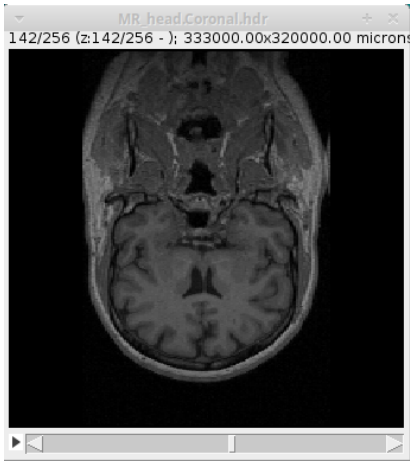
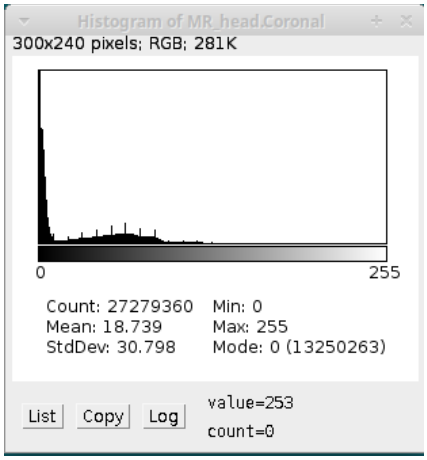
Compte-rendu TP1

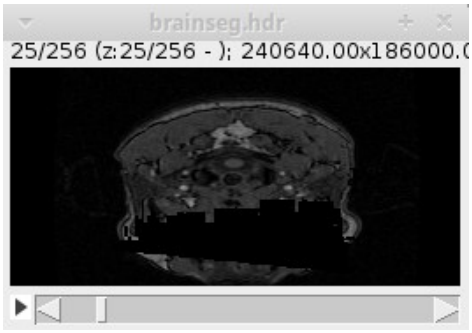
Question 2

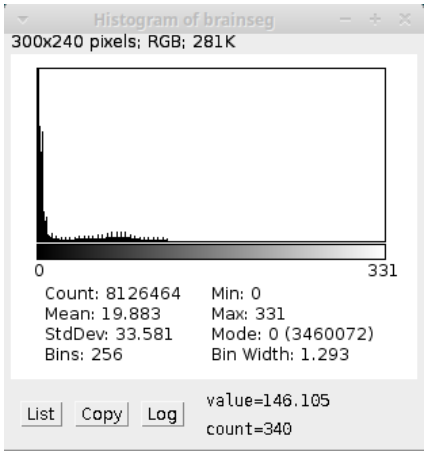
Head



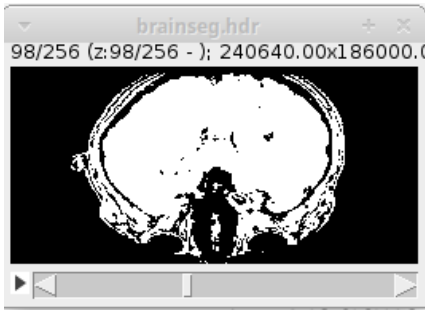


Brainseg

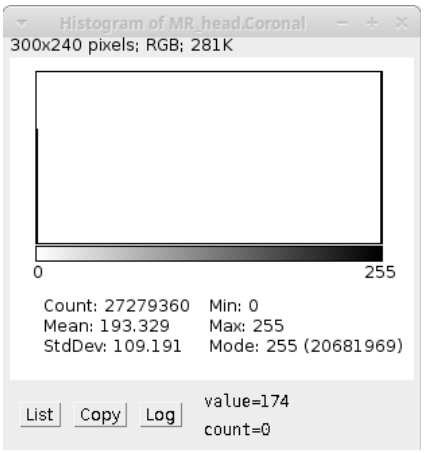


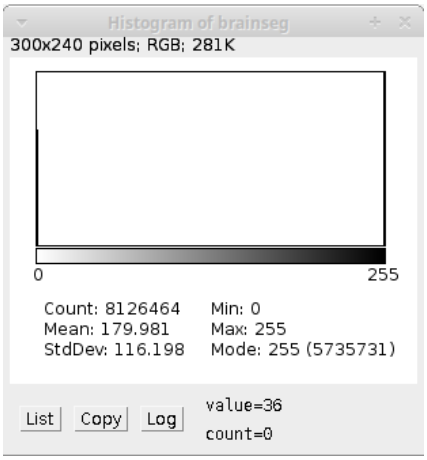


a) L'histogramme représente la répartition des niveaux de gris sur l'ensemble des 256 images de chaque image 3D. Vu que toutes les coupes des deux images 3D ont une forte dominance de la couleur noir, correspondant au fond de l'image, quand rien n'a été perçu par le scanner, cela paraît logique que les deux histogrammes soient fortement influencés par la valeur 0 et les valeurs se rapprochant de 0 car ces valeurs correspondent à la couleur noir.

