RDFIA - TME1-2-3-4

Calcul de descripteurs locaux et de dictionnaires visuels

Iyed Trimech – Yanis Chemli

Travaux encadrés par M. Cadene
Université Pierre et Marie Curie (UPMC)

Master IMA
26 septembre 2016

Table des matières

Table des figures

Introduction

L'objet de ce travail sur machine encadré est de mettre en place un système de classification d'image par une méthode d'apprentissage. On procédera alors en trois étapes : un descripteur sera étudié, mis en place puis testé sur une banque d'images, puis, sera mis en place un algorithme de clustering. Le but final est alors de créer un dictionnaire visuel. On utilisera finalement l'algorithme de classification supervisé SVM (Support Vector Machine) qui nous permettra de classer nos images.

Nous avons à disposition 4485 images que nous devrons classer en 15 catégories.

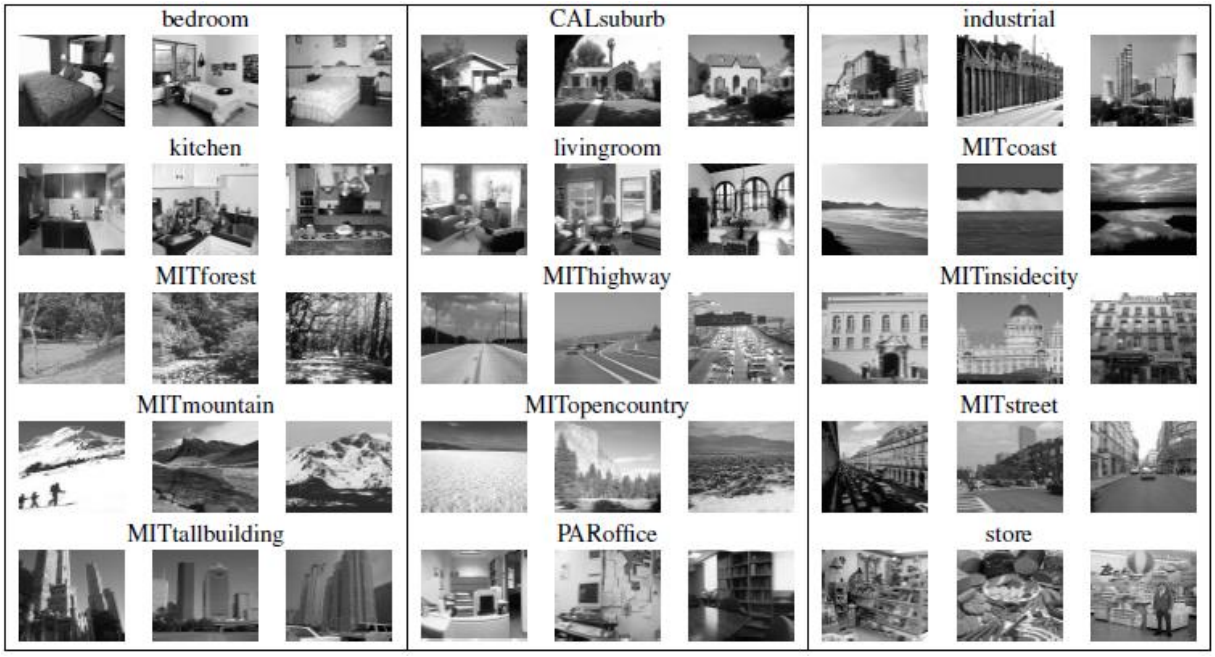


Figure 1 Nos 15 catégories

Extraction de descripteurs locaux : SIFT

Présentation générale du descripteur SIFT

SIFT, pour Scale Invarient Feature Transform, est un descripteur local d'image basé sur les histogrammes orientés de gradients. Très utilisé, il a été développé en 2004 à UBC par David Lowe. C'est un descripteur local dans la mesure où il est capable de donnée des informations de gradient (module et orientations) sur des zones précises ou patchs d'une image. Il est invariant en rotation, et aux changements d'illumination.

En pratique, le descripteur SIFT est calculé de façon automatique en des points d'intérêts d'une image. Pour ce travail, nous allons cependant calculer de façon dense les descripteurs sur une grille prédéfinie.

Figure 2 SIFTs calculés en des points d'intérêts (https://robwhess.github.io/opensift/)

Calcule d'un descripteur SIFT sur une image de test

On considère une image dans laquelle on va extraire des descripteurs locaux, c'est à dire qu'on calcule une caractéristique visuelle dans un ensemble de sous-régions. On considèrera ici le cas simple où les régions sont des patchs carrés de taille 16x16 pixels, échantillonnées avec un pas constant de 8 pixels.

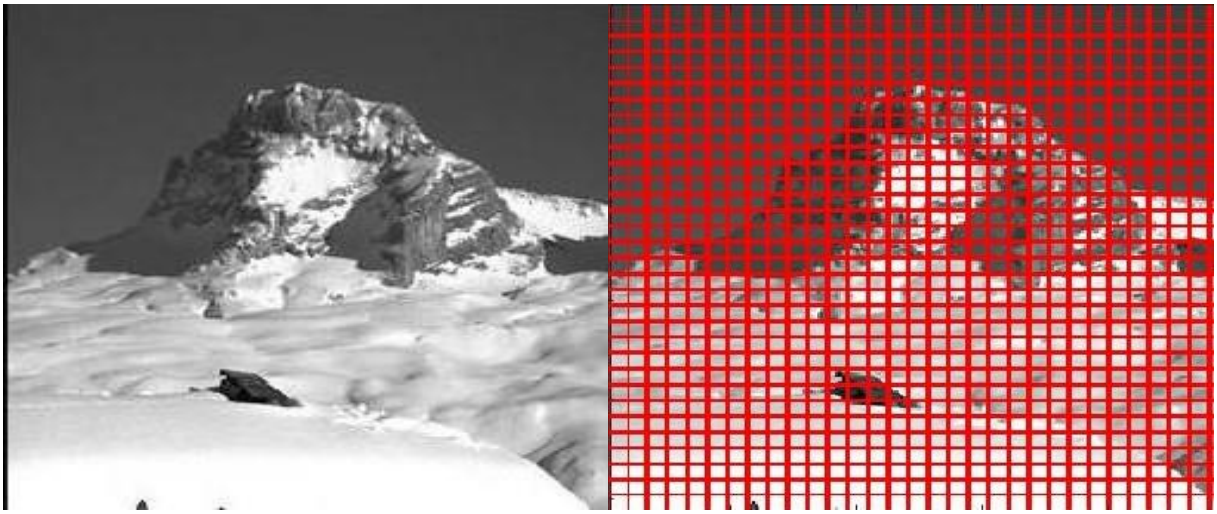


Figure 3 Image de test échantillonnée

C’est sur chacune de ces régions que nous allons extraire un descripteur SIFT.

