



Université de Picardie Jules Verne

UFR des Sciences Exactes et Naturelles
Département Informatique
Laboratoire MIS (Modélisation, Information et Systèmes)

Prototype du rapport de Stage

Visualisation 3D du Genou et Détection du Point d'Isométrie du LCA

Réalisé par :

Mohamed Amine Sobhi
Etudiant Licence 3 Informatique

Encadré par :

Monsieur Gilles Dequen
Vice-président Recherche et Stratégie scientifique
Professeur des Universités
Maître de stage

Année universitaire 2024–2025

Table des matières

1	Introduction	2
2	Technologies utilisées	2
3	Structure du projet	2
4	Tâche 1 — Lecture, segmentation et affichage	3
5	Tâche 2 — Interaction utilisateur (Pointage manuel)	6
6	Tâche 3 — Mesure interactive de distances	8
7	Annexes	11
8	Bibliographie	11

1 Introduction

Ce stage est réalisé au sein du laboratoire MIS de l'Université de Picardie Jules Verne. Il a pour objectif le développement d'un outil interactif permettant la visualisation 3D du genou humain à partir de fichiers STL. Ce projet inclut également l'intégration progressive d'un module de détection du point d'isométrie du ligament croisé antérieur (LCA), en lien avec des analyses biomécaniques futures.

2 Technologies utilisées

PyVista

PyVista est une surcouche Python de VTK, facilitant l'analyse de maillages 3D et la création d'environnements interactifs. Il permet :

- de lire des fichiers STL et de visualiser des objets 3D,
- de manipuler la géométrie (connectivité, surfaces, distances),
- d'ajouter des éléments graphiques (texte, sphères, lignes...).

Lien : docs.pyvista.org

Autres bibliothèques

- NumPy – calculs vectoriels sur les points 3D,
- PyQt5 – interface pour sélectionner les fichiers STL.

3 Structure du projet

Arborescence du projet

```
isoLCA/
|-- stl_files/          -> Fichiers STL à analyser
|-- src/               -> Code source
|   |-- stl_loader.py   # Chargement & segmentation STL
|   |-- viewer.py       # Affichage interactif 3D
|-- main.py            -> Script principal
|-- requirements.txt    -> Dépendances à installer
|-- README.md          -> Description du projet
```

4 Tâche 1 — Lecture, segmentation et affichage

Objectif

Segmenter automatiquement les 3 os du genou à partir d'un fichier STL multi-structures :