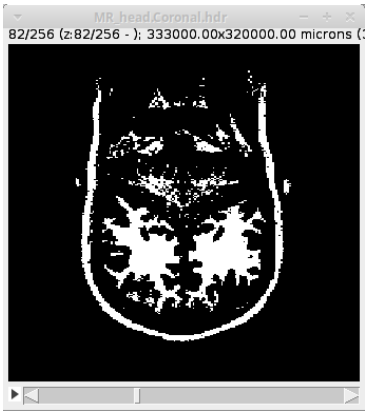


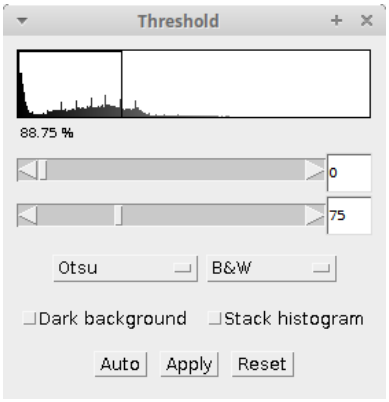
b) Les images qui résultent de l'application de la méthode Otsu sur les images de base ne sont répresentées qu'avec deux couleurs, le noir et le blanc. Le seuil correspond à la valeur qui va déterminer si un certain pixel sera de couleur noir ou blanche. Dans notre exemple, si la valeur initiale du pixel est au-dessus du seuil le pixel aura, après application de la méthode Otsu, 255 comme valeur, correspondant à la couleur blanche. A l'inverse si la valeur initiale du pixel est en-dessous du seuil on assignera 0 comme valeur à ce pixel, correspondant à la couleur noir. En soit, ce qui est utile est en blanc, ce qui ne l'est pas est en noir. Pour montrer qu'il n'y a bien que deux couleurs associées à chaque image, voici l'histogramme des deux images 3D après application de la méthode Otsu.





Question 3





J'ai choisi comme seuil la valeur 75 pour mettre en évidence la matière blanche sur l'image 3D de la tête. D'après moi il est assez difficile de dissocier les différentes parties de l'image avec un simple seuil.

Question 4

Paramètres :

- *p* : nombre d'érosion/dilatation à effectuer
- *seuil* : valeur clé séparant l'image en deux couleurs

Algorithme :

On applique un seuillage de valeur *seuil* pour créer un masque binaire M.
On applique *p* érosions sur ce masque, les voxels érodés sont marqués "épluché" (peeled).
Le paramètre *p* doit être suffisamment grand pour pouvoir faire disparaître complètement les "ponts" lors des érosions.
On identifie et trie par taille toutes les composantes connexes.
On affiche uniquement les voxels correspondant à la plus grande composante connexe.
On applique *p* dilatations pour "annuler" les effets des *p* érosions effectuées précédemment.

Résultat :

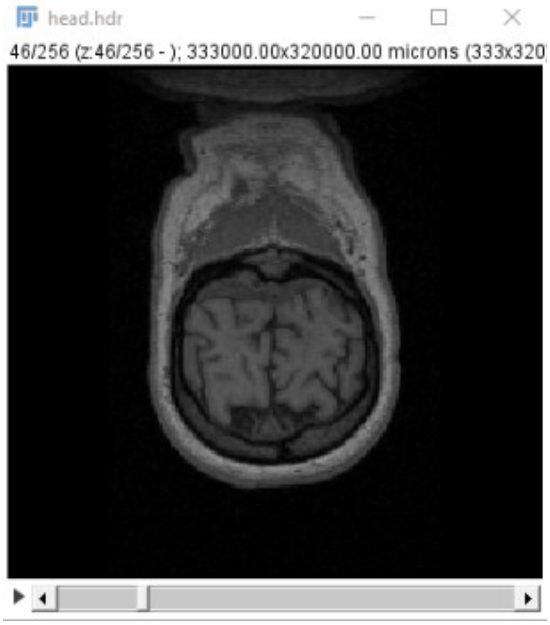
On voit uniquement le cerveau, la segmentation du cerveau est réussie.