Le descripteur SIFT, généralités et construction

SIFT se base sur le calcul des orientations est associées à chaque points en calculant les directions dominantes des vecteurs gradients dans les voisinages des points. On construit alors un vecteur à partir de l’accumulation successives de ces directions.

Concrètement, pour un patch P de taille 16x16, le vecteur SIFT est construit en divisant P en 16 sous matrices de tailles 4x4. Dans chacune d’elle on forme un histogramme d’orientation de gradients, en discrétisant l’orientation en 8 bins (ou directions). Pour cela, on a calculé le module ainsi que l’amplitude du gradient associés à chacun des patchs 4x4

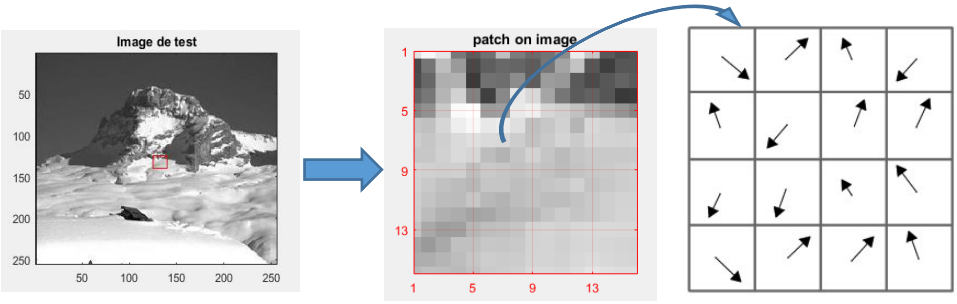


Figure 4 Extraction des sous matrice 4x4 et des orientations de gradients associées

On applique le procédé sur les 16 sous matrices pour finalement former un histogramme des orientations de gradients : Lors du passage au calcul de l’orientation sur une autre sous-matrice, on vient ajouter la nouvelle valeur à la valeur précédente, ce qui permet de former notre histogramme (vecteur de 128 bits).

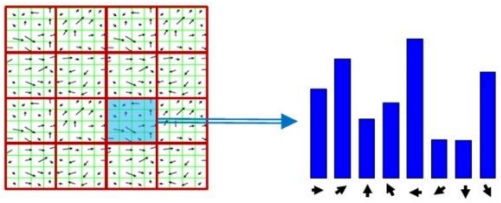


Figure 5 Création d'un histogramme pour un patch de 4x4 (https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/)

On concatène nos 16 histogrammes pour créer notre vecteur de 128 dimensions.

Figure 6 Le vecteur sift est formé des 16x8= 128 dimensions

Une étape importante préalable à la concaténation des vecteurs est l’application d’un masque gaussien afin de pondérer les gradients des pixels. En effets, la « force » des gradients les plus éloigné du centre du patch doit être atténué.

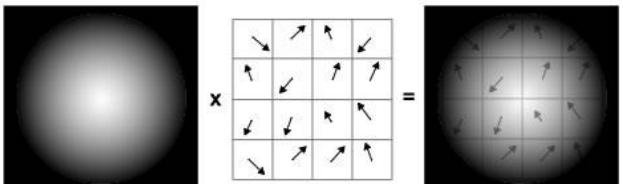


Figure 7 Application du masque gaussien (http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features/)

La dernière étape consiste à normaliser le vecteur sift. « Ceci a pour but de réduire l'influence des fortes valeurs de gradient afin d'être robuste aux variations non linéaires d'intensité, et peut être vu comme une normalisation du descripteur par rapport au contraste. ».

Application concrète sur l'exemple « marche »

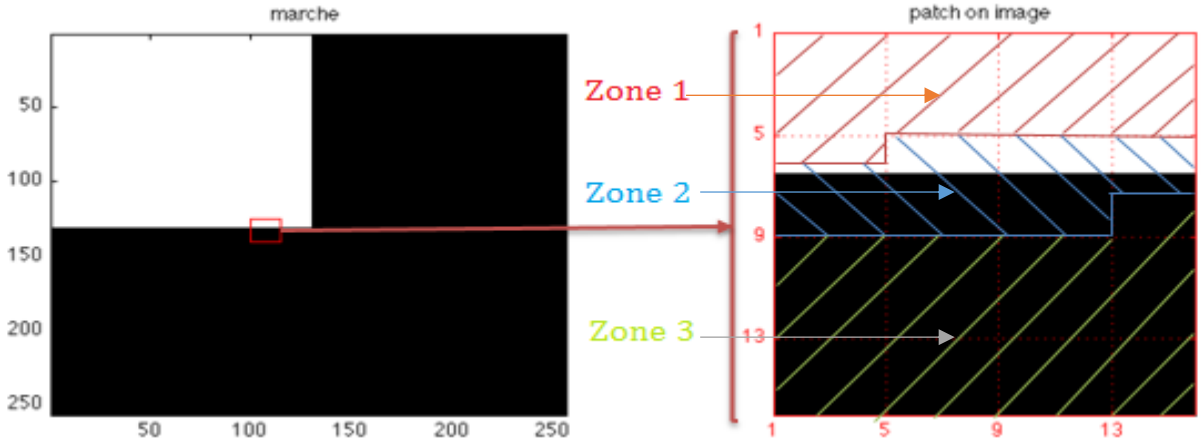


Figure 8 Découpage du patch et interprétation

Après avoir l'appliquer sur un patch de l'image, ce discripteur sera de la forme suivante :

