HAI804I – Analyse et Traitement d'Images

Fabien Caballero

April 18, 2023

Contents

1 Seuillage

Pour réaliser cela, je parcours tout mes voxels un par un, pour chacun d'entre eux je regarde si sa valeur est supérieure au seuil si ce n'est pas le cas on passe au suivant. Si c'est le cas on récupère tout les voxels voisins (les 6) et pour chacun des voisins on regarde si leur valeur est inférieure au seuil si c'est le cas on ajoute la face qu'ils ont en commun dans le fichier stl. Puis on écrit le fichier stl et on l'ouvre avec meshlab par exemple, pour le visionner.

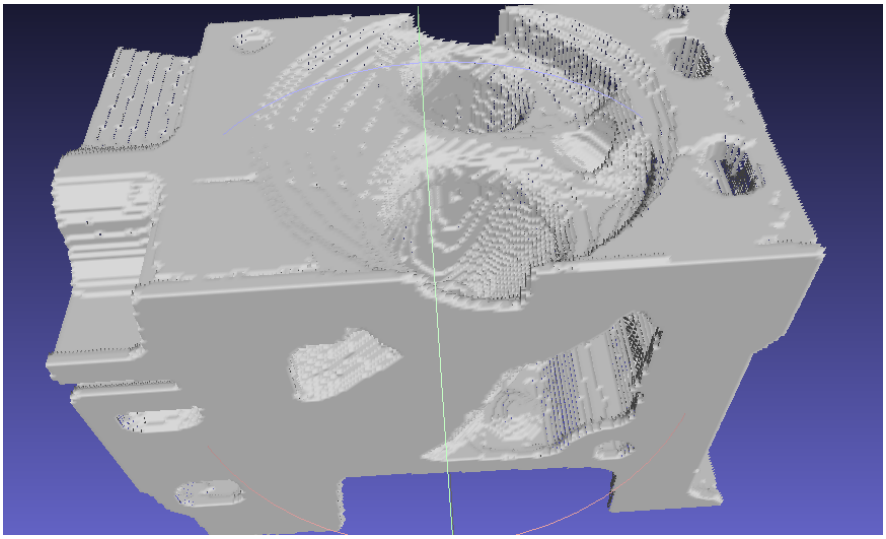

Figure 1: Engine seuil 100

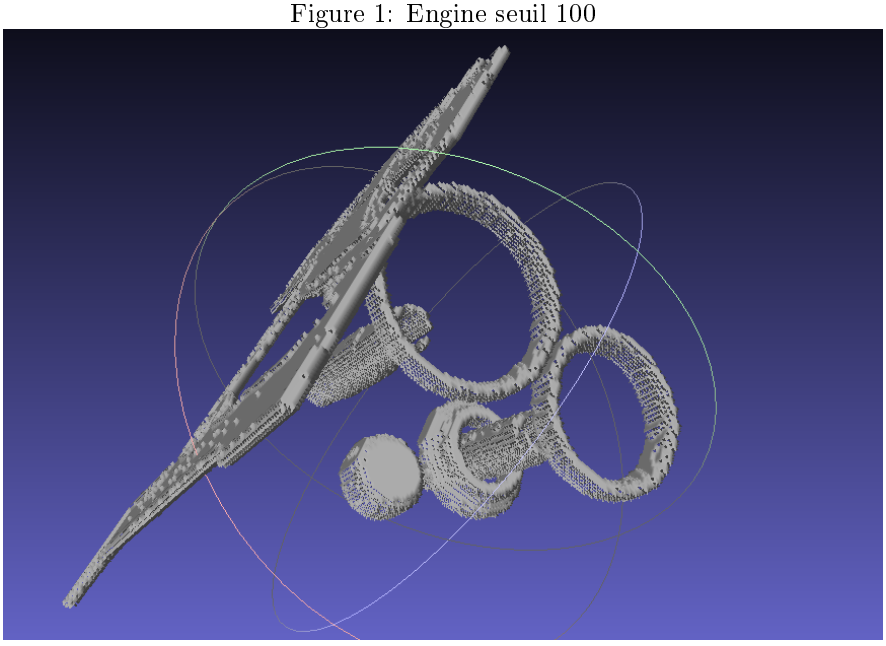

Figure 2: Engine seuil 200

2 Ce qu'il me reste à faire

Je n'ai pas eu le temps de faire pour les autres fichiers, je ne peux donc pas vérifier si c'est bon lorsque les voxels ne sont pas cubiques. Cependant je l'ai normalement géré en multipliant par la taille de mes voxels lors de la créations des sommets de mes voxels. Les tailles des voxels sont définies dans le code avec des variables je n'ai pas eu le temps de les mettres en paramètre de la ligne de commande.

