

# *HMIN318M: Imagerie médicale et 3D*

Rendu TP : Imagerie 4D

Tianning MA

M2 IMAGINA

29/10/2020

## Table de matière

A. Introduction .....	2
B. Compilation.....	2
C. Rendu et explication des exercices .....	2
1. Visualisation L'image.....	3
2. Détection des cellules dans les images 3D.....	3
2.1 Application du filtre médian pour éliminer les bruits.....	3
2.2 Seuillage pour éliminer le fond et ne garder que les noyaux cellulaires .....	4
2.3 Éliminer les quelques pixels isolés par morphologie mathématique .....	4
2.4 Identifier les cellules par composantes connexes .....	5
2.5 Calculer le barycentre de chacune des cellules et le sauvegarder dans un fichier.....	5
3. Algorithme de suivi de cellule.....	6
4. Visualisation de trajectoire 3D.....	6
Annexe .....	9
1. Code source du Program principal.....	9
2. Fichier pour les données de barycentres.....	17

## A. Introduction

Ce compte rendu est dédié au TP Imagerie 4D du cours d'Imagerie médicale 3D (HMIN318M). Le travail est basé sur Tp précédent (TP0).

Toutes les questions sont répondues. Vous trouverez le code source et les fichiers demandés dans la partie Annexe de ce rapport.

Vous pouvez aussi trouver l'intégralité de ce tp dans mon espace git avec l'adresse ci-dessous :

<https://github.com/matianning/ImagerieMedicale3D/tree/main/TP%20Imagerie4D>

## B. Compilation

Pour compiler sous linux : g++ -o main.exe main.cpp -O2 -L/usr/X11R6/lib -lm -lpthread -lx11

Exécution : ./main.exe

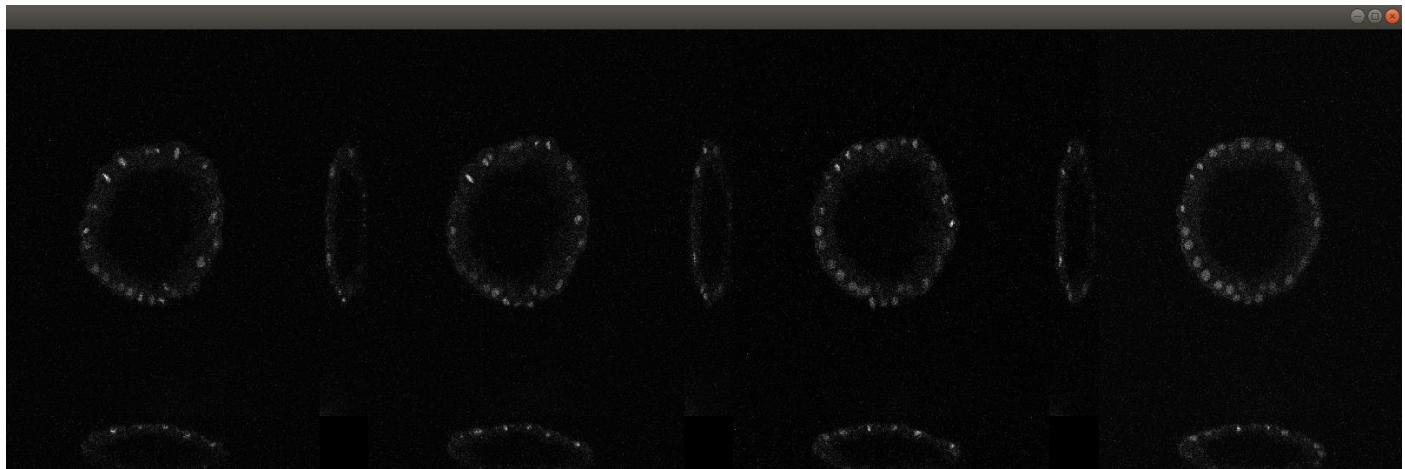
Les fichier stock0.text / stock1.text / stock2.text / stock3.text coorespondent les coordonnées de barycentre de chaque cellules dans chaque image 3D.

Le fichier Trajectoire.obj est pour la visualisation de trajectoire des cellules.

## C. Rendu et explication des exercices

## 1. Visualisation L'image

- Visualisation des images de dossier DATA1 (4 pas de temps : stack0-3):

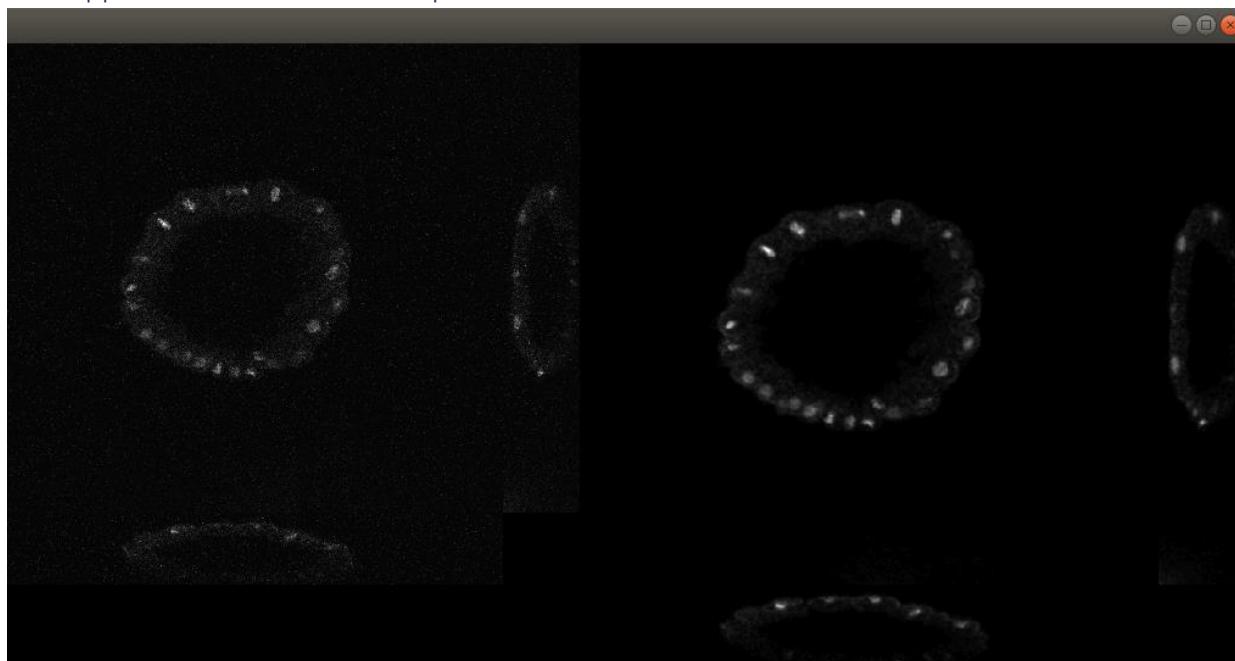


- Visualisation des images de dossier DATA2

## 2. Détection des cellules dans les images 3D

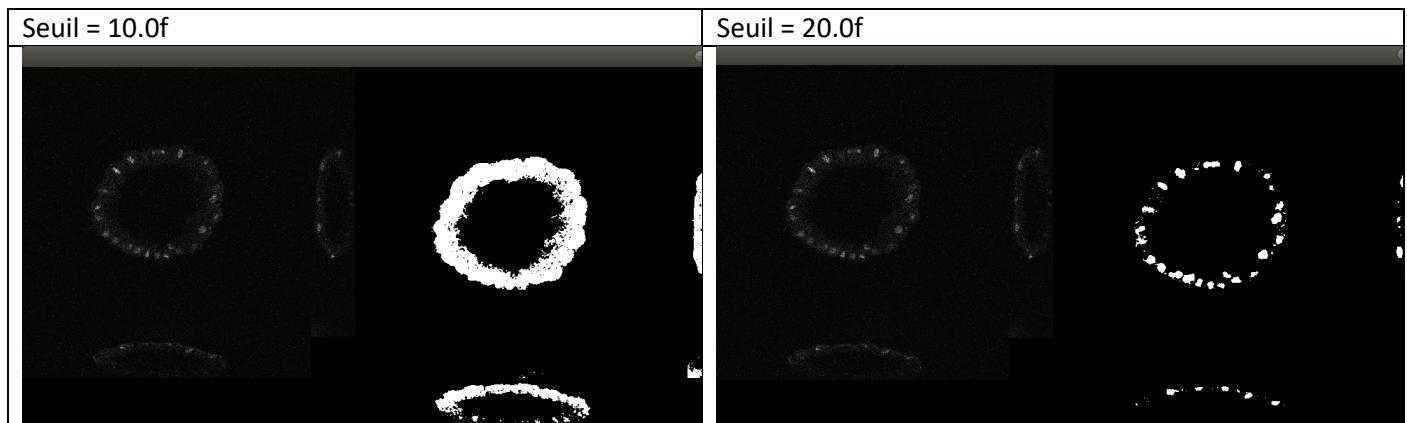
Pour cet étape, il faut faire les opérations pour chaque pas de temps. Pour commencer, prenons la première image 3D (stack0 dans DATA1) pour vous illustrer les opérations effectuées. Vous trouverez à la fin de cette étape, l'image obtenu pour l'ensemble de ces images.

