Université Abdelmalek Essaâdi

Faculté des sciences et techniques de Tanger

Département Génie Informatique

MST MBD

2023/2024

Compte-rendu du Mini-projet 2
CBIR (Modèles 3D)

Encadré par :

Pr. M'hamed AIT KBIR

Réalisé par :

EL ASRI Nacer
SBAIHI Chaymae
EL FALLAH Ayoub

Table des matières

I. L'ARTICLE : 3
La synthèse de l'article : 3
Résumé de l'article : 3
1. INTODUCTION **:** 3
2. Shape similarity search algorithm : 4
2.1 Pose normalization 4
2.2 Parameterized statistics : 5
2.3 Dissimilarity computation : 5
3. Experiments and results : 6
4. Summary and conclusion : 6
II. Implémenter les algorithmes qui calculent et affichent : 8
III. Proposer et implémenter une méthode de réduction du maillage d'un modèle 3D : 10
IV. Utiliser les trois descripteurs 3D proposés par l'article pour mettre au point un système de recherche dans une base des exemples des modèles 3D. 12
V. Conclusion : 17
REFERENCES : 18

I. L’ARTICLE :

La synthèse de l’article :

La recherche sur la similarité des formes de modèles 3D est un domaine en pleine expansion, notamment avec l'augmentation des modèles géométriques en trois dimensions sur internet et dans les industries créatives.

L'article discute d'une méthode proposée pour la recherche de similarité de forme dans les modèles de maillages polygonaux 3D. Cette méthode accepte des maillages triangulaires, y compris ceux comportant des anomalies telles que des polygones dégénérés ou des composants déconnectés.

Les caractéristiques de forme utilisées par l'algorithme sont trois statistiques paramétrées le long des axes principaux d'inertie du modèle : le moment d'inertie, la distance moyenne de la surface par rapport à l'axe et la variance de cette distance. Pour mesurer la dissimilarité entre les vecteurs de caractéristiques, deux distances sont utilisées : la distance euclidienne et la distance de correspondance élastique, cette dernière pouvant prendre en compte les déformations locales de la forme des modèles.

Des expériences ont montré que ces caractéristiques de forme et mesures de distance sont assez efficaces pour récupérer des modèles similaires dans une base de données de modèles VRML(Virtual Reality Modeling Language). Bien que le système de démonstration ne soit pas encore prêt pour des applications dans le monde réel, les composants développés sont prometteurs pour un futur système de recherche et de récupération de similarité de forme.

Résumé de l’article :

1. INTODUCTION :

L'introduction de l'article "Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics" aborde l'importance croissante de la recherche et de la récupération de modèles géométriques tridimensionnels (3D) basée sur le contenu. Avec la prolifération de modèles 3D sur Internet et dans les bases de données locales, la nécessité de développer des technologies efficaces de recherche et de récupération basées sur le contenu pour ces modèles est devenue un enjeu majeur. Alors que la recherche basée sur des annotations textuelles pourrait être une approche possible, elle est limitée par la subjectivité des annotations humaines. Ainsi, il est essentiel de disposer de systèmes de recherche et de récupération basés sur le contenu pour les modèles 3D, en se concentrant sur les caractéristiques intrinsèques des modèles, dont la plus

importante est la forme. L'introduction met en lumière l'importance croissante de la recherche de similarité de formes pour les modèles 3D et souligne le besoin de méthodes efficaces pour y parvenir.

2. Shape similarity search algorithm :

Cette partie décrit l'algorithme de recherche de similarité de formes proposé par les auteurs. Le système prend en entrée des maillages polygonaux 3D et stocke chaque modèle dans une base de données avec un vecteur de caractéristiques précalculé. Le système utilise une approche de requête par exemple, où l'utilisateur fournit un modèle 3D en exemple et demande au système de récupérer les k modèles les plus similaires.
L'algorithme commence par normaliser la position et l'orientation du modèle en utilisant le centre de masse et les axes principaux d'inertie du modèle. Les calculs du centre de masse et du moment d'inertie sont effectués en supposant que chaque surface polygonale a une densité uniforme, et en utilisant une approche de Monte-Carlo pour l'approximation.

