) = \frac{1}{1 + \frac{|\nabla u(x,y)|^2}{\lambda^2}}$$

λ est le paramètre de contraste.

Autre coefficient possible :

$$D(x,y) = e^{\frac{|\nabla u(x,y)|^2}{\lambda^2}}$$

3.2.2 Diffusion anisotrope

On veut diffuser perpendiculairement au gradient (selon les contours) :

$$\frac{\partial u}{\partial t} = \text{div}(T(x,y)\nabla u)$$

$T(x,y)$ tenseur de diffusion, dont les vecteurs propres perpendiculaires donnent les directions de diffusions, et les valeurs propres la diffusivité dans les directions respectives.

3.2.3 Filtrage (gaussien) bilatéral

Définition

Le filtre gaussien peux s'écrire

$$I_G(p) = \sum_{q \in S} I(q).G_{\sigma_s}(|p-q|)$$

Sur un contour, la distance spatiale entre p et q est faible, mais la différence de poids est grande. On risque donc de faire tout baver en ne pénalisant que la distance dans l'espace image.

Proposition

On fait intervenir un second terme qui pénalise la distance dans l'espace des "poids" des pixels. On empêche les pixels de valeur très différente d'intervenir dans le filtrage.

$$I_G(p) = \sum_{q \in S} I(q).G_{\sigma_s}(|p-q|).G_{\sigma_r}(|I(p)-I(q)|)$$

3.3 Application : Détection de droites

Une fois qu'on a détecté les contours (on a une image binaire), on veut essayer de détecter des droites.

Définition

On représente une droite donnée par un couple de valeur (r, θ).
L'espace de représentation des droites est donc $[0, 2\pi] \times [0, r_{max}]$
Équation des droites passant par (x_0, y_0) : $r = x_0 \cos\theta + y_0 \sin\theta$

On se fixe (x_0, y_0) et dans l'espace de représentation des droites, l'ensemble des droites passant par (x_0, y_0) est $r = x_0 \cos\theta + y_0 \sin\theta$ et donne une sinusoïde.
Avec un second couple (x_1, y_1), on a une autre sinusoïde.

Pour tous les points potentiellement sur des droites, on trace les sinusoïdes dans l'espace de représentation des droites, puis on regarde les méga-intersections.
A retenir : on peut utiliser cette méthode dès qu'on peut décrire les formes par des ensembles de paramètres. On fait un système de "vote" et c'est la droite qui a le plus de votes qui est élue Miss Droite.

3.4 Application : Detection de cercle

De la meme manière qu'on peux détecter les droites (2 paramètres) on peux détecter des cercles avec la transformée de Hough.

Définition

Dans cette méthode, un cercle est décrit par son équation cartésienne:

$$(x - a)2 + (y - b)2 = r2$$

où le point de coordonnées (a, b) est le centre du cercle et r en est le rayon.

Méthode Dans l'espace (a, b, r), un cercle est caractérisé par un point. L'ensemble des cercles passant par un point $M(x, y)$ donné forme un cône de sommet $(a = x, b = y, r = 0)$ et d'axe r. Un « bon candidat » correspond à l'intersection de plusieurs cônes.

Si le rayon est inconnu, la méthode de recherche consiste à construire une hypermatrice d'accumulation dont chaque cellule $A_{i,j,k}$ correspond à un cube de l'espace (a, b, r), en balayant tous les rayons possible de 1 pixel jusqu'à la dimension de l'image.

```python
    def calcul_acc_cercles(img_s, rad_min=5, rad_max=100):
    #init acc :
    [c_max, r_max] = img_s.shape
    r_min=1
    delta_r = 1
    N_r =  int((r_max-r_min+delta_r)/delta_r)
    delta_c = 1
    c_min = 1
    N_c =  int((c_max-c_min+delta_c)/delta_c)
    delta_rad = 1
    N_rad =  int((rad_max-rad_min+delta_rad)/delta_rad)
    acc = np.zeros((N_rad,N_r,N_c))
    print("Taille de l'accumulateur" + str(acc.shape))
    x,y = np.nonzero(img_s)
    for i in range(len(x)):
        for r in range(r_min, r_max+1, delta_r):
            for c in range(c_min, c_max+1, delta_c):
                if (x[i],y[i]) != (r,c):
                    rad = np.sqrt((r-x[i])**2 + (c-y[i])**2)
                    if rad < rad_max and rad > rad_min:
                        i_id = int((r-r_min)/delta_r)
                        j_id = int((c-c_min)/delta_c)
                        k_id = int((rad-rad_min)/delta_rad)
                        acc[k_id,i_id,j_id] += 1 /rad
    return acc
def cherche_N_maxima_cercles(accumulateur, exclusion_xy,
exclusion_rayon, N):
    accu2 = np.copy(accumulateur)
    [rad_max0, c_max0, r_max0] = accumulateur.shape
    liste_max = []
    for cercle in range(N):
        [rayon, w_0, h_0] = np.unravel_index(np.argmax(accu2),accu2.shape)
        rayon += 5
        w_0 += 1
        h_0 += 1
        accu2[rayon - exclusion_rayon : rayon + exclusion_rayon,
              w_0 -exclusion_xy : w_0 + exclusion_xy,
              h_0 - exclusion_xy : h_0 + exclusion_xy] =0
        liste_max.append([rayon, w_0, h_0])
    return liste_max
```

