

Dans cette partie, nous allons présenter notre approche, qui consiste à réaliser un système de segmentation d'images IRM cérébrale en utilisant l'algorithme division/fusion (split & merge) de la méthode classique approche région en se basant sur l'utilisation d'un système multi-agents (ou SMA).

1. Outils de développement :

Afin de réaliser notre système, plusieurs outils sont nécessaires, desquels :

1.1. Java comme langage de programmation :

Java¹ est un langage de programmation expressément conçu pour être utilisé dans l'environnement distribué de l'Internet. Il a été conçu pour avoir le "look and feel" du langage C ++, mais plus simple à utiliser et appliquer le modèle de programmation orienté objet. Java peut être utilisé pour créer des applications complètes qui peuvent s'exécuter sur un ordinateur unique ou réparties entre serveurs et clients dans un réseau. Il peut également être utilisé pour construire l'applet pour une utilisation dans le cadre d'une page Web. En plus, il possède les deux principales caractéristiques :

- **la portabilité**: compile once, run everywhere
Java permet l'exécution du système à base d'agents sur différents systèmes d'exploitation, et sur différentes machine.
- **La fiabilité** : les applications Java peuvent accéder et charger des pages sur internet via des URL avec la même facilité qu'elles accèdent au fichier local sur le système. de plus, les fonctionnalités réseau sont fiables.

1.2. JADE comme plate-forme :

La figure 5.1 montre l'interface graphique de JADE lorsqu'on tape la commande suivante dans l'invite de commande : **Java jade.Boot -gui.**

¹<http://searchsoa.techtarget.com/definition/Java>

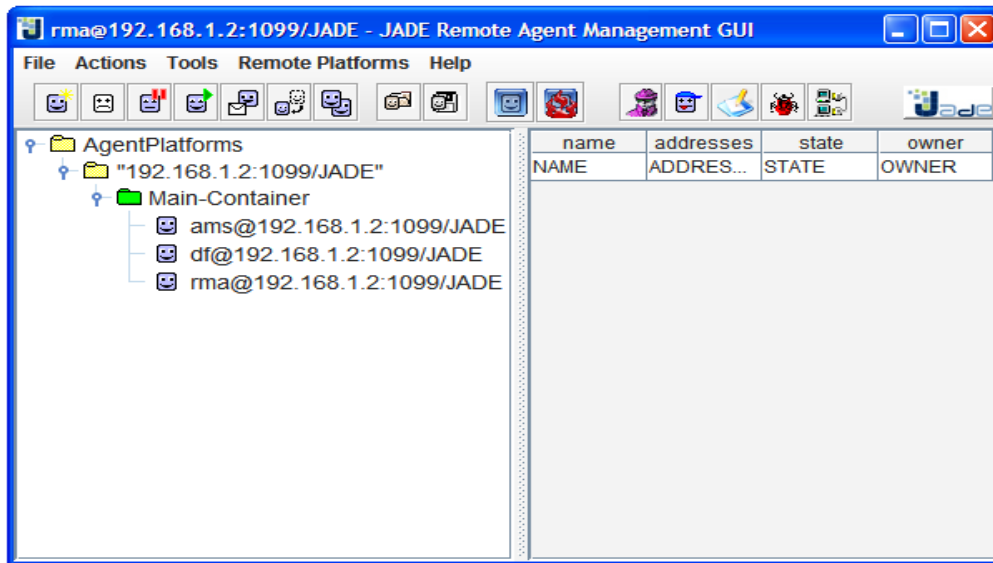


Fig. 5.1. Capture d'écran de JADE

1.2.1. Création d'un agent JADE :

Les Agents JADE sont définis comme des sous-classes de la classe prédéfinie Agent et leur code initial doit être placé dans une méthode appelée setup ().

```
import jade.core.Agent;

public class SimpleAgent extends Agent {
    protected void setup(){
        // Initialisation de l'agent
        System.out.println("SimpleAgent est prés...");
    }
    protected void takeDown(){
        // Traitement de fin
        System.out.println("SimpleAgent mort");
    }
}
```

