Crypto-Compression 3D

Compte Rendu 2

Potin Clément, Fournier Romain
Master 2 IMAGINE
Université de montpellier
2021

Problématique

L'objectif est de développer un algorithme qui, de manière conjointe, compresse avec ou sans pertes et chiffre un objet 3D. Les performances de cette méthode seront ensuite mesurées en termes de taux de compression et de qualité de reconstruction de l'objet 3D.

Approche

La compression 3D semble passer par deux "branches". La première, la quantification des vecteurs représentant les points d'un mesh. La seconde, une représentation efficace de la connexité de notre mesh.

Compression des coordonnées

Pour les positions des points, une quantification en considérant une grille régulière sur la boîte englobante de notre mesh permets une représentation inexacte, mais très peu coûteuse en mémoire. Si on considère qu'une position est représentée par 3 flottants, une position en 3D nécessite $3 \cdot 32\ bits = 96\ bits$. Si on considère une grille régulière composée de 2^k unités de côté, il nous faudra $3 \cdot k\ bits$ pour représenter notre position. Ainsi, Jarek Rossignac propose une représentation des positions [6] sur une grille de $2^{10} = 1024$ unités de côté, définie sur la boîte englobante de notre mesh. Nous nous retrouvons donc avec cette quantification à représenter une position sur $30\ bits$, pour un taux de compression de $\delta = \frac{96}{30} = 3,2$.

Une autre approche décrite par Michael Deering [3] s'appuie sur le fait que les coordonnées flottantes peuvent toutes avoir un exposant identique. En effet, un nombre flottant permet de représenter des longueurs qui vont du rayon de l'univers observable au rayon d'un proton. Cependant, si on représente notre mesh sur une échelle bien définie, nous n'aurons pas à redéfinir l'exposant pour chaque coordonnée, la donnée se trouvant en réalité dans la mantisse de notre nombre flottant. ainsi, nous pouvons représenter les positions sur $3 \cdot 24\ bits$ au lieu de $3 \cdot 32\ bits$, offrant ici un taux de compression de $\delta = \frac{32}{24} = 1,5$. Cela semble d'ailleurs analogue au fait de représenter les données sur une grille régulière de 2^{24} unités de côté.

Enfin, en plus de la quantification, une prédiction peut être réalisée afin de ne nous retrouver qu'avec des vecteurs de corrections. En effet, si nous représentons nos points par la correction d'une prédiction, nous pouvons, à condition que notre prédiction soit bonne, placer plus précisément notre point pour autant de donnée, ou bien, et c'est là que cette méthode est intéressante, réduire encore notre empreinte en mémoire.

Notre approche

Nous devrions donc, en quantifiant nos positions sur une grille régulière de taille 2^k, pouvoir d'une part paramétrer notre ratio de compression/distorsion et, d'autre part, du fait que les vecteurs de corrections sur une grille régulière seront entiers, nous retrouver avec beaucoup de doublons, et donc faciliter le travail de notre compresseur entropique.

Compression des Normales

Si nous représentons une normale par 3 flottants, $3 * 32\,bits = 96\,bits$, nous pouvons représenter 2^{96} vecteurs différents. Si nous utilisons 96 $bits$ pour représenter l'index d'une normale parmi 2^{96} réparties équitablement sur une sphère, nous nous retrouverons avec une normale toutes les $2^{-46}\,radians$ [3]. Un tel niveau de précision ne sera en général pas nécessaire. En effet, d'après des tests empiriques [3], un set de 100 000 normales réparties sur une sphère est une approximation plus que suffisante pour représenter les normales d'un mesh. Ainsi, nous pouvons représenter une normale par un index sur 17 $bits$ ($2^{17} = 131\,072 > 100\,000$).

Notre approche

Nous souhaitons donc encoder nos normales sur 17 $bits$ au lieu de 96 $bits$, ce qui devrait nous donner un taux de compression de $\delta = \frac{96}{17} \simeq 5.65$. Les normales pourront être reconstruites soit en gardant un dictionnaire de vecteurs en mémoire, soit sur commande à l'exécution de la décompression.

Compression du Maillage

Nous nous intéresserons ici aux maillages représentés par des triangles, car tout maillage n-gons peut être réduit à un maillage triangulaire. Un triangle peut être intuitivement décrit par ses 3 sommets. Si nous représentons chaque sommet par ses coordonnées et ses normales, un triangle prendrait au maximum $3 \cdot (96\,bits + 96\,bits) = 576\,bits$ pour être représenté. Même en utilisant les représentations proposées plus haut, nous aurions $3 \cdot (30\,bits + 17\,bits) = 141\,bits$. Nous pouvons aussi indexer les sommets, sur 32 $bits$, garantissant un gain de place si l'on référence en moyenne $\simeq 1,3$ fois nos sommets.

Cependant, pour ce qui est de la compression de maillage, l'algorithme *Edgebreaker* peut décrire un triangle en seulement 2 $bits$. Bien entendu, cette description sera purement topologique et il faudra y ajouter les données des points, cependant cela représente un très gros gain d'espace, comparé à nos 141 $bits$, si nous souhaitions définir nos triangles 1 à 1.

Notre approche

Étant donné qu'*Edgebreaker* construit les triangles points par point, nous pouvons nous en servir pour effectuer nos prédictions

Avec cet algorithme, nous pouvons nous attendre à compresser un mesh (n nombre de points) en $n \cdot (30\,bits + 17\,bits + 2\,bits) = n \cdot 49\,bits$, contre (si on considère un maillage à 2n triangles) $2n \cdot 3 \cdot (96\,bits + 96\,bits) = n \cdot 1152\,bits$. On pourrait alors s'attendre à un taux de compression de $\delta = \frac{1152}{49} \simeq 23.51$. Bien entendu, ces estimations sont faites "à la louche", on prend ici le pire des cas et un cas que nous n'avons pas encore implémenté. De plus, si nous voulons faire confiance à ces valeurs, il faudrait aussi faire confiance à notre raisonnement, et nous sommes loin d'être experts en compression 3D. La réalité devrait être bien moins flatteuse qu'un taux de compression > 20 !

