Université des Sciences de Caen

GREYC Equipe Image Caen

VARIETE DE PATCHS POUR LE TRAITEMENT D'IMAGE

Rapport de stage de Master Recherche LID

Année 2007-2008

Présenté par : Yu Lan

Responsables : M .Olivier Lézoray

M. Aberrahim Elomoataz

M. Vinh Thong Ta

Remerciements

Je tiens d'abord à remercier M.Olivier Lézoray, mon maître de stage, pour son aide, sa disponibilité et ses conseils tout au long de mon stage.

Je tiens également à remercier M. Aberrahim Elomoataz pour m'avoir permis de réaliser mon stage au sein de l'équipe image du GREYC, et pour son encadrement durant mon stage.

Enfin ,je remercie aussi M.Vinh Thong Ta pour m'avoir aidé à mon travail et pour tous les conseils.

Table Des Matières

Introduction Générale

Dans de nombreuses applications en traitement d'image ou de vidéos et en infographie, les pré-traitement des donnée est uns étape importante. Ce pré-traitement peut concerner le débruitage, la restauration, la simplification et aller jusqu'à la compression. Récemment, la prise en compte des interactions non locales dans les algorithmes de pré-traitement a ouvert de nouvelles perspectives qui ont d'ores et déjà des incidences sur ces algorithmes. Par exemple utiliser l'algorithme : le filtre à moyennes non locales[1]. Il remplace un pixel bruité par la moyenne pondérée d'autres pixels de l'image. Les poids de pondération reflètent la similarité entre les voisinages locaux du pixel étant traité et ceux des autres pixels. Il permet de supprimer le bruit tout en préservant la texture et les structures fines.

Dans le cadre de mon stage, nous avons utilisé un « patch » (un carré de taille fix et centré sur un pixel) qui permet tout comme le filtre à moyennes non locales.

Le but de ce stage est de s'attacher à réduire la complexité de telles approches non locales à base de patchs pour des problèmes de traitement d'images par régularisation sur graphes. Dans la suite du document, nous présentons notre approche dans cinq chapitres :

- Construction d'une représentation d'une image.
- Construction de la variété associée à la représentation.
- Définition d'une projection d'un patch quelconque sur la variété.
- Construction d'un graphe de voisinage adapté.
- Régularisation d'image (Retouche d'image dans mon stage)

Chapitre 1 :
Construction d'une représentaion d'une image

Dans ce chapitre, nous allons comencer par définir un bloc de $n \times n$ à partir d'une image, nous l'appelons un « patch» , c'est un carré de taille fixe et centré sur un pixel.

Ensuite, nous allons construire la matrice de similarité W entre tous les patchs de l'image divisée pour l'étape suivante.

1.1 Patch :

Le « patch » est un carré de taille fixe et centré sur un pixel.

Nous utilisons le « patch » comme décomposition de l'image. Les indices (les coordonnées du pixel au centre du patch) de chaque patch sont donc organisés dans un vecteur. La mesure de distance utilisée étant la distance euclidienne et calculée patch par patch.

La distance euclidienne **:**

$$Distance(P,Q) = \sum_{i \in X,Y,Z} \sqrt[2]{(P_i - Q_i)^2} \qquad (1)$$

Où :

- P et Q sont deux patchs carré de taille $n \times n$.
- $\sum_{i \in X,Y,Z} \sqrt[2]{(Pi - Qi)^2}$ correspond à la somme des carrés de toutes les distances sur les composantes X,Y et Z de l'espace couleur (RGB) de tous les pixels des patchs P et Q.

On réécrit formule (2) depuis formule (1).

$$Distance(P,Q) = \sum_{i \in X,Y,Z} \sqrt[2]{\sum_{j=-\frac{N}{2}}^{\frac{N}{2}} \left((p_i + j) - (q_i + j)\right)^2} \qquad (2)$$

Où :

- N est la longeur ou largeur de patch.
- p et q sont les coordonnées du pixel au centre des patchs P et Q. Ils sont de type Point2D en point 2D dans PANDORE[2]. Donc Point2d p correspond à la p.x et p.y
- $\sum_{i\in X,Y,Z}\sqrt{\sum_{j=-\frac{N}{2}}^{\frac{N}{2}}((p_i+j)-(q_i+j))^2}$ correspond à la somme des carrés de toutes les distances sur les composantes X,Y et Z de l'espace couleur (RGB) de tous les pixels des patchs P et Q.

1.2 Similarité :

Pour mesurer la similarité, nous utilisons les distances calculées avec un noyau Gaussien :

$$simi(P,Q) = exp^{-\frac{(Distance(P,Q))^2}{\sigma^2}} \qquad (3)$$

Où :

σ(sigma) est la moyenne de la matrice de distance. Nous avons choisi la moyenne de la matrice de distance après que nous ayons comparé les résultats avec d'autres approches par max ou min.

1.3 Matrice de dégree:

Dans le Laplacian eigenmaps(LE) [3], nous pouvons calculer le degré de chaque patch (l'image est représentée par un graphe complet où les noeuds sont les patchs), avec la similarité calculée par la distance de patch :

$$D(i) = diag\left(\sum_{j=0}^{n-1} simi(p_i, p_j)\right) \qquad (4)$$

Où :

- i est le numéro de ligne.
- j est le numéro de colonne.
- $\sum_{j=0}^{n-1} p_{ij}$ correspond à la somme de chaque ligne dans la matrice de similarité.

1.4 Principe d'utilisation :

➢ ***1ère étape :***

Nous découpons une image originale par le patch de taille de $n \times n$ et calculons le nombre de patchs (Nbp) de l'image divisée par la taille du patch.

➢ ***2ème étape :***

Nous construisons quatre vecteurs listPoint, D, S et U de taille Nbp et trois matrices W, P et V de taille $\mathrm{Nbp} \times \mathrm{Nbp}$.

➢ ***3ème étape :***

Nous cherchons les coordonnées du pixel au centre de tous les patchs par ordre de gauche à droite, de haut en bas dans l'image originale et puis les insérer dans le vecteur listPoint.

➢ ***4ème étape :***

À partir du vecteur listPoint, nous pouvons calculer la distance euclidienne avec la formule (2) et l'insérer dans la matrice W. Une étape importante est le calcul de sigma(σ) depuis la matrice W.

Après l'implémentation, nous choisissons sigma (σ) comme la moyenne de la distance entre tous les patchs.

$$\sigma = \frac{1}{n}\sum_{i=1}^{n} distance(P,Q) \qquad (5)$$

Enfin, nous calculons le degré des noeuds qui va normaliser W pour l'étape suivante. Donc nous utilisons la formule (4) dans le vecteur D.

➢ ***5ème étape :***

Après les étapes précédentes, nous pouvons calculer la similarité avec la formule (3). À la fin du calcul, nous remplissons la matrice W.

W =

p11	p12	.....	.....	p1n
p21	.....	.....	.....	.....
.....	.....	.....	.....	.....
.....	.....	.....	.....	.....
pn1	.....	.....	.....	pnn

Figure 1.1 Matrice de distances entre les patchs

Où :

- p11, la similarité entre le premier patch et lui-même.

- p12, la similarité entre le premier patch et le deuxième patch.
- p12, la similarité entre le deuxième patch et le premier patch,etc...

Chapitre 2 :
Construction de la variété associée à la représentation

Dans le chapitre précédent, après le découpage de l'image originale par le patch, nous avons obtenu la matrice de similarité et le vecteur de degré de tous les patchs de l'image divisée.

Nous allons voir pourquoi il est néccesaire d'avoir la matrice de similarité et le vecteur de degré. Nous parlerons ensuite de la technique de Décomposition en Valeurs Singulières (SVD) pour avoir facilement les informations utiles.

2.1 Marches aléatoires sur graphe :

✧ **Théorie des graphes [4]:**

Nous représentons un ensemble fini d'objets $\Omega = \{ v_1, v_2, ..., v_n \}$ où $v_i \in \mathcal{R}^n$ est un élément de dimension n . Il y a un graphe G = (V,E) qui consiste en un ensemble fini V= Ω possédant N noeuds et un ensemble fini $E \subseteq V \times V$ de M arêtes. Deux noeuds v_i et v_j sont adjacents si l'arête $(v_i, v_j) \in E$, ces deux noeuds sont alors appelés des noeuds voisins.

Un graphe est considéré comme un graphe pondéré, si on peut lui associer une fonction de poids $w : V \times V \rightarrow \mathcal{R}^+$ avec $w(v_i, v_j) = w(v_j, v_i)$ pour chaque noeud $v_i, v_j \in V$. Cette fonction de poids reflète le degré de similarité entre deux noeuds du graphe et décrit ainsi l'interaction du premier ordre entre les noeuds du graphe. Le graphe peut être représenté par sa matrice de similarité W: $W(v_i, v_j) = w(v_i, v_j)$ si les noeuds v_i et v_j sont adjacents et $W(v_i, v_j) = 0$ dans le cas contraire. Les noeuds étant liés à eux-mêmes nous avons pour tout noeud v_i $W(v_i, v_j) = 1$.

Le degré d'un noeud v_i est défini par :

$$d(v_i) = \sum_{v_i \in V} w(v_i, v_j) \qquad (6)$$

✧ **La probabilité de transition :**

Une marche aléatoire sur un graphe est un processus stochastique qui parcourt le graphe en sautant aléatoirement de noeud en noeud [5, 6]. La probabilité de transition d'un noeud v_i vers un noeud v_j est donneé par

$$p_{ij} = p_{v_i v_j} = \frac{w(v_i, v_j)}{d(v_i)} = \frac{w_{v_i v_j}}{d_i} \quad (7)$$

car la probabilité de transition en un saut d'un noeud v_i à un noeud v_j est proportionnelle à $w(v_i, v_j)$, la valution de l'arête reliant ces deux noeuds. La matrice de transition $P = (p_{ij})$ associée est alors définie par

$$P = D^{-1}S \quad (8)$$

Où D est la matrice des degrés des noeuds et S une matrice de similarité associée au graphe. p_{ij} peut être interprétée comme la probabilité de transition du noeud v_i au noeud v_j en saut.

P est généralement symétrique et pour chaque colonne la somme des éléments est de 1. Cette matrice est intéressante car elle reflète la géométrie intrinsèque des données [7]. Une marche aléatoire correspond à une chaîne de Markov homogène puisque les probabilités de transition restent les mêmes à chaque fois que l'on revient sur un noeud du graphe (les probabilités ne dépendant pas d'un facteur temps).