1.2.2. Cycle de vie d'un agent JADE :

Durant l'exécution d'un agent, son état peut être modifié

- INITIATED : l'agent est lancé mais non enregistré auprès de l'AMS (*Agent Management System*, Service de Pages Blanches : référence automatiquement les agents suivant leur nom dès leur entrée dans le système.), aucun nom, aucune adresse.
- ACTIVE : l'agent est répertorié auprès de l'AMS et peut accéder aux services.
- SUSPENDED : tous les comportements de l'agent sont suspendus.
- TRANSIT : l'agent migre vers une autre plateforme.
- WAITING : tous les comportements de l'agent sont temporairement interrompus.
- DELETED : l'exécution de l'agent est terminée et n'est plus répertorié au sein de l'AMS. [OLI10]

1.2.3. Identification d'un agent :

L'agent est identifié auprès de l'AMS par un identifiant unique, appelé GUID (Global Unique Identifier).

- *Cet identifiant est un objet de type `jade.core.AID`*

```
AID(java.lang.String name, boolean isGUID)
AID id = new AID(localname, AID.ISLOCALNAME);
AID id = new AID(name, AID.ISGUID);
```

- *Les attributs de cet objet sont :*
 - name : <nickname>@<plateforme> [`getLocalName()`, `getName()`]
 - addresses : adresses de la plateforme [`getAllAddresses()`]
 - resolvers : services de page blanche [`getAllResolvers()`]

Ces informations sont accessibles par les méthodes de la classe `jade.core.Agent` :

`getAID()` : retourne l'identifiant unique de l'agent,

`getName()` : retourne le nom de l'agent, i.e. `<nickname>@<plateforme>`

`getLocalName()` : retourne uniquement le nickname (nom local),

`getAMS()` : retourne l'AID de l'AMS sur laquelle l'agent est situé.

1.2.4. Les communications des agents JADE :

Pour que plusieurs agents JADE arrivent à collaborer, ils doivent s'échanger² des messages. Chaque agent JADE possède une sorte de boîte aux lettres qui contient les messages qui lui sont envoyés par les autres agents. Ces boîtes aux lettres sont sous forme d'une liste qui contient les messages selon l'ordre chronologique de leur arrivée. Les agents JADE utilisent des messages conformes aux spécifications de la FIPA (FIPA-ACL). Les messages JADE sont des instances de la classe `ACLMessage` du package `jade.lang.acl`.

- **Création et envoi d'un message :**

Un message est créé comme instance de la classe `ACLMessage`.

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);  
  
msg.addReceiver(new AID("destinataire", AID.ISLOCALNAME));  
  
msg.setContent("le contenu du message");
```

- **L'attente et la réception d'un message :**

```
ACLMessage msg1 = myAgent.receive();  
while (msg1 == null){  
    if (i==0)  
        System.out.println(" message ");  
    i++;  
}
```

²<http://djug.developpez.com/>