Le système utilise trois statistiques discrètement paramétrées le long des axes principaux d'inertie du modèle comme vecteur de caractéristiques : le moment d'inertie, la distance moyenne de la surface par rapport à l'axe, et la variance de la distance de la surface par rapport à l'axe. Pour calculer les statistiques paramétrées le long d'un axe d'inertie, le modèle est subdivisé en tranches le long de l'axe, puis pour chaque tranche, les trois statistiques sont calculées.

Le système calcule une valeur de dissimilarité entre une paire de vecteurs de caractéristiques en utilisant deux méthodes : la distance euclidienne et la distance de correspondance élastique. Cette dernière est destinée à accommoder certaines déformations locales de la forme du modèle, par exemple l'allongement de la partie du torse d'un modèle de cheval.

2.1 Pose normalization

Cette section décrit le processus de normalisation de la taille et de l'orientation du modèle avant l'extraction des caractéristiques de forme. Pour normaliser l'orientation, le système calcule les axes principaux d'inertie. Contrairement à une méthode antérieure qui supposait une masse ponctuelle unitaire située à chaque sommet, les auteurs ont adopté une approche où la masse est supposée être uniformément répartie sur la surface du maillage. Cette approximation est réalisée en plaçant un certain nombre de masses ponctuelles à des emplacements aléatoires sur chaque triangle, le nombre de points étant proportionnel à la surface du triangle. En utilisant ces points de masse sur la surface, la matrice de

covariance est calculée pour déterminer les axes principaux d'inertie du modèle. Les auteurs résolvent également le problème de symétrie des axes principaux en utilisant la distribution des points de masse autour du centre de masse du modèle.

2.2 Parameterized statistics :

Cette partie de l'article parle les statistiques paramétrées utilisées comme vecteur de caractéristiques pour la recherche de similarité de formes. Les auteurs utilisent trois vecteurs de valeurs statistiques discrètement paramétrées le long de l'un des axes principaux d'inertie du modèle : le moment d'inertie, la distance moyenne de la surface par rapport à l'axe, et la variance de la distance de la surface par rapport à l'axe. Les auteurs calculent neuf vecteurs au total, trois pour chaque axe principal d'inertie.

Pour calculer les statistiques paramétrées le long d'un axe d'inertie, le modèle est subdivisé en tranches le long de l'axe, puis pour chaque tranche, les trois statistiques sont calculées. Le moment d'inertie est calculé en utilisant la formule de la somme des carrés des distances des points de masse à l'axe. La distance moyenne de la surface par rapport à l'axe est calculée en utilisant la formule de la somme des distances des points de masse à l'axe divisée par le nombre de points de masse. La variance de la distance de la surface par rapport à l'axe est calculée en utilisant la formule de la somme des carrés des distances des points de masse à l'axe moins le carré de la distance moyenne, le tout divisé par le nombre de points de masse moins un.

Les auteurs concatènent les neuf vecteurs de statistiques paramétrées pour former un vecteur de caractéristiques global pour chaque modèle. Ce vecteur est ensuite utilisé pour calculer la dissimilarité entre les paires de modèles en utilisant la distance euclidienne ou la distance de correspondance élastique.

2.3 Dissimilarity computation :

La section traite le calcul de la dissimilarité entre les modèles en utilisant deux mesures de distance : la distance euclidienne et la distance de correspondance élastique.

La dissimilarité entre deux modèles est calculée comme la distance entre leurs vecteurs de caractéristiques. Pour la distance euclidienne, la dissimilarité est calculée en prenant la racine carrée de la somme des carrés des différences entre les éléments correspondants des deux vecteurs. En revanche, la distance de correspondance élastique est calculée en utilisant la programmation dynamique

pour trouver la correspondance optimale entre les points des deux modèles, prenant en compte les déformations locales de la forme du modèle.