Les chaînes de Markov sont définies en termes d'états et de transitions entre ces derniers. Les états sont dans notre cas les noeuds du graphe. Dans une chaîne de Markov, deux états i et j sont dits communicants si l'on peut atteindre l'un à partir de l'autre avec une probabilité finie; ce qui signifie que le graphe est connexe. Si l'on veut décrire la probabilité de transition $p_t(v_i, v_j)$ d'un noeud v_i à un noeud v_j en t sauts, il suffit de considérer des voisinages plus larges, ce qui correspond à élever la matrice P à la puissance t. Si le graphe est connexe et non bipartite. Alors la marche aléatoire converge vers une distribution stationnaire $\pi = [\pi_1, \ldots\ldots \pi_n]$ satisfaisant $\pi P^t = \pi$ avec

$$\pi_i = \frac{d_i}{vol\ (V)} \quad (9)$$

c'est à dire :

$$\lim_{t \to \infty} p_t(v_i, v_j) = \pi_i \quad (10)$$

2.2 Laplacian eigenmaps :

S'il existe une connexion forte entre les marches aléatoires et le clustering spectral[8]. En effet, les vecteurs propres $\boldsymbol{v}$ de P obtenus en résolvant $P\boldsymbol{v} = \lambda\boldsymbol{v}$ sont exactement les mêmes que ceux obtenus sur le Laplacien normalisé. Le laplacien non normalisé est défini par :

$$\Delta = D - S \quad (11)$$

Et le Laplacien normalisé par

$$L = D^{-1}\Delta = I - D^{-1}S = I - P \tag{12}$$

Les vecteurs propres de $(I - D^{-1}S)\boldsymbol{v} = \lambda'\boldsymbol{v}$ sont donc les mêmes que ceux de la matrice de transition P et l'on a $\lambda = 1 - \lambda'$. Les valeurs propres de P sont $\lambda_1 = 1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq -1$ et on a :

$$P = \sum_{i=1}^{|V|} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T \tag{13}$$

Un Parallèle peut alors être fait entre la matrice de transition P et le critère de coupe normalisé de SHI et MALIK[9]. Le critère de coupe normalisé N_{cut} est un critère pour trouver une coupe optimale des noeuds V d'un graphe en deux sous-ensembles A et Ā et est défini comme suit :

$$N_{cut}(\mathrm{A}, \bar{\mathrm{A}}) = \left(\frac{1}{Vol(\mathrm{A})} + \frac{1}{Vol(\bar{\mathrm{A}})}\right) \sum_{v_i \in \mathrm{A}, v_j \in \bar{\mathrm{A}}} S_{ij} \tag{14}$$

Où :

- ✧ Soit une série de pixels I ; chaque paire de pixel $i, j \in I$ à la similarité S_{ij} $S_{ij} = S_{ji} \geq 0$.
- ✧ Nous avons $d_i = \sum_{j \in I} S_{ij}$ qui est appelé le dégrée de noeud i. Et la valeur de série $A \in I$ est $Vol(\mathrm{A}) = \sum_{i \in A} d_i$
- ✧ Ā est complémentaire de A.

Ce problème étant NP-dur, SHI et Malik ont proposé une méthode spectrale de coupe basée sur le deuxième plus petit vecteur propre du Laplacien dont le signe des éléments définit la coupe. Avec la relation précédente, Cela permet de mettre en relation pourquoi le deuxième vecteur propre est utilisé pour trouver la meilleure coupe et non le premier qui correspond au plus grand vecteur propre $V_1 = 1$ de P et qui ne contient donc aucune information pour trouver une coupe. Le vecteur propre V_1 de P associé à λ_1 est connu comme le vecteur de FIEDLER de P et fournit de l'information concernant la structure en groupes du graphe. Cette mise en relation entre le problème de coupe normalisée et les marches aléatoires est due à MEILA et SHI [10].

À partir d'un ensemble $\{x_1, x_2, \ldots, x_N\}$ de N vecteur de dimension p avec $x_i \in \mathbb{R}^p$, on désire trouver un nouvel ensemble $\{y_1, y_2, \ldots, y_N\}$ de N vecteurs de dimension q avec $y_i \in \mathbb{R}^q$ et $q \ll p$ de manière à ce que y_i représente au mieux x_i. On cherche donc à avoir $||y_i - y_j||_2$ qui soit faible lorsque x_i et x_j sont proches. C'est le principe énoncé par BELKIN et NIYOGI[11] appelé « Laplacian Eigenmaps » qui correspond à minimiser sous contraintes

$$\frac{1}{2}\sum_{ij}||y_i - y_j||_2 S_{ij} = Tr(\boldsymbol{Y}^T \Delta \mathbf{Y}) \qquad (15)$$

avec $\boldsymbol{Y} = [y_1, y_2, \dots, y_N]$. ce critère peut être reformulé et les solutions du précédent problème d'optimisation correspondent à $(D - S)y = \lambda Dy$ et donc à $Py = \lambda y$ pour le Laplacien normalisé, ce qui nous ramène donc directement aux marches aléatoires. Nous pouvons donc définir une transformation de réduction de dimension comme $h : x_i \rightarrow \Big(y_2(i), \dots, y_q(i) \Big)$ où $y_k(i)$ désigne le i^{ième} élément du vecteur propre y_k.

On remarque que l'on ne considère pas le vectur propre y_1 correspondant à $\lambda_1 = 1$ car il ne porte aucune information. q désigne le nombre de vecteurs propres retenus. Grâce à ce type de méthodes, nous pouvons détecter des structures de faibles dimensions dans un espace initial de très grande dimension : ceci est désigné par le terme de manifold learning dans la communauté.

Les Laplacian eigenmaps permettent de détecter des structures non linéaires avec un graphe comme représentation discrète d'une variété (un manifold). LAFON et COIFMAN [6, 12, 13] ont étendu ce principe aux puissances P^t de la matrice de transition P et l'ont nommé « diffusion maps », la figure 2.1 montre, pour un ensemble de points représentant une structure non linéaire discrète (le maintenant classique « swiss roll ») la réduction de dimension obtenue. On remarque que la projection effectuée respecte bien la géométrie initiale des points. Les vecteurs propres obtenus sont également présentés dans la figure 2.1.

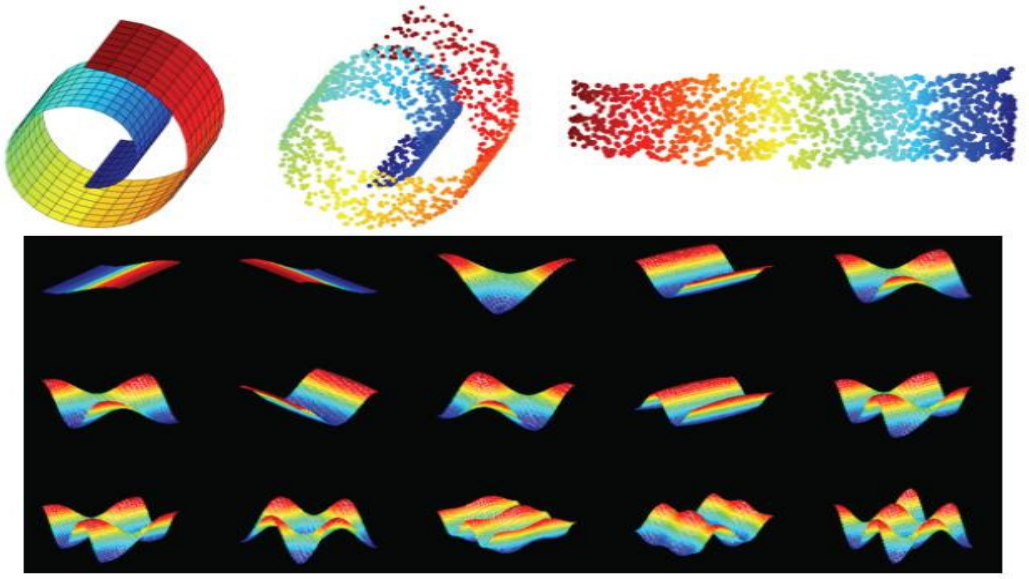

Figure 2.1 Réduction de dimension sur des points provenant d'une distribution de type « swiss-roll »

Dans la figure 2.1, la première ligne présente le SwissRoll, quelques points de

celui-ci et sa représentation après réduction de dimension. Les lignes suivantes présentent les vecteurs propres obtenus.

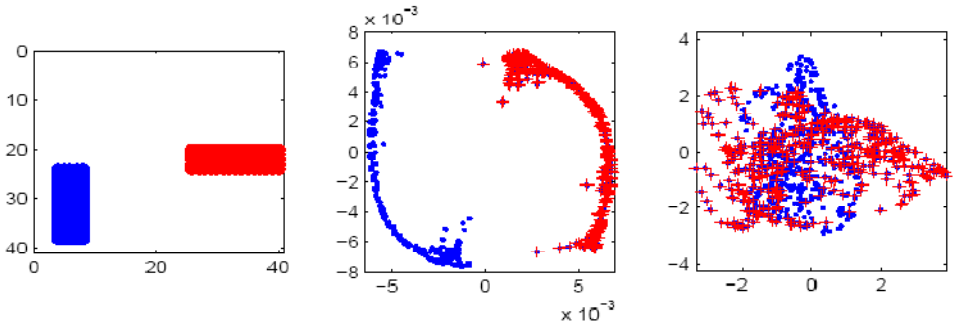

Figure 2.2 les résultat de Laplacian eigenmaps et Analyse en composantes principales

Dans le figure 2.2, un example montre l'intérêt des laplacian Eigenmaps pour la réduction de dimension non linéaire. Les données de l'image initiales sont projetées dans un espace de dimension 2 par Laplacian Eigenmaps et ACP. On remarque que seul les Laplacian Eigenmaps permettent une projection correcte.

2.3 Décomposition en Valeurs propres :

En mathématiques, le procédé d'algèbre linéaire de décomposition en valeurs propres (ou SVD, de l'anglais : Singular Value Decomposition) [14] d'une matrice est un outil important de factorisation des matrices rectangulaires réelles ou complexes. Ses applications s'étendent du traitement du signal aux statistiques, en passant par la météorologie.

Le théorème spectral énonce qu'une matrice normale peut être diagonalisée par une base orthonormée de vecteurs propres. On peut voir cette décomposition comme une généralisation du théorème spectral à des matrices arbitraires, qui ne sont pas nécessairement carrées.

✧ **Introduction [14] :**

Soit X une matrice m×n dont les coefficients appartiennent au corps K, où K = R ou K = C. Alors il existe une factorisation de la forme :

$$X = U\Sigma V^{T} \qquad (16)$$

avec U une matrice unitaire m×m sur K, Σ une matrice m×n dont les coefficients diagonaux sont des réels positifs ou nuls et tous les autres sont nuls (c'est donc une matrice diagonale dont on impose que les coefficients soient positifs ou nuls), et V^{T} est la matrice adjointe à V, matrice unitaire n×n sur K. On appelle cette factorisation la décomposition en valeurs propres de X.

- La matrice V contient un ensemble de vecteurs de base orthonormés pour M, dits « d'entrée » ou « d'analyse » ;
- La matrice U contient un ensemble de vecteurs de base orthonormés pour M, dits « de sortie » ;
- La matrice Σ contient les valeurs propres de la matrice M.

$$\Sigma = \begin{bmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & \\ 0 & & & \ddots \\ & & & 0 \end{bmatrix}$$

Où :

✧ $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$.

Une convention courante est de ranger les valeurs Σi, i par ordre décroissant. Alors, la matrice diagonale Σ est déterminée de façon unique par X (mais U et V ne le sont pas).

Plus généralement, étant donnée une décomposition en valeurs propres de M, alors on a :

$$X^T X = V\Sigma^{\mathrm{T}}\mathrm{U}^{\mathrm{T}}\mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}} = \mathrm{V}(\Sigma^{\mathrm{T}}\Sigma)\mathrm{V}^{\mathrm{T}} \text{ et } XX^T = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}\mathrm{V}\Sigma^{\mathrm{T}}\mathrm{U}^{\mathrm{T}} = \mathrm{U}(\Sigma\Sigma^{\mathrm{T}})\mathrm{U}^{\mathrm{T}}$$

Le côté droit de ces relations décrit la décomposition en valeurs propres du côté gauche. Ainsi, le carré des valeurs propres non-nulles de M est égal au carré des valeurs propres non-nulles de $X^T X$ et de XX^T. En outre, les colonnes de U (vecteurs propres à gauche) sont vecteurs propres pour XX^T, et les colonnes de V (vecteurs propres à droite) sont vecteurs propres de $X^T X$.

