Compte-Rendu TP noté d'Algèbre : PCA et SVD

HAI702 Noah Collinet

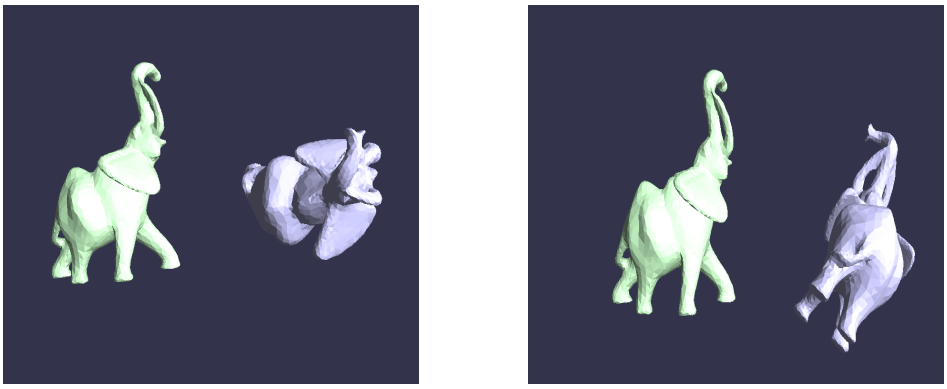
17 octobre 2022

Table des matières

1 Transformation

1.1 Question 1

Tout d'abord on applique des rotations sur le modèle, pour une transformation de 90° selon l'axe x puis y :



(a) Axe x (b) Axe y

FIGURE 1 – Rotation de 90° selon des axes

Maintenant appliquons une projection qui n'est pas orthogonale, avec une rotation de 60° selon l'axe z et en transformant la matrice scale en (1, 3, 2) :

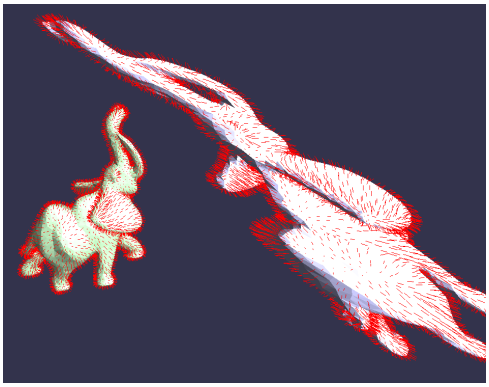


FIGURE 2 – Transformation pas orthogonale

On remarque que la répartition des normales n'est plus uniforme, que tout est déformé.

1.2 Question 2

Nous avons un espace Euclidien muni du produit scalaire canonique, un vecteur $v \in \tau_k$ (le triangle) et un vecteur n normal à ce triangle.
Nous avons bien que leur produit scalaire est nul : <v,n> = 0.
La matrice de transformation B que nous cherchons va nous donner que le produit scalaire <Mv, Bn> = 0.
Avec les transposées nous avons
<v,n> = $v^t n$ = $(Mv)^t Bn$ = $v^t M^t Bn$ = <Mv, Bn>.

Donc notre matrice B = $(T^t)^{-1}$ Donc $<B_n k, B_n k >= ||B_n k||^2$n'est pas forcément égal à 1.

1.3 Question 3

On va récupérer la matrice M avec la ligne :

```
i_transformation = m_transformation;
```

Ensuite on fait sa transposée avec la ligne :

```
i_transformation.transpose();
```

Et enfin on l'inverse en la mettant dans la matrice attendue :

```
m_vector_transformation = Mat3::inverse(i_transformation);
```

1.4 Question 4

Après normalisation des normales, l'affichage est le suivant :

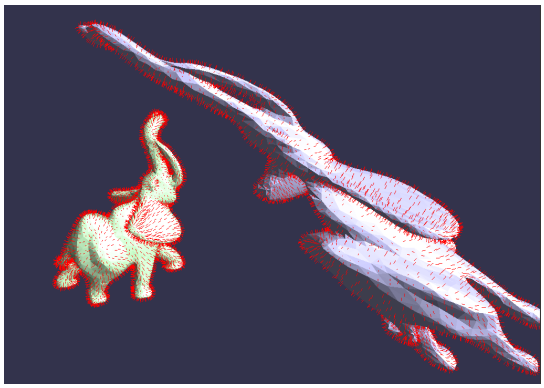


FIGURE 3 – Normales

2 Analyse en Composantes Principales

2.1 Question 1

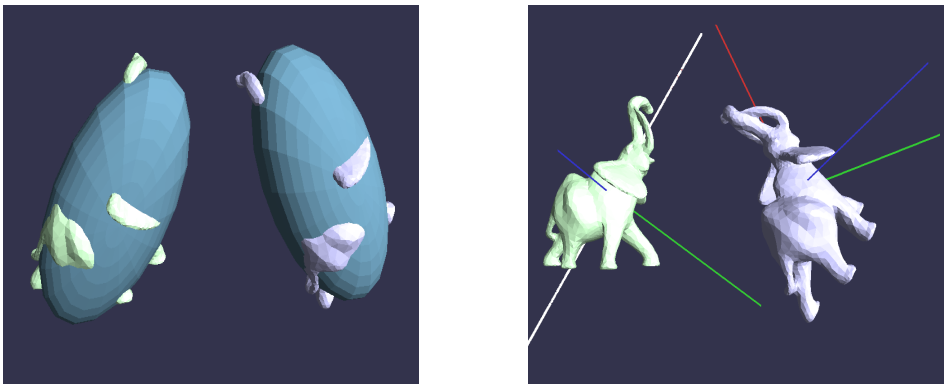
Nous avons une fonction *compute_principal_axis* qui va notamment calculer un vecteur p et une matrice C.
Pour chaque élément du vecteur ps rentré en paramètre, on l'ajoute au vecteur p, à la fin on divise p par la taille de ps avec n la taille de ps :

$$p = \frac{ps}{n}$$

Donc p est le centre de gravité du nuage de points ps.
La matrice C correspond à la variance du nuage de points ps, en effet, on calcule d'abord la norme de la différence entre p et ps, puis on la mets au carré et on la divise par n. Comme le nuage de points est centré, c'est en fait la matrice de covariance de ps.

2.2 Question 4

Affichons les ellipsoïdes ainsi que les directions sur nos modèles :



(a) Ellipsoïdes (b) Directions

FIGURE 4 – Affichage des ellipsoïdes et des directions

3 Décomposition en valeurs singulières

3.1 Question 1

Nous avons une fonction *find_ transformation_ SVD* qui va calculer deux vecteurs p et q et une matrice C.
Pour chaque élément du vecteur ps(respectivement qs) rentré en paramètre, on l'ajoute au vecteur p (respectivement q), à la fin on divise p par la taille de ps avec n la taille de ps et m la taille de qs :

$$p = \frac{ps}{n}$$

$$q = \frac{qs}{m}$$

La matrice C correspond d'abord à la pseudo-inverse du SVD de ps et qs, La matrice *rotation* est le calcul de la rotation depuis la matrice C.

3.2 Question 3

Voici un affichage après appui sur la touche 't' :

FIGURE 5 – SVD