### 2.1 Application du filtre médian pour éliminer les bruits



On constate que les petits points (bruits) sont bien éliminés.

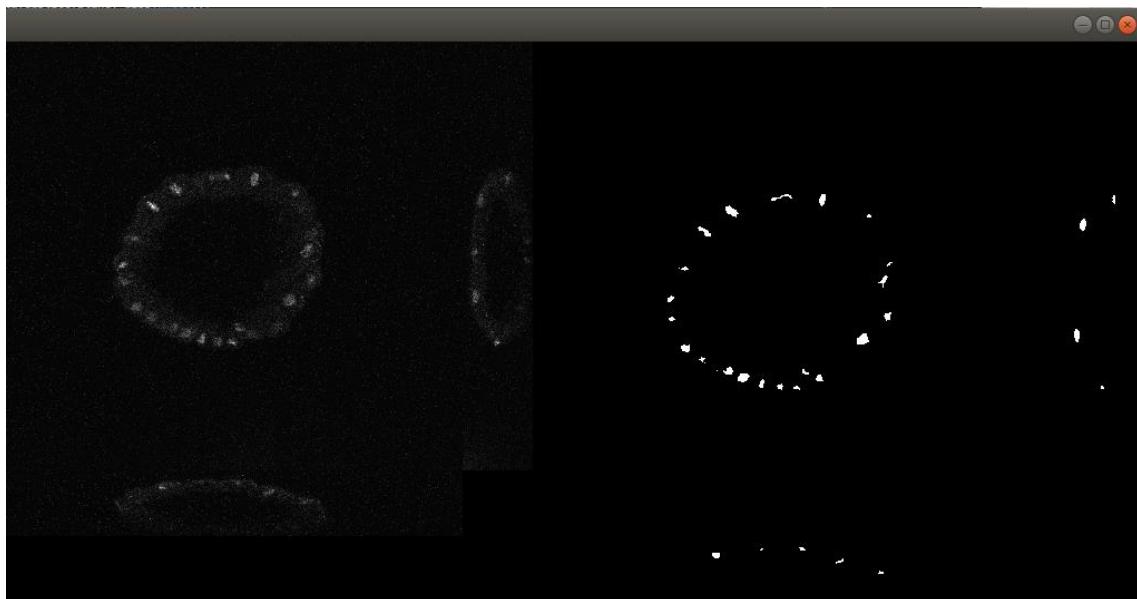
## 2.2 Seuillage pour éliminer le fond et ne garder que les noyaux cellulaires



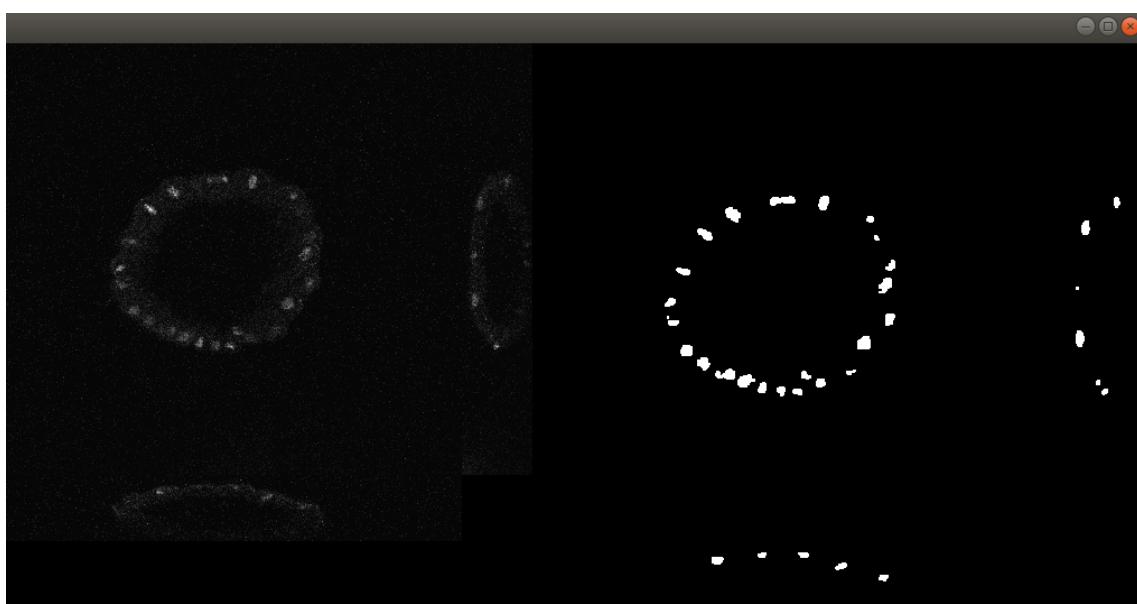
## 2.3 Éliminer les quelques pixels isolés par morphologie mathématique

Plusieurs essais pour appliquer l'érosion pour but d'éliminer les pixels isolés :

Seuil = 23, nombre de fois d'érosion = 3



Seuil = 22, nombre de fois d'érosion + dilatation = 3



Avec les paramètres ci-dessus, on obtient un résultat relativement idéal. On va appliquer les opérations avec ces paramètres pour l'ensemble des images.

## 2.4 Identifier les cellules par composantes connexes

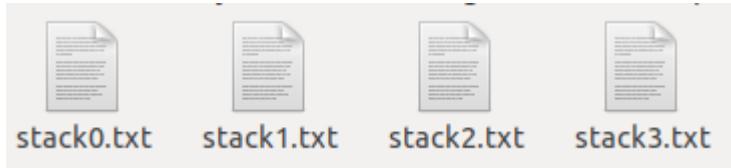
Pas de visualisation pour cet étape. Pour identifier les cellules, j'ai utilisé la fonction `get_label()` de librairie Cimg pour labéliser toutes les parties connexes pour but d'identifier les cellules différentes.

## 2.5 Calculer le barycentre de chacune des cellules et le sauvegarder dans un fichier

Après avoir identifié les différentes cellules, on pourra calculer le barycentre de chacune. Ici, l'idée est de collecter tous les pixels de chaque cellules dans un vector, et pour chaque cellule, on calcule simplement la moyenne de ses valeurs de positions (x,y,z).

Après un petit programme d'écriture de fichier, j'obtiens les fichiers correspondants à chaque image 3D.

(eg pour DATA1)



Dans ces fichiers (voir Annexe), on a bien les coordonnées de barycentre pour chaque cellule. (Exemple comme l'image ci-contre pour l'image stack0)

Remarque : les coordonnées sont sauvegardé au format :

*Num\_cellule : coordonnée\_x, coordonnée\_y, coordonnée\_z*

Donc à la fin de cet étape, on obtiens les images comme celle-ci :

```
visu5.cpp x stack0.txt x
1 : 261.630554, 249.134438, 13.967118
2 : 252.818832, 303.227173, 14.071172
3 : 234.712418, 221.491738, 14.583448
4 : 221.819748, 247.408585, 14.607725
5 : 246.878228, 276.344696, 14.978328
6 : 229.961639, 285.248814, 14.781224
7 : 267.481567, 219.901428, 16.173063
8 : 224.543533, 196.297684, 18.431593
9 : 183.537704, 267.174286, 16.781212
10 : 283.597595, 284.551941, 17.343664
11 : 194.789688, 299.265869, 19.267460
12 : 196.571747, 227.539536, 19.795021
13 : 177.883162, 253.823059, 19.036335
14 : 179.000000, 262.000000, 16.000000
15 : 230.977005, 323.192841, 18.841489
16 : 273.000000, 329.944031, 22.972486
17 : 252.968597, 188.893513, 55.588802
18 : 293.162994, 222.216644, 20.850287
19 : 281.845398, 187.320770, 22.275362
20 : 300.032043, 295.999420, 24.153618
21 : 198.213104, 197.734024, 22.424265
22 : 174.997589, 226.840149, 25.022837
23 : 295.153931, 258.834076, 24.906056
24 : 239.059769, 176.645401, 25.676899
25 : 191.261353, 336.761536, 34.166256
26 : 308.996399, 212.576782, 27.347513
27 : 152.403427, 261.444916, 27.230330
28 : 156.527359, 279.023224, 27.253731
29 : 227.451431, 342.195129, 29.321608
30 : 248.875009, 341.146729, 25.315218
31 : 302.394287, 181.319717, 30.316650
32 : 181.005524, 194.123627, 28.995584
33 : 172.683542, 308.991882, 27.864557
34 : 213.467209, 177.344940, 30.291498
35 : 317.655649, 241.084564, 30.330883
36 : 207.856781, 347.177826, 37.17207
37 : 272.791931, 165.714371, 35.330421
38 : 158.894913, 213.144913, 32.359203
39 : 305.046112, 324.211731, 30.813417
40 : 250.919662, 349.481995, 31.462605
41 : 147.506912, 234.633850, 44.150635
42 : 316.784851, 283.354553, 32.045456
43 : 236.332870, 355.531097, 37.838005
44 : 329.861237, 269.583679, 33.738614
45 : 239.215668, 161.706928, 38.479511
46 : 134.400925, 285.177277, 35.836617
47 : 336.699426, 197.611633, 35.136959
48 : 147.854553, 313.525452, 44.149693
49 : 302.798523, 340.471344, 41.256008
50 : 192.286102, 174.814346, 38.459194
51 : 165.456848, 196.235794, 37.878948
52 : 131.915619, 266.037506, 37.734375
53 : 163.887604, 328.637604, 36.223839
54 : 275.847107, 348.455536, 37.257412
55 : 285.000000, 347.000000, 34.600000
56 : 323.781677, 186.566654, 37.431267
57 : 339.579834, 248.899200, 37.679501
58 : 262.247742, 341.740173, 38.063442
```

