Projet MAM : Classification par ondelettes

Etienne LAPORTE

20 Fevrier 2013

Sommaire

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Le but de ce travail est d'exposer la réalisation d'une méthode de classification d'images par ondelettes de Haar. En effet dans le cardre du projet de fin d'année du département Mathématiques appliquées et Modélisation, l'élève Etienne LAPORTE a été amené à réaliser cette étude.

Chapitre 2

Motivations

2.1 Vision par ordinateur

La vision par ordinateur (aussi appelée vision artificielle, vision numérique ou plus récemment (vision cognitive) est une branche de l'intelligence artificielle dont le but est de permettre à une machine de comprendre ce qu'elle «voit» lorsqu'on la connecte à une ou plusieurs caméras. L'un des thèmes développés dans ce domaine est celui de l'imitation de la vision humaine par le moyen de composants électroniques. Cette manière de procéder peut être perçue comme un traitement des données visuelles par le biais de modèles basés sur la géométrie, la physique, les statistiques et la théorie d'apprentissage. La vision par ordinateur a aussi été décrite comme une initiative dans l'automatisation et l'intégration d'une vaste gamme de processus et de modèles sur la perception visuelle.

2.2 Application

Les applications vont de la vision industrielle, présente notamment dans l'industrie de fabrication de bouteilles, à la recherche dans le domaine de l'intelligence artificielle et des ordinateurs ou robots capables de comprendre le monde qui les entoure. La vision par ordinateur et la vision industrielle sont des domaines qui se croisent assez souvent. La vision par ordinateur recouvre la technologie centrale de l'analyse d'image automatique, qui est utilisée dans de nombreux domaines. La vision industrielle fait en général référence à la combinaison entre l'analyse d'image automatique et les technologies, afin de pouvoir inspecter de façon automatique et d'orienter les robots à fin d'applications industrielles.

En tant que discipline scientifique, la vision par ordinateur traite de la théorie qui se trouve derrière les systèmes qui traitent des images. Les données extraites se trouvent sous différentes formes : séquences vidéo, vues depuis différentes caméras, ou des données multidimensionnelles lorsqu'elles proviennent d'un scanner médical.

En tant que discipline technologique, la vision par ordinateur cherche à appliquer ses théories et ses modèles à différents systèmes. Quelques exemples de systèmes d'application de la vision par ordinateur :

- Procédés de contrôle, p. ex. dans la robotique industrielle.
- Navigation, p. ex. dans un véhicule autonome ou un robot mobile.
- Détection d'événements, p. ex. pour la surveillance ou le comptage automatique de personnes.
- Organisation d'informations, p. ex. pour indexer des bases de données d'images et de suites d'images.

2.3 But de l'article

L'information contenu dans les textures est résumé par des paramètres de densités de gaussienne généralisée (α, β). Ces gaussiennes généralisées modélisent la répartition des coefficients de décomposition en ondelettes.

Le but de l'article est de faire une comparaison entre une nouvelle méthode de Kullback-Leiber distance (KLD) et des anciennes méthodes (L1, L2 et L1+L2).

Ces distances sont utilisés pour faire la comparaison entre les coefficients qui ont été inférés pour la gaussienne généralisée entre les différentes images.Les distances servent à mesurer la simiralité ou dissimiralité entre les paramètres α et β pour les différentes textures.

Chapitre 3

Théorie

3.1 Résumé de la théorie des ondelettes

Une ondelette est une fonction à la base de la décomposition en ondelettes, décompostion similaire à la transformée de Fourier à court terme utilisée dans le traitement du signal.Elle correspond à l'idée intuitive d'une fonction correspondant à une petite oscillation d'où son nom. Cependant, elle comporte deux différences majeures avec la transformée de Fourier à court terme :

- Elle peut mettre en oeuvre une base différente, non forcément sinusoïdale.
- Il existe une relation entre la largeur de l'envellope et la fréquence des oscillations : on effectue ainsi une homothétie de l'ondelette et non seulement de l'oscillation.

L'ondelette de Haar crée par Alfréd Haar en 1909 est la première ondelette connue. C'est une fonction dilatée et/ou translatée de la fonction mère Φ qui vaut :

$$\Psi(t) = \begin{cases} 1 & \text{pour } 0 \leq t < 0.5 \\ -1 & \text{pour } 0.5 \leq t < 1 \\ 0 & \text{sinon} \end{cases}$$

(3.1)

Soit graphiquement :

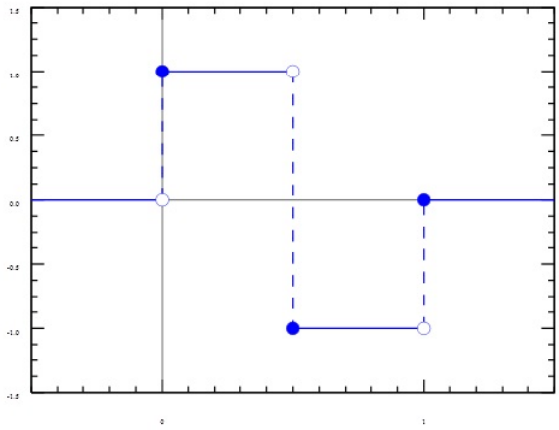

FIGURE 3.1 – Ondelette de Haar

La théorie des ondelettes est intéressante en analyse d'image car c'est une analyse multi résolution qui capture les détails de peu à très locaux et plus diffus dans l'image à plusieurs échelles selon les étapes du filtre (différente sous-bandes)

Pour l'estimation par ondelette il y a deux étapes :
– une étape approximation .
– une étape détails.
Puis ensuite sur l'étape approximation, on refait de manière itérative, une étape où l'on a encore une approximation et des détails.

Cette technique repose sur la décomposition sur la base d'ondelette d'une part sur une ondelette mère pour les détails et sur une ondelette père pour les approximation.

