Crypto-Compression 3D

Compte Rendu 4

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.

Méthodes d'évaluation

Cette semaine, nous nous sommes intéressés aux possibles méthodes d'évaluation de la qualité de reconstruction de nos objets 3D compressés. Autrement dit, à quel point l'objet en sortie de notre compresseur est différent de l'objet d'origine.

Distance de *Hausdorff*

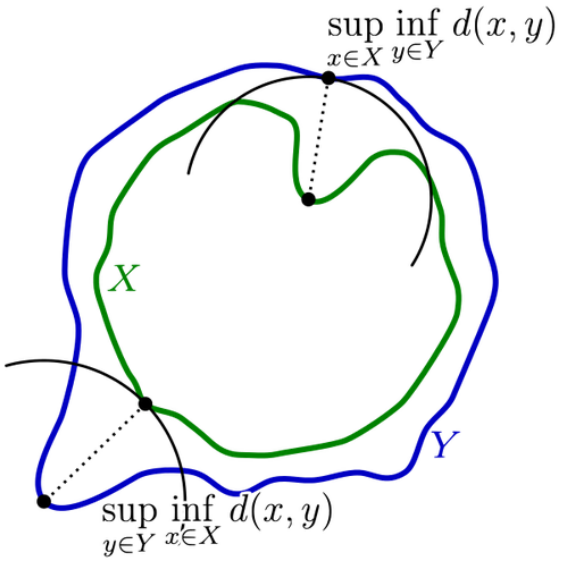
La distance de *HausDorff* est probablement la méthode d'évaluation de cette qualité la plus simple. Elle correspond au maximum de la distance minimale entre chaque point de l'objet d'origine et l'objet compressé. Pour être plus explicite, le principe d'un algorithme calculant la distance de *Hausdorff* est le suivant :

```
Pour chaque point i de l'objet d'origine :
      Pour chaque point j de l'objet compressé :
            distance min (i) = min(distance min (i), distance(i,j))
      distance de Hausdorff = max(distance de Hausdorff, distance min (i))
```

Pseudo-code de calcule de la distance de Hausdorff

La distance entre 2 points est calculée de la façon suivante :

$$Distance(i,j) = \sqrt{(ix - jx)^2 + (iy - jy)^2 + (iz - jz)^2}$$



Distance de Hausdorff en x et y

Donc, 2 objets sont proches l'un de l'autre en termes de distance de *Hausdorff* si chaque point du premier objet est proche d'au moins un point du second objet.

Néanmoins, ce calcul à lui seul ne peut évaluer de façon efficace la qualité de décompression d'objet 3D. En effet, il suffit qu'un point de l'objet décompressé soit fortement décalé pour que la distance de *Hausdorff* soit très élevée, même si tous les autres points ont été décompressés sans erreur (et que leur position est donc identique à celle qu'ils avaient dans l'objet d'origine).

Ce calcul nous servira donc "d'erreur maximale" de la décompression effectuée par nos algorithmes, mais il nous faudra une méthode d'évaluation supplémentaire.

Root-Mean-Square Error (RMSE)

La "Racine de l'Erreur Quadratique Moyenne (REQM)", qu'on notera "RMSE" pour son appellation internationale, est une mesure fréquemment utilisée des différences entre les valeurs prédites et les valeurs observées (réelle, de l'objet d'origine). La RMSE est toujours positive, et une valeur de 0 indiquerait un ajustement parfait aux données. À l'inverse, plus la RMSE est grande, et moins bien les modèles ont été recalés.

La RMSE est la racine carré de la MSE (Mean Squared Error), autrement dit :

$$RMSE = \sqrt{MSE}$$

La MSE est calculée comme suit :

$$MSE = \frac{\sigma^2}{n}$$

Où :

- σ^2 est la variance des distances entre les 2 objets
- n est le nombre de points d'un des deux objets (normalement n est identique pour les deux !)