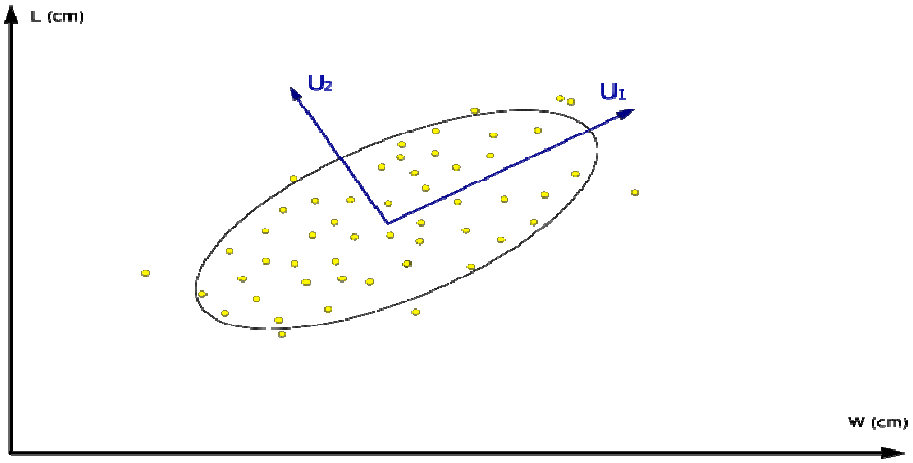

Figure 2.3 Deux premières vecteurs propres de matrice U

Le figure 2.3 représente l'effet de la décomposition SVD sur un ensemble de données, ici la largeur (W) et la hauteur (L) de visages humains. Les vecteurs U1 et U2 sont les deux premiers de la matrice U.

2.4 Principe d'utilisation :

L'utilisation de la décomposition en valeurs singulières est la représentation explicite de l'image et du noyau d'une matrice X. Les vecteurs singuliers à droite correspondant aux valeurs singulières nulles de X engendrent le noyau de X. Les vecteurs singuliers à gauche correspondant aux valeurs singulières non-nulles de X engendrent son image.

Par conséquent, le rang de X est égal au nombre de valeurs singulières non-nulles de X. De plus, les rangs de X, de X^TX et de XX^Tsont égaux. X^TX et XX^T ont les mêmes valeurs propres non-nulles.

Le calcul explicite, analytique, de la décomposition en valeurs singulières d'une matrice est difficile dans le cas général. On utilise, GNU Scientific Library (GSL) pour calculer le SVD.

GNU Scientific Library propose trois alternatives : l'algorithme de Golub-Reinsch, l'algorithme de Golub-Reinsch modifié (plus rapide pour les matrices possédant bien plus de lignes que de colonnes) et l'orthogonalisation de Jacobi [15].

- Ici, nous utilisons toujours la matrice $\mathrm{Nbp} \times \mathrm{Nbp}$.

- ***1ère étape :***

  Nous normalisons la matrice W par le vecteur D, comme $W_{ij} = W_{ij} / D_i$

  Enfin, nous remettons le résultat dans la matrice W.

- ***2ème étape :***

Nous utilisons l'algorithme de l'orthogonalisation de Jacobi de GSL [15]. Après l'utilisation de la fonction de **gsl_linalg_SV_decomp_jacobi** (*gsl_matrix * W, gsl_matrix * V, gsl_vector * S*), nous pouvons obtenir la matrice des vecteurs propres V et le vecteur des valeurs propres S.

- ***3ème étape :***

Nous utilisons les deux premières colonnes de la matrice des vecteurs propres V, pour projeter les patchs dans un espace de dimension 2 (on utilise par conséquent les vecteurs propres deux et trois).

2.5 Expérimentation :

Dans cette partie, nous utilisons deux images pour faire les tests.

Nous présentons les résultats de la projection par Laplacian Eigenmaps dans une nouvelle image de taille 1000*1000 (ce qui revient à appliquer un facteur 1000 aux vecteurs propres obtenus). Nous illustrons ensuite l'influence de taille du patch sur les résultats obtenus.

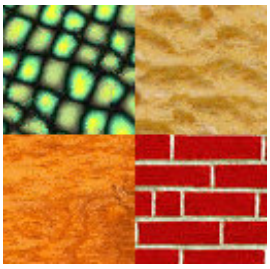

Figure 2.4 Image texture originale en 128 *128 pixels

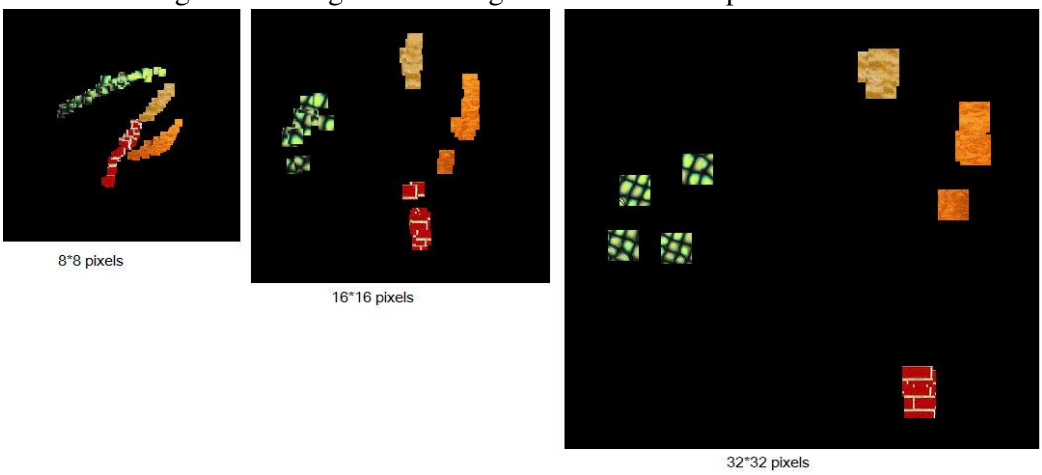

Figure 2.5 Les résultats entre les différentes tailles de patch de l'image 2.4

Figure 2.6 Image originale en 128 * 128 pixel

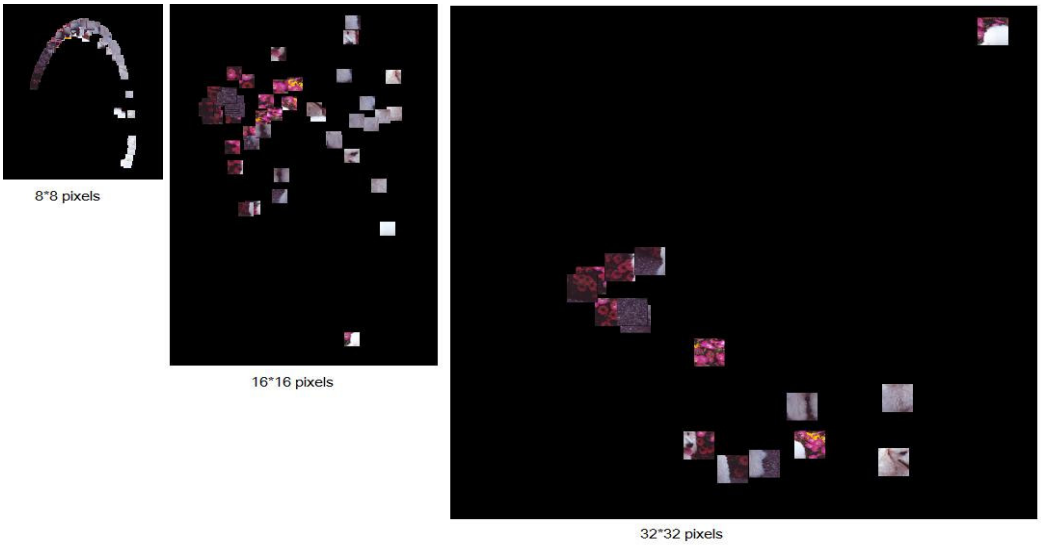

Figure 2.7 Les résultats entre les différentes tailles de patch de l'image 2.6

2.6 Conclusion :

Dans ce chapitre, nous avons considéré la méthode de laplacian eigenmaps pour une projection en deux dimensions. Pour trouver facilement les valeurs propres et les vecteurs propres, nous avons utilisé la technique de SVD.

Puis, dans la partie d'expérimentation, nous avons tenté de définir la valeur de sigma qui est utilisée pour calculer la similarité. Après comparaison des expériences qui utilisent le sigma comme le maximum, la moyenne, ou le minimum de la matrice de distance, on a trouvé que la moyenne de la matrice de distance est la meilleure. Dans le chapitre suivant, nous utiliserons toujours la moyenne pour la valeur de sigma. Nous présenterons des résultats dans la partie annexe.

Enfin, nous avons pu constater que les patchs similaires se regroupent dans la projection. Cependant, avec le découpage régulier de l'image en patchs, nous ne décrivons pas toutes les données initiales, et nous avons besoin de projeter un patch quelconque sur la variété 2D obtenue.

Chapitre 3 :
Définition d'une projection d'un patch quelconque sur la variété

Dans le chapitre précédent, nous avons déjà défini la projection de patchs provenant d'un découpage régulier d'une image. À partir de cette projection, nous devons définir une projection d'un patch quelconque.

3.1 Patch quelconque :

D'abord, nous devons construire le patch quelconque depuis une coordonnée bien définie. Cette coordonnée est le centre du patch, et construite avec la taille du patch.

3.2 Similarité :

Nous utilisons la distance euclidienne pour calculer les distances entre le patch d'entrée et les autres patchs dans le vecteur listPoint.

$$Distance(P,Q) = \sum_{i \in X,Y,Z} \sum_{j=0}^{n-1} \sqrt[2]{\left(P_i - Q_{ij}\right)^2} \qquad (17)$$

où :

- P est le nouveau patch carré est construit par le coordonnée d'entrée.
- Q_j est la coordonnée de $j^{ème}$ patch dans le vecteur listPoint.
- $\sum_{i \in X,Y,Z} \sum_{j=0}^{n-1} \sqrt[2]{\left(P_i - Q_{ij}\right)^2}$ correspond à la somme des carrés de toutes les distances sur les composantes X,Y et Z de l'espace couleur (RGB) de tous les pixels des patchs P et Q.

Ensuite, nous utilisons la formule (10) pour calculer la similarité entre le patch d'entrée et les autres patchs.

3.3 Méthode de Nyström :

La méthode de Nyström [16] est une technique utilisée pour trouver des approximations numériques aux problèmes de la fonction propre, la formule est :

$$\int_a^b k(x, x_i)\phi(x_i)dy = \lambda\phi(x) \qquad (18)$$

Après que nous changeons de l'intervalle [a,b] par [0,1], et construisons le système comme une matrice valeur propre :

$$K\hat{\Phi} = \hat{\Phi}\Lambda \qquad (19)$$

3.4 Nyström extension :

Nous pouvons avoir l'extension de Nyström avec les conditions ci-dessous depuis la formule (19) :

- ✧ $K_{ij} = k(x_i, x_j)$ est une matrice.
- ✧ $\Phi = [\varphi_1, \varphi_2, \ldots\ldots, \varphi_n]$ est n vecteur propre qui correspond à la valeure propre $[\lambda_1, \lambda_2, \ldots\ldots, \lambda_n]$.