3.1.1 1D

Le principe du processus itératif (1D) est le suivant : On commence avec le vecteur original, de taille N, qu'on veut transformer. La 1ere étape consiste à faire le produit matricielle par la matrice W_N (explicité plus bas). On obtient un vecteur dont les N/2 premières composantes est le "vecteur d'approximation" $y_{1,a}$ et les $N/2$ suivantes le "vecteur des détails" du niveau 1, $y_{1,d}$. A l'étape suivante on fait le produit de $W_{N/2}$ avec le vecteur d'approximation

obtenu à l'étape précédente, $y_{1,a}$. On obtient à nouveau un vecteur d'approximation et de détails. On continue ainsi de suite en appliquant à chaque fois la transformation sur le vecteur d'approximation obtenu à l'étape précédente.

3.1.2 2D

Le principe du processus itératif en 2D est très similaire. Au lieu de faire la transformation sur un vecteur on le fait sur une matrice.

$$Res = W_N A W_N^T = \begin{bmatrix} A & V \\ H & D \end{bmatrix} = \begin{bmatrix} Approximation\ moyenne & Detail\ Verticaux \\ Detail\ Horizontaux & Detail\ Diagonaux \end{bmatrix} \tag{3.2}$$

A chaque étape la transformation consiste à faire deux multiplications matricielles, une à gauche et une à droite avec W_N et W_N^T(la transposée de W_N).

$$W_N = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & \cdots & 0 \\ \vdots & & & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \tag{3.3}$$

Ici à chaque étape on obtient 4 matrices, chacune de taille $N/2$ x $N/2$. On a une matrice d'approximation et 3 matrices de détails. Avec N, la taille de la matrice d'approximation de l'étape précédente. Dans l'article de classification de textures on fait 3 itérations, ce qui donne 9 matrices de détails.

Ces matrices contiennent les coefficients de la décomposition en ondelette. Pour chacune d'elles on considère la distribution que suit ses coeffeciants, c'est là qu'intervient l'ajustement d'une gaussienne généralisée paramétrée par le couple (α, β). Ce qui fait un couple pour chacune des 9 matrices de détail.

3.2 GGD

3.2.1 Densité de loi

GGD désigne la loi gaussienne généralisée (Generalized Gaussian distribution) dont les support sont l'ensemble des réels. Cette loi rajoute un paramètre de forme à la loi normale.

La densité de probabilité est donnée par la formule suivante :

$$p(x; \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})} e^{-}(\frac{|x-\mu|}{\alpha})^{\beta} \tag{3.4}$$

où $\mu = 0$ dans notre étude.

3.2.2 Maximum de Vraissemblance

Pour estimer les caractéristiques α et β de ces lois, nous allons utiliser le maximum de vraissemblance. Soit :

$$\begin{aligned}
L(\underline{x};\alpha,\beta) &= Ln(\prod_{i=1}^{L} p(x_i;\alpha,\beta)) \\
&= Ln(\prod_{i=1}^{L} \frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})} e^{-}(\frac{|x|}{\alpha})^{\beta}) \\
&= Ln(\prod_{i=1}^{L} \frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}) + Ln(e^{-}(\frac{|x_i|}{\alpha})^{\beta})) \\
&= \sum_{i=1}^{L} Ln(\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}) + \sum_{i=1}^{L}(-(\frac{|x_i|}{\alpha})^{\beta}) \\
&= Ln(\prod_{i=1}^{L} \frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}) + Ln(e^{-}(\frac{|x_i|}{\alpha})^{\beta})) \\
&= \sum_{i=1}^{L} Ln(\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}) + \sum_{i=1}^{L}(-(\frac{|x_i|}{\alpha})^{\beta}) \\
&= L.Ln(\frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})}) - \sum_{i=1}^{L} \frac{|x_i|^{\beta}}{\alpha^{\beta}}) \\
&= L.Ln(\beta) - L.Ln(2\alpha\Gamma(\frac{1}{\beta})) - L\alpha^{-\beta} \sum_{i=1}^{L} |x_i|^{\beta}
\end{aligned}$$

Ainsi on obtient :

$$\boxed{L(\underline{x};\alpha,\beta) = L.Ln(\beta) - L.Ln(2\alpha\Gamma(\frac{1}{\beta})) - L\alpha^{-\beta} \sum_{i=1}^{L} |x_i|^{\beta}} \qquad (3.5)$$

Dérivée par rapport a aplha :

$$\begin{aligned}\frac{\partial L(\underline{x};\alpha,\beta)}{\partial \alpha} &= -L\frac{2\Gamma(\frac{1}{\beta})}{2\alpha\Gamma(\frac{1}{\beta})} - L((-\beta)\alpha^{-\beta-1}).\sum_{i=1}^{L}|x_i|^{\beta} \\ &= -\frac{L}{\alpha} + \frac{L}{\alpha^{1+\beta}}\sum_{i=1}^{L}|x_i|^{\beta} \\ &= -\frac{L}{\alpha} + \sum_{i=1}^{L}\frac{\beta|x_i|^{\beta}\alpha^{-\beta}}{\alpha}\end{aligned}$$

Ainsi :

$$\boxed{\frac{\partial L(\underline{x};\alpha,\beta)}{\partial \alpha} = -\frac{L}{\alpha} + \sum_{i=1}^{L}\frac{\beta|x_i|^{\beta}\alpha^{-\beta}}{\alpha}} \tag{3.6}$$

