Jean-Nicola Blanchet
GPA659 – Vision Artificielle

Exercice MATLAB 6:

Descripteur de texture

Cet exercice est à faire en classe ou à la maison. Vous devez soumettre votre réponse à l'enseignant à la fin du cours ou durant le laboratoire (selon ce qui a été indiqué en classe). Chaque exercice vaut 1%, jusqu'à concurrence de 6%. Vous avez deux semaines pour remettre chaque exercice.

L'algorithme des **motifs binaires locaux** (LBP, ou local binary patterns) est outil très puissant pour décrire la texture de manière invariable à l'échelle, la rotation, et la translation. L'algorithme est bien simple, on associe chaque pixel de l'image à un code (souvent représenté en binaire). Le code d'un pixel est calculé de la manière suivante :

1) Soit un pixel et son voisinage 8
2) On seuil les 8 pixels en utilisant le pixel du centre comme valeur de seuillage.
3) On concatène les 8 valeurs binaires pour former un code binaire.

Par exemple :

14	15	23
17	25	56
22	57	58

14	15	23
17	25	56
22	57	58

14	15	23	56	58	57	22	17

Cet exemple nous permet d'obtenir le code binaire **0001 1100**, ou **28** en code décimal.

Il existe donc 256 codes binaires distincts décrivant la texture locale de chaque pixel. Les codes similaires peuvent aussi être concaténés afin de réduire de nombre de codes uniques à 10. Par exemple : 1000 000 est similaire à 0100 000 si on ignore l'effet de la rotation.

La dernière étape consiste à générer un histogramme des codes identifiés. On obtient donc un vecteur de caractéristiques de taille 10, où chaque valeur représente la fréquence (ou la probabilité) de chacun des 10 codes uniques.

Jean-Nicola Blanchet
GPA659 – Vision Artificielle

Nous allons maintenant tester la robustesse du descripteur face à un changement d'échelle ou d'orientation d'une texture.

1) Charger les images `'bricks.jpg'` et `'carpet.jpg'` déjà connues de MATLAB.

2) Extraire le descripteur LBP des deux images. Nous allons utiliser les deux histogrammes LBP comme **prototypes de référence**. Utiliser la fonction extractLBPFeatures. Pour obtenir un histogramme de taille 10, nous devons spécifier de combiner les codes binaires selon une stratégie invariable à l'orientation. Ceci se fait en désactivant le paramètre Upright : `extractLBPFeatures(image,'Upright',false)`

3) Utilisez le code suivant pour générer une nouvelle image aléatoirement :

```
if randi(2) == 1
    M = imread('bricks.jpg');
else
    M = imread('carpet.jpg');
end

x0 = randi(size(M,1)/2);
x1 = randi(size(M,1)/2) + size(M,1)/2;
y0 = randi(size(M,2)/2);
y1 = randi(size(M,2)/2) + size(M,2)/2;
M = M(x0 :x1,y0 :y1);
M = imrotate(M,randi(5),'crop');

imshow(M);
```

4) Calculer l'histogramme LBP de la nouvelle image M.
5) Calculer la distance euclidienne entre l'histogramme LBP de l'image M et l'histogramme de référence pour la brique. Ceci peut se faire simplement via une soustraction suivie de la fonction `norm()`. La distance euclidienne fonctionne très bien entre deux vecteurs de taille 10 : $d(ref, m) = \sqrt{(ref_1 - m_1)^2 + (ref_2 - m_2)^2 + \cdots + (ref_{10} - m_{10})^2}$
6) Répéter l'étape 5 pour l'histogramme de référence de l'image du tapis.
7) En fonction de la plus petite distance obtenue (en 5 et 6), afficher l'un ou l'autre des titres suivants à la figure :

```
title('M contient de la brique');
title('M contient du tapis');
```

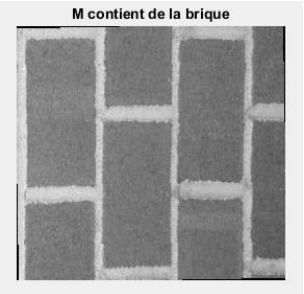

Figure 1 – exemple de sortie