Définitions

Hyperbole

L'hyperbole est une conique d'excentricité > 1,

Une conique, est une courbe plane qui est initialement définie comme l'intersection d'un cône de révolution avec un plan, voici différentes coniques :

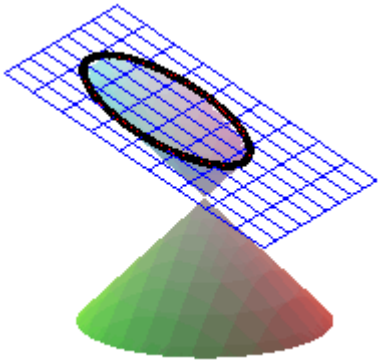


Figure 3: Ellipse

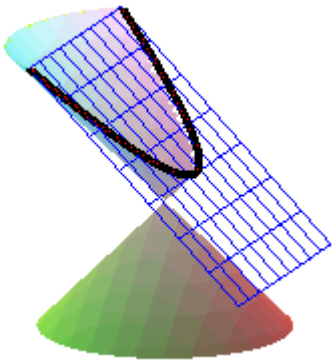


Figure 1: Parabole

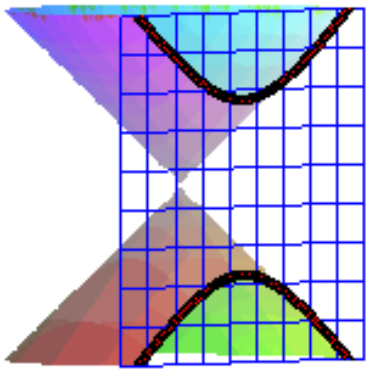
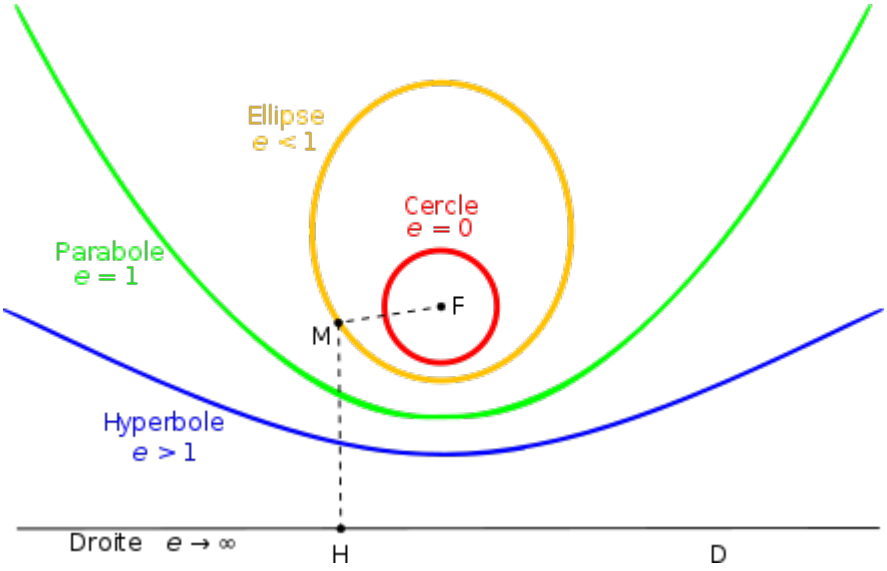


Figure 2: Hyperbole

L'excentricité est l'un des paramètres d'une courbe conique. C'est ce qui permet de mesurer le décalage du foyer sur l'axe principal de l'ellipse. Si elle est proche de 0, la trajectoire est presque circulaire, et si elle est proche de 1, l'ellipse est alors très allongée, voici un exemple :



On peut donc apercevoir un cercle avec une excentricité e=0, une ellipse avec e=0,5, une parabole avec e=1 et une hyperbole avec e=2. Une droite a une excentricité qui tend vers l'infini.

L'hyperbole est donc une **courbe plane**, et cette hyperbole s'inscrit dans un **plan**. C'est ce qu'il faut retenir pour la suite.