Chiffrement du Maillage

EdgeBreaker produit en sortie une suite de codes décrivant, à partir d'une arête, une topologie. Une première approche au chiffrement serait de mélanger la séquence. Ici, un chiffrement par transposition est important. En effet, *Edgebreaker* peut décrire un triangle en 2 bits car si on considère la séquence de code que nous sort cet algorithme comme une source aléatoire, cette dernière aura en général une entropie $1 \leq H \leq 2$.

De plus, nous pouvons transposer nos coordonnées ou vecteurs de correction. Si nous considérons la séquence composée des coordonnées de tous les vecteurs $X_1, Y_1, Z_1, X_2, Y_2, Z_2 \ldots X_n, Y_n, Z_n$, nous devrions garder la composante 3D de notre objet en transposant ces valeurs. Le résultat devrait être des vecteurs qui ne font pas sens dans la construction de notre objet, mais qui pourraient quand même être visualisés.

Enfin, les normales pourraient donner des indices sur l'orientation des faces adjacentes à un point. Nos normales étant encodées comme des index, on pourrait effectuer un chiffrement par substitution sans que notre entropie ne grandisse trop.

Le chiffrement des normales pourrait s'effectuer au fur et à mesure que l'algorithme *Edgebreaker* avance, la séquence de coordonnées et de codes créée par *Edgebreaker* devra cependant être transposée à la fin de la compression car nous avons d'une part besoin du maillage en clair pour nos prédictions, et d'autre part besoin que la séquence soit entière pour la mélanger.

Lien du Git

Notre travail sera mis à jour au lien suivant :
https://github.com/Romimap/3D-CryptoCompression/

Références

Demos & Softwares

1. Vladimir Agafonkin, “Edgebreaker, the Heart of Google Draco” :
https://observablehq.com/@mourner/edgebreaker-the-heart-of-google-draco

2. Google Draco:
https://github.com/google/draco

Papers

3. Michael Deering, “Geometry Compression”, sun Microsystems, 1995 :
http://web.cse.ohio-state.edu/~shen.94/Su01_888/deering.pdf

4. Chandrajit L Bajaj, Valerio Pascucci, Guozhong Zhuang, “Single Resolution Compression of Arbitrary Triangular Meshes with Properties”, University of Texas, 1997:
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.946&rep=rep1&type=pdf

5. Mike M. Chow, “Optimized Geometry Compression for Real-time Rendering”, Massachusetts institute of Technology, May 1997:
https://www.semanticscholar.org/paper/Optimized-geometry-compression-for-real-time-Chow/39babe9519e6b58bae4350e4f57be86dc45ce6f9

6. Jarek Rossignac, “Edgebreaker: Connectivity compression for triangle meshes”, Georgia Institute of Technology, 1999 :
https://www.cc.gatech.edu/~jarek/papers/EdgeBreaker.pdf

7. Daniel Cohen-Or, David Levin, Offir Remez, “Progressive Compression of Arbitrary Triangular Meshes”, Tel Aviv University, 1999:
https://www.tau.ac.il/~levin/vis99-dco.pdf

8. Jarek Rossignac, Alla Safonova, Andrzej Szymczak, “3D Compression Made Simple: Edgebreaker on a Corner-Table”, Georgia Institute of Technology, 2001:
https://www.researchgate.net/publication/3896746_3D_compression_made_simple_Edgebreaker_with_ZipandWrap_on_a_corner-table

9. Pierre Alliez, Mathieu Desbrun, “Progressive Compression for Lossless Transmission of Triangle Meshes”, University of Southern California, February 2002:

https://www.researchgate.net/publication/2534417_Progressive_Compression_for_Lossless_Transmission_of_Triangle_Meshes

10. Jarek Rossignac, “3D mesh compression”, College of Computing and GVU Center Georgia institute of Technology, January 2003:
https://www.researchgate.net/publication/27521282_3D_Mesh_Compression

11. Esam Elsheh, A. Ben Hamza, "Secret sharing approaches for 3D object encryption", Concordia Institute for Information Systems Engineering, Concordia University, Montréal, QC, Canada, 2011:
https://www.sciencedirect.com/science/article/abs/pii/S095741741100724X

12. In-Ho Lee and Myungjin Cho, "Optical Encryption and Information Authentication of 3D Objects Considering Wireless Channel Characteristics", Department of Electrical, Electronic, and Control Engineering, Hankyong National University, Ansung 456-749, Korea, October 2013:
https://www.osapublishing.org/DirectPDFAccess/766AED72-4C7E-47EE-BB28209C333886A8_276786/josk-17-6-494.pdf

13. Marc Éluard, Yves Maetz, and Gwenaël Doërr, "Geometry-preserving Encryption for 3D Meshes", Technicolor R&D France, November 2013:
https://www.researchgate.net/profile/Gwenael-Doerr/publication/273257218_Geometry-preserving_Encryption_for_3D_Meshes/links/54fc4b660cf2c3f52422a624/Geometry-preserving-Encryption-for-3D-Meshes.pdf

14. Ying Zhou, Lingling Wang, Lieyun Ding, Cheng Zhou, “A 3D model Compression Method for Large Scenes”, Huazhong Univ. of Science and Technology, 2018:
https://www.iaarc.org/publications/fulltext/ISARC2018-Paper207.pdf