Sujet 1 – Segmentation d'un objet dans une image de type fond/forme en utilisant des superpixels

Sylvie Chambon

Objectifs

Le but de ce TP est de **segmenter** une image grâce à une technique de découpage en **superpixels**, puis de **classer** simplement par un seuillage ou un critère de forme ces régions en régions extérieures (représentées en noir) ou intérieures (représentées en blanc) à la forme, comme montré sur la figure 1.

Données fournies

Pour ce sujet, nous vous fournissons une archive, cf. lien `moodle`, contenant un dossier `images` avec 36 images d'un dinosaure sur un fond bleu de nom `viff.0**.ppm` où `**` correspond au numéro de l'image, cf. figure 1. De plus, le dossier contient `TP_segmentation.m` : script à compléter permettant d'effectuer les étapes nécessaires pour segmenter.

(a) (b)

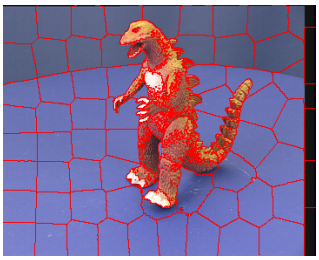
Figure 1 – Un exemple d'image fournie, (a), et du résultat de segmentation attendu, (b).

Travail noté

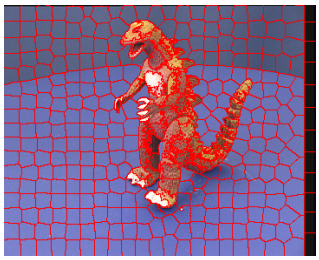
Vous avez **3 séances** pour réaliser ce sujet. Vous serez évalués en deux temps :

1. Vous aurez à répondre à des questions en ligne sur moodle, au cours d'une séance dédiée, le **lundi 12 décembre à 10h15**.
2. Vous déposerez une archive de nom `Nom1_Nom2.zip` contenant tous les fichiers `.m` nécessaires, sauf les images, et un `LISEZMOI.txt` expliquant comment exécuter vos programmes. Cette archive sera à déposer sur `moodle` et elle sera utilisée pour exécuter vos codes pendant la session de **tests de code** qui aura lieu à **la même séance**. Un ordre de passage sera précisé au préalable.

1 Segmentation en superpixels



100 superpixels



400 superpixels

FIGURE 2 – Exemple de sur-segmentations obtenues – Ici, pour une compacité constante, nous avons fait varier le nombre de superpixels. Nous pouvons observer que les superpixels sont assez compactes. C'est-à-dire que leur forme se rapproche d'un cercle et il n'y a pas de variations brusques de la forme du contour. Cela signifie que m a une valeur assez faible qui donne peu de poids au critère de ressemblance colorimétrique. Dans cet exemple, nous favorisons donc la distance en position.

Nous vous proposons de mettre en œuvre l'agorithme de construction de superpixels proposé par Achanta, SLIC pour *Simple Linear Iterative Clustering*, cf. l'article ainsi qu'une présentation disponible sur `moodle`. L'approche que vous allez également aborder en cours se résume à suivre les 3 étapes suivantes :

1. Placement régulier de germes : pour compléter, lire la présentation sur `moodle` ;
2. Calcul des superpixels avec une approche par k-moyenne : cette approche est décrite par la suite ;
3. Renforcement de la connexité **(optionnel)**: Pour coder simplement cette étape, il faut fusionner les petites régions avec leur plus grande région voisine. Plus précisément, en supposant que les régions ont une taille moyenne de $\frac{N_p}{K}$ avec N_p le nombre de pixels de l'image et K le nombre de superpixels à construire, une petite région correspond à une région de taille inférieure à un pourcentage de cette moyenne.

L'algorithme des k-moyennes ou *k-means* consiste à chercher la meilleure répartition des pixels en N classes, la valeur de N étant fixée par l'utilisateur. Il s'agit d'un algorithme itératif, dont les principales étapes sont :

a. Partition de l'image en N régions, suivant un critère c à définir ;
b. Estimation des centres C_k, $k \in [1, N]$, de ces régions ;
c. Tant que les centres sont modifiés :
   i. Pour chaque pixel de l'image, recherche du centre C_k le plus proche, au sens du critère c, et affectation du pixel considéré à la classe k ;
   ii. Mise à jour des centres C_k des N régions ainsi constituées.

Il n'existe pas de preuve de convergence de cet algorithme vers l'optimum global. D'ailleurs, le résultat dépend généralement de l'initialisation.

Dans la figure 2, nous illustrons le type de résultat que nous obtenons. Nous avons les paramètres d'entrée suivants :

— K, le nombre de superpixels ;

— m, la compacité, utilisée pour calculer le poids $(\frac{m}{S})^2$ donné au terme de distance en position par rapport à la distance en couleur, cf. la formule (4.4), page 58, dans le polycopié distribué.

Dans cet exemple, nous avons fait varier le nombre de superpixels utilisés, en utilisons une compacité constante.

À faire

Nous vous demandons de coder cette approche. Vous pouvez utiliser la fonction `kmeans` de `matlab` mais ce n'est pas obligatoire.

2 Segmentation binaire

Pour classer les régions obtenues en régions extérieures ou intérieures, vous pouvez effectuer un seuillage sur la couleur des centres ou une sélection en fonction de la forme plus ou moins compacte des régions obtenues (cf. critère de compacité défini en cours, formule (4.3), page 56 du polycopié).

À faire

Coder la segmentation binaire en choisissant l'approche que vous voulez.