Line 1, Column 1



Toutes les images 3D sont seuillé et toutes les cellules sont identifiés avec les coordonées de barycentre sauvegardé, il nous reste qu'à faire le suivi de cellule.

### 3. Algorithme de suivi de cellule

L'algorithme de suivi de cellule consiste à comparer deux à deux les images 3D, et chercher pour chaque cellule, le barycentre d'une autre cellule la plus proche. Pour faire cela, on va faire à partir les fichiers générés (les coordonnées d'ensemble de barycentre pour chaque image).

En pratique, j'ai crée un vector pour sauvegarder les trajectoires.

```
std::vector<int> tmp(nb_fichier, -1);
std::vector<std::vector<int>> trajectoire(max_cellule, tmp);
```

Par exemple, pour les cellules identifiés, si la cellule 0 de l'image (stack0) a est plus proche de la cellule 1 dans l'image (stack1) et qui ensuite a la cellule plus proche de cellule 1 dans l'image (stack2) et aussi la cellule plus proche de cellule 2 dans l'image(stack3)

Donc dans le vector trajectoire :

Pour la cellule 0 :

Trajectoire[0] : [ 0, 1, 1, 2 ]

...

Trajectoire[max\_cellule] : ...

Remarque : ici, les données 0,1,1,2 sont les indices dans le vector de barycentre. Donc grâce au vector de barycentre (sauvegardé par l'étape précédent), on pourra savoir les coordonnées exactes de chaque cellule identifiée.

```
File Edit Selection Find View Goto Tools Project
visu5.cpp Trajectoire.obj
1 v 261.630554 249.134430 13.967118
2 v 261.867554 248.791397 14.156733
3 v 219.847000 248.622955 14.938069
4 v 233.970673 221.042526 14.703813
5 l 1 2 3 4
6 v 252.818832 303.227173 14.071172
7 v 252.282135 302.954254 14.532680
8 v 262.364380 249.105698 14.559110
9 v 253.016449 302.694092 14.245066
10 l 5 6 7 8
11 v 234.712418 221.491730 14.583448
12 v 234.746658 221.168060 14.624582
13 v 251.451920 303.814575 14.322803
14 v 225.432770 250.470581 14.407563
15 l 9 10 11 12
16 v 221.819748 247.408585 14.607725
17 v 221.268387 247.627690 14.652904
18 v 234.865341 222.761124 14.485949
19 v 257.323730 249.230209 13.798561
20 l 13 14 15 16
21 v 246.878220 276.344696 14.978328
22 v 246.723557 276.276459 15.002275
23 v 247.083817 275.908142 15.113663
24 v 243.314285 272.714294 13.314285
25 l 17 18 19 20
26 v 220.961639 286.240814 14.701224
27 v 219.226334 288.997681 15.464203
28 v 218.897430 289.320526 15.548717
29 v 222.000000 293.000000 17.000000
30 l 21 22 23 24
31 v 267.481567 219.901428 16.173063
32 v 266.852356 217.892659 16.208437
33 v 266.742157 217.570267 16.477489
34 v 268.049225 218.676437 16.597889
35 l 25 26 27 28
36 v 224.543533 196.297684 18.431593
37 v 225.136520 195.926254 18.575985
38 v 225.821014 196.291183 18.955254
39 v 188.612503 298.200012 19.200000
40 l 29 30 31 32
41 v 183.537704 267.174286 16.781212
42 v 180.460526 261.322357 18.210526
43 v 188.571426 298.920624 17.015873
44 v 225.584259 195.056458 19.147989
45 l 33 34 35 36
46 v 283.597595 284.551941 17.343664
47 v 283.024841 284.669667 17.641596
48 v 180.668274 261.985565 18.516346
49 v 183.186142 267.376251 18.009901
50 l 37 38 39 40
51 v 194.789688 299.265869 19.267460
52 v 189.626083 298.130432 17.547827
53 v 282.959106 285.816589 17.788551
54 v 285.144928 283.901184 17.255600
55 l 41 42 43 44
56 v 177.883102 253.823059 19.036335
57 v 180.460526 261.322357 18.210526
58 v 229.845337 322.687988 19.626667
```

### 4. Visualisation de trajectoire 3D

Pour la visualisation, j'ai fait sauvegarder dans un fichier obj, les coordonnées de barycentre de chaque cellule pour chaque trajectoire calculé. Le fichier est comme l'image ci-contre.

J'ai utilisé le logiciel Paraview pour visualiser le fichier obj. J'ai obtenu le résultat comme l'image suivante. (Affichage sur les points gaussiens)