Soit $x \in \mathbb{R}^d$ un nouveau point d'entrée qui n'est pas dans l'ensemble d'apprentissage. L'extension de Nyström, [17], la j^{ème} coordonnée de la projection noyau φ pour ce point comme :

$$\hat{\varphi}_j(x) = \frac{1}{\sqrt{\lambda_j}} \sum_{i=1}^{n} k(x, x_i)\phi_j(x_i) \qquad j = 1, \ldots, n, \qquad (20)$$

Ou bien le vecteur comme:

$$\hat{\varphi}(x) = \frac{1}{\sqrt{\Lambda}} U^T k_x, \qquad (21)$$

Où :

- ✧ $k_x = (k (x, x_1), k (x, x_2), ..., k (x, x_n))$, la relation du noyau $k (x, x_1)$ présente une similarité entre le nouveau point d'entrée x et le point x_1 dans l'ensemble d'apprentissage.

✧ $\frac{1}{\sqrt{\Lambda}} = \sqrt{\Lambda}^{-1} = \text{diag}\left(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \ldots\ldots, \frac{1}{\sqrt{\lambda_n}}\right)$

3.5 Principe d'utilisation :

➢ ***1ème étape :***

D'abord, nous construisons un vecteur np pour mettre la similarité du patch d'entrée et les autres patchs dans le vecteur listPoint.

➢ ***2ème étape :***

À partir du vecteur listPoint, nous pouvons calculer la distance euclidienne entre le patch d'entrée et les patchs dans le vecteur listPoint avec la formule (10).

➢ ***3ème étape :***

Ensuite, nous mettons la distance dans le vecteur np .

Une étape importante est le calcul de sigma(σ) depuis le vecteur np.

Comme que nous avons déjà calculé dans l'étape précédente, nous choisissons aussi sigma(σ) comme la moyenne de la distance entre le patch d'entrée et touts les autres patchs.

➢ ***4ème étape :***

Après les étapes précédentes, nous pouvons calculer la similarité avec la formule(3). À la fin du calcul, nous remplissons dans le vecteur np.

➢ ***5ème étape :***

Enfin, depuis le vecteur np, nous pouvons mesurer la projection de ce patch avec formule (15), donc la fonction $k\,(x,\ x_i)$ est la similarité du nouveau patch x et les autres patchs dans le list listpoint x_i. Pour son utilisation, nous avons calculé et mis au vecteur np par ordre. λ_i est le valeur propre qui correspond au patch x_i. Comme au chapitre précédent, nous avons utilisé la technique SVD, donc nous pouvons remplir le vecteur S par les valeurs propres. $\phi_i\,(x_i)$ est le vecteur propre qui correspond au patch x_i. De la même façon, nous avons la matrice V qui est remplie par les vecteurs propres. Après le calcul, nous pouvons avoir la projection de ce patch dans le nouveau espace de dimensions deux.

3.6 Expérimentation :

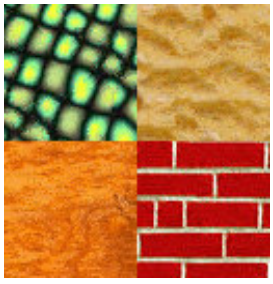

Figure 3.1 Image texture originale en 128 * 128 pixels

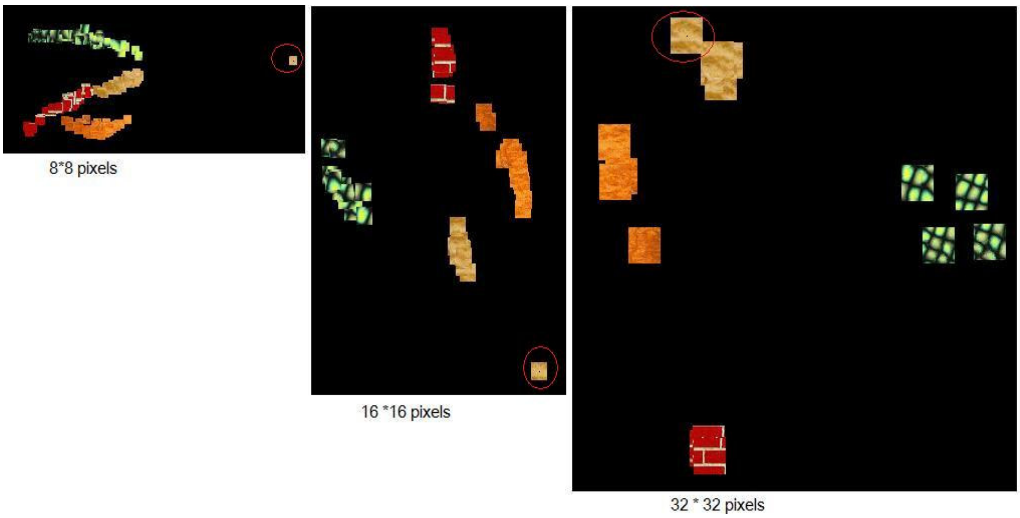

Figure 3.2 Un patch d'entrée de point au centre du (110, 40) et les différentes tailles de patch.

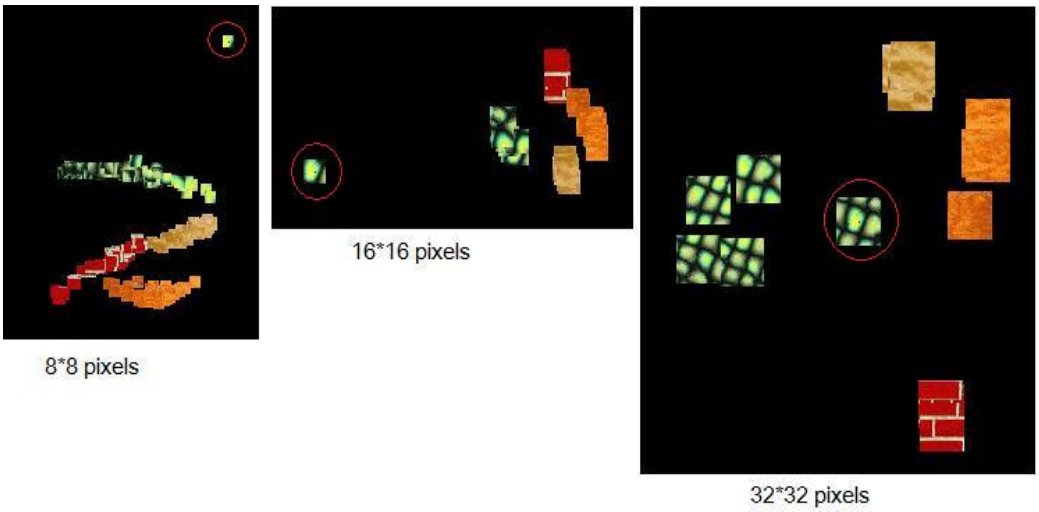

Figure 3.3 Un patch d'entrée de point au centre du (40, 40) et les différentes tailles de patch

- les patchs entourés d'un cercle rouge sont les patchs d'entrée projetés sur la variété existante.

3.7 Conclusion :

Dans ce chapitre, nous avons présenté la méthode de l'extension de Nyström. Grâce à cette méthode, nous pouvons projeter un patch quelconque depuis la projection que nous avons fait avec les patchs principaux.

Par contre dans ce chapitre, nous devons calculer les similarités entre ce patch d'entrée et les autres patchs principaux.

Ensuite, nous avons vu dans la partie d'expérimentation comment la taille du patch peut donner des résultats différents.

Chapitre 4:
Construction d'un graphe de voisinage adapté

4.1 Algorithme K Plus Proche Voisin :

L'algorithme K Plus Proche Voisin (KNN de l'anglais : K Nearest Neighbor) est le plus simple de toutes les algorithmes des machines d'apprentissage.

- ✧ Faire une classification sans faire d'hypothèse sur la fonction $y = f(x_1, x_2, \dots \dots x_p)$ qui relie la variable dépendante y aux variables indépendantes$(x_1, x_2, \dots \dots x_p)$

- ✧ Méthode de classification non-paramétrique puisqu'aucune estimation de paramètres n'est nécessaire comme pour la régression linéaire.On dispose de données d'apprentissage (training data) pour lesquelles chaque observation dispose d'une classe y. Si le problème est à 2 classes, y est binaire.

- ✧ L'idée de l'algorithme des knn est pour une nouvelle observation $(u_1, u_2, \dots \dots u_p)$ de prédire les k observations lui étant les plus similaires dans les données d'apprentissage.

- ✧ ...et utiliser ces observations pour classer l'observation dans une classe C.

- ✧ Si on connaissait la fonction f, on aurait juste qu'à calculer $C = f(u_1, u_2, \dots \dots u_p)$

- ✧ En pratique on peut identifier les observations des données d'apprentissage, collecter les valeurs y associées à ces observations (et procéder à une sorte d'interpolation). Quand on parle de voisin cela implique la notion de distance ou de dissimilarité.

- ✧ La distance la plus populaire est la distance euclidienne.

- ✧ Le cas le plus simple est k=1 (cas 1-NN). On cherche l'observation la plus proche et on fixe C = y.

- ✧ On peut montrer que si on disposait d'un très gros volume de données d'apprentissage, et en utilisant une règle de classification arbitrairement sophistiquée, on ne diminuerait l'erreur de mauvaise classification que d'un

facteur 2 par rapport à une méthode 1-NN.

✧ L'extension de 1-NN à k-NN se fait comme suit :
(1) Trouver les k plus proches observations
(2) Utiliser une règle de décision à la majorité pour classer une nouvelle observation

✧ L'avantage est que de grandes valeurs de k produisent un lissage qui réduit le risque de surapprentissage dû au bruit dans les données d'apprentissage.

✧ Typiquement les valeurs de k sont choisies dans des échelles de quelques unités à quelques dizaines plutôt que quelques milliers.

✧ Si on choisit k = n (nombre d'observations dans l'ensemble d'apprentissage) la classe retenue sera celle qui a la majorité dans les données d'apprentissage, indépendamment de l'observation inconnue $(u_1, u_2, \ldots\ldots\ u_p)$

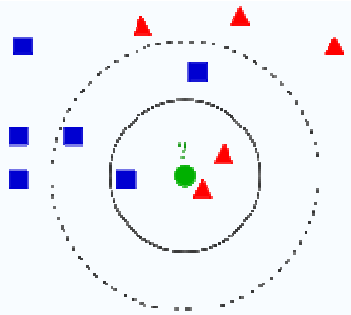

Figure4.1 KNN Classification

La figure 4.1 présente une exemple de la classment KNN. La prise d'essai (cercle vert) devrait être classée à la première classe de carrés bleus ou à la deuxième classe de triangles rouges. Si k = 3, il est classé à la deuxième catégorie car il y a 2 triangles et 1 carré seulement dans le cercle intérieur. Si k = 5, il est classé en première classe (3 places par rapport à 2 triangles dans le cercle extérieur).

On note deux aspects importants de l'algorithme KNN :

- D'une part, à chaque nouvelle classification il est nécessaire de parcourir l'ensemble de la base d'apprentissage, ce qui en fait un algorithme qui n'est pas nécessairement très efficaces (surtout que, habituellement, on cherche à avoir la base d'apprentissage la plus grande possible afin d'avoir un meilleur classifieur).
- Et d'autre part, un point crucial de cet algorithme est la fonction de distance utilisée pour mesurer la proximité des objets. Il n'existe pas de distance/similarité universellement optimale et une bonne connaissance du problème traité guide généralement le choix de cette distance/similarité.

4.2 Principe d'utilisation :

Dans le chapitre précédent, nous avons déjà projeté un patch quelconque et calculé la similarité avec les autres patchs.

