INF555 : Shape Comparison and Retrieval

Alexandre Matton, Adrien Nivaggioli

September 24, 2018

1 Introduction

Ce projet a été consacré à l'étude de deux méthodes, proches mais différentes, qui permettent d'associer à toute forme 3D une empreinte, sous la forme d'un vecteur de réels de taille raisonnable. Ces empreintes sont déterminées de façon représenter la proximité entre deux formes : elles sont donc censées être proches l'une de l'autre lorsqu'on compare deux formes "proches", et éloignées lorsqu'on compare deux formes très différentes. Ces méthodes se basent sur deux articles de recherche : *Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids* (publié par **M. Reuter, M. Wolter** et **M. Peinecke**, paru en 2005), et *Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape*(publié par **M. Rustamov** en 2007). Nous avons implémenté ces deux méthodes et nous les avons comparées sur les data-sets proposés du projet. Dans ce dossier, vous trouverez dans une première partie notre explication des idées utilisées dans ces papiers, puis quelques détails sur notre implémentation, et enfin les résultats que nous avons obtenus.

2 Résumé des articles de recherche

Tout d'abord, il s'agit de définir ce qu'on entend par proximité entre deux formes, et entre deux empreintes.

- La proximité entre deux empreintes est facile à définir : on prendra ici la norme euclidienne.
- La proximité entre deux formes est plus complexe à cerner, et elle est plus subjective. Concrètement, on voudrait que si deux objets se ressemblent, ou correspondent au même objet animé dont certains membres peuvent avoir bougé, les empreintes soient proches. Ainsi, les empreintes ne doivent pas changer si les distances géodésiques entre tous les points d'un objet sont conservées. Ceci garantit que l'objet peut prendre des postures différentes sans que son empreinte ne change.

On cherche donc avant tout un descripteur invariant par isométrie. Cependant, prendre un descripteur qui ne dépend que des distances géodésiques est une mauvaise idée, car celui-ci sera très influencé par le problème de court-circuit (i.e. des changements locaux de topologie), expliqué dans l'article 2.

Il a été vu en cours que l'opérateur de Laplace-Beltrami défini sur une surface est invariant par isométrie. Les deux méthodes utilisent cet opérateur pour créer des empreintes qui conservent cette propriété. Ces empreintes doivent donc ne provenir que de l'opérateur, c'est pourquoi celles qui sont choisies sont basées sur les valeurs et vecteurs propres de l'opérateur. En plus de respecter la propriété ci-dessus, il est montré dans les articles que ces empreintes sont robustes aux changements locaux de topologies, notamment car ils ne sont pas liés aux distances géodésiques. Ainsi, le problème du court-circuit est par la même résolu.

La différence entre les deux méthodes est la suivante : le premier papier utilise seulement la liste des valeurs propres les plus déterminantes, appelées Shape DNA, pour caractériser une forme, alors que le second papier se sert aussi des vecteurs propres pour calculer pour chaque point sa position (appelée GPS) dans un nouvel espace, qui caractérise ce point intrinsèquement, c'est-à-dire indépendamment de tout repère. Il utilise en suite un histogramme formé à partir des distances entre les GPS de points pris au hasard, pour créer une sorte de "spectre" qui correspond à la forme de l'objet.

Concernant la discrétisation du problème, le but est de créer un laplacien qui essaie de conserver les propriétés de la surface, même si elle est représentée à travers un triangle mesh. C'est pourquoi on évite par exemple d'utiliser un simple laplacien de graphe. On cherche notamment à garder la notion de distance entre les points des triangle meshes.

On implémente de la même manière le laplacien pour les deux méthodes, selon ce qui est expliqué dans le second document. Cela revient à appliquer la méthode des éléments finis, expliquée dans le premier article, à l'ordre 1.

