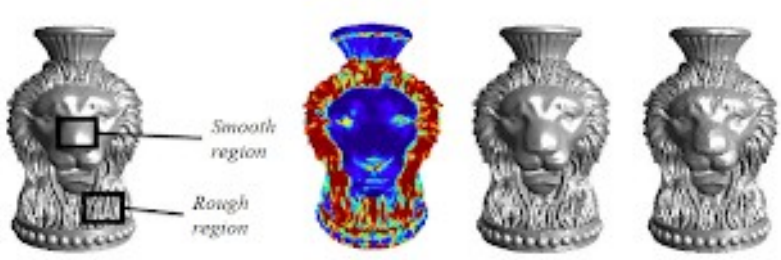
Décimation de maillages par la mesure de la qualité visuelle

proposé par : Romain Raffin romain.raffin@u-bourgogne.fr
Équipe MG, LIB

Les maillages (ou les nuages de points) obtenus par la numérisation (photogrammétrie, laser) sont de taille trop élevés pour permettre une manipulation aisée, d'autant plus lorsque l'interface de gestion est un client léger.
Les méthodes de décimation classiques ne permettent pas (pour la plupart) de conserver des lignes d'arêtes saillantes ou des parties d'objets détaillées. Ces défauts sont visibles à l'œil lors d'un rendu mais difficilement quantifiable en ne s'appuyant que sur des méthodes géométriques. La mesure de la qualité visuelle, avec ou sans a priori, pourrait donc être un critère de qualité de la déformation, voire une fonction intégrable dans l'algorithme de décimation.



Exemple de calcul de saillance

Dans le cadre de ce projet, les étapes à effectuer sont d'implémenter les travaux de G. Turk et P. Lindstrom (*image-driven simplification*) pour décimer un maillage. Puis de comparer les résultats de qualité visuelle sur les mêmes maillages issus de décimations basées sur la géométrie (via Meshlab par exemple). Enfin, proposer une méthode de détection, sur le maillage initial, des zones importantes pour la conservation de la saillance visuelle, ainsi qu'une méthode de décimation permettant de conserver ces zones durant le traitement du maillage et de préparer une transmission progressive (à la mode de H. Hoppe « *Progressive Mesh* » par exemple).

Le calcul de la qualité visuelle se fera selon 2 méthodes : « *Mesh Saliency* » de C.H. Lee et al (2005), et MEP2 de G. Lavoué (2011).

Le développement sera fait en C++ (OpenMesh et QT si besoin), la documentation en Doxygen. On privilégiera des dépendances faibles à l'IHM (chargement de fichier objet, configuration des traitements, export des résultats) pour que ces développements soient aussi utilisables depuis une ligne de commande.

Environnements : C++, OpenGL, Qt 5, OpenMesh ou CGAL ou autre, GIT

Références :

- publication de P. Lindstrom et G. Turk :
https://www.cc.gatech.edu/~turk/my_papers/image_simp_tog2000.pdf
- Mesh Saliency, C.H. Lee (la papier est sur les pages de A. Varshney) :
http://www.cs.umd.edu/gvil/projects/mesh_saliency.shtml
- depôt du papier de Lee 2005 :
https://github.com/climberpi/Mesh-Saliency
- API et soft de calcul de saillance G. Lavoué, LIRIS :
https://perso.liris.cnrs.fr/guillaume.lavoue/rech/soft.html