```
if (msg1!= null){  
    // Traitement du message  
}
```

1.2.5. Les Behaviours dans JADE (comportement) :

Pour qu'un agent JADE exécute une tâche, nous avons tout d'abord besoin de définir ces tâches. Les tâches dans JADE (appelées des behaviours ou des comportements) sont des instances de la classe `jade.core.behaviours`³. Pour qu'un agent exécute une tâche, on doit la lui attribuer par la méthode `addBehaviour(Behaviour)` de la classe `jade.core.Agent`. [OLI10]

1.3. Environnement de programmation : Eclipse

L'IDE (Environnement de développement intégré) Eclipse, considéré par beaucoup comme le meilleur outil de développement Java disponibles, contient tout ce dont on a besoin pour construire des applications Java et Java Enterprise Edition (Java EE).

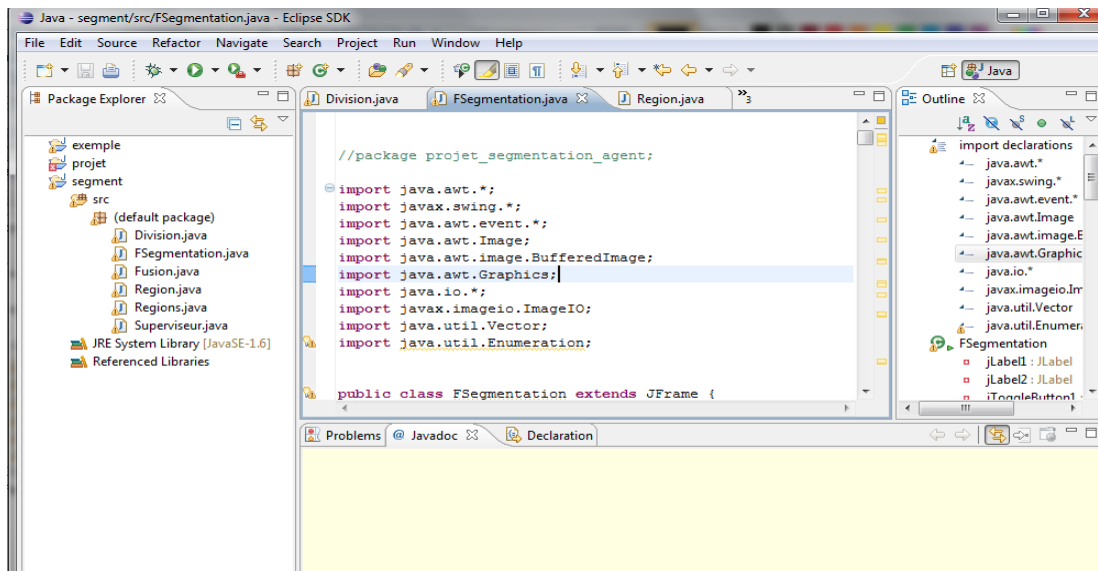


Fig.5.2.Capture d'écran du Java EE eclipse.

