Projet Segmentation par superpixels et modélisation 3D

1. Objectif du projet

Ce projet consiste à construire une chaîne complète de **traitement et modélisation 3D** à partir d'un ensemble d'images. Il s'appuie sur :

- la **segmentation d'image** via superpixels (algorithme SLIC),
- des critères de **couleur** et de **compacité** pour filtrer les régions pertinentes,
- une **optimisation par connexité** pour fusionner les petits superpixels,
- la **triangulation de Delaunay** et la **suppression de tétraèdres hors forme**,
- et enfin la **construction d'un maillage 3D** modélisant la forme d'un objet.

2. Structure du projet

```
├── main_Partie_1_2_3.m          % Script principal partie 1 à 3
├── Faibles_gradients.m          % Déplacements des centres vers les faibles gradients
├── Algorithme_SLIC.m            % Implémentation de l'algorithme SLIC
├── Optimisation_connexe.m       % Fusion des petits superpixels selon la connexité
├── Segmentation_couleur.m       % Masque binaire basé sur un critère de couleur
├── Segmentation_compacite.m     % Masque binaire basé sur un critère de compacité
├── Filtrage_centres.m           % Filtrage des centres de delaunay en dehors du mask
├── Trace_Axe_Median.m           % Tracé de l'axe médian à partir des centres delaunay
├── main_Partie_4.m              % Script principal partie 4
├── viff.xy                      % Fichier contenant les correspondances multi-vues 2D
├── donnees.mat                  % Points reconstruits et autres variables du projet
├── mask.mat                     % Masques binaires des 36 images
├── dino_Ps.mat                  % Matrices de projection des caméras (P{i})
├── Simulation/                  % Résultats de simulations
├── images/                      % Images source (viff.00X.ppm)
├── README.md                    % Documentation du projet
```

3. Pipeline général

1. **Chargement des images** et conversion en espace Lab
2. **Initialisation des centres** de superpixels espacés régulièrement
3. **Déplacement vers zones de faibles gradient**
4. **Segmentation Superpixels par itérations SLIC**
5. **Optimisation par connexité** (fusion de petits superpixels)
6. **Binarisation selon la couleur et la compacité**
7. **Récupération du contour et des centres de delaunay**
8. **Filtrage des centres de delaunay**
9. **Tracé de l'axe médian**
10. **Reconstruction des points 3D**
11. **Tétraédrisation + filtrage des tétraèdres** hors mask par projection
12. **Affichage du maillage 3D**
13. **Filtrage des faces + Affichage maillage surfacique**

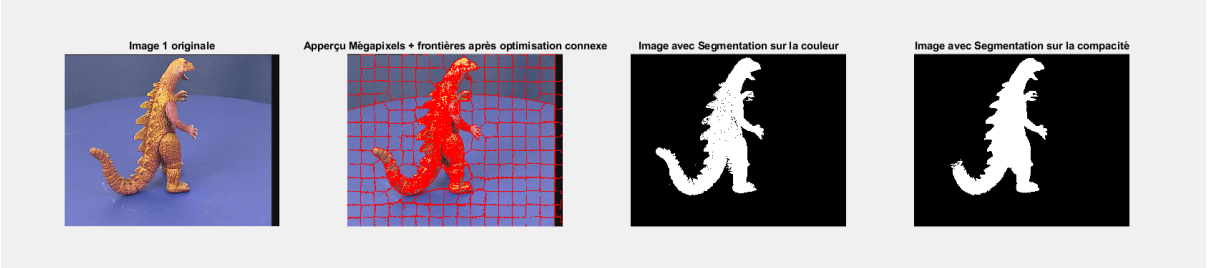
4. Paramètres principaux

Paramètre	Description	Valeur typique	Influence
numSuperpixels	Nombre de superpixels initiaux	200–400	grand = Plus précis, mais plus bruité.
m	Poids couleur vs distance dans SLIC	8 - 12	grand = Superpixels plus compacts spatialement (carré)
max_iterations	Nombre d'itérations de l'algorithme SLIC	100 - 150	petit = mauvaise répartition.