3 Implémentation

3.1 Le calcul des empreintes

Le graphe est représenté de manière à ce qu'on puisse rapidement récupérer les sommets et les faces qu'on veut. Chaque face pointe vers ses 3 sommets, chaque sommet stocke 3 coordonnées et l'indice qui correspond à sa position dans la liste générale des sommets. On utilise aussi une liste v_f pour trouver les faces adjacentes à un sommet. Nous n'avons pas cherché à optimiser la mémoire prise par la manière dont nous stockons les meshes puisqu'en pratique le nombre de sommets n'excède pas 100 000 dans les exemples utilisés et donc la RAM d'un ordinateur actuel peut stocker un nombre assez important de meshes à la fois.

L'étape suivante est de calculer le laplacien. Pour cela, on crée les deux matrices A et W comme sur l'article de recherche. Cependant, le nombre de sommets étant parfois très grand (50 000 voire plus), on ne peut se permettre de stocker en entier ces matrices. En effet, chacune de ses matrices pèserait environ 10 Giga si l'on considère que chaque case occupe 4 octets. Heureusement, le fait que ces matrices soient presque vides règle le problème. Concrètement, on utilise deux représentations de ces matrices, implémentées dans Python. La première (lil_matrix) permet de remplir les matrices presque vides aisément, puis on passe à la deuxième (csc_matrix) qui accélère les calculs auxquels elles sont sujettes (à savoir l'inversion, la multiplication, et les opérations nécessaires pour le calcul des valeurs et vecteurs propres). Le soucis de RAM ne se pose donc plus, et les calculs sont largement accélérés vu qu'on a pas besoin d'itérer sur les nombreuses cases vides des matrices.

Pour trouver les valeurs/vecteurs propres, on utilise les méthodes propres à Python. Selon la documentation, lorsqu'on met l'option pour chercher les plus petites valeurs, l'algorithme utilisé est ARPACK (c'est aussi celui qui est utilisé par le papier de recherche GPS). Cependant, celui-ci est surtout efficace pour trouver les plus grosses valeurs. Nous avons donc opté pour le mode "shift-invert", qui trouve les k valeurs propres près du paramètre qu'on lui donne, ici 1. Il faut donc être sûr que les k valeurs propres les plus petites sont celles qui sont les plus proches de 1, mais ceci a toujours été le cas dans nos tests, puisque généralement la deuxième plus petite valeur propre est déjà supérieure à 1. Le gain de vitesse est remarquable : pour des formes à 10 000 vertices, on passe de 45sec avec ARPACK à 0.2sec avec l'autre méthode, pour des résultats quasiment identiques, en tout cas sur les 40 premières valeurs propres.

L'algorithme des Shape-DNA s'arrête ici, on récupère un nombre k de valeurs propres, qui constitue notre empreinte. Pour l'autre algorithme, on construit à partir de cet ensemble les GPS de chaque point (leur taille est égal au nombre de valeurs propres conservées). Cependant, avant cela, on normalise les vecteurs propres selon le nouveau produit scalaire, qui vient de la discrétisation :

$$< v, w >= v^t A w$$

où A est la matrice diagonale (positive) des aires intervenant dans le calcul de L.

On prend un nombre N de pairs de points du graphe, et pour chaque pair on stocke le GPS des deux points, et la distance entre ces deux points. On rescale les GPS des points stockés de façon à ce qu'ils soient compris entre 0 et 1, et on introduit alors m sphères de rayon, de rayons $1/m$, $2/m$, ... On remplace les GPS des points par le numéro de la plus petite sphère à laquelle ils appartiennent. L'idée est alors de créer $m(m+1)/2$ listes différentes, où la liste $l[i, j]$ (avec $i < j$) contient l'ensemble des distances des pairs dont un point correspond au numéro i, et l'autre au numéro j. Dans chacune de ces $m(m+1)/2$ listes, on rescale les valeurs, et on crée un histogramme avec un nombre nb_cases de cases. La dernière opération à faire consiste à aplatir cet ensemble d'histogrammes, c'est-à-dire à les coller les uns à la suite des autres, pour obtenir une empreinte

de dimension :

$$\frac{m(m+1)}{2} * nb\_cases$$

Les librairies Python que nous avons utilisées sont les suivantes : Numpy et Scipy (et tout particulièrement la sous-libraire sparse de Scipy). Elles offrent des implémentations efficaces des matrices (pleines ou presque vides) et tout un ensemble de méthodes pour les manipuler, dont nous nous sommes permis l'utilisation puisque traiter les sujets de manipulation efficace de matrices presque vides nous éloignerait du sujet. Par ailleurs, les structures de données de Numpy prennent moins de place que celles définies par défaut dans Python, et surtout, les calculs sont beaucoup plus rapides.

