TP 1

Jouve Vincent 3671083

On cherche un moyen efficace pour représenter une image, c'est-à-dire extraire de l'information de l'image et en même temps passer de dimension variable, la taille de l'image, à vecteur de taille fixe et pas trop grande. Pour cela dans ce TP on se penche sur la technique du BoW pour représenter une image. Pour extraire l'information de l'image on va d'abord découper l'image en patchs qui représentent des régions de l'image. Dans ce TP on découpe selon une grille, la méthode à pour avantage d'être simple mais ne fourni pas de garanti sur qualité de l'information présente dans le patch. En cours on a vu une méthode qui se base sur le calcul de points d'intérêts, pour lesquels on découpe les patchs autour de ces points, ce qui à pour effet d'avoir une meilleure garantie sur la qualité de l'information porter par les patchs. Maintenant il faut extraire de l'information depuis ces patchs pour cela on utilise le descripteur SIFT, qui utilise les variations de couleurs/luminosité grâce au gradient de l'image, pour décrire le patch. Ce qui est intéressant dans les variations c'est l'orientation de la variation et l'intensité. Donc SIFT permet d'obtenir un vecteur qui représente le patch selon l'intensité et l'orientation des variations dans le patch. Comment agréger les vecteurs SIFT obtenu pour représenter l'image complète ? Pour cela, étant donné que l'espace des SIFT est infiniment grand on va le discrétiser, mais pas n'importe comment, on va faire cela à l'aide de l'algorithme des k-moyennes. L'inconvénient c'est que l'on a besoin d'un ensemble d'apprentissage et que la représentation finale d'une image va dépendre de l'ensemble d'apprentissage. Il faut donc bien choisir cet ensemble d'apprentissage selon les images qu'on veut décrire. Grâce à l'algo des k-moyennes on obtient k SIFT-type qui seront des représentant de chaque cluster. Ainsi pour chaque patch on peut trouver le SIFT-type le plus proche géométriquement. Et c'est comme ça que l'on va pouvoir agréger l'information contenu dans les patchs, pour décrire toute l'image. On peut, par exemple, compter pour chaque SIFT-type le nombre de fois que l'on trouve un SIFT où ce SIFT-type est le plus proche. Ce résultat final est le descripteur de l'image représenter sous la forme d'un BoW.

Partie 1

Question 1

M_x et M_y sont séparable, si on prend en compte le facteur $\frac{1}{4}$ avec $h_y = \begin{pmatrix} 0.5 \\ 1 \\ 0.5 \end{pmatrix}$ et $h_x = \begin{pmatrix} -0.5 \\ 0 \\ 0.5 \end{pmatrix}$ on a que $M_x = h_y \times h_x^\top$ et $M_y = h_x \times h_y^\top$

Question 2

On gagne en complexité et temps de calcul. Au lieu d'appliquer la matrice de convolution directement on applique deux filtres, un vertical et un horizontal, l'un à la suite de l'autre.

Question 3

En appliquant un masque gaussien sur les normes des gradients on fait en sorte que ce qui se passe au centre du patch est plus important que ce qui se passe sur les côtés. Au final le SIFT exprimera plus fortement ce qu'il se passe au centre du patch, bien sûr s'il ne ci passe rien alors ce qu'il se passe sur les côtés prendra le dessus.