Plan

Un plan est une surface bidimensionnelle infinie qui s'étend dans toutes les directions. Il suffit de 3 points non alignés pour déterminer de manière unique un plan. En d'autres termes, si l'on a 3 points distincts qui n'appartiennent pas à une même ligne droite, il existe un plan qui passe par ces 3 points.

Mr Jouk a défini les caractéristiques de notre plan comme suit :
Il doit impérativement passer par les 2 points tricuspide et pulmonaire, c'est à dire que ces deux points sont inscrits dans le plan. Or nous avons déjà ces deux points. Il faut maintenant déterminer le 3 ème point caractéristique de notre plan.

Mr Jouk nous a demandé de faire en sorte que la 3ème orientation du plan – car je le rappelle, nous avons déjà les 2 orientations du plan qui sont définies puisque nous avons 2 points – soit calculée par la méthode des moindres carrés :

Méthode des moindres carrés

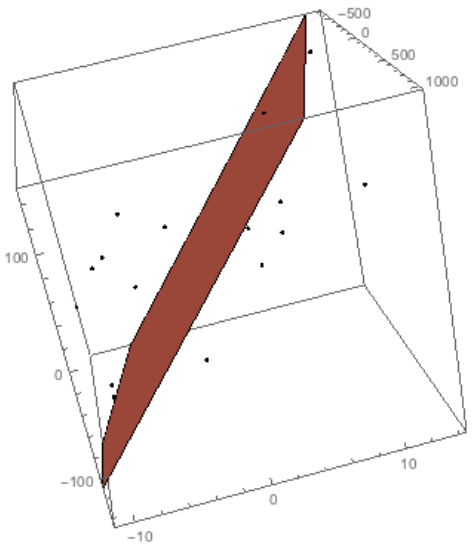
Un plan des moindres carrés est une méthode utilisée en statistiques et en géométrie pour ajuster un modèle mathématique à un ensemble de données expérimentales. Dans le contexte d'un nuage de points tridimensionnel, le but est de trouver le plan qui minimise la somme des carrés des distances perpendiculaires entre chaque point et le plan.

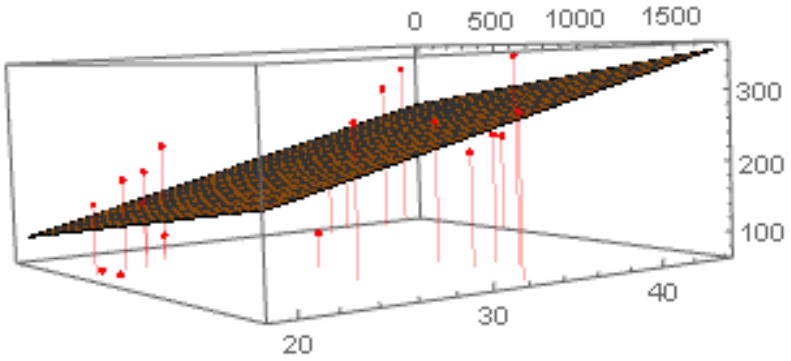
Supposons que nous ayons un ensemble de points (xi,yi,zi), où I représente les indices des points. Un plan des moindres carrés peut être décrit par une équation de la forme :