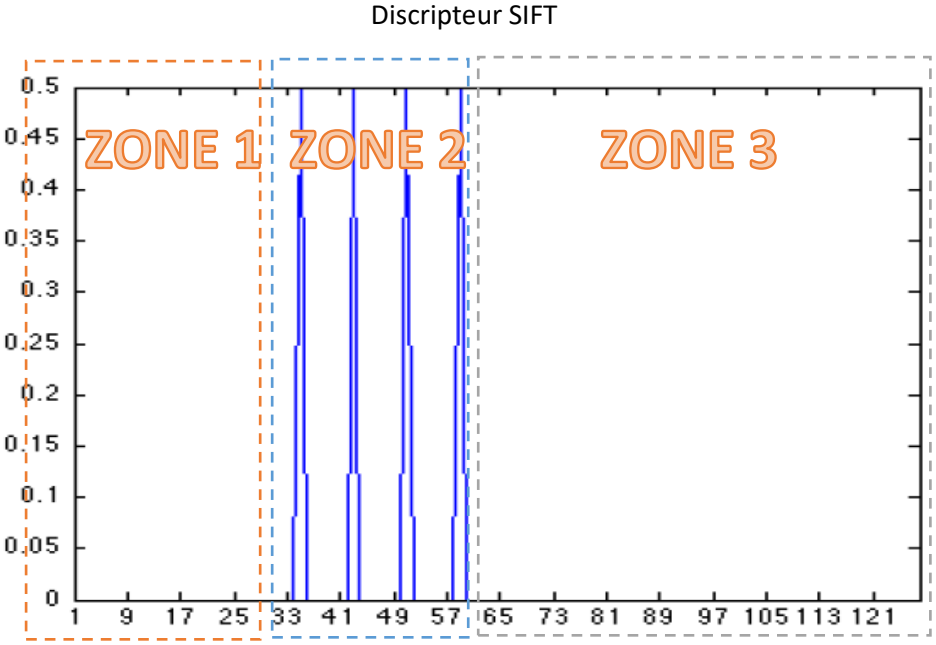


Figure 9 Le SIFT appliqué en un patch situé sur une zone de passage du blanc au noir

Interprétation :

On sais qu'un patch est de 16 blocs de (4x4) pixels, à l'intérieur de chacun de ces blocs on calcule l'orientation et la norme du gradient.

L'axe des abscisses répresente les 128 élements du vecteur SIFT.

L'axe des ordonnées répresente la norme de gradient pour chaque élement du vecteur SIFT.

- Pour les 4 premiers blocs du patch il n'y a que du blanc, on a pas de différence d'intensité du gradient donc, la norme du gradient est nulle pour toutes les orientations. Ce qui explique la forme plate de « SIFT discriptor » dans la Zone 1.
- Pour le 5^{ième} Bloc, les 4 premiers pixels sont aussi tous de couleur unie, donc, il appartiennent aussi à la Zone 1.
- A partir du 5^{ième} Pixel qui représente aussi le 5^{ième} élement de l'histogramme SIFT, on a un passe du noir au blanc, ce qui explique l'apparition de la norme de gradient pour la 7^{ième} orientation d'où l'apparition du premier pic dans la zone 2 au niveau de l'élement 39 du vecteur SIFT.
- Le reste des pixels du 5^{ième} élement sont tous noir, de norme de gradient nulle donc l'histogramme est plat jusqu'à l'apparition du 2ème pic au niveau de l'élement 45 du vecteur SIFT et vice versa jusqu'à la fin de la Zone 3.
- La zone 3 de l'histogramme est totalement plate, car il n'y encore une fois pas de changement d'intensité

On peut faire la même interprétation sur d'autres patches de l'image. En différents points on à :

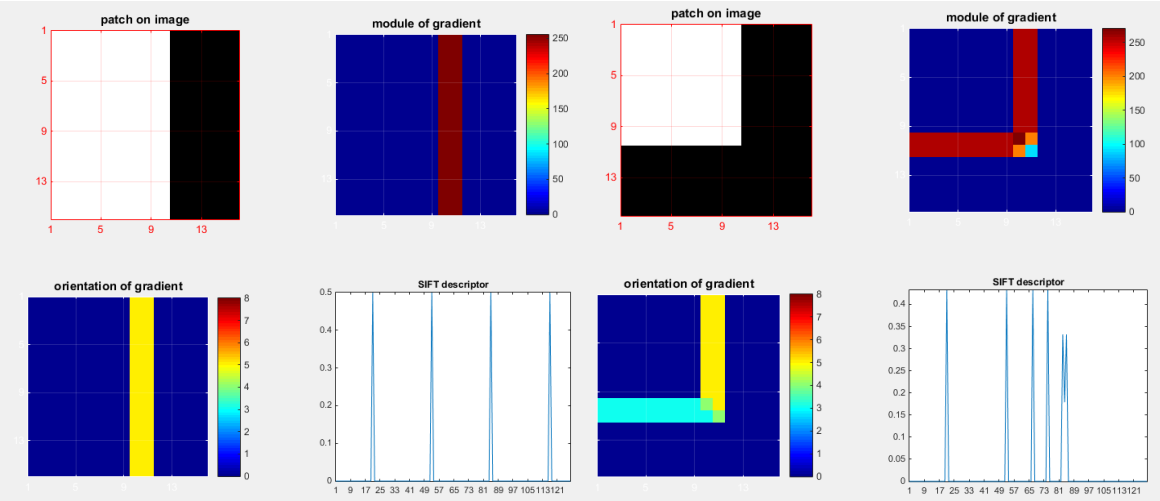


Figure 10 Test sur d'autres patchs

Calcule du gradient, opérateur de Sobel

On calcule le gradient de notre image en appliquant le masque de Sobel. Cet opérateur permet d'indiquer les directions des plus fortes variations du clair au sombre. Il est basé sur une approximation des dérivées partielles à l'ordre 1 selon les direction x et y de l'image. Ainsi, on obtient le vecteur gradient selon x et y, par la convolution de notre image par les matrices :

$$M_x = \frac{1}{4} \cdot \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \frac{1}{4} \cdot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

On a alors : $I_x = I*M_x$, $I_y = I*M_y$

On peut montrer que les masques M_x (resp.M_y) sont séparables. En effet on peut décomposer Mx (et Mx) en deux vecteurs hx et hy à une dimension. Pour hx = (-1, 0, 1) et hy = $\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$, on a bien

$\begin{pmatrix}1\\2\\1\end{pmatrix}$ x (-1, 0, 1) = $\begin{pmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{pmatrix}$ = Mx/4 et $\begin{pmatrix}-1\\0\\1\end{pmatrix}$ x (1, 2, 1) = $\begin{pmatrix}-1 & -2 & -1\\0 & 0 & 0\\1 & 2 & 1\end{pmatrix}$ = My/4

Cas des régions grossièrement homogènes

Dans le cas d'une région strictement homogène d'une image, le sift calculer en un point de cette région sera « plat ». En effet, dans ce type de région il n'y a aucune variation d'intensité et donc aucun gradient. Son module est donc nul et il n'a pas d'orientation. On peut vérifier cela en appliquant notre méthode sur une zone strictement homogène de l'image « marche ».

