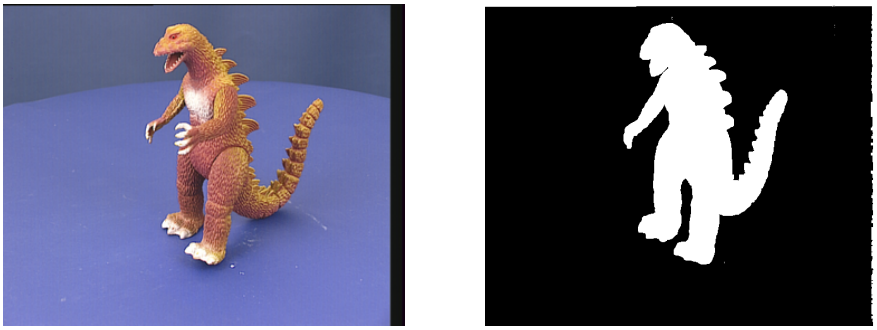
TP2 – Extraction de l'axe médian d'un objet dans une image de type fond/forme

Objectifs

Le but de ce TP est de retrouver l'axe médian d'un objet dans une image. Pour cela, on propose de segmenter une image grâce à une technique de découpage en superpixels, 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, puis d'appliquer un algorithme de calcul de l'axe médian.

Données fournies

Pour ce projet, 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.



(a) (b)

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

Ce qu'il faut rendre

Vous avez deux séances pour réaliser ce mini-projet. Par binôme, vous déposerez une archive de nom `Nom1_Nom2.zip` contenant uniquement les fichiers `.m` nécessaire, un `LISEZMOI.txt` expliquant comment exécuter vos programmes et un rapport de 4 pages maximum au format pdf et de nom `Rapport_Nom1_Nom2.pdf` expliquant et illustrant les résultats obtenus. Cette archive sera a déposer sur `moodle`.

1 Segmentation en superpixels

Nous vous proposons de mettre en œuvre l'agorithme de construction de superpixels proposée par Achanta, SLIC pour *Simple Linear Iterative Clustering*, cf. transparents 25 à 32 du cours 3. Comme nous l'avons vu, l'approche se résume à suivre les trois étapes suivantes :

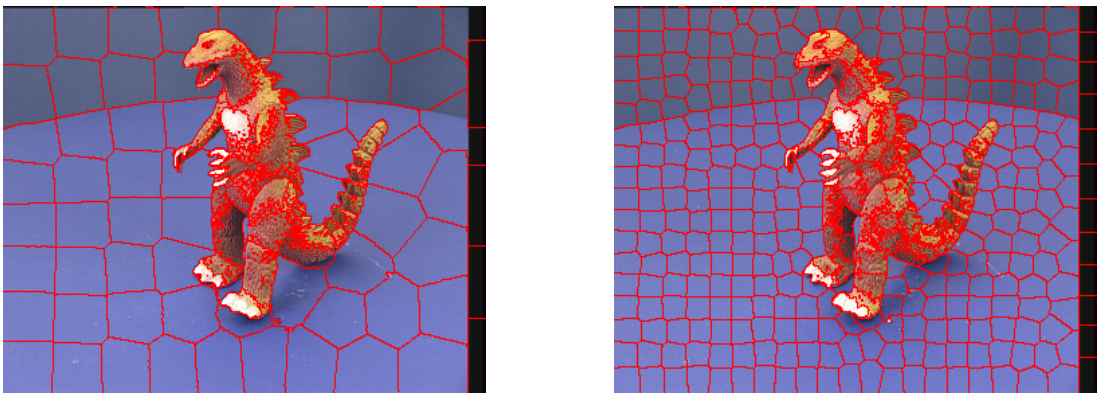
1. Placement régulier de germes ;
2. Calcul des superpixels avec une approche par k-moyenne ;
3. Renforcement de la connexité.

À faire

Coder cette approche en reprenant le code de la fonction `kmeans` de `matlab`, déjà étudiée au TP2 de la partie traitement d'images. Il faut, entre autres :

- Pour initialiser les centres, utiliser l'option `start`.
- Pour utiliser une distance comme dans l'approche SLIC, convertir l'image dans le système de représentation *Lab* puis modifier la manière de calculer la distance dans la fonction `kmeans`.
- Pour coder simplement la dernière étape, fusionner les petites régions avec leur plus grande région voisine. Plus précisément, en supposant que les régions ont une taille moyenne de $\frac{Nbpixels}{K}$ avec $Nbpixels$ 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.

Dans la figure 2, nous illustrons le type de résultat que l'on obtient avec différentes valeurs pour K, le nombre de superpixels, à compacité constante (la compacité correspond au poids donné au terme de distance en position par rapport à la distance en couleur).



100 superpixels 400 superpixels

FIGURE 2 – Exemple de sur-segmentations obtenues.

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 compactes des régions obtenues (cf. critère de compacité défini en cours, transparent 15 du cours 3).