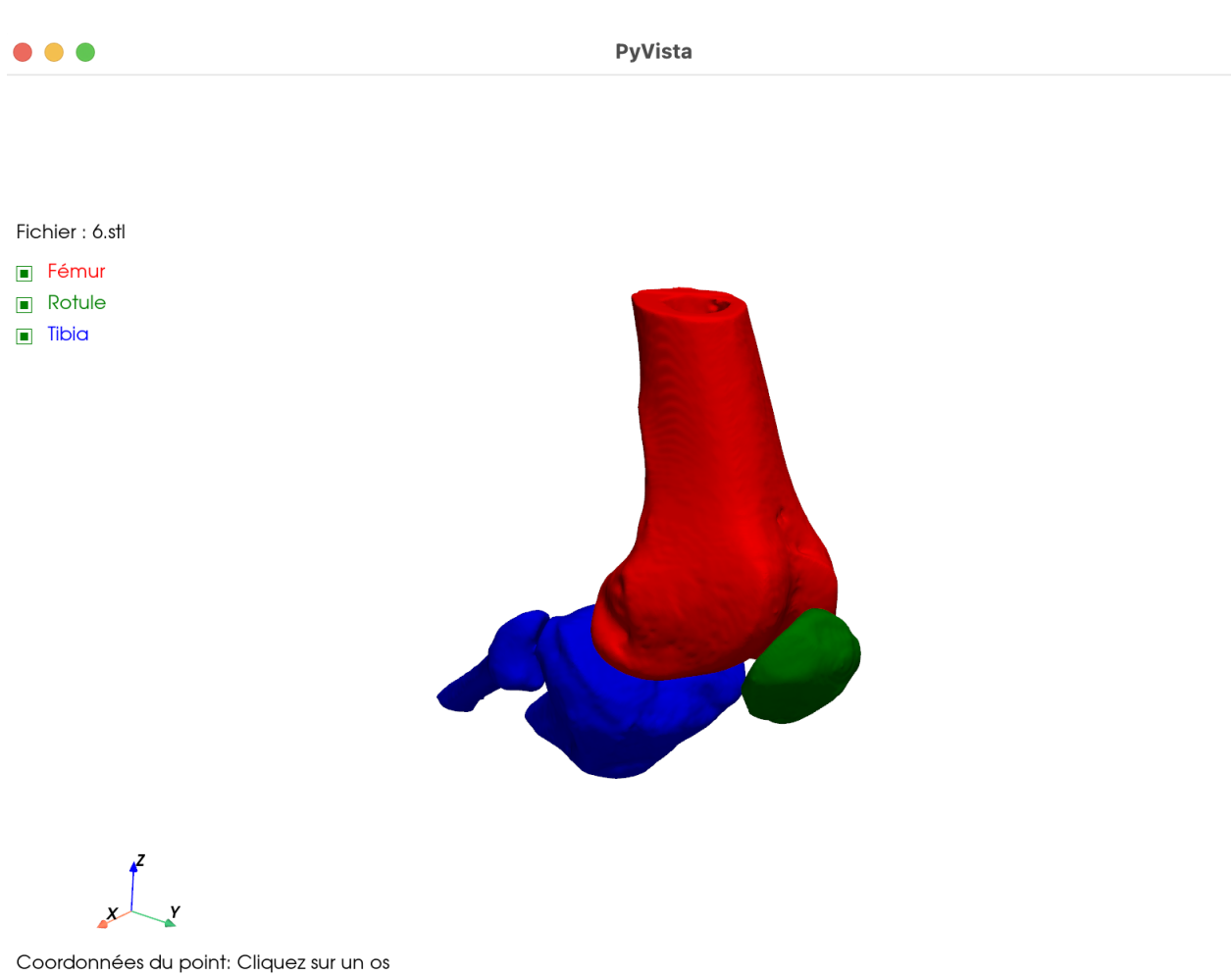
- rotule : plus petit en nombre de points,
- tibia : centre de gravité le plus bas (axe Z),
- fémur : centre de gravité le plus haut (axe Z).

Extrait du code – stl_loader.py

Segmentation automatique

```
def charger_et_segementer(filepath):  
    mesh = pv.read(filepath)  
    labeled = mesh.connectivity(mode='points', output_values='point_arrays')  
    infos = []  
    for i in range(int(labeled['RegionId'].max()) + 1):  
        r = labeled.threshold([i, i], scalars='RegionId')  
        infos.append({  
            'region': r,  
            'n_points': r.n_points,  
            'z_mean': np.mean(r.points[:, 2])  
        })  
    infos.sort(key=lambda r: r['n_points'])  
    return {  
        'rotule': infos[0]['region'],  
        'tibia': min(infos[1:], key=lambda r: r['z_mean'])['region'],  
        'femur': max(infos[1:], key=lambda r: r['z_mean'])['region']  
    }
```

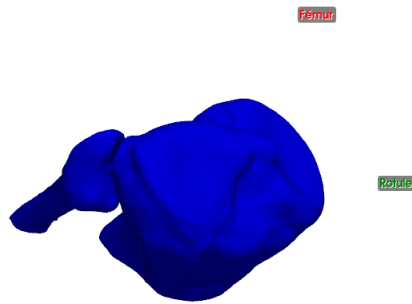
Résultats visuels





Fichier : 6.stl

☐ Fémur
☐ Rotule
☐ Tibia



Coordonnées du point: Cliquez sur un os

FIGURE 2 – Cases à cocher pour masquer/afficher chaque os (rotule et fémur masqués)

5 Tâche 2 — Interaction utilisateur (Pointage manuel)

Objectif

- Permettre à l'utilisateur de cliquer sur un os :
- Une sphère jaune apparaît à cet endroit,
 - Les coordonnées du point sont affichées à l'écran.