Dérivée par rapport a beta :

$$\begin{aligned}
\frac{\partial L(\underline{x};\alpha,\beta)}{\partial\beta} &= \frac{L}{\beta} - L\frac{\partial}{\partial\beta}(Ln(2\alpha\Gamma(\frac{1}{\beta}))) - \frac{\partial}{\partial\beta}(\sum_{i=1}^{L}(\frac{|x_i|}{\alpha})^{\beta}) \\
&= \frac{L}{\beta} - L\frac{\partial}{\partial\beta}(Ln(2\alpha\Gamma(\frac{1}{\beta}))) - \frac{\partial}{\partial\beta}(\sum_{i=1}^{L}e^{\beta ln(\frac{|x_i|}{\alpha})}) \\
&= \frac{L}{\beta} - L\frac{\partial}{\partial\beta}[Ln(2\alpha) + Ln(\Gamma(\frac{1}{\beta}))] - \sum_{i=1}^{L}ln(\frac{|x_i|}{\beta})e^{\beta ln(\frac{|x_i|}{\alpha})} \\
&= \frac{L}{\beta} - L\frac{\partial}{\partial\beta}[Ln(2\alpha) + Ln(\Gamma(\frac{1}{\beta}))] - \sum_{i=1}^{L}ln(\frac{|x_i|}{\beta})e^{\beta Ln(\frac{|x_i|}{\alpha})} \\
&= \frac{L}{\beta} - L\frac{\partial}{\partial\beta}[Ln(\Gamma(\frac{1}{\beta}))] - \sum_{i=1}^{L}Ln(\frac{|x_i|}{\alpha}).(\frac{|x_i|}{\alpha})^{\beta} \\
&= \frac{L}{\beta} - L\frac{\Gamma'(\frac{1}{\beta})}{\Gamma(\frac{1}{\beta})} - \sum_{i=1}^{L}Ln(\frac{|x_i|}{\alpha}).(\frac{|x_i|}{\alpha})^{\beta} \\
&= \frac{L}{\beta} - L\frac{\Gamma'(\frac{1}{\beta})}{\Gamma(\frac{1}{\beta})} - \sum_{i=1}^{L}Ln(\frac{|x_i|}{\alpha}).(\frac{|x_i|}{\alpha})^{\beta} \\
&\text{soit } \Psi(\frac{1}{\beta}) = \frac{\Gamma'(\frac{1}{\beta})}{\Gamma(\frac{1}{\beta})} \\
&= \frac{L}{\beta} - L(-\frac{1}{\beta^2}).\Psi(\frac{1}{\beta}) - \sum_{i=1}^{L}Ln(\frac{|x_i|}{\alpha}).(\frac{|x_i|}{\alpha})^{\beta} \\
&= \frac{L}{\beta} + \frac{L\Psi(\frac{1}{\beta})}{\beta^2} - \sum_{i=1}^{L}Ln(\frac{|x_i|}{\alpha}).(\frac{|x_i|}{\alpha})^{\beta}
\end{aligned}$$

Ainsi :

$$\boxed{\frac{\partial L(\underline{x};\alpha,\beta)}{\partial\beta} = \frac{L}{\beta} + \frac{L\Psi(\frac{1}{\beta})}{\beta^2} - \sum_{i=1}^{L}(\frac{|x_i|}{\alpha})^{\beta}Ln(\frac{|x_i|}{\alpha})} \tag{3.7}$$

On fixe $\beta > 0$ et ainsi (3.6) l'unique solution réelle positive est déterminé par :

$$-\frac{L}{\alpha}+\sum_{i=1}^{L}\frac{\beta|x_i|^{\beta}\alpha^{-\beta}}{\alpha}>0$$

$$\frac{L}{\alpha}<\sum_{i=1}^{L}\frac{\beta|x_i|^{\beta}\alpha^{-\beta}}{\alpha}$$

$$\frac{L}{\alpha}<\frac{\beta\alpha^{-\beta}}{\alpha}\sum_{i=1}^{L}|x_i|$$

$$1<\frac{\beta}{L\alpha^{\beta}}\sum_{i=1}^{L}|x_i|$$

$$\alpha^{\beta}<\frac{\beta}{L}\sum_{i=1}^{L}|x_i|$$

Ainsi l'estimateur de est donné par :

$$\boxed{\widehat{\alpha}=(\frac{\beta}{L}\sum_{i=1}^{L}|x_i|)^{\frac{1}{\beta}}} \tag{3.8}$$

Puis en injectant cette solution de dans l'équation (3.7), on obtient l'équation suivante à résoudre :

$$\boxed{1+\frac{\Psi(\frac{1}{\widehat{\beta}})}{\widehat{\beta}}-\frac{\sum_{i=1}^{L}|x_i|^{\widehat{\beta}}ln|x_i|}{\sum_{i=1}^{L}|x_i|^{\widehat{\beta}}}+\frac{ln(\frac{\widehat{\beta}}{L}\sum_{i=1}^{L}|x_i|^{\widehat{\beta}})}{\widehat{\beta}}=0} \tag{3.9}$$

3.2.3 Newton-Raphson

Cette équation (3.9) peut être résolu par une méthode de Newton-Raphson qui sera implémenté ici :

$$\beta_{k+1}=\beta_k-\frac{g(\beta_k)}{g'(\beta_k)} \tag{3.10}$$

où l'on a :

$$g(\beta)=1+\frac{\Psi(\frac{1}{\beta})}{\beta}-\frac{\sum_{i=1}^{L}|x_i|^{\beta}ln|x_i|}{\sum_{i=1}^{L}|x_i|^{\beta}}+\frac{ln(\frac{\beta}{L}\sum_{i=1}^{L}|x_i|^{\beta})}{\beta} \tag{3.11}$$

et :

$$g'(\beta) = -\frac{\Psi(\frac{1}{\beta})}{\beta^2} - \frac{\Psi'(\frac{1}{\beta})}{\beta^3} + \frac{1}{\beta^2} - \frac{\sum_{i=1}^{L} |x_i|^\beta ln(|x_i|)^2}{\sum_{i=1}^{L} |x_i|^\beta} + \frac{(\sum_{i=1}^{L} |x_i|^\beta ln|x_i|)^2}{(\sum_{i=1}^{L} |x_i|^\beta)^2} + \frac{\sum_{i=1}^{L} |x_i|^\beta ln|x_i|}{\beta \sum_{i=1}^{L} |x_i|^\beta} - \frac{ln(\frac{\beta}{L} \sum_{i=1}^{L} |x_i|^\beta)}{\beta^2}$$