Les auteurs ont expérimenté ces deux méthodes de calcul de la dissimilarité pour évaluer leur efficacité dans la recherche de modèles similaires. Les résultats ont montré que la distance de correspondance élastique était plus efficace pour récupérer des modèles similaires, en particulier pour les modèles ayant subi des déformations locales.

3. Experiments and results :

Ici on a la description des expériences menées pour évaluer l'efficacité de la méthode proposée pour la recherche de similarité de formes.

Les auteurs ont utilisé une base de données de 261 modèles 3D pour tester leur méthode. Ils ont effectué des expériences pour récupérer des modèles similaires à quatre modèles de référence différents, en utilisant la distance euclidienne et la distance de correspondance élastique pour calculer la dissimilarité entre les modèles.

Les résultats ont montré que la méthode proposée était efficace pour récupérer des modèles similaires, en particulier pour les modèles ayant une certaine forme de symétrie de rotation. Cependant, pour les modèles ayant des formes plus complexes, tels qu'un dragon chinois ou une figure humaine dans une configuration corporelle contorsionnée, la méthode n'a pas bien fonctionné.

Les auteurs ont également constaté que la distance de correspondance élastique était plus efficace que la distance euclidienne pour récupérer des modèles similaires, en particulier pour les modèles ayant subi des déformations locales.

En résumé, la section 3 décrit les expériences menées pour évaluer l'efficacité de la méthode proposée pour la recherche de similarité de formes, en utilisant une base de données de 261 modèles 3D. Les résultats ont montré que la méthode était efficace pour récupérer des modèles similaires, en particulier pour les modèles ayant une certaine forme de symétrie de rotation, et que la distance de correspondance élastique était plus efficace que la distance euclidienne pour récupérer des modèles similaires.

4. Summary and conclusion :

Cette partie de l’article présente un résumé et une conclusion de l'étude.

Les auteurs ont proposé une méthode pour la recherche de similarité de forme de modèles polygonaux 3D en utilisant des statistiques discrètement paramétrées le

long des axes principaux d'inertie du modèle pour représenter les caractéristiques de forme. Ils ont également utilisé deux mesures de distance pour calculer la dissimilarité entre les paires de vecteurs de caractéristiques : la distance euclidienne et la distance de correspondance élastique.

Les résultats des expériences ont montré que la méthode proposée était efficace pour récupérer des modèles ayant une forme similaire à partir d'une base de données de modèles VRML. Cependant, la méthode n'a pas bien fonctionné pour les modèles ayant des formes complexes, telles qu'un dragon chinois ou une figure humaine dans une configuration corporelle contorsionnée.

En conclusion, les auteurs ont souligné la nécessité de développer davantage les caractéristiques de forme pour une recherche de similarité de forme efficace, ainsi que la nécessité de développer des mécanismes pour accommoder les préférences et les goûts des utilisateurs dans la récupération de formes. Ils ont également noté la nécessité de trouver un ensemble de données standard et une méthode d'évaluation des performances pour comparer objectivement les algorithmes de recherche de similarité de forme développés.