Listing 2 – Création de l'accumulateur de Hough et détection des maximums

4 Points d'intérêt

4.1 Nécessité de l'invariance en TI

Les contours : traitement peu coûteux, détection robuste de courbes paramétriques mais pas suffisamment invariant.

Objectif identifier des structures invariantes par rapport aux rotations, changements échelle, etc.

4.2 Détection de coins

4.2.1 Les bases : Détecteur de Harris

Définition

Un *coin* est endroit de l'image qui présente une forte variation d'intensité en deux directions différentes.

Stratégie : Le contenu d'un patch centré dans le coin devrait varier dans toutes les directions.
Déplacement du patch

- dans une zone homogène : pas coin
- le long d'un contour : pas coin
- près d'un coin : coin !

On cherche donc un changement d'intensité par shift de $(\Delta x, \Delta y)$:

Définition

On utilise la fonction d'estimation de coins :

$$E(x,y,\Delta x,\Delta y) = \sum_x \sum_y \underbrace{w(x,y)}_{support} \left[\underbrace{I(x,y)}_{\text{Intensité}} - \underbrace{I(x+\Delta x, y+\Delta y)}_{\text{Intensite shifée}} \right]^2$$

plus E est grand et plus on a de chance de trouver un coin.

Cette methode est lourde en calcul et peux etre rendu plus rapide :

Proposition

En faisant une approx au premier ordre de la fonction on a :

$$E(x,y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \underbrace{\begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix}}_{\text{tenseur de structure}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Remarque: Le tenseur de structure est composé des dérivée de l'image sur le patch moyennée. Les vecteurs propres indiquent les direction principales de variation de gradient dans le voisinage du point (voir l'ellipse du changement constant)

Méthode : On peux calculer λ_1 et λ_2 explicitement mais c'est lourd. On préfère utiliser :

$$R = \det(M) - \alpha tr(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

Et on choisi $\alpha \in [0.04; 0.06]$. Les valeurs propres intéressante sont alors des maximums locaux de R.

Algorithme

- Calcul des gradients gaussien de l'image
- Calcul du tenseur des structures
- Calcul de R
- Seuillage et suppression des valeurs non maximales.

Comment gérer le changement d'échelle ? Algorithme qui détecte en chaque endroit de l'image l'échelle qui ferait que cet endroit serait un coin.

4.3 Détecteur FAST

4.4 Détecteur SIFT

```python
def harris(image, sigma=10, kappa=0.04):
    derivees = gradient(image)
    d_x = derivees[0]
    d_y = derivees[1]
    noyau = noyau_gaussien3(sigma)
    #TODO: convol DoG
    d_xx_lissee = ndimage.convolve(d_x * d_x,noyau , mode = 'nearest')
    d_yx_lissee = ndimage.convolve(d_x * d_y,noyau , mode = 'nearest')
    d_yy_lissee = ndimage.convolve(d_y * d_y,noyau , mode = 'nearest')

    determinant = d_xx_lissee * d_yy_lissee -d_yx_lissee * d_yx_lissee
    trace = d_xx_lissee + d_yy_lissee

    image_harris = determinant - kappa * trace * trace
    return image_harris

def maxlocal(image_harris, seuil,size_patch=10):
    """ Array*float -> Array """
    output = np.zeros(image_harris.shape)
    voisinage = np.zeros((3,3))
    for ligne in range(size_patch,image_harris.shape[0]-size_patch):
        for colonne in range(size_patch,image_harris.shape[1]-size_patch):
            current_val = image_harris[ligne, colonne]
            voisinage = image_harris[ligne - 1: ligne + 2, colonne - 1:colonne + 2].copy()
            voisinage[(1,1)] = 0
            output[ligne, colonne] = np.logical_and(current_val > seuil,
                                                    current_val > np.amax(voisinage))
    print(output)
    return(output)

def coord_maxlocal(image,seuil,size_patch=3):
    #seuil=np.mean(image_extrema)
    """ Array -> Array """
    x,y = np.nonzero(maxlocal(image,seuil))
    return np.array((y,x)).T

def get_corners(pict):
    image_harris = harris(pict,sigma=3,kappa=0.06)
    corners = coord_maxlocal(image_harris,np.mean(image_harris))
    return corners

pict1 = np.array(Image.open("set1-1.png").convert('L'),dtype='float')
pict2 = np.array(Image.open("set1-2.png").convert('L'),dtype='float')
corners1=get_corners(pict1)
corners2=get_corners(pict2)
```

5 La couleur

5.1 Les descripteurs

En niveaux de gris, le descripteur le plus simple est la luminosité. Un autre descripteur possible (Local Binary Pattern LBP) prend le voisinage 3x3 d'un pixel et fait un seuil des valeurs des pixels par rapport au pixel central.