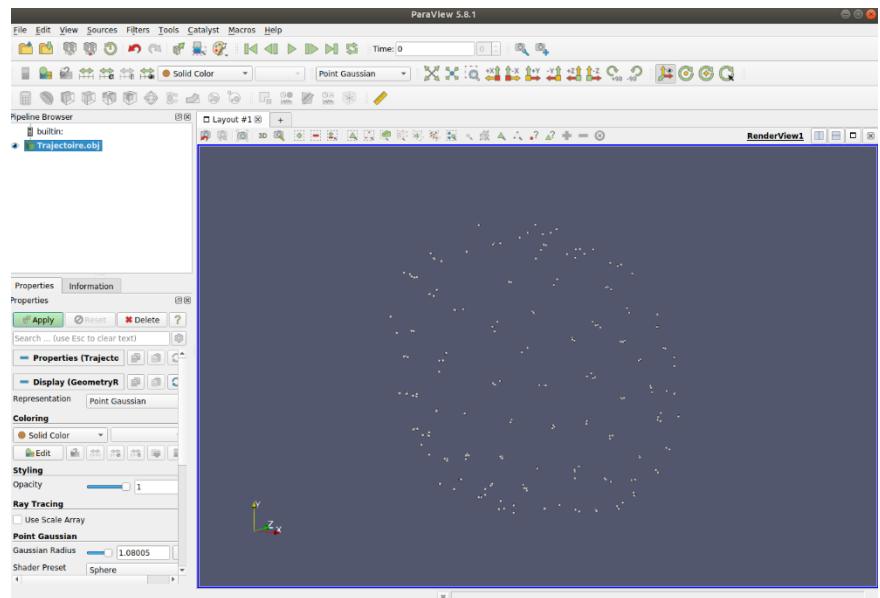
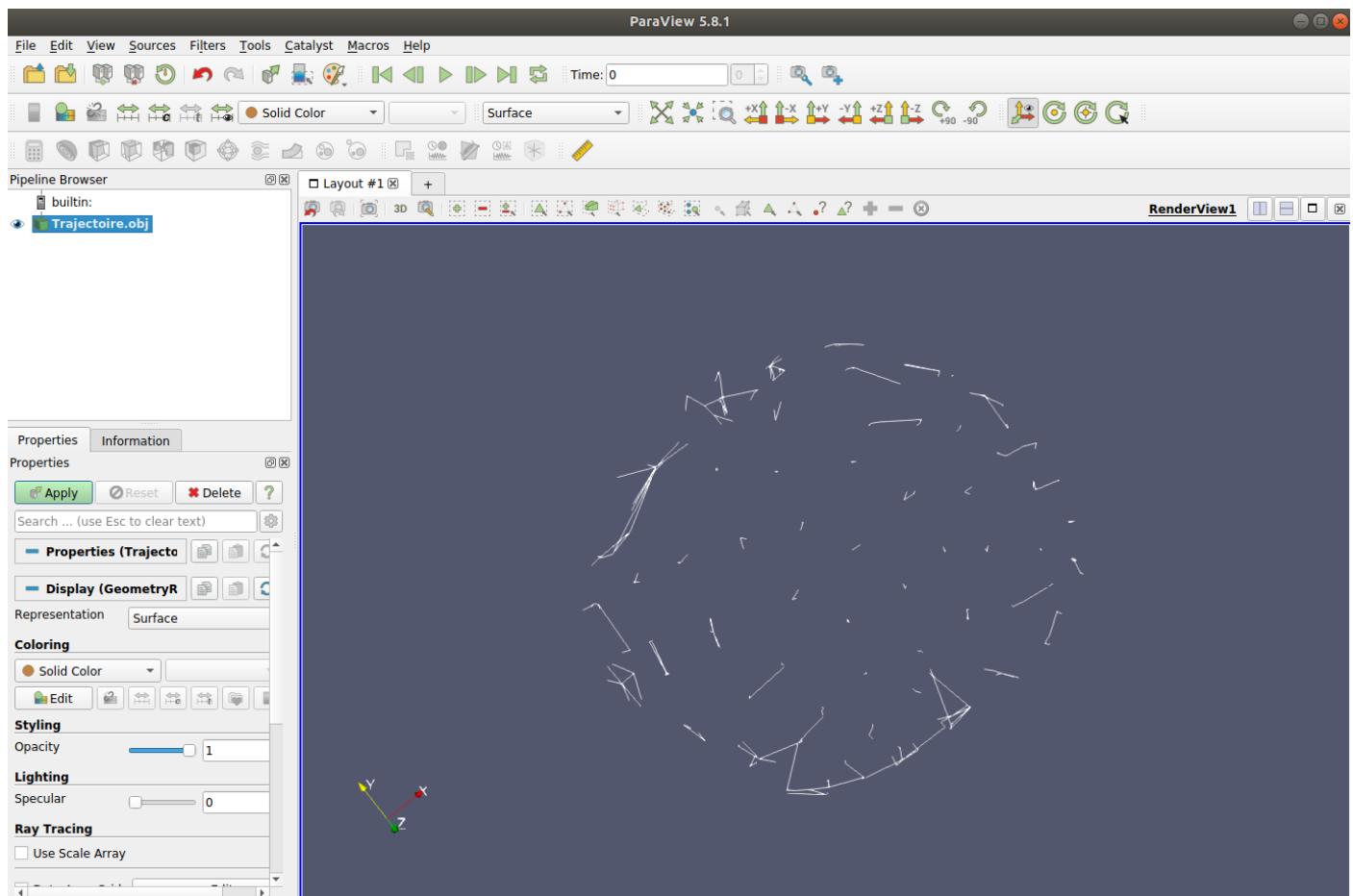
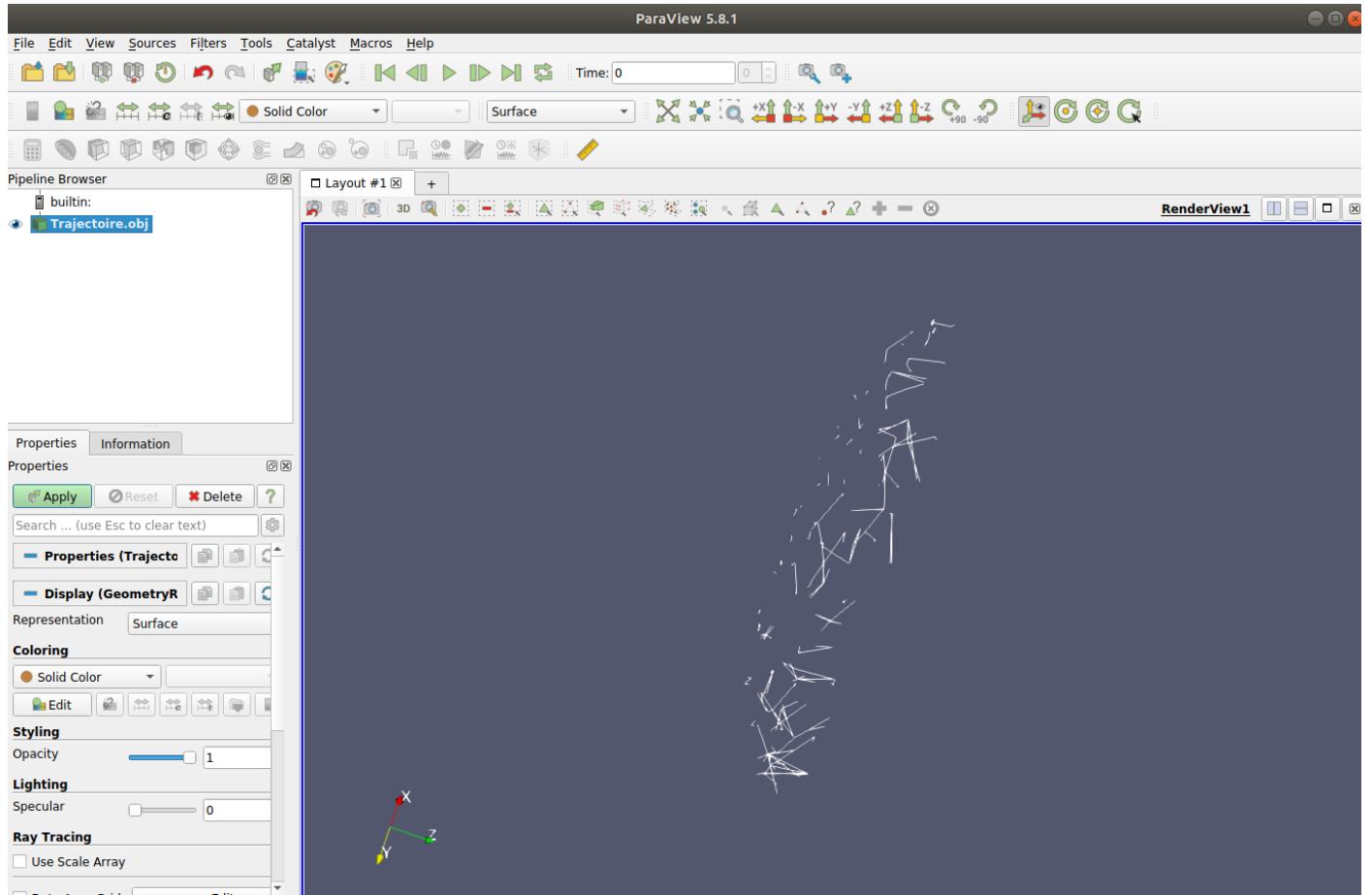


Image d'affichage pour mode surface





On constate que certaines edges sont assez emmelés. Je ne sais pas si cela correspond bien le mouvement dans le temps pour chaque cellule.

Remarque : je me suis bien aperçu que le nombre de cellules sont différents selon les différentes images 3D. Cela pourrait dépendre de la stratégie de seuillage ou de la morphologie mathématique (nombre de fois de convolution) mais aussi pourrait dépendre de l'apparition et la disparition de cellule dans le temps. Donc ce serait mieux d'améliorer le program pour bien différencier les différentes cellules de durée différentes. Ici à cause de manque de temps, je n'ai pas réalisé cela.

## Annexe

### 1. Code source du Program principal

Pour faciliter la lecture : Le code avec le fond bleu corresponds les questions demandées (le reste est pour l'initialisation et la fenêtre ou d'autres choses.)

```
/*
 * HMIN318
 * Compilation:
 * (Linux version) g++ -o main.exe main.cpp -O2 -L/usr/X11R6/lib -lm -lpthread -lX11

 * Execution :
 * ./main.exe
 */

#include "CImg.h"
#include <iostream>
#include <limits>
#include <string>
#include <vector>
#include <stdio.h>
#include <stdlib.h>

using namespace cimg_library;

float seuil = 22.0f;
float n_ero = 3.0;

void MIP(CImg<float>& img_in, CImg<float>& img_out){

    for(int y = 0 ; y < img_in.height() ; y++)
    {
        for(int x = 0 ; x < img_in.width() ; x++)
        {
            float current_max = std::numeric_limits<float>::min();
            int z_max = 0;

            for(int z = 0 ; z < img_in.depth() ; z++)
            {
                if(img_in(x, y, z) > current_max) {
                    current_max = img_in(x, y, z);
                    z_max = z;
                }
            }

            img_out(x, y, z_max) = current_max;
        }
    }
}

float distance (float x1, float y1, float z1, float x2, float y2, float z2){
    return sqrt(pow((x2-x1),2)+pow((y2-y1),2)+pow((z2-z1),2));
}

/* Main program */
int main(int argc,char **argv)
```

```

{
/*
if(argc != 2)
{
    printf("Usage : %s filename.hdr\n", argv[0]);
    exit(EXIT_FAILURE);
}
*/
/* Create and load the 3D image */
CImg<float> img, img1, img2, img3;
float voxelsize[3];
/* Load in Analyze format and get the voxel size in an array */
img.load_analyze("DATA1/stack-0.hdr",voxelsize);
img1.load_analyze("DATA1/stack-1.hdr",voxelsize);
img2.load_analyze("DATA1/stack-2.hdr",voxelsize);
img3.load_analyze("DATA1/stack-3.hdr",voxelsize);

/* Get the image dimensions */
int dim[]={img.width(),img.height(),img.depth()};
printf("Reading %s. Dimensions=%d %d %d\n",argv[1],dim[0],dim[1],dim[2]);
printf("Voxel size=%f %f %f\n",voxelsize[0],voxelsize[1],voxelsize[2]);

unsigned int voxel_coord[] = {256, 256, 12};
printf("Voxel(%u,%u,%u) = %lf\n", voxel_coord[0], voxel_coord[1], voxel_coord[2], img(voxel_coord[0],
voxel_coord[1], voxel_coord[2]));

/* Create the display window of size 512x512 */
CImgDisplay disp(512*4,512,"");
printf("disp.normalization() = %d\n", disp.normalization());

/* The 3 displayed slices of the MPR visualisation */
int displayedSlice[]={dim[0]/2,dim[1]/2,dim[2]/2};

/* Slice corresponding to mouse position: */
unsigned int coord[]={0,0,0};
/* The display window corresponds to a MPR view which is decomposed into the following 4 quadrants:
2=original slice size=x y      0 size=z y
1= size=x z                  -1 corresponds to the 4th quarter where there is nothing displayed */
int plane=2;

/* For a first drawing, activate the redrawing flag */
bool redraw=true;
std::string visu_mode = "MPR";
/* Manage the display windows: ESC, or closed -> close the main window */
while (!disp.is_closed() && !disp.is_keyESC()) // Main loop
{
    /* List of events */

    /* Resizing */
    if (disp.is_resized())
    {
        disp.resize();
    }

    if (disp.is_key("m"))
    {