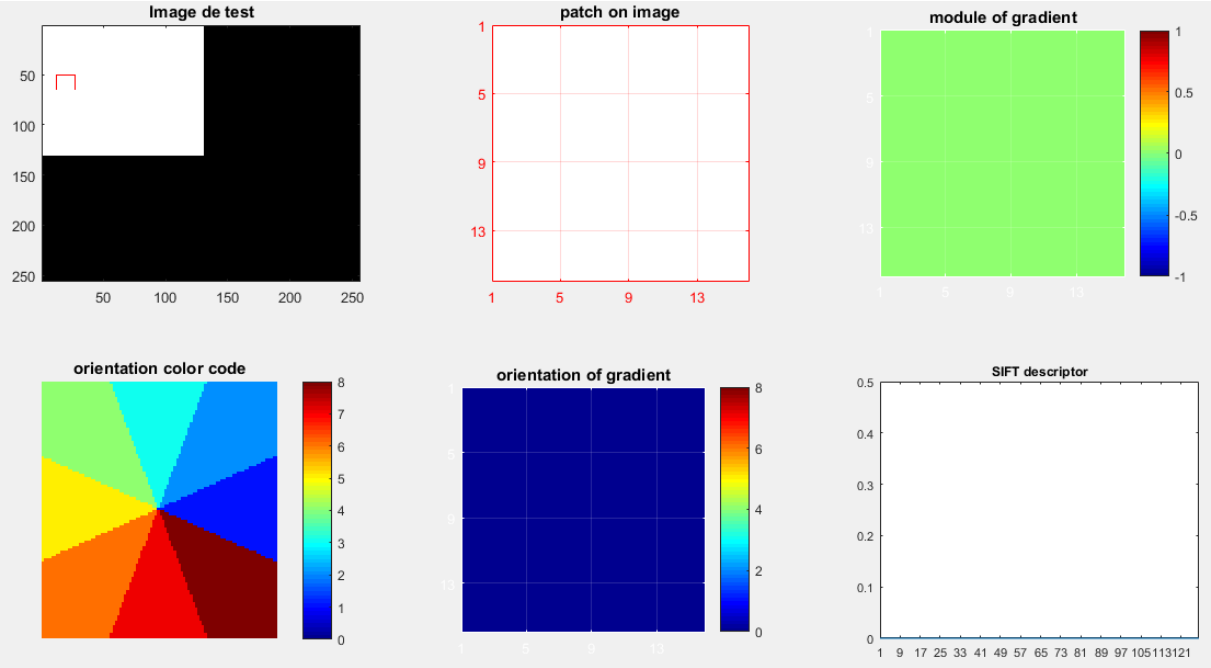
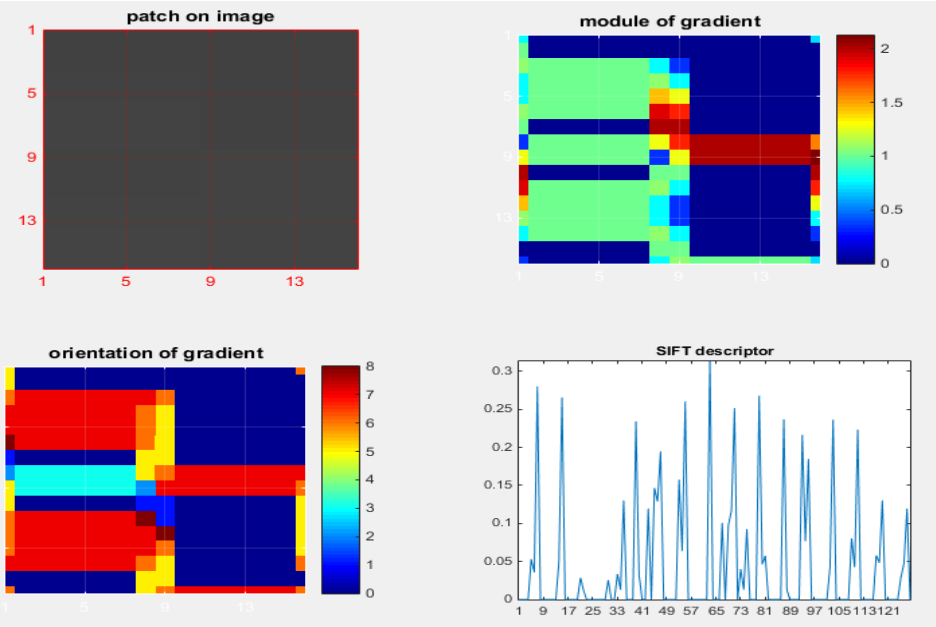


Figure 11 Cas d'une région strictement homogène

Dans le cas des zones grossièrement homogènes, on a de nombreux changement d'orientation ce qui implique de nombreux piques dans notre sift. Cependant, le module du gradient est proche de zéro car on a des faibles variations d'intensité dans ces zones.



Pour visualisé les résultats on dé-sature le seuil du sift pour voir que le patch porte tout de même du gradient mais à de faibles valeurs.

Cela est problématique dans la mesure où on l'on va accorder de l'importance a des vecteurs qui ne vont pas apporter d'information utile quant à la description de notre image. On donne à ces vecteurs sifts ont le même poids d'informations que les sifts portant de l'information utile. Or on a vu qu'il y a une étape finale lors de la construction de notre vecteur qui est sa normalisation (pour pallier aux problèmes de changement d'illumination). On a donc des vecteurs portant très peu d'information qui vont contribuer avec le même poids à décrire notre image.

On va donc modifier notre fonction computeSIFT afin de ne calculer le descripteur que pour les patchs où le contraste est ”suffisant”. On choisira comme critère la norme du descripteur SIFT : si elle est inférieure `a un seuil donné (avant normalisation), le descripteur SIFT est fixé au vecteur nul.