En couleurs, c'est plus complexe car il y a 3 canaux.

La couleur n'est pas forcément un atout car l'information supplémentaire n'est pas nécessairement un atout. En général, les algorithmes de traitement de couleurs sont construits pour des applications particulières : par exemple, suppression fond vert.

Capteurs RGB on peut difficilement réaliser un système permettant de séparer les canaux pour les envoyer sur 3 capteurs différents. On réalise donc une matrice du type :

R	V	R	V	R	V
V	B	V	B	V	B
R	V	R	V	R	V
V	B	V	B	V	B

En 1 point donné, on interpole entre les données voisines pour les différents canaux. Il y a plus de capteurs verts car c'est la couleur à laquelle l'oeil humain est le plus sensible.

Avantage : l'image RGB a donc la taille d'une image en niveaux de gris.

Inconvénient : sur les contours, on aura des aberrations à cause des interpolations. Par exemple, à la frontière entre une zone rouge et une zone verte, on fera apparaître du jaune.

Sources lumineuses on caractérise la lumière de manière spectrale, et on a don une grande variétés de spectres d'émission selon les sources. Donc en ne changeant ni l'objet, ni l'observateur, on peut avoir de grandes différences dans ce qui sera observé en fonction de l'éclairage "ambiant".

Perception l'important n'est pas ce qui compose l'environnement mais la manière dont on le perçoit. Par exemple : un damier sur laquelle une ombre est projetée, le cerveau humain sait que les carrés blancs sont toujours plus foncés que les gris, mais ce n'est pas forcément vrai entre un carré blanc "à l'ombre" et un carré gris dans la lumière.

Descripteurs principe de la trichromie : 3 couleurs primaires permettent de reproduire la quasi-totalité des couleurs

- Observateur standard RGB 1931 : il existe une partie négative de la partie rouge, qui n'est pas cohérente avec la théorie additive.

- Observateur standard CIE 1931 :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  Le Y est proche du V, le Z est proche du B.

5.2 Seuillage optimal méthode de Otsu

la méthode d'Otsu est utilisée pour effectuer un seuillage automatique à partir de la forme de l'histogramme de l'image1, ou la réduction d'une image à niveaux de gris en une image binaire. L'algorithme suppose alors que l'image à binariser ne contient que deux classes de pixels puis calcule le seuil optimal qui sépare ces deux classes afin que leur variance intra-classe soit minimale.

Proposition

On cherche le seuil t tel que :

$$t = \arg\max_t \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$$

Avec :

- $\omega_1(t) = \sum_{i=0}^{t} H_n(i)$ et $\omega_2(t) = \sum_{i=t+1}^{N-1} H_n(i)$
- $\mu_1(t) = \frac{1}{\omega_1(t)} \sum_{i=0}^{t} i \cdot H_n(i)$. et $\mu_2 = \frac{1}{\omega_2(t)} \sum_{i=t+1}^{N-1} i \cdot H_n(i)$
- .

6 Estimation

À partir d'une image on cherche à estimer un ensemble θ de paramètres (déplacememt, état d'un d'un système)

Source du problème On ne pas ignorer les outliers et déterminer les paramètre du modèle. Les methodes de tyoe moindres carré sont très sensibles aux outliers à cause de la fonction d'erreur quadratique associée.

Différentes approches sont possibles :

6.1 Analyse de l'ensemble des résidus

On utilise des variation de la méthode des moindres carrés :

Proposition *(Least Median of Squares)*

On remplace la somme par la médiane :

$$\theta = \arg\min_\theta \ \text{med}\ \rho(r_i)$$

Proposition ***(Least Trimmed Squares)***

On tri les résidus et on en sélectionne M où $N/2 < M < N$.

$$\theta \arg\min \sum_{i=0}^{M} \rho(r_i)$$

Une autre alternative est d'utilisé une autre fonction de cout (cf UE 451),qui doit rester symétrique définie positive

6.2 RANSAC

Définition

" RANSAC, abréviation pour RANdom SAmple Consensus, est une méthode pour estimer les paramètres de certains modèles mathématiques. Plus précisément, c'est une méthode itérative utilisée lorsque l'ensemble de données observées peut contenir des valeurs aberrantes (outliers). Il s'agit d'un algorithme non-déterministe dans le sens où il produit un résultat correct avec une certaine probabilité seulement, celle-ci augmentant à mesure que le nombre d'itérations est grand. L'algorithme a été publié pour la première fois par Fischler et Bolles en 1981"