3 Annexe (code)

```
    #include <stdio.h>
#include <iostream>
#include "image_ppm.h"
#include <vector>
#include <fstream>
using namespace std;

unsigned short getValue(unsigned short *img, int x, int y, int z, int dimX, int dimY, int
    dimZ)
{
    return (int)img[z * dimY * dimX + y * dimX + x];
}

unsigned short minElmt(unsigned short *img, size_t taille)
{
    unsigned short min = 0;
    for (size_t i = 0; i < taille; i++)
    {
        if (img[i] < min)
            min = img[i];
    }
    return min;
}

unsigned short maxElmt(unsigned short *img, size_t taille)
{
    unsigned short max = 0;
    for (size_t i = 0; i < taille; i++)
    {
        if (img[i] > max)
            max = img[i];
    }
    return max;
}

unsigned short inverse(unsigned short val)
{
    float o1 = floor(((double)val / 256.0));
    float o2 = val - o1 * 256;

    return o2 * 256 + o1;
}

vector<vector<int>> voisinage6(int i, int j, int k)
{
    vector<vector<int>> voisinage = {{i - 1, j, k}, {i + 1, j, k}, {i, j - 1, k}, {i, j + 1,
     k}, {i, j, k - 1}, {i, j, k + 1}};

    return voisinage;
}

int main(int argc, char const *argv[])
{
    FILE *f = fopen((char *)argv[1], "rb");
    unsigned short *contenu;

    int dimX = atoi(argv[2]);
    int dimY = atoi(argv[3]);
    int dimZ = atoi(argv[4]);

    int seuil = atoi(argv[5]);
```

```
60      allocation_tableau(contenu, unsigned short, dimX *dimY *dimZ);
61
62      size_t taille = fread(contenu, sizeof(unsigned short), dimX * dimY * dimZ, f);
63      for (size_t i = 0; i < taille; i++)
64      {
65          contenu[i] = inverse(contenu[i]);
66      }
67
68      int voxelSizeX = 1;
69      int voxelSizeY = 1;
70      int voxelSizeZ = 1;
71
72
73      ofstream file;
74      file.open("modelbis.stl");
75      file << "solid modelbis.stl" << endl;
76
77      for (size_t i = 1; i < dimX - 1; i++)
78      {
79          for (size_t j = 1; j < dimY - 1; j++)
80          {
81              for (size_t k = 1; k < dimZ - 1; k++)
82              {
83                  if (getValue(contenu, i, j, k, dimX, dimY, dimZ) < seuil)
84                  {
85                      continue;
86                  }
87                  vector<vector<int>> vect = voisinage6(i, j, k);
88                  size_t size = vect.size();
89                  for (size_t v = 0; v < 6; v++)
90                  {
91                      if (getValue(contenu, vect[v][0], vect[v][1], vect[v][2], dimX, dimY,
    dimZ) < seuil)
92                      {
93                          vector<float> center = {(float)vect[v][0], (float)vect[v][1], (float
    )vect[v][2]};
94                          vector<float> zero = {(center[0] - 0.5f) * voxelSizeX, (center[1] -
    0.5f) * voxelSizeY, (center[2] + 0.5f) * voxelSizeZ};
95                          vector<float> un = {(center[0] + 0.5f) * voxelSizeX, (center[1] -
    0.5f) * voxelSizeY, (center[2] - 0.5f) * voxelSizeZ};
96                          vector<float> deux = {(center[0] + 0.5f) * voxelSizeX, (center[1] -
    0.5f) * voxelSizeY, (center[2] - 0.5f) * voxelSizeZ};
97                          vector<float> trois = {(center[0] - 0.5f) * voxelSizeX, (center[1] -
     0.5f) * voxelSizeY, (center[2] - 0.5f) * voxelSizeZ};
98                          vector<float> quatre = {(center[0] - 0.5f) * voxelSizeX, (center[1]
    + 0.5f) * voxelSizeY, (center[2] + 0.5f) * voxelSizeZ};
99                          vector<float> cinq = {(center[0] + 0.5f) * voxelSizeX, (center[1] +
    0.5f) * voxelSizeY, (center[2] + 0.5f) * voxelSizeZ};
100                         vector<float> six = {(center[0] + 0.5f) * voxelSizeX, (center[1] +
    0.5f) * voxelSizeY, (center[2] - 0.5f) * voxelSizeZ};
101                         vector<float> sept = {(center[0] - 0.5f) * voxelSizeX, (center[1] +
    0.5f) * voxelSizeY, (center[2] - 0.5f) * voxelSizeZ};
102                         switch (v)
103                         {
104                         case 1:
105
106                             file << "facet normal 0 0 0" << endl;
107                             file << "\touter loop" << endl;
