Rappor

Fait par : **EL MEZIANE Chaïma**

OHMAD Abdessamade

Encadré par Pr : **M'hamed AIT KBIR**

Contexte

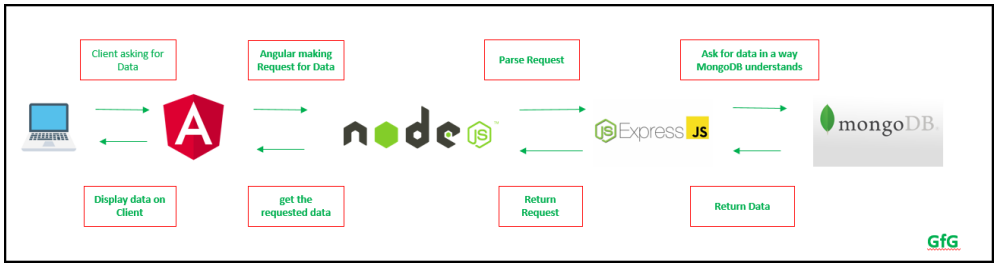
Le but de ce projet est de développer les fonctionnalités essentielles d'un système d'indexation et de recherche par le contenu dans une base de modèles 3D. Cela inclut le calcul des descripteurs de forme pour un modèle requête, suivi de la mesure de similarité avec les modèles de la base, afin d'identifier et de retourner ceux dont la forme est la plus proche. Il s'agit ici d'utiliser les descripteurs basés sur des transformations mathématiques permettent de représenter les modèles 3D dans des espaces de caractéristiques invariants, souvent plus compacts et efficaces pour la comparaison ou la recherche dans des bases de modèles 3D.

Architecture du système

L'architecture du système repose sur une conception modulaire et intégrée combinant des technologies modernes pour assurer une expérience utilisateur fluide et des performances optimales. Elle est structurée comme suit :

- Backend : Flask (microframework Python) avec l'extension Flask-RESTful pour la gestion des API.

- Frontend : MEAN Stack (MongoDB, Express, Angular, Node.js) pour l'interface utilisateur, incluant l'authentification et la gestion des images.



- Postman : Documentation et test des API.

Données

L'existence de jeux de données de référence est essentielle pour évaluer et comparer les méthodes de recherche basées sur le contenu. Bien qu'il n'existe pas de norme pour les benchmarks en correspondance de formes 3D, plusieurs jeux de données ont été proposés par des équipes de recherche et sont librement accessibles. Pour évaluer nos descripteurs de forme spécifiques à la poterie, nous avons créé un ensemble de modèles 3D polygonaux de vases.

Le dataset actuel comprend 1012 modèles 3D numérisés, modélisés manuellement ou générés semi-automatiquement via un générateur aléatoire. Ces modèles sont classés dans plusieurs catégories de formes de vases :

- **Poterie antique grecque :** Alabastron, Amphora, Hydria, Kantharos, Lekythos, Psykter, etc.
- **Poterie amérindienne :** Jar, Effigy, Bowl, Bottle, etc.
- **Poterie moderne et autres catégories.**

La classification actuelle a été effectuée avec l'aide du département du patrimoine culturel de l'ILSP/Athena Research Centre, selon un niveau sémantique orienté archéologie. Une classification orientée sur la forme sera présentée prochainement.

Les modèles 3D sont enregistrés au format Wavefront OBJ, accompagnés d'une vignette en format JPEG sans informations de texture. Un fichier texte et un fichier Excel sont également fournis, associant chaque nom de fichier à une classe, facilitant ainsi le calcul des métriques de performance.

Fonctionnalités Implémentées

Backend Flask

Classe FeatureExtractor3d

Cette classe permet d'extraire des descripteurs pour la recherche de similarité.

- Normalisation du maillage

Avant d'extraire les descripteurs, les modèles 3D sont normalisés pour garantir l'invariance à la translation, rotation et mise à l'échelle.

Mise à l'échelle :

- Extraction des descripteurs

a) Coefficients de Fourier 3D

La transformée de Fourier 3D permet de représenter un modèle en termes de ses composantes fréquentielles.

