BEROUKHIM Keyvan 3506789

Partie 1 - SIFT

1) Le calcul des gradients Ix, Iy de l'image est effectué à l'aide des filtres de Sobel, qui sont de taille 3x3.
Dans notre cas, chaque filtre est **décomposable en deux vecteurs** hx, hy: [-1, 0, 1], [1, 2, 1].

2) Cette séparation du filtre est **utile d'un point de vue calculatoire** : on travaille avec deux vecteurs de taille 3, au lieu de travailler avec deux matrices 3 × 3.
L'un des vecteur s'occupe de **détecter les contours** et l'autre **permet de flouter**.

3) On utilise un masque gaussien pour lisser l'image afin de **réduire le bruit** et donner **plus d'importance** au centre et de **passer graduellement des pixels pris en compte aux pixels ignorés**.

4) L'orientation des gradients est discrétisée pour que l'on puisse créer des **histogrammes comparables** et pour rendre notre modèle **plus robuste aux rotations**.

5) On remplace les descripteurs dont la norme est inférieure à un certain seuil par des vecteurs nuls : ils ne sont **pas intéressants car ils n'ont pas assez de contraste**.
Ensuite, on normalise, seuil, puis re-normalise l'image afin de rendre le descripteur **invariant aux changements de luminosité**.

6) L'avantage du **SIFT** est qu'il est **robuste aux variations de luminosité, de cadrage, d'angle et de zoom**.
Ainsi on peut l'utiliser pour comparer des images, même si ces dernières ont été prises dans des **conditions différentes**.

7) Les **images de gradient** décrivent la **variation d'intensité des pixels dans l'image**.
L'**histogramme des SIFTs** est **difficile à interpréter**, le nombre de pics dans l'histogramme des SIFTs est à peu près proportionnel au **nombre de parties dans l'image**.

Partie 2 – DICTIONNAIRE VISUEL

8) Le dictionnaire sert à identifier des **”patterns” récurrents** (identifiés dans plusieurs images).
De plus, cela peut permettre de **réduire l’espace de représentation**, en particulier si le nombre de mots du dictionnaire est petit. On cherche ensuite à rapprocher chaque SIFT d’un de ces patterns.

9) L’équation (4) correspond à de l’optimisation strictement convexe, on résout donc l’optimisation par annulation du gradient.

On a :

$$grad = -2 \sum_i (xi - c)$$

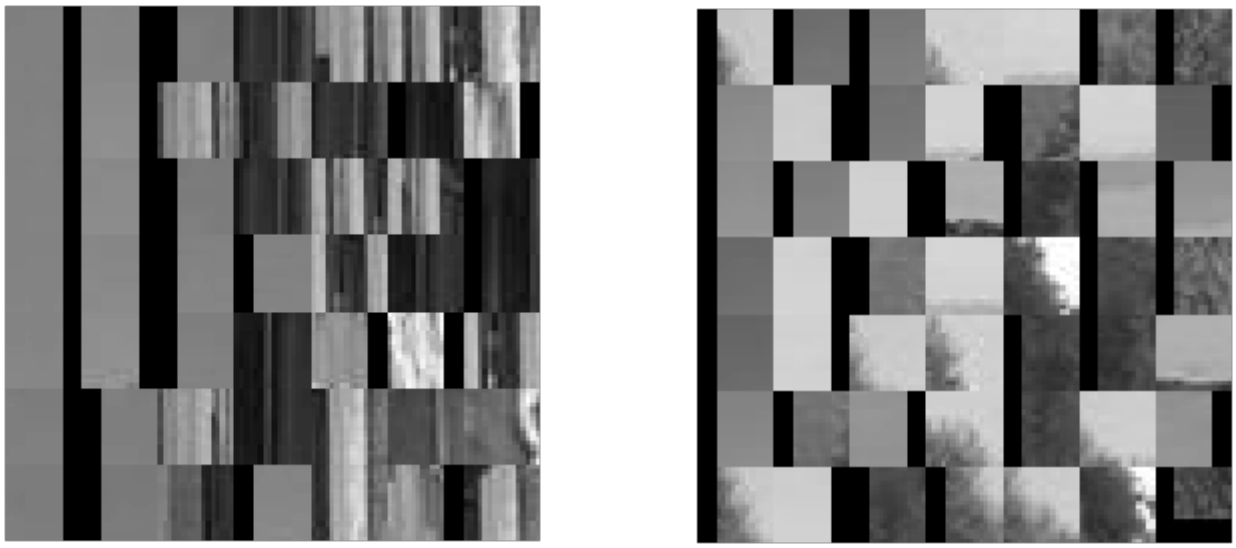
Donc :

