Universidad Autónoma Metropolitana-Iztapalapa

Plugin para OsiriX:
Segmentación por Corrimiento de Media, ponderado con Mapas de Confianza

Tesis presentada para obtener el grado de
Maestra en Ciencias (Ingeniería Biomédica)

Ing. Silvia Eunice Vides Cañas

Octubre de 2007

Dirigida bajo la supervisión de:

Dr. José Joaquín Azpíroz Leehan

Dr. Juan Ramón Jiménez Alaniz

Sinodales:
Dra. María Elena Martínez Pérez
Dr. Humberto Cervantes Maceda
Dr. Juan Ramón Jiménez Alaniz

octubre de 2007

A todos los pueblos latinoamericanos y sobretodo a mi El Salvador...
Porque a partir de su sufrimiento, creo que las cosas sí se pueden cambiar...

... si el desarrollo es el nuevo nombre de la paz, los pueblos que viven en subdesarrollo son una provocación continua de violencia (Homilía 3 de julio de 1977 de Monseñor Oscar Arnulfo Romero, asesinado cobardemente el 24 de Marzo de 1980 por escuadrones de la muerte).

... La riqueza es necesaria para el progreso de los pueblos, no lo vamos a negar. Pero un progreso como el nuestro, condicionado a la explotación de tantos que no disfrutarán nunca los progresos de nuestra sociedad, no es pobreza evangélica. ¿De qué sirven hermosas carreteras y aeropuertos, hermosos edificios de grandes pisos si no están más que amasados con sangre de pobres que no los van a disfrutar? (Homilía 15 de julio de 1979, Mons. Romero).

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Las Instituciones de Educación Superior y en especial las Universidades desempeñan un rol de suma importancia en la formación de recursos humanos de alto nivel, en la generación de conocimiento, y en la creación, desarrollo, transferencia y adaptación de tecnología de manera que llegue a la sociedad moderna, lo cual se constituye un imperativo estratégico para el desarrollo nacional. Las Universidades son reconocidas cada vez más como un instrumento de desarrollo de ciudades, regiones y países, y están consideradas como factor clave para incrementar la competitividad y calidad de vida. Los resultados y las conclusiones de la investigación universitaria deben generar provecho y bienestar para los sectores de la sociedad en la que se desarrolla. Bajo esta perspectiva es muy importante fomentar la vinculación entre la investigación que se realiza en centros universitarios y los posibles usuarios finales, es decir, el mundo del trabajo y los sectores de la sociedad. Por supuesto, para el sector salud es importante la transferencia y unión entre investigación y clínica.

En el caso particular de México, las instituciones universitarias que investigan problemáticas relacionadas con el cuidado de pacientes no siempre transportan sus resultados al sistema hospitalario. Para que este traslado sea adecuado y continuo es necesario plantear métodos y requisitos con la finalidad de que la investigación sea aplicable al diagnóstico. Un caso particular puede ser el procesamiento de imágenes médicas.

El desarrollo de tecnologías para adquirir y procesar imágenes médicas tiene gran importancia para el diagnóstico en los sistemas de salud modernos. El volumen de las imágenes médicas que se generan en el ambiente clínico ha crecido, debido tanto al aumento del número de exámenes que se realizan, como al rango de modalidades disponibles. Según estudios sobre la calidad de operación y las características de los servicios en imagenología médica efectuados en México [1], existe un serio problema derivado de la combinación entre la obsolescencia del equipamiento y la excesiva descentralización

de los servicios médicos. Por lo tanto es fundamental obtener la mayor cantidad de información de los estudios que se logran realizar, ya que sería un gran apoyo en el diagnóstico médico.

El procesamiento de imágenes médicas y el reconocimiento de patrones son herramientas de gran interés para algunos centros universitarios mexicanos y se han estudiado durante mucho tiempo. El inconveniente es que los problemas son tomados de los hospitales, las soluciones son planteadas y desarrolladas en las Universidades, pero la retroalimentación hacia el ambiente clínico no se consuma.

Específicamente, para que el intercambio de desarrollos de software entre los laboratorios dedicados al procesamiento de imágenes y el sistema hospitalario se complete es necesario que dicho software sea amigable para el usuario, de bajo costo, estable, seguro y confiable.

El presente trabajo tiene por objetivo explorar y evaluar un mecanismo que facilite el transporte de proyectos de investigación desarrollados para el procesamiento de imágenes médicas, al entorno clínico para lograr que estos desarrollos tengan trascendencia para el cuidado de los pacientes.

Las aplicaciones de código abierto y de libre acceso ofrecen la ventaja de que pueden utilizarse y distribuirse sin costos y pueden modificarse sin necesidad de reportar los cambios. Por otro lado, las aplicaciones con arquitecturas basadas en plugins o módulos de extensión, permiten adaptar metodologías muy específicas a aplicaciones ya desarrolladas e instaladas sin afectar el código fuente principal de dicha aplicación. Un plugin, es entonces, una unidad o módulo de extensión que añade funcionalidad y procedimientos específicos que satisfagan necesidades particulares a una aplicación ya desarrollada. Estas características son adecuadas para poder cumplir con el objetivo de este proyecto. En procesamiento de imágenes médicas, OsiriX, satisface ambas características por lo que la propuesta de este proyecto, siendo una transferencia de tecnología, se fundamenta en la posibilidad de incorporar metodologías de procesamiento de imágenes previamente investigadas y probadas en los laboratorios del área de Procesamiento Digital de Señales e Imágenes Biomédicas, en un ambiente de software de libre acceso; con la intención de reducir costos y simplificar el uso de dichos procedimientos por usuarios no expertos. Esto se logra utilizando un sistema de software abierto, extensible, modular y no interpretado, como es OsiriX.

En imagenología médica un procedimiento muy usual y requerido es la segmentación o separación de estructuras anatómicas de tejidos. Para enfermedades cerebrales, la segmentación de imágenes de resonancia magnética es una herramienta que permite ubicar el tejido con daño, facilita el diagnóstico y la planeación de tratamiento. Debido

a ésto, se han explorado formas de hacer una segmentación de imágenes por resonancia magnética de cerebro automática y precisa. Mucho se ha investigado y entre los métodos de segmentación, el procedimiento de corrimiento de media ha entregado muy buenos resultados, debido a que no asume ninguna distribución de los datos, es decir, permite que los datos se agrupen guiados por sus propias características. Lo único que se asume es que existe una función de densidad de probabilidad (*fdp*), pero no se asume ningún modelo de ésta. Por este motivo, la segmentación por corrimiento de media que además se pondera con mapas de confianza de bordes, ha tenido resultados muy buenos en separación de materia gris, blanca y líquido cefalorraquídeo[2]. Esta metodología fue desarrollada en el Laboratorio de Investigación en Neuroimagenología por el Dr. Juan Ramón Jiménez en su tesis doctoral [3]. Este desarrollo se realizó en Matlab. Debido a ésto, consideramos que el primer procedimiento que debíamos incorporar en OsiriX es el de segmentación por corrimiento de media ponderado con mapas de confianza porque, como se mencionó anteriormente, ha probado ser un método muy robusto y eficaz, pero además nos permitiría plantear y evaluar la metodología de transporte al comparar los resultados con los obtenidos por el Dr. Jiménez-Alaniz.

Para hacer el plugin de segmentación en OsiriX se plantea generar un modelo de la segmentación por corrimiento de media ponderado con mapas de confianza, utilizar diagramas en Lenguaje de Modelado Unificado (UML, por sus siglas en inglés), hacer la implementación, evaluar los resultados y describir la metodología de transporte.

A continuación se hace una breve descripción del contenido de esta tesis:

El Capítulo 2 describe el método de segmentación por corrimiento de media ponderado con mapas de confianza. Este método ha demostrado ser muy robusto en la segmentación de imágenes cerebrales por Resonancia Magnética [2], [3] y, por este motivo, se eligió para construir el primer plugin dentro de OsiriX. En éste capítulo se presentan las ecuaciones generales de los mapas de confianza, del corrimiento de media y de la fusión y el podado de regiones. En el Capítulo 3 se hace una conceptualización de las arquitecturas basadas en plugins, en específico la de OsiriX y se hace una descripción de las herramientas disponibles para el desarrollo en Mac OS X que fueron utilizadas para el desarrollo del plugin. En el Capítulo 4 se presenta la herramienta de modelado del proyecto y los diagramas de UML utilizados para construir el plugin. En el Capítulo 5 se presentan los resultados de la implementación de la metodología de segmentación por corrimiento como plugin dentro de OsiriX. También se hace una comparación con los resultados obtenidos con la implementación en Matlab. Al final del capítulo 5 se presenta la discusión de los resultados y las conclusiones del proyecto.

Capítulo 2

Metodología

La segmentación o separación de estructuras de interés en una imagen es un campo esencial en el área de procesamiento de imágenes. Específicamente en procesamiento de imágenes médicas, la segmentación es importante para la extracción de características, mediciones y despliegue de imágenes. En ciertas aplicaciones se busca clasificar los pixeles de la imagen en regiones, por ejemplo en huesos, músculos o vasos sanguíneos, mientras que, en otras, se buscan regiones patológicas, como cáncer, deformaciones de tejido y esclerosis múltiple. En imágenes por resonancia magnética de cerebro uno de los objetivos es separar las regiones de materia gris (MG), materia blanca (MB) y líquido cefalorraquídeo (LCR), esto es debido a que para el estudio, diagnóstico, planeación de cirugía o tratamiento de muchas enfermedades cerebrales, es necesario conocer con la mejor precisión posible el tipo de tejido o en algunos otros casos, la estructura anatómica afectada.

La segmentación de un objeto se logra ya sea identificando todos los pixeles o voxeles que pertenecen al objeto o localizando aquellos que forman su frontera. Algunos métodos están basados principalmente en la intensidad de los pixeles, pero otros atributos, como la localización espacial, pueden ser asociados con el pixel para realizar la segmentación, cuyo objetivo principal es dividir una imagen en regiones, también llamadas clases, que son homogéneas con respecto a una o mas características.

La delineación manual de MG, MB y LCR en imágenes de IRM por un experto humano es difícil porque requiere un conocimiento anatómico detallado y un examen cuidadoso de grandes cantidades de datos, su realización manual es tediosa, consume mucho tiempo, y por lo tanto es propensa a errores debido a la complejidad de las estructuras cerebrales; mostrando probablemente una gran variabilidad intra e interobservador. Sin embargo, la ventaja de la técnica manual es que refleja la interpretación del radiólogo, que es la única verdad valida disponible. Los resultados serán mas reproducibles porque

los métodos automatizados producirán los mismos resultados para los mismos datos. La confiabilidad aumenta porque los errores debidos a la fatiga se eliminan. La variabilidad puede disminuirse con las técnicas automáticas mientras se conserva alta la confiabilidad. Por lo anterior, el diseño de métodos de segmentación automáticos y confiables es altamente deseable por lo que ha sido uno de los intereses del Laboratorio de Investigación en Neuroimagenología (UAM- Iztapalapa) que ha desarrollado metodologías para la segmentación automática. Entre los métodos de segmentación desarrollados, existe uno de particular interés por el hecho de que no asume ninguna distribución de los datos, es decir no asume ningún modelo de función de distribución de probabilidad (*fdp*) sino que permite que los datos formen grupos en base a sus propias características. Ese procedimiento se conoce como segmentación por corrimiento de media y por las características que presenta, ha dado buenos resultados sobre imágenes médicas, particularmente en imágenes de resonancia magnética (IRM). El presente proyecto busca explorar formas de transporte de los desarrollos en el área de procesamiento de imágenes hechos en los laboratorios de investigación hacia ambientes hospitalarios, y se eligió como primer procedimiento a implementar, el proceso de segmentación propuesto por Jiménez-Alaniz[3], [4], [5], [2] para la detección de la moda y agrupamiento basado en el procedimiento de corrimiento de media. Se eligió, como se mencionó anteriormente, debido a los excelentes resultados obtenidos por este procedimiento en IRM cerebrales.

2.1. Segmentación por Corrimiento de Media ponderado con Mapas de Confianza

Como se mencionó anteriormente, la segmentación por corrimiento de media ponderado con mapas de confianza, es una metodología que ha probado ser muy robusta ante la presencia de diferentes tipos y niveles de ruido y ha sido desarrollada en el Laboratorio de Investigación en Neuroimagenología (UAM- Iztapalapa). Esta metodología se basa en el cálculo del corrimiento de media, el cual estima las modas locales de la función de densidad de probabilidad para definir los centros de clase en el espacio característico. Se hace una mejora en la calidad de las fronteras entre las regiones o clases utilizando un mapa de confianza de bordes, esto es, la medida de la confianza de que el pixel analizado es un borde. El corrimiento de media se pondera con la información del mapa de confianza de cada pixel, de esta manera, las técnicas de segmentación por región y la detección de bordes se combinan para generar resultados más precisos. En la Figura 2.1 se muestra el proceso planteado por [3]. Primero se hace el cálculo del mapa de

confianza de bordes de la imagen, con esta información se calcula el corrimiento de media, se hace un filtrado de la imagen, pero como se ha añadido la información del mapa de confianza, se preservan los bordes. Cuando se han obtenido las clases o regiones y sus centros, se hace una fusión de dichas regiones utilizando criterios de adyacencia y fuerza de bordes. Finalmente, se hace un podado de las regiones demasiado pequeñas y se sustituye cada pixel con el valor de la moda calculada.

En el presente proyecto se realizó el procedimiento mostrado en la Figura 2.1, para la segmentación de la imagen, sin embargo, para la clasificación de las regiones segmentadas en las estructuras de materia gris, materia blanca y líquido cefalorraquídeo hay un paso que no se incluye en esta implementación, el cual es la clasificación a través de un atlas probabilístico de fondo, materia gris, materia blanca y líquido cefalorraquídeo, para lo cual se debe hacer el registro al espacio de Talairach y luego la correlación de los atlas probabilísticos con la imagen segmentada (regiones resultantes del podado).

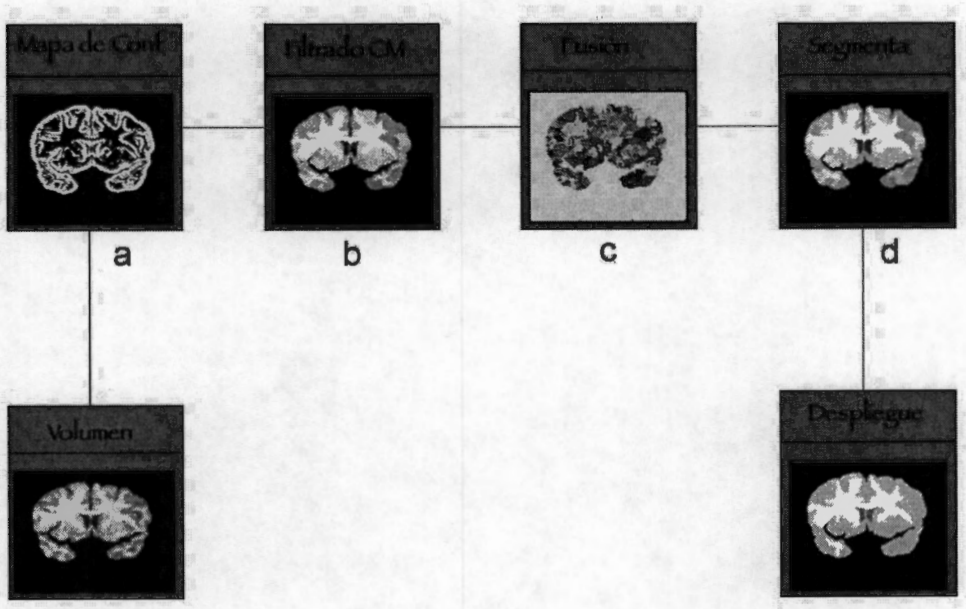


Figura 2.1: Metodología de segmentación por corrimiento de media ponderado con mapas de confianza de bordes, planteada por [3]. El primer paso (a) es el cálculo del mapa de confianza, es decir los bordes de la imagen. El segundo paso (b) es el filtrado por corrimiento de media. El tercer paso (c) es la fusión de regiones y el podado de regiones muy pequeñas y el paso (d) es la sustitución de cada pixel con la moda calculada.

2.1.1. Imágenes por Resonancia Magnética (IRM)

Aunque entre las modalidades de imágenes médicas la más común sigue siendo la de los rayos X, en los exámenes clínicos ahora se incluye la Tomografía Computarizada (CT), la resonancia magnética (RM), el ultrasonido y la medicina nuclear (NM) entre otros. Las imágenes de resonancia magnética (IRM) son altamente efectivas para observar la anatomía cerebral porque poseen excelente discriminación de tejidos suaves y alta resolución espacial. El análisis morfométrico proporciona mediciones cuantitativas de localización, volumen, forma y homogeneidad de componentes de las estructuras cerebrales. Este tipo de análisis, en conjunción con observaciones neuropsicológicas, neurológicas, psiquiátricas y acopladas con neuroimágenes funcionales se puede usar para responder interrogantes acerca de la estructura y función cerebral, tanto de sujetos sanos como enfermos. La segmentación de imágenes del cerebro es de interés en el estudio de muchos desórdenes, tales como la esclerosis múltiple, esquizofrenia, epilepsia, mal de Parkinson, enfermedad de Alzheimer, atrofia cerebral o asimetría. Otras aplicaciones de la segmentación incluyen: simulación y planeación de cirugías, medición de volumen de tumor, mapeo funcional, registro de imágenes, identificación de áreas anatómicas de interés para diagnóstico, y correlación mejorada de áreas anatómicas de interés con métricas funcionales localizadas.

En la IRM, los tejidos están caracterizados por intensidades de pixel resultado de señales de Resonancia Magnética (RM) diferenciadas espacialmente. Por lo tanto, la meta de la segmentación es dividir una imagen en regiones significativas, donde los pixeles de cada región deben ser homogéneos de acuerdo a una propiedad específica. Dado que los pixeles de un tejido están conectados espacialmente, el problema de la segmentación es encontrar las propiedades o estadísticas del tejido y etiquetar los pixeles.

La segmentación es la separación de estructuras de interés del fondo y de entre ellas mismas, es una función de análisis esencial para la que se han desarrollado muchos algoritmos en el campo del procesamiento de imágenes. En imágenes médicas, la segmentación es importante para la extracción de características, mediciones y despliegue de imágenes. En algunas aplicaciones se busca clasificar los pixeles de la imagen en regiones, tales como huesos, músculos y vasos sanguíneos, mientras que, en otras, en regiones patológicas, como cáncer, deformaciones de tejido y esclerosis múltiples. En estudios de IRM del cerebro el objetivo es dividir la imagen en subregiones de materia gris (MG), materia blanca (MB) y líquido cefalorraquídeo (LCR) [3].

2.1.2. Descripción de la metodología del Corrimiento de Media

La segmentación por espacio característico manual puede producir resultados incorrectos, debido a la delineación manual y la asociación con tipos de tejido hechas por el usuario. También, la clasificación puede variar de experto a experto y aún puede haber diferencias por fatiga, en imágenes clasificadas por el mismo experto. Para evitar este tipo de errores, la segmentación debe ser manejada por la aplicación, es decir, soportada por información con un alto nivel de independencia. Para esto se requiere de una representación bastante confiable de los datos de entrada. Lo anterior se puede lograr por el análisis de imágenes basado en el espacio característico, pero sin la intervención del usuario.

Un espacio característico es un mapeo de la entrada obtenido por el procesamiento de los datos en subconjuntos pequeños a la vez. Para cada subconjunto se obtiene una representación paramétrica de la característica de interés, y el resultado está mapeado hacia un punto en el espacio multidimensional del parámetro. Después de procesar la entrada, las características significativas corresponden a las regiones más densas en el espacio característico, es decir, a grupos (clusters). El objetivo del análisis de la imagen es la delineación de estos grupos.

En el enfoque de agrupamiento no paramétrico basado en la estimación de densidad, el espacio característico de una imagen se puede considerar como la función de densidad de probabilidad (*fdp*) empírica de parámetro representado. Las regiones densas en el espacio característico corresponden al máximo local de la *fdp*, esto es, a las modas de una función desconocida. Una vez que se determina la ubicación de una moda, el grupo asociado con ella se delinea basándose en la estructura local del espacio característico.

En muchos problemas de reconocimiento de patrones, se dispone de muy poca información acerca de la *fdp* verdadera y de su forma. Debido a esta falta de conocimiento se debe confiar en técnicas no paramétricas para obtener las estimaciones del gradiente de la densidad. Fukunaga y Hostetler [6], en su planteamiento para la detección de la moda y agrupamiento basado en el procedimiento de corrimiento de media, derivan un kernel general de estimación del gradiente multivariado usando las estimaciones de kernel de Parzen.

Una vez estimado el gradiente de densidad de probabilidad como el gradiente de la estimación de densidad, basándose en una función de kernel diferenciable, se aplica una función de kernel para entender el proceso involucrado en la estimación del gradiente, llegando a la interpretación intuitiva de que la estimación del gradiente de densidad es

esencialmente una medida ponderada del corrimiento de la media de las observaciones alrededor del punto de análisis. La interpretación intuitiva se demuestra sustituyendo el kernel de Epanechnikov [7] en la expresión de la estimación del gradiente de densidad, derivando la expresión de lo que se llama el corrimiento de media muestral [6],[3], que, como se deducirá más adelante, está dada por: $M_h(X) = \frac{1}{k} \sum_{X_i \in S_h(X)} (X_i - X)$, de las observaciones en una región o hiperesfera $S_h(X)$ alrededor de un punto X. El tamaño de la región $S_h(X)$ está dado en función del radio de una ventana h, y k que es el número de observaciones X_i que caen dentro de $S_h(X)$. Si el gradiente en X o la pendiente es cero, correspondiendo a una densidad uniforme sobre la región $S_h(X)$, el promedio del corrimiento de media será cero debido a la simetría de las observaciones cercanas a X. Sin embargo, con un gradiente de densidad diferente de cero apuntando en la dirección de mas rápido incremento de la *fdp* en el promedio, más observaciones deben caer a lo largo de su dirección que en cualquier otra dentro de $S_h(X)$. Así, el corrimiento de media promedio debe apuntar en esa dirección y tener una longitud proporcional a la magnitud del gradiente.

2.1.3. Mapas de Confianza de bordes

Una de las técnicas de extracción de características muy importante para el tipo de segmentación que se propone es la detección de bordes. Entre los métodos de detección de bordes, el de la estimación de la orientación y magnitud del gradiente es muy usado y se basa en la coherencia de los bordes cercanos a la dirección del gradiente. La estimación de la orientación y la magnitud del gradiente se realiza usando máscaras de diferenciación que se aplican sobre una ventana de la imagen. El problema que presenta el método es la dificultad de extraer una dirección de borde confiable. Este problema se puede resolver utilizando una medida de confianza del borde a través de la correlación entre una ventana $m \times m$ de datos y una serie de plantillas de borde ideal.
El cálculo de la medida de confianza (η) proporciona, por lo tanto, la seguridad de estar en presencia de un borde entre regiones adyacentes [6]. El mapa de confianza (MC), se define como la combinación lineal de la información del gradiente y la medida de confianza, y ayuda a mejorar la calidad de la segmentación debido a que preserva la información de los bordes. Matemáticamente se define por:

$$\varphi = \rho\beta + (1 - \beta)\,\eta \tag{2.1}$$

donde, ρ es la función de distribución acumulativa empírica con los rangos normalizados de la magnitud del gradiente de la imagen , β es el factor de mezcla entre ρ y η, es decir,

es el porcentaje o peso que se va a dar a estos valores y η es la medida de confianza del borde, que se define como:

$$\eta = \left| \boldsymbol{t}^{\mathrm{T}} \boldsymbol{a} \right| \tag{2.2}$$

Interpretado en el dominio de las imágenes, η es el valor absoluto del coeficiente de correlación entre los datos normalizados ($\boldsymbol{a}$) y la plantilla de borde ideal ($\boldsymbol{t}$) transpuesta.

2.1.3.1. Magnitud del Gradiente

Los puntos de contorno de un borde son las zonas de pixeles en las que existe un cambio abrupto de nivel de gris. Si pensamos en una imagen como una función continua $f(x, y)$, vemos que su derivada tiene un máximo local en la dirección de cambio de intensidad.

Es por esto que las técnicas más usadas en la detección de contornos se basan en la medida del gradiente [8].

La estimación del gradiente de una imagen, se lleva a cabo mediante un filtrado espacial empleando una máscara de diferenciación, $\boldsymbol{W}$, especificada en una ventana de $m \times m$ definiendo un subespacio de gradiente en $R^{m \times m}$.

El gradiente en un punto de la imagen se obtiene, entonces, mediante el producto interior de una ventana ($m \times m$) de datos de la imagen por dos máscaras de diferenciación en las dos direcciones perpendiculares (x, y) de la cuadrícula de la imagen.

El gradiente en x, se obtiene por diferenciación de renglones y suavizado de columnas simultáneo. Esto se hace a través del producto exterior de dos secuencias unidimensionales $s(i)$ y $d(i)$, con $i = -m, .., 0, ..m$.

La máscara en x, será:

$$\boldsymbol{W}_{dx} = \boldsymbol{s}\boldsymbol{d}^{\mathrm{T}} \tag{2.3}$$

donde $\boldsymbol{s}$ y $\boldsymbol{d}$ son secuencias de suavizado y diferenciación respectivamente definidas por los polinomios de Krawtchouk [9], [10], [11] que producen filtros para datos con pesos binomiales obtenidos de:

$$w(i) = \frac{1}{2^{2m}} \left(\frac{2m}{m+i} \right) = \frac{1}{2^{2m}} \frac{(2m)!}{(m-i)!\,(m+i)!} \tag{2.4}$$

para $i = -m, .., 0, ..m$.