Le β_0 est trouvé pas interpolation et une recherche au sein d'une table tabulée de F ci dessous :

$$F_M(\beta) = \frac{\Gamma(\frac{2}{\beta})}{\sqrt{\Gamma(\frac{1}{\beta})\Gamma(\frac{3}{\beta})}} \tag{3.12}$$

où la valeur cherchée est :

$$\beta_0 = F_M^{-1}(\frac{m_1}{\sqrt{m_2}}) \tag{3.13}$$

avec $m_1 = \frac{1}{L} \sum_{i=1}^{L} |x_i|$ et $m_2 = \frac{1}{L} \sum_{i=1}^{L} x_i^2$

3.3 KLD

La distance de Kullback-Leibler mesure la dissimilarité entre deux distributions de probabilités P et Q, voir l'image ci-dessous :

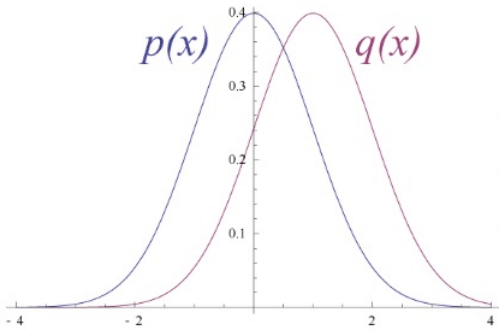

FIGURE 3.2 – KLD : Dissimilarité entre deux distributions

Cette distance se calcule par la formule suivante :

$$\boxed{D(p(.;\alpha_1,\beta_1)||p(.;\alpha_2,\beta_2)) = ln(\frac{\beta_1\alpha_2\Gamma(\frac{1}{\beta_2})}{\beta_2\alpha_1\Gamma(\frac{1}{\beta_1})} + (\frac{\alpha_1}{\alpha_2})^{\beta_2}\frac{\Gamma(\frac{\beta_2+1}{\beta_1})}{\Gamma(\frac{1}{\beta_1})} - \frac{1}{\beta_1}} \quad (3.14)$$

Dont la formule est implémenté par le fichier 7.9

Chapitre 4

Application à la classification de texture

On applique cette étude à la classification de texture, comme ci-dessous :

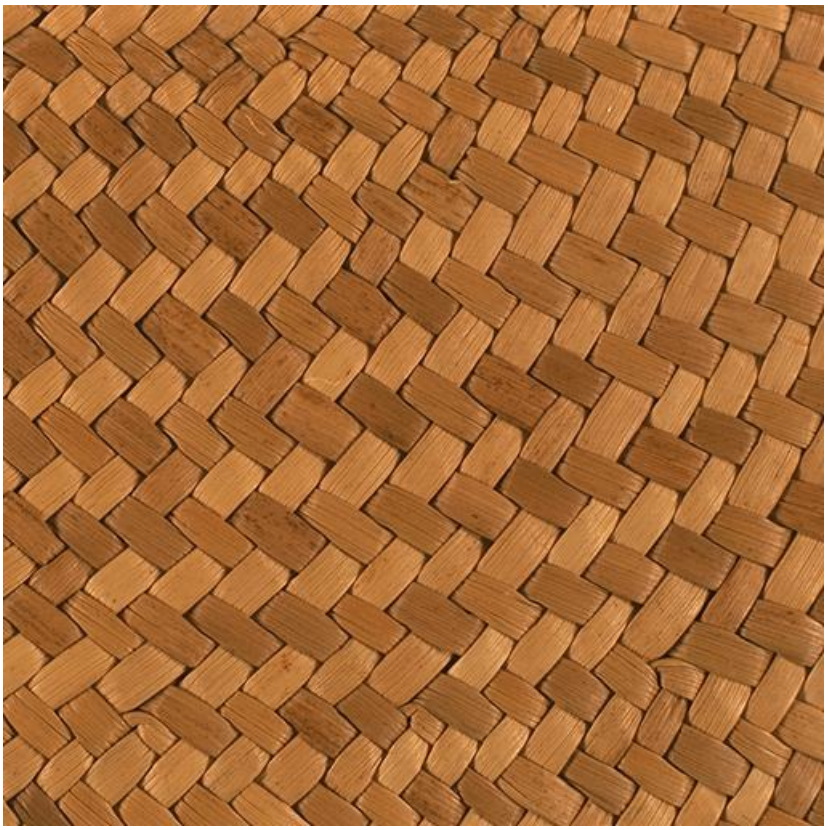

FIGURE 4.1 – Image de Texture

Chaque images est de taille 512*512 sera divisé en 16 sous images de taille 128*128 et chaque sous images aura 9 sous bandes de calculer par Haar avec 2 coefficients par ondelette. Le but de ce code est d'automatiser le traitement des images pour dans un premier en temps en faire une Base, le fichier de lecture en entrée sera la suivant :

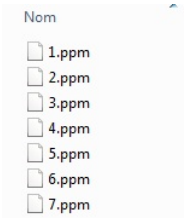

FIGURE 4.2 – Dossier en entrée du programme des images

Lu par le fichier BD.m et ont la sortie sera la Base. Lors de l'appel de cette fichier BD.m nous avons appel à plusieurs sous fonction :

- Separation.m (cf. section 7.1) qui sépare la grande image en 16 sous images
- obtenirMatsousbande0.m (cf. section 7.2)qui extrait les 9 sous bandes d'une sous-image sous forme de vecteur
- tableF.m (cf. section 7.7) renvoit la valeur d'un beta0 pour initialiser la fonction de Newton-Raphson
- newton.m (cf. section 7.4)qui permet de renvoyer la valeur du beta calculé par l'algorithme

Remarque : Lors de l'implémentation du code obtenirMatsousbande0.m certaines valeurs des vecteurs sous bande étaient nulles ce qui entrainer un problème dans la méthode de Newton-Raphson, en effet lors de l'appel du logarithme la valeur 0 renvoyer l'infini. Pour compenser cela, j'ai décalé ces valeurs nulles par une valeur de déviation par rapport à l'écart de type des valeurs du vecteur.