II. Implémenter les algorithmes qui calculent et affichent :

- o Les axes principaux d'inertie;
- o Les moments au long du premier axe principal;
- o La distance moyenne des faces d'un modèle au premier axe principal;
- o La variance de la distance des faces d'un modèle au premier axe principal;

```
import numpy as np

def calculate_principal_axes(model_data):
    covariance_matrix = np.cov(model_data, rowvar=False)
    # Calculer les vecteurs propres et les valeurs propres
    eigenvalues, eigenvectors = np.linalg.eigh(covariance_matrix)
    # Trier les vecteurs propres par ordre décroissant des valeurs propres
    sorted_indices = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[sorted_indices]
    eigenvectors = eigenvectors[:, sorted_indices]
    return eigenvectors, eigenvalues

def calculate_moments_along_axis(model_data, principal_axis):
    projected_data = np.dot(model_data, principal_axis)
    # Calculer les moments le long de l'axe principal
    moments = np.mean(projected_data**2)
    return moments

def calculate_mean_distance_to_axis(model_data, principal_axis):
    projected_data = np.dot(model_data, principal_axis)
    # Calculer la distance moyenne au premier axe principal
    mean_distance = np.mean(np.abs(projected_data))
    return mean_distance

def calculate_variance_of_distance_to_axis(model_data, principal_axis):
    projected_data = np.dot(model_data, principal_axis)
    # Calculer la variance de la distance au premier axe principal
    variance_distance = np.var(np.abs(projected_data))
    return variance_distance

# Exemple d'utilisation
model_data = np.random.rand(100, 3)
principal_axes, eigenvalues = calculate_principal_axes(model_data)

first_principal_axis = principal_axes[:, 0]
moments_along_axis = calculate_moments_along_axis(model_data,
first_principal_axis)
```

```
mean_distance = calculate_mean_distance_to_axis(model_data,
first_principal_axis)
variance_distance = calculate_variance_of_distance_to_axis(model_data,
first_principal_axis)

print("Premier axe principal:", first_principal_axis)
print("Moments le long du premier axe principal:", moments_along_axis)
print("Distance moyenne au premier axe principal:", mean_distance)
print("Variance de la distance au premier axe principal:", variance_distance)
```

Output :

```
Premier axe principal: [-0.7912724  -0.09401295  0.6041933 ]
Moments le long du premier axe principal: 0.11252693073628257
Distance moyenne au premier axe principal: 0.28282861620146227
Variance de la distance au premier axe principal: 0.03253490459384851
```

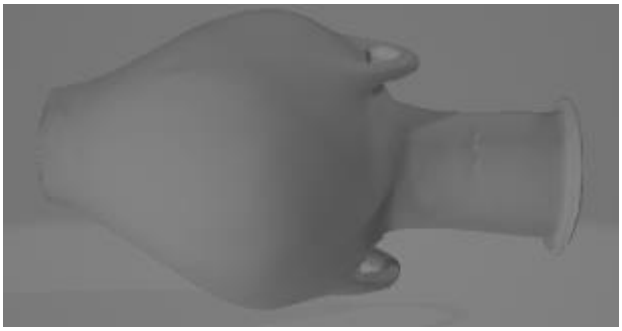
III. Proposer et implémenter une méthode de réduction du maillage d'un modèle 3D :

Pour implémenter une méthode de réduction du maillage d'un modèle 3D, on va utiliser une **technique de simplification de maillage telle que la décimation de maillage**. La décimation de maillage consiste à **réduire le nombre de triangles ou de faces dans le maillage d'un modèle** 3D tout en préservant au mieux sa forme et ses caractéristiques visuelles. Voici une approche générale pour implémenter cette méthode :

1. **Chargement du modèle 3D :**
2. **Algorithme de simplification (simplify_mesh) :** L'algorithme de simplification fonctionne en itérant jusqu'à ce que le nombre de triangles atteigne ou descende en dessous du nombre cible spécifié (target_triangles).
    - Pour chaque itération, l'algorithme calcule l'aire de chaque face du maillage.
    - Il identifie ensuite la face avec la plus petite aire et la supprime du maillage.
    - Les coordonnées des vertices sont mises à jour en conséquence, et les indices des faces sont ajustés pour refléter les nouvelles positions des vertices.
    - Ces étapes sont répétées jusqu'à ce que le nombre de triangles atteigne le nombre cible.
3. **Sélection de la face à supprimer :** La sélection de la face à supprimer se fait en fonction de l'aire de la face. La face avec la plus petite aire est considérée comme étant la moins importante, et donc elle est retirée en premier. Cela vise à conserver les parties du modèle qui contribuent le plus à la forme globale.
4. **Mise à jour des coordonnées des vertices et des indices des faces :** Après la suppression d'une face, les coordonnées des vertices sont ajustées, et les indices des faces sont mis à jour pour refléter la nouvelle configuration des vertices.
5. **Sauvegarde du modèle simplifié :** Une fois la simplification terminée, le modèle simplifié est sauvegardé dans un nouveau fichier OBJ.

