HUNOT-MARTIN
Alaric
M1 Imagine
Université Montpellier
26/09/23

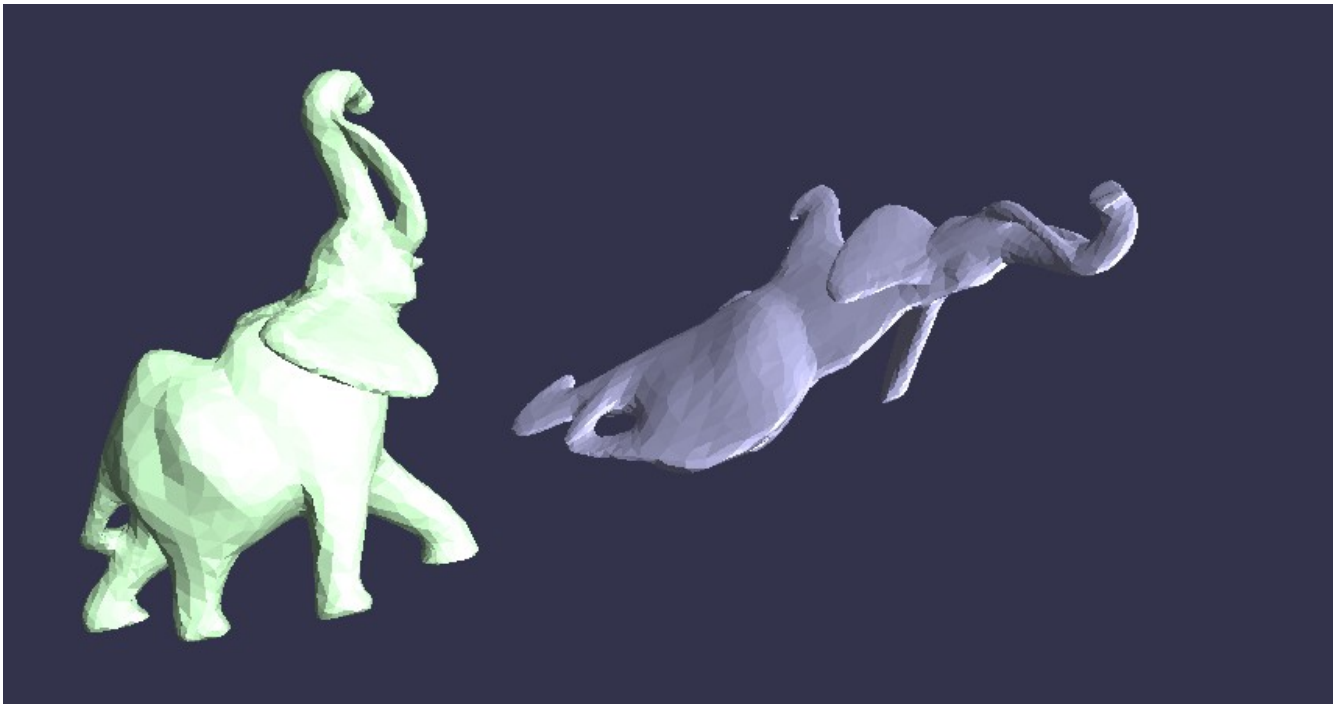
Compte Rendu TP2 HAI702

ACP et SVD

Exercice 1 :

1. on modifie le vecteur de translation, la matrice de transformation et la mise à l'echelle pour déformer l'éléphant

```
transformation = Mat3(1,0.5,0,1,0,-1,-1,0.5,1);
Vec3 translation = Vec3(1., 0., 0.);
Vec3 scale(1., 3., 1.);
```