$$z=ax+by+c$$

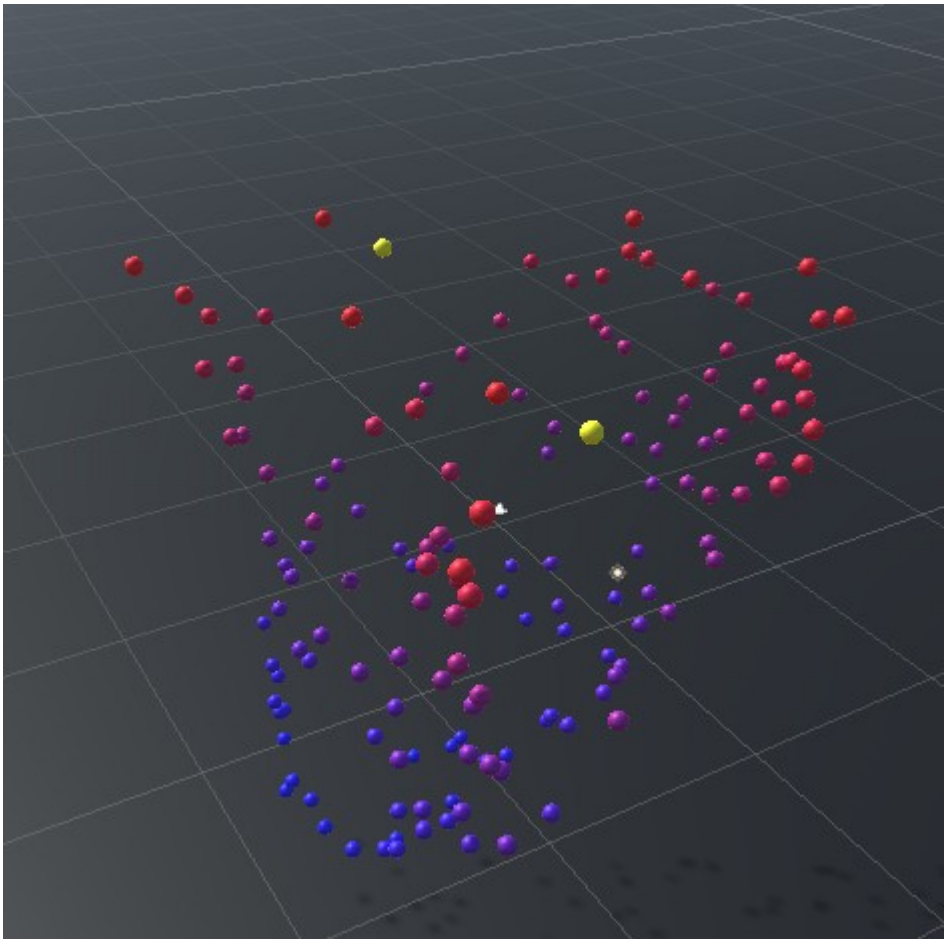
où a, b, et c sont les coefficients du plan que nous cherchons à déterminer. La méthode des moindres carrés consiste à ajuster ces coefficients de manière à minimiser la somme des carrés des distances verticales (ou résidus) entre chaque point **(xi,yi,zi)** et le plan **z=ax+by+c.**

Voici deux examples de représentations visuelles de plans des moindres carrés, qui minimisent les distances perpendiculaires au plan de chaque point :



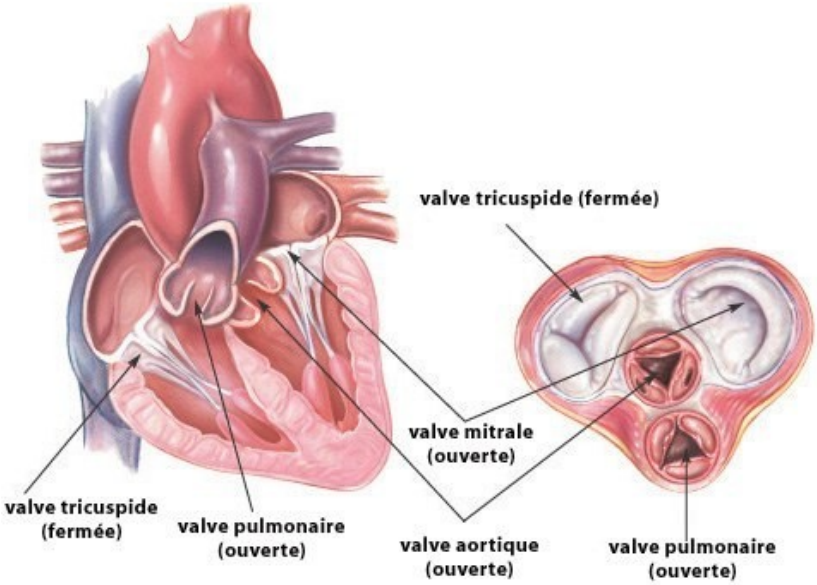


Voici tout d'abord le résultat du nuage de points sur Unity3D



Les deux sphères jaunes représentent respectivement le centre géométrique de la valve pulmonaire ainsi que le centre de de la valve tricuspide.