Question 4

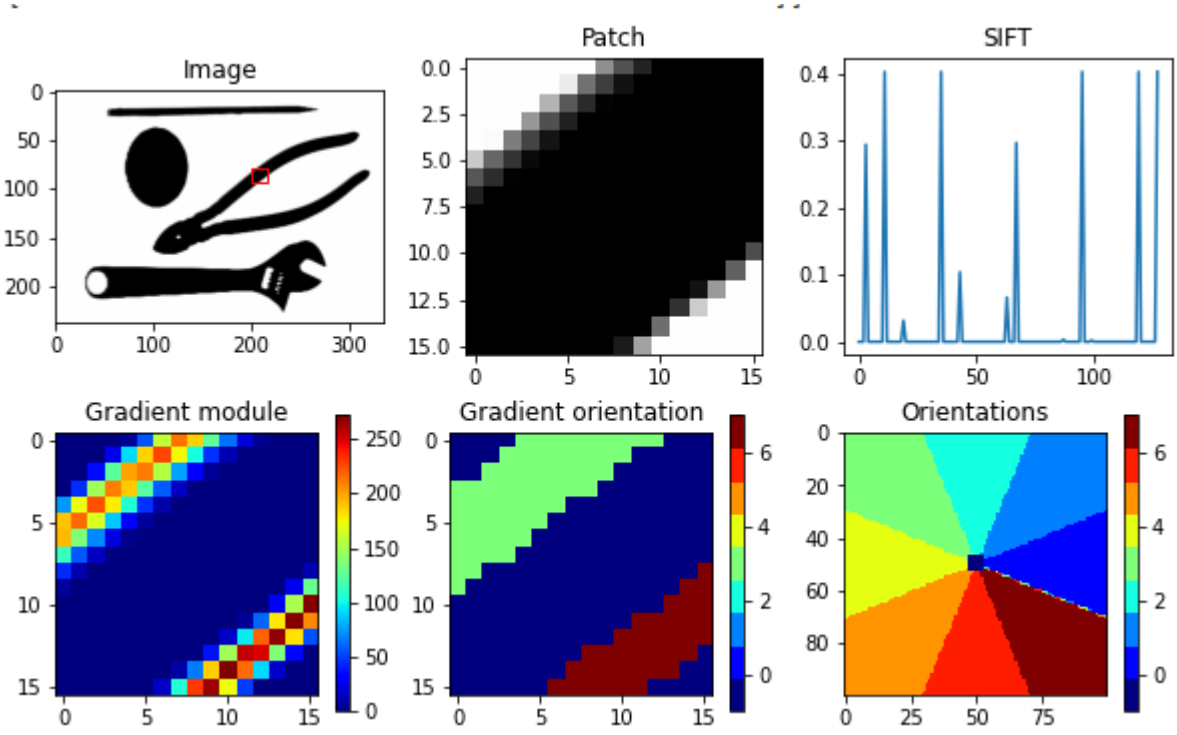
Le SIFT est une façon de représenter un patch, il donne de l'information. La discrétisation permet par la suite de compresser l'information d'un patch avec l'histogramme, on passe de deux matrices de 16*16 à un vecteur de taille 128. Un même SIFT peut être calculé grâce à deux patchs proches mais pas identique. On a donc compressé et généraliser l'information sans trop en perdre.

Question 5

Pour la 1ère étape on regarde si la norme est en-dessous d'un seuil, si oui on fixe le descripteur au vecteur nul. Ce sont des descripteurs portant quasiment aucune information et probablement que du bruit, c'est pour cela qu'on le fixe au descripteur nul.

Pour la seconde étape on normalise le descripteur, on aura donc le même ordre de grandeur pour toutes les composantes de tous les descripteurs. Cela permet d'être robuste à la luminosité de l'image, et de ne pas éloigné deux patch qui présenterai la même forme mais l'un moins lumineux que l'autre.
Pour la dernière étape les valeurs des composantes sont seuillées, car on ne veut pas qu'une direction l'importe et soit la seule représentante du patch, auquel cas on pourrait avoir du mal à différencier deux SIFT juste en se basant sur le fait qu'ils aient la même composante principale, ils seraient trop proche géométriquement parlant, et le reste ne compterai pas assez pour les différencier.

Question 7



Les orientations des gradients vont bien du noir vers le blanc. De même les normes des gradients sont d'autant plus grandes qu'elles sont sur une frontière noir blanc. Et le SIFT présente le bon nombre de piques (J'ai vérifié manuellement pour ce patch).