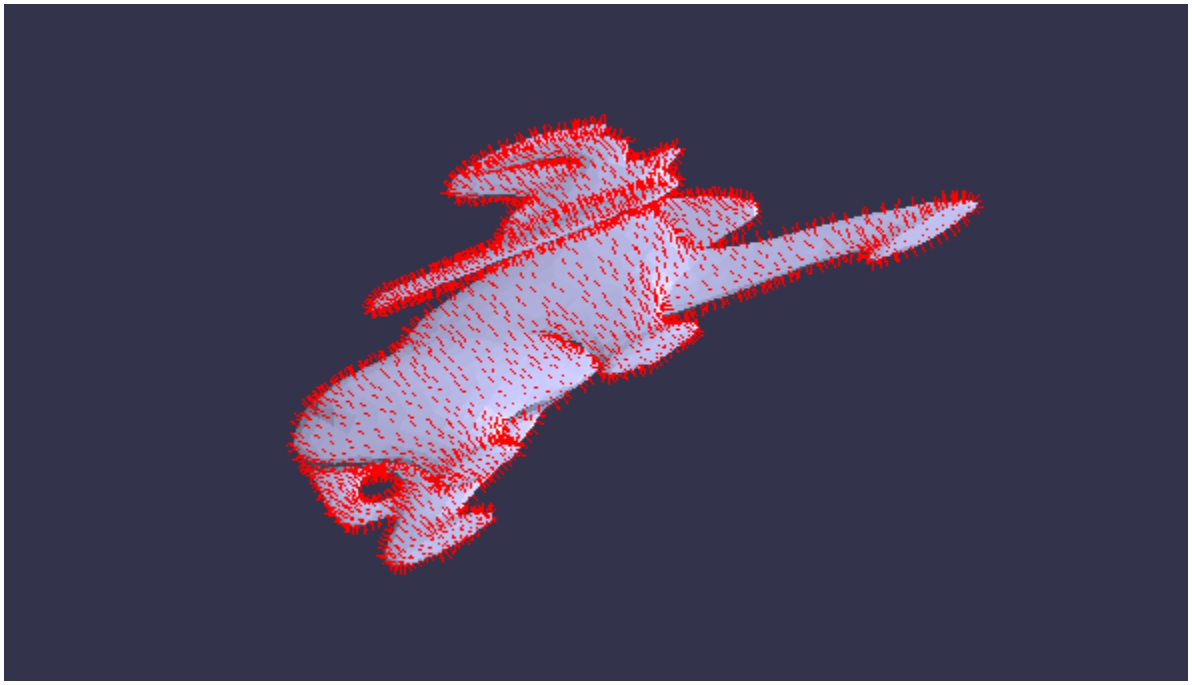
éléphant déformé

2. Il faut que B corresponde à l'inverse de la transposé de M.

3. et 4.

```
m_vector_transformation = i_transformation;
m_vector_transformation.transpose();
m_vector_transformation = Mat3::inverse(m_vector_transformation);
m_vector_transformation.norm();

Vec3 apply_to_normalized_vector(Vec3 const &k_vector) {
Vec3 result = m_vector_transformation * k_vector;
result.normalize();
//Question 1.4: TODO, compléter
return result;
}
```



éléphant avec normales

Exercice 2 :

1. elle génére le barycentre des données correspondant au vecteur p puis l'utilise afin de calculer la matrice de covariance C donnant la direction du nuage de points. C est ensuite diagonaliser pour obtenir les 3 axes de plus grande dispersion du nuage de points.

2.

```
gsl_eigen_symmv(covariance_matrix, gsl_eigenvalues, gsl_eigenvectors, workspace);
//Question 2.2: TODO, trouver la fonction gsl_eigen pour
// ordonner des vecteurs et valeurs propres par ordre décroissant (plus grande valeur propre en premier)
// gsl_eigen_...
gsl_eigen_symmv_sort (gsl_eigenvalues, gsl_eigenvectors, GSL_EIGEN_SORT_ABS_DESC);
```

