OS25 - TP Final

Décodage de codes Spotify

Introduction au projet

L'objectif est de décoder un code Spotify.

Les codes Spotify sont une fonctionnalité intégrée à l'application Spotify, offrant une manière pratique de partager de la musique avec d'autres utilisateurs. Ces codes prennent la forme de code barres avec des barres de 8 tailles différentes.

Exemple:

Le code

1 - Chargement de l'image

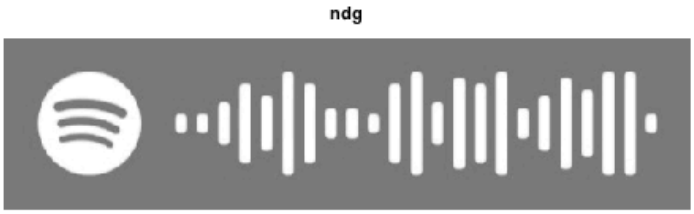
```
clear all;
close all;
pkg load image;

img = imread("chanson1.jpg");
subplot(2, 2, 1);
imshow(img);
title("code Spotify");
```

On commence par charger l'image "chanson1.jpg" :

2- Conversion en niveaux de gris

On convertit l'image rgb en ndg avant de faire le seuillage :

```
image_ndg = rgb2gray(img);
subplot(2, 2, 2);
imshow(image_ndg);
title("ndg");
```



3- Seuillage

On effectue un seuillage, on convertit l'image en niveau de gris en blanc et noir :

```
seuil = 150;
binary = 255 * (image_ndg > seuil);
subplot(2, 2, 3);
imshow(binary);
title("seuil");
```

> **Note**
>
> On réalise des tests pour trouver la valeur du seuillage optimale

4- Segmentation et étiquetage

```
label_image = bwlabel(binary);
num_objects = max(label_image(:));
```

- On utilise la fonction `bwlabel` pour étiqueter les objets dans une image binaire. Cette fonction attribue un label à chaque objet dans l'image binaire. Après cette étape, chaque pixel d'un objet est remplacé par un nombre entier correspondant à l'objet auquel il appartient, et les pixels de fond conservent la valeur zéro.
- Avec la fonction `max`, on attribue à `num_objects` la somme de tous les labels.

Ensuite, avec ce code :

```
imshow(label2rgb(label_image));
title("Objets etiquetes");
```

On affiche les labels avec une couleur différente.

Algorithme de décodage

On commence par calculer l'air de chaque étiquette :

```
props = regionprops(label_image, 'Area');
```

Ensuite, ces deux lignes de code calculent un intervalle moyen entre les seuils en utilisant les aires de deux objets de référence:

- la barre n°2 qui est toujours la plus petite.
- la barre n°13 qui est toujours la plus grande.
  Puis on définit les seuils pour classer les objets en fonction de leur air :

```
intervalle = (props(2).Area + props(13).Area) / 7;
seuilCode = [props(2).Area, props(2).Area + (1:6) * intervalle, 5000];
```

Maintenant on créé un vecteur data de la taille de `num_objects -1` (car on enlève le logo Spotify).

```
data = NaN(1, num_objects - 1);
```

Maintenant, on peut enfin décoder le code Spotify:

```
for i = 2:num_objects
    idx = find(seuilCode(1:end-1) ≤ props(i).Area & props(i).Area ≤ seuilCode(2:end));
    if ~isempty(idx)
        data(i-1) = idx;
    end
end
```

Ligne 2 :

- La 2ème ligne utilise la fonction `find` pour rechercher les indices des seuils dans le vecteur `seuilCode` qui sont respectés par l'aire de l'objet actuel, représentée par `props(i).Area`.
- `seuilCode(1:end-1)` représente les limites inférieures des seuils, tandis que `seuilCode(2:end)` représente les limites supérieures. Ainsi, chaque élément de `seuilCode(1:end-1)` est comparé à l'aire de l'objet, et chaque élément de `seuilCode(2:end)` est comparé à l'aire de l'objet pour déterminer dans quelle plage l'aire de l'objet se situe.

Ligne 3 :

- Si des indices sont trouvés (c'est-à-dire si l'objet correspond à au moins un seuil), alors l'indice de la plage dans laquelle l'aire de l'objet se situe est attribué à l'élément correspondant dans le vecteur `data`. Cela signifie que l'objet est classé dans la catégorie correspondante à cet indice.
- Si aucun seuil n'est respecté, l'élément correspondant dans `data` reste NaN, ce qui signifie que l'objet n'appartient à aucune catégorie définie par les seuils.

Tests

Test 1

Code

Décodage

data [23x1 double]

	1
1	1
2	1
3	3
4	6
5	4
6	8
7	6
8	2
9	2
10	1
11	6
12	8
13	3
14	7
15	6
16	8
17	2
18	4
19	7
20	5
21	8
22	8
23	1

Le décodage est bon.

Test 2

Code

Décodage

data [23x1 double]

	1
1	1
2	6
3	2
4	3
5	1
6	7
7	5
8	4
9	8
10	2
11	7
12	8
13	8
14	8
15	8
16	4
17	2
18	7
19	4
20	8
21	1
22	8
23	1
24	

Le décodage est bon

Test 3

Code

Décodage

data [23x1 double]

	1
1	1
2	4
3	2
4	2
5	1
6	2
7	8
8	6
9	6
10	4
11	8
12	8
13	3
14	3
15	8
16	8
17	7
18	4
19	5
20	2
21	7
22	6
23	1

Le décodage est bon