Pour illustrer ce raisonnement, voici la répartition d'un vecteur sous bande d'une sous images :

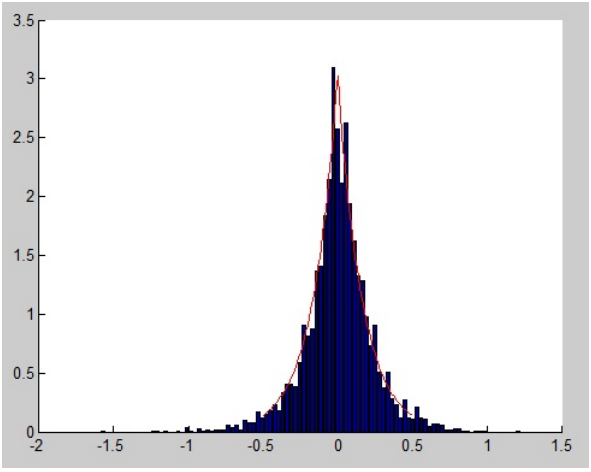

FIGURE 4.3 – Répartition et Approximation par GGD d'une sous-bande $\alpha = 0.1667$, $\beta = 1.0227$

A la fin de l'exécution du fichier BD.m la Base.m est ainsi crée c'est une base à 4 dimensions, avec pour dimensions :

- Les images
- Les sous-images
- Les sous-bandes
- Les coefficients α et β

Une fois que la base est créée, il ne reste plus qu'à calculer la distance (KLD, L1, L1+L2) séparant une sous-image de référence (toutes les sous-images vont être prises en tant que référence) à toutes les sous-images de chaque image. Puis de trier ces distances par ordres décroissants. La logique voudrait que les sous-images de la même image de référence soit en tête de liste, chose qui est vraie mais pas à 100% et ce pourcentage que nous allons regarder et comparer avec l'article.

Chapitre 5

Résultats

Voici les résultats que j'ai obtenu :

Distance	Article	Moi	Ecart relatif
L1	62.72	58.93	6.4%
L2	61.76	54.14	14.1%
L1+L2	57.20	64.48	12.7%
KLD	76.01	76.93	1.2%

TABLE 5.1 – Poucentages obtenus

Voici les temps de calcul en secondes :

Code	AMD C-50 Processor 1.00 GHz	Pentium(R) Dual-Core CPU 2.80 Ghz
BD.m	281	60
main_KLD.m	634	116
main_L1.m	36	7
main_L2.m	36	7
main_L1_L2.m	46	8

TABLE 5.2 – Temps d'éxécution

Base	KLD	L1	L2	L2
2	74.58	67.03	64.37	62.62
4	66.87	56.77	56.67	56.98
8	63.96	48.02	46.71	47.97
16	49.27	41.54	38.70	40.70
32	53.57	46.57	44.58	46.12
32	53.57	46.57	44.58	46.12
64	56.80	42.39	38.80	41.16
106	54.94	38.00	34.52	36.83
article	76.02	58.92	54.13	57.20
rand	64.28	49.05	45.95	48.06

TABLE 5.3 – Résultats du pourcentage pour KLD, L1, L2, L1+L2 suivant l'ordre d'apparition des images dans la base

Nombre de répétitions	Nombre d'images prises	KLD Moyenne	IC à 95%
100	2	95.74	[94.16 ;97.32]
100	4	97.38	[85.61 ;89.15]
100	8	83.10	[81.67 ;84.53]
100	16	74.32	[73.15 ;75.50]
100	32	63.71	[62.98 ;64.45]
10	106	41.96	X

TABLE 5.4 – Résultats du pourcentage pour la distance KLD en randomisant les images de la base (doublons possible)

Nombre de répétitions	Nombre d'images prises	KLD Moyenne	IC à 95%
100	2	95.82	[94.37 ;97.27]
100	4	91.51	[90.17 ;92.85]
100	8	84.84	[83.65 ;86.02]
100	16	78.39	[77.54 ;79.36]
100	40	67.88	[67.36 ;68.39]

TABLE 5.5 – Résultats du pourcentage pour la distance KLD en randomisant les images de la base (sans doublons)

Base	KLD	L1	L2	L1+L2
14	71.43	58.98	55.23	57.26

TABLE 5.6 – Résultats du pourcentage pour KLD, L1, L2, L1+L2 pour une base de 14 images provenant chacune de textures différentes

Texture	Base	KLD	L1	L2	L1+L2
Brick	4	72.08	60.26	60.31	61.15
Barck	11	59.17	49.20	47.72	48.30
Fabric	20	71.79	61.77	59.41	61.14
Flowers	4	80.83	73..54	70.31	71.86
Food	10	68.12	53.75	47.54	51.75
Grass	2	60.62	55.62	56.57	56.04
Leaves	12	66.42	48.47	45.21	47.40
Metal	6	70.97	62.12	59.51	61.46
Misc	4	71.67	59.79	54.79	57.39
Paitings	3	71.80	55.28	54.30	54.44
Sand	7	74.82	52.32	48.69	50.77
Water	2	62.50	56.67	56.46	56.25
Wood	3	70.97	65.97	62.92	63.61
Moyenne		69.37	58.06	55.68	57.04
Ecart type		5.98	6.94	7.12	6.84

TABLE 5.7 – Résultats du pourcentage pour KLD, L1, L2, L1+L2 pour des bases d'images d'une même texture

Chapitre 6

Conclusion

Le chiffre indique le pourcentage de nombre de bonnes images dans la liste des 15 premières images. Après il faut savoir que la visualisation par ordinateur est un problème difficile et que cette technique de mesure par distance de KLD date de 2002. Il faudrait dans une étude plus profonde s'intéresser sur les images qui sont parmi les 15 première et qui ne sont pas les bonne et voir si elles sont franchement différente ou plutôt similaire. Après ce qu'il faut voir c'est que ce sont les 15 premières images parmi une base de 640 (pour 40 images avec 16 petites et jusqu'à 1600 pour 100 images). On pourrait s'intéresser dans une seconde étude à regarder uniquement les cinq premières et donc peut être obtenir 100%. Après il faut bien être claire sur l'interprétation, ce pourcentage ne représente pas le pourcentage d'erreur à proprement parlé.