A ce sujet, la méthode implémentée permet de calculer les empreintes de presque tous les objets en moins de 30 secondes. Pour les objets avec 5000 sommets, le temps de calcul est d'environ 10s. Chaque étape est chronométrée et s'affiche lorsqu'on instancie un objet "Shape" avec le paramètre "verbose=True".

```
Reading  lion-poses/lion-01.obj ...
Done. Length of faces : 9996 and length of vertices :  5000
Filled in  1.4157941341400146 s. Inversing A and computing L...
Done in :  1.6160309314727783 s
Computing eigenvalues and eigenvectors...
Done in :  0.1337871551513672 s
Start computing D2. Normalization of eigenvectors according to new inner product...
Done in 0.0012271404266357422
Computing GPS...
Done in 0.4284822940826416
Taking random pairs, associating them to patches for D2 computation...
Done in 3.322587490081787
Computing histogram...
Done in 0.23670578002929688
```

Figure 1: Temps mis par l'exécution de l'algorithme sur "lion-01.obj"

3.2 Comparaison des différentes empreintes

Nous disposons donc de deux modèles pour calculer les empreintes de n'importe quelle surface 3D. Cependant, pour chaque modèle, plusieurs paramètres entrent en jeu. L'idée est donc de comparer les modèles entre eux, tout en essayant plusieurs valeurs pour les paramètres de chaque modèle afin de comprendre comment les paramètres influencent les résultats.

- Pour la méthode Shape DNA, le seul paramètre qui entre en jeu est le nombre de valeurs propres conservées. On sait que les résultats sont censés augmenter jusqu'à obtenir un nombre de valeurs propres représentatif de l'objet, puis on s'attend à ce que les résultats ne changent plus trop lorsqu'on dépasse ce nombre.

- Pour la deuxième méthode, d'autres paramètres rentrent aussi en jeu. Le premier est le nombre N de pairs choisies aléatoirement. Il faut que N soit assez élevé pour que la loi des grands nombres soit à peu près respectée. Nous avons fixé N à 100 000 car cela nous paraît être un bon compromis : ça ne ralentit pas trop l'algorithme, et les résultats ne sont plus trop fluctuant. Les deux derniers paramètres sont m (le nombre d'histogrammes de la méthode D2), et nb_cases, le nombre de cases par histogramme. On s'attend à ce que les résultats s'améliorent lorsque ces deux valeurs augmentent puisque le spectre D2 devient plus précis.

Une fois les empreintes calculées, que ce soit par la méthode Shape DNA ou la méthode GPS, nous pouvons clusteriser nos empreintes grâce à l'algorithme de k-means. Cependant, celui-ci ne marche parfois pas très bien lorsque l'on recherche un grand nombre de clusters : en effet, nos différentes empreintes d'objets n'ont pas forcément la même variance, ce qui entraîne de mauvais résultats dans le k-means. Nous en reparlerons dans la section suivante.

Afin de visualiser nos résultats, nous appliquons l'algorithme PCA pour passer en dimension 2 et pouvoir présenter nos empreintes sur un graphique.

