
DVI1: SIMILARITÉ À LA BASE DE LA COULEUR

MST: Artificial Intelligence and Data Sciences

Prepared by:
AHMED SAMADY
Supervised by:
Pr. M. AIT KBIR

1 But

Caractériser les images par des histogrammes couleurs et par les couleurs dominantes. Puis calculer la distance entre les images couleurs en se basant sur les caractéristiques calculées (Outils de développement: OpenCV sous Python).

2 Exercice 1

But: Calculer et afficher l'histogramme couleur d'une image RGB, l'image des trois graphes est à construire depuis le tout début à l'aide des fonctionnalités d'OpenCV.

Étapes de réalisation:

- Sélectionner aléatoirement deux images différentes du répertoire 1 avec `numpy.random.randint()`.
- Diviser les canaux de couleur (b, g, r) des deux images avec `cv2.split()`.
- Créer un histogramme pour chaque canal de couleur de l'image avec `cv2.calcHist()` et `cv2.line()`.
- Redimensionner l'image des histogrammes avec `cv2.resize()`.
- Afficher les histogrammes d'image avec `cv2.imshow()`.

Résultats:

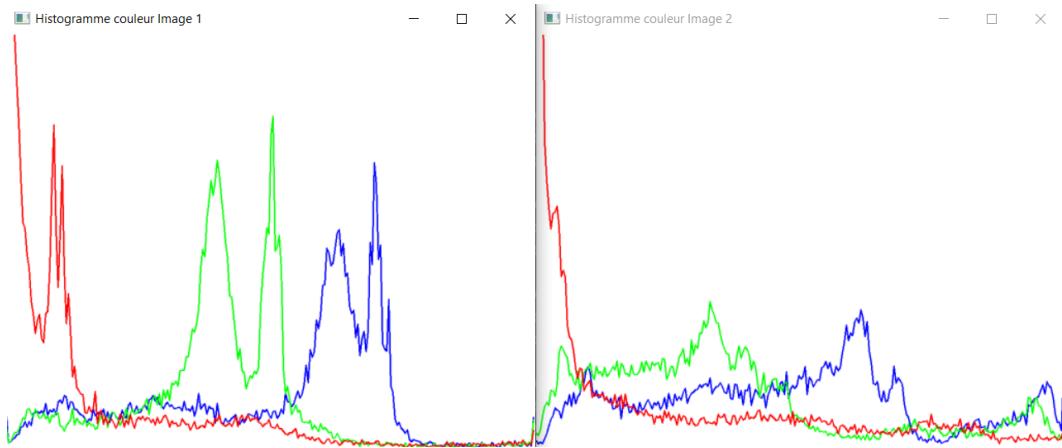


Figure 1: Histogramme couleurs des deux images

3 Exercice 2

But: Calculer les couleurs dominantes d'une images en prenant en compte les couleurs avec 95% de présence, pour les Kmeans, prendre un K élevé > 16 (Chaque image aura un nombre de couleurs dominantes différent).

Étapes de réalisation:

- Importer les bibliothèques OpenCV, Numpy, os, json et KMeans de sklearn.
- Charger les images depuis le répertoire Dossier 1 (os.listdir()).
- Pour chaque image:
 - Lire l'image (cv2.imread()).
 - Convertir l'image en espace de couleur Lab (cv2.cvtColor()).
 - Appliquer l'algorithme KMeans avec 16 clusters.
 - Compter le nombre de pixels pour chaque cluster.
 - Calculer le nombre total de pixels.
 - Trier les couleurs par nombre de pixels.
 - Extraire les couleurs dominantes.
 - Convertir les valeurs de float en int.
 - Afficher les couleurs dominantes.
 - Sauvegarder les couleurs dominantes dans un dictionnaire.
 - Charger les données existantes depuis un fichier JSON.
 - Mettre à jour les données avec les nouvelles couleurs dominantes.
 - Sauvegarder les données mises à jour dans le fichier JSON.

Résultats:

Extraits des résultats dans le fichier DV11_col.json:

```
{  
    "1468.jpg": [  
        [105, 119, 144],  
        [93, 119, 143],  
        [81, 118, 141],  
        [248, 124, 132],  
        [118, 121, 146],  
        [67, 119, 138],  
        [51, 119, 136],  
        [239, 120, 126],  
        [216, 112, 118],  
        [234, 128, 148],  
        [186, 129, 152],  
        [137, 124, 148],  
        [164, 128, 153],  
        [208, 130, 152],  
        [28, 122, 133],  
        [159, 121, 133]  
    ],  
    ...  
}
```

4 Exercice 3

But: Proposer une ou plusieurs distances pour mesurer la similarité entre les images par rapport la couleur pour chaque caractéristique.

Distance Euclidienne

La distance euclidienne est la distance la plus simple et la plus courante entre deux points dans un espace euclidien.

```
def euclidean_distance(c1, c2):
    return np.linalg.norm(np.array(c1) - np.array(c2))
```

Distance Bhattacharyya

La distance Bhattacharyya est une mesure de la similarité entre deux distributions de probabilité. Elle est basée sur la distance de Bhattacharyya, qui est une mesure de la divergence entre deux distributions de probabilité.

```
def bhattacharyya_distance(c1, c2):
    h1 = np.histogram(c1, bins=256, range=(0, 256))[0]
    h2 = np.histogram(c2, bins=256, range=(0, 256))[0]
    h1 = h1 / np.sum(h1)
    h2 = h2 / np.sum(h2)
    return np.sqrt(1 - np.sum(np.sqrt(h1 * h2)))
```

5 Exercice 4

But: Prendre en compte le jeu d'images du sous-Dossier DVII1 pour le test (Voir le dossier partagé Google Drive). Faire une comparaison des résultats de la similarité en utilisant chaque distance séparément puis la distance global.