Con estos pesos se calcularan las secuencias de la siguiente manera:

$$s(i) = w(i) \tag{2.5}$$

$$d(i) = \frac{2i}{m} w(i) \tag{2.6}$$

El gradiente en y, se obtiene diferenciando en columnas y suavizando los renglones, por lo tanto la máscara en y será la transpuesta de la máscara en x.

$$\boldsymbol{W}_{dy} = \boldsymbol{W}_{dx}^{\mathrm{T}} = \boldsymbol{ds}^{\mathrm{T}} \tag{2.7}$$

Con estas máscaras de diferenciación numérica se obtienen las derivadas en x y en y para cada pixel de la imagen:

$$G_x = \boldsymbol{W}_{dx}\boldsymbol{A} \tag{2.8}$$

$$G_y = \boldsymbol{W}_{dy}\boldsymbol{A} \tag{2.9}$$

donde $\boldsymbol{A}$ es una ventana de datos de $m \times m$ centrada en el pixel, (x, y).
Luego, el módulo del gradiente estimado en el pixel es:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.10}$$

que corresponde a la magnitud del gradiente de la imagen.
La magnitud del gradiente se normaliza en rangos y se calcula la función de distribución acumulativa empírica para poder usarla como criterio de selección para la presencia de un borde. Los rangos normalizados son los percentiles de la distribución acumulativa de los valores de la magnitud del gradiente, y se les denomina ρ y, como se mencionó anteriormente, se utiliza en el cálculo del mapa de confianza (Ecuación 2.1) .

2.1.3.2. Medida de Confianza

Para la medida de confianza (η) es necesario calcular la dirección del gradiente, es decir, hacia donde apunta el máximo cambio de intensidad para ese pixel, porque se asume que a 90° de este ángulo del gradiente, se encuentre un borde.
Así,

$$\hat{\theta}_b = \hat{\theta} - 90° \tag{2.11}$$

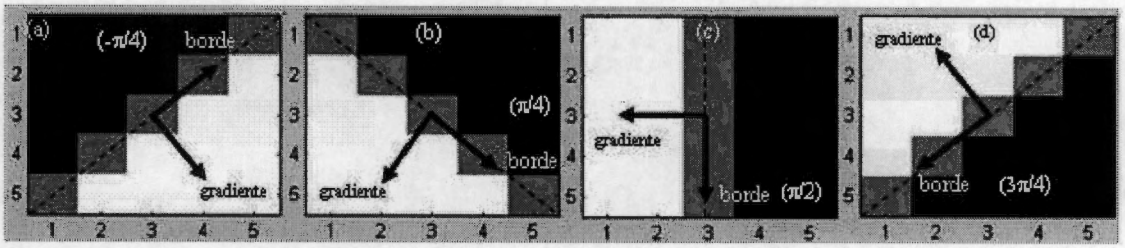


Figura 2.2: Ejemplos de plantilla de borde ideal de 5x5, para calculo de la medida de Confianza

donde, $\hat{\theta}_b$, es la estimación del ángulo del borde y $\hat{\theta}$, es el ángulo del gradiente que esta dado por $\hat{\theta} = \arctan\left(\frac{G_y}{G_x}\right)$.
Por lo tanto, una vez determinada la orientación del borde, a partir de la orientación estimada del gradiente, se elige la plantilla de borde ideal con la orientación determinada y se hace la correlación con la ventana de $m \times m$ pixeles, centrada en el pixel bajo análisis. Si el dato verdaderamente forma parte de un borde, la medida de confianza será 1 y 0 en caso contrario. Estas plantillas van desde $-180°$ a $180°$, y son matrices de 5×5, como se muestra en la Figura 2.2.

2.1.4. Corrimiento de Media

El método de segmentación que se utiliza en este proyecto es no paramétrico y está basado en la estimación de una función de densidad de probabilidad.
La estimación de densidad es la construcción de un estimado de la función de densidad de los datos. La estimación paramétrica supone que los datos son extraídos de una familia de distribuciones paramétricas conocida, por ejemplo la distribución normal con media μ y varianza σ^2. La densidad se puede estimar encontrando estimaciones de μ y σ^2 de los datos y substituyendo sus estimaciones en la fórmula de la densidad normal. La desventaja de la aproximación paramétrica es que algunas veces puede ser muy rígida. Para evitar dicha rigidez, se hace lo que se conoce como una aproximación no paramétrica, en la cual los mismos datos guiarán a la estimación que mejor los ajusta, a diferencia de lo que pudiera ser cuando esta restringida a una familia paramétrica dada.
En el caso de datos univariados, un estimador de densidad muy usado es el histograma, sin embargo para el caso multivariado, este método presenta dificultades. Entre otros métodos para la estimación de densidad de probabilidad se encuentran por ejemplo el estimador ingenuo (naive), que puede ser visto como una sumatoria de "cajas" centradas en las observaciones, por lo que presenta algunos brincos (debidos a que utiliza una

función w de pesos fijos) poco convenientes. Otro es el estimador por kernel, que es una generalización del estimador ingenuo, el cual reemplaza la función de pesos w por una función kernel K, la cual debe cumplir con ser continua y diferenciable. Este estimador de kernel es la suma de "lóbulos" (protuberancias) colocadas en las observaciones. La forma de los lóbulos depende del estimador K y el parámetro del ancho de ventana está dado por h, que también es llamado el parámetro de suavizado o ancho de banda y determina el ancho de cada lóbulo.

En materia de imágenes, se sugiere estimación no paramétrica, debido a que los espacios característicos de una imagen muchas veces están determinados por grupos de datos muy irregulares, cuyo número y formas no están disponibles. En el enfoque de agrupamiento no paramétrico basado en la estimación de densidad, el espacio característico de una imagen se puede considerar como la función de densidad de probabilidad (*fdp*) empírica del parámetro representado. En la estimación no paramétrica debe proporcionarse una detección confiable del máximo local de la densidad fundamental, i.e., las modas, porque son los centros de las regiones o "clusters", debido a que las regiones densas en el espacio característico corresponden al máximo local de la *fdp*, esto es, a las modas de la función desconocida. Una vez que se determina la ubicación de una moda, el grupo asociado con ella se delinea basándose en la estructura local del espacio característico. Para cierta dispersión en los datos (que es el caso de las IRM), la estimación por kernel es bastante adecuada, es sencilla y para kernels que obedecen condiciones suaves, el promedio de las estimaciones se aproxima a la función desconocida.

El estimador de densidad de kernel multivariado, $\hat{f}(x)$ con kernel K y ancho de ventana h se define por [2]

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left\{\frac{1}{h}(\boldsymbol{x} - \boldsymbol{X}_i)\right\} \tag{2.12}$$

donde la función kernel $K(\boldsymbol{x})$ es una función definida para $\boldsymbol{x}$ d-dimensional.

El kernel óptimo, es decir, el que minimiza el Error Cuadrático Integrado Medio (ECIM) es el Epanechnikov, que para el caso multivariado está dado por:

$$K_e(\boldsymbol{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)\left(1 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}\right) & si\ \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} < 1 \\ 0 & de\ otra\ manera \end{cases} \tag{2.13}$$

donde c_d es el volumen de la esfera unitaria en el espacio euclidiano $\boldsymbol{R}^d$ d-dimensional.

El uso de un kernel diferenciable permite definir la estimación del gradiente de densidad como el gradiente del estimador de densidad de kernel multivariado; así de

la expresión 2.12:

$$\hat{\nabla} f(\boldsymbol{x}) \equiv \nabla \hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K\left\{\frac{1}{h}(\boldsymbol{x} - \boldsymbol{X}_i)\right\} \tag{2.14}$$

Sustituyendo 2.13 en 2.14 y denotando con subíndice E para enfatizar que se utiliza el kernel de Epanechnikov:

$$\hat{\nabla} f_E(\boldsymbol{x}) = \frac{1}{n(h^d c_d)} \frac{d+2}{h^2} \sum_{\boldsymbol{X}_i \in S_h(\boldsymbol{x})} [\boldsymbol{X}_i - \boldsymbol{x}] = \frac{n_x}{n(h^d c_d)} \frac{d+2}{h^2} \left(\frac{1}{n_x} \sum_{\boldsymbol{X}_i \in S_h(\boldsymbol{x})} [\boldsymbol{X}_i - \boldsymbol{x}] \right) \tag{2.15}$$

donde la región $S_h(X)$ es una hiperesfera de radio h teniendo el volumen $h^d c_d$, centrada en x, y conteniendo n_x puntos de datos. El último término de 2.15:

$$M_h(\boldsymbol{x}) \equiv \frac{1}{n_x} \sum_{\boldsymbol{X}_i \in S_h(\boldsymbol{x})} [\boldsymbol{X}_i - \boldsymbol{x}] = \frac{1}{n_x} \sum_{\boldsymbol{X}_i \in S_h(\boldsymbol{x})} \boldsymbol{X}_i - \boldsymbol{x} \tag{2.16}$$

es lo que se conoce como corrimiento de media muestral.
La cantidad $\frac{n_x}{n(h^d c_d)}$ es la estimación $\hat{f}(x)$ de la densidad del kernel calculada en la hiperesfera $S_h(X)$ (el kernel uniforme), y así se puede escribir 2.15 como:

$$\hat{\nabla} f_E(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) \frac{d+2}{h^2} M_h(\boldsymbol{x}) \tag{2.17}$$

donde $M_h(\boldsymbol{x})$:

$$M_h(\boldsymbol{x}) = \frac{h^2}{d+2} \frac{\hat{\nabla} f(\boldsymbol{x})}{\hat{f}(\boldsymbol{x})} \tag{2.18}$$

De esta última expresión se puede decir que el procedimiento del corrimiento de media es una técnica de optimización ascendente con el mejor paso, con un tamaño de paso variable que es proporcional a la magnitud del gradiente, que calcula el vector de corrimiento de media (Ecuación 2.18) para cada uno de los datos, corre o traslada el kernel por esta cantidad, y repite los cálculos hasta que se alcanza una moda. Así, los datos se dividen en grupos basándose solamente en sus trayectorias del corrimiento de media, y cuando dos puntos de datos convergen a la misma posición final, se considera que pertenecen al mismo grupo (ver Figura 2.3). El resultado o propósito del agrupamiento por corrimiento de media es usar un máximo local como una caracterización de los datos, este máximo local se considera como el centro de la agrupación o clase.

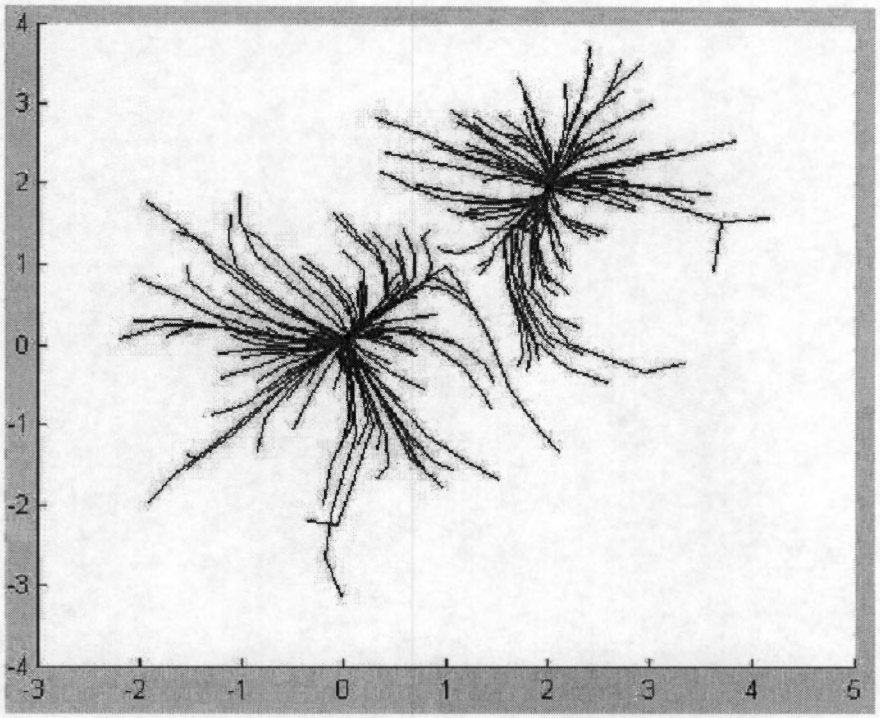


Figura 2.3: Trayectorias definidas por el corrimiento de media para una mezcla de gaussianas con media 0 y 2 y varianza 1 [3].

La segmentación y filtrado de una imagen conservando los bordes de alta calidad se pueden obtener aplicando el corrimiento de media en el domino combinado espacial-rango.

El método está basado en la misma idea de correr iterativamente una ventana de tamaño fijo hacia el promedio de los datos dentro de ella. Los detalles de la imagen se conservan debido al carácter no paramétrico del análisis, que no supone a priori alguna estructura particular de los datos. La segmentación basada en agrupamiento (clustering) supone estructuras constantes por tramos; sin embargo, se generan problemas con regiones difíciles de reasignar o recuperar cuando se presentan cambios lentos de intensidad o cuando hay regiones de alta densidad. Estos problemas se resuelven tomando en cuenta la información espacial, i.e., procesando en el dominio espacial-rango. Esto se refiere a utilizar la información espacial (x, y) de los pixeles de la imagen, pero además incluir otra característica, por ejemplo el nivel de gris, el color o la información espectral. En este proyecto se utiliza el nivel de gris en el dominio del rango. Al aplicar el algoritmo de corrimiento de media en el dominio conjunto espacial-rango, cada dato se asocia a un punto de convergencia que representa la moda local de la densidad en el espacio d-dimensional. En el proceso se definen el ancho de ventana espacial (h_e), y el ancho de ventana de intensidad (h_i) correspondiendo al rango, tomando en cuenta simultánea-

mente tanto la información espacial y de rango. La salida del filtro de corrimiento de media para un pixel de la imagen está definida como al información de rango alcanzada por el punto de convergencia. Este proceso logra un filtrado espacial de alta calidad que preserva discontinuidades.

Para realizar la tarea de segmentación, los puntos de convergencia suficientemente cercanos en el dominio conjunto son considerados como los centros de clase, y todos aquellos puntos que fueron llevados por el corrimiento de media al mismo punto, i.e., a una moda de la densidad, se fusionan para obtener las regiones homogéneas de la imagen.

El procedimiento para la segmentación por corrimiento de media en el dominio espacial-rango tiene el mismo diseño del filtrado (es un suavizado con preservación de bordes), es decir, se hace el reemplazo de pixeles tomando información de una ventana dinámica de la imagen que se mueve en la dirección del máximo incremento del gradiente. Cada dato se asocia a un punto de convergencia que representa la moda local de la densidad en el espacio d-dimensional. Por lo tanto, el filtrado por corrimiento de media se adapta mejor a la estructura local de los datos.

El mapa de confianza ayuda a mejorar la estimación por corrimiento de media (CM), ya que se pondera cada pixel dentro de una región por una función de confianza de borde, tal que los pixeles que están situados cerca del borde (confianza cercana a uno) influyan menos en la determinación del nuevo centro del agrupamiento. La ecuación del CM modificada, que incluye la ponderación de la confianza del borde se deriva de la Ecuación 2.16 de la siguiente manera:

$$M_h(\boldsymbol{x}) = \frac{1}{\sum(1-\varphi_i)} \sum_{\boldsymbol{X}_i \in S_h(\boldsymbol{x})} (1-\varphi_i)\, \boldsymbol{X}_i - \boldsymbol{x} \tag{2.19}$$

donde φ_i (de la Ecuación 2.1) es la confianza del borde asociado a $\boldsymbol{X_i}$.

Este nuevo vector de corrimiento de media (Ecuación 2.19) se usará en la etapa de filtrado para obtener regiones homogéneas en intensidad de las imágenes procesadas.

2.1.5. Fusión de Regiones

Cuando ya se ha filtrado la imagen es necesario agrupar con criterios de proximidad en los dominios de espacio (en 2D) y rango, esto quiere decir que todos los pixeles que convergieron al mismo punto son fusionados y etiquetados, para obtener regiones homogéneas.

Después, se refina esta selección incorporando una matriz de adyacencia de regiones para luego reducir el número de regiones con el algoritmo de encontrar y unir (unión-find). En la Matriz de Adyacencia de Regiones (MAR) se almacena un valor booleano para establecer si hay un borde (arista) entre dos regiones (vértices). Para construirla se

toma la imagen filtrada y se etiqueta, considerando cada etiqueta como una región y poniendo un valor de 1 entre regiones adyacentes y 0 de otra forma. De esta manera se identifica el vecindario de cada región resultante.
La operación de fusión une las regiones que tienen modas asociadas que están localizadas dentro de una distancia de separación de $h_i/2$, y satisface la condición de fuerza de borde débil. La fuerza del borde se calcula promediando los valores de los pixeles del mapa de confianza a la largo de la frontera que separa las regiones adyacentes. De esta manera, se fusionan las regiones adyacentes cuyas modas asociadas, en el dominio del rango, están localizadas dentro de una distancia predefinida unas de otras. La fusión de las regiones se realiza siempre y cuando se cumpla la condición de que la distancia, medida en modas entre dos regiones adyacentes, no sea mayor a 0.5. La distancia entre modas, i.e., la diferencia entre modas se normaliza con respecto al radio del rango h_i, por lo que la condición de fusión se traduce a que la distancia entre modas no sea mayor de $h_i/2$. Si se cumple la condición anterior y la fuerza del borde entre regiones es menor que un umbral, ξ, esto es, tienen un borde débil entre ellas, indicando que realmente no hay una frontera entre las regiones, entonces las regiones se fusionan en una sola. El proceso de fusión es iterativo con la condición de paro de que el número de regiones a la salida del proceso sea igual al número de regiones a la entrada del proceso; es decir, que ya no haya una fusión posible con los criterios utilizados.

2.1.5.1. Podado

Una vez que se ha obtenido el resultado final de la fusión de regiones de la imagen, se aplica un proceso de podado para remover las regiones cuya área (en número de pixeles), sea inferior al umbral, μ, definido por el usuario. El podado se hace recorriendo la MAR para unir, a la región candidata respectiva, las regiones que sean muy pequeñas. La región candidata es la que cumple con ser adyacente a la que se está podando, también que la distancia, medida como diferencia de modas, es de un mínimo, o es la única región adyacente que tiene un área mayor que el umbral. Después se toman las modas calculadas resultantes de cada región final, se asignan etiquetas a las nuevas regiones, y el proceso de podado se repite iterativamente hasta que el número de regiones no cambie o se alcance un número dado de iteraciones. Al terminar el podado se obtiene la imagen con el número de regiones reducidas al mínimo posible con los criterios utilizados, es decir tenemos la imagen segmentada.
La metodología para la segmentación por corrimiento de media ponderado con mapas de confianza, planteada para IRM por Jiménez-Alaniz [3], [2] se resume en 4 pasos fundamentales que son: el cálculo del mapa de confianza de la imagen, este cálculo

es la combinación lineal entre la función de distribución acumulativa empírica de la magnitud del gradiente de la imagen y la medida de confianza, que es el resultado de la correlación de ventanas de la imagen, con plantillas de borde ideal, definidas por la orientación del gradiente. Con la información del mapa de confianza se calcula el corrimiento de media, es decir se hace un filtrado de la imagen preservando bordes. El corrimiento de media estima las modas locales de la función de densidad de probabilidad para definir los centros de clase en el espacio característico, dando como resultado una imagen filtrada con preservación de bordes, conteniendo regiones de intensidad homogénea. Después del filtrado de la imagen, se construye un grafo de adyacencias que se analiza para fusionar las regiones adyacentes que están separadas por un borde de magnitud pequeña, calculada a partir del mapa de confianza, y su diferencia en modas de intensidad es menor de un valor predeterminado. Posteriormente, la imagen fusionada se somete a un proceso de podado, con el propósito de unir las regiones que tienen pocos pixeles, con la región adyacente que tiene un número suficiente de pixeles y una diferencia mínima en intensidad.

Capítulo 3

Herramientas de Desarrollo

El objetivo del presente trabajo, como se ha mencionado anteriormente, es buscar y evaluar herramientas que permitan el paso de los desarrollos de procesamiento de imágenes médicas hechos en laboratorios de investigación hacia el ambiente hospitalario para su uso en el diagnóstico. Es importante que el traslado sea sencillo y que ofrezca confiabilidad hacia los usuarios finales y que a su vez, sea práctico para los desarrolladores que van a dedicarse a hacer ese transporte. Para cumplir este objetivo, se evaluaron las posibilidades que ofrece la arquitectura basada en plugins porque es una opción versátil, flexible y adaptable que permite incorporar funciones a una aplicación ya instalada, esto se conoce como "extensibilidad tardía". Es importante que la aplicación sea de Código Abierto [12],[13], es decir, que haya libertad de conocer y modificar su código fuente porque así puede mejorarse o adaptarse a la necesidad particular. OsiriX, es un software de procesamiento de imágenes que además de ser una gran herramienta para visualizar y manipular grandes cantidades de imágenes, es de código abierto y posee una arquitectura basada en plugins. En este capítulo se describe de manera conceptual la arquitectura basada en plugins de OsiriX y cómo el proceso descrito en el capítulo anterior se incorpora a dicha arquitectura.

3.1. Arquitecturas basadas en Plugins

Desde hace algunos años los usuarios de servicios de cómputo han presenciado el surgimiento de muchas aplicaciones que pueden ser extendidas con módulos o programas de extensión, llamados Plugins. Entre estas aplicaciones se pueden mencionar web browsers, ambientes de desarrollo integrado (IDE, por sus siglas en inglés), herramientas gráficas e incluso algunas aplicaciones de mensajes. Estas aplicaciones se caracterizan por el hecho de que una vez que han sido instaladas, pueden evolucionar cuando los

usuarios finales añadan plugins para extender su funcionalidad inicial.

Las aplicaciones basadas en plugins son construidas con la idea de que tengan *extensibilidad tardía*. En ciencias de la computación, la extensibilidad representa la habilidad de añadir nuevas características a un programa existente sin alterar el código existente. La extensibilidad tardía ocurre cuando la adición de características se lleva a cabo después de que la aplicación ha sido instalada [14].

Un plugin es una unidad de extensión para aplicaciones cuya arquitectura permite la introducción de funciones por parte de usuarios finales, una vez que la aplicación ha sido desarrollada, a esto se le conoce como *arquitectura basada en plugins*. Los plugins se adaptan en lugares muy específicos, son opcionales y la aplicación a la que extienden funciona sin ellos. Sin embargo, los plugins si dependen de la aplicación y no se pueden ejecutar sin ella.

Por lo tanto, algunas características de los plugins son:

1. Unidad de extensión para aplicaciones ya desarrolladas
2. Añade funcionalidad sin afectar el código
3. Es introducido por usuarios finales en lugares específicos, una vez que la aplicación ha sido instalada
4. Son opcionales
5. La aplicación no depende de ellos
6. Ellos si dependen de la aplicación para funcionar
7. Están limitados por la arquitectura de la aplicación

En la Figura 3.1 se muestra un esquema general de la ubicación de un plugin dentro de la aplicación principal. Es muy importante notar que los plugins no intervienen con el cuerpo o "core" de la aplicación y tienen una forma y localización muy particular para funcionar con la aplicación basada en plugins.

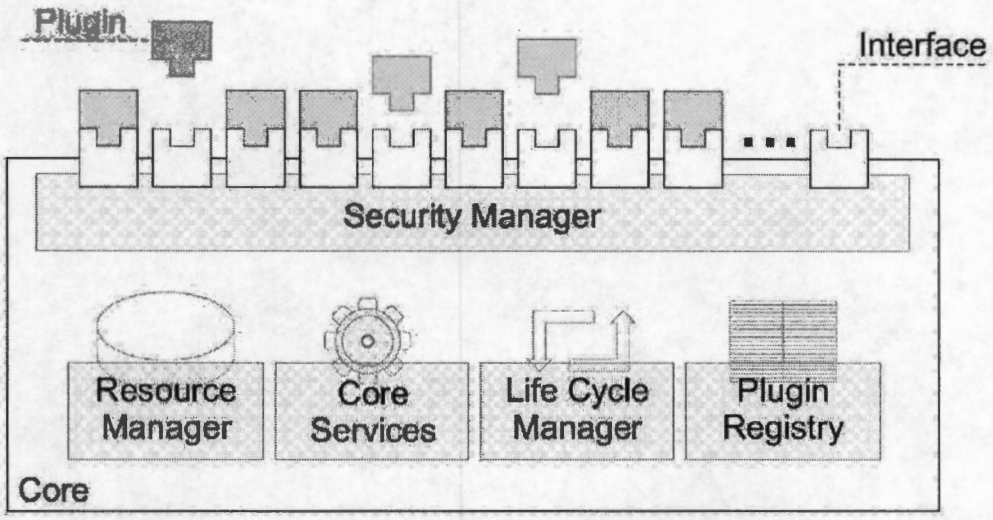


Figura 3.1: Arquitecturas basadas en plugins

3.2. OsiriX

OsiriX, es un programa de código abierto dedicado al procesamiento de imágenes en formato DICOM (Digital Imaging and Communications in Medicine), estándar en imagenología médica [15], [16]. Se distribuye bajo la Licencia General Pública de GNU (GPL, por sus siglas en inglés, de GNU). Presenta la ventaja de tener una arquitectura extensible con plugins que añaden una característica o función específica a OsiriX. Estos plugins pueden utilizar todas las funciones de manejo y visualización que tiene OsiriX. El programa ha sido implementado como una aplicación de OS X (sistema operativo de los sistemas Apple Macintosh), basada en varios elementos de código abierto, como se muestra en la Figura 3.2. Entre los elementos mencionados están librerías como VTK, ITK (ver Apéndice B), la librería de clases Cocoa que agrega muchas ventajas para el desarrollo de plugins en OsiriX.