Ce n'est pas une règle de décision que nous avons ici, le pourcentage représente en effet sur les 15 premier résultats, quelle est le pourcentage de bons résultats. Pour avoir 100%, il faudrait donc avoir toutes les images qui se rapportent à la même texture soit les 15 premières ce qui est donc une contrainte qui est relativement assez forte.

Si maintenant on voulait faire un algorithme qui décide de mettre un nom sur une texture, ce qu'on pourrait faire c'est considéré parmi les 15 premières textures sachant l'image départ, quelle est celle qui a le plus de sous-images parmi les résultats.

La distance de Kullback Leiber donne éffectivement des résultats plus précis moyennant +10 à 20% sur les autres méthodes (L1,L2,L1+L2) cependant le temps de calcul est plus important (16 fois plus long pour une base de 40 images).

Chapitre 7

Codes

7.1 Separation.m

```
function [Images]=Separation(Im)

global Images

Images{1}=Im(1:128,1:128);
Images{2}=Im(129:256,1:128);
Images{3}=Im(257:384,1:128);
Images{4}=Im(385:512,1:128);

Images{5}=Im(1:128,129:256);
Images{6}=Im(129:256,129:256);
Images{7}=Im(257:384,129:256);
Images{8}=Im(385:512,129:256);

Images{9}=Im(1:128,257:384);
Images{10}=Im(129:256,257:384);
Images{11}=Im(257:384,257:384);
Images{12}=Im(385:512,257:384);

Images{13}=Im(1:128,385:512);
Images{14}=Im(129:256,385:512);
Images{15}=Im(257:384,385:512);
Images{16}=Im(385:512,385:512);

end
```

7.2 obtenirMatsousbande0.m

```
function []=obtenirMatsousbande0(Av)

global  v_D1 v_D2 v_D3
global  v_H1 v_H2 v_H3
global  v_V1 v_V2 v_V3

vec=cell(9,1);
S_AV=cell(1,3);
S_D=cell(1,3);
S_V=cell(1,3);
S_H=cell(1,3);


    %% centre les valeurs
    moy=mean(Av(:));
    ecart_t=std(Av(:));
    Av=(Av-moy)./ecart_t;

    %% Calculs
    t=length(Av);
    i=1;
    while i<=3
        WN=Mat_W(t);
        res=WN*Av*WN';

        %Calculs
        Av=res(1:t/2,1:t/2);
        V=res(1:t/2,(t/2)+1:t);
        H=res((t/2)+1:t,1:t/2);
        D=res((t/2)+1:t,(t/2)+1:t);

        if (res~=[Av,V;H,D])
            disp('Probme dans ObtenirMatsousBande0');
        end

        %Sauvegarde
        S_AV{i}=Av; %S_AV{} pour r cupr la valeur
            contenue dans la cellule.
```

```
        S_D{i}=D;
        S_V{i}=V;
        S_H{i}=H;

        % I t ration  suivante
        t=length(Av);
        i=i+1;
end


%% Matrices soubandes
D1=S_D{1};
V1=S_V{1};
H1=S_H{1};

D2=S_D{2};
V2=S_V{2};
H2=S_H{2};

D3=S_D{3};
V3=S_V{3};
H3=S_H{3};

%% Distribution des 9 Matrices sous-bandes
v_D1=D1(:);
v_V1=V1(:);
v_H1=H1(:);

v_D2=D2(:);
v_V2=V2(:);
v_H2=H2(:);

v_D3=D3(:);
v_V3=V3(:);
v_H3=H3(:);

v_D1(v_D1==0)=0.5./ecart_t;
v_V1(v_V1==0)=0.5./ecart_t;
v_H1(v_H1==0)=0.5./ecart_t;

v_D2(v_D2==0)=0.5./ecart_t;
```

```
    v_V2(v_V2==0)=0.5./ecart_t;
    v_H2(v_H2==0)=0.5./ecart_t;

    v_D3(v_D3==0)=0.5./ecart_t;
    v_V3(v_V3==0)=0.5./ecart_t;
    v_H3(v_H3==0)=0.5./ecart_t;

    vec={v_D1;v_V1;v_H1;v_D2;v_V2;v_H2;v_D3;v_V3;v_H3};

end
```

7.3 Mat_W.m

```
function [WN]=Mat_W(t)
r2=sqrt(2);
nl=t/2;
M_H=zeros(nl,t);
M_G=zeros(nl,t);
dec=3;
for i=1:nl
    if i==1
        M_H(1,1)=r2/2;
        M_H(1,2)=r2/2;
        M_G(1,1)=r2/2;
        M_G(1,2)=r2/-2;
    else
    M_H(i,dec)=r2/2;
    M_H(i,dec+1)=r2/2;
    M_G(i,dec)=r2/2;
    M_G(i,dec+1)=r2/-2;
    dec=dec+2;
    end
end
WN=[M_H;M_G];
end
```

7.4 newton.m

```
function [b,nb,bb]=newton(g,dg,b0,v,epsilon)

%% Variables
% bb liste des it r de beta
% nb nombre d'it rations pour r pondre la condition
    eps
% b valeur de b ta alu chaque iteration
% bt valeur de b ta alu l'iteration p r cnte



%% Algorithmes
err=1;
bt=b0;
b=b0;
bb=[b0];
nb=0;
     while (err > epsilon)
          nb=nb+1;
          b=b-feval(g,b,v)/feval(dg,b,v);
          if b<0
               b=0.01;
          end
          err=norm(b-bt);
          bt=b;
          bb=[bb,b];
     end


end
```