Voici un schéma pour mieux se représenter où se situent ces valves.

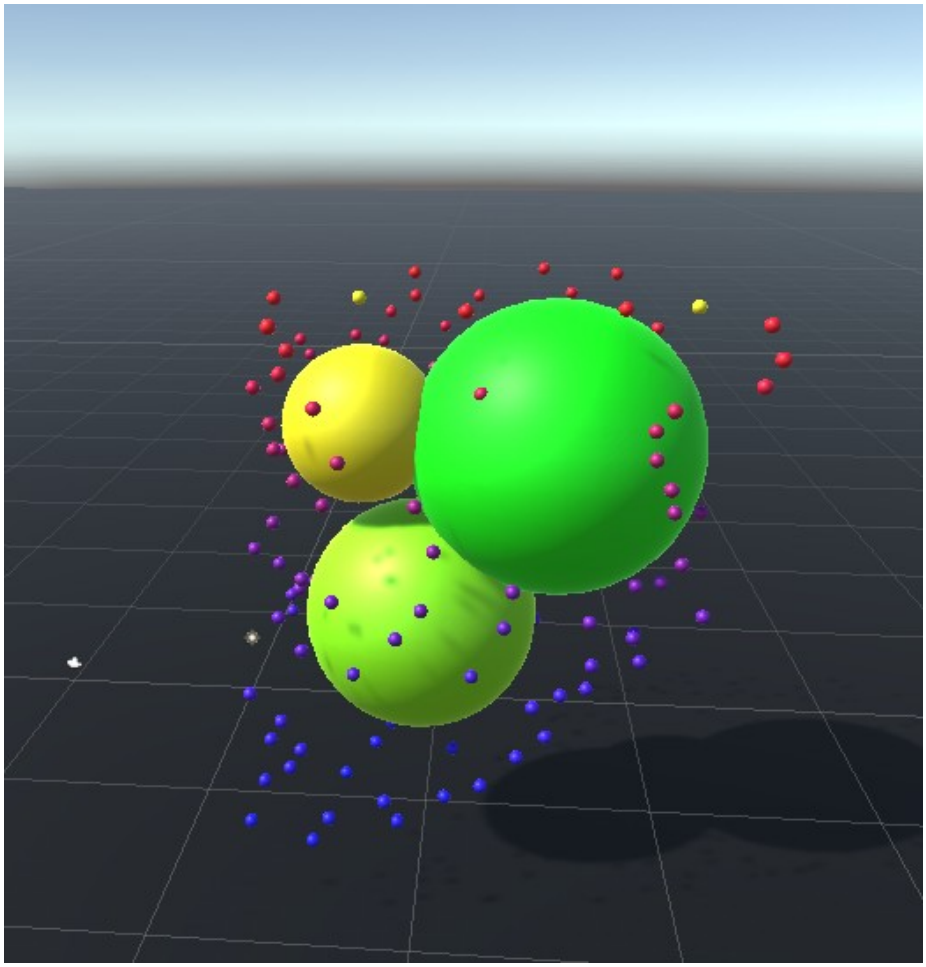


Nous avons ensuite placé des sphères à l'intérieur de ce nuage de points. Ces sphères devaient être positionnées manuellement à un endroit précis, et avec une taille précise, afin de se placer au mieux à l'intérieur de ce nuage de points.

Ces sphères serviront à déterminer, à l'aide de leur centre géométrique, l'hyperbole que nous cherchont à obtenir.

On effectuera un ajustement linéaire (méthode des moindres carrés) par rapport aux centres de ces sphères.