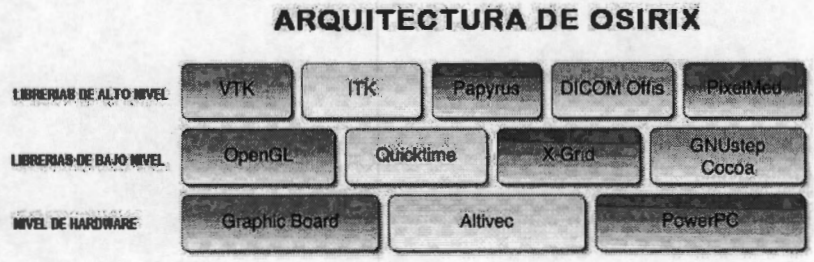


Figura 3.2: Arquitectura de OsiriX basada en elementos de código abierto

Cocoa, es una colección de clases divididas en: Foundation, que es el conjunto elemental de utilidades (caracteres, datos, listas, etc.) y AppKit, que se refiere a interfaces gráficas de usuario (botones, ventanas, dibujos, etc.) [17]. La construcción de un plugin tiene acceso inmediato a todas estas clases.

El lenguaje de programación utilizado en el proyecto de construcción de un plugin, es Objective-C, un lenguaje orientado a objetos, que se basa en SmallTalk-80 y algunas librerías de C [18]. Objective-C es parte del ambiente de desarrollo de la Fundación para el Software Libre de GNU.

Otra herramienta, es el ambiente de desarrollo, Xcode, que es una aplicación que permite compilar, revisar y ejecutar programas en forma sencilla.

Es muy importante enfatizar que un plugin para OsiriX es un proyecto de desarrollo en Xcode, que se vincula a una librería de clases llamada Cocoa para facilitar el desarrollo de dicho proyecto y Objective-C es el lenguaje de codificación.

La interface de plugins en OsiriX permite tener acceso a los datos DICOM y provee elementos para trabajar con ellos. Con este propósito la interface de plugins consiste de las siguientes clases, cuyos encabezados o *headers* son: `PluginFilter.h`, `DCMPix.h`, `DCMView.h`, `dicomFile.h`, `MyPoint.h`, `ROI.h` y `ViewerController.h`. El plugin en sí es una subclase de la clase `PluginFilter`.Esta clase es diseñada para que solo una instancia de cada plugin sea creada en el inicio de OsiriX y dicha instancia es reutilizada cada vez que se llama al plugin.

Para poder cumplir el objetivo de implementar metodologías de procesamiento de imágenes, en particular, la segmentación por corrimiento de media, en aplicaciones que permitan su utilización en ambientes clínicos, una buena herramienta son las arquitecturas basadas en plugins, que además son de código abierto. OsiriX cumple con ambas características, y además ofrece herramientas de visualización y manejo de imágenes que fortalecen su inserción en el medio hospitalario. Para poder introducir el método de segmentación por corrimiento de media ponderado con mapas de confianza como un plugin para OsiriX es necesario utilizar herramientas de desarrollo en MAC OS X y definir el punto de contacto entre el nuevo plugin y OsiriX. En la Figura 3.3 se observan los puntos de interface que OsiriX ofrece para añadir un plugin, es decir, el plugin puede ser filtro de imágenes, herramienta de ROI (región de interés) u "otros". Esto, en la ventana de OsiriX, se muestra en la Figura 3.4 hay opciones de menú específicas para ubicar al plugin en desarrollo (Imagen Filter, ROI, others).

En `Info.plist` se elige en que parte del menú aparecerá el nuevo plugin y si éste va a tener, a su vez, submenús (ver Apéndice C). La ubicación del plugin es opcional para el desarrollador pero es conveniente que el tipo de proceso corresponda al menú donde

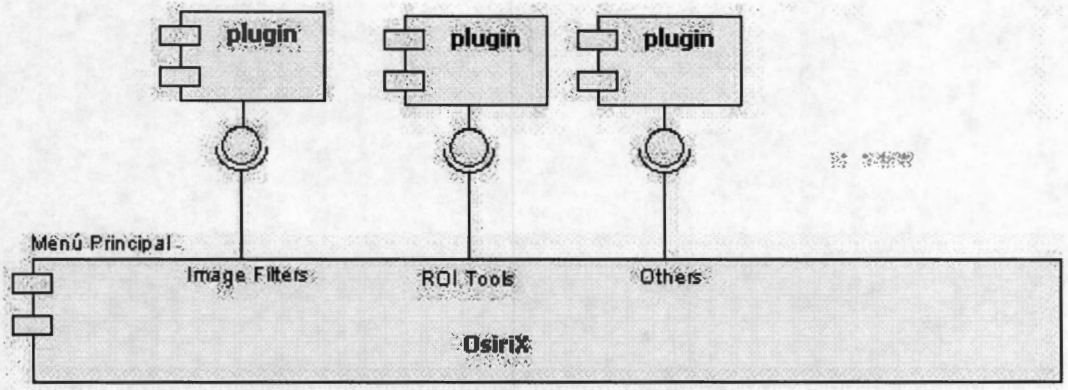


Figura 3.3: Diagrama de componentes de la arquitectura basada en plugins de OsiriX donde se observan los puntos específicos en donde OsiriX permite la adición de un plugin (Image Filters, ROI Tools y Others). Los puntos suspensivos significan que en el futuro, en nuevas versiones de OsiriX, probablemente hayan otros tipos de plugin, y que el enlace con OsiriX será semejante.

se ubica el plugin, por ejemplo, si es un plugin para hacer procesamiento sobre una región de interés (ROI), tendría que estar en el menú de ROI Tools, si es un filtro sobre imágenes, debe ser parte de Image Filters, y para cualquier otra aplicación, hay una opción de Others (otros). Como puede verse en la Figura 3.3estos son los puntos específicos donde incorporar el nuevo plugin. Lo importante para la comunicación, como se mencionó anteriormente y se muestra en el Apéndice C, no sólo es el menú, sino también que el plugin sea subclase de `PluginFilter`. El desarrollo que se hizo en este proyecto es un filtro de imágenes, pero como se muestra en la Figura 3.4, es decisión del desarrollador donde poner el plugin en el menú, y para este caso, se decidió poner todos los desarrollos en el menú Others. En la Figura 3.4 se observan las dos opciones que presenta este plugin (*Segmentación por Rebanada(s)* y *por Ajuste de Parámetros*), como se verá en el Capítulo 4, corresponden a los dos casos de uso de este proyecto.

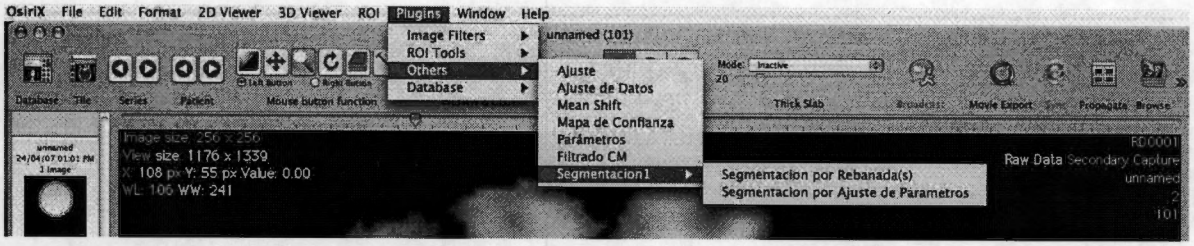


Figura 3.4: Menú de plugins dentro de OsiriX, se observan las tres posibilidades: Image Filters (filtro de imágenes), ROI Tools (herramientas para una región de interés) y Others (otro tipo). Se observa, también, la opciones del plugin de Segmentación (*Segmentación por Rebanada(s)* y *por Ajuste de Parámetros*)

3.3. Herramientas de desarrollo en MAC OS X.

3.3.1. XCODE

Cuando se escriben programas en MAC OS X, se tienen dos opciones para compilar y ejecutar dichos programas. La primera, es utilizar el compilador GNU de Objective-C, también llamado GCC, y para revisar el programa utilizar el debugger de GNU o GDB, ambos desde la ventana de Terminal de la máquina. Esta opción no es muy práctica si lo que se busca es crear programas con cierto grado de complejidad.

La otra opción es utilizar Xcode, que es el ambiente de desarrollo integrado (IDE) que permite escribir código, compilar, revisar y ejecutar programas. Cuando la intención es desarrollar implementaciones como la de un plugin en OsiriX, Xcode ofrece muchas ventajas.

Xcode permite visualizar, en la misma ventana: El archivo principal o main, los directorios de las clases, tanto interfaces como implementaciones (archivos .h y .m); las interfaces gráficas (.nib), los directorios de recursos disponibles para aplicaciones y herramientas, por ejemplo el marco de trabajo o conjunto de clases y los ejecutables que utiliza la aplicación.

Con Xcode se construye y se ejecuta la aplicación, permite revisar errores y editar el código de los programas. También, Xcode permite crear nuevas librerías, nuevas clases y aplicaciones específicas en Cocoa [17], Carbon, Java o AppleScript. En la figura 3.5 se muestra la ventana principal para el desarrollo de un proyecto en Xcode. La barra superior muestra las herramientas para el desarrollo. En la barra vertical izquierda (Groups & Files) está la lista de clases (código fuente), que para el caso de un plugin para OsiriX, incluye los encabezados "OsiriX Headers" (Apéndice B). También se encuentra la información de recursos como Interfaces Gráficas, información sobre versión e idioma, el

programa de ejecución de ese proyecto, los marcos de trabajo o "frameworks" incluidos en el proyecto y la lista de errores de compilación. Xcode permite la edición y codificación en la ventana principal.

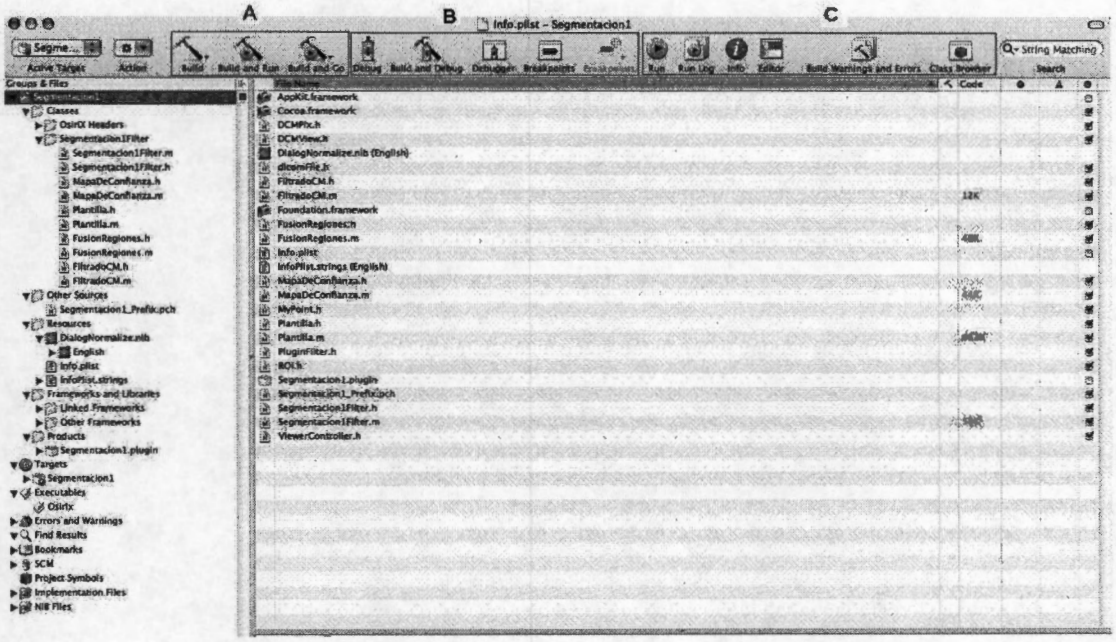


Figura 3.5: Ventana principal de Xcode donde se encuentran las herramientas de compilación (A), depuración (B) y ejecución (C) de las clases

3.3.2. OBJECTIVE-C

Objective-C, es un lenguaje diseñado por Brad J. Cox [18] al principio de los años 80. Este nuevo lenguaje estaba basado en un lenguaje llamado SmallTalk-80. Después, fueron incorporadas varias extensiones de C, con lo que era posible crear y manipular objetos.

NeXT Software creo la licencia del lenguaje Objective-C, en 1988 y desarrolló librerías y un ambiente de desarrollo llamado NeXTSTEP. En 1992, Objective-C fue incorporado al ambiente de desarrollo GNU de la Fundación para el Software Libre (FSL). Este software es de dominio público, lo cual quiere decir que, cualquiera que quiera aprender a programar en Objective-C, puede hacerlo con solo descargar de la red sus herramientas, de manera gratuita.

En 1994 NeXT Computer y Sun Microsystems realizaron una especificación estandarizada del sistema NeXTSTEP, llamada OPENSTEP. La implementación de OPENSTEP por parte de la FSL, se llama GNUStep [18].

En 1996, Apple Computer anunció que estaba por adquirir NeXT Software y el ambiente de desarrollo NeXTSTEP/OPENSTEP fue la base para el sistema operativo de Apple, OS X. La versión de Apple de este ambiente de desarrollo fue llamada Cocoa. Con el soporte del lenguaje Objective-C unido al desarrollo de herramientas como Project Builder (o su sucesor Xcode) e Interface Builder, fue creado un ambiente para el desarrollo de aplicaciones en MAC OS X [18].

3.3.2.1. Clases y Objetos

El lenguaje Objective-C, es un conjunto de instrucciones de ANSI C con extensiones para programación orientada a objetos. Estas extensiones permiten definir clases, categorías y protocolos, permiten crear objetos que provienen de clases y enviar mensajes o métodos entre objetos. Es decir, en Objective-C, se manejan objetos o estructuras que ocupan una cierta cantidad de memoria y que tienen algunas variables dentro, se manejan clases, que son estructuras que pueden crear objetos y se manejan mensajes o métodos.

En Objective-C, también, se utiliza una colección de clases que han sido diseñadas para ser utilizadas en conjunto. Este conjunto de clases o "framework", se conoce como Cocoa Framework.

Objective-C es un lenguaje reflexivo, esto es, que tiene la capacidad de modificar su estructura de alto nivel o que en tiempo de ejecución puede discernir sobre su propio procedimiento. Es un tipo de lenguaje que distingue entre tiempo de ejecución y tiempo de compilación y tiene diferencias entre compilar o interpretar el código y la reflexión. Esta propiedad, heredada de SmallTalk, le permite a Objective-C tener algunas características disponibles en tiempo de ejecución que, de otro modo, serían muy difícilmente realizables en lenguajes de más bajo nivel (C). Por ejemplo, descubrir y modificar construcciones de código fuente (tales como bloques de código, clases, métodos, etc.) como objetos de "categoría superior" en tiempo de ejecución. También, ser reflexivo le permite convertir una cadena que corresponde al nombre simbólico de una clase o función en una referencia o invocación a esa clase o función y poder evaluar una cadena como si fuera una sentencia de código fuente en tiempo de ejecución.

En Objective-C existen dos tipos de archivos: los *.m* (module) que tiene la implementación de las funciones (procedimientos externos) y métodos (procedimientos que pertenecen a una clase) y los *.h* (header) que tiene los prototipos de las funciones y clases.

3.3.3. COCOA

Cocoa es un conjunto de librerías y de directorios que contienen clases, métodos y documentación que facilitan el desarrollo de proyectos en Objective-C. Cocoa es un ambiente de desarrollo orientado a objetos para aplicaciones diseñadas exclusivamente en MAC OS X.

Cocoa es el resultado de una serie de avances y mejoras que se hicieron sobre las librerías de OPENSTEP y NeXTSTEP, en 1993. Es debido a este origen, que las clases de Cocoa inician con las letras NS (NeXTSTEP), por ejemplo, NSArray, NSString, etc.

Un marco de trabajo o "framework" es una colección de clases que se utilizan en conjunto. Esto significa que las clases fueron compiladas juntas en una librería reusable de código. Todos los recursos relacionados y la librería son puestos en el mismo directorio para ser rebautizado con la extensión .framework. Las aplicaciones de Cocoa utilizan dos de estos frameworks:

Foundation Proporciona la base o fundamento para todos los desarrollos de programas. Permite trabajar con objetos básicos, como números y palabras, y con colecciones de objetos, como arreglos, diccionarios y sistemas. También proporciona otras posibilidades, como trabajar con fechas y horarios, manejo automático de la memoria, trabajar con archivos, almacenamiento de objetos y estructuras de datos geométricas. Algunas clases importantes en Foundation framework son:

NSObject, que es la clase raíz para la mayoría de las jerarquías de clases en Objective-C. A través de NSObject, los objetos heredan una interfaz básica para el tiempo de ejecución y la habilidad de comportarse como objetos de Objective-C. Algunas de las clases heredadas de NSObject son: NSArray, que maneja arreglos estáticos de objetos; NSData, que maneja objetos estáticos de datos; NSString, representa objetos de caracteres; NSNumber, que ofrece una serie de métodos para el manejo y acceso de números escalares etc., hay 123 clases en Foundation framework.

Application Kit o AppKit Contiene una extensa colección de clases y métodos para desarrollar aplicaciones con interfaz gráfica. Estas, proporcionan la capacidad de trabajar en forma sencilla con textos, menús, herramientas, tablas, documentos y ventanas [18]. Algunas de estas clases son:

NSView, que es una clase abstracta que define un dibujo básico, manejo de eventos y despliegue de una aplicación; NSCell, es una clase que provee mecanismos para desplegar un texto o una imagen en un objeto NSView; NSImage, ofrece la posibilidad de manejar o crear imagen; NSMenu, que define un objeto que maneja el menú de una aplicación,

etc. , hay 161 clases disponibles en AppKit framework [17].

3.3.3.1. Archivos nib

La interfaz gráfica se guarda en un archivo "`.nib`" que significa "NeXT Interface Builder". Los archivos Nib son creados con una aplicación llamada *Interface Builder* que maneja todas las clases de `AppKit`. La clase principal del plugin solicita a `NSApplicationMain()` que llame al archivo Nib. Una vez que la aplicación ha cargado el archivo Nib, es el usuario el que toma las decisiones para que las diferentes clases funcionen.

3.4. Plugins para OsiriX

Para desarrollar un plugin en OsiriX, es necesario crear un proyecto en Xcode (archivo `.xcodeproj`) que contenga la siguiente información (ver Figura 3.6) :
A. Clases, es decir el código fuente (archivos .m y .h) de cada clase, donde se encuentra el procedimiento o la metodología de procesamiento. En este concepto se incluye la clase de interfaces gráficas, es decir los archivos .nib y toda la información referente a la ventana de interface.
B. Entorno de OsiriX, este aspecto se refiere a todos los archivos y la información que se requiere para hacer la interface entre el plugin y OsiriX. Aquí se encuentran los frameworks que se van a utilizar (Cocoa, por ejemplo), también se encuentra el subdirectorio con los encabezados de las 7 clases de interface de comunicación entre OsiriX y el plugin (OsiriX Headers), que son los siguientes[19]:

- `PluginFilter.h`, que siempre debe ser la superclase del nuevo plugin y que contiene métodos y variables de instancia básicos para el manejo de la imagen.
- `ViewerController.h`, es un objeto de la clase NSWindowController, que maneja la ventana de despliegue en 2D.
- `DCMViewer.h`, es una clase del tipo NSOpenGLView y contiene la imagen desplegada.
- `DCMPix.h`, es una clase que maneja los pixeles de la imagen.
- `dicomFile.h`, contiene la información del archivo DICOM.
- `ROI.h`, es la clase donde esta contenida una región de interés (ROI) de la imagen seleccionada.

- `MyPoint.h`, es la clase que describe un punto en 2D.

C. Información adicional, es decir, los archivos que proporcionan la ubicación y tipo de plugin, éstos son:

1. Los archivos `version.plist` e `Info.plist` que contienen información sobre el nombre y la ubicación del plugin dentro del menú de OsiriX (ImageFilter, ROI, others) y el número de versión.
2. El subdirectorio `English.lproj` que contiene datos para crear un plugin en otros lenguajes.
3. El archivo `.pch`, que contiene el índice del archivo.

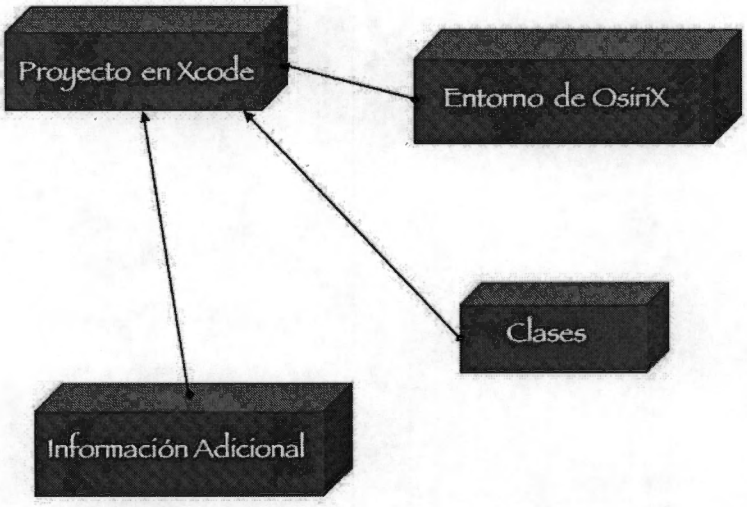


Figura 3.6: Diagrama de construcción del directorio para un plugin para OsiriX.

El procedimiento de creación de un plugin está resumido en la Figura 3.7 pero la construcción paso a paso del plugin se encuentra en el Apéndice C. De manera general, primero, el nuevo proyecto debe ser creado ya sea con el generador de plugins o con la opción de duplicar un plugin ya existente. El nuevo proyecto contiene la información mostrada en la Figura 3.6 y debe abrirse en Xcode para poder codificar las clases y definir la ubicación del nuevo plugin en el menú de OsiriX. Luego, es importante definir el ejecutable del proyecto que se esta construyendo, ya que Xcode es una herramienta general para MAC OS X. Luego se hace la compilación y depuración del plugin y

se coloca el archivo generado (es un archivo `.plugin`, que aparece en un directorio *Build* dentro del directorio del nuevo plugin) o un alias de éste en */library/Application Support/OsiriX/Plugins*. Lo último es ejecutar OsiriX y buscar el plugin en el menú, aplicarlo sobre una imagen y verificar que en tiempo de ejecución (runtime) no genere problemas. También es importante valorar el resultado sobre la imagen.

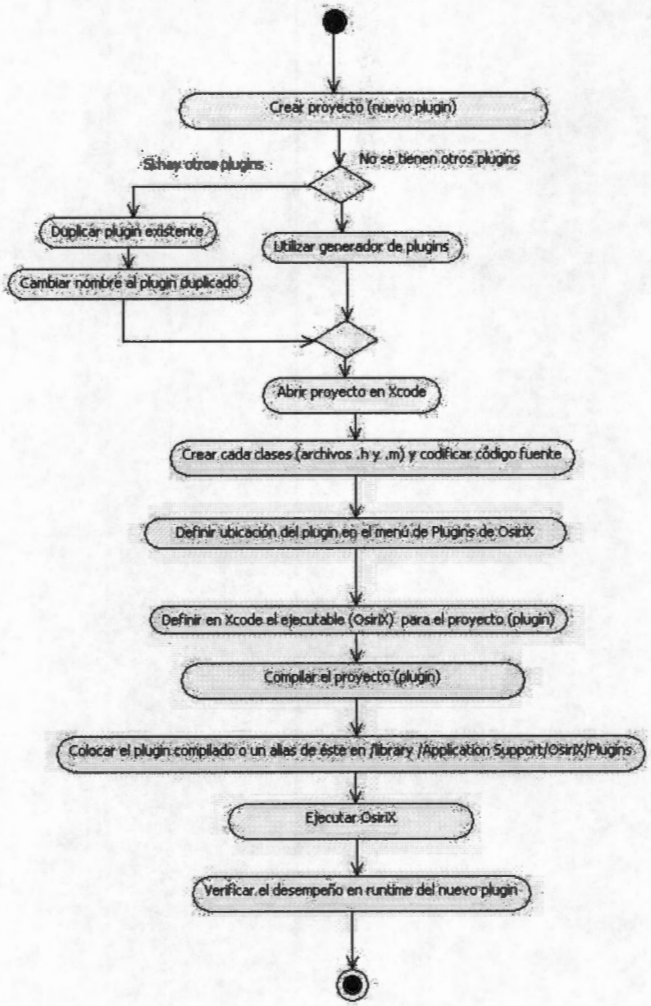


Figura 3.7: Diagrama de actividad para la construcción de un plugin. Primero, se genera el nuevo proyecto, ya sea con el generador de plugins o con la opción de duplicar un plugin ya existente. El nuevo proyecto debe abrirse en Xcode para poder codificar las clases y definir la ubicación del nuevo plugin en el menú de OsiriX. Luego, se define a OsiriX como ejecutable de ese proyecto, se compila y depura, si es necesario. El plugin compilado se coloca en el directorio Plugins, se ejecuta OsiriX para verificar el comportamiento del plugin al aplicarse sobre imágenes.