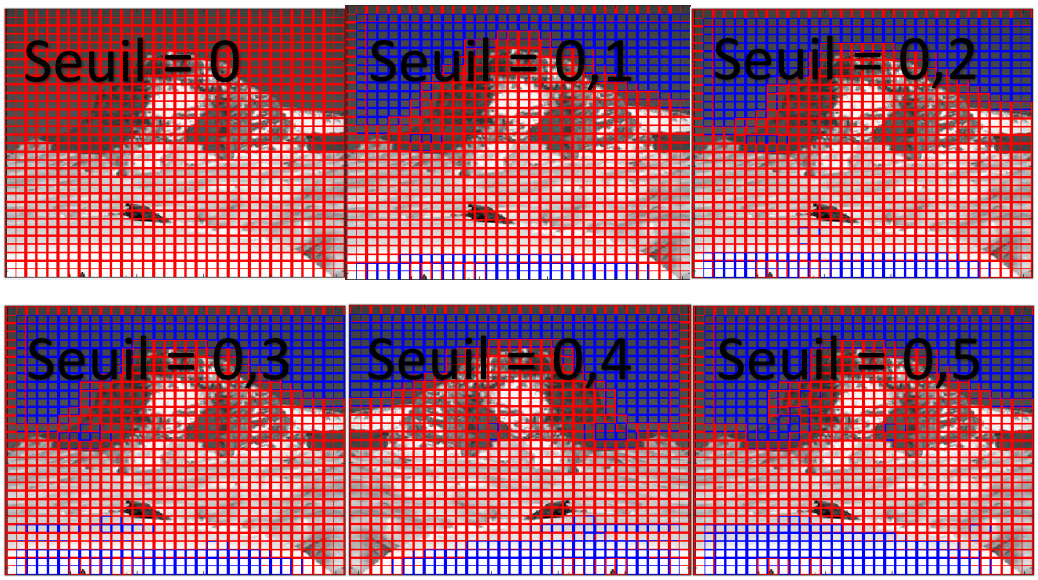


Figure 12 Effet de la modification du seuil de rejet de patchs homogènes

Ainsi on a mis à 0 les sifts dont le patch est situé sur une zone grossièrement homogène (une zone qui n'apporte donc pas d'information utile). Après tests sur plusieurs images, un seuil excluant les valeurs de la norme du gradient inférieur à **0,4** semble correct.

On peut donc à ce stade calculer l'ensemble des descripteurs sifts sur notre base d'image pour la génération du dictionnaire visuel.

La différence la plus notable avec la méthode de Lowe est le fait que notre méthode n'apporte pas d'invariances au changement d'échelles. Or la méthode de Lowe fait intervenir un calcule des descripteurs sift de façon multi échelle. En effet Lowe, utilise une approche pyramidale pour calculer les sifts de points d'intérêts et afin de d'être robuste au changement d'échelles.

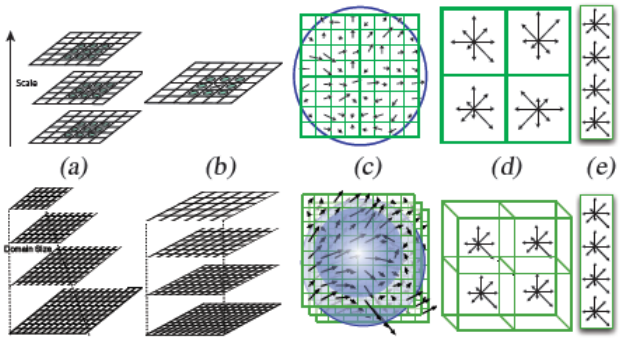


Figure 13 Approche multi-échelle : http://vision.ucla.edu/~jingming/proj/dsp/

Génération du dictionnaire visuel

Quantification et K-Means

A partir du vecteur SIFT (dimension d = 128) extrait en chaque patch de l'image, on veut calculer un dictionnaire visuel. L'objectif est de pouvoir déterminer un ensemble de K centres de l'espace minimisant la distorsion des données.

On définit alors N points décrits par leur coordonnées $x_i \in \mathbb{R}^{128}$ dans l'espace des SIFTs. On a :

$\{x_i\}$; i∈ $\{1, N\}$

De même, on définit M clusters de l'espace des SIFTs et leurs centres associés $c_m \in \mathbb{R}^{128}$. On a :

$\{c_m\}$; m ∈ $\{1, M\}$

On veut minimiser une certaine fonction de cout. Cependant cela revient à résoudre un problème d'optimisation difficile, c'est-à-dire un problème dont la solution optimale est difficilement (ou pas du tout) calculable directement. On a alors recours à une heuristique.

L'algorithme K-Means, qui permet dans beaucoup de cas d'obtenir une solution satisfaisante est utilisé.

Il a pour paramètres d'entrés $\{x_i\}$ et K (ou M) le nombre de cluster par lequel on souhaite diviser l'espace des sifts.

Il retourne $\{\widehat{c_m}\} \in \mathbb{R}^{128}$ les coordonnées des K (ou M) centres optimaux dans $\mathbb{R}^{128}$.

C'est un algorithme (non supervisé) itératif basé sur deux phases principales. La première phase d'assignement retourne l'indice des points les plus proches des centres des clusters placés aléatoirement sur notre plan. La deuxième phase est la phase de mise à jour des centres. Dans cette phase, on calcule la distance de chaque point x_i par rapport au centre du cluster qui leur est associé. On déplace alors le centre d'une distance égale à la valeur moyenne du calcul précédent. Lorsque l'on converge, c'est-à-dire lorsque les centres c_m ne se déplacent plus, on sort de la boucle.

Cette heuristique ne garantit pas d'aboutir à une solution optimale mais on montre que dans le cas général, le barycentre des points appartenant au cluster minimise la distorsion des données :

$$\widehat{c_m} = \mathbb{E}(c_m) = \frac{1}{|c_m|} \sum_{x_i \in c_m} x_i = \underset{\{c_m\} \in \mathbf{R}^d}{\arg\min} \sum_{x_i \in c_m} ||x_i - c_m||_2^2$$

En effet, on a pour dim(x) = n

$$\widehat{Cm} = argmin \sum_{xi \in Cm} \left[\sum_{j=1}^{n} ||x_{ij} - Cm_j||^2 \right]$$

$$<=> \widehat{Cm} = argmin \sum_{j=1}^{n} \left[\sum_{xi \in Cm} (x_{ij} - Cm_j)^2 \right]$$

Les n équations sont indépendantes, on peut donc écrire :

$$\widehat{Cm} = \sum_{j=1}^{n} \left[argmin \sum_{xi \in Cm} (x_{ij} - Cm_j)^2 \right]$$

On cherche à minimiser chacune des équations convexes. Il existe un minimum global que l'on détermine en annulant la dérivée selon Cm :

$$\frac{d}{dCmj} \sum_{xi \in Cm} (x_{ij} - Cm_j)^2 = 2 \sum_{xi \in Cm} x_{ij} - Cm_j = 0$$

$$<=> \sum_{xi \in Cm} xij = \sum_{xi \in Cm} Cm_j$$

Ainsi :

$$Cmj = 1/card(Cm) \sum_{xi \in Cm} x_{ij}$$

Ce qui correspond bien à un barycentre.

Application sur notre base de SIFTs

On peut utiliser l'algorithme sur la base des sifts précédemment extraits de nos 15 scènes.