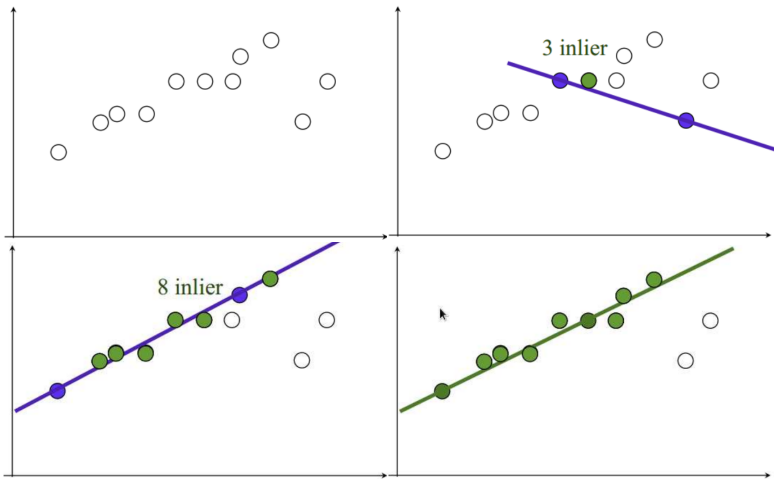

FIGURE 1.4 – Méthode RANSAC

Algorithme
entrées :

```
    data   - un ensemble d'observations
    modele - un modèle qui peut être ajusté à des données
    n      - nb min de données nécessaires pour ajuster le modèle
    k      - nb max d'itérations de l'algorithme
    t      - seuil d'appartenance au modèle
    d      - seuil de données nécessaire pour valider le modèle
```

sorties :

```
    meilleur_modèle - les paramètres du modèle qui correspondent le mieux aux données
    meilleur_ensemble_points - données à partir desquelles ce modèle a été estimé
    meilleure_erreur - l'erreur de ce modèle par rapport aux données


itérateur := 0
meilleur_modèle := aucun
meilleur_ensemble_points := aucun
meilleure_erreur := infini
tant que itérateur < k
    points_aléatoires := n valeurs choisies au hasard à partir des données
    modèle_possible := paramètres du modèle correspondant aux points_aléatoires
    ensemble_points := points_aléatoires

    Pour chaque point des données pas dans points_aléatoires
        si le point s'ajuste au modèle_possible avec une erreur inférieure à t
            Ajouter un point à ensemble_points

    si le nombre d'éléments dans ensemble_points est > d
        (ce qui implique que nous avons peut-être trouvé un bon modèle,
        on teste maintenant dans quelle mesure il est correct)
        modèle_possible := paramètres du modèle réajusté sur ensemble_points
        erreur := ecart entre données et modèle.
        si erreur < meilleure_erreur
            (nous avons trouvé un modèle qui est mieux que tous les précédents,
            le garder jusqu'à ce qu'un meilleur soit trouvé)
            meilleur_modèle := modèle_possible
            meilleur_ensemble_points := ensemble_points
            meilleure_erreur := erreur

    incrémentation de l'itérateur

retourne meilleur_modèle, meilleur_ensemble_points, meilleure_erreur
```

Listing 3 – Algorithme de la méthode randsac

6.3 La segmentation

poly de l'ENSTA très bien fait

La segmentation consiste à regrouper les différents pixels de l'image en un nombre (donné) de région (peut aussi consister a une taille de région maximale, les deux, etc...). En pratique on réalise une partition de l'image

6.4 Segmentation par découpage

Définition

L'idée des algorithmes de type « Split & Merge » est de produire automatiquement une partition initiale en régions petites (Split), qui vont ensuite croître en se regroupant (Merge)

- La partition initiale (Split) est réalisée en *divisant récursivement* l'image en régions de tailles identiques lorsqu'un certain critère d'homogéneité n'est pas satisfait: R *est divisée* (ex: si $\sigma_R > seuil$)
- Lors de cette phase, le graphe d'adjacence, ou Region Adjacency Graph (RAG) est créé : à chaque région est associé un sommet du graphe, et des arêtes relient les sommets correspondants à deux régions qui se touchent.
- La *phase de regroupement* (Merge) utilise le RAG pour modifier la partition initiale : pour chaque sommet R du RAG, on cherche s'il existe un sommet R' voisin dans le RAG et de valeur suffisamment proche, et si c'est le cas, on les fusionne: $RetR'$ sont fusionnées si $\mu_R - \mu_{R'} < seuil$

Phase de découpage Lors de la phase de découpage on forme le quadtree et le graphe d'adjacence :