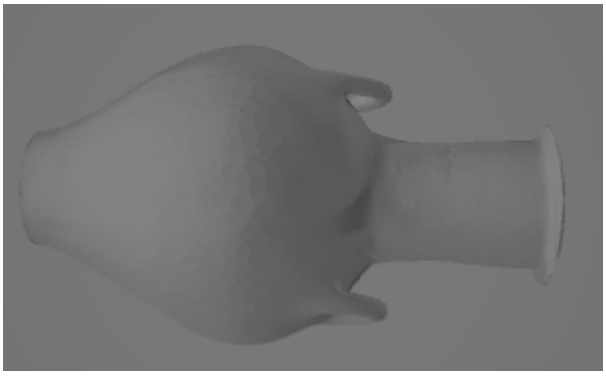
Exemple d'application :

On a l'objet suivant qui contient jusqu'à 91206 lignes dans fichier.obj



```
91204   f 1788/1787/1799 17401/21629/1
91205   f 17402/21630/17518 1750/1749/
91206   f 1750/1749/1761 17402/21630/1
91207
```

Apres application d'une méthode de réduction on obtient même objet avec 50400 lignes dans fichier.obj :

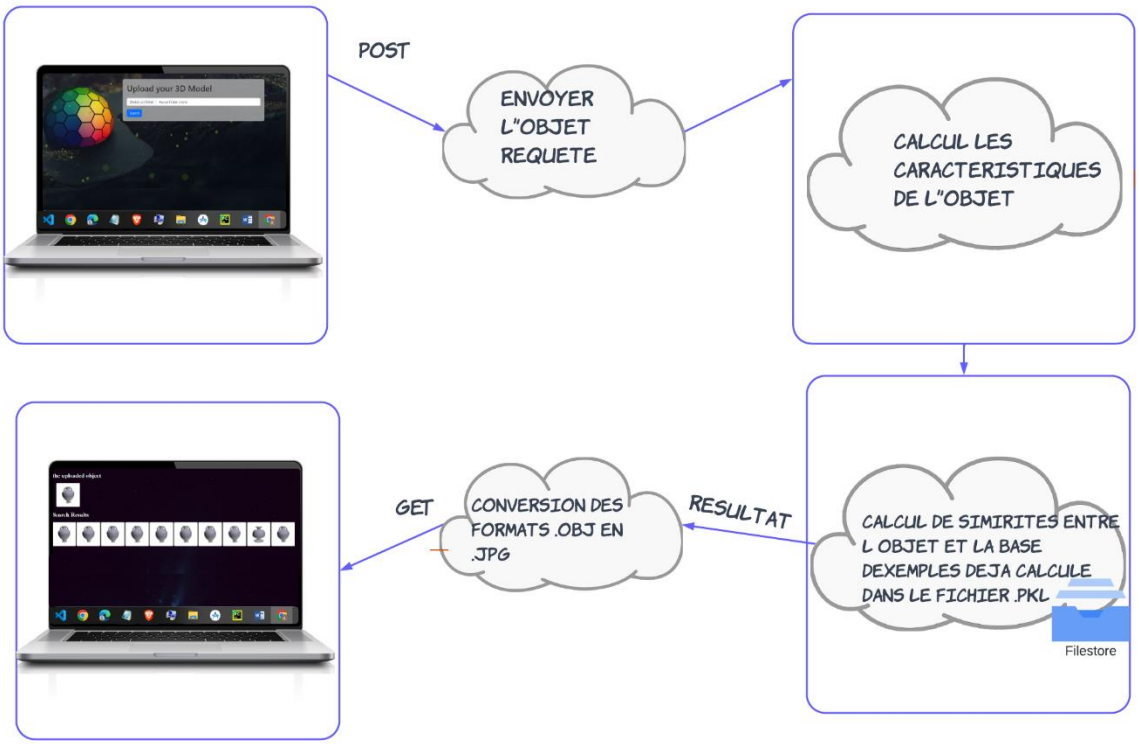


```
50399   f 13149 17303 17347
50400   f 1750 17402 17403 17357
50401
```

IV. Utiliser les trois descripteurs 3D proposés par l'article pour mettre au point un système de recherche dans une base des exemples des modèles 3D.