3.4.0.2. Interface OsiriX-Plugin

Como se mencionó anteriormente, el plugin es una subclase de `PluginFilter` y dentro de esta clase o superclase hay varios métodos de interés para el desarrollo del plugin:
`initPlugin`: es un método que reserva espacio de memoria para el plugin y es llamado al iniciar OsiriX.
`viewerControllersList`: es un método que da acceso a la lista de ViewerController de los conjuntos de datos abiertos.
`filterImage`: contiene el código fuente que va a ser ejecutado cuando el plugin es llamado, por eso se coloca allí el código de la clase principal del procedimiento.
Cuando OsiriX inicia, busca dentro del directorio de "*plugins*" localizado en */library/Application Support/OsiriX*, los archivos con la extensión `.plugin`. Después intentará encontrar si uno se estos archivos contiene una clase que es subclase de `PluginFilter` y si la clase tiene un método en particular llamado `filterimage`, porque la entrada de éste es el nombre del plugin que seleccionó el usuario y allí es donde debe comenzar la ejecución porque ese método indica a OsiriX el inicio de la clase principal y por lo tanto, el inicio del código que tiene que ejecutar.
El método `filterimage` es `((long) filterImage:(NSString*) menuName)` que, como se mencionó anteriormente, es el método principal del plugin. OsiriX siempre utilizará esta función para llamar al plugin. La cadena `menuName` contiene el menú seleccionado por el usuario [19].
Las aplicaciones de código abierto y con arquitecturas basadas en plugins ofrecen la posibilidad de incorporar metodologías muy específicas, es decir, permiten adaptar software a necesidades particulares. En el área de procesamiento de imágenes médicas, OsiriX permite navegar en grandes cantidades de imágenes, tiene funciones de visualización y procesamiento de imágenes, funciona como una estación de trabajo dentro de un esquema de PACS de software libre, de código abierto y que permite la extensibilidad con plugins. Para incorporar el método de segmentación por corrimiento de media como un plugin para OsiriX es necesario utilizar herramientas de desarrollo para MAC OS X y respetar los requerimientos de extensibilidad de OsiriX.

Capítulo 4

Modelado del Plugin

Entre los paradigmas de programación dominantes en la industria del software se destacan el paradigma procedural y el orientado a objetos. Haciendo una comparación, se considera que el paradigma orientado a objetos tiene ventajas en las posibilidades de modificación y expansión, así como en organización estructural sobre el paradigma procedural. OsiriX usa Programación Orientada a Objetos (POO) pero no está modelado como tal. Lo importante es que para el buen desarrollo de un plugin en OsiriX es recomendable generar un modelo orientado a objetos antes de implementar las clases para el procesamiento. Es posible utilizar funciones o codificar secuencialmente pero se desaprovecharían las ventajas de Objective C y Cocoa, además de las clases a las que OsiriX permite tener acceso.Por eso, antes de la implementación, se hizo un modelo de la segmentación por corrimiento de media ponderada con mapas de confianza. Se utilizó el Proceso Unificado de Desarrollo de Software (UP, por sus siglas en inglés) y el Lenguaje de Modelado Unificado (UML, por sus siglas en inglés) para este propósito.

4.1. Enfoque Estructurado y Enfoque Orientado a Objetos

En software, hay varias maneras de hacer un modelo. Las dos formas más comunes son el enfoque estructurado y el enfoque orientado a objetos.
La visión tradicional para desarrollo de software es el enfoque estructurado. En esta perspectiva, el bloque principal de construcción de todo el software es el procedimiento o la función. Esta perspectiva hace a los desarrolladores enfocarse en aspectos de control y en la descomposición de procedimientos grandes en otros más pequeños, como se muestra en la Figura 4.1. La desventaja de esta forma de modelado, es que tiende a

generar sistemas frágiles. Cuando los requerimientos cambian, el sistema crece y cuando los sistemas tienen esta perspectiva procedimental se vuelven difíciles de manejar.

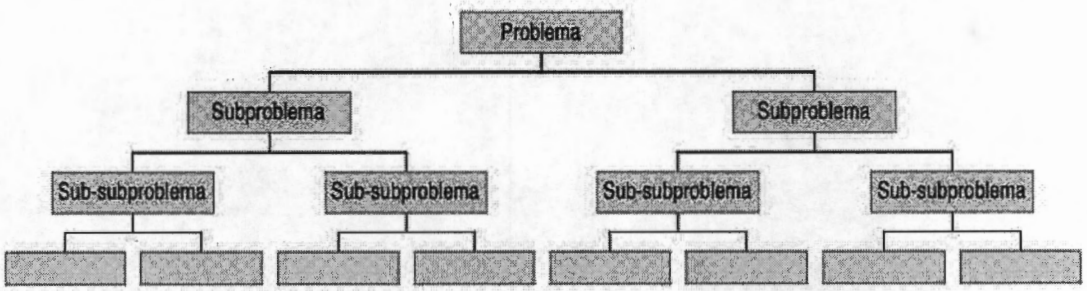


Figura 4.1: Enfoque estructurado o procedimental para la construcción de modelos de software. Hay relación de "divide y vencerás".

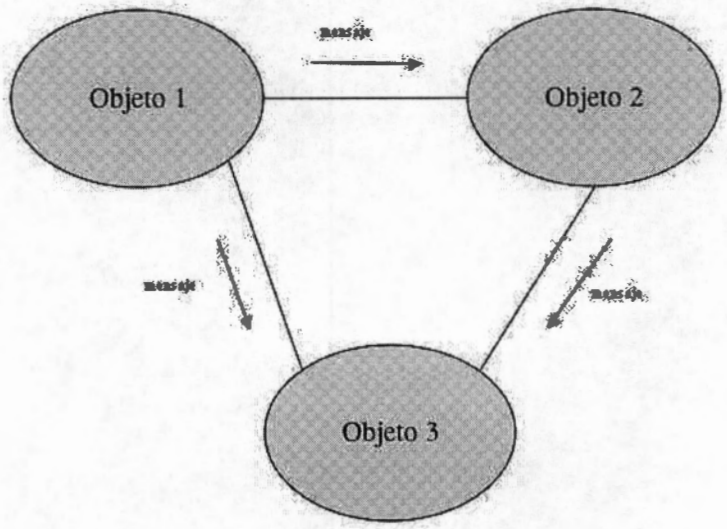


Figura 4.2: Enfoque de construcción de software orientado a objetos. Hay relaciones colaborativas entre los objetos

En la perspectiva orientada a objetos el bloque principal de construcción de todo el software es el objeto o la clase, como se muestra en la Figura 4.2. El acercamiento orientado a objetos para el desarrollo de software es parte de la estructura básica porque

ha demostrado ser de mucho valor en la construcción de sistemas en muchas áreas y se adapta a cualquier tamaño o complejidad. Muchos de los lenguajes contemporáneos, sistemas operativos y herramientas actuales son orientadas a objetos en cierta forma; ésto le da mayor sentido a ver el mundo en término de objetos. Al aplicar proceso de desarrollo de software y hacer un modelo, los resultados son la forma de visualizar el diseño y verificarlo contra los requerimientos del usuario antes de que se genere código [20], [21].

4.2. Proceso Unificado de Desarrollo de Software

Los requerimientos son la descripción de necesidades o funciones que se desean de un producto. Para transformar requerimientos del usuario en programas de software es necesario establecer algunos procedimientos, es decir, utilizar un proceso de desarrollo de software. El Proceso Unificado de Desarrollo de Software o simplemente Proceso Unificado (UP, por sus siglas en inglés) es un método o marco de trabajo que ayuda a organizar los avances de un proyecto. EL UP es considerado como centrado en arquitectura, iterativo incremental y guiado por casos de uso [22].

Centrado en arquitectura significa que la arquitectura del sistema es el plano guía para la conceptualización, construcción, manejo y evolución del sistema en desarrollo [20]. En el caso de los plugins, la arquitectura esta dada por la aplicación para la cual son creados. Por lo tanto la arquitectura no es una actividad de este proceso en particular.

El desarrollo *iterativo incremental* se refiere a un proceso iterativo que involucra el manejo del flujo de avances y un proceso incremental que implica una continua integración de la arquitectura del sistema para producir dichos avances. Cada nuevo avance aporta mejoras incrementales al paso anterior. Entonces, un desarrollo iterativo incremental es un proceso manejado por riesgos que se centra en obtener resultados en pasos pequeños y manejables y que cada nuevo paso ataca y reduce los riesgos mas significativos para obtener un resultado de éxito. Cuando todos los pasos planteados se han terminado, se obtiene un producto completo.

Un caso de uso es una herramienta para mejorar el entendimiento de los requerimientos. El caso de uso describe la funcionalidad de un sistema. En el desarrollo *guiado por casos de uso*, cada uno de los requerimientos debe ser expresado como un caso de uso y cada producto obtenido se prueba de manera que cumpla como tal [21]. El caso de uso es utilizado como un primer elemento para establecer el comportamiento deseado en el sistema, también sirve para validar la arquitectura, probar el sistema y establecer comunicación con los usuarios finales del proyecto.

El desarrollo guiado por casos de uso, centrado en arquitectura e iterativo incremental puede ser dividido en fases. Una fase es un periodo de tiempo entre dos resultados del proceso. Cuando en uno de estos periodos los objetivos se cumplen, los elementos propuestos se completan y se toman las decisiones para la siguiente fase, es posible continuar con el siguiente paso. Hay cuatro fases en el ciclo de vida del desarrollo de software: planeación, elaboración, construcción y transición. Cada uno de estos pasos es retomado una y otra vez hasta que el proceso se concluye [22].

4.3. Diagramas UML

Un modelo es una simplificación de la realidad, proporciona una conceptualización abstracta de una entidad o sistema y permite observarlo desde diferentes puntos de vista. En la construcción de software un modelo proporciona una mejor comprensión del sistema que se está desarrollando. Específicamente un modelo permite:

1. Visualizar un sistema existente o futuro. Los modelos ayudan a comprender, conceptualizar y visualizar cualquier tipo de sistema, incluso los ya existentes.
2. Comunicar decisiones o cambios en el proyecto. Un modelo puede ser comprendido por personas con diferentes conocimientos o experiencias, no necesariamente es para los desarrolladores del proyecto.
3. Documentar las decisiones tomadas en cada desarrollo.
4. Definir las partes de estructura (parte estática) y de comportamiento (parte dinámica) del sistema.
5. Servir como un plano guía para la construcción del sistema [22].

El Lenguaje de Modelado Unificado (UML) es un lenguaje gráfico estándar que puede ser utilizado para visualización, especificación, construcción y documentación de artefactos de un sistema intensivo de software [20]. UML es una herramienta en el proceso de desarrollo de software. UML es independiente del proceso, por lo tanto puede utilizarse en un proceso guiado por casos de uso, centrado en arquitectura e iterativo-incremental. A través de UML es posible capturar información sobre clasificación estructural, comportamiento dinámico y manejo del modelo del sistema de software [21]. Cuando se utilizan diagramas de UML, se construye el modelo con bloques básicos como son clases, interfaces, colaboraciones, componentes, nodos, dependencias, generalizaciones, y asociaciones. Los diagramas son el significado conjunto de estos bloques básicos. Un

diagrama es una representación gráfica de un conjunto de elementos, representado como un grafo conectado de vértices (elementos) y arcos (relaciones) [20]. Cada diagrama UML permite a los desarrolladores y a los usuarios finales ver un sistema desde diferentes perspectivas y en varios niveles de abstracción. Ningún sistema complejo puede ser comprendido completamente desde una sola perspectiva, por eso UML define una serie de diagramas para enfatizar diferentes aspectos del sistema, en forma independiente.
Utilizando UML un modelo esta compuesto con:

- Elementos (objetos y relaciones).
- Diagramas (construidos a base de los elementos).
- Vistas (diagramas que muestran diferentes perspectivas de un modelo) [23].

Diagramas Estructurales Los diagramas estructurales sirven para visualizar, especificar, construir y documentar los aspectos estáticos de un sistema. Esto es, el esqueleto o la estructura básica del sistema, como son las clases, componentes, interfaces, colaboración y nodos.
Estos diagramas están organizados en torno a un aspecto específico:

- Diagrama de Clases: muestra un conjunto de clases con relaciones de asociación y herencia entre ellas.
- Diagrama de Objetos: muestra objetos de instancia específicos y las relaciones entre ellos. Este diagrama representa una vista de los objetos en un determinado momento en el tiempo de ejecución.
- Diagrama de Componentes: representa los mayores componentes del sistema y como se integran. Estos diagramas pueden incorporar elementos de software no orientados a objetos, como son documentos web u otros archivos de código.
- Diagrama de Implantación: muestra los nodos de hardware en el sistema.

Diagramas de Comportamiento Los diagramas de comportamiento sirven para visualizar, especificar, construir y documentar los aspectos dinámicos del sistema. Esto es, las partes cambiantes del sistema, como el intercambio de mensajes en el tiempo o el movimiento físico de componentes a través de la red.
Estos diagramas están organizados alrededor de diferentes aspectos dinámicos como son:

- Diagrama de Casos de Uso: organiza el comportamiento del sistema. Muestra quien usa al sistema y que procesos puede ejecutar con él.
- Diagrama de Secuencia: se enfoca en como se lleva a cabo un proceso a través de objetos y actores. En este diagrama se hace énfasis en el orden temporal de los mensajes.
- Diagrama de Colaboración: esta centrado en la organización estructural de los objetos que envían y reciben mensajes.
- Diagrama de Estado: enfocado en el cambio de los estados de comportamiento de un objeto debido a los eventos que le acontecen.
- Diagrama de Actividad: está enfocado en el flujo de actividades en un proceso.

4.4. Modelado de Plugin de Segmentación por Corrimiento de Media

Para el desarrollo del plugin de Segmentación por Corrimiento de Media ponderado con Mapas de Confianza se utilizaron 4 diagramas UML, 3 de comportamiento (diagrama de Casos de Uso, Colaboración y de Secuencia) y un diagrama estructural (diagrama de Clases).

Diagrama de Casos de Uso Estos diagramas son muy importantes en la organización y el modelado de los comportamientos del sistema. Un diagrama de casos de uso captura la funcionalidad del sistema e ilustra el enfoque dinámico en forma global o parcial. Es la descripción de un conjunto de secuencia de acciones, incluyendo las variantes que el sistema ejecuta para entregar un resultado de valor para el actor. Lo que se busca con este primer modelo es definir los casos de uso del sistema en desarrollo, la relación con los actores y la relación entre casos. El diagrama de casos de uso de la Figura 4.3, ilustra las diferentes posibilidades en el funcionamiento del plugin. Hay 2 casos de uso o dos opciones posibles para realizar.

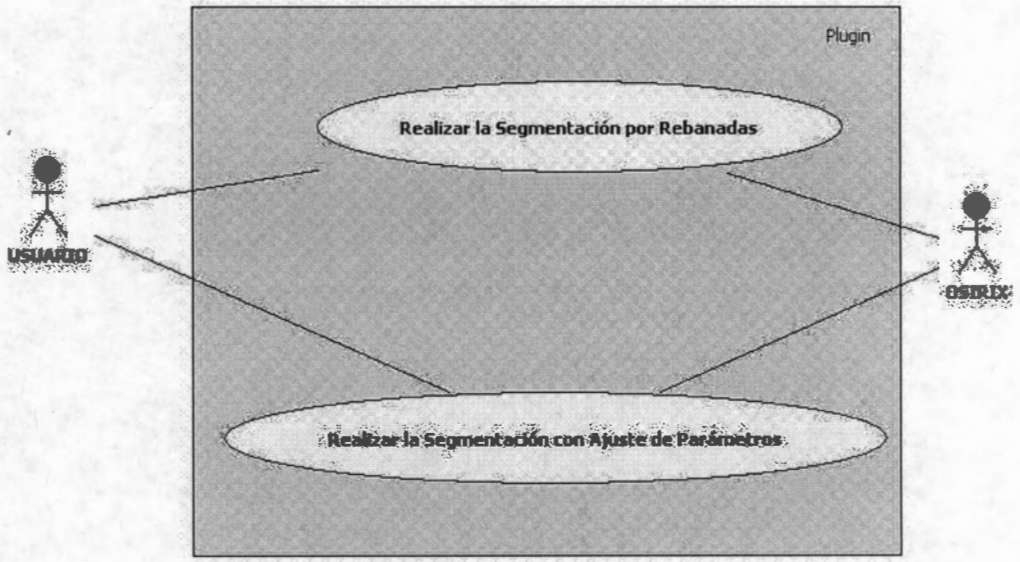


Figura 4.3: Diagrama de Casos de Uso

El primer caso de uso o el de *"Realizar la Segmentación por Rebanada"*, calcula todo el procedimiento (todas los pasos) para n rebanadas de un estudio. Este caso de uso tiene la función de hacer la segmentación de los datos que se necesitan, sin mostrar resultados parciales, como por ejemplo mapa de confianza o filtrado. Las actividades asociadas a este caso de uso se muestran en la Figura 4.4. Primero se cargan los datos o el volumen a procesar, luego se elige el plugin en el menú de OsiriX y se selecciona la opción se *Segmentación por Rebanada(s)*. En este caso de uso tanto el número de rebanadas como los parámetros se ajustan al inicio y hay que esperar el resultado total para evaluarlo. El otro caso de uso es el de *"Realizar la Segmentación con Ajuste de Parámetros"* y tiene la funcionalidad de mostrar resultados parciales para someter a evaluación los parámetros elegidos. El diagrama de actividades de este caso de uso se muestra en la Figura 4.5. Se inicia de la misma manera, se carga el volumen, se elige el plugin y se selecciona la opción *Con Ajuste de Parámetros*. Luego, se introduce la rebanada, se introducen los parámetros del mapa de confianza (umbral, δ y factor de mezcla, β(ver Capítulo 2, sección 2.1.3)), se acepta la opción mapa de confianza oprimiendo el botón que lo indica y cuando se muestra el resultado al usuario éste puede reintroducir los parámetros del mapa de confianza para que se recalcule o introducir los parámetros del filtrado por corrimiento de media (radio espacial, h_e y radio de intensidad, h_i (ver Capítulo 2, sección 2.1.4)) y elegir el botón de corrimiento de media. Cuando se muestra el resultado del filtrado por corrimiento de media, el usuario puede reintroducir los parámetros y recalcular el corrimiento de media o coloca los parámetros de fusión y

podado y aceptar esa opción. El cálculo continua hasta mostrar la imagen fusionada y podada y pueden hacerse los cambios necesarios en los parámetros hasta que el usuario este satisfecho con el resultado. El caso de uso de *"Realizar la Segmentación con Ajuste de Parámetros"* permite elegir la mejor combinación de parámetros para un estudio antes de hacer el cálculo completo del volumen. Por ello, solo permite que se aplique la segmentación sobre una rebanada seleccionada por el usuario.

Figura 4.4: Diagrama de actividad del caso de uso *Realizar la Segmentación por Rebanadas*

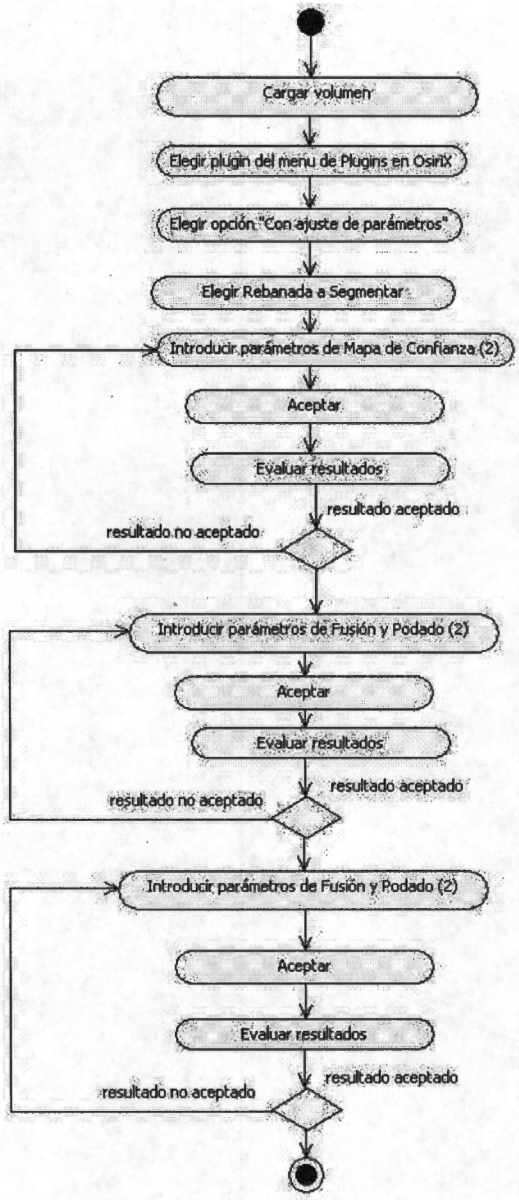


Figura 4.5: Diagrama de actividad del caso de uso *Realizar la Segmentación con Ajuste de Parámetros*

Diagrama de Clases Modela la estructura de clases y el contenido de cada una, utilizando elementos de diseño como clases, paquetes y objetos. También muestra relaciones de herencia, asociación y otras. Es un diagrama estructural que muestra un grupo de clases e interfaces, su colaboración y relación. Un diagrama de clases ilustra el diseño estático del sistema [20]. También es en este diagrama donde se involucra el vocabulario específico para la construcción de sistemas ejecutables. En la Figura 4.6 se muestra el

resultado del análisis y diseño seguido en este proceso. Las clases resultantes poseen los métodos y atributos necesarios para cumplir con la funcionalidad (casos de uso) planteadas al principio de este capítulo. La clase principal, `SegmentacionFilter`, tiene una relación de composición con clases como `Mapa de Confianza` o `Corrimiento de Media`, ésto es debido a que si `SegmentacionFilter` desaparece, todas las clases con las que tiene relación también desaparecen. La clase de `Fusión` está asociada a la clase `Mapa de Confianza` y `Corrimiento de Media`, debido a que en sus atributos tiene una instancia de estas dos clases.
Es muy importante que la clase principal sea una subclase de *PluginFilter*, una de las clases de OsiriX. También `Interfaz`, `Parametros` y `Despliegue` deben ser subclases de *PluginFilter* debido a que entre los atributos y métodos de esta clase hay información importante sobre los datos de la imagen que son utilizados para la interfaz gráfica de usuario y para la adquisición de parámetros. El resto de clases no son heredadas de ningún header de OsiriX, por lo tanto son subclase de *NSObject*, que es la clase raíz.

Diagramas de Interacción Ese es el nombre colectivo para los diagramas de colaboración y de secuencia.