³<http://djug.developpez.com/>

2. Description du système :

2.1. Composants du système :

Pour réaliser l'objectif de ce mémoire, notre application s'articule autour des étapes suivantes :

- ✓ Dans la Phase d'Initialisation, l'agent Superviseur est lancé l'agent Division relatif au premier germe.
- ✓ L'agent Division est divisé l'image en régions en quatre parties de même taille (basée sur la notion de « quadtree »).
- ✓ Lorsque l'agent Superviseur reçoit un message par l'agent Division « région inexistante » donc il envoie un message à l'agent Fusion « Start fusion ».

Dans ce qui suit, nous présentons les différents agents utilisés dans le développement de notre système :

❖ Agent Superviseur :

L'agent superviseur est appartient à la classe Superviseur. Cet agent a pour but de contrôler et d'organiser le travail des autres agents. L'agent superviseur est à tout moment à l'état d'écoute des messages des autres agents.

❖ Agent Division :

L'agent Division est appartient à la classe Division. Cet agent est chargé pour diviser les images IRM à des régions (phase de split de l'algorithme Split and merge) et ensuite renvoyer ces régions à l'agent Superviseur.

❖ Agent Fusion :

L'agent Fusion est appartient à la classe Fusion. Cet agent est permet de fusionner les régions adjacentes de chaque bloc divisée (la phase de merge de l'algorithme Split and merge) et après les renvoyer à l'agent Superviseur.

2.2. Fonctionnement de l'algorithme Split and merge :

Tout d'abord, l'image est divisée en régions homogènes, puis les régions adjacentes qui répondent à de fusion sont fusionnées.

Soient $X_i, i=1, \dots, N$ l'ensemble des régions constituant une partition de l'image I (on peut avoir $N=1, X_1=I$). Soit H un prédicat d'homogénéité.