Dans cette section, nous explorerons un exemple concret de l'implémentation des trois descripteurs. Cette démarche sera divisée en trois volets : la conception, le backend et le frontend.

- ✓ **Partie Conception** :



- ✓ **Partie Backend** :

Étape 1 : Extraction des caractéristiques des objets 3D dans la base d'exemples à l'aide des trois descripteurs et enregistrements dans « descriptors_database.pk pour la sérialisation des données :

```
descriptors_database = []
# Définir le vecteur d'axe
axis_vector = np.array([1, 0, 0])
# Liste tous les fichiers .obj dans le dossier
fichiers_obj = [f for f in os.listdir(dossier_obj) if f.endswith('.obj')]
# Itération sur chaque fichier .obj
for obj_file in fichiers_obj:
    # Construire le chemin complet vers le fichier
```

```
    chemin_fichier = os.path.join(dossier_obj, obj_file)

    # Charger le maillage 3D à partir du fichier .obj
    mesh = trimesh.load(chemin_fichier)

    # Obtenir les vertices et les faces du maillage
    vertices = mesh.vertices
    faces = mesh.faces

    # Calculer le moment d'inertie, la distance moyenne et la variance de la
distance
    moment_of_inertia = compute_moment_of_inertia(vertices, faces)
    average_distance = compute_average_distance_from_axis(vertices,
axis_vector)
    variance_distance = compute_variance_of_distance_from_axis(vertices,
axis_vector)

    # Ajouter les descripteurs à la base de données
    descriptors_database.append({
        'file': obj_file,
        'moment_of_inertia': moment_of_inertia,
        'average_distance': average_distance,
        'variance_distance': variance_distance
    })
```

Étape 2 : Récupérer la requête de l'utilisateur et procéder au calcul de ses caractéristiques

```
def search():
    if request.method == 'POST':
        user_query_file = request.files['file']
        if user_query_file:
            # Save the user's uploaded file
            user_query_file.save(os.path.join('uploads',
user_query_file.filename))

            # Load descriptors from the file
            descriptors_file_path = 'descriptors_database.pkl'
            axis_vector = np.array([1, 0, 0])

            if os.path.exists(descriptors_file_path):
                with open(descriptors_file_path, 'rb') as file:
                    descriptors_database = pickle.load(file)
            else:
```

```
                return "Descriptors file not found. Please calculate
descriptors first."

            obj_folder_path = '3D Models/All Models'
            obj_files = [file for file in os.listdir(obj_folder_path) if
file.endswith('.obj')]

            user_mesh = trimesh.load(os.path.join('uploads',
user_query_file.filename))
            user_vertices = user_mesh.vertices
            user_faces = user_mesh.faces
            user_moment_of_inertia = compute_moment_of_inertia(user_vertices,
user_faces)
            user_average_distance =
compute_average_distance_from_axis(user_vertices, axis_vector)
            user_variance_distance =
compute_variance_of_distance_from_axis(user_vertices, axis_vector)
```

Étape **3: calcul des similarités**

```
            # Calculate similarities
            similarities = []
            for descriptor in descriptors_database:
                moment_of_inertia_sim = np.abs(descriptor[0] -
user_moment_of_inertia).sum()
                average_distance_sim = np.abs(descriptor[1] -
user_average_distance)
                variance_distance_sim = np.abs(descriptor[2] -
user_variance_distance)
                similarity = moment_of_inertia_sim + average_distance_sim +
variance_distance_sim
                similarities.append(similarity)
```

Étape 4 **: obtention des top 10 plus similaires à notre objet**

```
 # Sort the results by similarity
            results = [x for _, x in sorted(zip(similarities, obj_files),
key=lambda pair: pair[0])]

            return render_template('results.html', results=results[:10])
```