Diagrama de Colaboración Muestra la interacción organizada en torno a los objetos y sus relaciones de colaboración. No incluye el tiempo, como dimensión por lo que utiliza números para mostrar la secuencia de mensajes. En los diagramas de Secuencia se hace mayor énfasis en el modelado del orden temporal de dichos mensajes.
Enfatiza la organización estructural de los objetos que envían y reciben mensajes. El diagrama de colaboración ilustra la visión dinámica del sistema. En las Figuras 4.7 y 4.8 se muestra el diagrama de colaboración para el caso de uso de *"Realizar la Segmentación por Rebanadas"* y *"Realizar la Segmentación con Ajuste de Parámetros"*, respectivamente. Se presenta la interacción de los actores (usuario y OsiriX) con el plugin. Es importante destacar que el plugin no es responsable de introducir los datos, por que es OsiriX el que maneja esta tarea. Si no hay datos no se habilita la opción de ejecutar los plugins. Cuando el usuario ya haya cargado los datos y haya elegido en el menú el plugin de *"Segmentación"*, se considera que se ha dado el mensaje de "inicio" (mensaje 1) entonces, se envía el mensaje "*filterImage*" (mensaje 2) que indica al plugin que debe comenzar a funcionar. La clase encargada de manejar el desarrollo del proceso dentro del plugin es la clase principal o `SegmentacionFilter`. OsiriX envía los datos a la clase `Volumen`, el total de rebanadas es almacenado en la clase `SegmentacionFilter`

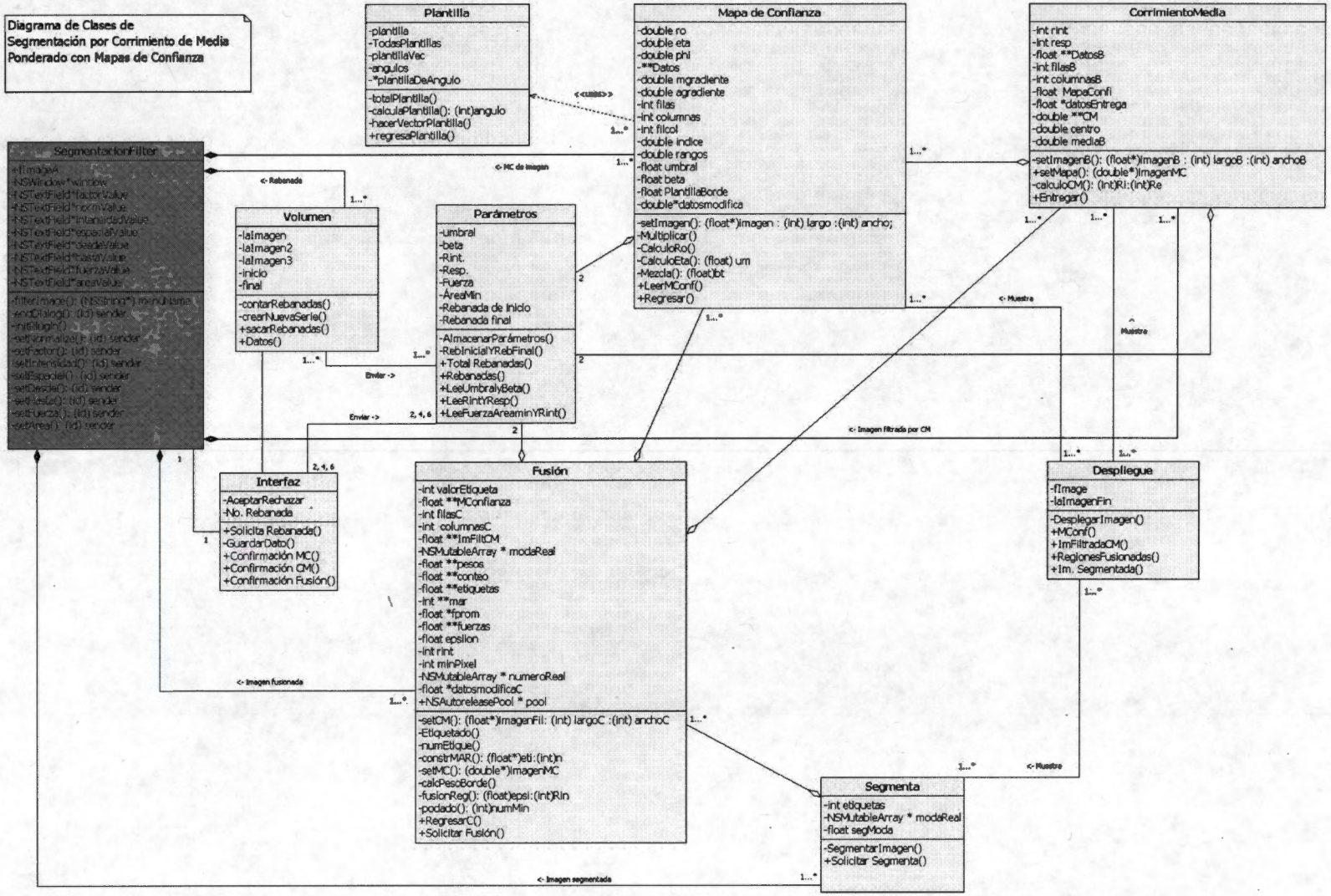


Figura 4.6: Diagrama de Clases

quien a su vez envía el dato a la clase **Interfaz**, quien solicita al usuario la rebanada inicial y la rebanada final que van a procesarse y compara que los valores que introdujo el usuario sean congruentes con la cantidad de rebanadas que tiene el volumen. La clase **Interfaz** envía a **Parámetros** los datos de inicio y final (y los otros 6 parámetros necesarios del proceso) para validar si los valores son congruentes con la cantidad de rebanadas del volumen, si todo esta bien, solicita a la clase **Volumen** que tome las rebanadas que van a procesarse. **SegmentacionFilter** solicita el cálculo del mapa de confianza. La clase de **Mapa de Confianza** solicita los datos (imagen o imágenes), los parámetros (umbral, δ y factor de mezcla, β) y la plantilla de borde ideal a la clase de **Plantilla**, para poder calcular la medida de confianza (η). Cuando ya se calculó el mapa, la clase principal solicita que se haga el filtrado por corrimiento de media. La clase **Corrimiento de Media** solicita los datos, los parámetros (radios espacial y de intensidad, h_e, h_i) y el mapa de confianza para poder hacer el filtrado. Cuando el filtrado ya se ha calculado, **SegmentacionFilter** solicita a la clase de **Fusión** que haga la fusión de regiones y el podado de la imagen resultante del filtrado. La clase **Fusión** solicita el mapa de confianza y los parámetros (fuerza de borde, ξ y área mínima μ). Cuando ya ha terminado la fusión y el podado, la clase principal solicita a la clase **Segmenta** la sustitución de la imagen de las regiones resultantes por las modas calculadas resultantes del proceso. Después de esta sustitución la clase **Segmenta** envía la información a la clase **Despliegue** para mostrar los resultados al usuario. En el caso de uso de *"Realizar la Segmentación con Ajuste de Parámetros"*, Figura 4.8, los resultados de las clases **Mapa de Confianza, Corrimiento de Media y Fusión,** envían resultados parciales para ser desplegados. La clase **Interfaz Gráfica** solicita la aceptación del resultado por parte del usuario para continuar con el siguiente paso del proceso.

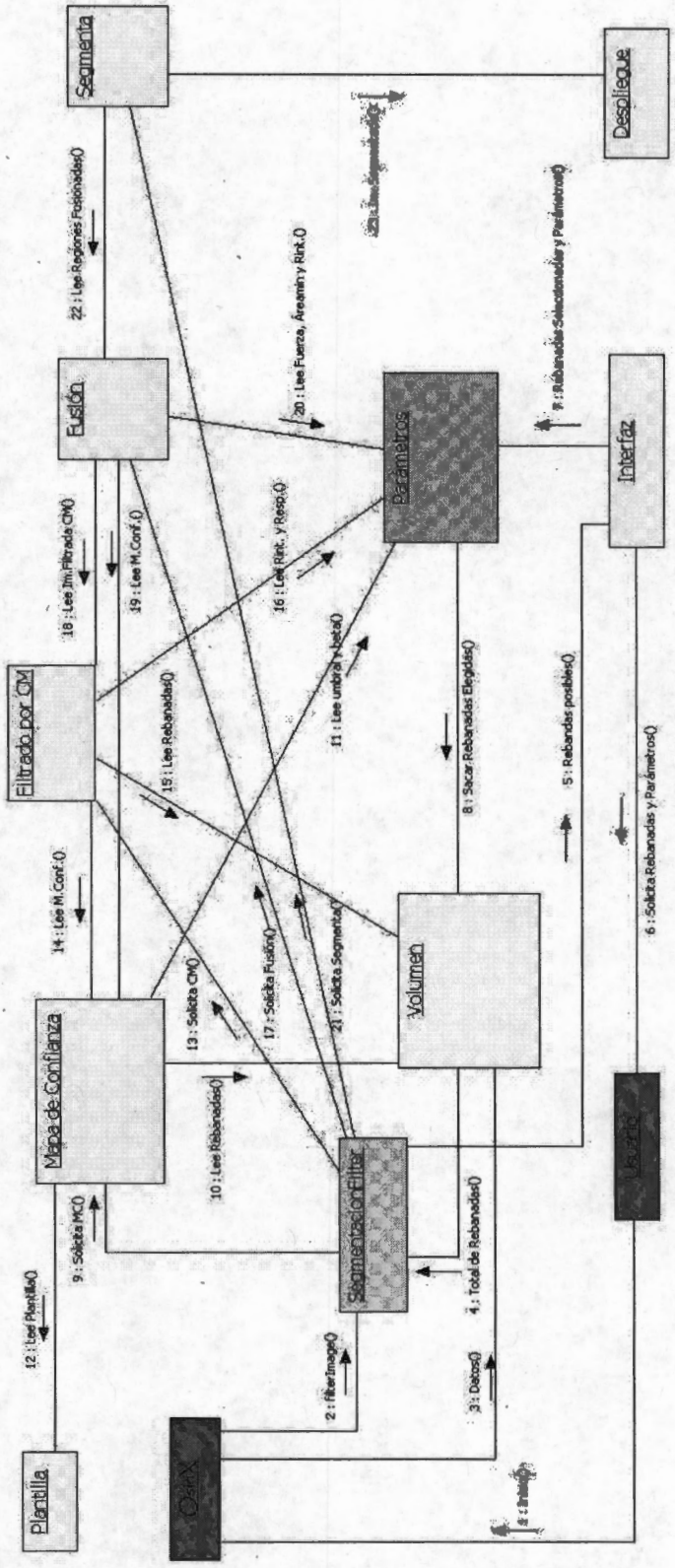


Figura 4.7: Diagrama de Colaboración de Caso de Uso de *"Realizar la Segmentación por Rebanada"*

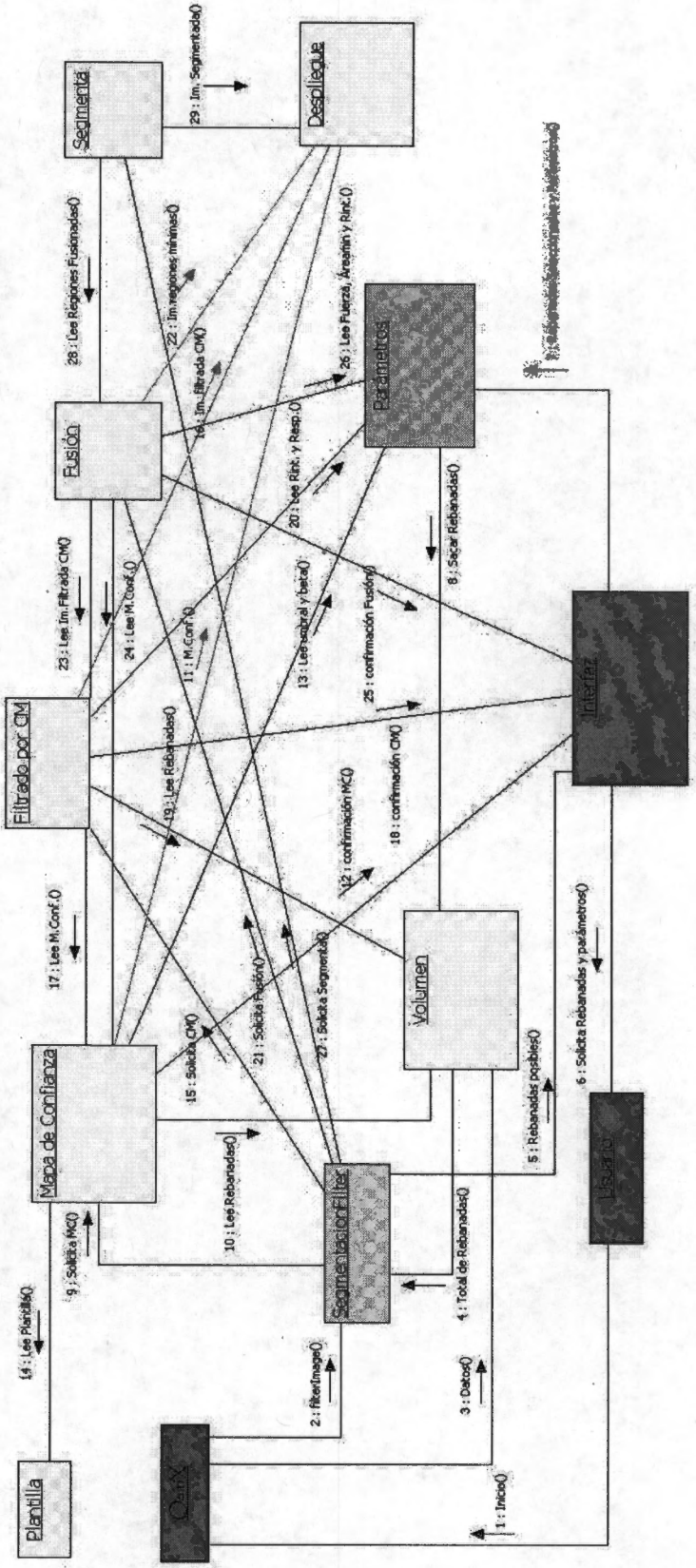


Figura 4.8: Diagrama de Colaboración Caso de Uso *"Realizar la Segmentación con Ajuste de Parámetros"*

Diagrama de Secuencia Muestra el orden temporal o la secuencia de los mensajes. Consiste en una dimensión vertical (tiempo) y una dimensión horizontal (los objetos que participan). De este diagrama obtenemos también los métodos que se requieren en la implementación del proyecto. Enfatiza el orden temporal de los mensajes. El diagrama de secuencia muestra un conjunto de objetos y de mensajes enviados y recibidos por esos objetos. El diagrama de secuencia ilustra también la visión dinámica de un sistema. En los diagramas que se muestran en las Figuras 4.9 y 4.10, puede observarse el énfasis que se hace del orden temporal de los mensajes. En estos diagramas es muy importante la línea de vida de cada objeto (línea vertical punteada) que representa la existencia del objeto a través del tiempo. En el caso del plugin, los objetos no pueden existir si OsiriX no ha recibido el mensaje de inicio, porque, como se mencionó en las características del plugin, no puede existir sin la aplicación para la cual fue hecho. Otro aspecto muy importante es el foco de control, que se representa como un rectángulo delgado, sobre la línea de vida. El foco de control muestra los periodos de tiempo que el objeto esta ejecutando una acción. De estos diagramas, se obtienen los métodos que son necesarios para el desarrollo del diagrama de clases.

En la Figura 4.9 se observan en los extremos, los dos actores (OsiriX y el usuario), porque también son parte de este diagrama. De la misma manera que en el diagrama de colaboración, el primer mensaje es el del usuario hacia OsiriX, cuando carga las imágenes y elige el plugin a utilizar. El segundo mensaje es el de OsiriX a la clase principal . Los mensajes son los mismos, pero en este diagrama se observa con claridad el fujo temporal de ellos y se determinan los métodos necesarios para crear el diagrama de clases. Como en el diagrama de colaboración, el usuario es el que envía el mensaje de inicio a OsiriX, cuando ejecuta el programa y carga las imágenes, luego, cuando se elige el plugin *Segmentación* en el menú de Plugins de OsiriX, éste envía el mensaje (*Imagefilter()*) a la clase principal que en este caso es la clase `SegmentationFilter` (porque tiene el método `imagefilter`) que va a controlar el flujo y el orden de los mensajes. OsiriX envía los datos a la clase `Volumen` (*Datos()*) y ésta a su vez envía el total de rebanadas a `SegmentacionFilter`. La clase `Interfaz` recibe la información del número de rebanadas que tiene el volumen y solicita al usuario los parámetros y la rebanada inicial y final a procesar, compara los datos de inicio y final con el total de rebanadas del volumen y luego, envía estos datos a `Parámetros`, quien a su vez los envía a `Volumen`. La clase `SegmentacionFilter` solicita que se haga el cálculo del mapa de confianza, y esta clase lee las rebanadas y solicita los parámetros. `Mapa de Confianza` lee la plantilla correspondiente. El flujo de mensajes es el mismo que en el diagrama de colaboración, pero en el diagrama de secuencia hay mayor énfasis en el orden temporal.

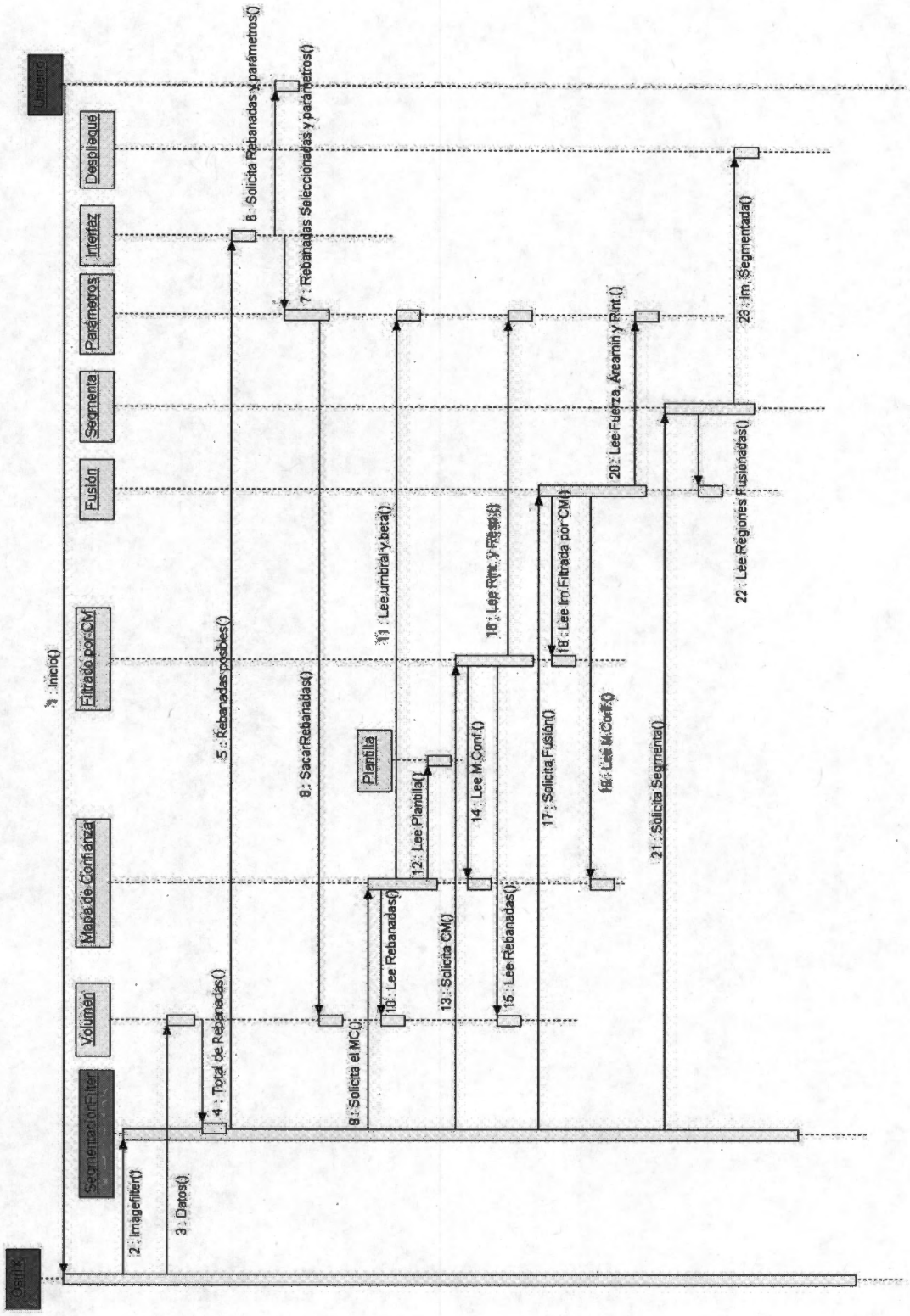


Figura 4.9: Diagrama de Secuencia de Caso de Uso de *"Realizar la Segmentación por Rebanada"*

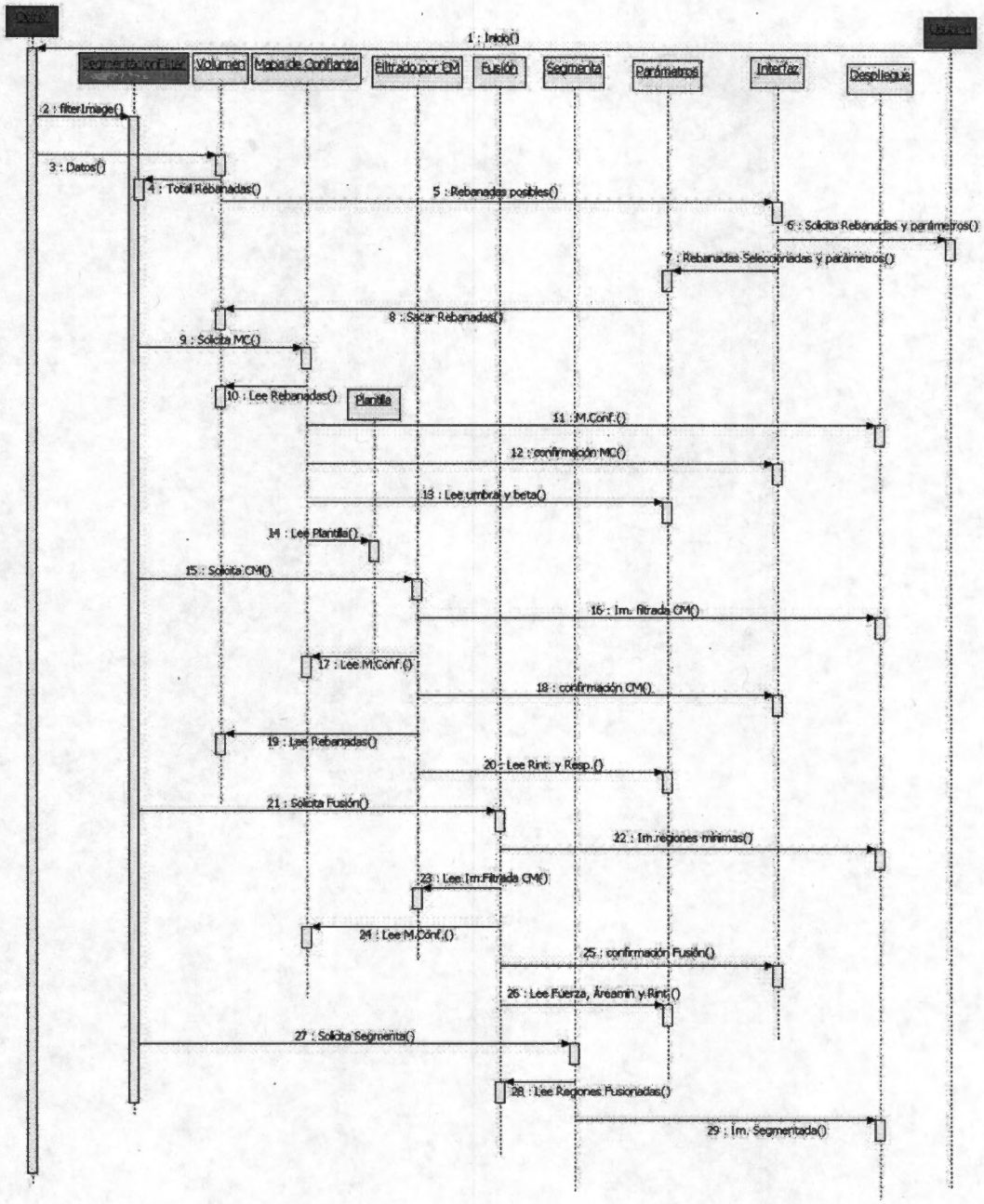


Figura 4.10: Diagrama de Secuencia de Caso de Uso de *"Realizar la Segmentación con Ajuste de Parámetros"*

En la construcción de software es necesario hacer un modelo para poder comprender y visualizar el sistema que se está desarrollando. Por otro lado, OsiriX utiliza POO y aunque no hay un modelo formal de éste, es recomendable utilizar enfoque orientado a objetos para modelar los plugins o módulos de extensión para esta aplicación. Esta sugerencia se debe a que OsiriX utiliza 7 clases como interface con el plugin y si se

hace un modelo orientado a objetos se pueden aprovechar los métodos y los atributos que OsiriX ofrece en esas 7 clases para el manejo y procesamiento de las imágenes. El Proceso Unificado es el método utilizado para el desarrollo de software ya que es iterativo incremental, guiado por casos de uso y basado en arquitectura y ofrece grandes ventajas para el desarrollo de este proyecto. El modelado se hizo utilizando los siguientes diagramas de UML: diagrama de casos de uso, de actividades, de clases, de colaboración y de secuencia. Una vez modelado el proyecto esta listo para ser implementado como clases dentro de un nuevo plugin para OsiriX.

Capítulo 5

Resultados y Discusión

Con el diagrama de Clases se construyó el plugin *"Segmentación"* (ver Anexo C). Después de instalar y probar el funcionamiento del plugin dentro del ambiente de OsiriX, se hizo necesario verificar la funcionalidad de la metodología utilizando dos tipos de datos: sintéticos y reales. Los datos sintéticos fueron de dos tipos, uno, es el obtenido de un fantoma de dos esferas concéntricas, son 5 estudios o series[24] del mismo fantoma pero con diferentes niveles de ruido, que simulan de manera muy burda al cerebro; el otro tipo de dato sintético es un fantoma digital de cerebro. Los datos reales fueron obtenidos del Internet Brain Segmentation Repository (IBSR) [24] y corresponden a un estudio de cerebro normal, obtenido del Center for Morphometric Analysis del Massachusetts General Hospital y están disponibles en el nodo del IBSR.

5.1. Interfaz Gráfica

La clase `Interfaz` entrega una ventana con botones y varios campos para que el usuario introduzca los parámetros necesarios para hacer funcionar las diferentes clases que componen el plugin de *"Segmentación"*. Como se muestra en la Figura 5.1 los primeros dos parámetros definen la rebanada inicial y la final que van a ser procesadas de todo el volumen. Los parámetros que corresponden a la clase de `Mapa de Confianza` son el umbral de borde, δ y el factor de mezcla, β. Los del `Corrimiento de Media` son tanto el radio espacial, h_e como el radio de intensidad, h_i. Por último, los parámetros correspondientes a la clase de `Fusión` y Podado siendo éstos fuerza de borde, ξ y área mínima, μ. Todos ellos presentan un valor predeterminado por omisión, que es el sugerido por [3], sin embargo, el usuario tiene posibilidades de cambiarlos para ajustarlos a los datos que se procesan.

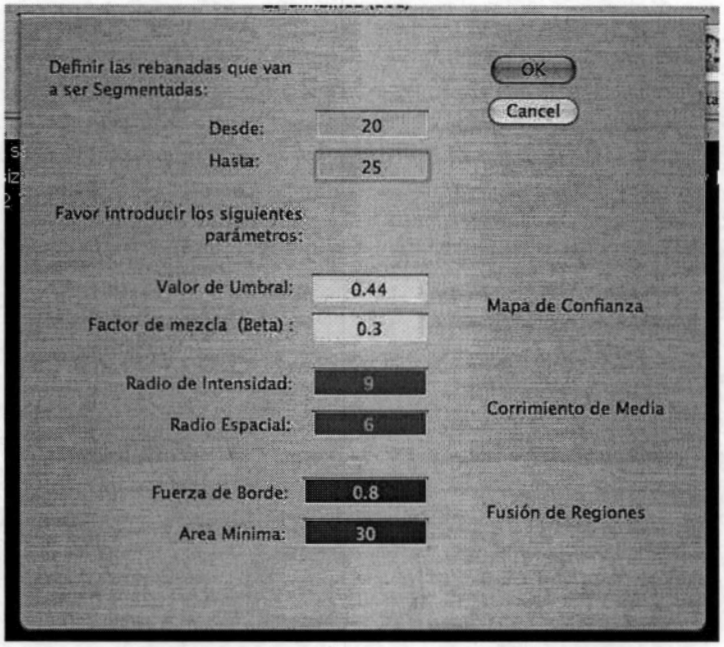


Figura 5.1: Interfaz Gráfica de *"Segmentación por Rebanada(s)"*

En la Figura 5.2 se presenta la interfaz de *"Ajuste de Parámetros"*, en la cual se permiten hacer varias pruebas de segmentación en una sola rebanada para ajustar los parámetros que sean más convenientes para la imagen.