108                             file << "\t\tvertex " << zero[0] << " " << zero[1] << " " <<
    zero[2] << endl;
109                             file << "\t\tvertex " << trois[0] << " " << trois[1] << " " <<
    trois[2] << endl;
110                             file << "\t\tvertex " << quatre[0] << " " << quatre[1] << " " <<
     quatre[2] << endl;
111
```

```
112                             file << "\tendloop" << endl;
113                             file << "endfacet" << endl;
114 
115                             file << "facet normal 0 0 0" << endl;
116                             file << "\touter loop" << endl;
117                             file << "\t\tvertex " << trois[0] << " " << trois[1] << " " <<
        trois[2] << endl;
118                             file << "\t\tvertex " << sept[0] << " " << sept[1] << " " <<
        sept[2] << endl;
119                             file << "\t\tvertex " << quatre[0] << " " << quatre[1] << " " <<
         quatre[2] << endl;
120 
121                             file << "\tendloop" << endl;
122                             file << "endfacet" << endl;
123                             break;
124                         case 2:
125 
126                             file << "facet normal 0 0 0" << endl;
127                             file << "\touter loop" << endl;
128                             file << "\t\tvertex " << un[0] << " " << un[1] << " " << un[2]
       << endl;
129                             file << "\t\tvertex " << deux[0] << " " << deux[1] << " " <<
        deux[2] << endl;
130                             file << "\t\tvertex " << cinq[0] << " " << cinq[1] << " " <<
        cinq[2] << endl;
131 
132                             file << "\tendloop" << endl;
133                             file << "endfacet" << endl;
134 
135                             file << "facet normal 0 0 0" << endl;
136                             file << "\touter loop" << endl;
137                             file << "\t\tvertex " << deux[0] << " " << deux[1] << " " <<
        deux[2] << endl;
138                             file << "\t\tvertex " << six[0] << " " << six[1] << " " << six
        [2] << endl;
139                             file << "\t\tvertex " << cinq[0] << " " << cinq[1] << " " <<
        cinq[2] << endl;
140 
141                             file << "\tendloop" << endl;
142                             file << "endfacet" << endl;
143                             break;
144                         case 3:
145 
146                             file << "facet normal 0 0 0" << endl;
147                             file << "\touter loop" << endl;
148                             file << "\t\tvertex " << zero[0] << " " << zero[1] << " " <<
        zero[2] << endl;
149                             file << "\t\tvertex " << un[0] << " " << un[1] << " " << un[2]
       << endl;
150                             file << "\t\tvertex " << trois[0] << " " << trois[1] << " " <<
        trois[2] << endl;
151 
152                             file << "\tendloop" << endl;
153                             file << "endfacet" << endl;
154 
155                             file << "facet normal 0 0 0" << endl;
156                             file << "\touter loop" << endl;
157                             file << "\t\tvertex " << un[0] << " " << un[1] << " " << un[2]
       << endl;
158                             file << "\t\tvertex " << deux[0] << " " << deux[1] << " " <<
        deux[2] << endl;
159                             file << "\t\tvertex " << trois[0] << " " << trois[1] << " " <<
        trois[2] << endl;
160 
161                             file << "\tendloop" << endl;
```

```
162                             file << "endfacet" << endl;
163                             break;
164                         case 4:
165 
166                             file << "facet normal 0 0 0" << endl;
167                             file << "\touter loop" << endl;
168                             file << "\t\tvertex " << quatre[0] << " " << quatre[1] << " " <<
