Shape Comparison and Retrieval

Project report, December 13, 2019
INF574 – Digital Representation and Analysis of Shapes

Martín Cepeda
Promotion X2017
École Polytechnique
martin.cepeda@polytechnique.edu

Timothée Darcet
Promotion X2017
École Polytechnique
timothee.darcet@polytechnique.edu

***Abstract*—In this project we implement two methods for characterizing and comparing triangular meshes, both based on the eigen-decomposition of the discrete Laplace-Beltrami operator: Shape-DNA and Global Point Signature. We find that even with a small part of the spectrum of this operator, some of the intrinsic properties of a given shape can be captured (such as a certain degree of pose invariance) and we provide the tools to parallelize and visualize our scheme.**

I. Introduction

Given two or more different shapes, the task of comparing them, or more generally representing them in an unified manner (i.e. having a descriptor in some vector space $\mathbb{R}^d$) is a non trivial problem. To compare all vertices between two meshes is not only expensive (both in computation time and memory) but also it makes hard to define correspondences between points and/or capture more general features of the studied shapes.

To identify or characterize a shape *intrinsically* via a descriptor, we would like to capture several properties, such as isometry (location/representation independence), scaling (size independence), similarity (continuous variance on deformations), completeness (inversibility of the descriptor) and compression (non-redundancy of the descriptor components).

For this task, the spectrum of a shape defined as the eigen-decomposition of the Laplace-Beltrami operator permits to capture the properties of isometry, scaling (by normalization) and similarity [1]. For a mesh, the discretized Laplacian can be described by angle co-tangents in the mesh surface (approach widely used in finite elements analysis): given a mesh with n vertices, the laplacian is the matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ [2] with elements:

$$L_{ij} = \begin{cases} \cot(\alpha_{ij}) + \cot(\beta_{ij}) & , j \in N(i) \\ -\sum\limits_{k\neq i} L_{ik} & , i = j \\ 0 & , j \notin N(i) \text{ and } j \neq i \end{cases}$$

where $N(i)$ are the vertices adjacent to (neighboring) vertex i, and α_{ij}, β_{ij} are the angles opposite to edge ij.

II. Shape-DNA

A first approach studied is a simple descriptor which only considers the first d non-zero eigenvalues of $\mathbf{L}$ [3]:

$$shapeDNA = [\lambda_1, \lambda_2, \cdots, \lambda_d]$$

This descriptor holds a limited amount of information and it's defined for a complete mesh (instead that a single vertex), which makes the comparison between two meshes an $\mathbb{R}^d$ distance problem.

This method has the advantage of being faster to compute (as we only need the eigenvalues) and it gives an immediate descriptor of the entire shape (an $\mathbb{R}^d$ vector).

III. GPS embeddings

A second approach studied in this project is to compute the eigen-decomposition of the Laplacian matrix $\mathbf{L}$ to define its "Global Point Signature" (GPS) as the vector:

$$GPS(v) = \left[\frac{\phi_1(v)}{\sqrt{\lambda_1}}, \frac{\phi_2(v)}{\sqrt{\lambda_2}}, \cdots, \frac{\phi_d(v)}{\sqrt{\lambda_d}}\right]$$

where $\{\phi_i\}_i$ and $\{\lambda_i\}_i$ are the eigen-vectors corresponding to the non-zero eigen-values of $\mathbf{L}$ sorted in increasing order (whose existence is proven in [2]) which are evaluated at a vertex v.

As the spectrum of $\mathbf{L}$ has length n (the number of vertices, which can go from some dozens to millions), we must specify the depth d on which we want to compute the embeddings. This of course doesn't give all the information of the mesh but preserves the main properties of isometry and some degree of similarity. The normalization by $\sqrt{\lambda}$ makes that the distances in the GPS domain are dominated by low frequency eigenvectors, which give already some geometric data of the mesh [3].

Once we have the GPS of each vertex, the studied approach divides the GPS space in m origin-centered nested spheres:

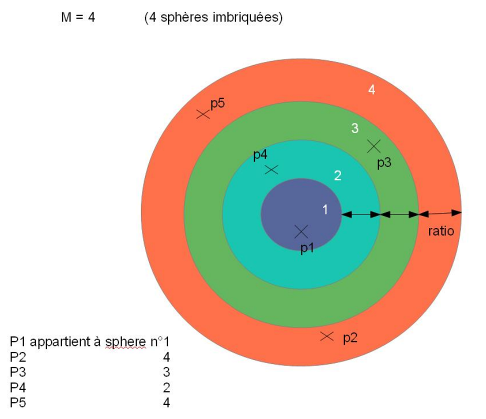

Figure 1. GPS space division (image taken from here)

Then we compute the *G2* distributions defined as the histogram of pairwise distances between the points uniformly sampled in 2 regions. By this operation, we can construct $m(m+1)/2$ histograms for each region combination. By computing then the distances between histograms of different shapes we can compute how distinct they are as their spectrum substantially changes beyond the threshold of a mesh deformation (such as pose changes, small topology changes, etc.)