seuil_connexite	Fraction de taille moyenne pour fusion	0.2 - 0.5	grand = beaucoup de superpixels fusionné
seuil_couleur	Seuil absolu sur les paramètres l, a et b	mean(a) mean(b) mean(l)	a & b -> Filtre les couleurs L -> Filtre les pixels trop sombres
seuil_compacite	Seuil absolu sur la compacité des superpixels	150–250	Trop strict = perd des bons superpixel

5. Résultats

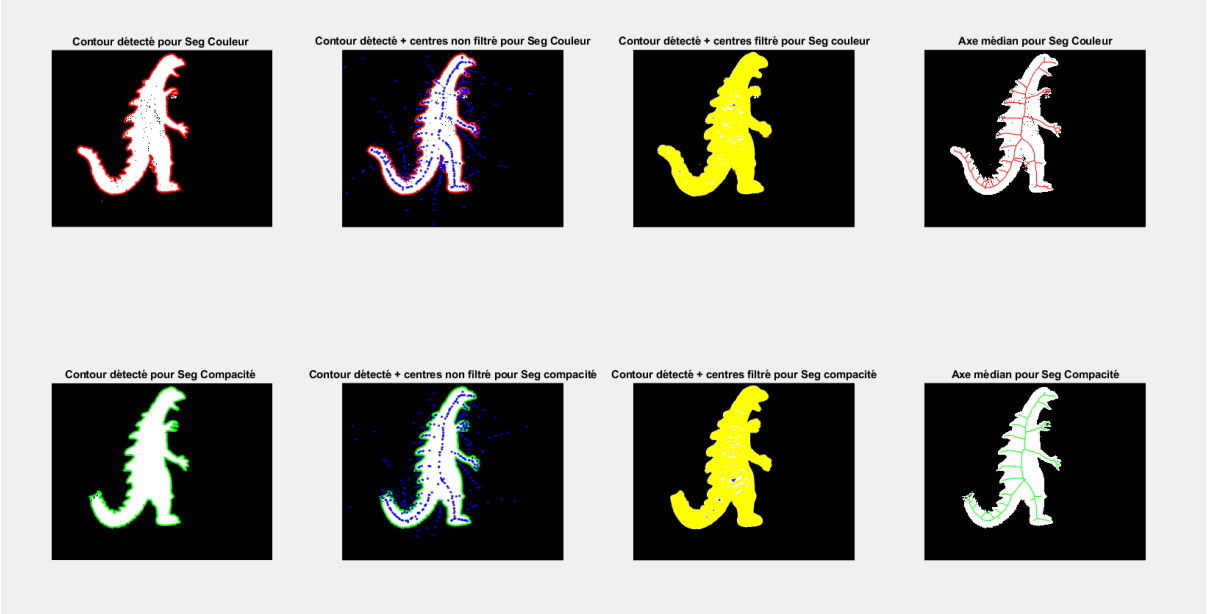
Partie 1 et 2 :



Commentaires : La segmentation couleur demande un réglage des paramètre de seuil et un poids m plus judicieux et fait apparaître quelques points noir au centre du mask qui pourront fausser le filtrage des tetraèdre dans la suite du projet (la projection de certain centre pourtant à l'intérieur du mask pourrait tomber sur un pixel noir).

