Threshold-Free Cluster Enhancement

Luigi Giugliano[1], Marco Mecchia[1]

[1]Università degli studi di Salerno

3 maggio 2016

Overview

Codice

Overview

Codice

Codice

Andremo ora a spiegare il codice prodotto per il plugin TFCE.

I file principali che compongono il plugin sono:

- Tfce.cpp
- Utilities.cpp

Tfce è il core del plugin, dove avviene il calcolo degli score.

Utilities invece contiene tutti le funzioni di supporto per l'esecuzione del plugin stesso.

L' unica funzione che viene esposta dal file **Tfce.h** è:

```
1 #ifndef TFCE_H
2       #define TFCE_H
3       #include <float.h>
4       float * tfce_score(float * map, int dim_x, int
            dim_y, int dim_z, float E, float H, float dh);
5 #endif //TFCE_H
```

Le funzioni che espone **Utilities.h** sono:

```
1  #ifndef UTILITIES_H
2      #define UTILITIES_H
3
4      #include <float.h>
5      #include <stdio.h>
6
7      void findMinMax(float *map, int n, float *min,
           float *max, float * range);
8
9      int confront(float a, float b, char operation);
10
11     int * getBinaryVector(float * map, int n, int
           (*confront)(float, float), float value, int *
               numOfElementsMatching);
```

```
1     float * fromBinaryToRealVector(float * map, int n,
          int * binaryVector);
2
3     float * fill0(int n);
4
5     void apply_function(float * vector, int n, float (*
          operation) (float a, float b), float argument);
6
7     int linearIndexFromCoordinate(int x, int y, int z,
          int max_x, int max_y);
8
9     void coordinatesFromLinearIndex(int index, int
          max_x, int max_y, int * x, int * y, int * z);
10
11    float * copyAndConvertIntVector(int * vector, int n);
12
13 #endif //UTILITIES_H
```

Tfce.cpp oltre al metodo visto precedentemente fornisce i seguenti metodi:

```
1 int * find_clusters_3D(int * binaryVector, int dim_x,
      int dim_y, int dim_z, int n, int * num_clusters)
```

questo metodo preso in input una **mappa binaria in 3D ma linearizzata**, le sue tre dimensioni, il numero totale voxel della mappa e il puntatore ad un intero che indica il numero attuale di cluster trovati.

Cercando i cluster all'interno della mappa 3D utilizzando la **26-connectivity**

Restituisce un'altra mappa in cui al posto degli uno viene sostituito l'identificativo del cluster.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 3 & 3 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 2 \\ 1 & 0 & 0 & 2 & 2 \end{bmatrix}$$

In questo piccolo esempio in 2D viene mostrato il funzionamento della nostra funzione.