IV. Implementation

A. General pipeline

In order to implement the two methods described in the previous sections, we proceeded in two stages:

- Firstly, we use the *libigl* library [4] in C++, presented throughout the course, to read, load in memory, compute the Laplacian and its eigen-decomposition via the methods `igl::cotmatrix`, `igl::massmatrix` and `igl::eigs`. Once we have the corresponding spectra, we build the GPS matrices and shapeDNA vectors for each mesh file and save them to a text file.
- Secondly, we read the computed spectra using Python and process them to compute the m regions, the corresponding histograms, distances and dimensionality reduction to visualize the different embeddings' performance.

We decided to work under this scheme because of the mesh-processing capabilities of *libigl*, mainly the memory-efficient laplacian computation and matrix eigen-decomposition via sparse matrix data structures (although this last was not that robust), and also because of Python's almost ready-to-go array manipulation and data visualization functions to compute the histograms and plots.

Additionally, we used a driver file written in python to distribute the calculation of GPS and Shape-DNA descriptors on the computers from the lab rooms. Since there are about 200 computers in these rooms, all with good CPUs, the calculation was extremely efficient and we are quite sure that this pipeline could scale extremely well to bigger datasets, and we were able to do our calculations on the Princeton segmentation dataset. A further optimisation step could be to try to make use of the GPUs that are on these computers too.

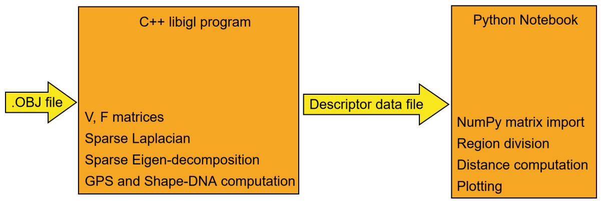

Figure 2. Implemented 2-stages pipeline

B. Descriptor computation

By this framework, the executable program compiled from `main.cpp` receives 2 arguments, an `.OBJ` file path and a desired depth of the spectrum to compute. We specify that this is a desired depth because the `igl::eigs` method may fail to converge in the eigen-decomposition computation. This is because the *libigl* implementation is a simplified version of MATLAB's eigs function so we only have an iterative Power Method in *libigl*.

If `igl::eigs` doesn't converge for a given mesh and d, we smooth the laplacian matrix and prune the "too small" values using the size-normalized Frobenius norm to decide (inspired by this topic in MATLAB forums) and if the eigen-decomposition of this smoothed matrix fails to converge again, we reduce by 1 the depth and we start the computation again.

Finally, we launched this script both in our own computers to test it and in the *Salles Info* via SSH and then we recovered the files to proceed with their analysis.

C. Descriptor analysis

The second script works through a Jupyter Notebook where we defined the necessary functions to import the descriptors produced in the previous program as an `ndarray`, divide the descriptors in regions (as described in III), sample the GPS embeddings, estimate the density from a given distance distribution via a Gaussian Kernel Density Estimation, a wrapper for three dimensionality reduction methods (Multidimensional Scaling – MDS, Principal Component Analysis – PCA and t-Distributed Stochastic Neighbor Embedding – t-SNE) and plotting functions to show annotated scatter plots (for Shape-DNA vectors) and histograms (for GPS embeddings).

For this to work, we specify in the script the m regions to divide the GPS space, the proportion of points to be sampled, the relative paths to descriptor files and the seed to be used in all random-based computations.

V. Results

We applied the implementation described above in the "Mesh Data from Deformation Transfer for Triangle Meshes" dataset [5], which consists of 337 meshes grouped in 8 objects: camel, cat, elephant, face, flamingo, head, horse and lion.