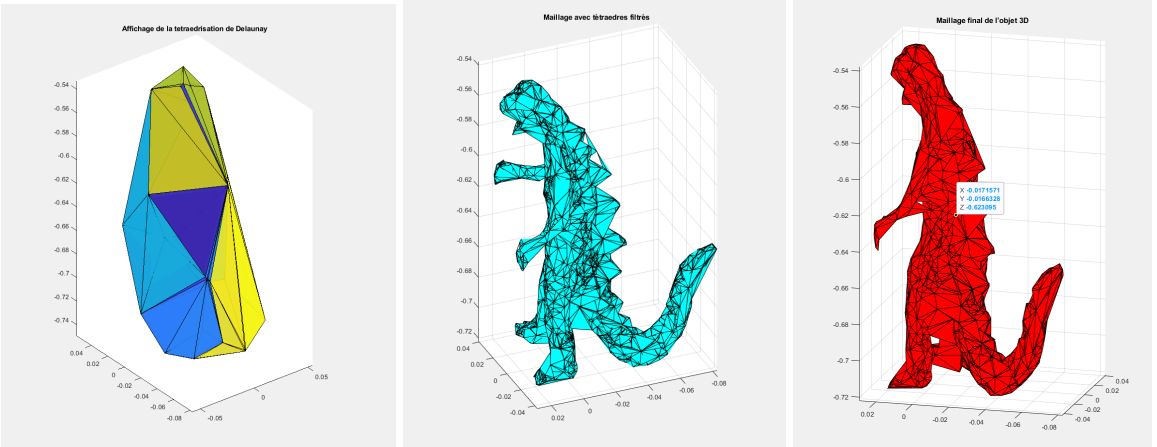
La segmentation sur le critère de compacité des superpixels est plus exploitable pour la suite du projet, bien qu'un superpixel au bout de la queue n'a pas été compris dans le masque.

Partie 3 :



Commentaires : Pour les deux binarisions, nous reconstruisons correctement l'axe médian. La 3ème correspond à l'affichage des centres filtrés ainsi que les cercles circonscrits de la triangularisation de Delaunay. Ces cercles reproduisent bien le masque ce qui confirme la justesse du filtrage des centres.

Partie 4 :



Commentaires : Bien que nous pouvions générer nos 36 mask obtenus par notre méthode personnelle de segmentation + binarisation, nous avons préféré dans un soucis de temps de

nous servir des mask fournit. La première tétraédrisation ne permet pas de reconstruire proprement le maillage 3D car certains tétraèdres sont formés en dehors des masques. Nous devons alors utiliser la matrice de projection fourni dans le TP pour vérifier que chaque tétraèdres appartient bien à chaque masque en 2D par projection. Après filtrage de ces tétraèdres on obtient bien le maillage 3D de notre dino. Pour obtenir le maillage surfacique on supprime ensuite les faces appartenant à plus de 1 tétraèdre (cad qui ne sont pas sur les bords).

Première Tétraédrisation : 16 678 tétraèdres trouvés
Filtrage des tétraèdres : 11 078 tétraèdres restants
Faces dans maillage 3D : 44 296 faces trouvés
Filtrage des faces pour maillage surfacique :3 582 faces restantes

6. Pseudo Algorithmes

A. Déplacement des centres vers les faibles Gradients

Objectif : Ajuster la position des centres des superpixels pour qu'ils se trouvent dans des zones homogènes, en évitant les contours (zones de fort gradient).

Entrées : **L** : Image en niveaux de gris (canal L du Lab), c'est la luminance
centers : Liste des positions initiales des centres
N : Nombre de centres
rows, cols : Dimensions de l'image

Sortie : **centers** : Liste mise à jour des centres, déplacés vers des zones de plus faible gradient

Pseudo-code :

1. **Définir les filtres Sobel** pour détecter les bords horizontaux et verticaux
2. **Calculer le gradient de l'image** : Gx et Gy puis gradMag = $\sqrt{(Gx^2 + Gy^2)}$
3. **Pour chaque centre** k de 1 à N :
    - Récupérer ses coordonnées (x,y)
    - Explorer une fenêtre 3×3 autour de lui
    - Pour chaque pixel dans cette fenêtre :
        - Comparer le gradient local à celui du centre
        - Si le gradient est plus faible, déplacer le centre vers ce pixel
    - Arrondir les coordonnées finales des centres (valeurs entières)
    - Retourner la nouvelle liste de centres ajustés

B. Segmentation superpixels SLIC

Objectif : Segmenter une image en superpixels en regroupant les pixels proches en position (x,y) et en couleur (Lab), via un algorithme de type k-means localisé.