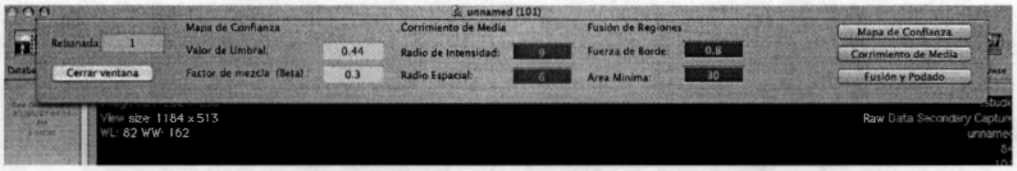


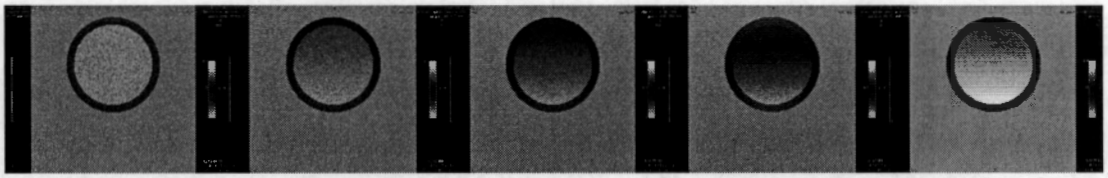
Figura 5.2: Interfaz Gráfica de *"Con Ajuste de Parámetros"*

5.2. Pruebas en Datos Sintéticos

5.2.1. Fantoma Esférico

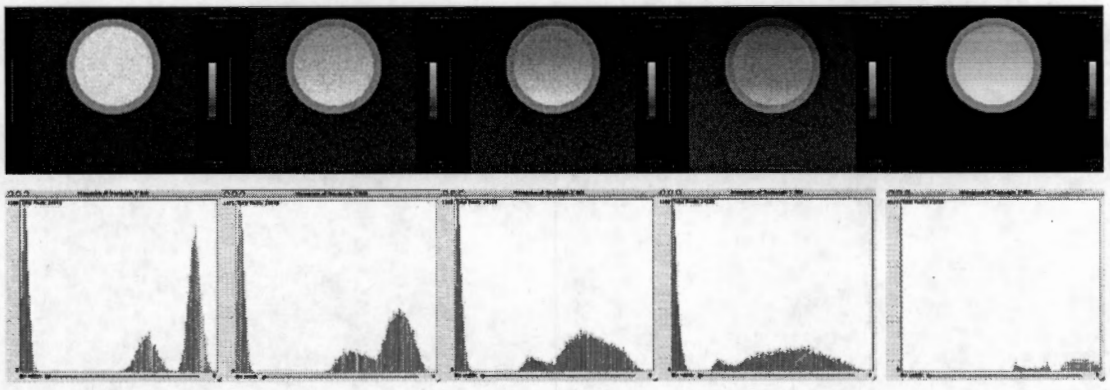
Los datos sintéticos del fantoma esférico son imágenes sencillas de dos esferas concéntricas utilizadas para probar o validar el plugin en forma cualitativa. En el fantoma se representan la materia gris (esfera exterior) y la materia blanca (esfera interior). Cada uno de los 5 estudios están contaminados con diferentes niveles de ruido y gradientes de intensidad. Los volúmenes están formados por 52 rebanadas de 256×256, pero para las pruebas se utilizó una rebanada central (rebanada 25). En la Figura 5.3 se muestra la

rebanada central para los 5 estudios. Este fantoma nos interesa porque podemos evaluar la capacidad que tiene el procedimiento de segmentación por corrimiento de media de separar estructuras a pesar de que la información esté "mezclada o entrelazada", es decir, el grado de dificultad para separar clases sea mayor. En la Figura 5.4, se muestan los 5 estudios y sus histogramas, donde se puede apreciar las inhomogeneidades en las imágenes y se puede apreciar, de manera cualitativa, la dificultad que cada una presenta para poder separar la información. En este sentido, el estudio del fantoma esférico 4 (Figura 5.4 (d)) presenta mayor dificultad para la segmentación o separación de clases, porque éstas estan mezcladas, como puede apreciarse en el histograma, la información no se distinguen de manera evidente; y esto se debe a que es el estudio con mayor gradiente y ruido.



(a) (b) (c) (d) (e)

Figura 5.3: Se presentan las imágenes en color falso para apreciar las diferencias (a) Fantoma esférico 1, con ruido y sin gradiente, (b) Fantoma esférico 2, con ruido y gradiente pequeño, (c) Fantoma esférico 3, con ruido y gradiente mayor al anterior, (d) Fantoma esférico 4, con ruido y gradiente aún mayor que el fantoma 3, (e) Fantoma esférico 5, sin ruido y con gradiente igual a la esfera 2 [24].



(a) (b) (c) (d) (e)

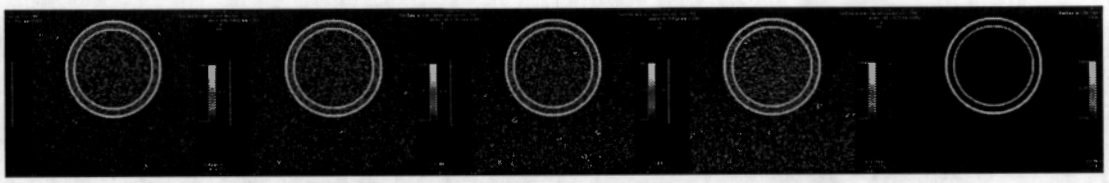
Figura 5.4: Estudios del fantoma esférico que presentan diferentes inhomogeneidades (superior) , histogramas de los estudios (inferior). Con los histogramas se puede apreciar la complejidad que presenta cada estudio para distinguir y separar las clases.

Para las imágenes del fantoma esférico se utilizaron los siguientes parámetros de cálculo:

Tabla 5.1: Valores de los parámetros para el proceso de segmentación en datos sintéticos

Parámetro	Símbolo	Fantoma Esférico
Factor de mezcla	β	0.3
Umbral de borde	δ	0.44
Fuerza de borde	ξ	0.8
Radio espacial	$\boldsymbol{h_e}$	6
Radio de intensidad	$\boldsymbol{h_i}$	25
Área mínima de región	μ	30

El primer paso del proceso de segmentación es el cálculo del mapa de confianza, teniendo por objeto calcular un valor para la presencia de un borde en los datos a segmentar. La detección de los bordes de la imagen se realiza estimando la magnitud y orientación del gradiente en una ventana de datos de 5×5 pixeles. Con la magnitud del gradiente se determina la función de distribución acumulativa empírica, obteniéndose los rangos normalizados; y con la estimación de la orientación del gradiente se elige la plantilla del borde ideal para cada dato. Si el valor del rango normalizado es mayor que el umbral del borde ($\rho_k > \delta$) para un dato dado, se calcula el valor de confianza, de otra manera no se considera parte de un borde y su valor de confianza es cero. Los resultados para la clase `Mapa de Confianza` aplicada sobre los 5 estudios del fantoma esférico (Figura 5.4), se muestra en la Figura 5.5. Los resultados para las esferas con ruido presentan, incluso, la definición de los bordes de dicho ruido; por lo que, cualitativamente, el resultado mostrado en la Figura 5.5 (e) (estudio 5) que corresponde al estudio con niveles de inhomogeneidad solamente, define mucho mejor los bordes y puede considerarse un mejor resultado, lo cual coincide con lo presentado por [3].



(a) (b) (c) (d) (e)

Figura 5.5: `Mapa de Confianza` aplicado sobre los 5 estudios del fantoma esférico. Se utilizaron los parámetros de la Tabla 5.1 que son los sugeridos en [3].

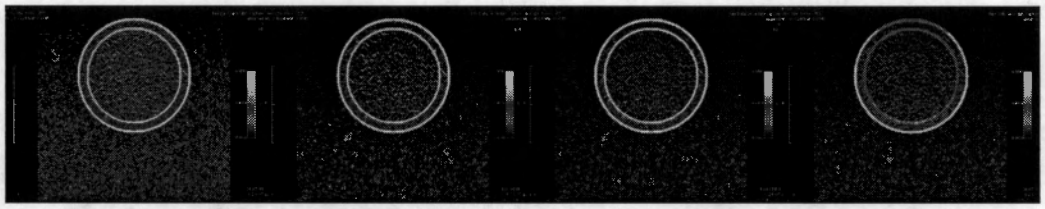
Los parámetros que requiere la clase `Mapa de Confianza` son:

- Umbral de borde, δ , es un valor de referencia para la distribución acumulativa de la magnitud del gradiente, ρ, que se utiliza para elegir que datos de la imagen participan en el cálculo de la medida de confianza, η. Esto se debe a que si el

valor de ρ es muy pequeño, no es necesario calcular la medida de confianza de ese pixel, porque no puede ser un borde (ver Capítulo 2, sección 2.1.3).

- Factor de mezcla, β , para determinar el peso (porcentaje) de ρ y de η que se incluyen en el cálculo del mapa de confianza.

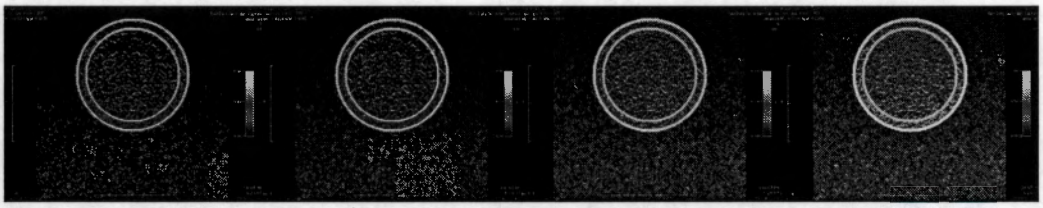
Se utilizó el caso con mayores niveles de ruido y de inhomogeneidad, es decir el estudio 4 del fantoma esférico (Figura 5.3 (d)), para mostrar cómo afecta el cambio del parámetro de umbral de borde en el cálculo del mapa de confianza y se observa en la Figura 5.6 que, mientras menor sea el umbral, mayor parte de los pixeles participan en el cálculo de η e incluso mayor parte del ruido (cuyo valor de ρ es pequeño) forma parte del mapa. Esto se aprecia al comparar, en la Figura 5.6 de la (a) a la (d). Cuando el umbral es mayor, hay parte de información de los bordes de la imagen que se pierde, debido a que para el cálculo de η solo se consideran los pixeles de la imagen con valores muy altos de ρ.



(a) (b) (c) (d)

Figura 5.6: Mapa de Confianza para esfera 4, con β =0.3 y δ variable, (a) $\delta = 0.1$; (b) δ =0.44; (c) δ =0.85; (d) δ =0.95

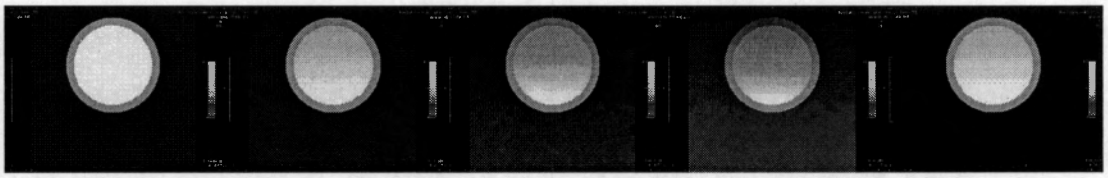
En la Figura 5.7 se observa el mapa de confianza del estudio 4 del fantoma esférico (Figura 5.3 (d)) con diversos factores de mezcla. Si β es cercano a uno se da mayor peso a ρ y si es cercano a cero lo que se observa es η. Observamos que el factor de mezcla es útil para eliminar ruido sin perder información. El simple cálculo de los bordes con la magnitud del gradiente se mejora con la incoroporación de la medida de confianza.



(a) (b) (c) (d)

Figura 5.7: Mapa de confianza del estudio 4 del fantoma esférico, con δ =0.44 y β variable, (a) β =0; (b) $\beta = 0.3$; (c) β =0.65; (d) $\beta = 1$

Luego de la determinación del mapa de confianza continúa el cálculo del corrimiento de media, en el que se utilizan los datos del mapa de confianza por que el filtrado se hace conservando los bordes. El filtrado se realiza aplicando la clase de `Corrimiento de Media` (CM), a datos tridimensionales: dos dimensiones espaciales y una de rango o intensidad. Como ya se mencionó, el CM define una trayectoria de los datos hacia sus modas de la función de densidad, corriendo cada dato, a lo largo de la trayectoria, con un paso de tamaño equivalente a su media muestral en cada iteración. Para calcular la media muestral se determina el vecindario del dato a analizar, considerando como parte del vecindario a todos aquellos pixeles que estén a una distancia Euclidiana menor o igual al radio espacial (h_e); en seguida, para cada dato del vecindario, se evalúa su diferencia en intensidad con el dato que se está analizando, y si la diferencia está dentro del radio de resolución en intensidad (h_i), entonces el vecino se toma en cuenta para el cálculo de la media. Para preservar los bordes en el filtrado de la imagen, se utiliza el mapa de confianza como una ponderación en el cálculo de la media muestral, esto es, para que el dato tenga un peso relevante en la determinación de la media, no debe formar parte de un borde, como queda establecido al usar la ecuación 2.19 en la determinación del vector de corrimiento de media. Los resultados el filtrado por corrimiento de media son los presentados en la Figura 5.8 y pueden apreciarse las regiones o "clusters" resultantes del filtrado. Para el cálculo sobre los 5 estudios del fantoma, se utilizaron los valores de h_i y h_e sugeridos por [3], que se encuentran en la Tabla 5.1.



(a) (b) (c) (d) (e)

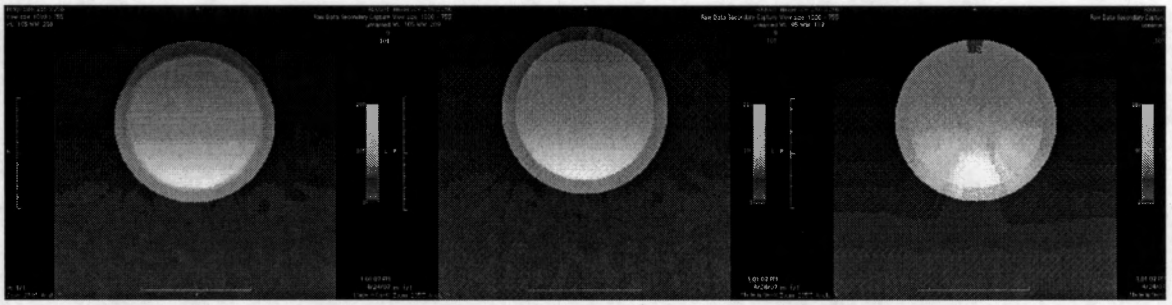
Figura 5.8: Filtrado por Corrimiento de Media sobre los 5 estudios del fantoma esférico con color falso para apreciar las regiones resultantes en el cálculo del CM. Se utilizaron los parámetros para CM de la Tabla 5.1 que son los sugeridos en [3].

Como se mencionó anteriormente, los parámetros que requiere la clase de filtrado por `Corrimiento de Media` son:

- Radio espacial, h_e, que determina el vecindario del dato a analizar (ver Capítulo 2, sección 2.1.4).
- Radio de intensidad, h_i, que determina la diferencia máxima de intensidad entre

cada dato del vecindario y el dato a analizar, para ser tomado en cuanta en el cálculo de la media (ver Capítulo 2, sección 2.1.4).

La Figura 5.9 muestra el efecto de una combinación de parámetros h_e y h_i en el filtrado por corrimiento de media diferente a la sugerida en la Tabla 5.1 y cómo afecta en el resultado. Lo que puede observarse en 5.9 (a) es que si el radio de intensidad h_i, es demasiado grande, se agrupan datos que no deberían (debido al tipo de fantoma esférico) ser parte de la misma clase, se pierde información. En 5.9 (c) , es decir, un radio espacial, h_e, más grande que el de la Tabla 5.1, se observa que las regiones son más grandes (el vecindario es mayor, cuando el h_e es mayor) y homogéneas, es decir, se pierde alguna información importante para la futura fusión de regiones. Otra consecuencia de aumentar el h_e , es que el tiempo de procesamiento aumenta.

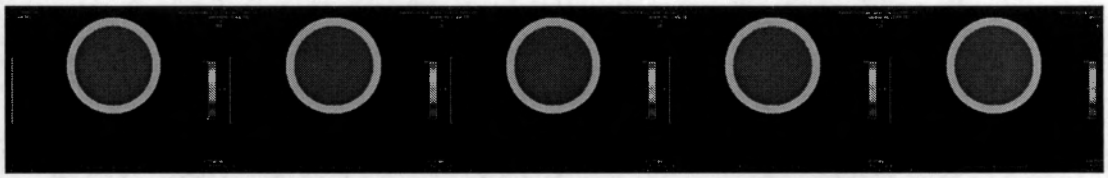


(a) (b) (c)

Figura 5.9: Fantoma esférico 4 filtrado por corrimiento de media con diferentes valores de h_e y h_i; (a) h_e= 6 y h_i= 80; (b) h_e = 6 y h_i= 25, que son los valores sugeridos en la Tabla 5.1 para el conjunto de imágenes ; (c) h_e = 25 y h_i= 25

El tercer paso en el proceso de segmentación tiene que ver con el análisis de adyacencia de regiones, para fusionar y disminuir el número de regiones encontradas en la imagen filtrada. La etapa de filtrado por CM produce una gran cantidad de regiones. Las regiones homogéneas en intensidad están formadas por todos los pixeles que convergieron a la misma moda y para realizar el análisis de adyacencia, lo primero que se debe hacer es etiquetar cada región con un valor escalar, que corresponde a la etiqueta que identifica la región. Las imágenes etiquetadas son obtenidas a partir de las imágenes producidas en el paso de filtrado por corrimiento de media. Para realizar el análisis de adyacencias, se crea un grafo, como se mencionó en la Sección 2.1.3, quedando representado por su matriz de adyacencias de regiones (MAR). De la imagen etiquetada se determinan las adyacencias entre regiones, que es la información que permite establecer las regiones que son adyacentes entre sí y se almacena en la MAR como una estructura de datos.

Con esta matriz, se obtiene el vecindario de una región para intentar fusionar la región con alguna otra que tengan un borde en común.
El criterio que determina si se fusionan o no dos regiones es que la diferencia en intensidad, normalizada con respecto al ancho de banda en intensidad, h_i, sea menor a 0.5 en el caso de segmentación de IRM cerebrales y que la fuerza del borde entre ellas, tomada del mapa de confianza, sea menor al umbral de fuerza de borde (ξ). La condición de intensidad busca regiones que sean similares, y la condición de fuerza busca las regiones con bordes débiles entre ellas, así, si dos regiones son similares en intensidad y comparten un borde débil entonces se fusionan. Una vez realizado el análisis de adyacencias para todas las regiones, se reetiqueta la imagen y se calcula la moda de las regiones nuevas, con el promedio de las modas de las regiones que se fusionaron. Las operaciones de fusión, reducción del grafo, y actualización de la imagen resultante, se repiten iterativamente hasta que ya no es posible fusionar más regiones.
Después de fusionar las regiones, el podado tiene como objetivo remover las regiones que tienen una superficie (expresado como un número de pixeles) menor a un umbral preestablecido (μ). La poda toma las imágenes etiquetadas después de aplicar la fusión, como los datos de entrada para construir la MAR, y empieza a recorrerla buscando las regiones menores al umbral. Cuando encuentra una región pequeña, con la información de la MAR determina las regiones vecinas, y establece la región candidata con la cual fusionarla. La región candidata debe satisfacer las siguientes propiedades: es la más parecida en intensidad, o es la única con superficie mayor al umbral. En la clase Segmenta, se sustituyen los pixeles de la imagen podada por el valor de la moda calculada en el proceso. Los resultados de las clases de `Fusión`, `Podado` y `Segmenta` para los 5 estudios del fantoma esférico se muestran en la Figura 5.10. En los 5 estudios del fantoma (Figura 5.4) el número de regiones finales es de 3, lo cual es lo correcto, dado que el fantoma utilizado para adquirir estos estudios consta de dos esferas concéntricas y el fondo.



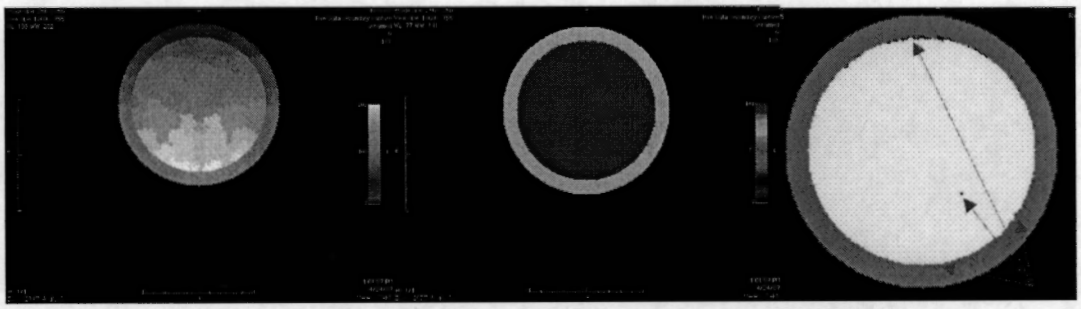
(a) (b) (c) (d) (e)

Figura 5.10: Clases `Fusión`, `Podado` y `Segmenta`, aplicadas sobre los 5 estudios del fantoma esférico 5.4 utilizando los parámetros que están en la Tabla 5.1. Se observa que para los 5 estudios el resultado final es de 3 regiones finales, fondo, esfera exterior y esfera interior, lo cual es lo correcto, debido al fantoma utilizado.

Como se mencionó anteriormente, los parámetros necesarios para la clase de Fusión son:

- Radio de intensidad, h_i (ver Capítulo 2, sección 2.1.4) que sirve como criterio para normalizar la diferencia de intensidad entre dos regiones candidatas a fusionarse.
- Umbral de fuerza de borde, ξ (ver Capítulo 2, sección 2.1.5), que determina el peso máximo del borde entre regiones, para hacer la fusión. Es decir, si el borde es débil (menor a ξ) se fusionan las regiones, si el borde es mayor a ξ, entonces se preservan las regiones.
- Área mínima, μ (ver Capítulo 2, sección 2.1.5), que establece el número mínimo de pixeles a considerarse como región dentro del análisis de podado.

En la Figura 5.11 se muestran los resultados de diferentes elecciones de umbral de fuerza de borde y de área mínima. En la Figura 5.11 (a) con un umbral muy bajo, respecto al sugerido por [3] y que se muestra en la Tabla 5.1, por lo tanto la fusión se dará entre menor número de regiones y al finalizar tendremos mayor número de regiones que el esperado para este estudio del fantoma. En la Figura 5.11 (c) se muestra que el parámetro de área mínima utilizado en ese cálculo de podado permite que queden sin fusionar regiones muy pequeñas.