$$\begin{aligned} & grad = 0 \\ \Leftrightarrow & \sum_i (xi - c) = 0 \\ \Leftrightarrow & c = 1/n \sum_i (x) \end{aligned}$$

10) On peut choisir le nombre de cluster idéal en faisant un “**grid search**”, ce qui consiste à essayer différentes valeurs pour ce paramètre et prendre celle qui convient le mieux. Cependant, le **temps d’exécution** de l’algorithme est élevé et **augmente avec le nombre de clusters**. En pratique, on choisira le **nombre de cluster maximal sans avoir à trop attendre**.

11) **On ne peut pas visualiser directement les éléments du dictionnaire**. On cherche donc les **patchs dont les représentations sont les plus proches de ces vecteurs** pour les visualiser.

12) Pour visualiser le dictionnaire visuel obtenu, on choisit des clusters aléatoires et on affiche des patchs qui leur sont proches.



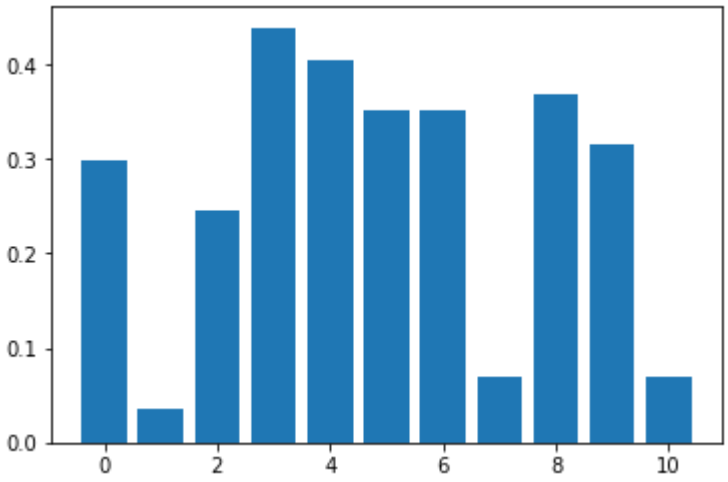
Régions proches de deux centres de cluster choisis aléatoirement

Le cluster de gauche correspond à des **lignes verticales**, alors que celui de droite représente à peu près le fait de contenir une **forme arrondie**.