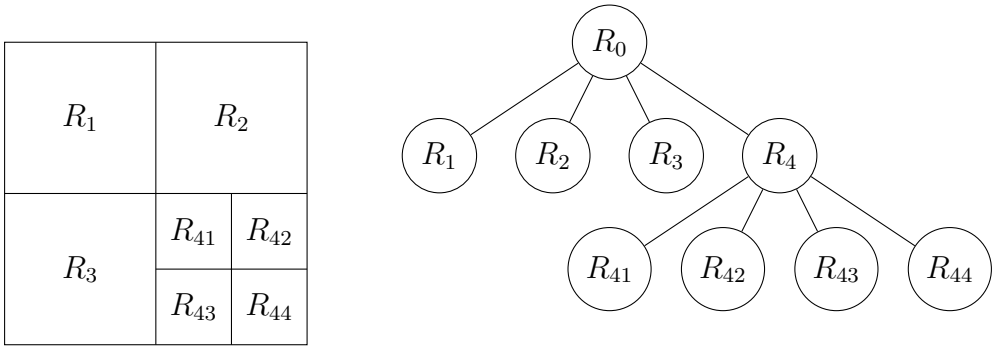

FIGURE 1.5 – Quadtree et graphe d'adjacence

Remarque: L'algorithme de découpage est implantable très facilement de manière récursive

6.5 Segmentation par optimisation

Proposition

Cette fois ci on cherche à déterminier la fonction f bidimensionnelle, constante par morceaux (une valeur , une région) proche de l'image I , avec une partition simple du point de vue géométrique. Pour cela on utilise la fonctionnelle de cout :

$$K = \sum_i \iint_{R_i} (I(x,y) - f_i)\mathrm{d}x\mathrm{d}y + \mu \sum_j \int_{\Gamma_j} dl$$

où Γ_j sont les portions de contour entourant la région de la région

Remarque: Il n'existe pas de solution directe au problème de minimisation. Il y a deux techniques pour approcher la solution :

- méthodes variationnelles sur des courbes fermées
- méthodes markoviennes par itération à partir d'une segmentation initiale

méthode variationnelles On utilise une fonction de cout $E = E_i + E_e$ tel que

- $E_i = \int_0^1 \alpha(C'(s))^2 ds$ pénalise la longueur du contour ("snake")
- $E_e = \int_0^1 -\nabla I(x(s), y(s)) ds$ favorise l'alignement sur les forts gradients.

Technique markoviennes On teste les changements de pixel/sous-région d'une région à une autre, et on prend le meilleur (ie qui minimise la fonctionnelle). Pour cela on part souvent d'une sur-segmentation.

7 La classification

Objectif :

- Obtenir une représentation simplifiée mais pertinente des données originales
- Mettre en évidence les similiratité entre les objets.

Problématique de la visualisation des données Il est difficile de se représenter la classification de manière "brute" au dela de 5 paramètres on peux utiliser l'analyse en composante principale (par réduction de la dimension) ou l'analyse linéaire discriminante.

7.1 Analyse en composantes principales

- indispensable quand le nombre de variables est très grand
- analyse de la variabilité / dispersion des données
- objectif : décrire à partir de q $<$ d dimensions cette variabilité
- réduction des données a q nouveaux descripteurs
- visualisation si q = 2 ou q = 3
- interprétation des données : liaisons inter-variables

Algorithme

1. recentrage des données $\mathbf{X} = (\mathbf{x} - \mu)^T$.
2. Calcul de la matrice de covariance Σ.
3. diagonalisation de Σ et classement par valeur propres croissantes.
4. sélection des q premiers vecteurs propres C_k
5. Calcul des valeur réduites a_i qui remplacent x_i par $a_{ik} =< x_i, C_k >$

Remarque: Le PCA ne prend pas en compte la notion de classe. on peux aussi utiliser l'analyse linéaire discriminante.

7.2 Analyse linéaire discriminante

Algorithme

1. recentrage des données $\mathbf{X} = (\mathbf{x} - \mu)^T$
2. Calcul de la matrice de covariance Σ.
3. Calcul de la matrice de covariance interclasse $\mathbf{B}$.

$$\mathbf{B} = \sum_{k=1}^{c} n_k \frac{(\mu_k - \mu)(\mu_k - \mu\rangle)^T}{n}$$

   où k représente la classe
4. diagonalisation de $\Sigma^{-1}\mathbf{B}$ et classement par valeur propres croissantes
5. sélection des q premiers vecteurs propres C_k.
6. Calcul des valeurs réduites $\mathbf{a_i}$ qui remplacent x_i par $a_{ik} =< x_i, C_k >$
7. classification d'une nouvelle observation par la distatnce au centroïde le plus proche.
8. classification linéaire : médiane entre les centroïdes.