1 - Phase d'assignement

Pour chaque point extrait de la base, il faut déterminer le cluster le plus proche. On utilise pour ça une méthode d'assignement dont voici le code sans boucle :

```
function [ nc ] = assignementKMeans(sifts, clusters, matNormClusters)
distances = matNormClusters + repmat(sum(sifts.^2),size(clusters,1),1)
- 2 * clusters * double(sifts);
[~, nc] = min(distances, [], 1);
end
```

2 - Phase de mise à jour

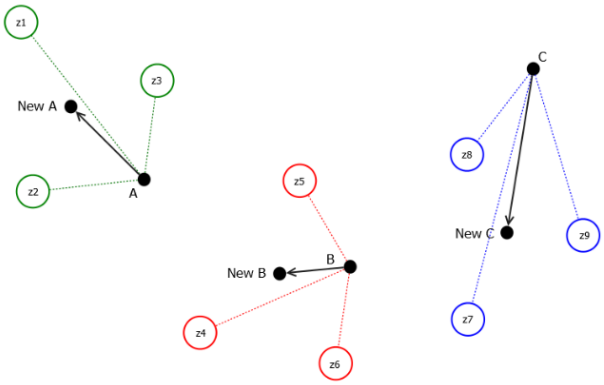


Figure 14 http://www.turingfinance.com/clustering-countries-real-gdp-growth-part2/

Dans cette phase nous mettons à jours le centre des clusters selon la moyenne des points xi.

Une fois convergence, on a formé nos 1000 mots visuels.

Calcul de représentation image : le modèle Bag of Words (BoW)

Le dictionnaire visuel est construit en deux étape par une étape de coding et une étape de pooling.

La première étape consiste à affecté la valeur 1 au plus proche voisin pour un x donné. Pour cela on compare les distances du point à chacun des centres.

La seconde étape d'agrégation consiste à construire un histogramme d'occurrence des différents mots visuels.

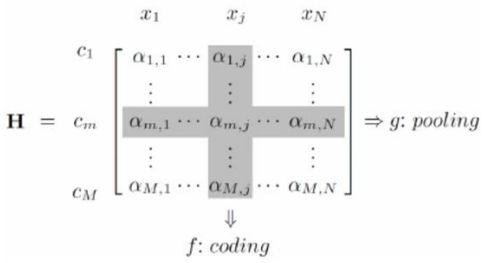


Figure 15 Coding & Pooling

Formellement on définit les deux étapes comme suit :

1 - Effectuer un codage au plus proche voisin : f = f_Q

$$f_Q(x_j) = \begin{cases} 1 & \text{if } m = \underset{k \in \{1;M\}}{\operatorname{argmin}} \|x_j - c_k\|^2 \\ 0 & \text{otherwise} \end{cases}$$

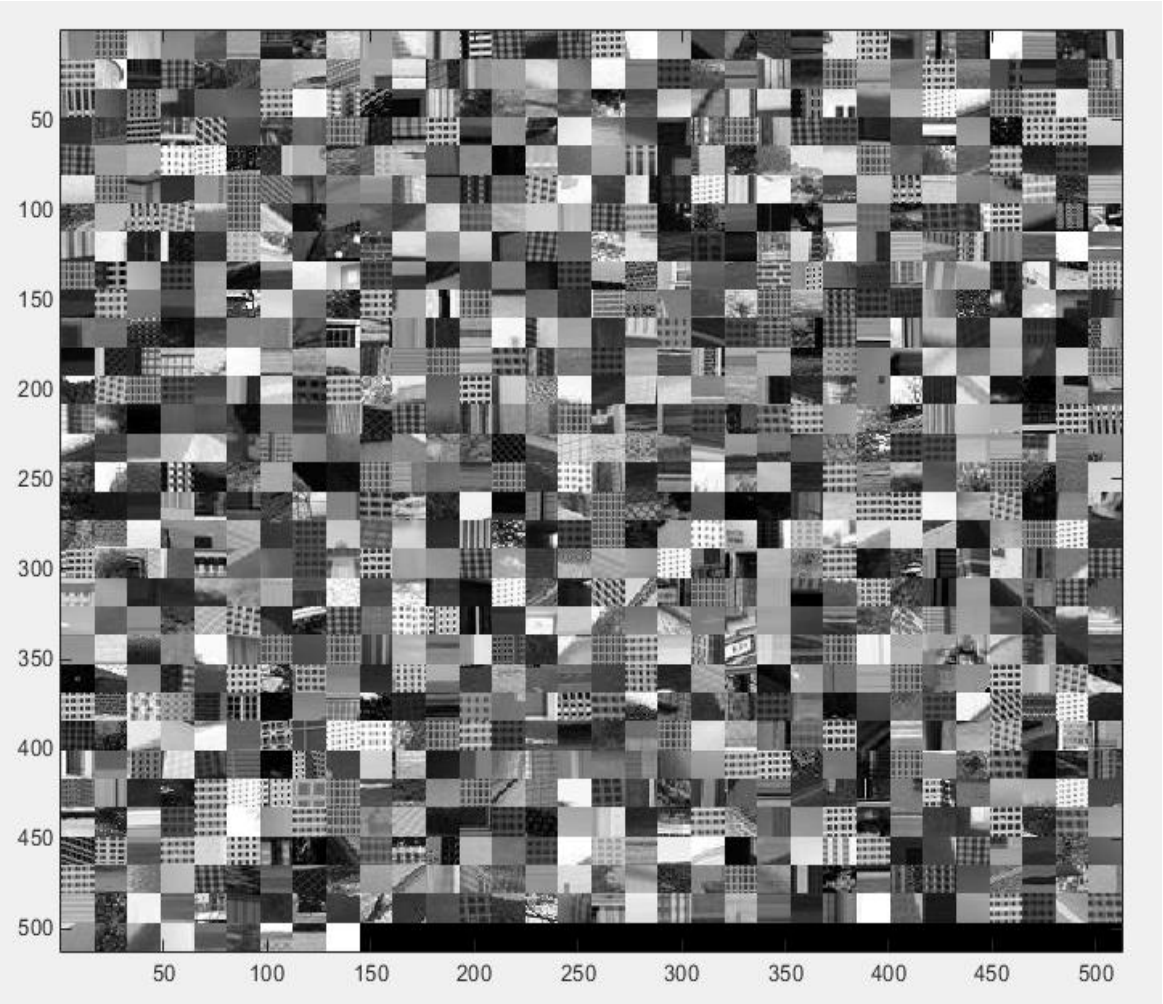
2 – Création d'un histogramme des occurrences des mots :

$$z_m = \sum_{j=1}^{N} \alpha_{m,j}$$

Visualisation du dictionnaire visuel

A l'aide de l'outil Matlab est de la fonction visuDico, on a pu visualisé un dictionnaire appris sur notre base de 15 scène. On obtient la figure suivante :

Le résultat retourné dépend de la clustérisassions par K-means qui n'est pas fixé et qui dépend de conditions initiales.



Le résultat retourné dépend de la clustérisassions par K-means qui n'est pas fixé et qui dépend de conditions initiales.

La fonction nous retourne ainsi la visualisation des patches les plus proches de chaque mot visuel de la base. Ainsi on a 1001 patchs car 1001 mots visuels.