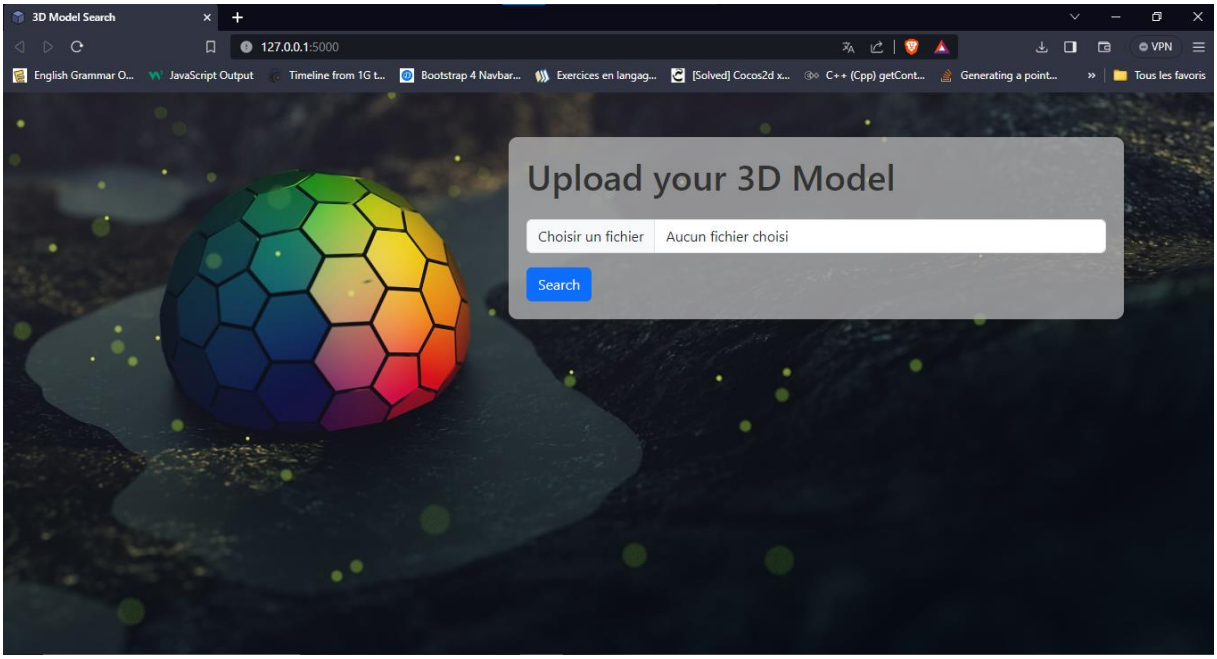
```
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI s
erver instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 682-611-804
127.0.0.1 - - [19/Jan/2024 18:48:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Jan/2024 18:48:09] "GET /static/bg.jpg HTTP/1.1" 304 -
127.0.0.1 - - [19/Jan/2024 18:48:09] "GET /static/cube.png HTTP/1.1" 304 -
127.0.0.1 - - [19/Jan/2024 18:48:09] "GET /static/cube.png HTTP/1.1" 304 -
```

✓ Partie Utilisateur (Frontend)