À partir de cette projection, nous pouvons utiliser la méthode KNN pour chercher les k plus proches voisins, c'est-à-dire chercher les k plus petites distances entre la projection du patch d'entrée et tous les autres patchs dans ce nouveau espace de dimension deux. Nous utilisons simplement la fonction de GSL[18] **gsl_sort_vector_smallest_index** (*size_t * p, size_t k, const gsl_vector * v*). Elle nous permet de trouver facilement l'index qui correspond aux patchs dans ces k plus petites distances.

4.3 Expérimentation :

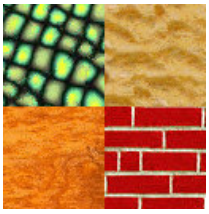

Figure 4.2 Image texture originale en 128 * 128 pixels

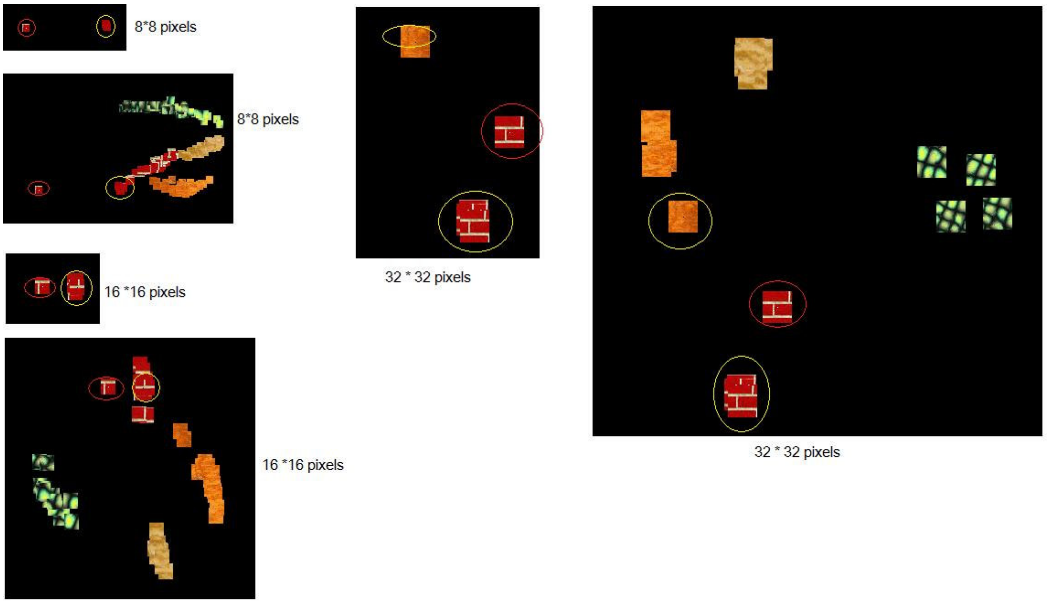

Figure 4.3 Les résultats entre différentes tailles de patch et les 5 plus proches voisins (dans le cercle jaune).

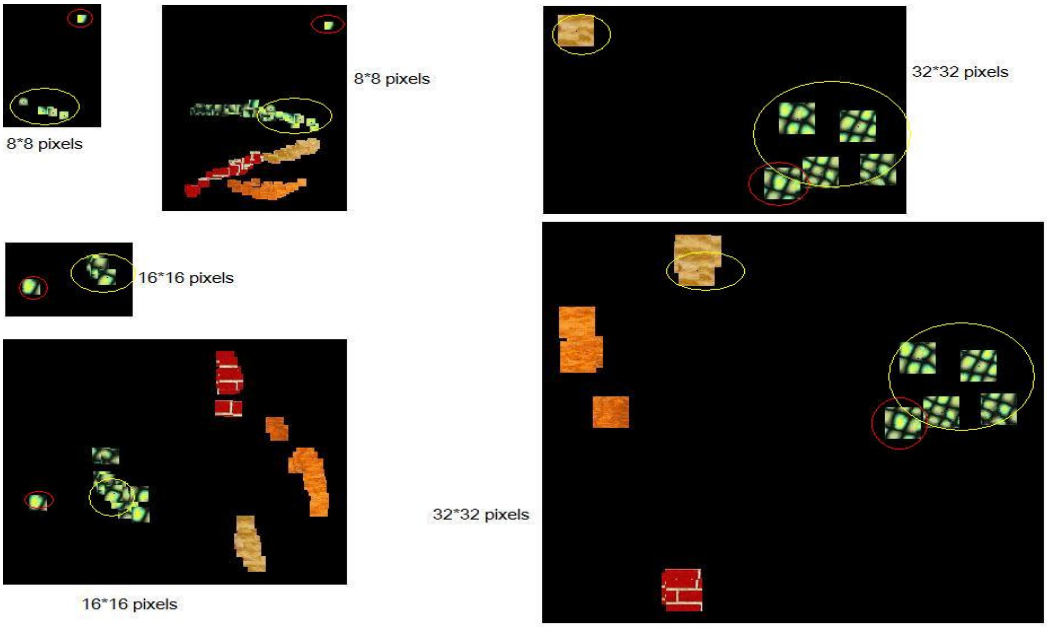

Figure 4.4 Les résultats entre différentes tailles de patch et les 5 plus proches voisins (dans le cercle jaune).

Figure 4.5 Image originale en 128 * 128 pixels

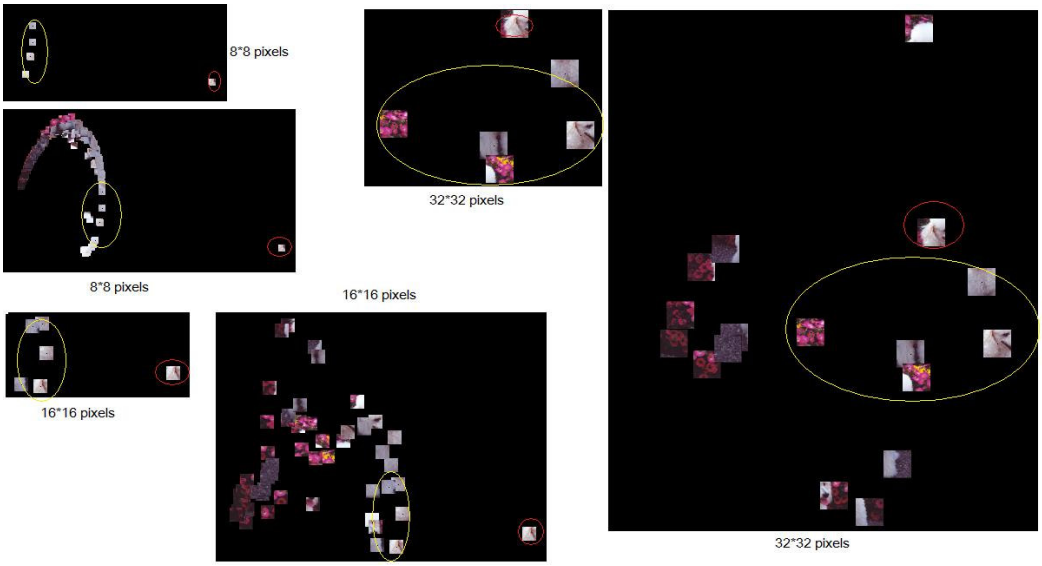

Figure 4.6 Les résultats entre différentes tailles de patch et les 5 plus proches voisins (dans le cercle jaune).

4.4 Conclusion :

Dans ce chapitre, nous avons présenté la méthode de K Plus Proche Voisin**,** il est le plus simple de toutes les algorithmes des machines d'apprentissage. Donc il peut nous aider pour trouver le résultat rapidement. Enfin, nous avons utlisé une fonction de GSL[14] **gsl_sort_vector_smallest_index** (*size_t * p, size_t k, const gsl_vector * v*). Elle nous aide de trouver facilement l'index qui correspond aux patchs dans ces k plus petites distances.

La distance que nous avons vu ici, c'est le distance des projections entre les patchs principaux et le patch d'entrée.

Chapitre 5 :
Inpainting d'image (Retouche d'image)

Dans les chapitres précédents, nous avons déjà fait plusieurs expérimentations. Ainsi, nous avons essayé de retoucher deux images qui ont des parties manquantes. Nous considérons deux images, une image à laquelle il manque une partie au centre, une autre à laquelle il manque le contour de l'image.

5.1 Le contour de partie manquée d'image :

- **Image avec trou :**

Pour commencer nos travaux, il faut d'abord trouver le contour du trou d'image incomplet. Il existe déjà des opérateurs dans PANDORE[19] permettant d'effectuer ceci.

- ✓ D'abord, nous avons besoin de l'image d'entrée en binaire, PANDORE nous propose un opérateur « pbinarization ». Par contre, « pbinarization » peut fonctionner qu'avec les images en gris. Donc, nous devons changer l'image en couleur à l'image en gris avec un opérateur « pimc2img ».
- ✓ Puis, utiliser un modèle de morphologie mathématique erosion. Pour cela, nous pouvons utiliser l'opérateur « perosion ».
- ✓ Enfin, nous prenons la différence entre l'image en gris et l'image après l'effet « erosion ». Avec l'utilisation de «pdif », nous pouvons avoir un contour du trou de l'image.
- ✓ Nous prenons tous les coordonnées des pixels sur ce contour dans un vecteur.
- ✓ Avec ces coordonnées, nous allons passer au travail suivant.

- **Image sans contour :**

Pour l'image qui manque de contour, nous avons utilisé des opérateurs différents que l'image avec trou.

- ✓ Cette partie, nous devons changer aussi l'image en binaire comme nous avons fait dans la partie de l'image avec trou .
- ✓ Ensuite, nous utilisons l'opérateur « pinverse » pour la dilatation.
- ✓ Puis, en utilisant le modèle de morphologie mathématique dilatation. Pour cela, nous pouvons utiliser l'opérateur « pdilatation »
- ✓ Ici, nous devons utiliser l'opérateur « pinverse » pour comparer après.
- ✓ Enfin, nous prenons la différence entre l'image en gris et l'image après « inverse », avec l'utilisation de «pdif ». Nous pouvons avoir un contour de trou d'image gris.
- ✓ Nous prenons tous les coordonnées des pixels sur ce contour dans un

vecteur.

- ✓ Avec ces coordonnées, nous allons faire le travail suivant.

5.2 La projection :

Une fois que nous avons obtenu les coordonnées des pixels sur le contour de l'image d'entrée, nous pouvons faire comme ce que nous avons fait au chapitre 4, et cherchons les 5 plus proches voisins du patch qui est centré un pixel sur le contour. Nous remarquons que la matrice de similarité des patchs principaux est juste calculée à partir des parties complètes.

Dans un premier temps, nous remplaçons la moyenne des 5 pixels au centre des patchs qui sont les 5 plus proches voisins du patch d'entrée.

Mais le résultat n'est pas très bon, Nous cherchons donc à reconstruire le patch :

- ✓ D'abord, nous construisons le patch depuis un pixel sur contour.
- ✓ Puis, nous pouvons remplire la partie manquée par la partie symétrique au centre.
- ✓ Nous recalculons la similarité depuis ces nouveaux patchs.

Après l'implémentation, nous avons obtenu des résultats qui semblent acceptables.

Afin d'avoir la similarité, pour chaque nouveau patch, nous pouvons le projeter et nous allons trouver ses 5 plus proches voisins. À partir de ces voisins nous calculons la moyenne des pixels au centre et nous remplaçons le pixel au centre du nouveau patch par cette moyenne.

Nous continuons en boucle :

- ✓ Chercher le contour de la partie manquante.
- ✓ Mettre les coordonnées des pixels se trouvant sur ce contour dans un vecteur.
- ✓ Construire les patchs depuis ce vecteur.
- ✓ Reconstruire ces patchs.
- ✓ Calculer les similarités et projeter ces patchs.
- ✓ Chercher les 5 plus proches voisins de chaque patch et calculer la moyenne des pixels au centre de ces 5 patchs voisinaux.
- ✓ Et enfin, remplacer les anciens pixels au centre par la moyenne des pixels calculée auparavant.