7.5 g.m

```
function y=g(b,v)
L=length(v);

p1=1+(psi(1./b)./b);

p2_1=sum((abs(v).^b).*(log(abs(v))));
p2_2=sum(abs(v).^b);
p2=p2_1./p2_2;

p3_1=log((b./L).*sum(abs(v).^b));
p3_2=b;
p3=p3_1./p3_2;

y=p1-p2+p3;
end
```

7.6 dg.m

```
function y=dg(b,v)
L=length(v);


p1=psi(1./b)./(b.^2);
p2=psi(1,(1./b))./(b.^3);%derivee de psi
p3=(1./b.^2);

p4_1=sum((abs(v).^b).*(log(abs(v))).^2);
p4_2=sum(abs(v).^b);
p4=p4_1./p4_2;

p5_1=sum((abs(v).^b).*(log(abs(v))));
p5_2=sum(abs(v).^b);
p5=(p5_1./p5_2).^2;

p6_1=sum((abs(v).^b).*(log(abs(v))));
p6_2=b.*sum(abs(v).^b);
p6=p6_1./p6_2;

p7_1=log((b./L).*sum(abs(v).^b));
p7_2=b.^2;
p7=p7_1./p7_2;

y=-p1-p2+p3-p4+p5+p6-p7;


end
```

7.7 tableF.m

```
function [beta0]=tableF(m1,m2)

b_cal=m1./(sqrt(m2));

x=0:0.01:10;
y=F(x);

val_rechercher=b_cal;

% Recherche dans F(x) la position valeur la plus proche
    de b_cal
 [C,indice]=min(abs(y - val_rechercher));

% Recherche de la valeur de cette position dans le
   vecteur des beta
beta0=x(indice);



end
```

7.8 main KLD.m

```
clear all
close all
clc

tic
profile on

%% Importation de la Base de donnes
load('Base.mat');


%% Distance KLB
taille=size(Base,1);
Distance=zeros(1,taille);

j=1;
k=1;
p=zeros(1,taille*16);

for numItest=1:taille
    for imitest=1:16

        for numI=1:taille
            for imi=1:16
                D=0;
                for s=1:9
                    D=D+KLD(Base(numItest,imitest,s,1),
                        Base(numItest,imitest,s,2),Base(
                        numI,imi,s,1),Base(numI,imi,s,2)
                        );
                end
            Distance((numI-1).*16+imi)=D;
            end
        end

        Liste_R=[1:(taille*16);Distance];%%Liste de
            toute les images avec distance
        %Classement=sort(Liste_R,2);
```

```
        Classement=sortrows(Liste_R.',2).';
        Classement=Classement(1,:);
        liste_classement=Classement(2:17);%%Prend les
            15 meilleurs hormis l'image de   r fnce
        p(j)=pourcentage(liste_classement,16.*k);
        j=j+1;


    end
    k=k+1;
end

Proportion=mean(p);
disp('Proportion :');
disp(Proportion);

%sortrows(Liste_R.',2).'

toc
t=toc;

profile viewer
```

7.9 KLD.m

```
function D=KLD(a1,b1,a2,b2)

p1=log((b1.*a2.*gamma(1./b2))./(b2.*a1.*gamma(1./b1)));
p2=((a1./a2).^b2).*((gamma((b2+1)./b1))./(gamma(1./b1))
    );
p3=1./b1;

D=p1+p2-p3;

end
```

7.10 pourcentage.m

```
function p=pourcentage(liste,zone)

b_inf=zone-16;
b_sup=zone;
p=(length(find( b_inf <= liste & liste <= b_sup))./15)
    .*100;


end
```

7.11 L1.m

```
function [r]=L1(v_a1,v_b1,v_a2,v_b2)

p1=sum(abs(v_a1-v_a2));
p2=sum(abs(v_b1-v_b2));
r=p1+p2;

end
```

7.12 L2.m

```
function [r]=L2(v_a1,v_b1,v_a2,v_b2)

p1=sum((v_a1-v_a2).^2);
p2=sum((v_b1-v_b2).^2);
r=p1+p2;


end
```

7.13 L1L2.m

```
function [r]=L1L2(v_a1,v_b1,v_a2,v_b2)

p1=sum(abs(v_a1-v_a2));
p2=sum(abs(v_b1-v_b2));
p3=sum((v_a1-v_a2).^2);
p4=sum((v_b1-v_b2).^2);
r=p1+p2+p3+p4;




end
```

7.14 densite.m

```
function y=densite(a,b,x)
y=b./(2.*a.*gamma(1./b)).*exp(-(abs(x)./a).^b);
end
```

7.15 F.m

```
function y=F(b)
y=(gamma(2./b))./(sqrt(gamma(1./b).*gamma(3./b)));
end
```

7.16 verif.m

```
function []=verif(a1,b1,v)

i=1;
for x=-10:0.01:10;
    y(i)=densite(a1,b1,x);
    i=i+1;
end
xvar=-10:0.01:10;

[z,w]=hist(v,100);
hold on
bar(w,z/trapz(w,z));
plot(xvar,y,'-r');
%title(['Histogramme pour image:',num2str(imi),' sous
   bande :',num2str(s)]);
hold off

end
```

7.17 test main.m

```
clear all
close all
clc

tic
profile on

%% Importation de la Base de donnes
load('Base.mat');


%% Distance KLB
taille=Base{end,2};
Distance=zeros(1,taille);

j=1;
k=1;
p=zeros(1,taille*16);

for numItest=1:taille
    for imitest=1:16

        for numI=1:taille
            for imi=1:16
                D=0;
                for s=1:9
                    D=D+KLD(Base{ibase(s,imitest,
                        numItest),5},Base{ibase(s,
                        imitest,numItest),6},Base{ibase(
                        s,imi,numI),5},Base{ibase(s,imi,
                        numI),6});
                end
            Distance((numI-1).*16+imi)=D;
            end
        end

        Liste_R=[1:(taille*16);Distance];%%Liste de
            toute les images avec distance
```

```
            [~,Classement]=sort(Liste_R,2);
            liste_classement=Classement(2,2:17);%%Prend les
                15 meilleurs hormis l'image de   r fnce
            p(j)=pourcentage(liste_classement,16.*k);
            j=j+1;

        end
        k=k+1;

end


Proportion=mean(p);
disp('Proportion :');
disp(Proportion);

%sortrows(Liste_R.',2).'

toc
t=toc;

profile viewer
```

