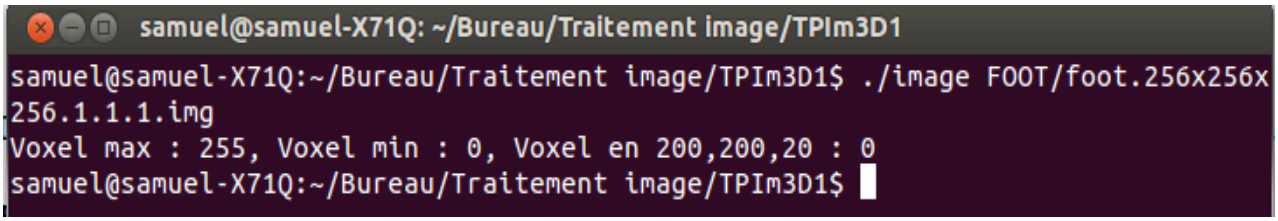
Compte rendu TP imagerie 3D n°1

Samuel Bricas

La valeur des Voxels

Une image 3D est composé de Voxels ayant chacun une valeur (= une intensité). Les fichiers contenant les images à traité sont codé sous la forme suivante : pour chaque image une matrice de n lignes et m colonnes. Pour pouvoir lire la valeur d'un Voxel à partir d'un fichier « .img », il faut parcourir le fichier en lisant deux octets par deux octets (car la valeur d'un Voxel est codé sur deux octets). Mais les deux octets ne sont pas lus dans le bonne ordre, il faut donc les inverser (voir code). Les résultats lus peuvent être enregistrer dans un tableau à trois dimension (une dimension pour les lignes, une dimension pour les colonnes et une dimension pour chaque matrice) pour facilité la lecture (voir code : to3DTab()).

Données récupérées avec le fichier "Brainix"



Données récupérées avec le fichier "foot"

Ces informations sont facilement récupérable grâce à la fonction getValue(x,y,z) qui donne la valeur du Voxel à la ligne x, la colonne y et la matrice z (la Zième image).

Volume Rendering

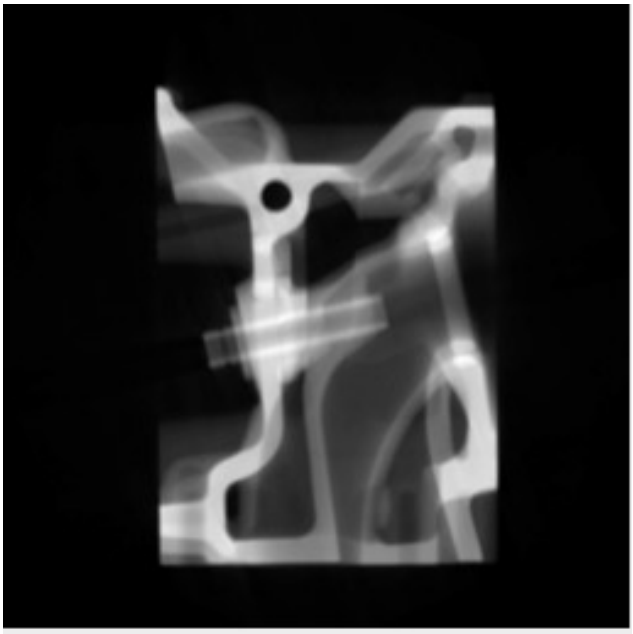
Une technique de volume rendering consiste à calculer la moyenne des intensité d'un Voxel sur l'axe des Z. C'est à dire que l'on calcule la moyenne des valeurs d'un Voxel pour chaque image. L'algorithme ici utilisé est le suivant : Pour chaque image qui compose l'image 3D (pour chaque Z) on récupère l'intensité de chaque Voxel que l'on additionne aux Voxels ayant la même position (même ligne, même colonne) dans les autres images. La somme des valeurs de ces

Voxels est ensuite diviser par le nombre d'image (= moyenne).

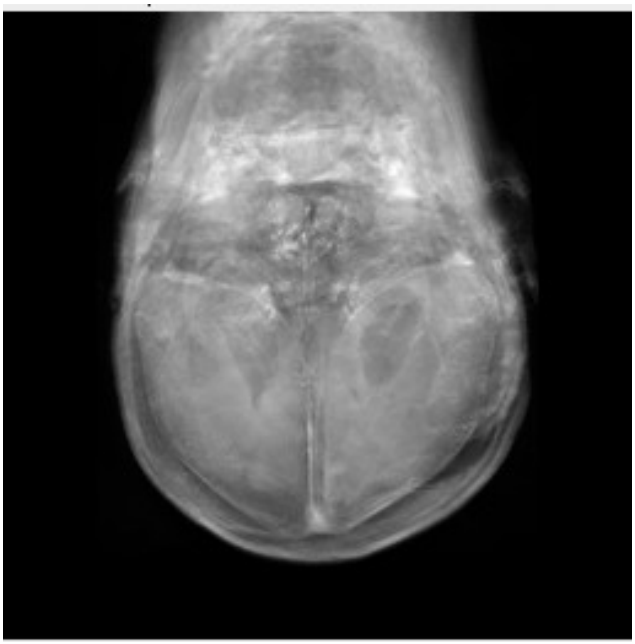
On peut également réaliser un volume rendering en calculant la valeur minimum d'un Voxel sur chaque image, ou en calculant la valeur maximum.