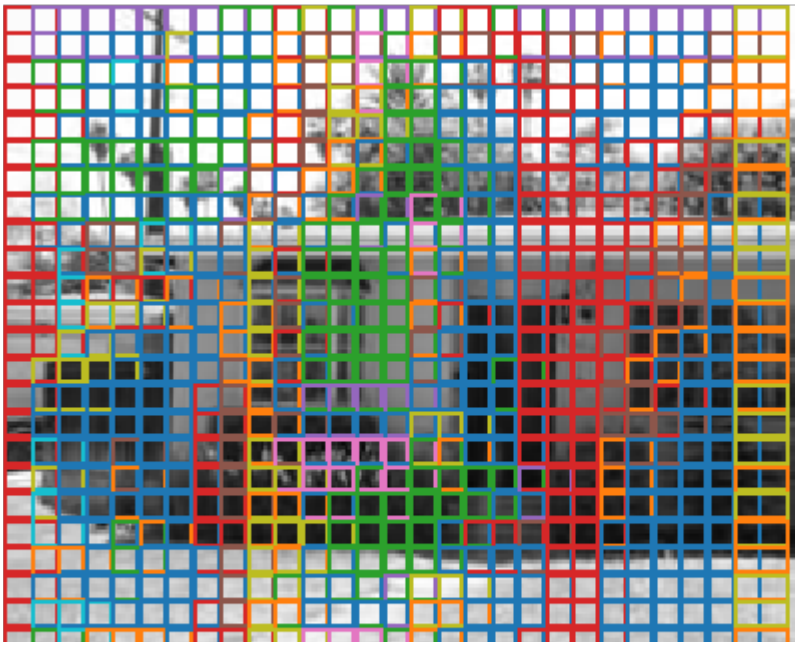
Partie 3 – BAG OF WORDS

13) z est une **représentation de notre image**. Ici on utilise un coding au plus proche voisin puis un pooling somme, il s'agit donc de la **fréquence des clusters parmi les patchs**, en rapprochant chaque patch du centroïde le plus proche.

14) Créer le dictionnaire visuel est coûteux à cause de la complexité de l'algorithme k-means. On a donc choisit ici de n'utiliser que 10 clusters afin d'accélérer la vitesse d'exécution.



Histogramme de la répartition des zones de l'image dans les différents clusters



Association de chaque zone de l'image à son cluster

Représentation de quelques clusters : le cluster violet représente une zone uniforme, le cluster bleu représente les régions possédant un objet dans le coin inférieur droit

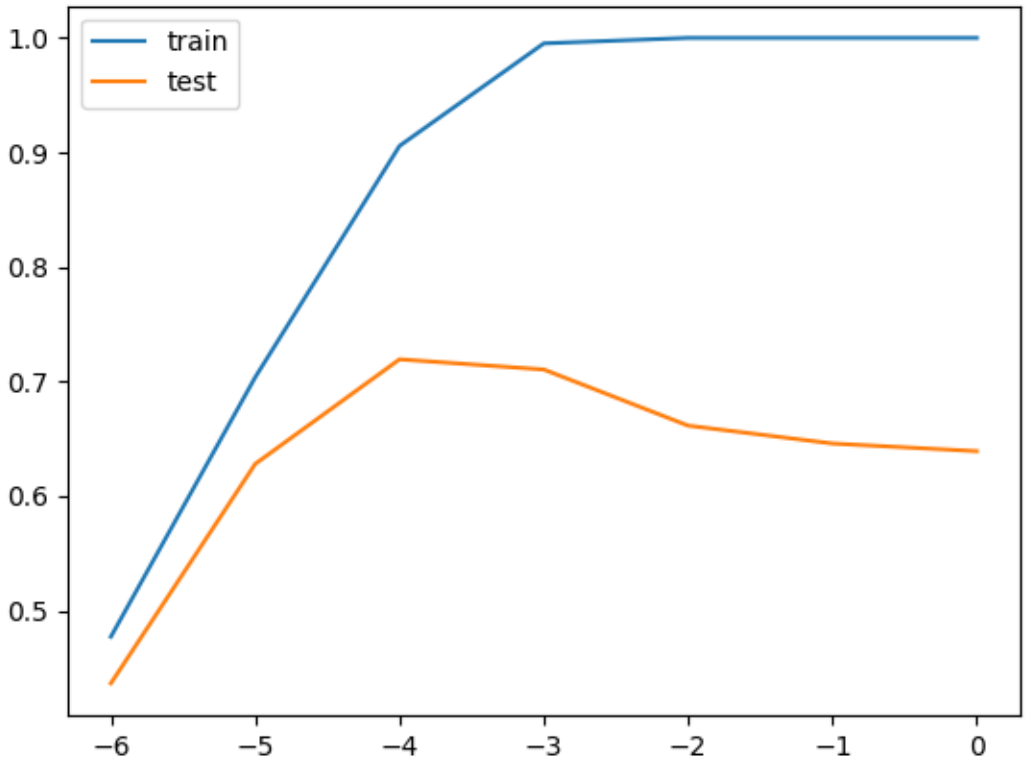
15) Le codage '**au plus proche voisin'** est celui permettant l'**implémentation la plus efficace**, notamment grâce à l'utilisation de **kd-tree** (qui, de plus, offrent la possibilité d'augmenter encore la rapidité d'exécution au coût d'une perte de précision de la solution trouvée).
Un autre codage pourrait être de faire un ''**soft assignement**" : une distribution d'appartenance de l'exemple à toutes les classes ce qui offre une meilleure **expressivité** et une meilleur **stabilité**.