Partie 2

Question 8

Le but étant de prendre en entrée une image et de sortir une ou des images qui y ressemble le plus possible. On fait ça en passant par les SIFT que l'on va regrouper à l'aide de représentant fourni par le dictionnaire. Donc le dictionnaire contient N représentant de SIFT, à l'aide de ce dictionnaire on peut rapprocher 2 SIFT qui ont le même représentant, ainsi si on peut rapprocher plusieurs SIFT (et donc patchs) de deux images, alors elles-mêmes sont proche. C'est comme ça qu'on peut sortir une image qui ressemble à l'image d'entrée.

Question 9

Pour trouver le minimum il suffit de trouver où la dérivé s'annule.

$$\begin{aligned} 2 * \sum_{i=1}^{n} x_i - c = 0 &\iff \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} c = 0 \\ &\iff \sum_{i=1}^{n} x_i - n * c = 0 \\ &\iff c = \frac{1}{n} \sum_{i=1}^{n} x_i \end{aligned}$$

Le c qui minimise l'équation est bien le barycentre.

Question 10

Soit on a une idée du nombre qu'il nous faut, imaginons qu'on veuille différencier 4 classes de patch, on aura alors besoin de 4 clusters. Sinon on test pour pleins de K et à l'aide d'indice comme l'indice de dunn et/ou xie-beni qui donne une indication sur la qualité des clusters, on choisit le meilleur nombre de cluster.

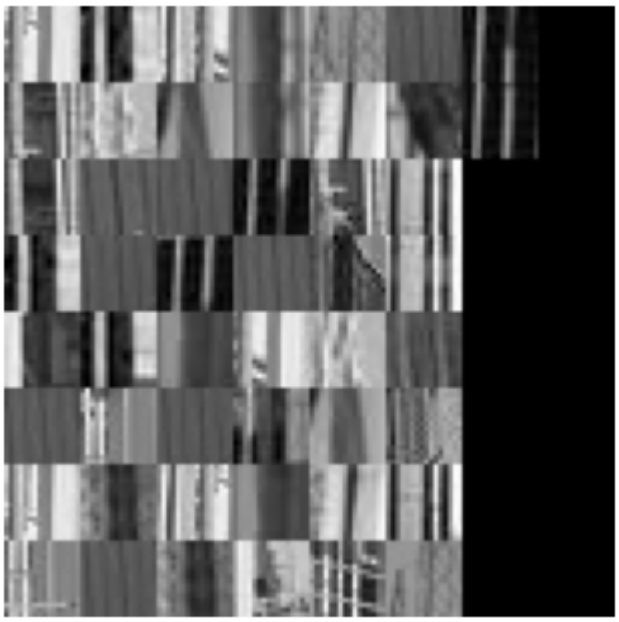
Question 11

Le représentant théorique que l'on obtient grâce à l'algorithme des k-means est un SIFT et on ne peut pas directement le convertir en une image, car

l'algorithme de calcul des SIFT n'établit pas une bijection entre SIFT et images. Il faut donc passer par l'intermédiaire des exemples pour pouvoir visualiser leur patch associer, et avoir une image de quelque chose d'interprétable pour l'homme.

Question 12

Parmi les clusters on peut trouver ce genre de chose, qui représente les 50 régions d'images les plus proche du représentant du cluster.