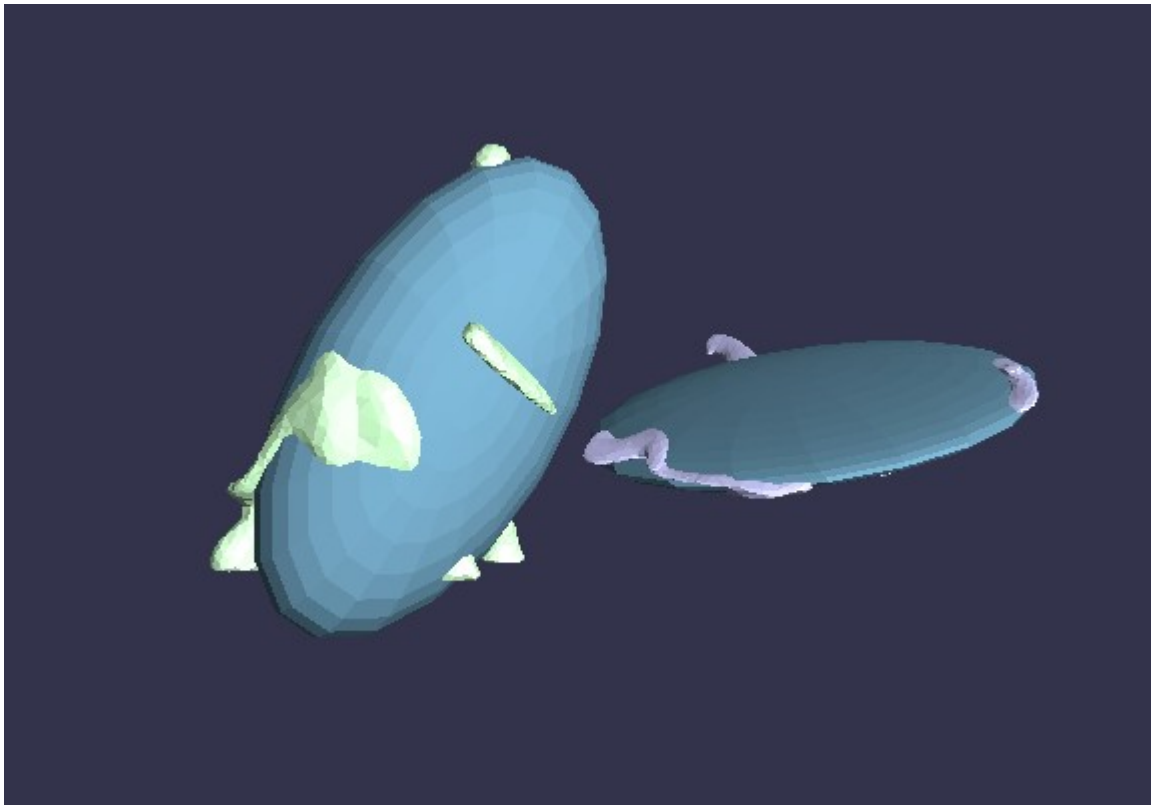
3.

```
Vec3 project(Vec3 const &input_point, Plane const &i_plane) {

//Question 2.3: TODO, projeter input_point sur le plan i_plane
return input_point-(Vec3().dot(input_point-i_plane.point,i_plane.normal)/
i_plane.normal.squareLength())*i_plane.normal;
}

// Calcul de la projection d'un point sur une droite (définie par un vecteur et un point)
Vec3 project(Vec3 const &input_point, Vec3 const &i_origin, Vec3 const &i_axis) {
//Question 2.3: TODO, projeter input_point sur l'axe
return i_origin+(Vec3().dot(input_point-i_origin,i_axis)/i_axis.squareLength())*i_axis;}
```

4.