- **Etape 1 : SPLIT (algorithme récursif)**

Pour toute région X_i Si $H[X_i]$ faux Alors
Split (X_i).

- **Etape 2 : MERGE**

Pour tout couple (X_i, X_j) Si Voisin (X_i, X_j) et $H[\text{Union}(X_i, X_j)]$
Alors
Fusion (X_i, X_j)

➤ **Phase de Split (Division) :**

L'algorithme se résume en les étapes suivantes :

1. Diviser l'image initiale en 4 régions (principe de structure quartée) où l'on a
 $H(\text{images}) = \text{Faux}$.
2. Si pour une région R_i , le prédicat d'homogénéité H n'est pas vérifié
 $(H(R_i) = \text{faux})$, alors diviser la région en 4.
3. Aller à (2). L'algorithme s'arrête lorsque toutes les régions satisfont le prédicat H .

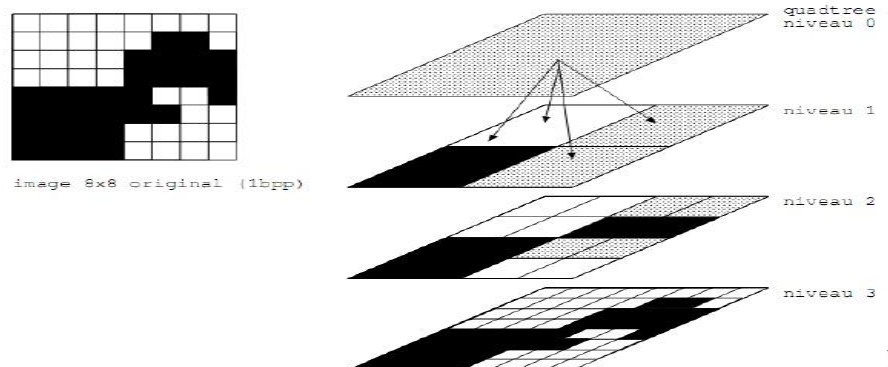


Fig.5.3. Découpage par quadtree d'une image 8x8

➤ **Phase de Merge (fusion) :**

L'algorithme commence par une image sur-segmentée, pour ensuite :