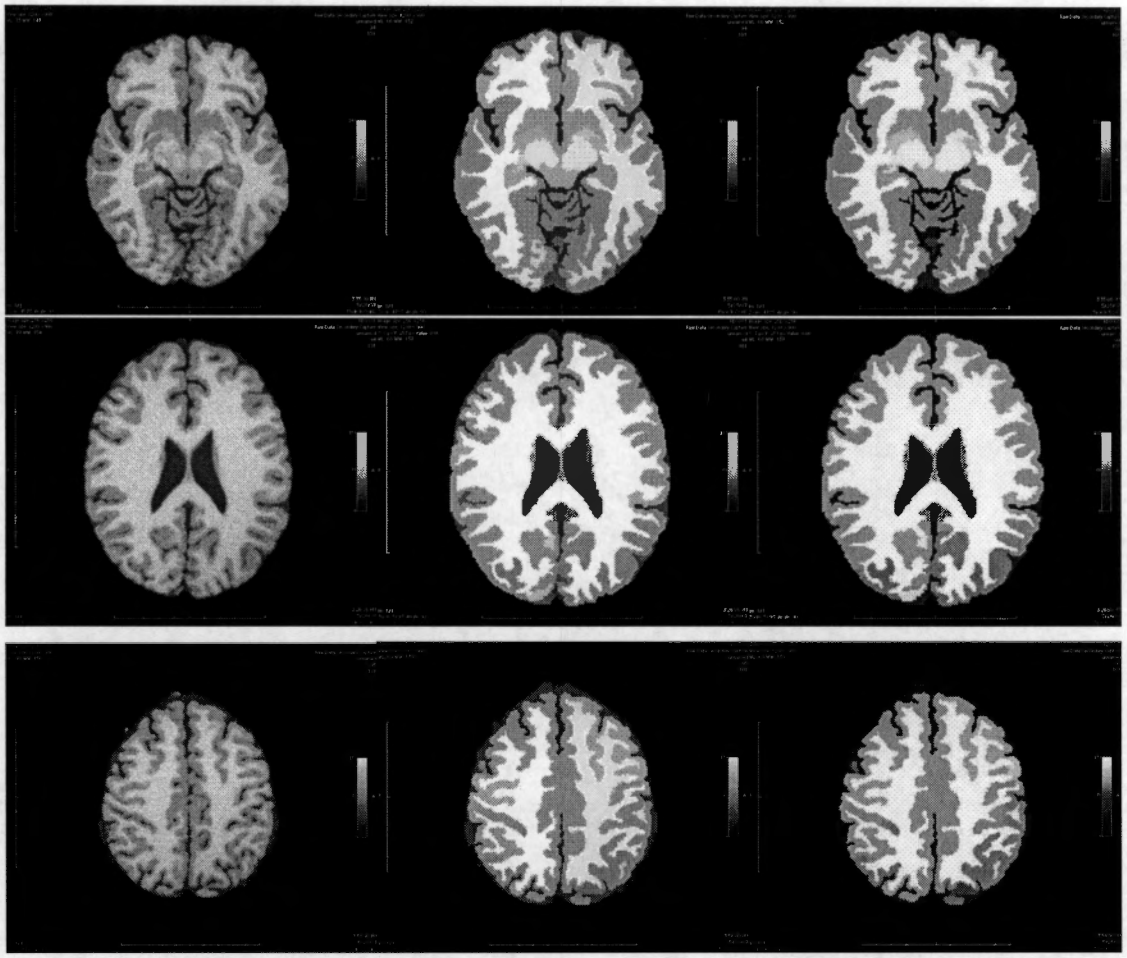
(a) (b) (c)

Figura 5.11: Fusión y podado sobre el estudio 4 del fantoma esférico; (a) con $\xi = 0.2$, que es un umbral muy bajo, por lo tanto la fusión se dará entre menor número de regiones y al finalizar tendremos mayor número de regiones que el esperado para este fantoma y $\mu = 30$ pixeles, ; (b) $\xi = 0.8$ y $\mu = 30$ pixeles, que son los valores sugeridos para este tipo de datos; (c) $\xi = 0.8$ y $\mu = 3$ pixeles, por lo que quedan algunas regiones muy pequeñas sin fusionar.

5.2.2. Fantoma Digital de Cerebro

El volumen de cerebro sintético se obtuvo del McConnell Brain Imaging Centre (BIC) [25], y corresponde a un fantoma digital. Este fantoma consiste en imágenes ponderadas

T1, el tamaño del volumen es de 181x217x181 con un espesor de rebanada de 1mm, con ruido blanco de 3 % relativo al tejido más brillante y una inhomogeneidad de intensidad del 20 %. Se utilizaron los parámetros de los datos reales (Tabla 5.2). Los resultados se muestran en la Figura 5.12. El número final de regiones para cada rebanada se muestra en la Tabla 5.5.



(a) (b) (c)

Figura 5.12: Datos de cerebro simulados: (a) imagen original; (b) imágen final con regiones homogéneas; (c) imagen final con regiones homogéneas en color falso

5.3. Pruebas en Datos Reales

5.3.1. Datos Reales de Cerebro

Como datos reales se procesó un estudio de RM de un sujeto sano, proporcionado por el Hospital General de Massachussets y disponible en el sitio IBSR (Internet Brain

Segmentation Repository) de Internet [24]. El estudio es el número 1_24 y consta de imágenes T1 en corte coronal de dimensión 256 x 63 x 256 y con voxel de tamaño 1 x 3 x 1. Los parámetros que se utilizaron para el cálculo de la segmentación en datos reales son los mismos que para datos sintéticos a excepción del radio de intensidad $\boldsymbol{h_i}$, y son los sugeridos por [3] los cuales son los siguientes:

Tabla 5.2: Valores de parámetros para datos reales

Parámetro	Símbolo	Datos de cerebro sintéticos y reales
Factor de mezcla	β	0.3
Umbral de borde	δ	0.44
Fuerza de borde	ξ	0.8
Radio espacial	$\boldsymbol{h_e}$	6
Radio de intensidad	$\boldsymbol{h_i}$	9
Área mínima de región	μ	30

En la Figura 5.13 se muestran las rebanadas originales (a), el mapa de confianza (b) , filtrado por corrimiento de media (c) y las regiones resultantes después de la fusión y podado (d), de cada una.

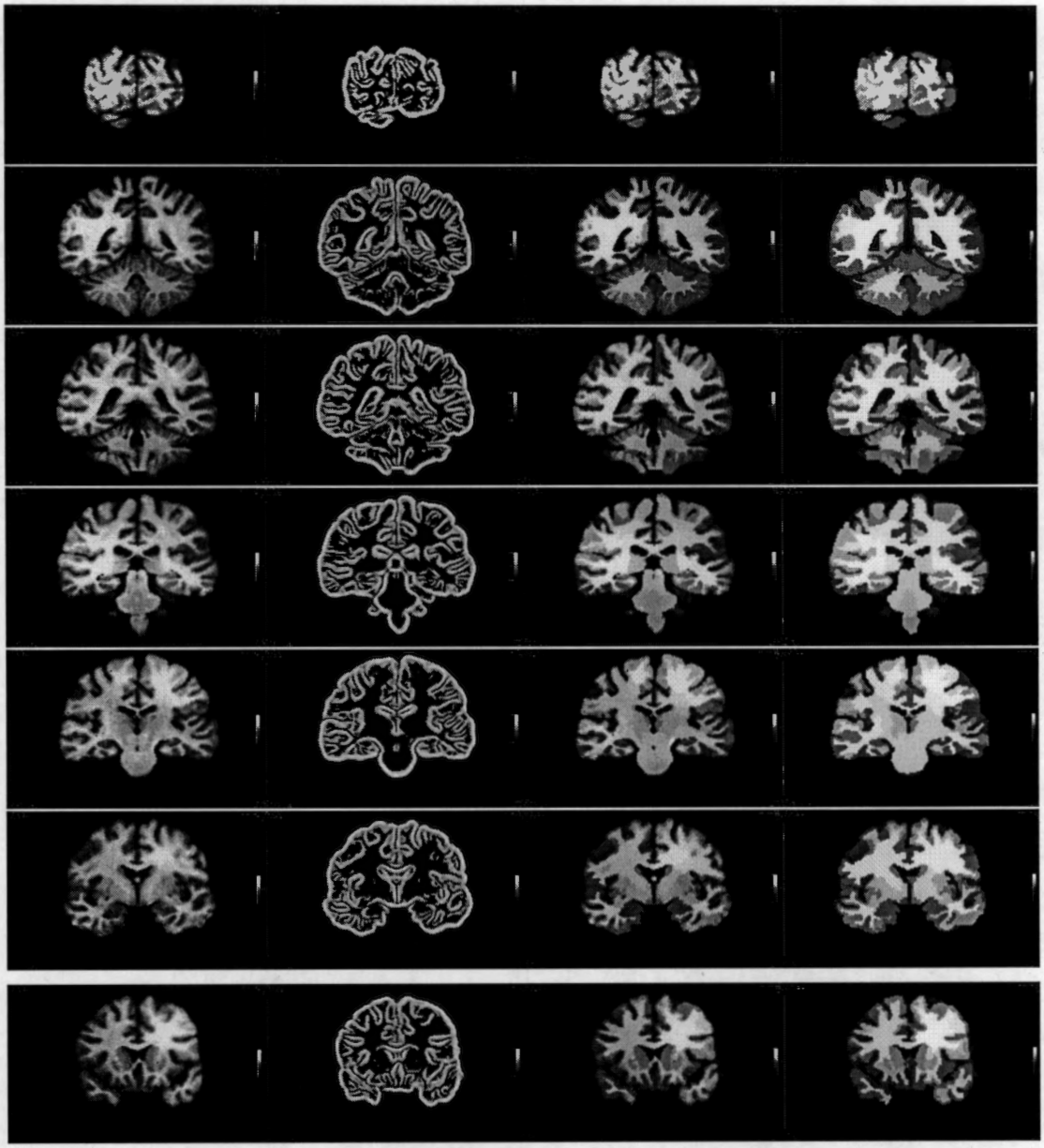


Figura 5.13: Datos reales, (a) Imagen original; (b) Mapa de Confianza; (c) Filtrado por Corrimiento de Media; (d) Regiones resultantes después de Fusión y Podado

5.4. Discusión y Conclusiones

El transportar el algoritmo de segmentación por corrimiento de media (probado en el ambiente de Matlab), realizado en nuestros laboratorios hacia un ambiente de software libre como es OsiriX, tiene como intención hacer más fácil su utilización para el usuario

final, pero también busca plantear un mecanismo para acercar las necesidades clínicas y los resultados de las investigaciones y así proveer de mayor número de herramientas al gremio médico para resolver los problemas de diagnóstico.

Al hacer una comparación entre los resultados obtenidos por [3] y los obtenidos por el plugin de OsiriX, se espera obtener resultados iguales, lo que se observa en la Tabla 5.3 (resultados de OsiriX) y en la Tabla 5.4(resultados de Matlab) en base al número de regiones resultantes después del corrimiento de media, la fusión y el podado, es que los resultados finales son los mismos para los fantomas esféricos, pero los procesos intermedios no son iguales, esto se debe a la diferencia en el manejo de datos entre Objective-C y Matlab. Como puede apreciarse en las Tablas 5.3 y 5.4 en los modelos esféricos hay diferencia en las regiones de convergencia del corrimiento de media (columna *Etiquetado*), por este motivo, las regiones después de fusión (columna *Fusión*) no son las mismas; sin embargo, después del podado (columna *Podado*), las regiones finales en todos los casos fueron 3. Este dato es el esperado, dado el tipo de fantoma utilizado, que solamente simula materia gris (circunferencia exterior y de mayor intensidad) materia blanca (circunferencia interior y de menos intensidad) y fondo. Por lo tanto, a pesar de las diferencias iniciales, las clases de OsiriX y el algoritmo en Matlab coinciden en el resultado.

Tabla 5.3: Número de regiones presentes en el análisis de adyacencias para los datos sintéticos procesados con OsiriX

Esfera	Etiquetado	Fusión	Podado
1	586	9	3
2	605	8	3
3	766	14	3
4	2104	67	3
5	89	3	3

Tabla 5.4: Número de regiones presentes en el análisis de adyacencias para los datos sintéticos procesados con Matlab

Esfera	Etiquetado	Fusión	Podado
1	581	9	3
2	602	8	3
3	768	15	3
4	2098	59	3
5	89	3	3

En el procesamiento de datos sintéticos de cerebro (Tabla 5.5), al comparar los resultados obtenidos con Matlab [3] y los datos obtenidos con OsiriX, en base al número

de etiquetas obtenidas después del filtrado por corrimiento de media (columna *Etiquetado*), las regiones después de la fusión (columna *Fusión*) y las regiones después del podado (regiones finales, columna *Podado*), se aprecia una diferencia aproximadamente igual a 1.2% en el número de regiones resultantes del corrimiento de media (columna *Etiquetado*). Las regiones después de la fusión y el podado se hacen muy diferentes a las resultantes con Matlab, estas diferencias pueden deberse al manejo de los datos de la imagen que tiene OsiriX (float) y Matlab (double), ya que el corrimiento de media implica sumatorias y promedios por lo que las diferencias de valor en el orden de las centésimas o décimas de millar se acumulan y finalmente hacen coincidir a los datos, en diferentes modas. Otro motivo es la forma en la que OsiriX lee los datos, porque éstos estan en 12 bits y OsiriX lee 8 y 16 bits, por lo que, para poderlos leer, hubo que convertirlos utilizando Matlab. Las diferencias en las regiones resultantes son altas, sin embargo, para la separación en materia gris, materia blanca y líquido cefalorraquídeo hay un paso posterior a la segmentación y éste es, la clasificación de las regiones resultantes de la segmentación (Podado) a través de un atlas probabilístico, por lo que, éste resultado no es definitivo.

Tabla 5.5: Número de regiones presentes en el análisis de adyacencias para datos de cerebro sintéticos, utilizando OsiriX y Matlab

	Rebanada	Etiquetado	Fusión	Podado
OsiriX	63	2221	1126	112
Matlab	63	2248	1115	65
OsiriX	96	1737	1100	87
Matlab	96	1717	1085	53
OsiriX	118	1666	1017	91
Matlab	118	1667	1011	56

Como se planteó al principio, el objetivo general de este proyecto que es explorar y evaluar una herramienta que permita transportar desarrollos en el área de procesamiento de imágenes médicas (metodologías de procesamiento o sistemas) hechos en centros de investigación al entorno clínico, para que sea utilizado como apoyo en el diagnóstico. En particular, se evaluó OsiriX, un software de código abierto y con arquitectura basada en plugins, como herramienta de desarrollo y para valorarlo se implementó la metodología de segmentación por corrimiento de media ponderado con mapas de confianza de bordes. Con los resultados obtenidos, lo que podemos concluir es que OsiriX es una herramienta que ofrece las ventajas y facilidades para cumplir con el objetivo general. Cuando se trabaja en procesamiento de imágenes, siempre tiene que construirse una etapa de visualización, de reconstrucción y de manejo de las imágenes procesadas, con OsiriX

se tienen estas opciones y cuando se incorpora un plugin, pueden utilizarse todas estas herramientas; por esto, el desarrollador puede enfocarse y trabajar en el procedimiento particular que va a implementar. OsiriX, por ser software libre y de código abierto puede ser actualizado sin costo y permite ser modificado y distribuido sin necesidad de rendir cuentas a un propietario, por este motivo también, puede actualizarse constantemente sin estar encadenado a un propietario. Otra ventaja es que OsiriX funciona como una estación de trabajo para imágenes en formato DICOM, dentro de una red o PACS (Picture Archiving and Communication System) también basada en componentes de software libre (free PACS), lo que permite pensar en una solución que abarque un sistema completo, visualización y procesamiento, almacenamiento y comunicación de imágenes, todo con elementos de software libre. Para el desarrollo de proyectos, OsiriX posee clases que con sus métodos y atributos facilitan el procesamiento de imágenes, además acepta todas las clases de Cocoa para la implementación del plugin. Al utilizar Proceso Unificado de Desarrollo de Software (UP) y la representación de los resultados a través de diagramas de UML, se obtuvo un modelo del proceso de segmentación por corrimiento de media ponderado con mapas de confianza, lo cual, también es un aporte de este trabajo, ya que aunque el desarrollo fue terminado y probado en Matlab, no se hizo un modelo de la metodología.

La implementación de un plugin de OsiriX se facilita con modelado orientado a objetos para la mejor comprensión y solución del problema, y para futuras extensiones del plugin desarrollado.

OsiriX, a pesar de los errores de precisión debido al manejo en float de los datos, es una buena herramienta de transporte e implementación de metodologías de procesamiento y ofrece muchas posibilidades para la comunicación y el uso en la vida real de metodologías o técnicas de procesamiento que han sido desarrollados en universidades y centros de investigación.

OsiriX, por lo tanto, ofrece muchas ventajas en el desarrollo de herramientas para el procesamiento de imágenes médicas. Por ejemplo, la posibilidad que tiene OsiriX de trabajar con grandes cantidades de imágenes y con formatos diferentes simplifica la implementación de plugins que ya no necesitan solventar este tipo de problemas.

La arquitectura extensible de OsiriX (y de otros programas de código abierto) a través de plugins, fortalece en gran medida la continuidad y la comunicación de resultados entre desarrolladores y usuarios. La utilización de un lenguaje orientado a objetos y de clases, permite la modularidad en la construcción de un proyecto y la fácil comprensión para futuros programadores que necesiten extender la aplicación.

El ejemplo de programación anteriormente presentado, muestra cómo es posible adaptar

desarrollos de software propios, a un sistema extensible de código abierto para hacer accesible nuevos desarrollos en el área de la segmentación y el procesamiento de imágenes al campo clínico. La unión entre los desarrollos de los centros de investigación y la estación de visualización basada en OsiriX permitirá a muchos hospitales iniciar una migración de sus sistemas de radiografía convencional a sistemas de almacenamiento y comunicación de imágenes sin tener que depender exclusivamente de un proveedor que los convierta en clientes cautivos.

Es muy importante mencionar que una aplicación completa en ingeniería incluye al usuario final, esto es fundamental. El trabajo desarrollado a partir de ideas preexistentes son trabajos trascendentes. Los proyectos de Ingeniería Biomédica relacionan usuarios finales con desarrollos científicos, esta es una de las importantes características de esta disciplina.

El trabajo que se presenta en esta tesis es un ejemplo de un desarrollo de ingeniería que adapta conocimiento científico.

5.5. Trabajo Futuro

Como trabajo futuro se plantea:

1. Evaluar y ponderar las diferencias entre los resultados de la implementación en OsiriX y en Matlab.

2. Completar la metodología propuesta por [3], incorporando información *a priori* con mapas probabilísticos de materia gris, materia blanca y líquido cefalorraquídeo. Esta etapa es la clasificación de las regiones resultantes de la segmentación, para lo cual es necesario hacer el registro al espacio de Talairach para hacer la correlación.

Apéndice A

Software Libre y GNU

A.1. Software Libre

Desde hace dos décadas, surgió una nueva filosofía acerca del uso y distribución de software. Este nuevo proyecto se contrapone a las suposiciones que las compañías de software hacen para justificar sus "derechos". Una de estas presunciones es que estas compañías tienen el derecho natural e incuestionable de poseer el software y por lo tanto tener poder total sobre los usuarios. Otro punto que asumen es que la única cosa importante del software es el trabajo que permite realizar en forma individual y no el tipo de sociedad que nos hace tener esta visión privativa. La tercera presunción es que no se puede tener un software totalmente utilizable si no se le da a la compañía el poder total sobre los usuarios del programa.

Estos conceptos cambian totalmente si se piensa en función del usuario. Por eso se inicia un proyecto que busca una nueva visión de software que pueda ser libremente utilizado y ajustado a las necesidades particulares.

Software de libre acceso es software que tiene la autorización para ser utilizado, copiado, distribuido y modificado por cualquier usuario, ya sea en forma gratuita o con honorarios. En general, lo que esto implica es que el código fuente debe ser accesible.

Hay que hacer notar que el término "Software libre", se refiere a libertad, no a precio.

El software de libre acceso es típico de libertad para utilizar, copiar, distribuir, estudiar, cambiar, mejorar el software. Se refiere principalmente a cuatro tipos de libertad para los usuarios del software:

* Libertad de correr el programa, con cualquier fin o propósito (libertad 0).
* Libertad de estudiar como trabajan los programas, y adaptarlo a las necesidades particulares (libertad 1). Una condición para esto es el acceso al código fuente.
* Libertad de redistribuir copias, y así poder ayudar a otros (libertad 2).

* Libertad de mejorar el programa, y transmitir las mejoras al público en general, y de esta forma la comunidad entera se beneficia (libertad 3). Por lo tanto, el acceso al código es vital en este punto.

También se tiene la libertad de hacer modificaciones y usarlas en forma privada, sin mencionar que existen. Si se desea publicar los cambios, no es necesario notificar a nadie.

La libertad de distribuir copias incluye formas binarias o ejecutables del programa, así como código fuente, para versiones modificadas y no modificadas.

Son aceptables algunas reglas sobre la manera de redistribuir el software de libre acceso, siempre y cuando no haya conflictos con las libertades centrales. Por ejemplo el "copyleft" es una regla en la cual, cuando se distribuye un programa no se pueden añadir restricciones para negar a otros las libertades básicas. Esta regla, en vez de entrar en conflicto con las libertades, las protege.

Software de libre acceso no significa "No comercial". Un programa libre puede ser accesible para uso, desarrollo y distribución comercial.

El término "Open source" o "código abierto" es utilizado por algunas personas como sinónimo de software de libre acceso. Aunque no es exactamente la misma categoría de software, ya que el código abierto acepta algunas licencias que se consideran muy restrictivas y también hay licencias de software de libre acceso que los códigos abiertos no aceptan. De todas formas, las diferencias en extensión son pequeñas: casi todo el software de libre acceso es código abierto y casi todo el software de código abierto es libre [12][13].

Apéndice B

Software de Código Abierto: OsiriX

B.1. OsiriX

OsiriX es un software de código abierto [16] que se distribuye bajo la Licencia General Pública de GNU [13] y está dedicado al manejo de imágenes en formato DICOM (Digital Imaging and Communications in Medicine).

OsiriX es un desarrollo hecho por los médicos radiólogos Antoine Rosset y Osman Ratib [15] del Departamento de Radiología de la Universidad de Ginebra, Suiza y del Departamento de Radiología en la Universidad de California, Los Angeles (UCLA), respectivamente, y como se mencionó, puede descargarse gratuitamennte de la página web de OsiriX [16].

OsiriX presenta una serie de posibilidades en el manejo de imágenes:

* Manejo y visualización de imágenes multimodales y multidimensionales.

* La visualización 3D y 4D, ofreciendo reconstrucción multiplanar (MPR), representación de superficies, representación de volúmenes y proyección de máxima intensidad (MIP).

* Fusión de imágenes.

* Número arbitrario de imágenes simultáneamente desplegadas.

* El usuario puede controlar los parámetros de visualización (opacidad, tamaño, visibilidad, etc.).

* Funciona como una estación para Picture Archiving and Communication System (PACS) de imágenes DICOM.

Posee arquitectura extensible con plugins, es decir con módulos o programas que añaden una característica o función específica a un sistema mayor, en este caso OsiriX. Estos módulos o "Plugins" permiten adecuar la herramienta a las necesidades particulares de un proyecto o investigación en imagenología médica [15], [26].

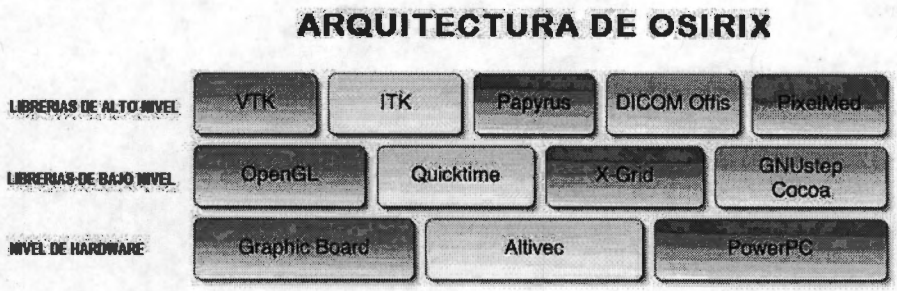


Figura B.1: Arquitectura general de OsiriX que muestra los componentes de código abierto y las librerías que utiliza

OsiriX fue creado como software de código abierto para que pueda adaptarse o ser versátil. La primera versión o propuesta de OsiriX fue el visualizador de DICOM OSIRIS que podía ejecutarse en Macintoch y en Computadoras Personales (PC) con Windows. OSIRIS tuvo gran aceptación y por eso, sus creadores decidieron extender las capacidades de este software para poder navegar en grandes series de datos multidimensionales. El nuevo nombre, OsiriX se adoptó para marcar la transición a la nueva plataforma añadiendo la "X" que indica la migración de los sistemas operativos de Macintosh a la versión 10, también llamada Mac OS X. La X también indica compatibilidad con Unix y la adopción del paradigma de código abierto [15].

B.1.1. OsiriX: Programa de código abierto

El programa de software es desarrollado como una aplicación para el sistema operativo Mac OS X. OsiriX fue desarrollado en base a una arquitectura construida con componentes de código abierto ya existentes, los cuales se muestran en la figura B.1. No es necesario conocer todas las librerías y componentes de OsiriX para poder aprovechar las ventajas que ofrecen.
Entre los principales componentes se encuentran:

1. "GNUstep/Cocoa": es un marco de trabajo orientado a objetos y multiplataforma para el desarrollo de las interfaces gráficas. Cocoa, como lo llamó Apple Computer in el ambiente MacOS X, es distribuido en formato de código fuente como GNUstep y permite a los usuarios crear complejas interfaces.

2. "OpenGL": es una librería gráfica estándar en la industria, sirve para la visualización de imágenes en 3D. Open GL originalmente fue desarrollado por SGI (Silicon Graphics) con el nombre de IrixGL y solo estaba disponible en grandes terminales dedicadas en estaciones de trabajo Unix. Luego, OpenGL fue adaptado por manufactureros de tarjetas gráficas como NVIDIA y ATI para la construcción de PCs. OpenGL permite

al usuario tomar ventaja de la aceleración del hardware a través de las tarjetas gráficas 3D cuando las hay. Es la única librería multiplataforma en 3D disponible diseñada para la aceleración de hardware y aunque fue diseñada para reconstrucción en 3D tiene un gran desempeño en representaciones en 2D.

3. "Visualization Toolkit" (VTK): es una librería de código abierto, orientada a objetos y de multiplataforma para el procesamiento y visualización de imágenes en 3D. Esta herramienta ofrece una gran cantidad de funciones para la manipulación y despliegue de series de imágenes 3D. Ha sido construida sobre OpenGL para asegurar representaciones que usen aceleración de hardware en diferentes plataformas. VTK permite a OsiriX muchas de las características de manipulación y despliegue tales como MIP, MPR, reconstrucción 3D, etc.

4. "Insight Segmentation and Registration Toolkit" (ITK): es un extenso conjunto de librerías para el procesamiento de imágenes médicas. Es una extensión de VTK y esta basado en el mismo marco de trabajo. ITK provee de una serie de algoritmos de procesamiento para resolver necesidades en el ámbito de la imagenología médica. También se distribuye bajo la licencia de código abierto.

5. "Papyrus Toolkit": para el manejo de imágenes DICOM. Esta librería de dominio público desarrollada en la Universidad de Ginebra ofrece todas las funciones necesarias para leer y escribir archivos de DICOM. Esta librería se distribuye con el código fuente de ANSI C, y es fácilmente adaptable a cualquier plataforma. Provee de un conjunto de funciones para el manejo de imágenes en el complejo e incómodo formato DICOM.