        quatre[2] << endl;
169                             file << "\t\tvertex " << cinq[0] << " " << cinq[1] << " " <<
       cinq[2] << endl;
170                             file << "\t\tvertex " << sept[0] << " " << sept[1] << " " <<
       sept[2] << endl;
171 
172                             file << "\tendloop" << endl;
173                             file << "endfacet" << endl;
174 
175                             file << "facet normal 0 0 0" << endl;
176                             file << "\touter loop" << endl;
177                             file << "\t\tvertex " << cinq[0] << " " << cinq[1] << " " <<
       cinq[2] << endl;
178                             file << "\t\tvertex " << six[0] << " " << six[1] << " " << six
       [2] << endl;
179                             file << "\t\tvertex " << sept[0] << " " << sept[1] << " " <<
       sept[2] << endl;
180 
181                             file << "\tendloop" << endl;
182                             file << "endfacet" << endl;
183                             break;
184                         case 5:
185                             file << "facet normal 0 0 0" << endl;
186                             file << "\touter loop" << endl;
187                             file << "\t\tvertex " << trois[0] << " " << trois[1] << " " <<
       trois[2] << endl;
188                             file << "\t\tvertex " << deux[0] << " " << deux[1] << " " <<
       deux[2] << endl;
189                             file << "\t\tvertex " << sept[0] << " " << sept[1] << " " <<
       sept[2] << endl;
190 
191                             file << "\tendloop" << endl;
192                             file << "endfacet" << endl;
193 
194                             file << "facet normal 0 0 0" << endl;
195                             file << "\touter loop" << endl;
196                             file << "\t\tvertex " << deux[0] << " " << deux[1] << " " <<
       deux[2] << endl;
197                             file << "\t\tvertex " << six[0] << " " << six[1] << " " << six
       [2] << endl;
198                             file << "\t\tvertex " << sept[0] << " " << sept[1] << " " <<
       sept[2] << endl;
199 
200                             file << "\tendloop" << endl;
201                             file << "endfacet" << endl;
202                             break;
203                         case 6:
204                             file << "facet normal 0 0 0" << endl;
205                             file << "\touter loop" << endl;
206                             file << "\t\tvertex " << zero[0] << " " << zero[1] << " " <<
       zero[2] << endl;
207                             file << "\t\tvertex " << un[0] << " " << un[1] << " " << un[2]
      << endl;
208                             file << "\t\tvertex " << quatre[0] << " " << quatre[1] << " " <<
        quatre[2] << endl;
209 
210                             file << "\tendloop" << endl;
211                             file << "endfacet" << endl;
```

```
212
213                             file << "facet normal 0 0 0" << endl;
214                             file << "\touter loop" << endl;
215                             file << "\t\tvertex " << un[0] << " " << un[1] << " " << un[2]
       << endl;
216                             file << "\t\tvertex " << cinq[0] << " " << cinq[1] << " " <<
       cinq[2] << endl;
217                             file << "\t\tvertex " << quatre[0] << " " << quatre[1] << " " <<
        quatre[2] << endl;
218
219                             file << "\tendloop" << endl;
220                             file << "endfacet" << endl;
221                             break;
222
223                         default:
224                             break;
225                         }
226                     }
227                 }
228             }
229         }
230         cout << i << " / " << dimX << endl;
231         ;
232     }
233
234     file << "endsolid" << endl;
235
236     file.close();
237
238     return 0;
239 }
```