Jusqu'à ce que nous ne puissions plus trouver aucun pixel vide.

5.3 Expérimentation :

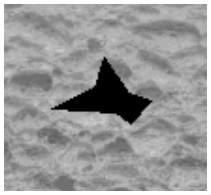

Figure 5.1 Image d'entrée

Figure 5.2 Avec un patch en taille 16*16 pixel

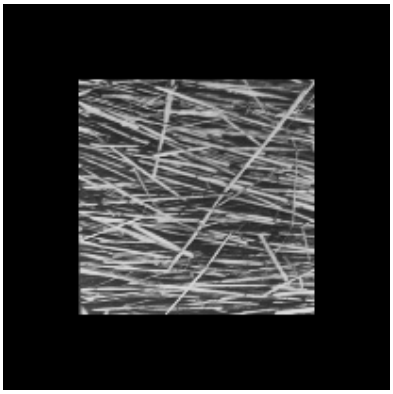

Figure 5.3 Image d'entrée

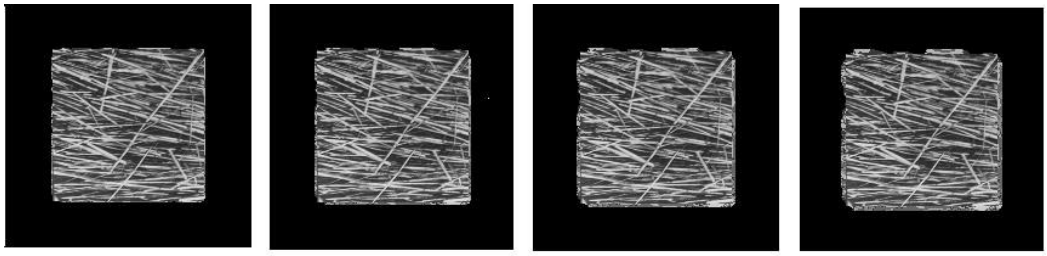

Figure 5.4 Avec un patch en taille 16*16 pixel

5.4 Conclusion :

Dans la première partie, nous avons pu obtenir un résultat acceptable de retouche d'image. En changeant la taille du patch on obtient des résultats différents, présentés en annexe.

Par contre, dans l'autre cas, cela ne fonctionne pas bien. Nous allons essayer d'améliorer nos programme dans le futur pour avoir un bon résultat.

Conclusion:

Dans ce rapport, nous avons présenté une méthode exploitant une variété de patchs extraits d'une image. Cette méthode permet de fonctionner comme l'algorithme « filtre à moyenne non local». Et nous avons déjà réalisé une application dans le domaine de l'inpainting d'image. Durant l'étude de cette méthode, nous avons combiné plusieurs techniques, par exemple, LE(Laplacian eigenmaps), SVD(Décomposition en Valeurs propres) et enfin le KPPV (k plus proch voisin), etc.

Dans un premier temps, avant l'application au traitement d'image, nous avons décidé de couper l'image échantillon par le patch avec la taille du patch considéré. Nous calculons les similarités entre les patchs. Une fois que nous avons obtenu la matrice de similarité, nous pouvons projeter les patchs avec ses vecteurs propres d'après la technique de SVD. C'est comme un espace de dimension 2 de patch échantillon.

Ensuite, nous devons calculer le pixel que nous voulons traiter. D'abord, nous constuisons un patch depuis les coordonnées du pixel d'entrée, puis il faut calculer ses similarités avec les patchs échantillons. Comme nous l'avons vu, avec son vecteur de similarité, nous pouvons projeter ce patch à l'aide de l' extension de Nyström.

Puis , nous avons obtenu la projection entre tous les patchs échantillons et le patch d'entrée. Après la méthode de KPPV, nous pouvons obtenir les 5 patchs de plus proches voisins (ici, nous considérons que k = 5) dans cette projection. Nous remarquons ici, la distance est la distance de projection entre les patchs que nous avons mesuré dans le KPPV.

Enfin, nous calculons la moyenne des 5 pixels au centre des 5 patchs que nous avons trouvé . Quand nous remplaçons le pixel d'entrée par cette moyenne, notre travail est terminé.

Les expérimentations ont montré les potentialités de la méthode même si les résultats ne sont pas satisfaisants pour le moment.

Bibliographie:

[1]T. BUADES, B. COLL et J-M. MOREL "*A review of image denoising algorithms, with a new one"*, à paraître dans Multiscale Mathematical Modelling, 2004, prépublication No 2004-15 du CMLA http://www.cmla.ens-cachan.fr/Cmla/, No 2004-15. L'algorithme et ses applications font l'objet d'une demande de brevet déposé le 5 Mai 2004.

[2] R. CLOUARD, A.ELMOATAZ, F.ANGOT, , A.DURET-LUTZ, et O.LEZORAY, "*PANDORE Handbook for version 6.3"*, GREYC équipe Image, Mars 18 2008.

[3] J. SHI et J.MALIK "*A random walks view of spectral segmentation*" In AI and Statistics (AISTATS), 2001

[4]Y. CHAHIR , Y. ZINBI et K. E. AZIZ "*Catégorisation des expressions faciales par marches aléatoires sur graphe*" Papier soumis à CORESA 2007

[5]U. VON LUXBURG. "*A Tutorial on Spectral Clustering"* . Rapport technique 149, Max Planck Institute for Biological Cybernetics, 2006. To appear in Statistics and Computing.

[6]R. COIFMAN et S. LAFON. "*Diffusion maps"*. Applied and Computational Harmonic Analysis : Special issue on Diffusion Maps and Wavelets, vol. 21, pages 5–30, 2006.

[7]A. ROBLES-KELLY et E. R. HANCOCK. "*String Edit Distance, Random Walks And Graph Matching"*. IJPRAI, 18(3), pages 315–327, 2004.

[8]M. HEIN, J.-Y. AUDIBERT et U. VON LUXBURG. "*From Graphs to Manifolds-Weak and Strong Pointwise Consistency of Graph Laplacians*". Dans COLT,pages 470–485, 2005.

[9]J. SHI et J. MALIK. " *Normalized Cuts and Image Segmentation*". IEEE Trans. Pattern Anal. Mach. Intell., 22(8), pages 888–905, 2000.

[10]M. MEILA et J. SHI. "*A random walks view of spectral segmentation*". Dans In AI and Statistics (AISTATS), 2001.

[11]M. BELKIN et P. NIYOGI. "*Laplacian Eigenmaps for Dimensionality Reductionand Data Representation*". Neural Computation, 15(6), pages 1373–1396, 2003.

[12] S. LAFON, Y. KELLER et R. R. COIFMAN. "*Data Fusion and Multicue Data Matching by Diffusion Maps*". IEEE Trans. Pattern Anal. Mach. Intell., 28(11), pages 1784–1797, 2006.

[13]S. LAFON et A. LEE. "*Diffusion Maps and Coarse-Graining : A Unified Framework for Dimensionality Reduction, Graph Partitioning and Data Set Parameterization*". IEEE Trans. Pattern Anal. Mach. Intell., 28(9), pages 1393–1403, 2006.

[14] "*Décomposition en valeurs singulières*" *http://fr.wikipedia.org/wiki/D%C3%A9composition_en_valeurs_singuli%C3%A8res*

[15] *"Manuel de la GSL concernant la SVD"*. http://www.gnu.org/software/gsl/manual/html_node/Singular-Value-Decomposition.html#Singular-Value-Decomposition

[16] C. FOWLKES, S. BELONGIE et J. MALIK. "*Efficient Spatiotemporal Grouping Using the Nyström Method"* Comput. Vision and Pattern Recognition, Hawaii, December 2001
[17] P.ARIAS, G. RANDALL,et G.SAPIRO. "*Connecting the Out-of-Sample and Pre-Image Problems in Kernel Methods"* Comput. Vision and Pattern Recognition, Hawaii, December 2001
[18]"*Manuel de la GSL de Selecting the k smallest or largest elements",* http://www.gnu.org/software/gsl/manual/html_node/Selecting-the-k-smallest-or-largest-elements.html
[19] R. CLOUARD, A. ELMOATAZ, F. ANGOT, A. DURET-LUTZ, O. LEZORAY, S .SCHÜPP, S.BOUGLEUX, P .BELHOMME, L.QUESNEL et J.FADILI. "*Index des opérateurs*". Rapport de recherche, GREYC équipe Image, août 2004.

.

Annexe :

A.1 Les images orginaux :

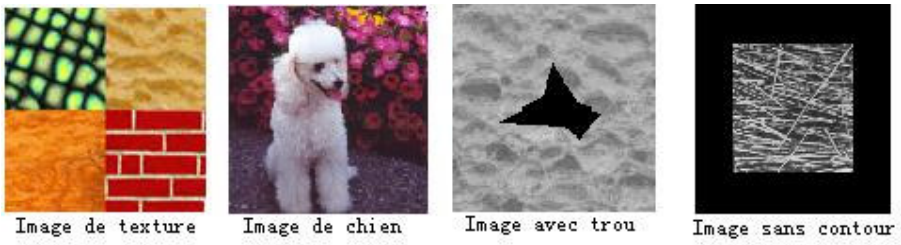

Figure A.1 Les image que nous avons utilisé.

A.2 Les projection des différents tailles de patch et les différents sigma(σ) :

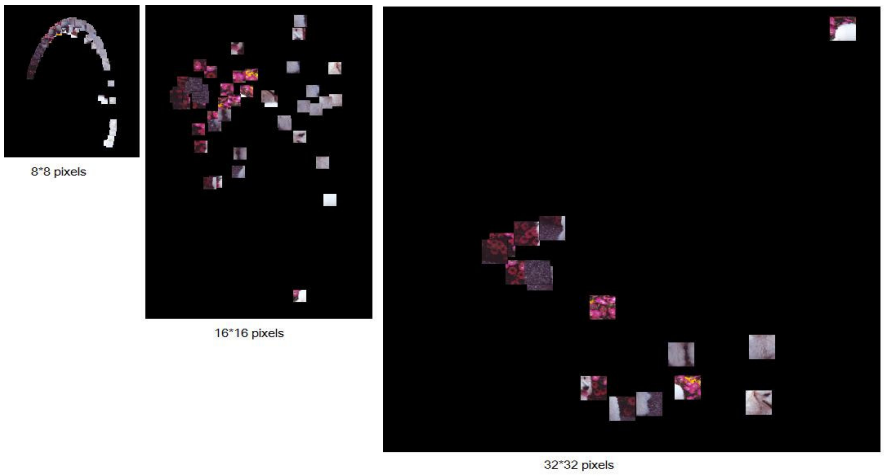

Figure A.2.1 Avec sigma est la moyenne de similarité des patchs.