7.3 Intégration de la connaissance

7.4 Classification non supervisée - K-means

on cherche à déterminer des "clusters" dans les observations. En minimisant

$$\min_{\mu_1 \ldots \mu_k} \sum_{i=1}^{k} \sum_{x_j \in C_i} (x_j - \mu_i)^2$$

Fonctionnement

```
choose mu_i , i in [1 k] // initialisation
while (any mu_i changes){
  assign all x_j to closest mu
  update mu_i , i in [1 k]
}
```

À chaque itération la fonction objectif diminue. c'est rapide et parallèlisable.

Chapitre 2

Inversion de donnée en imagerie

le poly distribué est très bien fait, ici il n'y aura que des prise de note et l'essentiel du cours

1 Philosophie et difficultés

1.1 Introduction

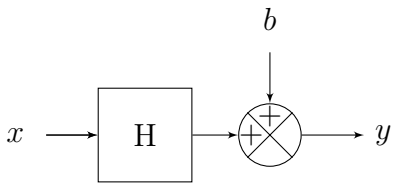

FIGURE 2.1 – Modélisation du problème direct

Méthode On fait des hypothèse sur x pour déterminer $\hat{x}$ qui permette de reconstituer un y proche de celui mesuré.

On a une connaissance parfaite des hypothèses que l'on a fait.

1.2 Problème mal posé

Définition

Les *Condition de Hadamard* permettent de savoir si un problème est bien posé.

- L'existence d'une solution quelques soit l'ensemble des donneés $\mathcal{Y} = Im(H)$
- L'unicité: $\text{Ker}(H) = \{0\}$
- Continuité :lorsque l'erreur δytend vers 0 ,δx tend aussi vers 0.

1.3 Discrétisation et linéarisation

Pour $x \in \mathbb{R}^M$et $y \in \mathbb{R}^N$ on considère que H est un opérateur linéaire.

Proposition

On note $p = rg(H)$

- $p = N = M$ Alors H bijectif, $\hat{\mathbf{x}} = H^{-1}\mathbf{y}$.
- $p < M$ pas d'unicité mais on a :

$$\hat{\mathbf{x}} = (\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1})\mathbf{y}$$

- $p > M$ pas d'existance mais on peux trouver l'inverse généralisé

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

Conditionnement de la matrice En ajoutant une erreur $\delta\mathbf{x}$ a$\hat{\mathbf{x}}$ on peux calculer comment la matrice H "amplifie le bruit"

Définition

À partir de l'inverse généralisé on a :

$$\|\delta x\| \leq \left|\left|\left|(\mathbf{H}^T\mathbf{H})^{-1}\right|\right|\right| \left|\left|\left|\mathbf{H}^T\right|\right|\right|$$

avec $|||\mathbf{H}||| = \sqrt{\max\{Sp(\mathbf{H})\}}$ Alors on défini le nombre de condition:

$$\delta x \leq c\delta y$$

Avec :

$$c = \sqrt{\frac{\lambda_{max}}{\lambda_{min}}}$$

Si il y a un mauvais conditionnement, le bruit (qui est presente sur toutes les composantes de la base modale) est amplifié de manière disproportionnées sur certaine composantes.

Décomposition en valeur singulière tronquées On réduit la matrice à ces plus grandes valeurs propres pour réduire le conditionnement

$$\tilde{\mathbf{H}} = \mathbf{U_t}\mathbf{\Lambda_t}\mathbf{V_t}$$

L'estimateur devient :

$$\hat{\mathbf{x}} = (\tilde{\mathbf{H}^T}\tilde{\mathbf{H}})^{-1}\tilde{\mathbf{H}^T}\mathbf{y}$$

2 Quelques méthode d'inversion classique

2.1 Le filtre de Wiener

Le principe du filtre de Wiener consiste à déterminer un filtre donc d'appliquer une opéartion liénaire invariante afin de séparer le bruit des données. Wiener a démontrer que dans le cas d'un

signal stationnaire on peux trouver un filtre optimum dans le sens ou c'est le meilleur filtre qui sépare le bruit de l'objet si on connait l'autocorrélation des données γ_{yy} et l'intercorrélation entre les données et l'objet que l'on cherche γ_{yx}.

Proposition *(Relation de Wiener Hopf)*

Pour $p \in \mathbb{Z}$on a :

$$\sum_{k\in\mathbb{Z}} g(k)\gamma_{yy}(p-k) = \gamma_{xy}(p)$$

Où g est le filtre disctret optimum permettant d'estimer au mieux x. LE critère choisi est la minimisation de l'espérance du carré de l'erreur de prédiction.