Voici le résultat obtenu :



Une fois les sphères placées, il faut calculer le plan :

Pour tracer le plan, nous avons déjà les 2 points jaunes (centres géométriques de la valve tricuspide et la valve pulmonaire) qui vont permettre de déterminer une des orientations du plan, et il faut donc trouver ce 3ème point qui permettra d'obtenir un plan fixe, unique.

Pour cela, la dernière orientation sera calculée de façon à satisfaire la condition suivante : le 3 ème point caractéristique du plan sera placé de façon à **minimiser la somme des carrés des distances des centres des 3 sphères, par rapport au plan**. (méthode des moindres carrés).

Le point a été calculé à l'aide d'un algorithme fait en C# :

```
Vector3 CalculateLeastSquaresPoint(params Vector3[] points) {
// Calculer le point des moindres carrés sur la ligne passant par les points donnés
// Ce point minimise la somme des distances au carré entre les points d'origine et la ligne

// Calculer le centre de masse (centroid) des points
Vector3 centroid = Vector3.zero;
foreach (Vector3 point in points)
{
centroid += point;
}
centroid /= points.Length;

// Calculer la somme des vecteurs différence entre chaque point et le centre de masse
Vector3 sum = Vector3.zero;
foreach (Vector3 point in points)
{
sum += (point - centroid);
```

```
}

// Retourner le point des moindres carrés
return centroid + sum / points.Length;
}
```

Explication du code

Calcul du centre de masse (centroid) :

Un vecteur est initié à zéro. [0, 0 ,0]

La fonction itère à travers tous les points d'entrée, et pour chaque point, elle ajoute ses composantes au vecteur centroid. Centroid contiendra donc toutes les composantes de chaque point.

Une fois que tous les points ont été traités, le vecteur centroid est divisé par le nombre total de points pour obtenir le centre de masse moyen.

Calcul de la somme des vecteurs différence :

Un autre vecteur, sum, est initié à zéro.

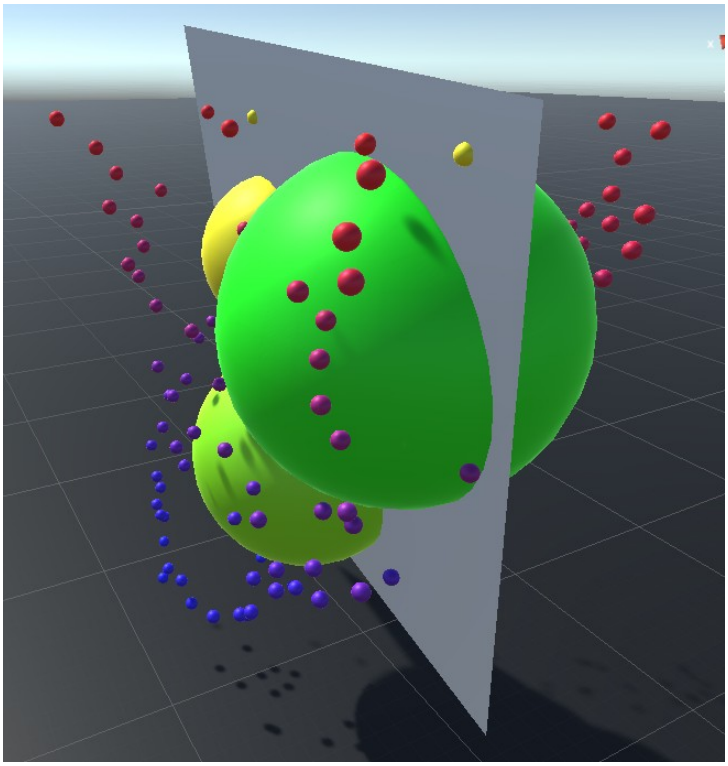
La fonction itère à nouveau à travers tous les points d'entrée.