Entrées : **L_channel, a_channel, b_channel** : composantes L, a, b de l'image
centers : liste des centres superpixels
S : distance spatiale (espacement initial entre centres)
numClusters : nombre de superpixels
rows, cols : taille de l'image
max_iterations : nombre d'itérations de l'algorithme
m : pondération entre la distance de couleur et la distance spatiale

Sortie : **labels** : matrice indiquant à quel superpixel appartient chaque pixel
centers : positions et attributs mis à jour des centres après convergence

Pseudo-code :

1. **Initialisation** labels avec -1 et distances à l'infini

2. **Pour chaque itération** (jusqu'à max_iterations) :
    - Pour chaque centre k :
        - Extraire sa position (xk,yk)
        - Définir une **fenêtre carrée locale** de recherche centrée sur le centre (de taille 2S × 2S)
        - Pour tous les pixels dans cette région :
            - Calculer la **distance de couleur**
            - Calculer la **distance spatiale**
            - Combiner les deux en une **distance globale D**
        - Comparer D à la distance actuelle dans distances
            - Si D est plus petite, affecter le pixel au cluster kkk
            - Mettre à jour labels et distances
    - **Après chaque itération** :
        - Pour chaque cluster, recalculer :
            - le **barycentre spatial** des pixels qui lui sont affectés (x, y)
            - la **moyenne des composantes L, a, b**
        - Mettre à jour les coordonnées et la couleur du centre
        - Arrondir les positions (x, y) des centres pour obtenir des entiers

C. Optimisation des superpixels par connexité

Objectif : Améliorer la cohérence des superpixels en fusionnant ceux dont la taille est inférieure à un seuil avec leurs voisins les plus grands.

Entrées : **labels** : matrice indiquant à quel superpixel appartient chaque pixel
rows, cols : dimensions de l'image
numSuperpixels : nombre total de superpixels
seuil_connexite :coefficient pour déterminer la taille minimale acceptée d'un superpixel

Sortie : **labels :** matrice mise à jour après fusion des petits superpixels

Pseudo-code :

1. **Calcul de la taille moyenne** des superpixels et du seuil de taille
2. **Mesurer la taille** de chaque superpixel (nombre de pixels dans chaque superpixels)
3. **Pour chaque petit superpixels** (taille<seuil) :
    - Récupérer ses coordonnées (r,c)
    - Identifier les labels voisins
    - Supprimer le label actuel des voisins pour éviter l'auto-fusion
    - Sélectionner le voisin le plus gros
    - Fusionner : remplacer tous les pixels de l'ancien label par le nouveau
    - Mettre à jour les tailles des superpixels fusionnés

D. Binarisation couleur

Objectif : Identifier les pixels correspondant à l'objet d'intérêt, en déterminant des seuils sur les canaux dans l'espace Lab.

Entrées : **L_channel, a_channel, b_channel** : composantes L, a, b de l'image

Sortie : **masque_binaire_couleur :** matrice binaire de la même taille que l'image, où 1 signifie que le pixel est considéré comme appartenant à l'objet

Critère de couleur : Nous avons premièrement opté pour un seuillage innocent en prenant les 3 moyennes des channels, mais les résultats n'étaient pas concluant. Nous avons donc pris la moyenne sur les canaux a et b qui contiennent l'information colorimétrique, puis avons ajouté une condition sur la luminance car le fond plus sombre sur la partie supérieur de l'image dégradait la détection de contour.

E. Binarisation par compacité

Objectif : Isoler uniquement les régions suffisamment compactes dans une image segmentée en superpixels, en éliminant les formes allongées, dispersées ou irrégulières.

Entrées : **labels :** matrice indiquant à quel superpixel appartient chaque pixel
boundary : image binaire où les contours des superpixels sont à 1
(**seuil_compacite :** est défini dans la fonction mais pourrait être passé en argument)

Sortie : **masque_binaire_compacite :** image logique, avec 1 pour les pixels appartenant à un superpixel compact

Critère de compacité: Nous avons premièrement opté pour un seuil innocent en prenant la médiane de nos valeurs de compacités, mais ce seuil ne permettait une bonne binarisation. Nous avons donc déterminé empiriquement une valeur de 215 qui nous permettait de récupérer tous les bons mégapixels (sauf 1 situé au bout de la queue).

F. Méthode de Delaunay

Objectif : Extraire une approximation de l'axe médian interne d'une forme 2D fermée en utilisant une triangulation de Delaunay sur un échantillon de son contour.

