Compte-rendu TP4

Question 2

```
typedef struct coordinateSum
{
    int x = 0;
    int y = 0;
    int z = 0;
    int size = 0;
} CoordinateSum;


int computeCentroids(int index, char* baseName){
    stringstream filenameSS;
    filenameSS << "DATA/" << baseName << "-" << index << ".hdr";
    char* filename = strdup(filenameSS.str().c_str());
    cout << "Processing " << filename << "..." << endl;

    float voxelSize[3];
    CImg<> img;
    img.load_analyze(filename, voxelSize);

    // application du filtre médian
    img = img.get_blur_median(2);

    // application du seuil
    img.threshold(25);

    // elimination des pixels isolés
    img.erode(3, 3, 3);
    img.dilate(3, 3, 3);

    // identification des cellules par composantes connexes
    img = img.get_label();

    int numberOfLabels = 200;
    CoordinateSum coordinateSum[numberOfLabels];
    // calcul des sommes des coordonnées pour chaque cellule/label
    for(int x=0; x<img.width(); x++){
        for(int y=0; y<img.height(); y++){
            for(int z=0; z<img.depth(); z++){
                int label = (int) img(x, y, z);
                coordinateSum[label].x += x;
                coordinateSum[label].y += y;
                coordinateSum[label].z += z;
                coordinateSum[label].size += 1;
            }
        }
    }

    // création du fichier stockant les barycentres des cellules
    stringstream resFilenameSS;
    resFilenameSS << baseName << "-" << index << " centroids.dat";
    filename = strdup(resFilenameSS.str().c_str());
    FILE* file = fopen(filename, "w+");
    for(int i=0; i<numberOfLabels; i++){
        int size = coordinateSum[i].size;
        if(size > 0){
            // calcul du barycentre de la cellule actuelle
            fprintf(file, "%f %f %f\n", (float)coordinateSum[i].x/size,
                (float)coordinateSum[i].y/size, (float)coordinateSum[i].z/size);
        }
    }
    fclose(file);
}
```

J’ai appelé la fonction *computeCentroids* pour chaque image *stack*, 21 fois en tout.
Dans cette fonction je commence par ouvrir l’image, puis je lui applique les actions suivantes :

- application d’un filtre médian pour éliminer le bruit
- application d’un seuil pour ne garder que les cellules
- élimination des pixels isolés grâce à une érosion suivi d’une dilatation
- identification des cellules par composantes connexes

Une fois que toutes les cellules, identifiées par un label unique, sont à notre disposition on peut calculer le barycentre en effectuant la somme des trois coordonnées de chaque voxel de la cellule et en divisant le tout par le nombre de voxels qui composent la cellule.
Tout ça est alors stocker ligne par ligne, une ligne correspondant à une cellule, dans un fichier.

Question 3 & 4

```
float getDistance(Voxel voxel1, Voxel voxel2){
    return (float)sqrt(pow(voxel1.x-voxel2.x, 2) + pow(voxel1.y-voxel2.y, 2) + pow(voxel1.z-voxel2.z, 2));
}

int computeDistances(char* baseName, int n1, int n2){
    // Lecture des deux fichiers comprenant les barycentres des cellules
    stringstream firstFile, secondFile;
    firstFile << baseName << "-" << n1 << " centroids.dat";
    secondFile << baseName << "-" << n2 << " centroids.dat";
    char* filename1 = strdup(firstFile.str().c_str());
    char* filename2 = strdup(secondFile.str().c_str());

    stringstream resFile;
    resFile << "dist" << " " << n1 << " " << n2 << ".dat";
    char* filename = strdup(resFile.str().c_str());
    FILE* res = fopen(filename, "w+");

    // Dans mon cas, avec le seuil que j'ai appliqué,
    // la première image n'a que 104 cellules
    // on ne va donc suivre que ces 104 cellules
    int maxIndex = 104;

    ifstream file1(filename1);
    int counter1 = 0;
    string line1, line2;
    while(getline(file1, line1)){
        counter1++;
        // Pour ne pas prendre des cellules qui n'existent pas dans toutes les images
        if(counter1 <= maxIndex){
            istringstream iss1(line1);
            Voxel voxel1;
            // error
            if(!(iss1 >> voxel1.x >> voxel1.y >> voxel1.z))
                break;

            // On cherche la distance minimale
            Voxel voxel;
            float minDistance = 10000000;
            ifstream file2(filename2);
            int counter2 = 0;
            int matchIndex = 0;
            while(getline(file2, line2)){
                counter2++;
                // Pour ne pas prendre des cellules qui n'existent pas dans toutes les images
                if(counter2 <= maxIndex){
                    istringstream iss2(line2);
                    Voxel voxel2;
                    // error
                    if(!(iss2 >> voxel2.x >> voxel2.y >> voxel2.z))
                        break;

                    float distance = getDistance(voxel1, voxel2);
                    if(distance < minDistance){
                        minDistance = distance;
                        voxel = voxel2;
                        matchIndex = counter2;
                    }
                }
            }
            // On écrit l'index correspondant à la cellule courante dans l'image suivante
            fprintf(res, "%d\n", matchIndex);
        }
    }
    fclose(res);
}
```

```
int main(int argc, char** argv){

    if(argc != 2){
        printf("Usage: baseFilePath \n");
        exit(1);
    }

    char* baseFilename = argv[1];


    for(int i=0; i<=20; i++){
        computeCentroids(i, baseFilename);
    }

    for(int i=0; i<20; i++){
        computeDistances(baseFilename, i, i+1);
    }

    getPath(baseFilename);
    return 0;
}
```