Voici des résultats obtenu pour quelques images données pour le TP (les images sont lus avec le logiciel Fiji) :

Volume Rendering sur "foot" (AIP)



Volume Rendering sur "engine" (AIP)



Volume Rendering sur "brainix" (AIP)

Volume Rendering sur "beaufix" (AIP)

WHAT IS IT ?

Le Volume Rendering appliquer à « whatisit » :

Volume Rendering sur "whatisit" (AIP)

Le Volume Rendering nous permet de répondre à la question : what is it ? C'est une écrevisse !

```
#include <iostream>
#include <cstdlib>
#include <math.h>
#include <stdio.h>


using namespace std;
int X = 256; // ligne
int Y = 256; // colonne
int Z = 100; // image

int getValue(int ***data,int x, int y, int z){
  int res = data[z][x][y];
  return res;
}

int*** to3DTab(int *data, size_t taille){
      int ***data3D = new int**[Z];
  int cpt =0;
      for(int i = 0;i<Z;i++){// pour chaque matrice
    data3D[i] = new int*[X];
           for(int j =0;j<X;j++){// pour chaque lignes
      data3D[i][j] =  new int[Y];
                  for(int k=0;k<Y;k++){// pour chaque colonnes
                        data3D[i][j][k]= data[cpt];
        cpt++;
                  }
             }
      }
  return data3D;
}


/*
 * Creer une image de volume rendering AIP
 */
void volumeRendering(int ***data){
  int intens;
  unsigned short *buff[1];
  FILE* out = fopen("out.0.raw","wb");
  for(int i =0;i<X;i++){
    for(int j = 0; j<Y;j++){
      intens =0;
      for(int k = 0;k<Z;k++){
        intens += getValue(data,i,j,k);
      }
      intens = intens/Z;
      int oct1 = intens/256;
      int oct2 = intens - oct1*256;
      buff[0] = (unsigned short *)(oct2*256 + oct1);
      fwrite(buff,sizeof(unsigned short),1,out);
    }
  }
  fclose(out);
}

/*
 * Creer un deux fichier : image de volume rendering MIP
 * Image de volume rendering MinIP
 */
void volumeRenderingMIPMinIP(int ***data){
  int max;
  int min;
  int res;
```

```
  unsigned short *buff[1];
  FILE* out = fopen("outMIP.0.raw","wb");
  FILE* out2 = fopen("outMinIP.0.raw","wb");
  min = getValue(data, 0,0,0);
  max = getValue(data,0,0,0);
  for(int i =0;i<X;i++){
    for(int j = 0; j<Y;j++){
      for(int k = 0;k<Z;k++){
        res = getValue(data,i,j,k);
        if(res > max){
          max = res;
        }
        if(res <min){
          min = res;
        }
      }
      int oct1 = max/256;
      int oct2 = max - oct1*256;
      buff[0] = (unsigned short *)(oct2*256 + oct1);
      fwrite(buff,sizeof(unsigned short),1,out);
      oct1 = min/256;
      oct2 = min - oct1*256;
      buff[0] = (unsigned short * )(oct2*256 + oct1);
      fwrite(buff,sizeof(unsigned short),1,out2);
    }
  }
  fclose(out);
  fclose(out2);

}



int main(int argc, char* argv[]){

      char cNomImgLue[250];
      if(argc != 2){
            cout<<"Usage : Image"<<endl;
            exit (1);
      }

      sscanf (argv[1],"%s",cNomImgLue);
      FILE*  f = fopen(cNomImgLue,"rb");

      long size;
      size_t nbLue;

      fseek (f , 0 , SEEK_END);
      size = ftell (f);
      rewind (f);

      unsigned short *buff = new unsigned short[size];

      nbLue = fread((char *)buff, 2,size,f);
      short oct1 = buff[0]%256;
      short oct2 = buff[0]/256;
      int val = oct1*256+oct2;
      int max = val;
    int min = val;
      int *data = new int[nbLue];
      data[0]= val;

      for(int i =1;i<nbLue;i++){
            oct1 = buff[i]%256;
```

```
            oct2 = buff[i]/256;
            val = oct1*256+oct2;
            if(val>max){
                  max =val;
            }
      if(val < min){
        min = val;
      }
            data[i]=val;
      }
    int*** dt = to3DTab(data,nbLue);
    int p = getValue(dt,200,200,20);
      cout<<"Voxel max : "<<max<<", Voxel min : "<<min<<", Voxel en 200,200,20 :
"<<p<<endl;
    volumeRendering(dt); // Création AIP
    volumeRenderingMIPMinIP(dt); // Création MIP et MinIP



}
```