4 Résultats obtenus

4.1 Importance du nombre de valeurs propres

Comme attendu, prendre en compte plus de valeurs propres tend à améliorer nos résultats. (voir Figure 2)

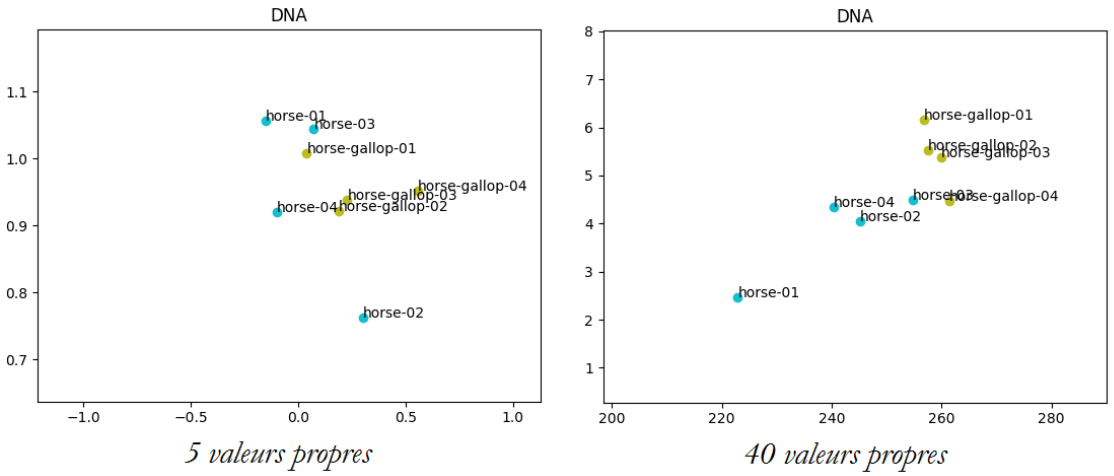

Figure 2: Augmenter le nombre de valeurs propres améliore généralement les résultats...

Nous atteignons assez rapidement une valeur limite, pour laquelle les résultats n'évoluent plus trop, voire diminuent. Il est intéressant de remarquer que les valeurs propres les moins importantes sont celles qui sont les plus élevées, donc qui influencent le plus la "distance" entre deux formes. Ceci explique pourquoi les résultats baissent à partir d'un certain nombre de valeur propres conservées. (Voir Figure 3)