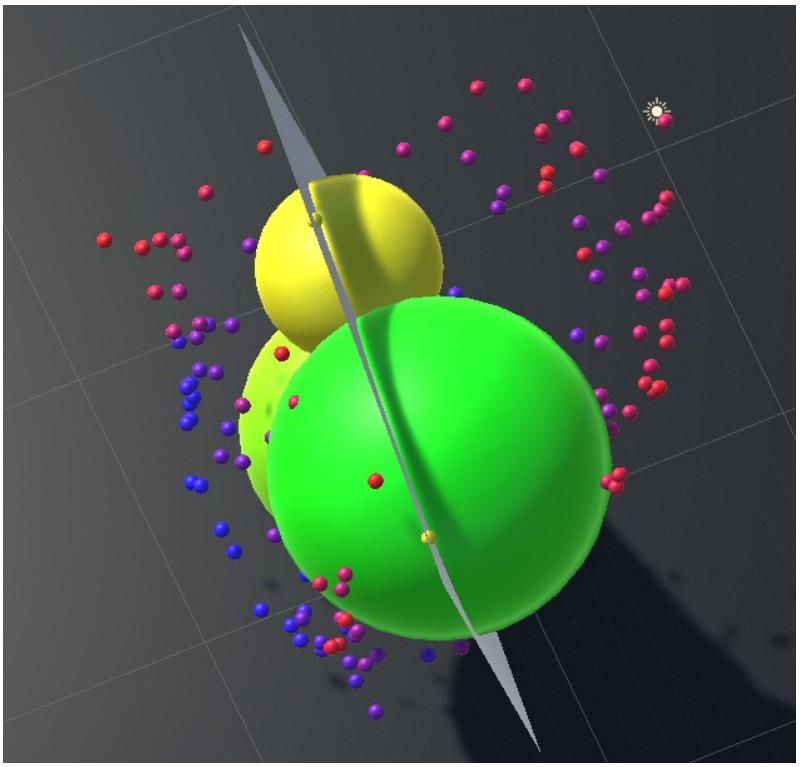
Pour chaque point, elle calcule le vecteur différence entre ce point et le centre de masse, puis l'ajoute au vecteur sum.

Calcul du point des moindres carrés :

Le point des moindres carrés est obtenu en ajoutant au centre de masse la somme des vecteurs différence, le tout divisé par le nombre total de points.

Ainsi nous obtenons le 3 ème point caractéristique permettant de tracer notre plan , voici le résultat :





Comme on peut l'aperçevoir sur le rendu, le plan passe bien par les 2 points d'entrée des valves, mais il ne passe pas nécessairement par le centre des 3 sphères, **il passe par le point qui minimise la distance des projections des centres des sphères sur le plan, afin de l'ajuster au mieux entre les 3 sphères**.

Le nuage de points représenté ici, est en fait un ensemble de points d'intérêts minutieusement relevés à partir des coupes du coeur depuis ImageJ, et ces points représentent en fin de compte les bords externes du ventricule droit, que l'on appelera "disinclinaisons".
Cependant, Mr Jouk avait besoin de plus de repères pour améliorer le placement des sphères qu'il jugeait inexact, et pour cela il nous a demandé de modéliser l'entièreté de l'enveloppe du muscle cardiaque.