Extrait minimal – viewer.py

Affichage des coordonnées du point sélectionné

```
def update_measurement(point):
    meas_handler.add_point(point)
    coord_str = (
        f"Coord: X={point[0]:.2f}, Y={point[1]:.2f},"
        f" Z={point[2]:.2f}"
    )
    plotter.add_text(
        coord_str, position=(20, 40),
        name="coord_display", render=True
    )
    if len(meas_handler.current_points) == 2:
        meas_handler.complete_measurement()
```

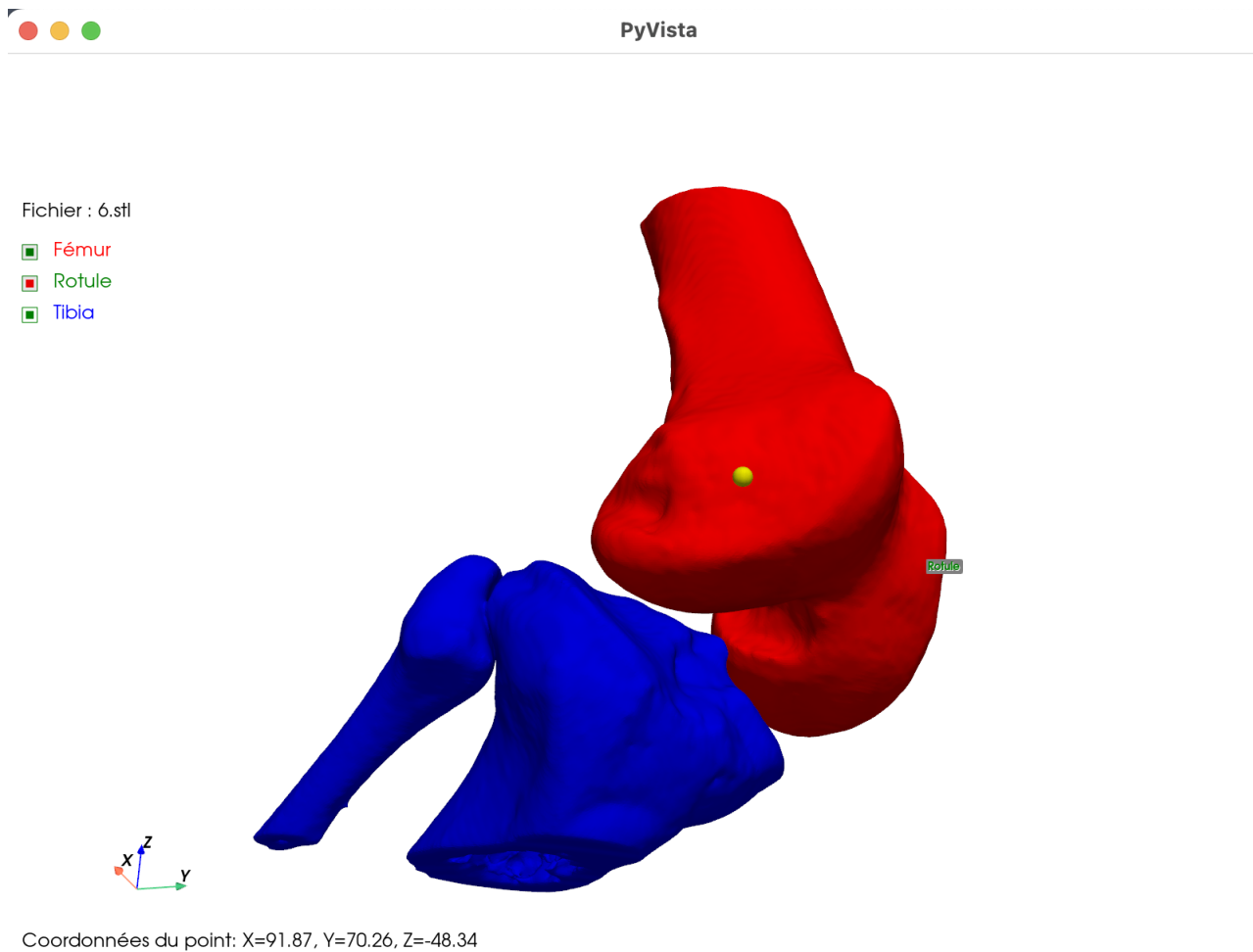


FIGURE 3 – Pointage d'un os, sphère/point jaune et coordonnées 3D.

6 Tâche 3 — Mesure interactive de distances

Objectif

Permettre de mesurer la distance (en mm) entre deux points sélectionnés :

- sur un même os,
- ou sur deux os différents.