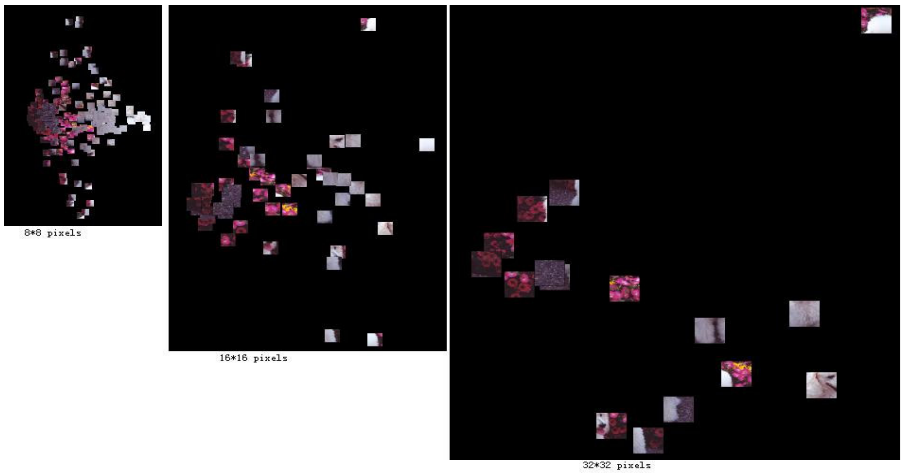

Figure A.2.2 Avec sigma est le maximum de similarité des patchs.

Figure A.2.3 Avec sigma est la moyenne de similarité des patchs.

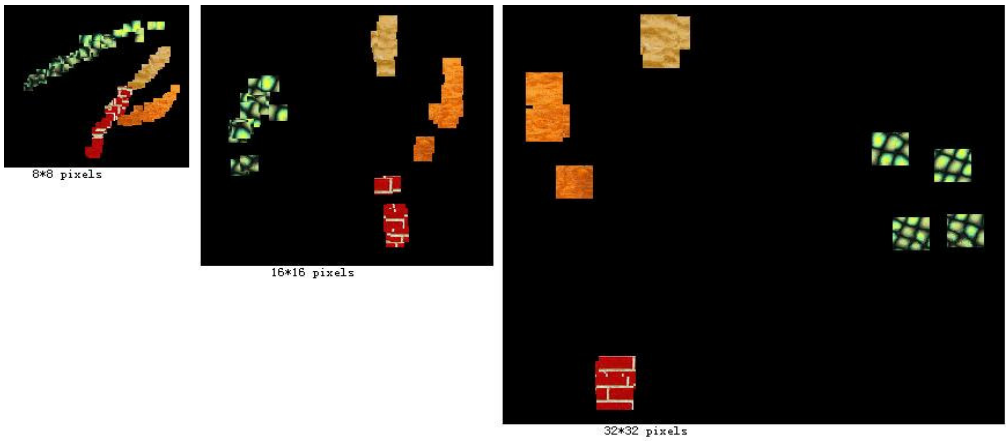

Figure A.2.4 Avec sigma est le maximum de similarité des patchs.

A.2 Les projections des différents tailles de patch et les différents patchs d'entrée:

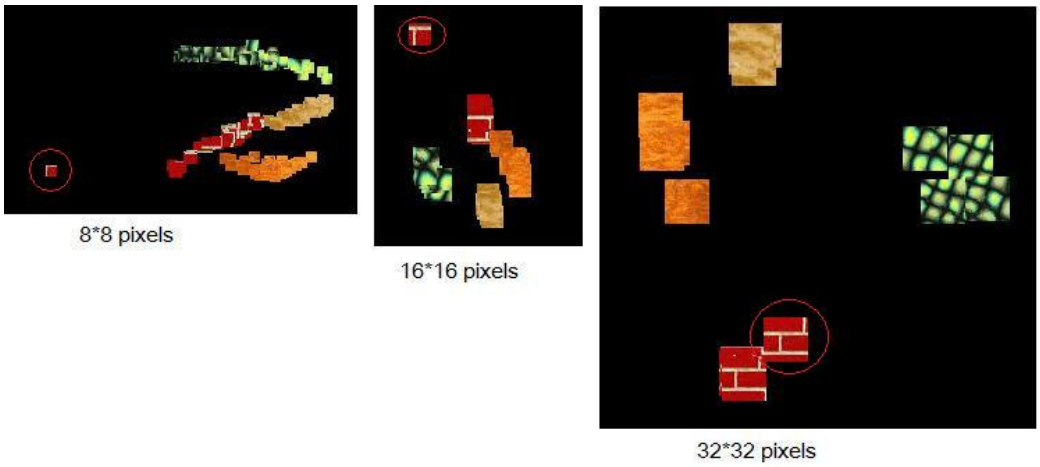

Figure A.3.1 La projection de touts les patchs et le patch d'entrée au centre (110,110).

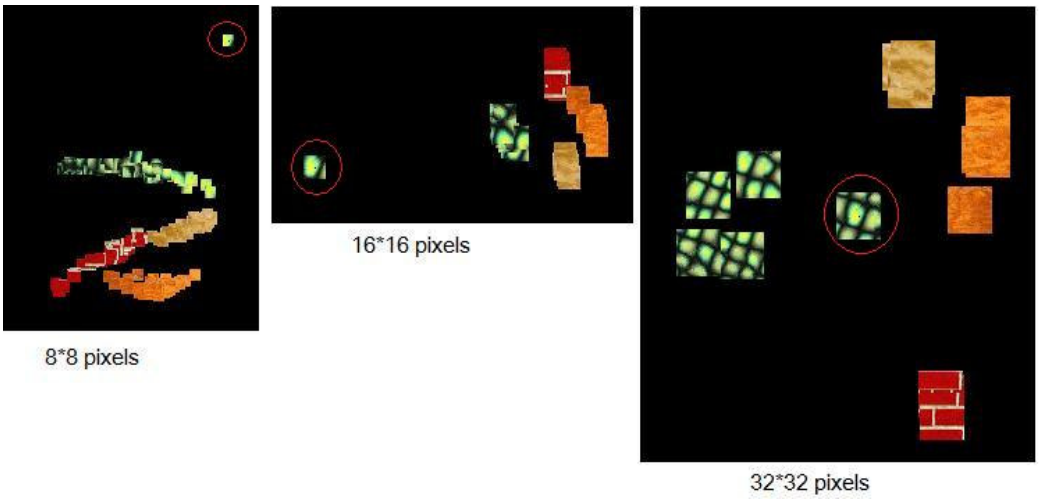

Figure A.3.2 La projection de touts les patchs et le patch d'entrée au centre (40,40).

Figure A.3.3 La projection de touts les patchs et le patch d'entrée au centre (110,40).

Figure A.3.4 La projection de touts les patchs et le patch d'entrée au centre (40,110).

A.4 Les 5 plus proches voisins du patch d'entrée différent avec les différents tailles de patch :

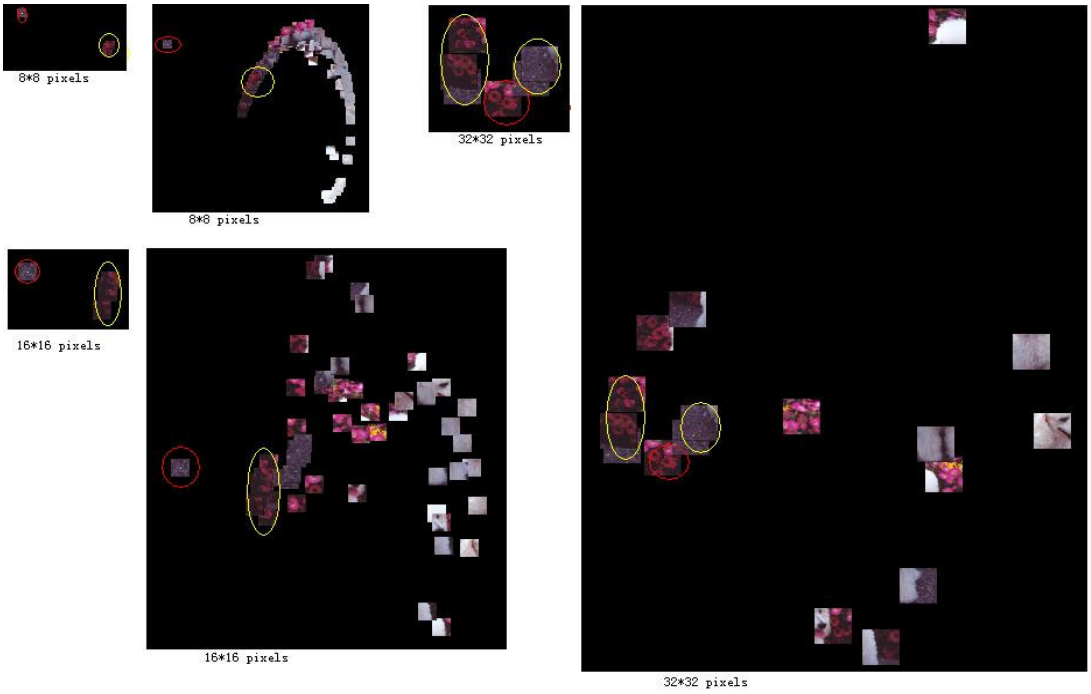

Figure A.4.1 La projection du patch d'entrée au centre (110,110) et ses 5 plus proches voisins.

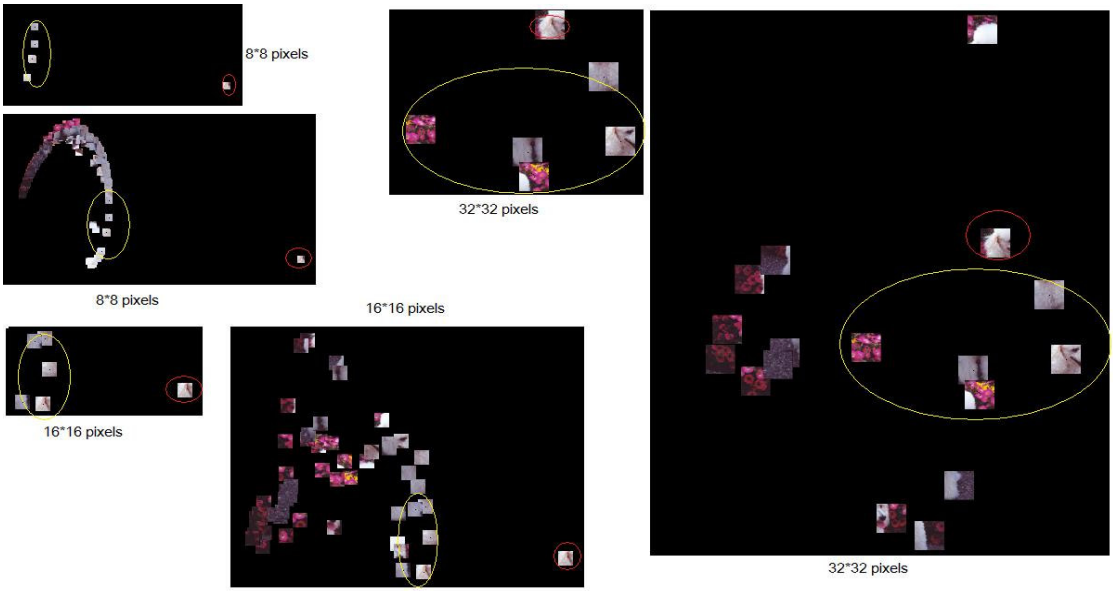

Figure A.4.2 La projection du patch d'entrée au centre (40,40) et ses 5 plus proches voisins.