Entrées : **contour1, contour2 :** contours fermés de l'objet (obtenus par bwtraceboundary**)**
pas_delaunay : intervalle d'échantillonnage des points du contour (réduit le nombre de triangles pour éviter des artefacts ou un sur-échantillonnage)

Sortie : **rayon1 / rayon2** : rayons associés à ces cercles
centre_final1 / centre_final2 : coordonnées des centres des cercles circonscrits de chaque triangle de Delaunay

Étapes de la méthode :

1. **Échantillonnage du contour** du mask
2. **Récupération des triangles de delaunay**
3. **Calcul des centres et rayons des cercles circonscrits**
4. **Arrondis des centres** pour récupérer des valeurs entières

G. Filtrage des tétraèdres

Objectif : Conserver uniquement les tétraèdres dont tous les barycentres biaisés sont visibles dans toutes les projections d'images, c'est-à-dire projetés dans une région blanche du masque binaire associé à chaque vue.

Entrées : **X :** points 3D (colonnes)
T.Triangulation : indices des sommets des tétraèdres
P{j} : matrices de projection pour chaque image (caméra)
im_mask : masques binaires (1 = intérieur, 0 = extérieur) pour chaque image

nb_barycentres : nombre de barycentres à tester par tétraèdre
poids : matrices de pondération pour calculer les barycentres biaisés
nb_images : nombre de vues (images)

Sortie : mask_keep : vecteur logique indiquant quels tétraèdres sont conservés (true si à garder)

Pseudo-code :

1. Initialisation des poids des barycentres
2. Calcul des barycentres biaisés (5 barycentres : 1 uniforme et 4 biaisés pour chaque sommet)
3. Vérification de la projection des barycentres
4. Initialisation de mask_keep
5. Parcours des tétraèdres i
    - Pour chaque barycentre k :
        - Pour chaque image j :
            - Projeter le barycentre
            - Arrondir pour avoir une valeur entière
            - Si la projection n'est pas dans le mask -> on rejette le tétraèdre

H. Filtrage des faces

Objectif : À partir d'un ensemble de tétraèdres (maillage 3D), extraire uniquement les faces qui sont visibles à l'extérieur (i.e. non partagées par deux tétraèdres).

Entrées : faces : liste de toutes les faces des tétraèdres, chacune composée de 3 indices (sommets)

Sortie : faces_surface : sous-ensemble des faces qui ne sont présentes qu'une seule fois

Étape de l'algorithme :

1. Initialisation de la matrice face
2. Tri de chaque sommet dans les faces
3. Tri de chaque faces
4. Si 2 lignes de la matrice face sont égale alors supprimer cette face

7. Remarques

- **Passage de RGB à LAB**

Intérêt : Meilleure robustesse à l'éclairage/ ombre et mieux adaptés à la vision humaine.

Le canal L permet de mesurer la luminosité tandis que les canaux a et b contiennent l'information de la couleur pure. Sur notre dataset d'images de dino, on a des zones d'ombres qui sont donc prises en compte pour notre canal L, ce qui n'aurait pas été le cas dans l'espace RGB. Avec nos canaux a et b on peut donc isoler des teintes spécifiques indépendamment de leur luminosité/éclairage.

Étapes d'une conversion RGB en LAB :

1. **Normalisation :**
2. **Passage RGB → XYZ (espace intermédiaire) :** XYZ est un modèle physique de perception

- Formule linéaire :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$ avec M une matrice de transformation dépendant de l'espace de couleur

- Cela donne une description **physiologique** des couleurs.

3. **Passage XYZ → Lab :** Conversion non-linéaire pour rendre l'espace perceptuel

- Limites du TP

Pour concevoir entièrement la modélisation 3D et le maillage surfacique à partir d'un dataset d'images, il nous manque 2 principaux éléments dans ce projet.

Il nous faut pouvoir déterminer la matrice de projection correspondant à ce dataset d'images. De même, nous nous sommes arrêtés à l'extraction de l'axe médian des images 2D, il nous faudrait donc pouvoir faire la reconstruction des points en 3D à partir de cet axe. Ces notions seront abordées en 3ème année.