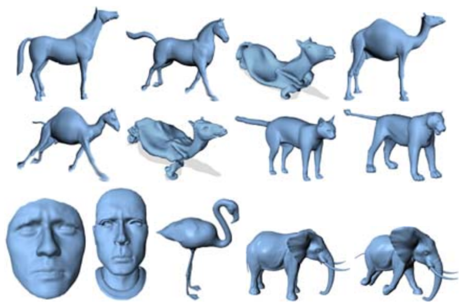

Figure 3. Snapshot of some of the meshes in the dataset

A. Descriptor computation results

Via the *libigl* available functions, we managed to compute 185 emebeddings ranging from 12 to 20 in the spectrum's depth. This first result is due mainly to the partial eigen-decomposition and the limits imposed by the `eigs` method (iterations, tolerance). We chose a desired depth of 20 to make some kind of trade-off between the values used for Shape-DNA (50 eigenvalues) [3] and the ones for GPS (15 eigenvalues) [2].

If we augmented the desired depth beyond 20, we realised that the eigen-decomposition method almost never converged. This is related to the size of the different Laplacian matrices ($\approx 25000 \times 25000$ due to meshes' vertex size) and the iteration limit to approximate the truncated eigen-decomposition (bigger the matrix, slower it is to converge). Although if we increased the maximum iterations, the process became very slow.

We didn't want to explore new eigen-decomposition methods on sparse matrices neither because we thought it escaped the scope of this project. Nevertheless, in order to compare the different embeddings, we had to keep a common spectrum length (of 12) to compare the descriptors.

B. Shape-DNA

We found that with a relatively small descriptor size (12 compared to 50 in the original publication), it was possible to obtain "clusters" of shape descriptors that correspond to the same base shape (among the 8 types listed earlier). With a PCA dimesionality reduction we obtained the most neat groups:

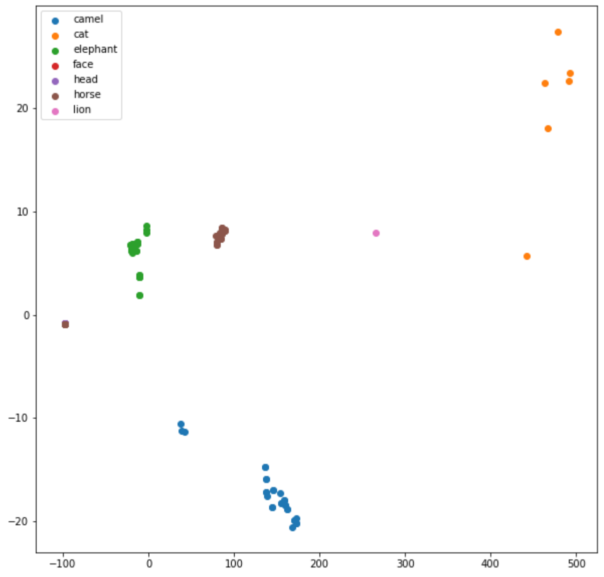

Figure 4. PCA on Shape-DNA vectors (size 12), only base shape tags

With MDS (the visualization proposed in [3]) we have a similar representation but the camel region is no longer convex (horse is in-between):

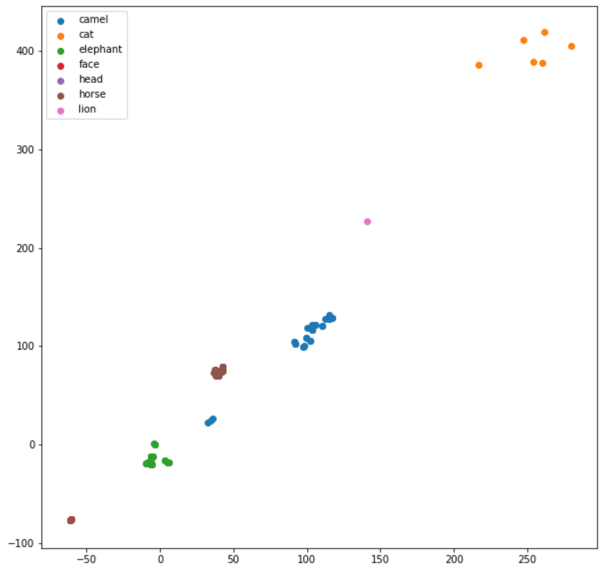

Figure 5. MDS on Shape-DNA vectors (size 12), only base shape tags

In both cases, it's difficult to distinguish between one of horse and face. This can be due to the length of the descriptor (i.e. very close in low-frequency eigenvalues).

C. Global Point Signature

We saw in the previous subsection that the Shape-DNA embeddings could be very sensitive to the pose of an object (namely in camel), which is one of the properties that GPS embeddings tries to revert via the eigen-vector normalization and the random sampling in the m regions. For visualization purposes, we used $m = 1$.