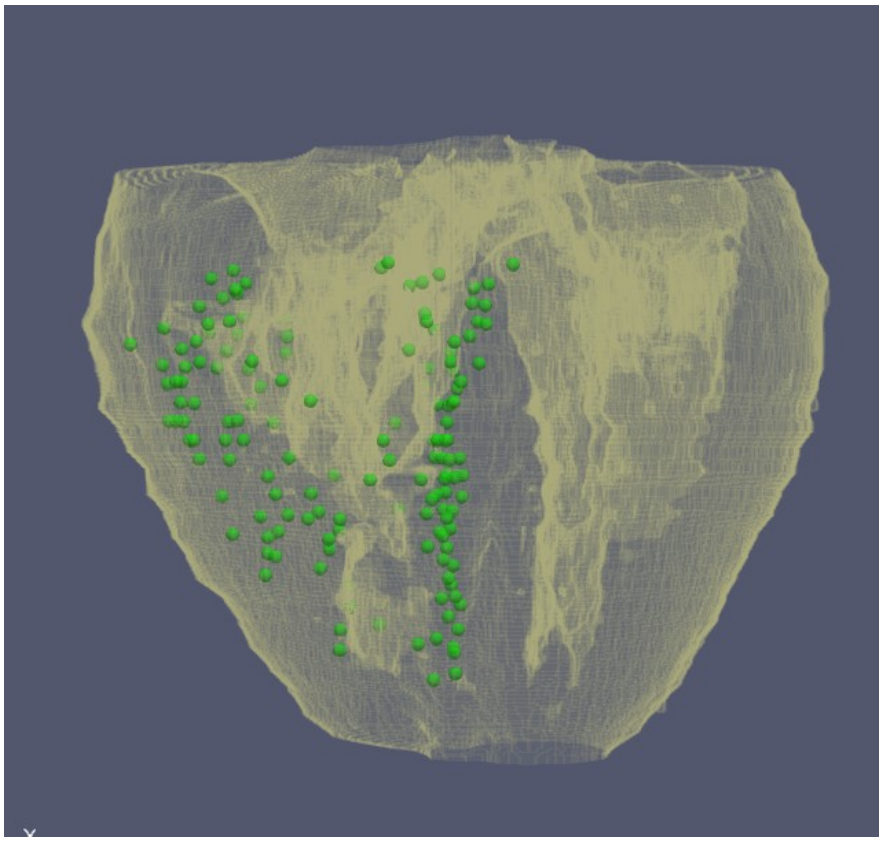
Pour cela il fallait extraire depuis un fichier .tif, qui contient 320 slices (coupes) représentant le volume du coeur, l'enveloppe, chaque coordonnée en xyz de tous les pixels blancs.
Voici un lien vers une animation .gif représentant le fichier tif, partant de la coupe 1 à la coupe 320.

Le script :

```
setBatchMode(true);
open("/home/aitlhoui/workspace/imagej/VOLUME.tif.zip");
getDimensions(width, height, channels, slices, frames);
getPixelSize(unit, pixelWidth, pixelHeight);
nom_fichier = "/home/aitlhoui/workspace/imagej/enveloppe.csv";
fichier = File.open(nom_fichier);
File.close(fichier);
for (plane=1; plane <= slices; plane++)
{
	showProgress(plane, slices);
	setSlice(plane);
	for (ligne = 0; ligne < height; ligne++)
	{
		for (colonne = 0; colonne < width; colonne++)
		{
			valeur = getPixel(colonne, ligne);
			if (valeur == 255)
			{
				x = colonne * pixelWidth;
				y = (height - ligne) * pixelHeight;
				z = plane * pixelWidth;
				str = "" + d2s(x,3) + "," + d2s(y,3) + "," + d2s(z,3);
				File.append(str,nom_fichier);
			}
		}
	}
}
print("fini");
setBatchMode("exit & display");
```

le script boucle sur chaque slice, puis dans chaque slice, il sélectionne chaque ligne, et sur chaque ligne, il boucle sur chaque pixel de la ligne, vérifie la couleur du pixel, s'il est blanc, il ajoute les coordonnées de chacune de ses composantes dans une variable x, y, z . Puis formate l'ensemble de ces valeurs dans une chaîne str en convertissant chaque x, y ,z du format décimal au format string en spécifiant le nombre de décimales après la virgule, en choisissant la virgule comme délimiteur (convention csv).