Démonstration : cf. Poly TR ■

Proposition

De facon pratique on détermine lefiltre dans l'espace de fourier :

$$G(\nu) = \frac{\Gamma_{xy}(\nu)}{\Gamma_{yy}}$$

2.2 Estimateur des moindres carrés

Proposition

L'estimateur des moindres carré cherche à minimiser la norme quadratique :

$$\hat{\mathbf{x}}_{MC} = \arg\min \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

2.3 Estimateur des moindres carrés régularisé

cf. UE 451 et poly

On veux améliorer le conditionnement de la matrice.

Proposition

On modifie la fonction de cout des moindres carrés

$$Q_{MCR} = \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu\mathcal{R}(\mathbf{x})$$

2.3.1 Régularisation quadratique

Plusieurs régularisation classiques sont possibles :

- Rappel à un objet connu
$$\mathcal{R}(x) = (\mathbf{x} - \mathbf{x}_\infty)^T(\mathbf{x} - \mathbf{x}_\infty)$$
- Terme séparable
$$\mathcal{R}(x) = \mathbf{x}^T\mathbf{x}$$
- Terme de différences (mesure de régularité)
$$\mathcal{R}(x) = \sum_i (x_{i+1} - x_i)^2 = \mathbf{x}^T\mathbf{D}^T\mathbf{D}\mathbf{x}$$

2.3.2 Régularisation convexe différentiable

Pour pénaliser de moins fortes valeurs on peux choisir une autre fonction de cout comme la fonction de Hubert (ou terme L_2L_1)

Définition

On appelle fonction de Huber

$$\phi_s(\tau) = \begin{cases} \tau^2 & |\tau| < s \\ 2s|\tau| - s^2 & |\tau| \geq s \end{cases}$$

Et sa généralisation vectorielle:

$$\mathbf{\Phi} = \sum \phi_s(x_n)$$

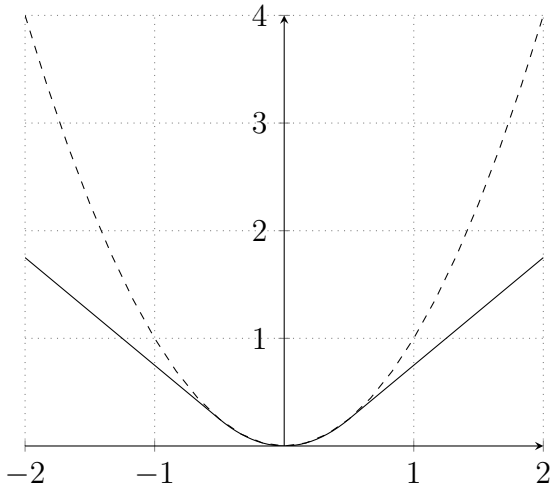

FIGURE 2.2 – Fonction de Huber (s = 0.5) et quadratique

Comme précédemment on utilise différente fonction de régularisation.

- Rappel à un objet connu
$$\mathcal{R}(x) = \Phi_s(\mathbf{x} - \mathbf{x}_\infty)$$
- Terme séparable
$$\mathcal{R}(x) = \Phi_s(\mathbf{x})$$
- Terme de différences (mesure de régularité)
$$\mathcal{R}(x) = \sum_i \phi_s(x_{i+1} - x_i) = \Phi_s(\mathbf{Dx})$$

2.4 Application de ces approches au problème de débruitage

On souhaite résoudre le problème suivant :

$$\mathbf{y} = \mathbf{x} + \mathbf{b}$$

Avec $\mathbf{x}$, blanc de variance σ_x et $\mathbf{b}$ centré de variance σ_b.

2.4.1 Wiener

On a le filtre de wiener :
$$G(\nu) = \frac{\Gamma_{xy}(\nu)}{\Gamma_{yy}(\nu)} = \frac{1}{1 + \frac{\sigma_b^2}{\sigma_x^2}}$$

Dans le cas ou x a une matrice de covariance $\mathbf{D^T D})^{-1}$ on a :

$$G(\nu) = \frac{\Gamma_{xy}(\nu)}{\Gamma_{yy}(\nu)} = \frac{1}{1 + \frac{\sigma_b^2}{\sigma_x^2}|D(\nu)|}$$

Alors :
$$\boxed{\hat{\mathbf{x}}_W = TF^{-1}[G(\nu)Y(\nu)]}$$

2.4.2 MC et MCR

Moindre carrés Sans régularisation on a $H = I_n$ donc directement

$$\boxed{\hat{\mathbf{x}}_{MC} = \mathbf{y}}$$

Moindres carrés régularisé En prenant en compte une régularisation on a :

$$\boxed{\mathbf{x}_{MCR} = (\mathbf{I_d} + \mu \mathbf{D^t D})^{-1}\mathbf{y}}$$

En utilisant les propriétés sur les matrices circulantes on retrouve dans le donmaine de fourier le filtre de Wiener avec $\mu = \frac{\sigma_b^2}{\sigma_x^2}$

$$\boxed{\hat{\mathbf{x}}_{MCR} = TF^{-1}\left[\frac{Y(\nu)}{1 + \mu D(\nu)}\right]}$$

2.4.3 Ondelette et parcimonie

cf poly.