16) Le '**pooling somme**' est celui traditionnellement utilisé pour du 'hard coding'. Il a pour avantage de **favoriser les mots visuels apparaissant beaucoup dans l'image**.
Une alternative au pooling somme est le **max pooling** qui résume les activations par la valeur des activations la plus élevée.

17) On utilise la normalisation L2 pour pouvoir comparer les représentations des images, on obtient un **vecteur unitaire** donc pour lequel **seule la direction importe**. Avec une autre norme, comme la norme L1, on n'obtiendrait pas cette propriété.

Partie 4 – CLASSIFICATION

18) L'erreur en train est inférieure à l'erreur en test. Par **grid search**, la valeur de C optimale trouvée est 10^{-3}. Au delà de ce seuil, le modèle est **moins tolérant aux fautes sur l'ensemble de train** et commence à **sur-apprendre** : la **précision en train augmente** et tend même vers 1 mais la **précision en test diminue**.



Courbes de précision en fonction de la valeur de l'hyper-paramètre C (échelle logarithmique)

Une fois la valeur optimale de C trouvée, on réentraîne un SVM sur l'ensemble des données d'apprentissage et de validation et on mesure ses performances sur le jeu de test. On obtient une **précision de 70 %** environ (soit un taux d'erreur de 30%). Cette précision est largement supérieure à l'aléatoire car avec 15 classes, en classifiant aléatoirement les images on aurait une précision de 1/15 = 7 % (soit un taux d'erreur de 93%). Le classifieur entraîné a donc un **taux d'erreur trois fois inférieur à celui de l'aléatoire**.

19) On ne peut pas utiliser l'ensemble de test pour le grid search car utiliser l'ensemble de test pour l'apprentissage **biaiserait les scores obtenus**.

20) Étant donné une nouvelle image on utilise la chaîne de traitement suivante :

1 - Déterminer les **points d'intérêt** en trouvant les angles de l'image (détecteur de **HARRIS**) (on ne l'a pas fait dans ce TP)

2 - On **extrait les caractéristiques** de l'image au niveau des points d'intérêt avec des **SIFTs**. On obtient une représentation en **Bag Of Features**

3 - En utilisant le **dictionnaire de mots visuels** déjà **calculé sur la base de données** d'apprentissage, on détermine la Représentation **Bag Of (Visual) Words** en affectant les sifts de l'image aux mots du dictionnaire dont il est proche. Dans notre cas, on fait du "hard coding" puis du "sum pooling" mais il existe d'autres méthodes

4 - On utilise ensuite notre **SVM préentraîné** afin de déterminer la **classe de l'image**. Le cas multi-classe est traité en "ovr", c'est à dire en "Un Contre Tous"

21) On peut améliorer notre chaîne de traitement de différentes manières,

- Pour l'extraction de features, on pourrait
  - déterminer les **points d'intérêt** de l'image
  - rendre nos SIFT "**scale invariant**"
  - faire un "**soft assignation**" au lieu d'un "hard coding" afin de mieux gérer les points à égale distance de deux mots du dictionnaire par exemple.
- Pour la classification, on pourrait
  - essayer **d'autres modèles**, par exemple un **"Random Forest" car ce type de modèle gère bien le multi-classe** ou un réseau de neurones.