Visualisation des BoW

On va maintenant utiliser la fonction `visuBoW` sur des images aléatoires. Cette fonction retourne le sac de mot de l'image i.e. l'histogramme des occurrences des mots, la visualisation de la localisation spatiale des 8 mots du dictionnaire les plus fréquents dans l'image ainsi que l'apparence des 8 mots du dictionnaire les plus fréquents dans l'image.

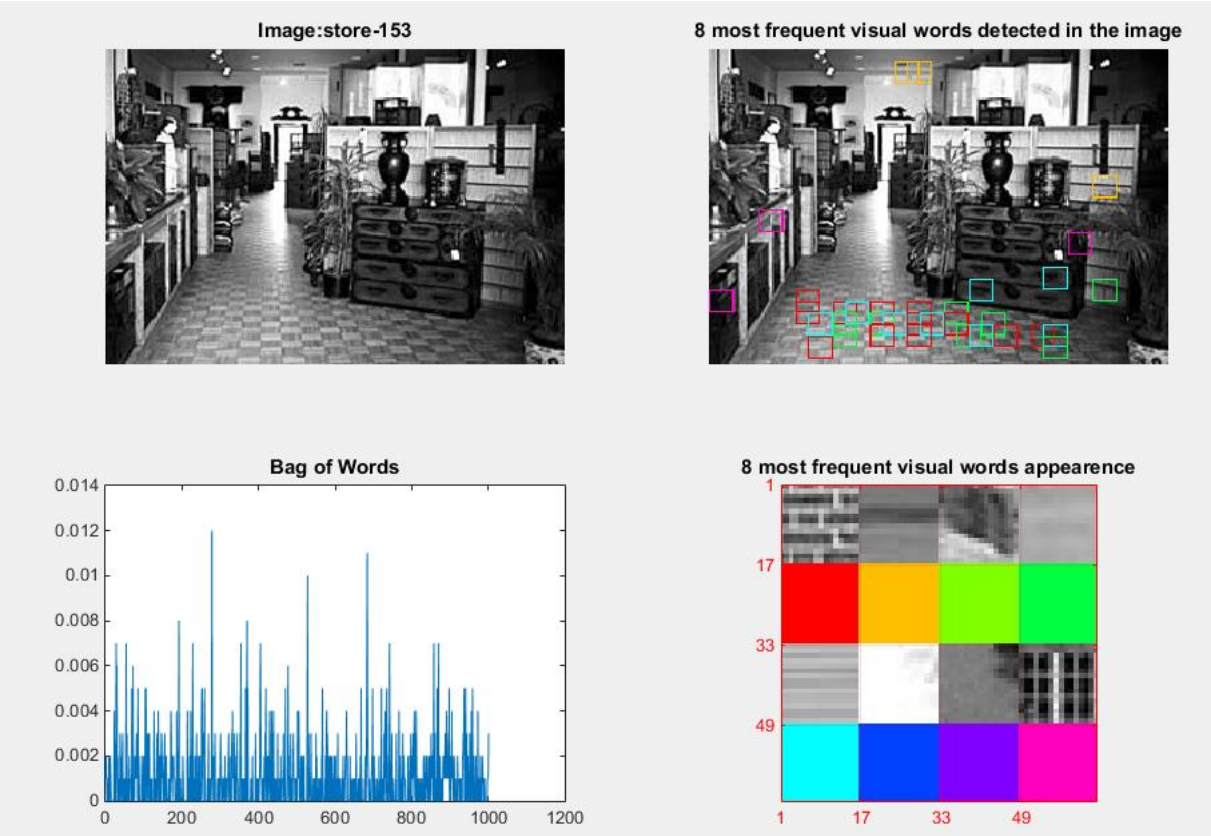


Figure 16 Exemple de BoW pour une image

On repère ainsi certain motif du dictionnaire visuel. On observe sur différentes images que les mots les plus fréquents correspondent à des motifs particuliers tel que des motifs répétés (fenêtres), des jonctions, des zones homogènes, des lignes parallèles etc.

Afin de procéder à l'apprentissage d'un algorithme de classification, nous avons calculé l'ensemble des sacs de mots de notre base d'image.

Apprentissage supervisé pour la classification sémantique d'images

A partir des représentions BoW calculées pour chaque image de la base, on veut mettre en place un apprentissage supervisé dans l'espace des BoW. L'objectif consiste ensuite à proposer un système capable de prédire la classe parmi les 15 possibles pour une image quelconque de la base qui n'aura pas été apprise.

Apprentissage par SVM et performances

La méthode SVM est initialement un classifieur binaire (il sépare 2 classes). On utilisera la méthode un contre tous pour faire notre classification multi classe.

Cette méthode propose de séparer linéairement les donnée (on considère ici un noyau linéaire pour des données linéairement séparables) par un hyperplan. On a affaire ici à un problème d'optimisation où l'on doit déterminer l'équation d'un hyperplan qui sépare les données avec la plus grande marge.

Ainsi la méthode maximise la distance des points les plus proches de la frontière (les vecteurs de support).

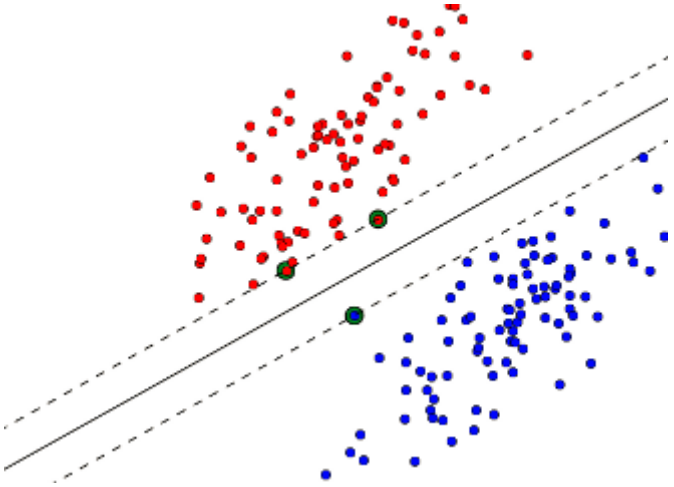


Figure 17 On cherche la plus vaste marge de séparation

Formellement, on cherche à minimiser une fonction g avec :

$$g(w,b) = \frac{1}{2}||w||^2 + C \cdot \sum_{i=1}^{N} max[0, 1 - yi \cdot f(z_i)] = \frac{1}{2}||w||^2 + C \cdot \sum_{i=1}^{N} L[f(z_i), yi]$$

On a donc mis en place la méthode eval 15Scene qui a permis l'apprentissage et le test du système sur la base 15-Scene, à partir des signatures vectorielles de type BoW précédemment extraites pour chaque image de la base. On calcule dans ce script la matrice de confusion qui permet d'évalué les performances de la classification.