Question 5

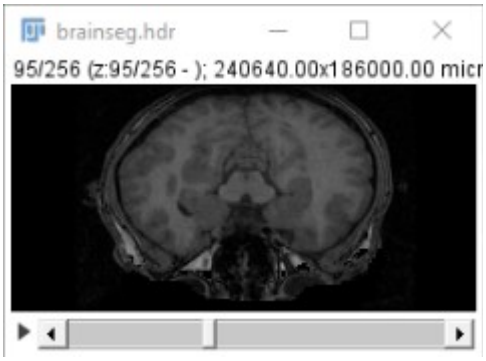
```cpp
#include "CImg.h"
#include <iostream>
using namespace cimg_library;
using namespace std;

// Effectue la segmentation de l'image en fonction des paramètres passés en arguments
CImg<float> segmentation(char* filename, int threshold, int p){
    if(p < 3)
        p = 3;

    CImg<float> img;
    float voxelSize[3];

    img.load_analyze(filename, voxelSize);

    // On applique le seuil
    img.threshold(threshold);

    // On applique les p erosions
    for(int i=0; i<p; i++){
        img.erode(2,2,2);
    }

    // On label les zones pour pouvoir recuperer celle du cerveau
    img.label();

    int labels[2500];
    for(int i=0; i<2500; i++){
        labels[i] = 0;
    }

    int maxLabel = -1;
    int maxOcc = -1;
    // On cherche le label le plus present
    for(int i=0; i<img.width()*img.height()*img.depth(); i++){
        if(img[i] != 0){
            labels[(int)img[i]]++;
            if(labels[(int)img[i]] > maxOcc){
                maxOcc = labels[(int)img[i]];
                maxLabel = img[i];
            }
        }
    }

    // On affiche uniquement les pixels correspondant au label le plus present
    for(int i=0; i<img.width()*img.height()*img.depth(); i++){
        if(img[i] == maxLabel)
            img[i] = 255;
        else
            img[i] = 0;
    }

    // On applique les p dilatations
    for(int i=0; i<p; i++){
        img.dilate(2,2,2);
    }

    // On sauvegarde l'image resultante
    img.save_analyze("Results/res.hdr", voxelSize);

    return img;
}
```

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

    if(argc < 4){
        printf("Usage: filename threshold p\n");
        exit(1);
    }

    char* filename = argv[1];
    int threshold = atoi(argv[2]);
    int p = atoi(argv[3]);

    CImg<float> img = segmentation(filename, threshold, p);

    return 0;
}
```

Question 6



Voici la segmentation du cerveau de l'image 3D *head* avec un seuil égal à 50 et p égal à 3.



Voici la segmentation du cerveau de l'image 3D *brain* avec un seuil égal à 47 et p égal à 3.

Vu que j'ai rendu mon programme interactif j'ai pu tester plusieurs combinaisons de paramètres pour pouvoir sélectionner celle qui donne le meilleur résultat. Effectuer un seuil aux alentours des 50, puis 3 érosions suivis de 3 dilatations me semble être la meilleure approche pour avoir une segmentation du cerveau la plus optimale possible, même si le résultat n'est pas parfait sur quelques coupes.

Question 7

```cpp
/*
    Les touches + et - du pavé numérique augmente et baisse la valeur du seuil
    Les flèches du haut et du bas augmente et baisse la valeur de p, le nombre de transformations
*/
int main(int argc, char** argv){

    if(argc < 4){
        printf("Usage: filename threshold p\n");
        exit(1);
    }

    char* filename = argv[1];
    int threshold = atoi(argv[2]);
    int p = atoi(argv[3]);

    CImg<float> img = segmentation(filename, threshold, p);

    CImgDisplay main_display(img, "Brain segmentation");
    int slice_index = img.depth()/2;
    CImg<> slice = img.get_slice(slice_index);
    main_display.display(slice);

    while (!main_display.is_closed()) {
        main_display.wait();
        if(main_display.wheel()){
            // On change la coupe en fonction de l'action de la molette
            int counter = main_display.wheel();
            slice_index += counter;
            if(slice_index < 0)
                slice_index = 0;
            if(slice_index >= img.depth())
                slice_index = img.depth()-1;

            // On affiche la nouvelle coupe
            slice = img.get_slice(slice_index);
            main_display.display(slice);

            // On remet la molette à 0 pour faciliter sa lecture
            main_display.set_wheel();
        }
        else if(main_display.is_keyPADADD()){
            threshold += 1;
        }
        else if(main_display.is_keyPADSUB()){
            threshold -= 1;
        }
        else if(main_display.is_keyARROWUP()){
            p += 1;
        }
        else if(main_display.is_keyARROWDOWN()){
            p -= 1;
        }

        // Dans ces cas là il faut mettre à jour le display
        if(main_display.is_keyPADSUB() or main_display.is_keyPADADD() or main_display.is_keyARROWUP() or main_display.is_keyARROWDOWN()){
            cout << "Calcul en cours..." << endl;
            img = segmentation(filename, threshold, p);
            cout << "Nouvelle segmentation avec un seuil de " << threshold << " et " << p << " transformations." << endl;
            // On affiche la nouvelle coupe
            slice = img.get_slice(slice_index);
            main_display.display(slice);
        }
    }

    return 0;
}
```

J'ai modifié le main de sorte à ce que, au lieu de simplement sauvegarder l'image dans un fichier, le programme ouvre directement l'image 3D résultante de la segmentation.
J'ai ensuite ajouté la possibilité de changer les valeurs du *seuil* et de *p* pour rendre plus facile la comparaison entre les différentes segmentations en fonction des paramètres utilisés.

Les touches + et - du pavé numérique augmente et baisse la valeur du seuil.
Les flèches du haut et du bas augmente et baisse la valeur de p, le nombre de transformations.