On commence par créer les fichiers contenant les indexes des cellules entre deux images. On va donc comparer les fichiers des barycentres deux à deux. Cela va résulter en d'autres fichiers dans lesquels le numéro de la ligne et la valeur de la ligne indique le chemin que prend la cellule dans le fichier suivant.
Pour effectuer cela j'ai comparé un premier fichier avec le suivant dans la liste des images 3D. Le premier fichier regarde donc, pour chacun de ses barycentres, le barycentre le plus proche dans le second fichier et stocke l'indice, qui correspond au numéro de la ligne dans le second fichier, dans un fichier.

```
// Construit le chemin contenant les indexes
int getPath(char* baseName){
    int cellPaths[104][20];
    for(int i=0; i<20; i++){
        stringstream firstFile;
        firstFile << "dist " << i << " " << (i+1) << ".dat";
        char* filename = strdup(firstFile.str().c_str());
        ifstream file(filename);
        string line;
        int counter = 0;
        int tmp[104];
        while(getline(file, line)){
            istringstream iss(line);
            int index;
            // error
            if(!(iss >> index))
                break;

            // La première fois, vu qu'il n'y a rien à comparer,
            // on ajoute juste la valeur dans le tableau
            if(i == 0){
                cellPaths[counter][i] = index;
            }
            // Sinon on le stocke dans un tableau temporaire pour plus tard
            else{
                tmp[counter] = index;
            }
            counter++;
        }
        // Si on ne se trouve pas dans la première image
        if(i != 0){
            for(int j=0; j<counter; j++){
                // On ajoute le bon index en regardant à l'index du tableau temporaire
                // qui correspond à l'index précédent du chemin
                cellPaths[j][i] = tmp[cellPaths[j][i-1]-1];
            }
        }
    }

    for(int i=0; i<104; i++){
        int path[21];
        // On n'oublie pas de stocker le première indice du chemin de la cellule
        // qui n'est au final que le numéro de la ligne
        path[0] = (i+1);
        for(int j=1; j<21; j++){
            path[j] = cellPaths[i][j-1];
        }
        // On crée un fichier .obj pour chaque cellule
        getPathFile(baseName, (i+1), path);
    }
}
```

Ensuite il s’agit de retrouver le chemin d’indices qu’a chaque cellule.
Cette fonction stocke les 21 indices, correspondants aux 21 images, de chaque cellule dans un tableau pour le passer à la fonction *getPathFile* dont je parlerai plus tard.
Pour retrouver le chemin d’indices de chaque cellule je parcours les fichiers des indices et forme le chemin en regardant à chaque fois la valeur à la ligne avec le numéro qui correspond à l’indice précédent du chemin.
Je pense que c’est plus simple d’expliquer cela par un exemple.