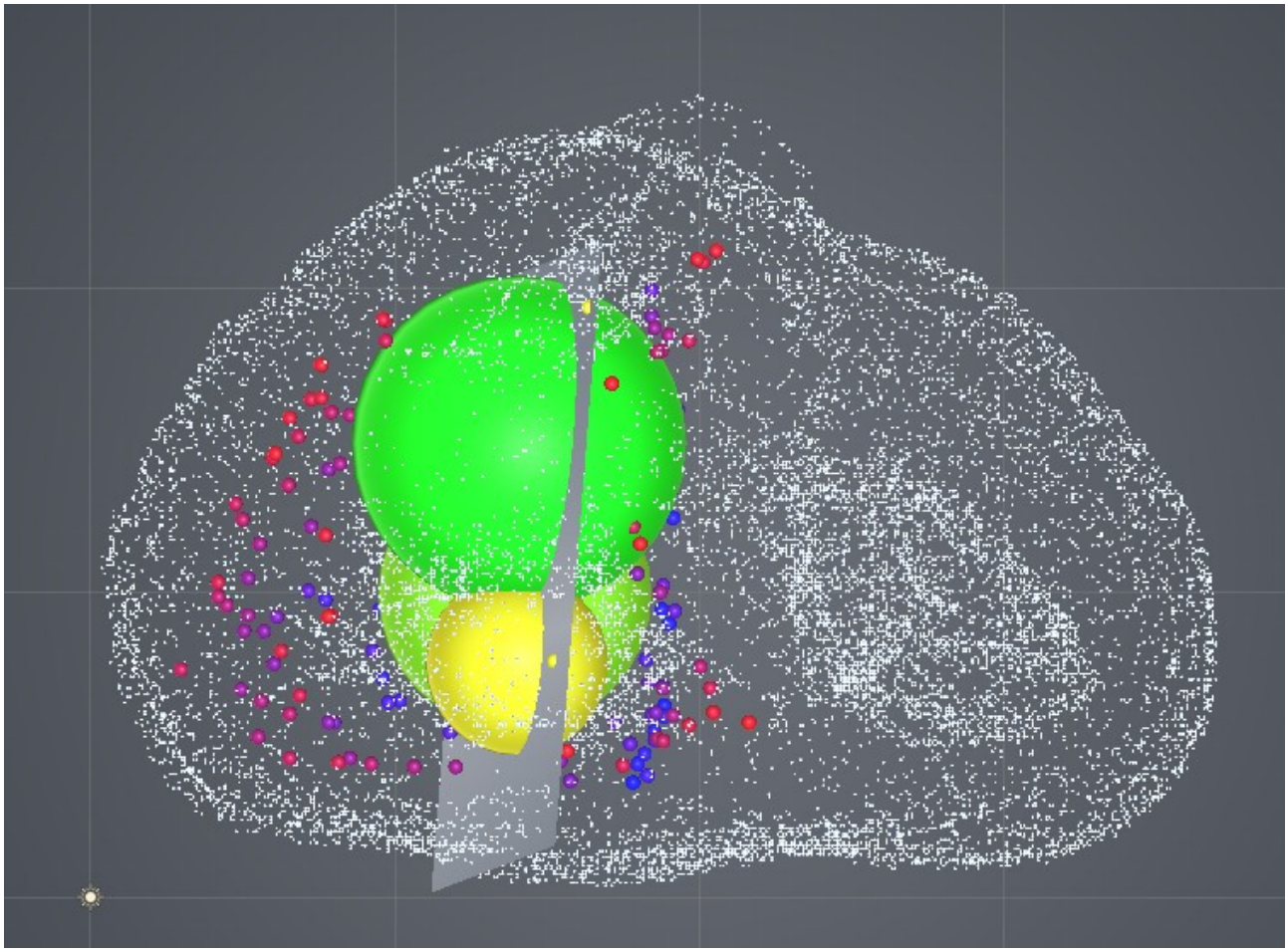
Une fois la liste des coordonnées obtenues (+ de 650 000 ...)
On peut enfin modéliser l'enveloppe, premièrement je l'ai fait sur ParaView, ce qui était très pratique et léger à modéliser :



Cette visualisation est très intuitive, on peut bien aperçevoir les points verts, qui représentent les disinclinaisons sélectionnées au départ (semaine 1) et le reste représente le volume de l'enveloppe du coeur.

Mais ParaView n'était pas pratique pour positionner des sphères et calculer des plans. Nous sommes donc retournés sur Unity3D, nous avons fait face à des gros problèmes de ressources, la modélisation de 650 000 points était beaucoup trop lourde alors le script a été très légèrement modifié pour modéliser 1 point sur 37 afin d'alléger le coût de traitement.

Voici une capture du dessus :



Et voici une capture de profil :