La transformée de Fourier discrète 3D est donnée par :

$$F(u,v,w) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{z=0}^{N-1} f(x,y,z)e^{-2\pi i(ux+vy+wz)/N}$$

Où :

- $f(x,y,z)$ est la valeur de densité des voxels à la position (x,y,z),
- $F(u,v,w)$ est le coefficient de Fourier dans l'espace fréquentiel.

b) Moments de Zernike 3D

Les moments de Zernike exploitent des polynômes orthogonaux pour décrire la structure globale d'un objet 3D.

Les moments de Zernike sont définis comme suit :

$$Z_{nlm} = \iiint_S R_{nl}(r)Y_{lm}(\theta,\phi)f(r,\theta,\phi)dV$$

Où :

- $R_{nl}(r)$ est le polynôme radial de Zernike,
- $Y_{lm}(\theta,\phi)$ est l'harmonique sphérique,
- $f(r,\theta,\phi)$ représente la fonction de densité des points du modèle.

Classe ModelSearch

Cette classe permet d'envoyer une requête avec un modèle 3D et d'obtenir les modèles les plus similaires.

- Validation du fichier :

Vérification du format (OBJ uniquement).

Sauvegarde temporaire du fichier.

Extraction des descripteurs avec FeatureExtractor3d.

Calcul de similarité entre le modèle requête et ceux de la base.

- Similarité avec la distance Euclidienne normalisée :

Pour mesurer la similarité entre deux descripteurs A et B, on utilise :

$$S(A,B) = 1 - \frac{\|A - B\|}{\|A\|}$$

où $\|A - B\|$ est la norme Euclidienne de la différence entre les deux vecteurs de caractéristiques.

Retour des n meilleurs résultats.

Classe ComparativeModelSearch

Permet une comparaison avec et sans réduction de maillage.

- Techniques de réduction de maillage :

Vertex Clustering : Cette méthode regroupe les sommets proches en clusters en utilisant l'algorithme K-means. Les nouveaux sommets sont les centres de ces clusters. Cela réduit le nombre total de sommets tout en maintenant une approximation raisonnable de la géométrie initiale.

L'algorithme regroupe les sommets proches en utilisant K-means :

$$V' = \frac{1}{|C|} \sum_{i \in C} V_i$$

où :

- V' est le nouveau sommet après clustering,
- C est un cluster de sommets,
- V_i sont les coordonnées des sommets appartenant à C.

Edge Collapse : La décimation par collapse des arêtes consiste à fusionner deux sommets connectés par une arête, en minimisant l'erreur géométrique.

On minimise l'erreur géométrique Q en fusionnant deux sommets v_1 et v_2 en un nouveau sommet v' :

$$Q(v') = v'^T Q v'$$

où Q est une matrice définissant l'erreur quadratique liée aux sommets fusionnés.

Extraction des descripteurs sur les versions réduites.

Comparaison des similarités entre les versions originales et réduites.

Frontend MEAN

Authentification des utilisateurs

L'authentification des utilisateurs a été implémentée dans le frontend de l'application en utilisant la stack MEAN (MongoDB, Express.js, Angular, Node.js). Voici les étapes principales :

Enregistrement des utilisateurs :

- Les utilisateurs peuvent créer un compte via un formulaire sécurisé. Les données sont transmises au backend Node.js, où les mots de passe sont hachés avant d'être stockés dans MongoDB.

Connexion et génération de JWT :

- Lorsqu'un utilisateur se connecte, ses identifiants sont vérifiés. En cas de succès, un JSON Web Token (JWT) est généré. Ce token inclut les informations nécessaires pour identifier l'utilisateur (comme son ID) et a une durée de validité limitée pour des raisons de sécurité.

Stockage sécurisé du token :

- Le token JWT est envoyé au frontend Angular et stocké dans le localStorage ou les cookies (selon le niveau de sécurité souhaité). Cela permet de maintenir la session de l'utilisateur.

Validation des requêtes via le middleware :

- Chaque requête nécessitant une authentification (comme la gestion des images ou l'accès aux résultats de recherche) passe par un middleware sur le backend Express.js. Ce middleware valide le JWT et permet uniquement aux utilisateurs authentifiés d'accéder aux services.

Protection des routes Angular :

- Sur le frontend, un guard Angular empêche l'accès aux pages protégées si l'utilisateur n'est pas connecté ou si le JWT est expiré.

Déconnexion et gestion des sessions :

- Les utilisateurs peuvent se déconnecter, ce qui supprime le JWT côté client. Les sessions expirées sont automatiquement invalidées grâce à la durée limitée du token.