3 Caractérisation statistique des estimateurs

3.1 Espérance

Définition

soit x une VA de densité de probabilité f. On note $E[x]$ *l'espérance* de x:

$$E[x] = \int_{x \in \mathcal{X}} x f(x) \mathrm{d}x$$

3.2 Biais

Le biais caractérise les estimateurs

Définition

Soit $\hat{x}$ un estimateur de x. Alors le *biais* s'écrit :

$$b(\hat{x}) = E[\hat{x}] - x$$

3.3 Variance

La variance est un autre outils pour caractériser les estimateurs. elle donne un intervalle de confiance autour de la valeur de l'estimateur.

Définition

On défini la *variance* d'un estimateur comme:

$$Var(\hat{x}) = E[(\hat{x} - E[\hat{x}])^2]$$

3.4 Erreur quadratique moyenne

L'EQM est un très bon outils pour comparer les estimateurs

Définition

L'erreur quadratique est défini comme:

$$EQM(\hat{x}) = E[(\hat{x} - x)^2]$$

Proposition

On a la relation suivante entre biais, variance et EQM :

$$\begin{aligned} EQM(\hat{x}) &= E[(\hat{x} - x)^2] \\ &= Var(\hat{x}) + b(\hat{x})^2 \end{aligned}$$

Démonstration :

$$\begin{aligned} EQM(\hat{x}) &= E[(\hat{x} - x)^2] \\ &= E[(\hat{x} - E[\hat{x}] + E[\hat{x} - x])] \\ &= E[(\hat{x} - E[\hat{x}])^2] + 2E[\hat{x} - E[\hat{x}]] + (E[\hat{x}] - x)^2 \\ &= Var(\hat{x}) + b(\hat{x})^2 \end{aligned}$$

■

4 Interprétation bayésienne

4.1 Vraisemblance

Définition

En choisissant une ddp gaussienne pour le bruit on a:

$$f(\mathbf{y}|\mathbf{x}) = k_0 \exp\left[\frac{1}{2\sigma_b^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2\right]$$

Comme en pratique on connais $\mathbf{y}$ on a une fonction de $\mathbf{x}$ et σ_b^2. Que l'on appelle fonction de *vraisemblance*.

4.2 Loi *a priori* et a *posteriori*

Définition

- *Loi a priori*

$$f(\mathbf{x}|\sigma_0^2,\sigma_1^2) = k_1 exp\left[\frac{1}{2\sigma_1^2}\|\mathbf{Dx}\|^2 - \frac{1}{2\sigma_0^2}\|x\|^2\right]$$

  La matrice D^tD (covariance de $\mathbf{x}$) n'est pas de rang plein il faut ajouter le deuxième terme pour que la gaussienne soit bien multivariée sur $\mathbb{R}^N$.

- *Loi a posteriori* À partir de la règle de Bayes:

$$f(\mathbf{x}|\mathbf{y},\sigma_b,\sigma_1,\sigma_0) = \frac{f(\mathbf{y}|\mathbf{x},\sigma_b^2)f(\mathbf{x}|\sigma_0,\sigma_1)}{f(\mathbf{y}|\sigma_b^2,\sigma_0^2,\sigma_1^2)} = Kf(\mathbf{y}|\mathbf{x},\sigma_b^2)f(\mathbf{x}|\sigma_0,\sigma_1)$$

  La loi a posteriori rassemble toute l'information que l'on a sur $\mathbf{x}$

4.3 Estimateur du maximum a posteriori

Dans le cas gaussien la moyenne, la médiane et le maximum sont confondus.

Définition

On défini le maximum a posteriori comme:

$$\hat{\mathbf{x}}_{MAP} = \arg\max_x f(\mathbf{x}|\mathbf{y},\sigma_b,\sigma_1,\sigma_0)$$

Proposition *(Cas Gaussien)*

On a :

$$\hat{\mathbf{x}}_{MAP} = \arg\max_x K \exp\left[-\frac{1}{2\sigma_b^2}(\|\mathbf{y}-\mathbf{Hx}\|^2 + \mu\|\mathbf{Dx}\|^2 + \mu_0\|\mathbf{x}^2\|)\right]$$

Soit encore :

$$\hat{\mathbf{x}}_{MAP} = \arg\min_x \|\mathbf{y}-\mathbf{Hx}\|^2 + \mu\|\mathbf{Dx}\|^2 + \mu_0\|\mathbf{x}^2\|$$

5 Application à un cas simple d'observation multiple

6 Application à la déconvolution problème d'optimisation

7 Application de ma méthodologie bayésienne