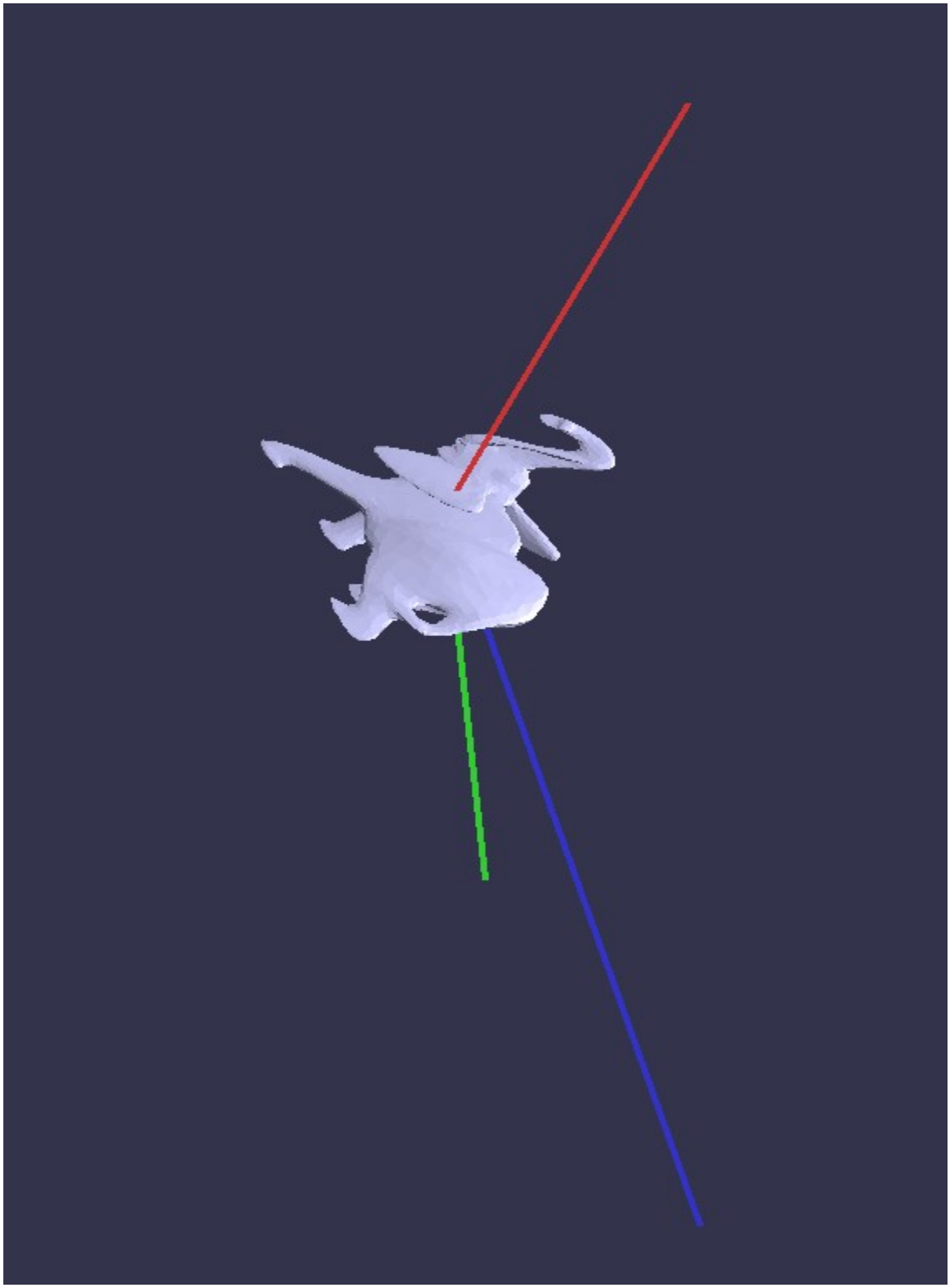


éllipsoide

5.

```
// Question 2.5: TODO Compléter
// variance[i] =...
for (unsigned int v=0; v<mesh.vertices.size(); v++) {
variance[i]+=(projection_on_basis[v]-basis.origin()).length();
}
variance[i]/=mesh.vertices.size();
}
```

6.



repére ACP

Exercice 3 :

1. Elle commence par calculer le centre de masse des ensembles de points ps et qs, représentés respectivement par p et q. Pour ce faire, elle additionne tous les points dans chaque ensemble et divise le résultat par le nombre de points, afin d'obtenir le barycentre de chaque ensemble. Ensuite, elle initialise une matrice C de taille 3x3 à zéros. Cette matrice est utilisée pour calculer la rotation par la méthode de la SVD (décomposition en valeurs singulières). La boucle for parcourt chaque paire de points correspondants dans les ensembles ps et qs, et pour chaque paire, elle calcule un tenseur en utilisant la différence entre le point qi de l'ensemble qs et le barycentre q, et la différence entre le point pi de l'ensemble ps et le barycentre p. Ces tenseurs sont ajoutés à la matrice C pour accumuler les informations nécessaires pour le calcul de la rotation. Enfin, la fonction appelle la méthode setRotation sur la matrice C, qui effectue la décomposition en valeurs singulières pour extraire la rotation souhaitée. Cette rotation est ensuite assignée à la variable rotation. Enfin, la translation est calculée en soustrayant le barycentre p transformé par la rotation à partir du barycentre q. Cette translation est ensuite assignée à la variable translation.