- a) Fusionner tout couple de régions adjacentes qui vérifie un critère d'homogénéité.
- b) Définir un prédicat Fusionne (R_i, R_j) où R_i et R_j sont deux régions adjacentes.

3. Quelques interfaces de l'application:

➤ **Fenêtre principale :**



Fig.5.4. Fenêtre principale de l'logiciel.

- **Fenêtre de choix de l'image** : la possibilité de choisir une image.

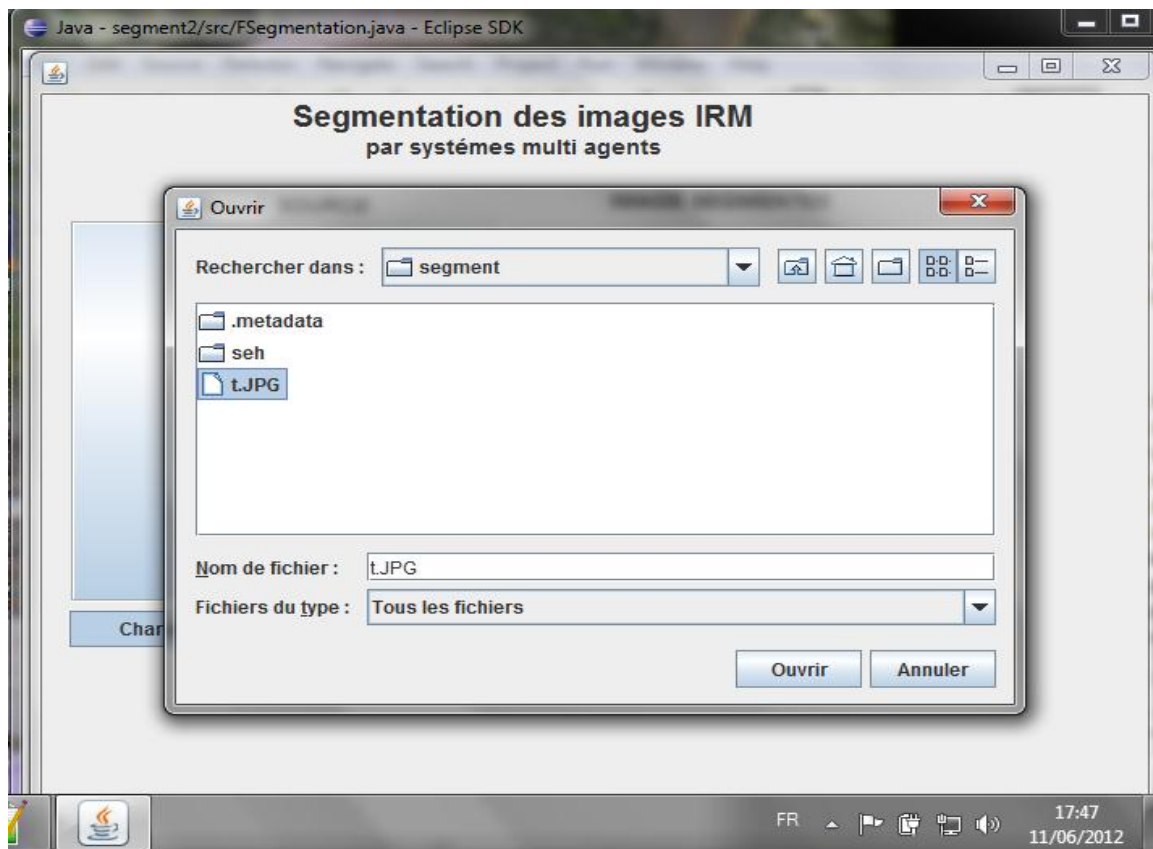


Fig.5.5. Charger l'image à partir d'un répertoire.

- Fenêtre de la segmentation d'image IRM : charger l'image

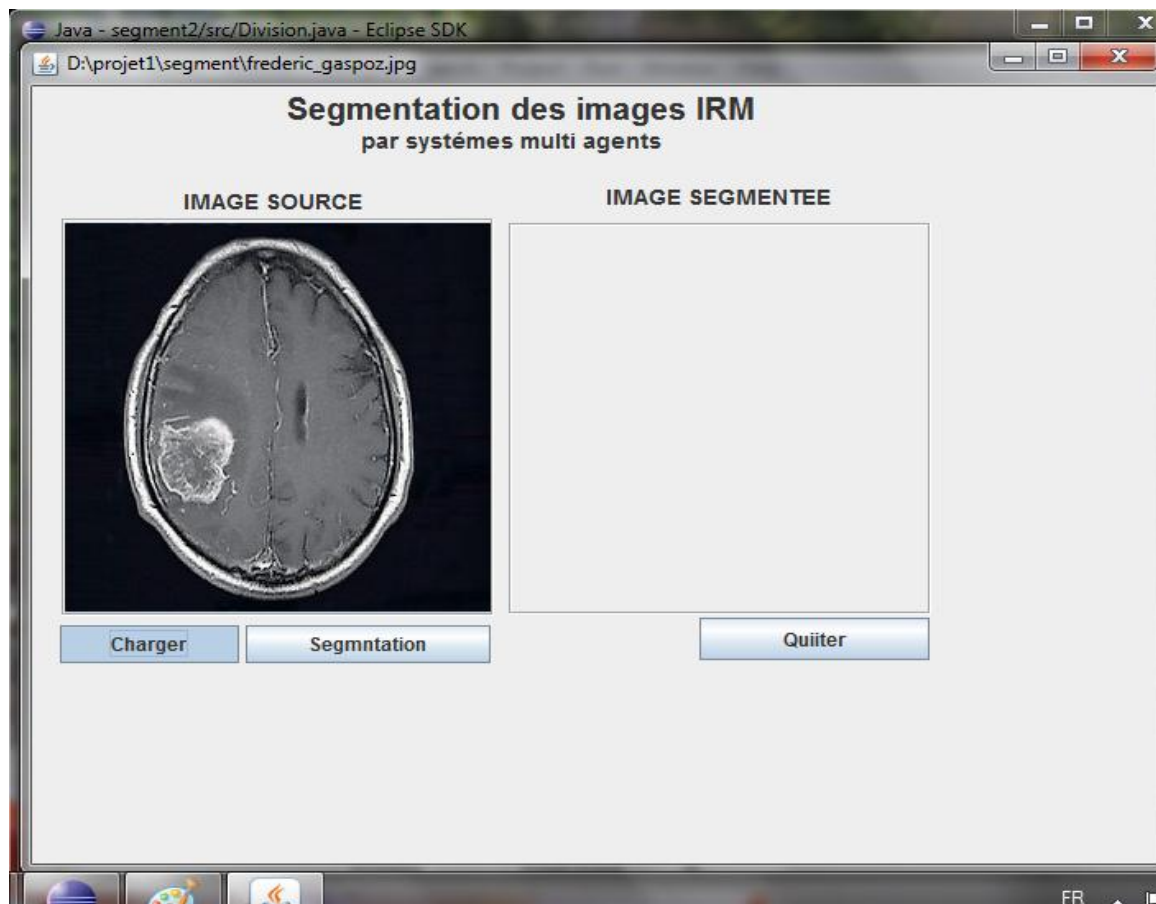


Fig.5.6. Charger l'image pour la segmentation.

➤ **Résultat de la segmentation :**

- Début de la segmentation

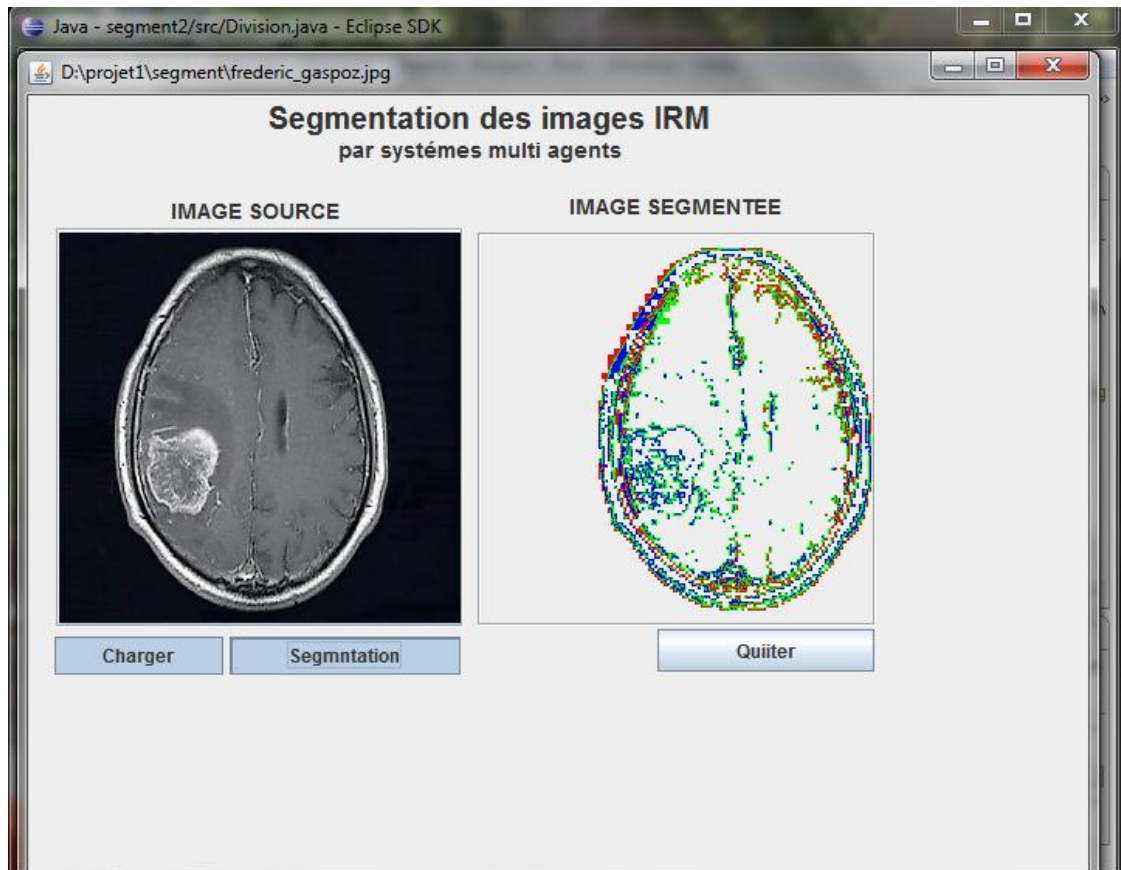


Fig.5.7.Résultat de la segmentation.

- **Résultat finale de la segmentation**

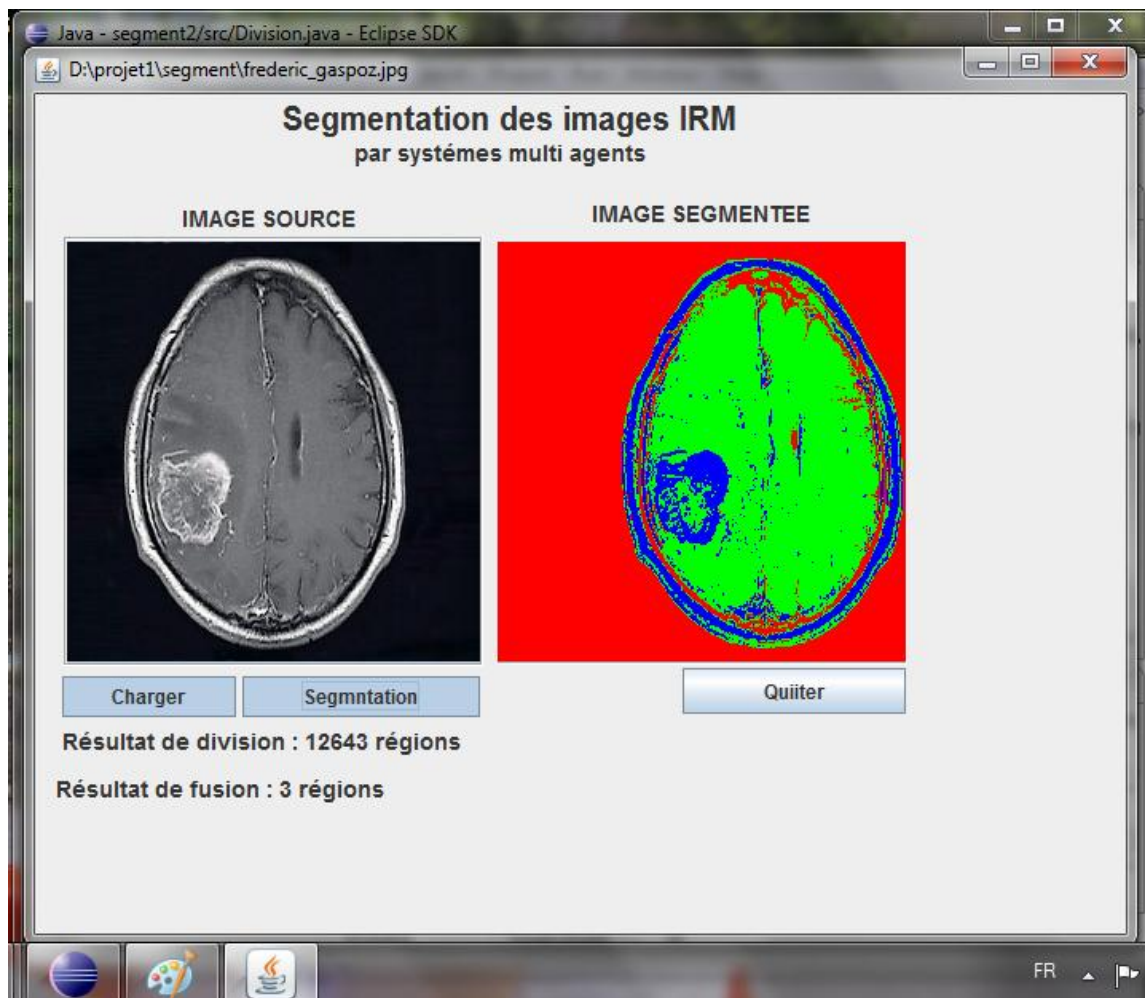


Fig.5.8. Résultat finale de la segmentation.