In the case of the head meshes:

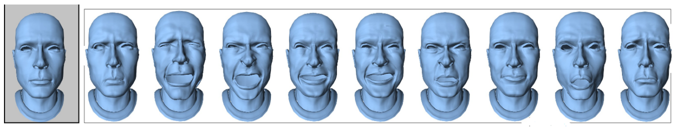

Figure 6. All head poses, image taken from [6]

We obtained an astonishingly accurate result in the proximity between the distance histograms:

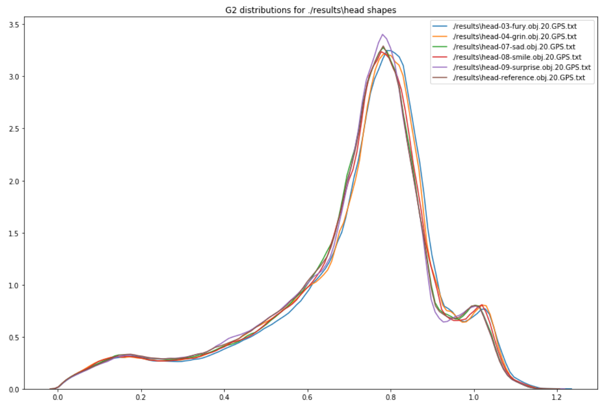

Figure 7. Gaussian-KDE for GPS distances histogram ($m = 1$), all head poses

This effect is seen across all types of objects but with different degrees of proximity. For instance, for the cat meshes:

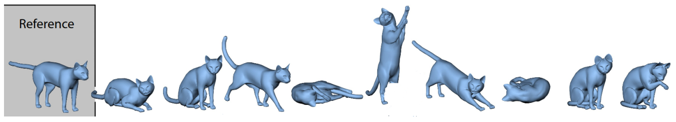

Figure 8. All cat poses, image taken from [6]

We obtain more different distance histograms:

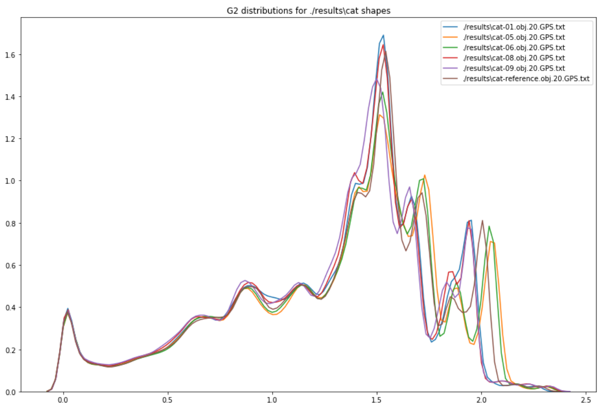

Figure 9. Gaussian-KDE for GPS distances histogram ($m = 1$), all cat poses

This can be partly explained by the different poses for head and cat. In the latter, the mesh has long extremities (legs) that offer a different type of deformations. Nevertheless, solely the distance histogram offers a sort of "profile" for each type of shape, which can be compared with others (just as the frequency signature of a song, for example).

We then computed a dissimilarity mesure between meshes using the kolmogorov smirnov test for empirical distributions, allowing us to compute a MDS of datasets.

VI. Conclusion

Both descriptors Shape-DNA and GPS proved to be accurate with respect to isometry, similarity and compression along the studied meshes. Although the original implementations described in [2] and [3] didn't mention how to implement all the comparisons and distributions, we show a simplified pipeline and provide the code to do it.

References

[1] M. Ovsjanikov, "Spectral methods for isometric shape matching and symmetry detection," PhD dissertation, Stanford University, 2011.

[2] R. Rustamov, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Proceedings of the fifth Eurographics symposium on Geometry processing*. Eurographics Association, 2007, pp. 225–233.

[3] M. Reuter, F.-E. Wolter, and N. Peinecke, "Laplace-beltrami spectra as 'shape-dna' of surfaces and solids." *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006.

[4] A. Jacobson and D. Panozzo. (2019) libigl – a simple c++ geometry processing library. [Online]. Available: https://libigl.github.io/

[5] R. Sumner and J. Popovic. (2004) Mesh data from deformation transfer for triangle meshes. [Online]. Available: http://people.csail.mit.edu/sumner/research/deftransfer/data.html

[6] R. W. Sumner and J. Popovic, "Deformation transfer for triangle meshes." *ACM Trans. Graph.*, vol. 23, no. 3, pp. 399–405, 2004.