5.1 Couleurs dominantes

Étapes de réalisation:

- Définir les chemins des dossiers contenant les images : `dossier1` et `dossier2`.
- Lister les images dans `dossier1` (16 images) et `dossier2` (4 images) avec `os.listdir()`.
- Initialiser un dictionnaire `res` pour stocker les distances entre les images.
- Sélectionner une image aléatoire de `dossier2` avec `numpy.random.randint()`.
- Lire l'image sélectionnée avec `cv2.imread()`.
- Convertir l'image en espace de couleur Lab avec `cv2.cvtColor()`.
- Redimensionner l'image en un tableau de pixels avec `reshape()`.

- Appliquer un clustering des couleurs avec KMeans (`n_clusters=16`) et `fit()`.
- Calculer les couleurs dominantes et les convertir en entiers.
- Boucler sur les autres images et calculer les distances euclidiennes et bhatta entre les couleurs dominantes de l'image de test et les autres images.
- Calculer la distance globale des distances et les sauvegarder dans le dictionnaire `res`.
- Trier les distances par ordre croissant selon la distance globale.
- Afficher l'image de requête et les 6 images les plus similaires avec `matplotlib.pyplot`.

Résultats:



Figure 2: Résultats de la similarité des images par rapport à la couleur dominante - "3402.jpg"



Figure 3: Résultats de la similarité des images par rapport à la couleur dominante - "8965.jpg"



Figure 4: Résultats de la similarité des images par rapport à la couleur dominante - "1468.jpg"



Figure 5: Résultats de la similarité des images par rapport à la couleur dominante - "8116.jpg"

5.2 Histogrammes

Étapes de réalisation:

On suit les mêmes étapes que pour les couleurs dominantes, mais on remplace le calcul des couleurs dominantes par le calcul des histogrammes des images.

Résultats:



Figure 6: Résultats de la similarité des images par rapport à l'histogramme - "3402.jpg"

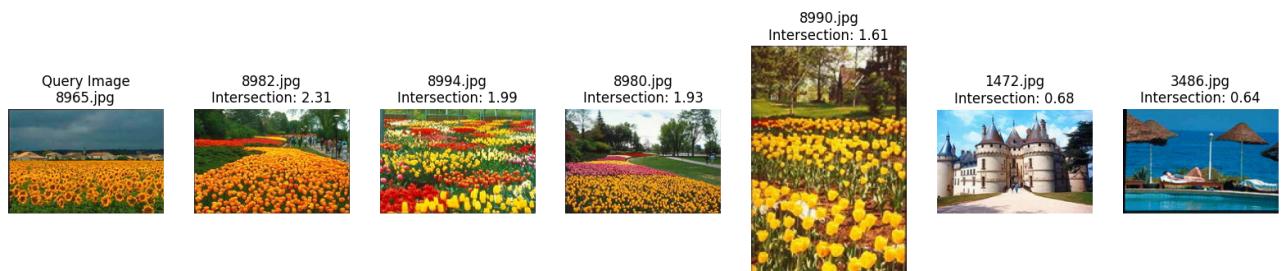


Figure 7: Résultats de la similarité des images par rapport à l'histogramme - "8965.jpg"

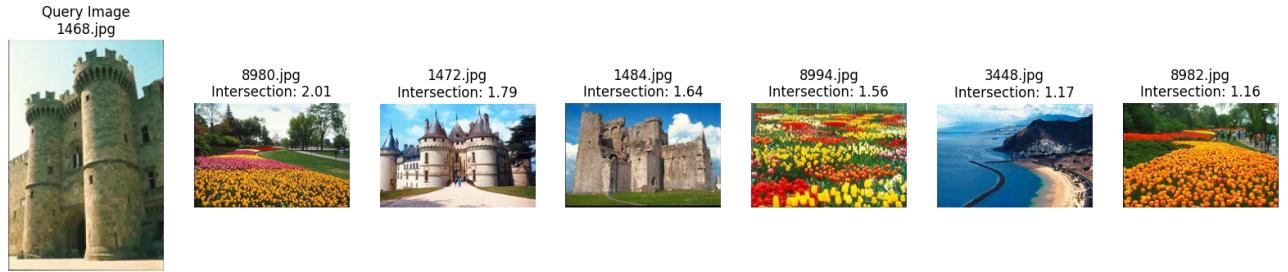


Figure 8: Résultats de la similarité des images par rapport à l'histogramme - "1468.jpg"



Figure 9: Résultats de la similarité des images par rapport à l'histogramme - "8116.jpg"

5.3 Conclusion

Les résultats obtenus sont globalement satisfaisants, mais il y a des images de la catégorie des manoirs où les résultats sont complètement erronés. De plus, il semble que l'intersection des histogrammes soit plus efficace que celle des couleurs dominantes, car elle est plus cohérente.