Si la cellule est la numéro 2, 2 sera alors le première élément dans le tableau d'indices de cette cellule.
Ensuite, je regarde dans le fichier des indices entre l'image 0 et 1 à la ligne 2.
Cette valeur, disons 3, je la stocke dans le tableau d'indices à la position suivante, donc 1, elle sera donc le deuxième élément de mon tableau d'indices.
Puis, je regarde dans le fichier des indices entre l'image 1 et 2 à la ligne correspondant à mon dernier élément du tableau des indices, 3.
Je stocke donc la valeur à la ligne 3 de ce fichier dans la tableau d'indices en tant que troisième élément.
Etc…

Au final cela me donne un tableau contenant les 21 indices de chaque cellule.
Ce tableau je le passe à la fonction *getPathFile*.

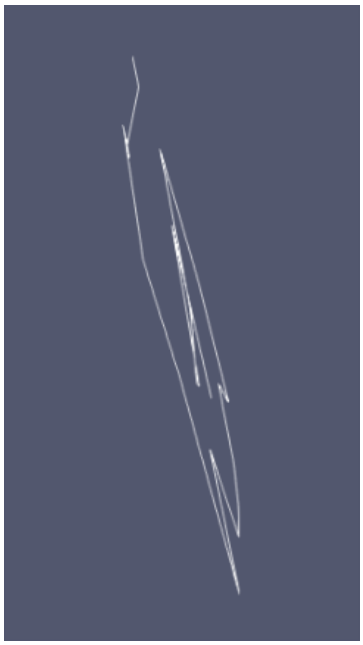
```
// Construit le fichier contenant les voxels de la trajectoire de la cellule
int getPathFile(char* baseName, int cellIndex, int path[]){
    stringstream resFile;
    resFile << "Path " << cellIndex << ".obj";
    char* filename = strdup(resFile.str().c_str());
    FILE* res = fopen(filename, "w+");

    // On parcourt les fichier contenants les barycentres
    for(int i=0; i<=21; i++){
        stringstream fileStream;
        fileStream << baseName << "-" << i << " centroids.dat";
        filename = strdup(fileStream.str().c_str());
        ifstream file(filename);
        string line;
        int counter = 0;
        while(getline(file, line)){
            counter++;
            // Si le numéro de la ligne correspond au chemin on ajoute la coordonnée au fichier
            if(counter == path[i]){
                istringstream iss(line);
                Voxel voxel;
                // error
                if(!(iss >> voxel.x >> voxel.y >> voxel.z))
                    break;
                fprintf(res, "v %f %f %f\n", voxel.x, voxel.y, voxel.z);
            }
        }
    }

    // On ajoute la dernière ligne dans le fichier pour que ParaView puisse le lire
    fprintf(res, "l ");
    for(int i=1; i<=21; i++){
        fprintf(res, "%d ", i);
    }
    fprintf(res, "\n");
    fclose(res);
}
```

Cette fonction va construire le fichier *.obj* qui permettra de lire le chemin de chaque cellule.
On fait cela en regardant chaque indice dans le tableau d'indices et en écrivant le barycentre, de l'image d'indice actuel, correspond à l'indice dans le tableau.

On a donc au final un fichier par cellule dans lequel se trouve le chemin qu'a effectué cette cellule au cours de l'acquisition des 21 images.
Voici, par exemple, le contenu du fichier pour la première cellule :

```
v 47.548912 47.471184 39.022491
v 47.497318 47.414978 39.017601
v 47.414616 47.276424 39.012417
v 47.397236 47.258205 39.009556
v 47.461285 47.330688 39.010609
v 47.154350 46.970562 39.011185
v 46.771408 46.574009 39.001732
v 45.611515 45.344463 38.973526
v 46.460072 46.274017 38.990593
v 45.979656 45.786541 38.974892
v 46.116348 45.950970 38.977905
v 46.351402 46.217804 38.988777
v 46.683704 46.576904 38.998352
v 46.579166 46.481819 39.004230
v 47.001495 46.919910 39.013950
v 47.362080 47.284332 39.015980
v 46.935402 46.815346 39.018166
v 46.691639 46.527283 39.014812
v 46.678818 46.518345 39.014038
v 47.175938 47.073883 39.021366
v 46.596668 46.455257 39.019466
1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
```

Ce qui donne la trajectoire suivante dans ParaView :



Je n'ai pas vraiment compris comment manipuler cet outil…
En tout cas, j'espère avoir été assez clair dans mes explications.