```

```

        img.blur(0.5);
    }

    if (disp.is_key("r"))
    {
        visu_mode = "MPR";
    }

    if (disp.is_key("a"))
    {
        visu_mode = "MIP";
    }

    if (disp.is_key("z"))
    {
        visu_mode = "MinIP";
    }

    if (disp.is_key("e"))
    {
        visu_mode = "AIP";
    }

if(disp.mouse_x()>=0 && disp.mouse_y()>=0)
{
    unsigned int mX = disp.mouse_x()*(dim[0]+dim[2])/disp.width();
    unsigned int mY = disp.mouse_y()*(dim[1]+dim[2])/disp.height();
    if (mX>=dim[0] && mY<dim[1])
    {
        plane = 0;
        coord[1] = mY;
        coord[2] = mX - dim[0];
        coord[0] = displayedSlice[0];
    }
    else
    {
        if (mX<dim[0] && mY>=dim[1])
        {
            plane = 1;
            coord[0] = mX;
            coord[2] = mY - dim[1];
            coord[1] = displayedSlice[1];
        }
        else
        {
            if (mX<dim[0] && mY<dim[1])
            {
                plane = 2;
                coord[0] = mX;
                coord[1] = mY;
                coord[2] = displayedSlice[2];
            }
            else
            {
                plane = -1;
                coord[0] = 0;
            }
        }
    }
}

```

```

        coord[1] = 0;
        coord[2] = 0;
    }
}
redraw = true;
}

if (disp.button()&2 && (plane!=-1))
{
    for(unsigned int i=0;i<3;i++)
    {
        displayedSlice[i]=coord[i];
    }
    redraw = true;
}

if (disp.wheel())
{
    displayedSlice[plane]=displayedSlice[plane]+disp.wheel();

    if (displayedSlice[plane]<0)
    {
        displayedSlice[plane] = 0;
    }
    else
    {
        if (displayedSlice[plane]>=(int)dim[plane])
        {
            displayedSlice[plane] = (int)dim[plane]-1;
        }
    }
}

/* Flush all mouse wheel events in order to not repeat the wheel event */
disp.set_wheel();
redraw = true;
}

if (redraw)
{
    CImg<float> visu = img;
    CImg<float> mpr_img =
visu.get_projections2d(displayedSlice[0],displayedSlice[1],displayedSlice[2]);

    //2.1 -Utiliser un filtre(par exemple médian)pour éliminer le bruit dans les données

    CImg<> res = img.get.blur.median(3.0);
    CImg<> res1 = img1.get.blur.median(3.0);
    CImg<> res2 = img2.get.blur.median(3.0);
    CImg<> res3 = img3.get.blur.median(3.0);

    //2.2 -Utiliser un seuil pouréliminer le fond et de ne garder que les noyaux cellulaires.

    res = res.get.threshold(seuil);
    res1 = res1.get.threshold(seuil);
}

```

```

res2 = res2.get_threshold(seuil);
res3 = res3.get_threshold(seuil);

//2.3 -Éliminer les quelques pixels isolés par morphologie mathématique
res.erode(n_ero,n_ero,n_ero); res.dilate(n_ero,n_ero, n_ero);
res1.erode(n_ero,n_ero,n_ero); res1.dilate(n_ero,n_ero, n_ero);
res2.erode(n_ero,n_ero,n_ero); res2.dilate(n_ero,n_ero, n_ero);
res3.erode(n_ero,n_ero,n_ero); res3.dilate(n_ero,n_ero, n_ero);

//2.4 -Identifier les cellules par composantes connexes
CImg<> labels = res.get_label(false);
CImg<> labels1 = res1.get_label(false);
CImg<> labels2 = res2.get_label(false);
CImg<> labels3 = res3.get_label(false);

//2.5 -Calculer le barycentre de chacune des cellules et le sauvegarder dans un fichier
//compteur pour chaque labels de l'image
std::vector<int> compteur(labels.max() + 1 , 0 );
std::vector<int> compteur1(labels1.max() + 1 , 0 );
std::vector<int> compteur2(labels2.max() + 1 , 0 );
std::vector<int> compteur3(labels3.max() + 1 , 0 );
std::vector<float> coord = {0.0f,0.0f,0.0f};
//somme[i][0] : x somme[i][1] : y somme[i][2] : z
std::vector<std::vector<float>> somme(compteur.size(),coord);
std::vector<std::vector<float>> somme1(compteur1.size(),coord);
std::vector<std::vector<float>> somme2(compteur2.size(),coord);
std::vector<std::vector<float>> somme3(compteur3.size(),coord);

for(size_t x = 0; x < labels.width(); x++){
    for(size_t y = 0; y < labels.height(); y++){
        for(size_t z = 0; z < labels.depth(); z++){
            if (labels.atXYZ(x, y, z) > 0) { // Enlever label du fond
                //Pour image 0
                compteur[(int)labels.atXYZ(x,y,z)]++;
                somme[(int)labels.atXYZ(x,y,z)][0] += x;
                somme[(int)labels.atXYZ(x,y,z)][1] += y;
                somme[(int)labels.atXYZ(x,y,z)][2] += z;
                //Pour image 1
                compteur1[(int)labels1.atXYZ(x,y,z)]++;
                somme1[(int)labels1.atXYZ(x,y,z)][0] += x;
                somme1[(int)labels1.atXYZ(x,y,z)][1] += y;
                somme1[(int)labels1.atXYZ(x,y,z)][2] += z;
                //Pour image 2
                compteur2[(int)labels2.atXYZ(x,y,z)]++;
                somme2[(int)labels2.atXYZ(x,y,z)][0] += x;
                somme2[(int)labels2.atXYZ(x,y,z)][1] += y;
                somme2[(int)labels2.atXYZ(x,y,z)][2] += z;
                //Pour image 3
                compteur3[(int)labels3.atXYZ(x,y,z)]++;
                somme3[(int)labels3.atXYZ(x,y,z)][0] += x;
                somme3[(int)labels3.atXYZ(x,y,z)][1] += y;
                somme3[(int)labels3.atXYZ(x,y,z)][2] += z;
            }
        }
    }
}