Les distances entre les points des 2 objets seraient déterminées, comme pour la méthode utilisant la distance de *Hausdorff*, en fonction de la distance minimale de chaque point de l'objet d'origine par rapport à chaque point de l'objet décompressé. Les calculs de variance et de moyenne s'appliqueraient ensuite à ces distances.

La RMSE est une mesure de précision qui sert à comparer les erreurs de différents modèles prédictifs pour un ensemble de données particulier, et non entre différents ensembles de données, car elle dépend de l'échelle. Les comparaisons entre différents jeux de données ne seraient donc pas valides (car la mesure dépend de l'échelle relative des nombres utilisés).

L'idée serait donc d'utiliser une version normalisée de la formule de la RMSE.

Normalized RMSE (NRMSE)

La NRMSE facilite la comparaison entre des ensembles de données ou des modèles à différentes échelles. Bien qu'il n'y ait pas de moyen consistant de normalisation dans la littérature, les choix courants sont :

- La moyenne des distances entre les 2 objets :

$$NRMSE = \frac{RMSE}{\mu}$$

Où μ représente la moyenne des distances entre les 2 objets.

- L'étendue (définie comme la valeur maximale moins la valeur minimale) entre les 2 objets (autrement dit la distance maximale) :

$$NRMSE = \frac{RMSE}{max(distances(objet1, objet2))}$$

- Une autre méthode possible pour faire de la RMSE une mesure de comparaison plus utile consiste à la diviser par l'écart interquartile. Lors de la division du RMSE par l'EI, la valeur normalisée devient moins sensible aux valeurs extrêmes de la variable cible. La formule deviendrait alors :

$$NRMSE = \frac{RMSE}{EI}$$

Où $EI = Q3 - Q1$, avec $Q1 = Q(0.25)$ et $Q3 = Q(0.75)$.

Plus de tests seront à réaliser en pratique pour déterminer l'efficacité de chaque méthode.

Ecriture du fichier

Nous avons également commencé à travailler sur une représentation binaire de notre maillage. Il est mentionné dans les papiers [3, 6], et nous avions l'intention comme expliqué dans le CR2, de représenter nos données sur n bits, n pas forcément un multiple de 8. Une partie du travail réalisé cette semaine à été de stocker et de pouvoir lire un flux binaire dans un fichier. Nous avons comme représentation :

Header (228 bits)			
4 bits	32 bits	96 bits	96 bits
paramètre de quantification k	Nombre de points N	position min de la boîte englobante	position max de la boîte englobante

Positions ($N \cdot$ 30 bits)		
3k bits (30 bits)	…	3k bits (30 bits)
Coordonnée XYZ quantifié du point 1	…	Coordonnée XYZ quantifié du point N

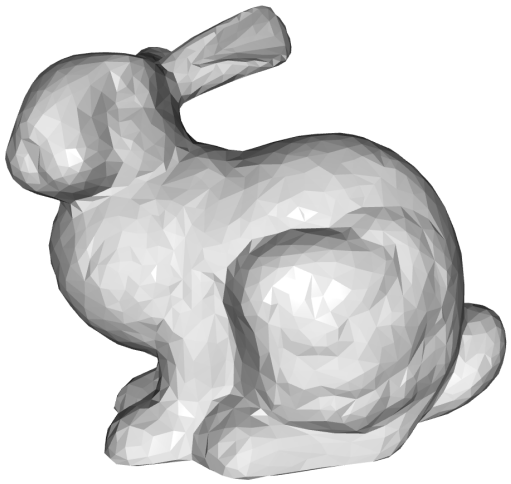
Normales ($N \cdot$ 17 bits)		
17 bits	…	17 bits
Normale quantifiée du point 1	…	Normale quantifiée du point N

Avec cette représentation, nous obtenons déjà un taux de compression intéressant. Si on récupère les lignes “v” et “vn” d’un fichier .obj (les position et les normales des points, nous obtenons les résultats suivants :

