Projet - Synthèse et classification de textures

Julie Digne

Ce projet comporte deux parties, la première consiste à synthétiser des textures, la deuxième partie développe un outil de classification de textures, à tester ensuite sur vos textures synthétisées.

Les données de ce projet sont dues à Efros, Leung (textures 0,1,2,6), Brodatz (texture 3) et l'archive IPOL(4,5), vous les trouverez ici: `http://liris.cnrs.fr/julie.digne/cours/textures_data.zip`

Le projet peut se faire en binome mais les notes seront individuelles avec une soutenance.

Les librairies autorisées sont pillow, numpy, matplotlib et PyMaxFlow `https://pmneila.github.io/PyMaxflow/` pour la partie graphcut (si vous faites le choix d'implémenter la synthèse de textures par graphcut).

1 Synthèse de Textures

- Implémenter un algorithme de synthèse de textures vu en cours (le plus facile étant Efros-Leung)
- Tester la synthèse de textures
- Tester différents paramètres. Pour Efros-Leung: epsilon, taille du patch, choix de la couleur du pixel central: aléatoire ou moyenne des pixels centraux des patchs de Ω'

2 Classification non supervisée de textures

Les données peuvent être téléchargées ici: `https://multibandtexture.recherche.usherbrooke.ca/colored%20_brodatz.html`. Il s'agit de la base de données Brodatz colorée (U. Sherbrooke, Canada), on va s'intéresser à la classification de patchs de textures.

La première étape est de coder un classifieur et de vérifier que tout fonctionne en 2D et en 3D. Vous pouvez pour cela utiliser les données de mélanges de gaussiennes disponibles ici: `http://liris.cnrs.fr/julie.digne/cours/classif_data.zip`

1. **Classification - cas test**
   - Implémenter un algorithme de classification au choix sur des données 2D fournies. Attention ce classifieur doit être ensé pour fonctionner en dimension quelconque.
2. **Préparation des données de texture**
   - Découper les images en patches (sans recouvrement)
   - Garder la moitié des patches pour optimiser la classification (ensemble $\mathcal{D}_O$ et l'autre comme ensemble de test $\mathcal{D}_T$.
3. **Classification des textures**
   - Choisir une façon de décrire les patches de textures (histogrammes, filtres...)
   - Optimiser le classifieur sur l'ensemble $\mathcal{D}_O$, vérifier que les données de $\mathcal{D}_T$ sont bien classifiées.

- Utiliser la synthèse de texture pour synthétiser des données à partir de patches de $\mathcal{D}_T$, vérifier que les patches des nouvelles textures sont bien classifiés.
- Une fois que le classificateur est optimisé, ajouter du bruit et du flou et vérifier que les textures sont bien classifiées.

3 Quelques commandes

```
from PIL import Image
import numpy as np
img = Image.open("textures_data/text0.png")
npimg = np.array(img)
imgtext = Image.fromarray(np.uint8(npimg))
```

References

[1] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In Proceedings of the International Conference on Computer Vision Volume 2, ICCV '99, pages 1033–1068, Washington, DC, USA, 1999. IEEE Computer Society.