Pour notre teste, on a utilisé 100 images d'entrainement par catégories. Le reste des images font office d'images de tests.

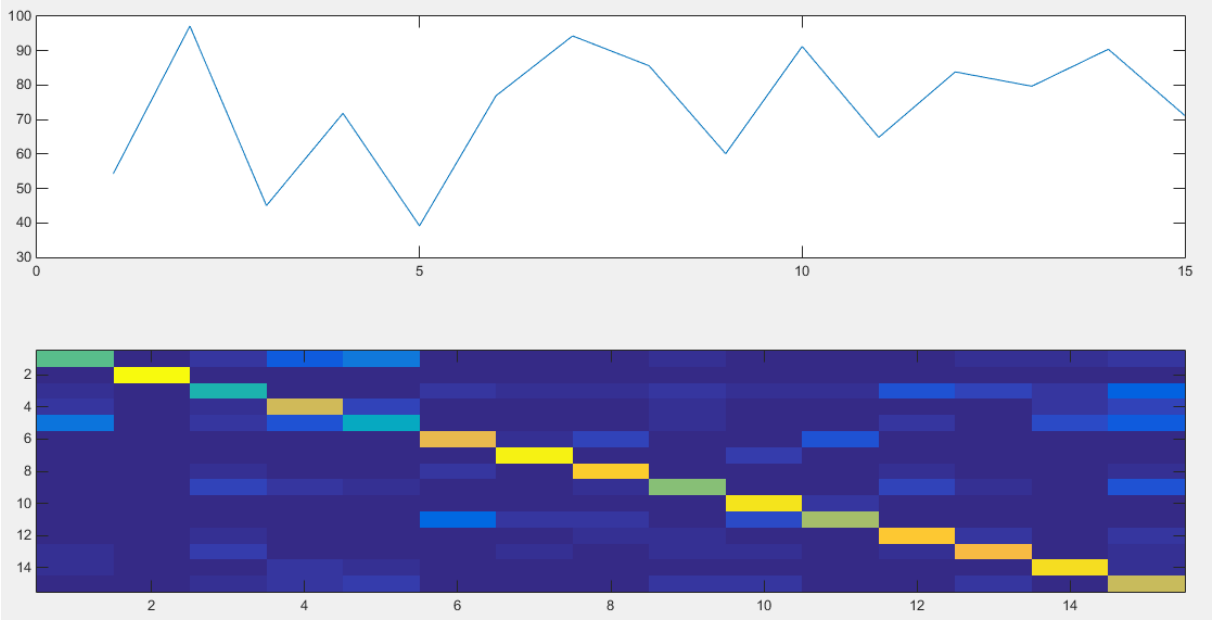


Figure 18 Matrice de confusion

Chaque colonne de la matrice représente le nombre d'occurrences d'une classe estimée, tandis que chaque ligne représente le nombre d'occurrences d'une classe réelle.

Carte de chaleur de classification

Dans cette dernière étape, il nous a été proposé d'analyser l'impact de chaque descripteur local dans le processus de décision de l'image. L'objectif est de mettre en place une fonction permettant de calculer une "carte de chaleur de classification".

Conclusion

Nous sommes donc arrivés par différentes étapes à classer des images par une méthode d'apprentissage supervisée. Nous avons commencé par extraire de façon dense des caractéristiques des images par la méthode SIFT. Par l'algorithme de clustering K-means, nous sommes parvenus à construire un dictionnaire de mots visuels. Par les techniques de codage et d'agrégation nous avons pour chaque image calculé un sac de mots. Enfin nous avons utilisé la méthode d'apprentissage linéaire SVM pour classé des images de tests.

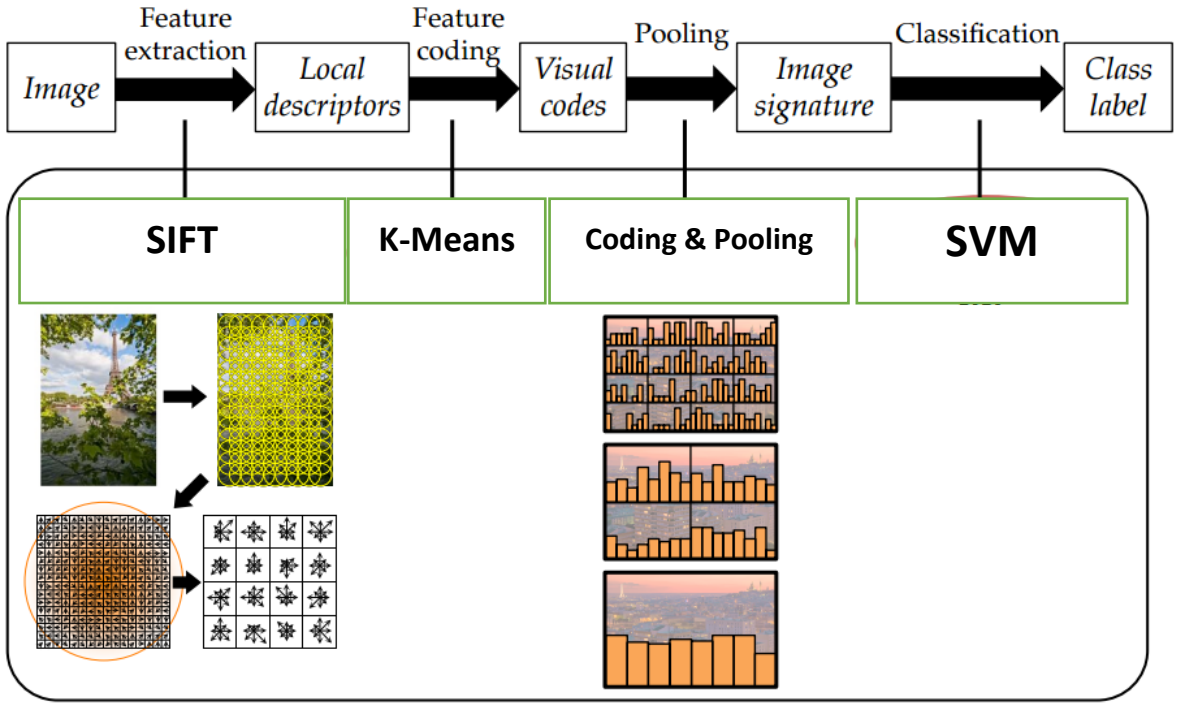


Figure 19 Pipeline pour la classification