```

```

//Écriture de fichiers pour sauvegarder les barycentres
//ici j'ai crée les vectors pour sauvegarder ces données aussi juste pour faciliter la tache
std::vector<std::vector<float>> barycentres(compteur.size(),coord);
std::vector<std::vector<float>> barycentres1(compteur1.size(),coord);
std::vector<std::vector<float>> barycentres2(compteur2.size(),coord);
std::vector<std::vector<float>> barycentres3(compteur3.size(),coord);
FILE* fichier = NULL;
int nb_fichier = 4;
std::cout << "Ecriture des fichiers en cours..." << std::endl;
for(int f = 0; f < nb_fichier; f++){
    std::string filename = "stack";
    char buffer[sizeof(int)]; sprintf(buffer,"%d", f);
    filename += buffer;
    filename += ".txt";
    std::cout << filename;
    fichier = fopen(filename.c_str(),"w+");
}

if (fichier != NULL)
{
    if(f == 0){
        for(int i = 0; i < somme.size(); i++){
            fprintf(fichier, "%d : %f, %f, %f \n", i,
                    somme[i][0] / (float) compteur[i], somme[i][1] / (float)
compteur[i], somme[i][2] / (float) compteur[i]);
            barycentres[i][0] = somme[i][0] / (float) compteur[i];
            barycentres[i][1] = somme[i][1] / (float) compteur[i];
            barycentres[i][2] = somme[i][2] / (float) compteur[i];
        }
        fclose(fichier);
    }
    if(f == 1){
        for(int i = 0; i < somme1.size(); i++){
            fprintf(fichier, "%d : %f, %f, %f \n", i,
                    somme1[i][0] / (float) compteur1[i], somme1[i][1] / (float)
compteur1[i], somme1[i][2] / (float) compteur1[i]);
            barycentres1[i][0] = somme1[i][0] / (float) compteur1[i];
            barycentres1[i][1] = somme1[i][1] / (float) compteur1[i];
            barycentres1[i][2] = somme1[i][2] / (float) compteur1[i];
        }
        fclose(fichier);
    }
    if(f == 2){
        for(int i = 0; i < somme2.size(); i++){
            fprintf(fichier, "%d : %f, %f, %f \n", i,
                    somme2[i][0] / (float) compteur2[i], somme2[i][1] / (float)
compteur2[i], somme2[i][2] / (float) compteur2[i]);
            barycentres2[i][0] = somme2[i][0] / (float) compteur2[i];
            barycentres2[i][1] = somme2[i][1] / (float) compteur2[i];
            barycentres2[i][2] = somme2[i][2] / (float) compteur2[i];
        }
        fclose(fichier);
    }
    if(f == 3){
        for(int i = 0; i < somme3.size(); i++){
            fprintf(fichier, "%d : %f, %f, %f \n", i,
                    somme3[i][0] / (float) compteur3[i], somme3[i][1] / (float)
compteur3[i], somme3[i][2] / (float) compteur3[i]);
        }
    }
}

```

```

        barycentres3[i][0] = somme3[i][0] / (float) compteur3[i];
        barycentres3[i][1] = somme3[i][1] / (float) compteur3[i];
        barycentres3[i][2] = somme3[i][2] / (float) compteur3[i];
    }
    fclose(fichier);
}
}

disp.display((res,res1,res2,res3));

//3. Algorithme de suivi de cellule
//On a <barycentre> qui corresponds à l'ensemble de coordonnée des cellules dans une
image
int max_cellule = compteur.size(); if(max_cellule < compteur1.size()) max_cellule =
compteur1.size();
if(max_cellule < compteur2.size()) max_cellule = compteur2.size();
if(max_cellule < compteur3.size()) max_cellule = compteur3.size();

std::vector<int> tmp(nb_fichier, -1);
std::vector<std::vector<int>> trajectoire(max_cellule, tmp);
for(int i = 0; i < compteur.size(); i++){trajectoire[i][0] = i;}
std::cout<<std::endl;
std::cout<<"Algorithme de suivi de cellule en cours..."<<std::endl;

for(int i = 0; i < nb_fichier; i++){
    if(i!=nb_fichier - 1) { //sauf pour la dernière image

        for(int j = 0; j < compteur.size(); j++){ // pour chaque cellule dans l'image de
départ

            if(i == 0){
                float min_dist = 10000.0f;
                for(int k = 0; k < compteur1.size(); k++){//pour chaque cellule
dans l'image 1
                    float current_dist = distance(barycentres[j][0],
barycentres[j][1], barycentres[j][2],
barycentres1[k][0], barycentres1[k][1], barycentres1[k][2]);
                    if(min_dist > current_dist){
                        min_dist = current_dist;
                        trajectoire[j][i+1] = k;
                    }
                }
            }
            if(i == 1){
                float min_dist = 10000.0f;
                for(int k = 0; k < compteur2.size(); k++){
                    float current_dist = distance(barycentres[j][0],
barycentres[j][1], barycentres[j][2],
barycentres2[k][0], barycentres2[k][1], barycentres2[k][2]);
                    if(min_dist > current_dist){
                        min_dist = current_dist;
                        trajectoire[j][i+1] = k;
                    }
                }
            }
        }
    }
}

```

```

        }

        if(i == 2){
            float min_dist = 10000.0f;
            for(int k = 0; k < compteur3.size(); k++){
                float current_dist = distance(barycentres[j][0],
barycentres[j][1], barycentres[j][2],

barycentres3[k][0], barycentres3[k][1], barycentres3[k][2]);
                if(min_dist > current_dist){
                    min_dist = current_dist;
                    trajectoire[j][i+1] = k;
                }
            }
        }
    }

    //Ecriture de fichier obj
    std::cout << "Ecriture de fichier OBJ..."<<std::endl;
    FILE* fichierObj = NULL;
    int counter = 1;

    fichierObj = fopen("Trajectoire.obj", "w+");

    if (fichier != NULL)
    {
        for(int i = 0; i < trajectoire.size(); i++){
            if(trajectoire[i][0]==-1 || trajectoire[i][1]==-1 || trajectoire[i][2]==-1 ||
trajectoire[i][3]==-1 ) continue;
            fprintf(fichierObj, "v %f %f %f\nv %f %f %f\nv %f %f %f\nv %f %f %f\nl %d %d %d
\n",
barycentres[trajectoire[i][0]][0],barycentres[trajectoire[i][0]][1],barycentres[trajectoire[i][0]][2],
barycentres1[trajectoire[i][1]][0],barycentres1[trajectoire[i][1]][1],barycentres1[trajectoire[i][1]][2],
barycentres2[trajectoire[i][2]][0],barycentres2[trajectoire[i][2]][1],barycentres2[trajectoire[i][2]][2],
barycentres3[trajectoire[i][3]][0],barycentres3[trajectoire[i][3]][1],barycentres3[trajectoire[i][3]][2],
counter, counter+1, counter+2, counter+3);
            counter+=4;
        }

        fclose(fichier);
    }

    redraw=false;
}

}

return 0;
}