Extrait du code – MeasurementHandler

Affichage mesure 3D

```
def complete_measurement(self):
    p1, p2 = np.array(self.current_points)
    dist = np.linalg.norm(p2 - p1)
    line = pv.Line(p1, p2)
    self.plotter.add_mesh(line, color="black", line_width=4)
    self.plotter.add_point_labels(
        [(p1 + p2) / 2], [f"{dist:.2f} mm"],
        font_size=20, background_color="white", shadow=True
    )
    self.current_points = []
```

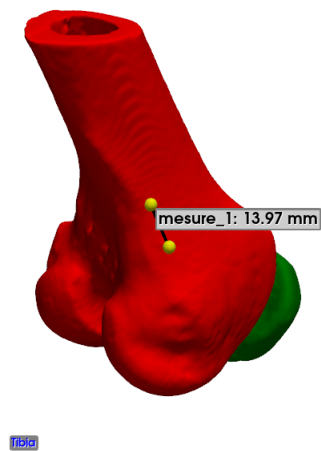
Résultats visuels

Fichier : 6.stl

■ Fémur

■ Rotule

■ Tibia



Coordonnées du point: X=81.61, Y=53.16, Z=-21.99

FIGURE 4 – Mesure entre deux points sur un même os.

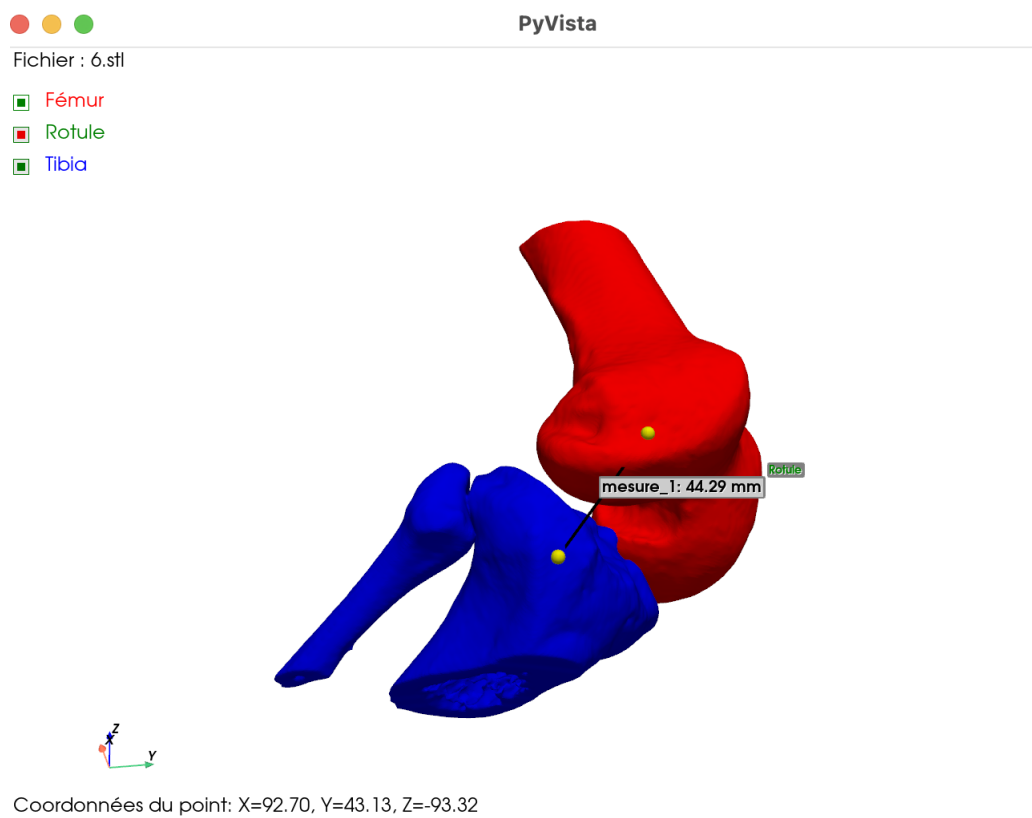


FIGURE 5 – Mesure entre deux points sur deux os différents (cas illustré : fémur - tibia).

7 Annexes

8 Bibliographie

- PyVista – <https://docs.pyvista.org/>
- VTK – Visualization Toolkit – <https://vtk.org/>
- NumPy – <https://numpy.org/>
- PyQt5 – <https://riverbankcomputing.com/software/pyqt/>