Gestion des images

La gestion des images dans le système repose sur les opérations CRUD (Create, Read, Update, Delete), avec une interface utilisateur conviviale développée sous Angular et des

fonctionnalités backend robustes utilisant Node.js et MongoDB. Voici les principales fonctionnalités implémentées :

Upload et gestion des images :

- L'utilisateur peut charger une ou plusieurs images via un formulaire d'upload sur l'interface Angular.
- Les images sont transférées au backend via une API REST Express.js et stockées dans un système de fichiers ou une base de données adaptée (comme MongoDB avec GridFS pour les fichiers volumineux).
- Chaque image est associée à des métadonnées (nom, taille, type, catégorie, etc.) qui sont enregistrées dans la base MongoDB pour permettre une gestion et une recherche efficaces.

Recherche par nom d'image:

- Une fonctionnalité de recherche permet de filtrer les images par leur nom. L'utilisateur entre le nom (ou une partie du nom) dans un champ de recherche sur l'interface Angular.
- Le backend effectue une recherche textuelle dans MongoDB pour récupérer les images correspondantes, en renvoyant une liste paginée des résultats.
- Les images trouvées sont affichées sur l'interface avec leurs métadonnées, permettant à l'utilisateur de les visualiser ou d'agir dessus.

Téléchargement et suppression des images :

- Téléchargement : Chaque image listée peut être téléchargée directement en cliquant sur un bouton de téléchargement. Le backend Express.js renvoie le fichier via une réponse HTTP pour que l'utilisateur puisse l'enregistrer localement.
- Suppression : Un utilisateur peut supprimer une image via un bouton dédié. Une requête DELETE est envoyée au backend, qui supprime l'image du système de fichiers et efface ses métadonnées de MongoDB.
- Des mécanismes de confirmation sont intégrés dans l'interface Angular pour éviter toute suppression accidentelle.

Ces fonctionnalités offrent une gestion complète et intuitive des images, essentielle pour la manipulation et l'organisation des données nécessaires au système de recherche par contenu.

Système de recherche

Moteur de recherche basé sur des descripteurs

- L'utilisateur peut téléverser un modèle 3D via l'interface Angular.
- Requête HTTP vers Flask avec le fichier OBJ.
- Le backend extrait les descripteurs et calcule les similarités.
- Récupération des résultats sous forme d'une liste d'images similaires.
- Affichage des résultats sur l'interface Angular avec le pourcentage de similarité.

Interface utilisateur

- Upload de fichier avec aperçu des images.
- Résultats affichés sous forme de galerie interactive.
- Option pour comparer les modèles originaux et réduits.

Simulations et Résultats

Testant par un model 3DMillenium_bowl01.obj, qui est de type Bowl :

Vertex Clustering :

Nombre de sommets réduit de 1585 à 792 (ratio : 49,97%).

Les modèles les plus similaires :

1. 3DMillenium_bowl01.jpg (similitude : 0,8518)

2. London B 675.jpg (similitude : 0,8156)

3. CeramicTechnologies_kwlweb.jpg (similitude : 0,7703)

Edge Collapse :

Nombre de sommets réduit de 1585 à 1065 (ratio : 67,19%).

Les modèles les plus similaires :

1. 3DMillenium_bowl01.jpg (similitude : 0,9677)

2. Coventry Iron Planter.jpg (similitude : 0,7851)

3. London B 675.jpg (similitude : 0,7787)

Étude comparative :

Original vs Vertex Clustering :

- Les similitudes obtenues avec Vertex Clustering sont légèrement inférieures, mais la réduction de maillage améliore significativement l'efficacité computationnelle.

Original vs Edge Collapse :

- Les résultats avec Edge Collapse montrent des similarités proches de celles obtenues avec les modèles originaux, tout en réduisant le nombre de sommets.

Comparaison des méthodes :

- La réduction par Vertex Clustering est plus agressive mais peut affecter la précision.
- Edge Collapse préserve mieux les caractéristiques géométriques tout en offrant une réduction significative.

Conclusion

Le système permet une recherche efficace et rapide de modèles 3D en utilisant des descripteurs mathématiques avancés. La réduction de maillage permet d'améliorer la vitesse sans compromettre significativement la précision.