```

## 2. Fichier pour les données de barycentres

Stack0.txt	Stack1.txt
0 : -nan, -nan, -nan	0 : 229.385117, 258.377106, 34.119061
1 : 261.630554, 249.134430, 13.967118	1 : 221.268387, 247.627090, 14.652904
2 : 252.818832, 303.227173, 14.071172	2 : 261.867554, 248.791397, 14.156733
3 : 234.712418, 221.491730, 14.583448	3 : 252.282135, 302.954254, 14.532680
4 : 221.819748, 247.408585, 14.607725	4 : 234.746658, 221.168060, 14.624582
5 : 246.878220, 276.344696, 14.978328	5 : 246.723557, 276.276459, 15.002275
6 : 220.961639, 286.240814, 14.701224	6 : 219.226334, 288.997681, 15.464203
7 : 267.481567, 219.901428, 16.173063	7 : 266.852356, 217.892059, 16.208437
8 : 224.543533, 196.297684, 18.431593	8 : 225.136520, 195.926254, 18.575985
9 : 183.537704, 267.174286, 16.781212	9 : 189.626083, 298.130432, 17.547827
10 : 283.597595, 284.551941, 17.343664	10 : 180.460526, 261.322357, 18.210526
11 : 194.789688, 299.265869, 19.267460	11 : 283.024841, 284.669067, 17.641596
12 : 196.571747, 227.539536, 18.795021	12 : 252.794098, 188.164551, 19.751055
13 : 177.883102, 253.823059, 19.036335	13 : 230.585541, 323.261047, 18.982302
14 : 179.000000, 262.000000, 16.000000	14 : 293.210754, 221.293716, 20.326233
15 : 230.977005, 323.192841, 18.844189	15 : 196.221558, 227.185623, 19.173653
16 : 273.000000, 329.944031, 22.972486	16 : -nan, -nan, -nan
17 : 252.968597, 188.803513, 19.558802	17 : 273.168640, 329.907288, 23.584318
18 : 293.162994, 222.216644, 20.050207	18 : 282.106110, 186.279266, 22.147560
19 : 281.845398, 187.320770, 22.275362	19 : 299.469574, 295.656006, 24.810452
20 : 300.032043, 295.999420, 24.153618	20 : 198.652374, 196.712097, 23.274120
21 : 198.213104, 197.734024, 22.424625	21 : 291.441559, 256.337677, 24.948051
22 : 174.997589, 226.840149, 25.022837	22 : 167.372543, 305.764709, 26.044117
23 : 295.153931, 258.834076, 24.900656	23 : 174.392975, 226.562302, 25.303514
24 : 239.059769, 176.645401, 25.676899	24 : 239.588806, 174.395203, 26.268801
25 : 191.261353, 336.761536, 34.166256	25 : 150.411270, 261.625366, 26.242254
26 : 308.996399, 212.576782, 27.347513	26 : 155.238327, 279.312042, 26.673218
27 : 152.403427, 261.444916, 27.230330	27 : 165.277924, 327.793152, 39.437817
28 : 156.527359, 279.023224, 27.253731	28 : 210.242386, 344.175476, 37.759895
29 : 227.451431, 342.195129, 29.321608	29 : 308.911530, 210.386063, 27.793566
30 : 248.875000, 341.146729, 25.315218	30 : 297.285706, 263.914276, 25.028572
31 : 302.394287, 181.319717, 30.316650	31 : 251.333328, 347.000000, 28.817461
32 : 181.005524, 194.123627, 28.995584	32 : 275.714294, 162.040817, 34.081635
33 : 172.083542, 308.991882, 27.864557	33 : 212.081604, 176.135208, 30.819733
34 : 213.467209, 177.344940, 30.291498	34 : 303.323029, 180.508499, 30.731066
35 : 317.655640, 241.084564, 30.330883	35 : 180.662338, 192.850647, 30.120131
36 : 207.856781, 347.177826, 37.171207	36 : 317.973297, 240.384338, 30.781139
37 : 272.791931, 165.714371, 35.330421	37 : -nan, -nan, -nan
38 : 158.894913, 213.144913, 32.359203	38 : 147.529709, 238.692078, 33.227722
39 : 305.046112, 324.211731, 30.813417	39 : 318.561157, 283.323730, 32.244606
40 : 250.919662, 349.481995, 31.462605	40 : -nan, -nan, -nan
41 : 147.506912, 234.633850, 44.150635	41 : 160.257797, 202.508881, 37.396065
42 : 316.784851, 283.354553, 32.045456	42 : 329.856506, 269.451019, 33.380409
43 : 236.332870, 355.531097, 37.838005	43 : 306.437958, 324.583954, 31.474453
44 : 329.861237, 269.583679, 33.730614	44 : 135.229401, 275.408234, 41.636078
45 : 239.215668, 161.700928, 38.479511	45 : 150.002151, 312.880646, 41.261292
46 : 134.400925, 285.177277, 35.836617	46 : 246.047180, 160.946732, 44.178082
47 : 330.099426, 197.611633, 35.136959	47 : 190.935059, 173.796539, 38.608948
48 : 147.854553, 313.525452, 44.149693	48 : 339.004517, 250.450150, 37.524170
49 : 302.798523, 340.471344, 41.256008	49 : 321.077393, 304.910706, 37.797619
50 : 192.286102, 174.814346, 38.459194	50 : 278.615631, 348.592834, 38.732899
51 : 165.456848, 196.235794, 37.878948	51 : 329.424133, 196.813232, 36.128407
52 : 131.915619, 266.037506, 37.734375	52 : -nan, -nan, -nan
53 : 163.887604, 328.637604, 36.223839	53 : -nan, -nan, -nan
54 : 275.847107, 348.455536, 37.257412	54 : 324.309998, 180.130005, 37.973331
55 : 285.000000, 347.000000, 34.000000	55 : 307.851410, 338.577637, 43.312187
56 : 323.781677, 180.560654, 37.431267	56 : -nan, -nan, -nan
57 : 339.579834, 248.899200, 37.679501	57 : 265.079987, 343.230011, 38.779999
58 : 262.247742, 341.740173, 38.063442	58 : -nan, -nan, -nan
59 : 318.820587, 308.182068, 38.415565	59 : 344.371552, 281.856262, 40.888378
60 : 225.694199, 347.947968, 47.239269	60 : 339.111115, 225.925919, 41.222221
61 : 254.719131, 357.891968, 38.302467	61 : 240.857147, 361.178558, 40.946430
62 : 344.147461, 282.451752, 40.821716	62 : 258.666656, 360.666656, 39.833332
63 : 342.919708, 226.737228, 41.284672	63 : 148.996292, 232.535187, 53.085186
64 : 284.000000, 350.000000, 41.000000	64 : -nan, -nan, -nan
65 : 341.788544, 294.691620, 44.015419	65 : 222.888885, 158.155548, 45.222221
66 : 222.994019, 163.025406, 45.073242	66 : 344.753418, 229.780823, 42.219177
67 : 176.110855, 170.445755, 46.457546	67 : 341.328888, 294.556152, 44.344921
68 : 337.173340, 187.328003, 45.175999	68 : 338.986664, 185.546661, 45.193333
69 : 305.173065, 159.956726, 45.725960	69 : 304.406464, 159.883865, 46.045162
70 : 346.216827, 203.699112, 46.216816	70 : 175.208481, 170.293289, 46.989399
71 : 144.000000, 314.333344, 43.333332	71 : 210.432434, 160.918915, 49.486488
72 : 318.643402, 168.790695, 45.651161	72 : 317.786652, 169.053329, 45.813332

73 : 161.398193, 183.724319, 47.445045 74 : 132.516068, 282.891724, 48.275803 75 : 160.925919, 328.539886, 47.002850 76 : 274.875000, 344.750000, 45.625000 77 : 160.924454, 209.915115, 50.562305 78 : 352.734406, 243.888000, 48.716801 79 : 194.734238, 352.509003, 47.468468 80 : 211.422104, 166.175720, 49.661232 81 : 136.426590, 263.708130, 49.364422 82 : 255.092850, 162.994644, 49.799999 83 : 351.220154, 270.431000, 50.998451 84 : 316.509003, 331.810822, 49.954956 85 : 334.623474, 300.117401, 51.655872 86 : 282.574463, 158.765961, 52.489361 87 : 278.960785, 345.509796, 52.058823 88 : 194.329071, 169.060333, 54.705666 89 : 243.887100, 162.225800, 54.048386 90 : 351.955444, 226.495056, 54.336632 91 : 190.871628, 338.299561, 54.943695 92 : 180.620773, 180.103867, 55.961353 93 : 336.727264, 181.772720, 54.500000 94 : 245.144577, 343.771088, 56.228916 95 : 217.060974, 171.890244, 57.243904 96 : 147.494629, 291.333344, 57.596775 97 : 250.000000, 342.000000, 57.000000 98 : 346.404755, 265.357147, 60.227890 99 : 324.500000, 315.500000, 58.000000 100 : 184.399994, 331.942871, 58.328571 101 : 251.500000, 196.500000, 65.500000	73 : 348.803925, 202.588242, 46.509804 74 : 353.212860, 244.761047, 48.797188 75 : 156.653336, 208.053329, 51.026669 76 : 245.053696, 348.157715, 50.154362 77 : 351.488373, 271.004639, 51.560467 78 : 337.076935, 302.307678, 52.743591 79 : 282.377563, 159.438782, 52.020409 80 : 193.834747, 168.813553, 54.766949 81 : 352.304047, 226.972977, 54.587837 82 : 243.144821, 162.496552, 54.013794 83 : 335.799988, 182.399994, 53.200001 84 : 279.782623, 346.652161, 52.956520 85 : 180.949203, 180.168442, 56.128342 86 : 151.000000, 290.000000, 58.000000 87 : -nan, -nan, -nan 88 : 217.578949, 171.315796, 58.078949 89 : -nan, -nan, -nan 90 : 325.086945, 315.173920, 57.956520 91 : -nan, -nan, -nan 92 : 346.115082, 264.940491, 60.261906 93 : -nan, -nan, -nan 94 : -nan, -nan, -nan 95 : -nan, -nan, -nan 96 : -nan, -nan, -nan 97 : -nan, -nan, -nan 98 : -nan, -nan, -nan 99 : -nan, -nan, -nan 100 : -nan, -nan, -nan 101 : -nan, -nan, -nan 102 : -nan, -nan, -nan
Stack2.txt	Stack3.txt
0 : 232.303787, 257.101410, 33.090412 1 : 234.865341, 222.761124, 14.485949 2 : 251.451920, 303.814575, 14.322803 3 : 219.847000, 248.622955, 14.938069 4 : 262.364380, 249.105698, 14.559110 5 : 247.083817, 275.908142, 15.113663 6 : 218.897430, 289.320526, 15.548717 7 : 266.742157, 217.570267, 16.477489 8 : 188.571426, 298.920624, 17.015873 9 : 225.821014, 196.291183, 18.955254 10 : 180.668274, 261.985565, 18.516346 11 : 282.959106, 285.816589, 17.788551 12 : -nan, -nan, -nan 13 : 293.568634, 223.351547, 20.114845 14 : 229.845337, 322.687988, 19.626667 15 : 196.571228, 225.460434, 19.435970 16 : 253.820694, 188.008743, 20.612246 17 : 284.545441, 191.568176, 23.386364 18 : 297.547729, 255.432159, 23.532663 19 : 273.722229, 321.500000, 18.833334 20 : 266.000000, 324.000000, 21.000000 21 : 293.857147, 300.428558, 21.142857 22 : 196.874329, 196.500900, 23.177738 23 : 300.723694, 296.013153, 24.976316 24 : 282.153839, 181.307693, 22.230770 25 : 309.500000, 211.435959, 27.371922 26 : 174.043808, 225.973328, 25.906666 27 : 167.757141, 306.328583, 26.260714 28 : 151.651703, 261.454224, 27.296230 29 : -nan, -nan, -nan 30 : 240.756104, 174.084015, 27.073172 31 : 141.859650, 275.005859, 37.662117 32 : 165.923630, 327.192719, 40.109093 33 : 209.210678, 343.233673, 38.287994 34 : 278.500000, 158.000000, 35.000000 35 : 308.250000, 178.750000, 29.458334 36 : -nan, -nan, -nan 37 : 250.172134, 341.180328, 26.680328 38 : 277.616791, 342.547455, 31.007299 39 : 208.292450, 173.872635, 30.509434 40 : -nan, -nan, -nan 41 : 316.735382, 242.477661, 29.725086 42 : 178.804077, 193.922455, 30.669388 43 : 321.285706, 237.571426, 30.714285 44 : -nan, -nan, -nan 45 : -nan, -nan, -nan 46 : -nan, -nan, -nan	0 : 232.855881, 254.826965, 31.693153 1 : 257.323730, 249.230209, 13.798561 2 : 243.314285, 272.714294, 13.314285 3 : 253.016449, 302.694092, 14.245066 4 : 233.970673, 221.042526, 14.703813 5 : -nan, -nan, -nan 6 : -nan, -nan, -nan 7 : -nan, -nan, -nan 8 : 225.432770, 250.470581, 14.407563 9 : 216.751999, 283.600006, 15.128000 10 : 285.144928, 283.901184, 17.255600 11 : 268.049225, 218.676437, 16.597889 12 : 225.584259, 195.056458, 19.147989 13 : 222.000000, 293.000000, 17.000000 14 : 294.166962, 224.116516, 19.926956 15 : 188.612503, 298.200012, 19.200001 16 : -nan, -nan, -nan 17 : 183.186142, 267.376251, 18.009901 18 : 285.750000, 189.916672, 21.750000 19 : 299.232269, 256.283875, 23.025806 20 : 231.240967, 322.433746, 20.337349 21 : 254.465805, 186.390015, 20.905729 22 : 196.688843, 225.710220, 20.418053 23 : 303.623840, 294.454132, 24.483946 24 : -nan, -nan, -nan 25 : 268.846161, 328.730774, 20.903847 26 : 178.203568, 253.425003, 20.825001 27 : -nan, -nan, -nan 28 : 299.000000, 299.000000, 21.000000 29 : 196.304871, 194.073166, 24.101625 30 : -nan, -nan, -nan 31 : -nan, -nan, -nan 32 : 311.352936, 212.775085, 27.397924 33 : 321.600006, 241.440002, 27.959999 34 : 152.541290, 260.976135, 28.025688 35 : 170.058487, 304.647156, 27.275473 36 : 213.814011, 344.062714, 40.504356 37 : -nan, -nan, -nan 38 : 241.037430, 172.689835, 27.294117 39 : 321.594177, 179.986740, 36.705570 40 : -nan, -nan, -nan 41 : 156.480682, 278.856232, 28.051502 42 : 170.595108, 325.772125, 41.030132 43 : -nan, -nan, -nan 44 : 253.132355, 349.649506, 33.455883 45 : 275.391174, 342.082184, 37.138508 46 : 275.799988, 159.199997, 34.533333