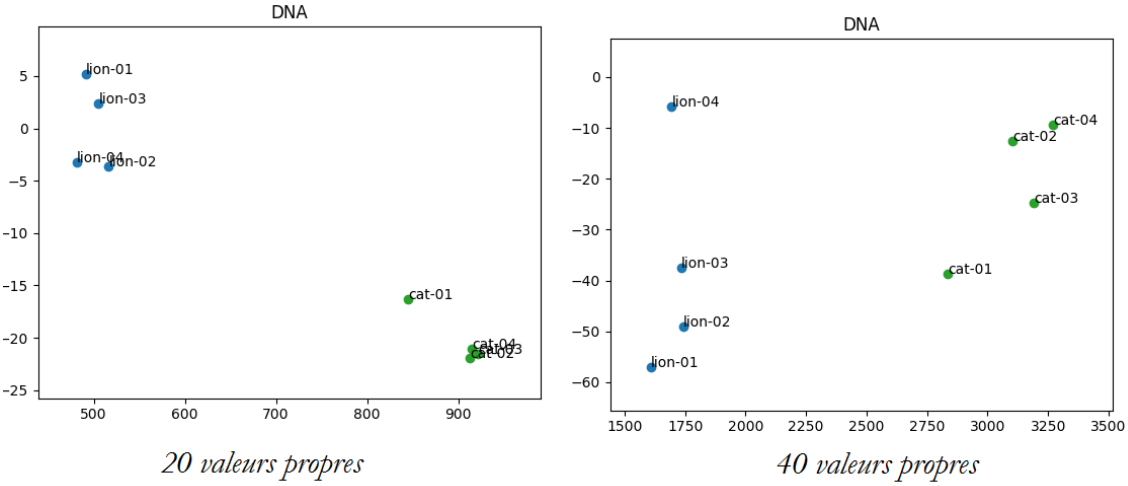

Figure 3: ...Mais cette tendance s'inverse à partir d'un certain seuil

4.2 Inconvénients des k-means

Tous les objets n'ont pas forcément la même variance lorsqu'ils sont manipulés. Hors, l'algorithme des k-means ne peut marcher correctement que lorsque tous les ensembles recherchés sont supposés avoir des variances similaires. Dans notre cas, malgré de très bons résultats, notamment avec la méthode Shape-DNA, nous ne pouvons utiliser correctement les k-means uniquement pour détecter des groupes d'objets ayant une variance proche (Un chat et un lion, par exemple, comme on peut le voir dans la Figure 3.) En effet, comme on peut le constater dans la Figure 4, les deux premiers

"groupes" en partant de la gauche ont une variance très faible comparés aux trois autres groupes : exécuter l'algorithme des k-means sur de tels résultats ne fonctionnerait pas.

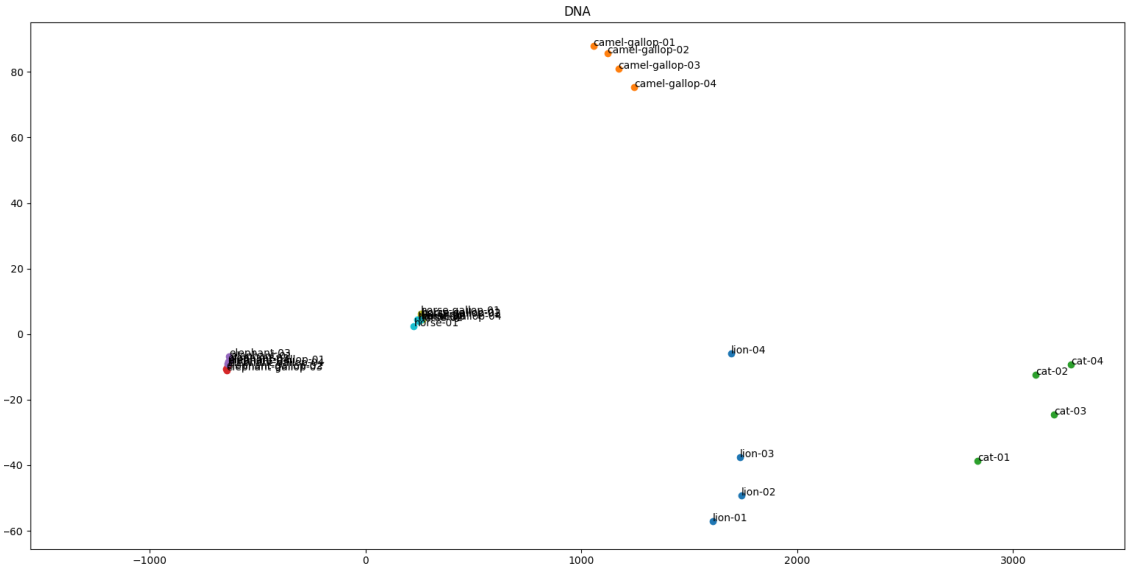

Figure 4: Les variances des différents objets ne sont pas toujours similaires

4.3 Calcul d'un score plus représentatif

Pour pouvoir comparer nos différents résultats, nous avons donc eu besoin de créer un score plus représentatif que ce que le k-means pouvait nous donner. Nous sommes partis de l'idée suivante : pour chaque groupe d'images correspondant à un même objet, calculer la somme des distances entre les empreintes de chaque image, puis sommer le tout. Ainsi, un gros score est équivalent au fait que pour un même objet, les empreintes sont parfois différentes. Cependant, il ne faut pas oublier qu'une mise à l'échelle est nécessaire, nous avons donc divisé ce résultat par la somme des distances entre toutes les pairs possibles du dataset. La formule du score est donc la suivante :

$$\frac{\sum_{objet} \sum_{(im1,im2)\in objet} D(im1, im2)}{\sum_{(im1,im2)} D(im1, im2)}$$

Un test basé sur ce score n'est pas exhaustif (la distance entre les clusters n'est notamment pas explicitement prise en compte), mais il permet d'obtenir des premiers résultats quant à l'efficacité de nos algorithmes.