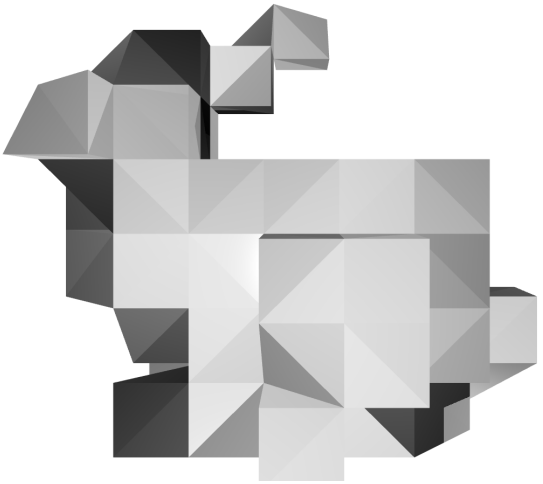
Bunny.obj

fichier quantifié à $k = 10$	52.8 Ko
lignes v (vn non présentes)	120 Ko

$\delta = 120/85.5 \simeq 1.40$



$k = 10$

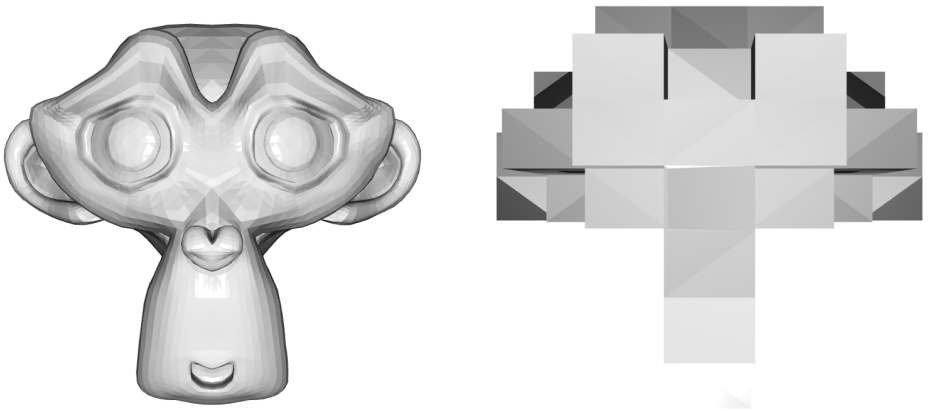


$k = 3$

Suzanne.obj

fichier quantifié à $k = 10$	266.6 Ko
lignes v, vn	486 Ko

$\delta = 486/266.6 \simeq 1.82$



$k = 10$ $k = 3$

Il est intéressant de constater qu'avant même compression entropique, nous utilisons moins de place que pour un format classique (du moins pour ce qui est des positions / normales). Il est important de noter aussi que les deux formats ne sont pas nécessairement comparables, un .obj stockera les positions et les normales séparément, pour ensuite créer les points pendant la construction des faces. Il y aura donc au plus autant de lignes "v", et "vt" qu'il y a de points dans le maillage.

Sur nos deux fichiers de test, il nous reste respectivement 34 Ko et 219.4 Ko pour encoder notre maillage. Avec une technique pouvant atteindre les 1.5 bits par triangles [6], la marge semble correcte. Nous devrions donc pouvoir compresser, ne serais-ce que d'un taux de $1,X$ par le simple fait de quantifier et de représenter nos données comme un flux binaire.

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. Xin Chen, Jingbin Hao, Hao Liu, Zhengtong Han and Shengping Ye, “Research on Similarity Measurements of 3D Models Based on Skeleton Trees”, School of Mechatronic Engineering, China University of Mining and Technology, Daxue Road 1, Xuzhou 221116, China, State Key Laboratory of Materials Forming and

Mould Technology, Huazhong University of Science and Technology, Luoyu Road 1037, Wuhan 430074, China, April 2017: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi91fmtwZj0AhUI2BoKHYcKA1AQFnoECAMQAQ&url=http%3A%2F%2Fwww.mdpi.com%2F2073-431X%2F6%2F2%2F17%2Fpdf-vor&usg=AOvVaw0lfo-8VxxYFuVQTnyCt6JI

15. 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