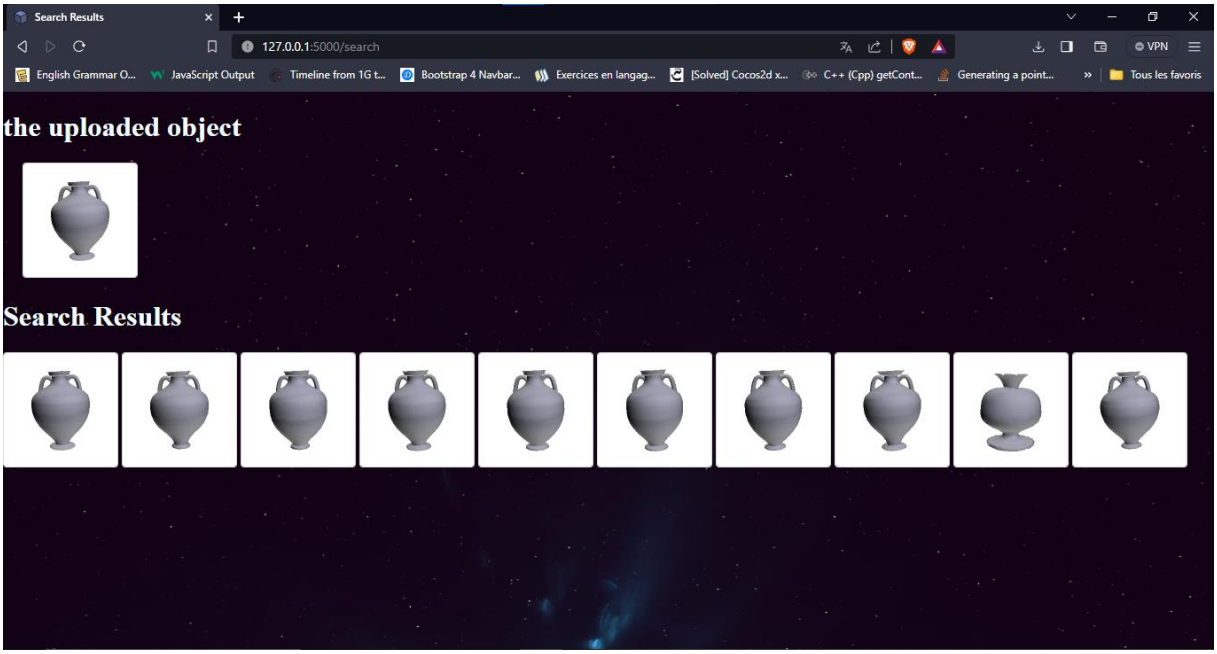
Cette section comporte deux éléments fondamentaux : une interface dédiée à la recherche et une autre dédiée aux résultats.

Interface Rechreche :

Dans cette interface , Notre utilisateur doit importer un object 3D (fichier de type .obj)



Interface Resultants:



V. Conclusion :

L'objectif de ce devoir était d'implémenter les fonctionnalités de base d'un système d'indexation et de recherche par le contenu dans une base d'exemples de modèles 3D. Cela impliquait le calcul des descripteurs de formes liés à un modèle, ainsi que la mesure de similarité par rapport aux modèles de la base d'exemples.

Pour atteindre cet objectif, les tâches suivantes ont été réalisées :

1. Synthèse de l'article en attachement : Une synthèse de l'article "Shape-Similarity Search of Three-Dimensional Models Using Parameterized Statistics" a été effectuée pour comprendre les concepts et les méthodes de recherche de similarité de formes pour les modèles 3D.

2. Implémentation des algorithmes de calcul et d'affichage des caractéristiques de forme : Les algorithmes pour calculer et afficher les axes principaux d'inertie, les moments le long du premier axe principal, la distance moyenne des faces d'un modèle au premier axe principal, et la variance de la distance des faces d'un modèle au premier axe principal ont été mis en œuvre.

3. Proposition et implémentation d'une méthode de réduction du maillage d'un modèle 3D : Une méthode de réduction du maillage d'un modèle 3D a été proposée et implémentée pour simplifier les modèles tout en préservant leurs caractéristiques importantes.

4. Utilisation des descripteurs 3D proposés par l'article pour mettre au point un système de recherche dans une base d'exemples de modèles 3D : Les descripteurs 3D proposés par l'article ont été utilisés pour développer un système de recherche dans une base d'exemples de modèles 3D, en se basant sur les concepts et les méthodes présentés dans l'article.

En conclusion, ce devoir nous permis de mettre en œuvre les concepts et les méthodes de recherche de similarité de formes pour les modèles 3D, en se basant sur l'article fourni. Il a également permis de développer des fonctionnalités de base pour un système de recherche par le contenu dans une base d'exemples de modèles 3D, en mettant l'accent sur le calcul des descripteurs de formes et la mesure de similarité.

REFERENCES :

Article

http://www.ipet.gr/~akoutsou/benchmark/

https://stackoverflow.com/

https://chat.openai.com/

https://www.youtube.com/

https://3dviewer.net/