.obj	GPS					Shape DNA					
	m\nb_cases	5	10	20	50	k	1	5	10	20	40
	1	**0,386**	0,408	0,453	0,425		0,231	**0,145**	0,161	0,156	0,163
	5	0,575	0,596	0,642	0,678						
	10	0,616	0,643								
.off	GPS					Shape DNA					
	m\nb_cases	10	20	50		k	1	5	10	20	40
	1	**0,895**	0,964	1,056			1,421	0,762	0,543	0,529	**0,513**
	5	1,048	1,067								
	10	1,129	1,138								

Figure 5: Résultats obtenus

4.4 Paramétrisation de la méthode GPS

Les résultats que nous avons obtenus laissent penser que des valeurs faibles de m et assez faibles de nb-cases nous donnent de meilleurs résultats. (Voir Figure 5) Nous avons aussi observé qu'un

nombre trop faible pour nb_cases lorsque m vaut 1 donne de mauvais résultats, ce qui semble logique, puisque dans ce cas, on a perdu trop d'informations.

4.5 Shape-DNA/GPS

Pour le premier dataset, l'algorithme de Shape-DNA nous donne un score de 0.145, tandis que le meilleur score obtenu par la méthode GPS est de 0.386. (Voir figure 6)

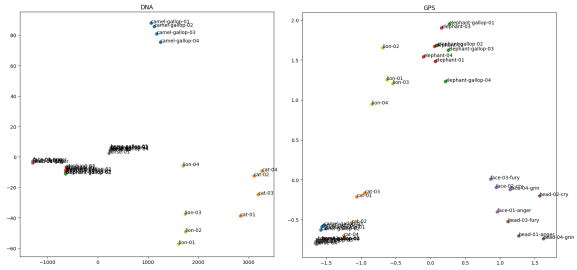

Figure 6: Résultats pour le premier dataset

Pour le premier dataset, l'algorithme de Shape-DNA nous donne un score de 0.513, tandis que le meilleur score obtenu par la méthode GPS est de 0.895.(Voir figure 7)

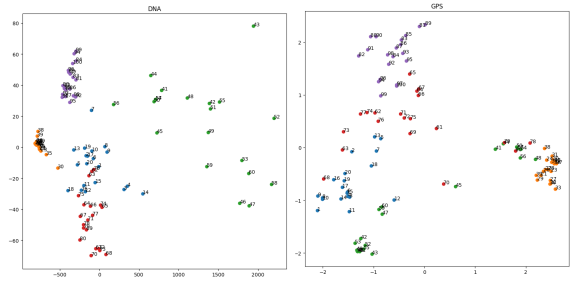

Figure 7: Résultats pour le second dataset

5 Conclusion

Après avoir implémenté les deux méthodes proposées, nos résultats semblent indiquer que la méthode la plus récente (GPS) est moins efficace que celle sur laquelle elle est basée (Shape-DNA). Après avoir fait de nombreux tests avec des paramètres différents pour la méthode GPS, nous obtenons de meilleurs résultats avec des paramètres m et nb-cases faibles.

Le principal avantage de la méthode GPS réside dans le fait que le produit scalaire sur lequel elle est basée a une vraie signification. Cette méthode s'appuie donc sur une base théorique plus solide que celle de Shape-DNA, pour laquelle la distance euclidienne n'a pas autant de sens. En effet, elle dépend surtout des valeurs propres les plus grandes parmi celles qui sont gardées, alors que ce ne sont pas les plus importantes.

Nos résultats globaux sont néanmoins très encourageants, la méthode Shape-DNA permettant de caractériser assez précisément les différentes formes et positions (c'est-à-dire que l'on peut différencier un éléphant d'un cheval, mais également un cheval qui marche d'un cheval au galop). Quant à la méthode GPS, elle donne également des résultats encourageants et globalement corrects, mais définitivement moins bons que la Shape-DNA.