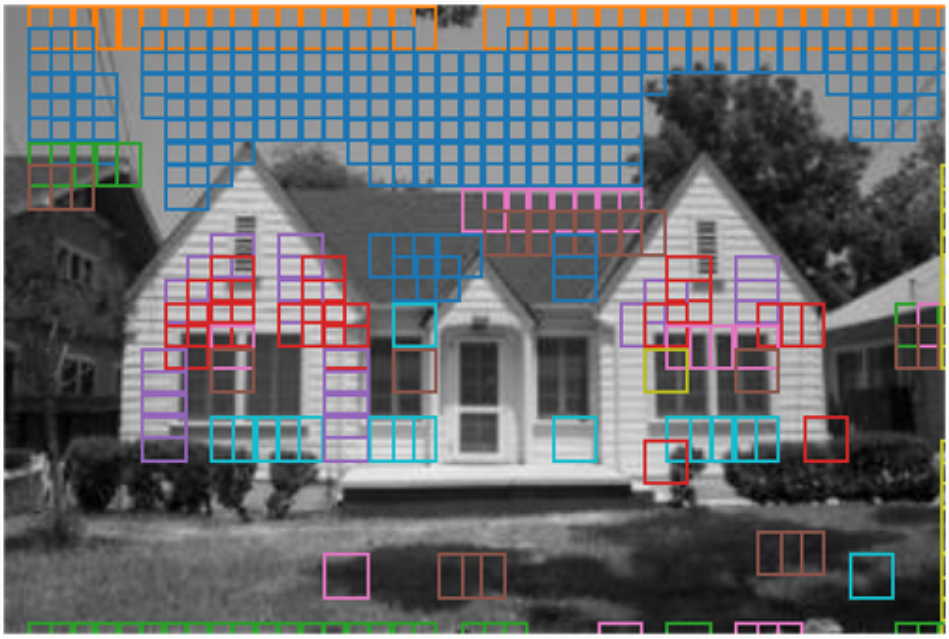
Ici on voit que pour ce cluster cela fonctionne plutôt bien, on a bien regroupé les régions qui présentent deux barres parallèles. Pour certaines régions on peut même y voir des barreaux.

Partie 3

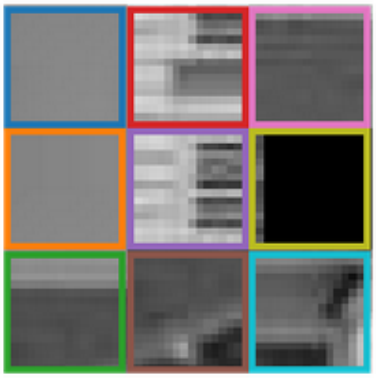
Question 13

Chaque composante i de z représente le nombre de fois que le SIFT-type i est le plus proche d'un SIFT correspondant à un patch de l'image.

Question 14



(9, 16, 16)



Ici on peut voir visuellement les 9 SIFT-type les plus présent dans l'image et on voit même où est-ce qu'ils sont présents. Le bleu est très présent dans le ciel et un peu sur le toit, c'est le SIFT-type qui représente aucune variation de luminosité, le vecteur nul. Pour le reste on voit que certain SIFT-type représentant le bord des fenêtres, ils sont assez nombreux. Ce qui est pas mal pour classifier ensuite cette image comme une maison, puisque toutes

les maisons vont avoir beaucoup de ce SIFT-type.

Question 15

Il permet d'unifier les données, que chaque patch soit représenté par les mêmes caractéristiques. Ici le codage correspond à un vecteur nul de taille k (k étant aussi la taille du dictionnaire) avec un 1 sur l'élément qui correspond au SIFT-type le plus proche du SIFT calculer à partir du patch. On pourrait utiliser un codage à n plus proches voisins, ce qui donnerai un vecteur nul de taille k avec un 1 sur les éléments correspondant aux n SIFT-type les plus proches.

Question 16

On cherche à avoir une description de l'image de façon. Ici on a découpé l'image en patch qu'on a encodé, il faut maintenant les agrégés pour avoir un vecteur caractérisant l'image entière, c'est à ça que sert le polling. On pourrait utiliser un polling tf-idf qui ferait ressortir les SIFT les moins observé parmi toutes les images, et donc ferait ressortir les spécificités des images.

Question 17

L'intérêt de normaliser est d'avoir toutes les composant du vecteur z à la même échelle. Encore une fois pour ne pas éloigner géométriquement deux images. Il pourrait y avoir une trop grande différence du fait que les images ne sont pas toutes de la même taille et donc elles n'auraient pas toutes le même nombre de patchs et des échelles bien différente. On pourrait utiliser la normalisation L1.