47 : 156.016129, 212.577057, 33.057346	47 : 208.500000, 170.000000, 30.000000
48 : 151.057236, 236.584335, 33.222893	48 : 178.770264, 173.974899, 49.774132
49 : 143.437500, 239.518753, 36.562500	49 : -nan, -nan, -nan
50 : -nan, -nan, -nan	50 : -nan, -nan, -nan
51 : 320.078430, 284.235291, 32.274509	51 : -nan, -nan, -nan
52 : 309.152557, 322.915253, 31.762712	52 : 156.759995, 213.127136, 33.088570
53 : 251.920532, 350.264893, 32.172184	53 : 145.131317, 235.670029, 36.410774
54 : 329.622223, 197.547226, 35.458332	54 : 333.466675, 270.833344, 33.516666
55 : 331.822662, 269.748779, 33.408867	55 : 308.294128, 322.852936, 32.049019
56 : 239.940445, 159.178680, 38.313480	56 : 349.930237, 246.586044, 42.897675
57 : 341.613403, 245.860825, 36.108246	57 : 240.095810, 160.556885, 48.350300
58 : -nan, -nan, -nan	58 : 329.978027, 197.789017, 35.562637
59 : 151.214691, 311.902069, 43.180790	59 : -nan, -nan, -nan
60 : 170.485168, 177.395126, 45.952332	60 : -nan, -nan, -nan
61 : 307.633728, 337.569763, 36.819767	61 : 307.718750, 164.041672, 46.302082
62 : -nan, -nan, -nan	62 : 137.201920, 285.214752, 38.586540
63 : 323.872498, 180.315430, 37.899330	63 : 151.359741, 311.010712, 48.158459
64 : 267.582092, 343.298492, 38.865673	64 : -nan, -nan, -nan
65 : 343.384613, 224.778839, 39.865383	65 : 161.800949, 184.867294, 47.398106
66 : 354.612915, 245.788528, 48.584229	66 : 306.830383, 337.500793, 45.334412
67 : -nan, -nan, -nan	67 : 346.018188, 223.890915, 39.836365
68 : 213.992126, 161.283463, 49.299213	68 : 137.762344, 264.438263, 44.935184
69 : 160.269730, 207.640350, 52.010963	69 : 348.289764, 278.551239, 44.968197
70 : 257.802185, 358.516479, 39.098900	70 : -nan, -nan, -nan
71 : 347.348358, 279.016510, 43.972973	71 : 337.037170, 187.753372, 44.824326
72 : 244.439331, 353.297058, 45.794979	72 : -nan, -nan, -nan
73 : -nan, -nan, -nan	73 : 342.105835, 294.621155, 44.526463
74 : -nan, -nan, -nan	74 : 346.307678, 203.671799, 46.164104
75 : 341.779358, 294.636139, 44.169014	75 : 160.116440, 208.346573, 51.083561
76 : -nan, -nan, -nan	76 : -nan, -nan, -nan
77 : 338.366058, 187.098221, 44.482143	77 : 151.027176, 232.275818, 53.936142
78 : 303.288879, 161.644440, 45.888889	78 : 281.062500, 160.687500, 52.062500
79 : 346.331024, 204.186203, 45.710346	79 : -nan, -nan, -nan
80 : 317.111115, 171.148148, 46.111111	80 : 336.387085, 303.032257, 52.935482
81 : -nan, -nan, -nan	81 : 195.362503, 168.817505, 55.240002
82 : 308.321838, 338.264374, 47.915710	82 : 241.363632, 160.863632, 54.363636
83 : 255.598358, 161.733612, 50.758198	83 : -nan, -nan, -nan
84 : 151.045349, 233.630386, 54.455784	84 : -nan, -nan, -nan
85 : -nan, -nan, -nan	85 : -nan, -nan, -nan
86 : 352.497040, 226.721893, 54.272190	86 : -nan, -nan, -nan
87 : 336.462677, 302.149261, 52.522388	87 : -nan, -nan, -nan
88 : 195.096497, 169.182022, 54.896931	88 : -nan, -nan, -nan
89 : -nan, -nan, -nan	89 : 324.423065, 315.384613, 58.538460
90 : 281.500000, 345.000000, 53.000000	90 : -nan, -nan, -nan
91 : -nan, -nan, -nan	91 : -nan, -nan, -nan
92 : 242.000000, 161.214279, 54.285713	92 : 250.000000, 195.000000, 64.000000
93 : 181.580002, 179.643997, 55.939999	93 : -nan, -nan, -nan
94 : -nan, -nan, -nan	94 : -nan, -nan, -nan
95 : 184.603180, 331.825409, 58.158730	95 : -nan, -nan, -nan
96 : -nan, -nan, -nan	96 : -nan, -nan, -nan
97 : 345.593750, 263.031250, 59.656250	97 : -nan, -nan, -nan
98 : -nan, -nan, -nan	98 : -nan, -nan, -nan
99 : -nan, -nan, -nan	99 : -nan, -nan, -nan
100 : -nan, -nan, -nan	100 : -nan, -nan, -nan
101 : -nan, -nan, -nan	101 : -nan, -nan, -nan
102 : -nan, -nan, -nan	102 : -nan, -nan, -nan
103 : -nan, -nan, -nan	103 : -nan, -nan, -nan
104 : -nan, -nan, -nan	104 : -nan, -nan, -nan
105 : -nan, -nan, -nan	105 : -nan, -nan, -nan
106 : -nan, -nan, -nan	106 : -nan, -nan, -nan
107 : -nan, -nan, -nan	107 : -nan, -nan, -nan
108 : -nan, -nan, -nan	
109 : -nan, -nan, -nan	
110 : -nan, -nan, -nan	
111 : -nan, -nan, -nan	