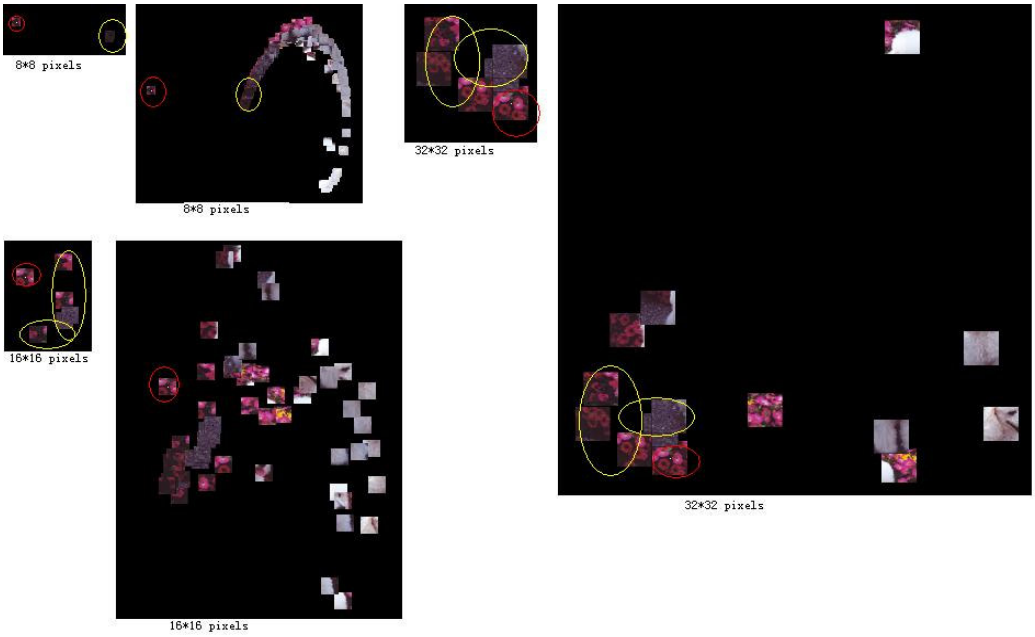

Figure A.4.3 La projection du patch d'entrée au centre (110,40) et ses 5 plus proches voisins.

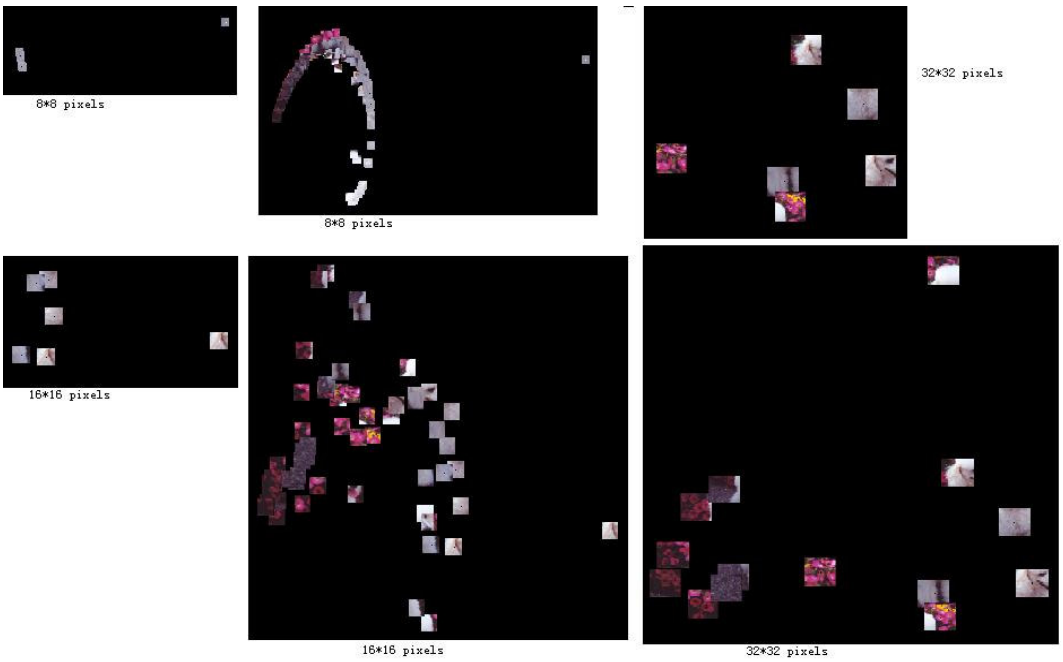

Figure A.4.4 La projection du patch d'entrée au centre (40,110) et ses 5 plus proches voisins.

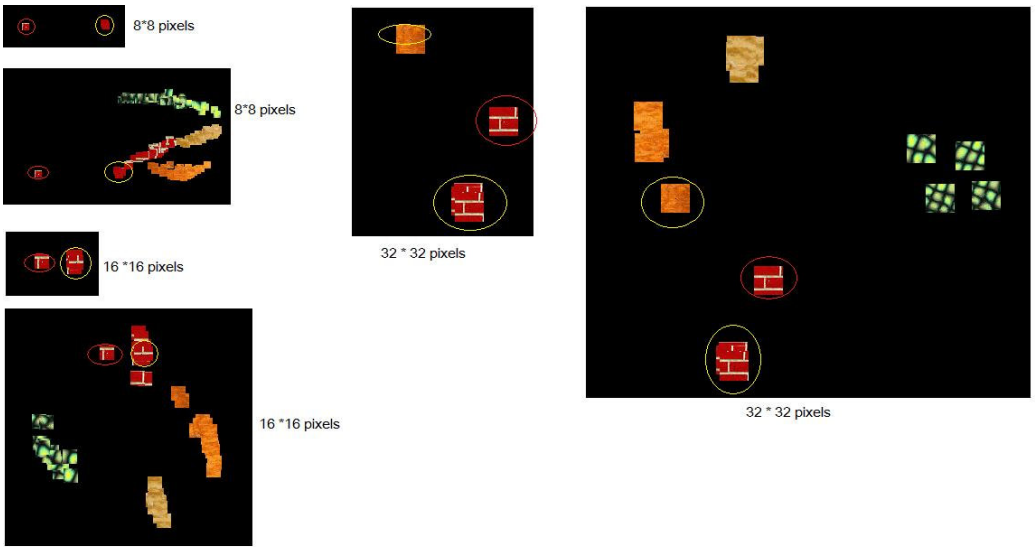

Figure A.4.5 La projection du patch d'entrée au centre (110,110) et ses 5 plus proches voisins.

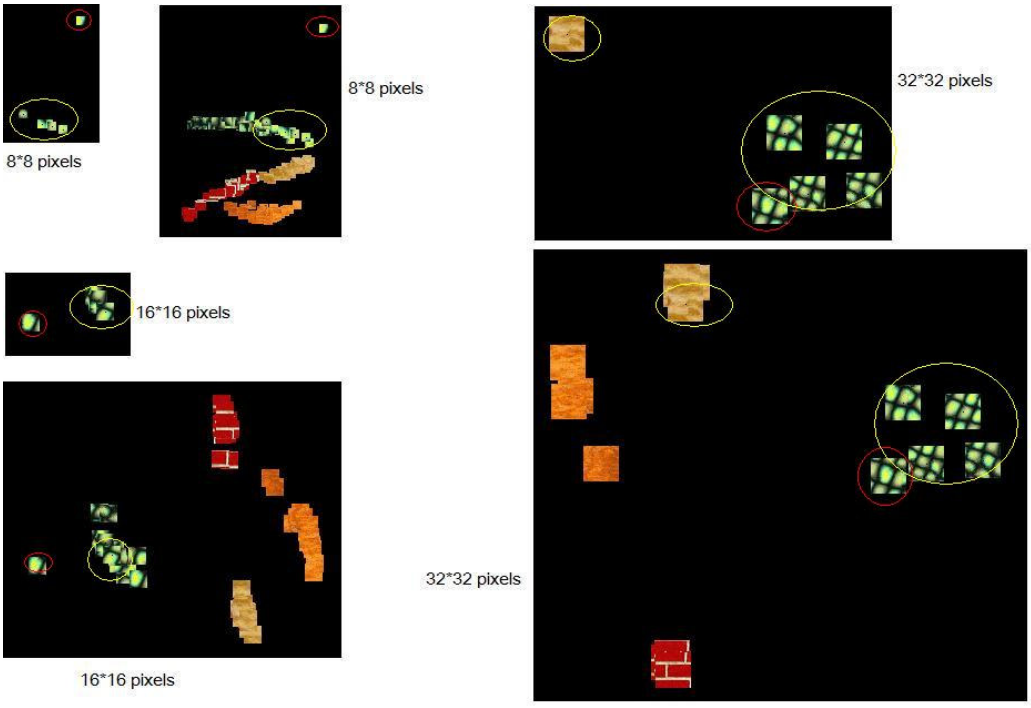

Figure A.4.6 La projection du patch d'entrée au centre (40,40) et ses 5 plus proches voisins.

Figure A.4.7 La projection du patch d'entrée au centre (110,40) et ses 5 plus proches voisins.

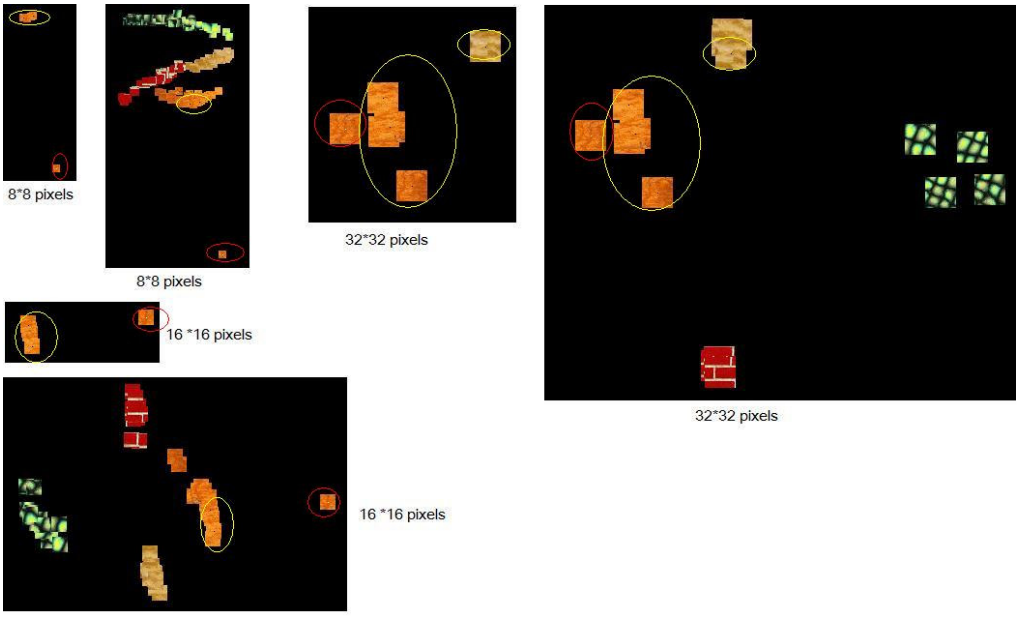

Figure A.4.8 La projection du patch d'entrée au centre (40,110) et ses 5 plus proches voisins.

A.5 Les résultats entre les différents tailles de patch :

Figure A.5.1 Avec la taille de patch de 4*4 pixels.

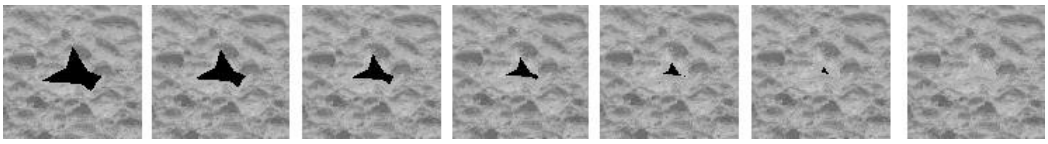

Figure A.5.2 Avec la taille de patch de 8*8 pixels.

Figure A.5.3 Avec la taille de patch de 16*16 pixels.