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

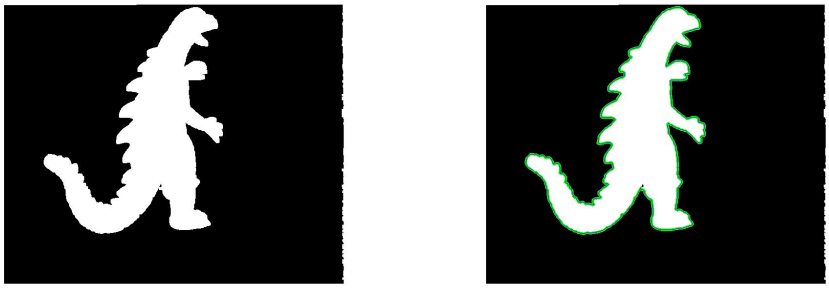
3 Estimation de l'axe médian

Dans cette partie, vous calculez un discrétisation de l'axe médian associé à la forme binaire que vous avez obtenue. Attention, votre image binaire doit être *a priori* un objet blanc (1) sur un fond noir (0).

3.1 Estimation de la frontière

À faire A partir de l'image binaire, retrouvez les coordonnées dans l'image des points de la frontière de la forme. L'ordre n'est pas important.

Vous pourrez, par exemple, utiliser la fonction `bwtraceboundary`.



(a) (b)

FIGURE 3 – L'image binaire, (a), et la frontière trouvée (en vert), (b).

3.2 Estimation des points du squelette

Nous avons vu en cours que les points du squelette sont donnés par les sommets du diagramme de Voronoï associé à un ensemble de points segmentant le bord, c'est-à-dire, les centres des cercles circonscrits aux triangles de la triangulation de Delaunay. Vous pourrez ajuster la densité des points du bord pour avoir un résultat correct, en gardant un temps de calcul raisonnable.

À faire Trouver les points appartenant au squelette, soit en utilisant la triangulation de Delaunay, soit avec le diagramme de Voronoi. `Matlab` propose des fonctions `delaunay` et `voronoi`, et variantes, permettant de calculer l'une ou l'autre. Pour chaque point de l'axe, on stockera l'information de rayon. On ne s'intéresse qu'à l'axe médian interne, il faut donc filtrer les points pour ne garder que ceux qui sont à l'intérieur de la forme.

3.3 Estimation de la topologie du squelette

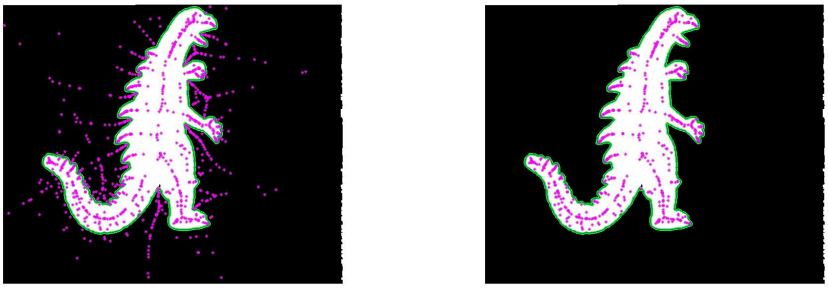
Trouver une condition pour qu'il existe une arête entre deux points du squelette.

À faire Tracer l'axe médian.

Remarque : vous pouvez utiliser une matrice d'adjacence. L'axe médian connecté peut dans ce cas être tracé avec la fonction `gplot`.

3.4 (Bonus) M.A.T et filtrage

Vous pouvez dessiner les cercles autour de chaque point de l'axe médian pour vérifier que l'axe médian avec les rayons associés modélisent effectivement la forme binaire.



(a) (b)

FIGURE 4 – Les points de l'axe médian, (a), et les points de l'axe médian interne, (b).

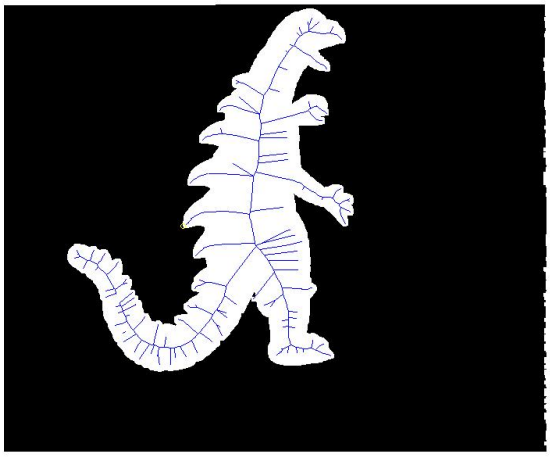


FIGURE 5 – L'axe médian interne (b).

Vous pouvez aussi proposer une méthode de filtrage des données, pour avoir un axe médian un peu plus propre (moins de branches).