7.18 plotg.m

```
function []=plotg(v)


i=1;
for beta=0.01:0.01:2
    f(i)=g(beta,v);
    i=i+1;
end

bvar=0.01:0.01:2;
plot(bvar,f);




end
```

7.19 tester 1 I 1 S.m

```
clear all
close all
clc

global Images
global  v_D1 v_D2 v_D3
global  v_H1 v_H2 v_H3
global  v_V1 v_V2 v_V3
global imi
global s
global v


epsilon=1e-7;
v_Long=[4096,4096,4096,1024,1024,1024,256,256,256];
a=cell(16,9);
b=cell(16,9);
beta0=cell(16,9);



%% Chemin image
        chemin='C:\Users\Utilisateur\Desktop\Projet 5A\
            Codes\Base texture\Avec Chiffres\32.ppm';
        image = imread(chemin);

    %% Matrices Luminence
        IM=rgb2gray(image);
        Av=double(IM);

    %% Separation en 16 images
        r=Separation(Av);

    %% Boucle sur les 16 images de l'image
    imi=4;

        Av=Images{imi};
        obtenirMatsousbande0(Av);
```

```
        vec={v_D1;v_H1;v_V1;v_D2;v_H2;v_V2;v_D3;v_H3;
            v_V3};

        %% Newton pour le beta sur chaque sous bande de
            chaque images
    s=5;

                    %Paramtre Newton
                    L=v_Long(s);
                    m1=(1./L).*sum(abs(vec{s}));
                    m2=(1./L).*sum(vec{s}.*vec{s});
                    v=vec{s};

                    figure(1);
                    plotg(v);

                    b0=tableF(m1,m2);
%                       if g(b0,v)<0
%                           b0=tableF(0,m2);
%                       end
%
                    beta0{imi,s}=b0;

                    %% Appel fonction Newton
                    [b{imi,s},nbiter,bb]=newton('g','dg',b0
                        ,v,epsilon);
                    kk=b{imi,s};
                    a{imi,s}=((b{imi,s}./L).*sum(abs(v).^b{
                        imi,s})).^(1./b{imi,s});

                    %% Vrification de l'approximationn
                    a1=a{imi,s};
                    b1=b{imi,s};
```

7.20 main choix.m

```
clear all
close all
clc


nomBase=input('Choix de la base ? ','s'); %% Choix nom
    de la base
rand_oui=input('Random Base ? ','s');
methode_analyse=input('Choix de la m thode analyse ? ',
    's'); %% Choix : 1 2 3 4

disp('La base est en cours...')

switch rand_oui
     case 'oui'
          fonction_BD_rand(nomBase);
     otherwise
         fonction_BD(nomBase);
end

disp('La base a     ablie ');

disp('Le processus de reconnaissance est en cours...');

switch  methode_analyse

     case 'KLD'
          main_KLD;
     case 'L1'
          main_L1;
     case 'L2'
          main_L2;
     case 'L1L2'
          main_L1_L2;
     case 'ALL'
           main_KLD;
           main_L1;
           main_L2;
```

```
            main_L1_L2;
end
```

7.21 fonction BD.m

```
function [Base]=fonction_BD(nom_BD)

tic

global Images
global  v_D1 v_D2 v_D3
global  v_H1 v_H2 v_H3
global  v_V1 v_V2 v_V3
global imi
global s
global v


epsilon=1e-7;
v_Long=[4096,4096,4096,1024,1024,1024,256,256,256];

a=cell(16,9);
b=cell(16,9);
beta0=cell(16,9);

rep='F:\Base texture avec choix\Bases choix\';
rep=[rep,nom_BD,'\'];
rep=dir(rep);
taille=length(find([rep.isdir]==0));

for numI=1:taille

    %% Chemin image
        chemin= 'F:\Base texture avec choix\Bases choix
            \';
        chemin=[chemin,nom_BD,'\'];
        chemin = [chemin,num2str(numI),'.ppm'];

        image = imread(chemin);

    %% Matrices Luminence
        IM=rgb2gray(image);
        Av=double(IM);
```

```
%% Separation en 16 images
    r=Separation(Av);

%% Boucle sur les 16 images de l'image
for imi=1:16

    Av=Images{imi};
    obtenirMatsousbande0(Av);

    vec={v_D1;v_H1;v_V1;v_D2;v_H2;v_V2;v_D3;v_H3;
       v_V3};

    %% Newton pour le beta sur chaque sous bande de
        chaque images
    for s=1:9

            %Paramtre Newton
            L=v_Long(s);
            m1=(1./L).*sum(abs(vec{s}));
            m2=(1./L).*sum(vec{s}.*vec{s});
            v=vec{s};

            b0=tableF(m1,m2);
            beta0{imi,s}=b0;


            %% Appel fonction Newton
            [b{imi,s},nbiter,bb]=newton('g','dg',b0
                ,v,epsilon);
            kk=b{imi,s};
            a{imi,s}=((b{imi,s}./L).*sum(abs(v).^b{
                imi,s})).^(1./b{imi,s});



            %% Enregistrement base
            a1=a{imi,s};
            b1=b{imi,s};

            Base(numI,imi,s,1)=a1;
```

```
                    Base(numI,imi,s,2)=b1;


            end
        end
end



save('Base.mat','Base');

toc
t=toc;




end
```

Chapitre 8

Bibliographie

- http ://infoscience.epfl.ch/record/33839/files/DoV02
- http ://fr.wikipedia.org/wiki/Vision_par_ordinateur
- http ://www.developpez.net