6. "DICOM Offis": para las funciones de redes DICOM: es una librería multiplataforma que soporta los protocolos de comunicación DICOM que hacen posible las preguntas, el envío, la recuperación y la recepción de imágenes DICOM a través de un PACS. El código fuente en C++ también está disponible. Esta librería provee de un conjunto muy completo de funciones que facilitan el manejo y la implementación de protocolos de red DICOM muy complejos.

7. "Altivec" de PowerPC es función de nivel bajo (nivel ensamblador) para la aceleración de funciones de procesamiento digital de señales (DSP), lo cual asegura alto desempeño e interacción en tiempo real. Altivec es la única plataforma específica y no de código abierto disponible para plataformas de PowerPC.

8. "Quicktime" para el intercambio de imágenes en formato multimedia. Muchos visualizadores comerciales de DICOM o estaciones de trabajo en 3D permiten al usuario ver y manipular imágenes, pero carecen de una forma conveniente de exportar imágenes a un formato multimedia estándar. Esto limita la capacidad de comunicar los resultados a otros usuarios o aplicaciones. Quicktime permite dar soporte a un mayor número de

formatos, tales como TIFF, Photoshop, JPEG, JPEG2000 y BMP y a secuencias en vídeo en formatos cono AVI, MPEG y MPEG4. Quicktime permite importar a OsiriX formatos diferentes a DICOM.

Apéndice C

Metodología para el Desarrollo de un Plugin en OsiriX

C.1. Cómo construir un plugin

OsiriX es una aplicación para MAC OS X, por lo que las herramientas necesarias para desarrollar un plugin para OsiriX son:

- **OsiriX**, debe estar instalado en *Applications*, si no es así, descargarlo de ***http://www.osirix-viewer.com/*** [16].
- **Xcode**, debe estar instalado en *Developer/Applications/*, de no ser así puede descargarse de la página de Apple Developer Connection (***http://developer.apple.com/macosx/***). Se recomienda la lectura de [17].
- **Objective-C**, es el lenguaje de programación que se va a usar. Se recomienda la lectura de [18].
- **Cocoa**, la librería de clases o "framework" que contiene métodos útiles para la programación de interfaces gráficas y para el manejo de valores estándar. Se recomienda la lectura de [17].
- **Interface Builder,** que es un aplicación para construir y probar interfaces gráficas. Trabaja con Xcode y es parte de MAC OS X, pero también puede descargarse de ***http://developer.apple.com/macosx/***, la página de Apple Developer Connection.

Por lo tanto, un plugin para OsiriX es un proyecto de aplicación en *Xcode* que utiliza recursos como la librería de clases, *Cocoa* y las interfaces gráficas de *Interface Builder*.

Para que el proyecto de aplicación en Xcode se convierta en un plugin para OsiriX es necesario añadir un directorio con 7 clases importantes para la comunicación del plugin y OsiriX: `PluginFilter`, `ViewerController`, `DCMViewer`, `DCMPix`, `dicomFile`, `ROI`, `MyPoint` [19].

C.1.1. Tutorial para crear un plugin en OsiriX con *OsiriX Plugin Generator*

1. Entrar en la página web *https://osirix.svn.sourceforge.net/svnroot/osirix/plugins/_help/* [19] y elegir y descargar el *OsiriX Plugin Generator.app* (ver Figura *C.1*). Allí también se puede descargar un Manual de Plugins (PluginsManual.pdf) que tiene información, escrita por el autor de OsiriX, para crear un plugin.
2. Hacer doble click sobre *OsiriX Plugin Generator.* Aparecerá la ventana que se muestra en la Figura C.2, introducir el nombre del nuevo plugin, que en este caso es Pruebas. Se desplegará, también, una ventana para el creador del nuevo plugin.
3. Buscar el directorio *Pruebas* (se genera en el directorio donde se encuentra el *OsiriX Plugin Generator.app,* que en este caso y como se ve en la Figura C.3 es TESISSV) , como se muestra en la Figura C.3 este generador de plugins crea el ambiente mínimo indispensable para crear un plugin, es decir, el proyecto en Xcode (`Pruebas.xcodeproj`), las clases de interface con OsiriX (`OsiriX Headers`), el archivo prefijo (`Pruebas_Prefix.pch`), el archivo de lenguaje (`English.lproj`), archivos de información general (nombre y ubicación, `Info.plis` y versión, `version.plist`) y genera también, una clase (archivo `PruebasFilter.m` y `PruebasFilter.h`), la clase principal, donde esta el método `filterimage`, es decir, `((long) filterImage:(NSString*) menuName)`.
4. Hacer doble click sobre `Pruebas.xcodeproj, el` proyecto en Xcode y se va a desplegar la ventana que se muestra en la Figura C.4, donde puede observarse la información mencionada en el paso anterior y además información sobre los "frameworks", que en este caso es Cocoa: AppKit y Foundation.
5. El ejecutable no está definido por lo que, hay que posicionarse sobre el directorio *Executables*, apretar *control* + click y como se muestra en la Figura C.5, se despliega una ventana. Elegir *Add* y *New Custom Executable.* Inmediatamete se despliega una ventana que solicita colocar el nombre y la ubicación (Choose) de la aplicación, en este caso OsiriX.

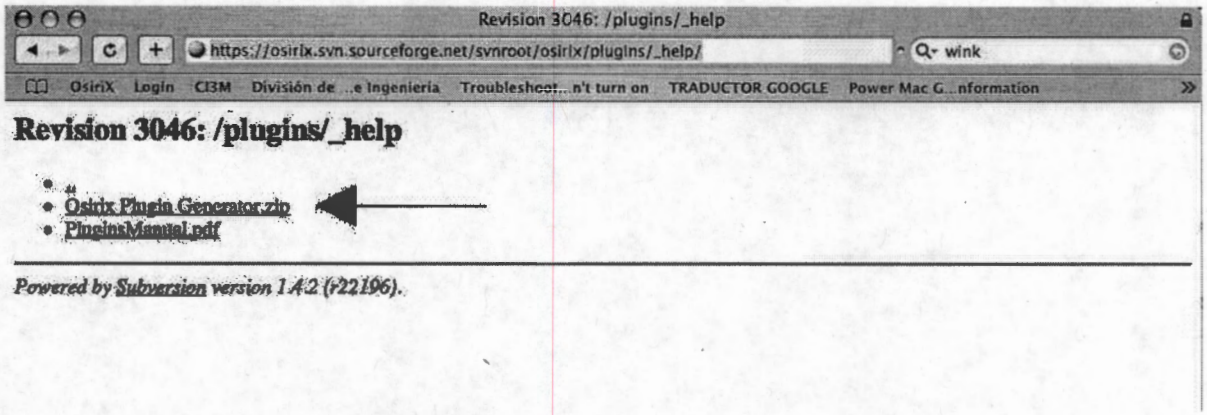


Figura C.1: Se elige el OsiriX Plugin Generator para crear un nuevo plugin

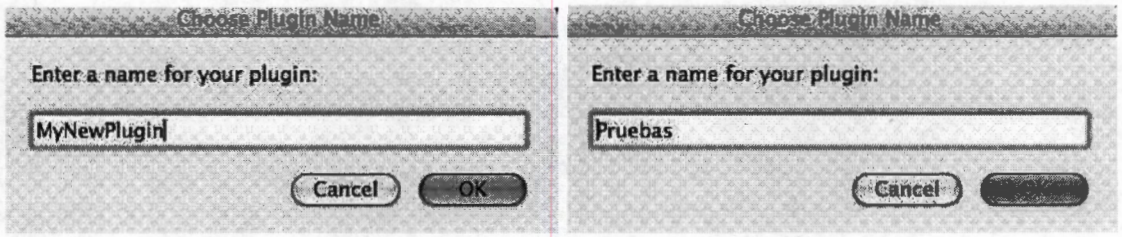


Figura C.2: Nuevo plugin generado con *OsiriX Plugin Generator.app*.

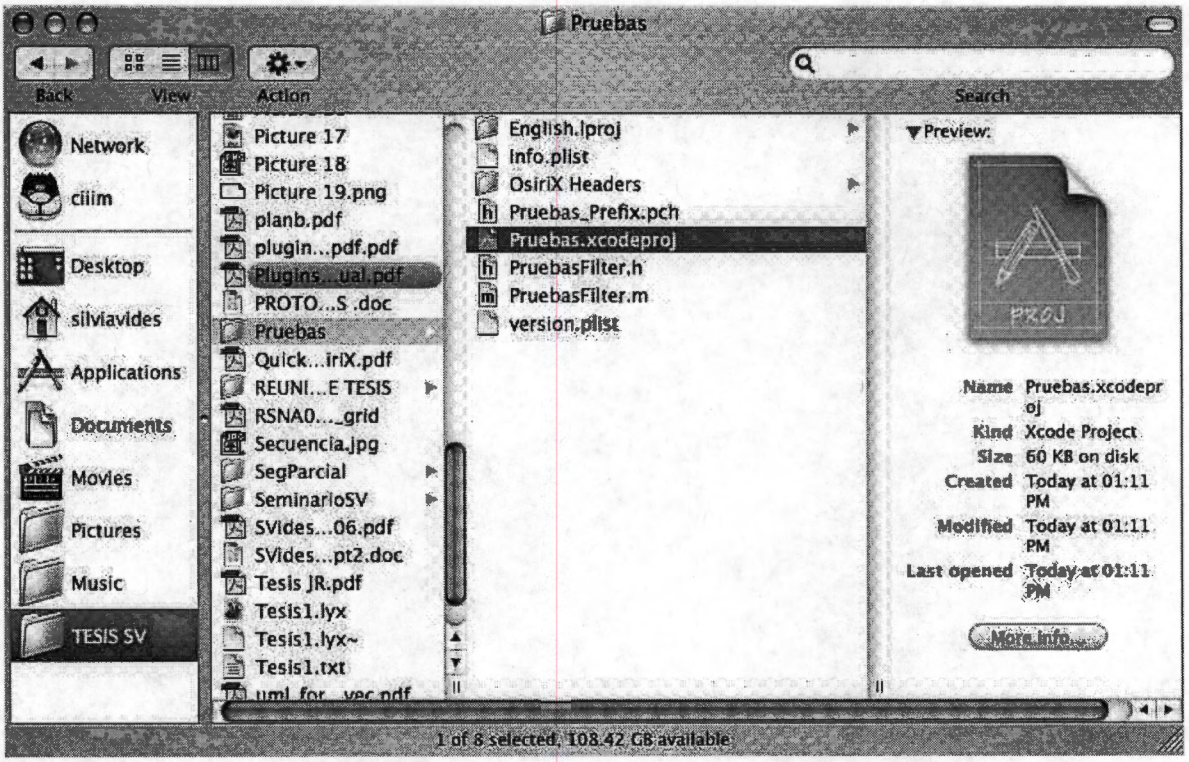


Figura C.3: Directorio del nuevo plugin Pruebas

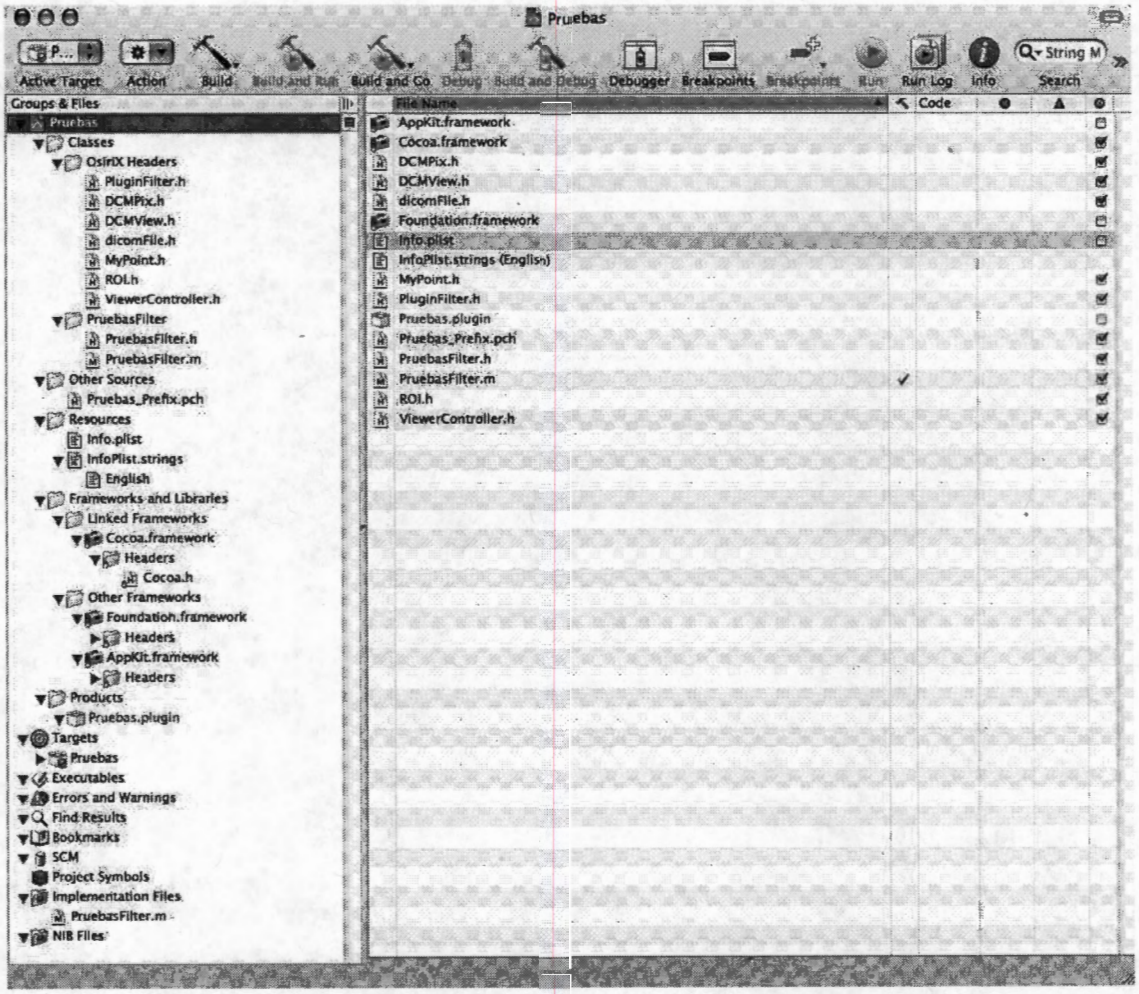


Figura C.4: Proyecto en Xcode del plugin *Pruebas*

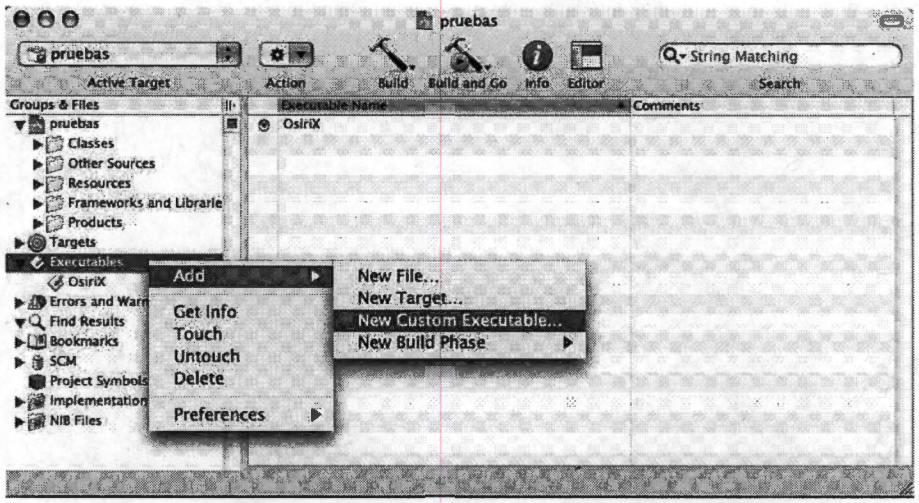
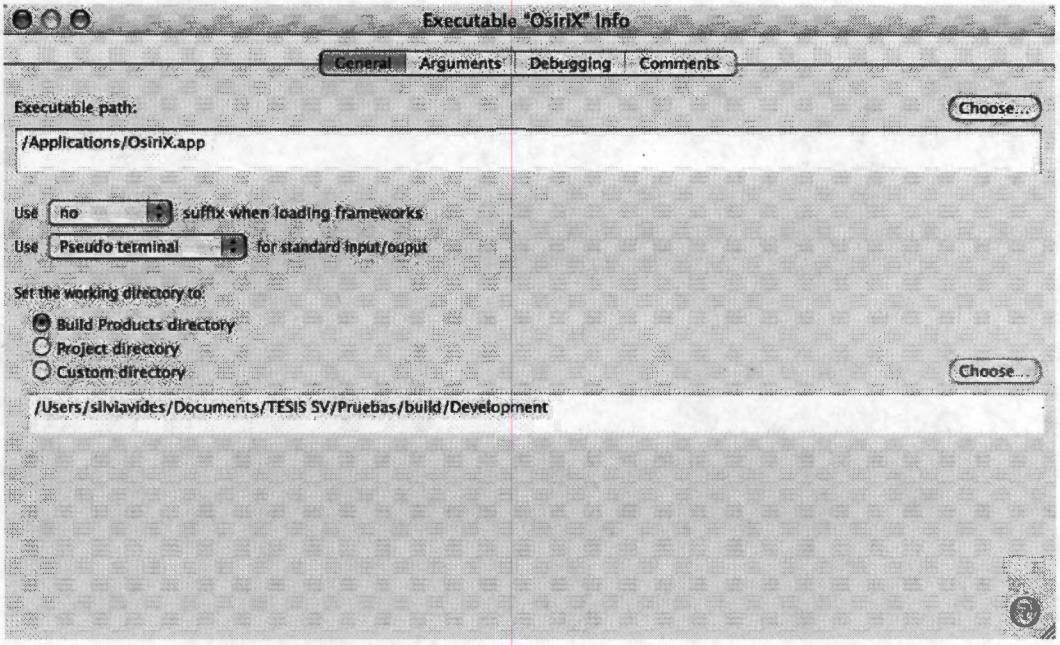
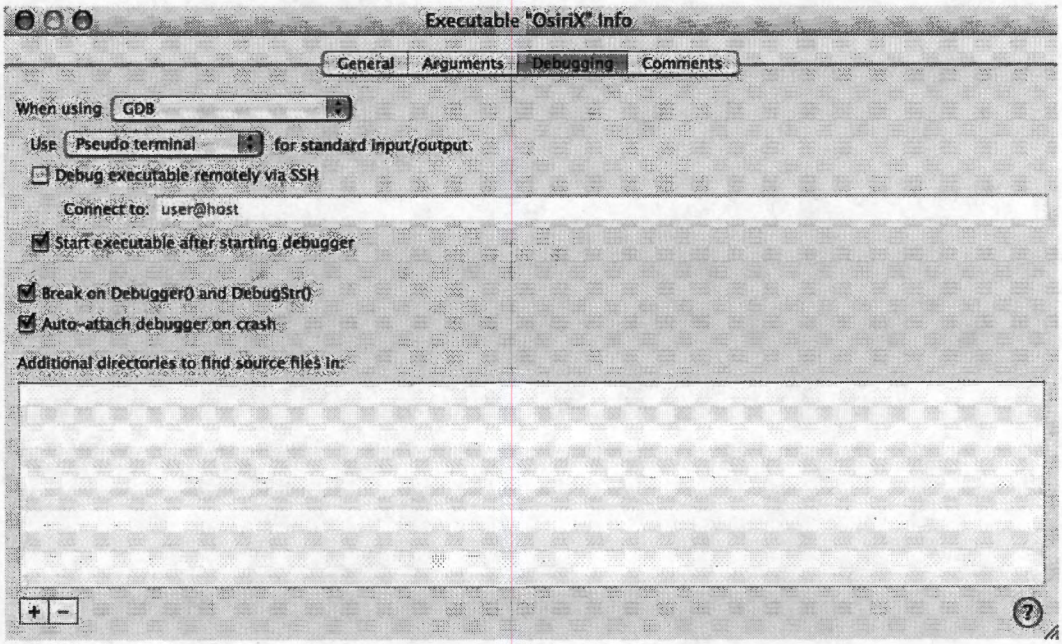


Figura C.5: Cómo añadir OsiriX como ejecutable de una aplicación de Xcode, en este caso, de un plugin.

Para ajustar algunos parámetros, colocarse sobre OsiriX, *control* + click y elegir *Get Info*, entonces aparece la ventana de información que se muestra en la Figura C.6, se recomienda elegir los parámetros para *General* y *Debugging* que se muestran. Cuando se terminan los ajustes de depuración, se cierran las ventanas y se regresa a la vista de la Figura C.4, es decir, el proyecto en Xcode.



(a)



(b)

Figura C.6: Información sobre *Executables*. En el caso de un plugin para OsiriX, se recomienda dejar los parámetros de *General* (a) y de *Debugging* (b) como se muestran en estas imágenes.

6. Hacer doble click sobre *PruebasFilter.h* y *PruebasFilter.m* , se despliegan los archivos que se muestran en la Figura C.7 donde se van a escribir o codificar los atributos y

métodos de la metodología de procesamiento de imágenes que se va a implementar. Por definición está escrito el código para hacer un duplicado de la ventana de la imagen que se está procesando. El único método es el `((long) filterImage:(NSString*) menuName)` que define el inicio del plugin.

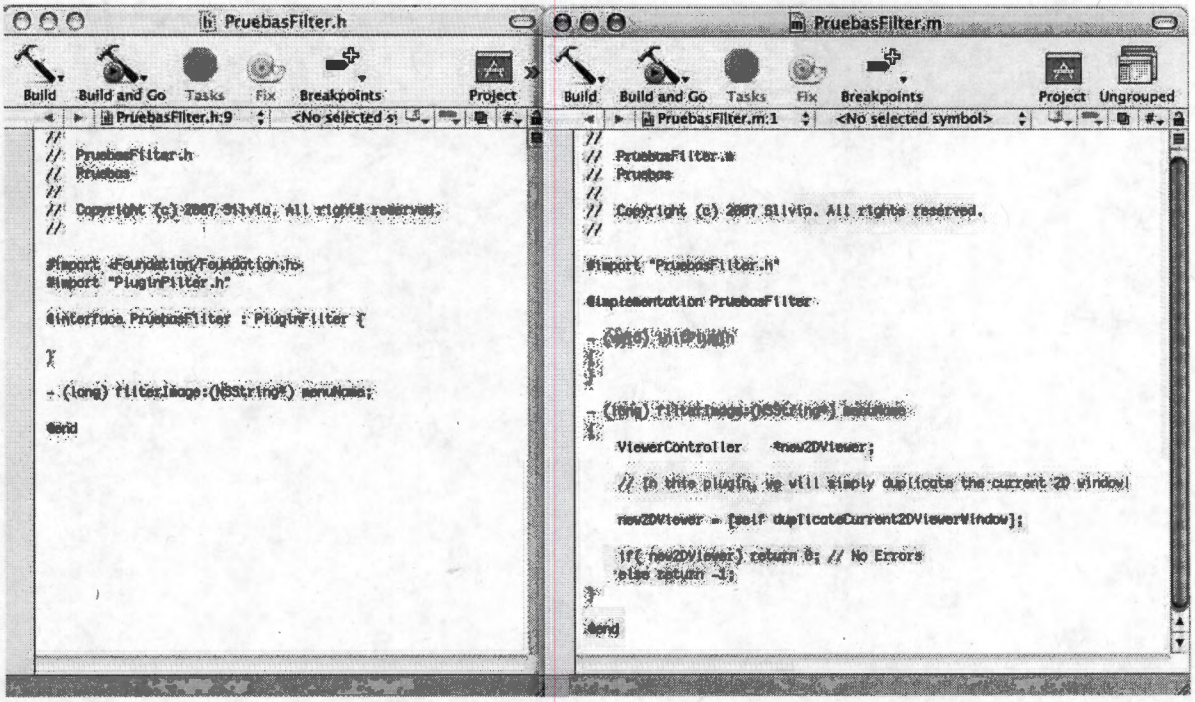


Figura C.7: Archivos .h (header) y .m (module) de la clase PruebasFilter, para editar los atributos y métodos de la metodología de procesamiento que se va a implementar. Por definición, se añade el método `((long) filterImage:(NSString*) menuName)`

7. En la ventana de Xcode, oprimir el botón de *Build*, para construir el plugin (Figura C.8 (A)) para generar el archivo compilado, este archivo es Pruebas.plugin y se encuentra en el directorio Build dentro del directorio de Pruebas (Figura C.3), como se muestra en la Figura C.9. Se hace un alias de Pruebas.plugin y se coloca en */Library/Application Support/OsiriX/Plugins* porque allí lo va a buscar OsiriX cuando sea ejecutado.

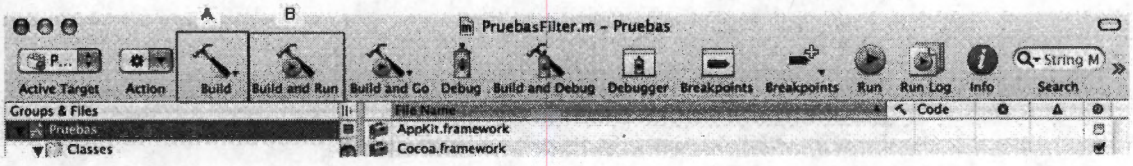


Figura C.8: Ventana de Xcode (A) Build, (B) Build and Run

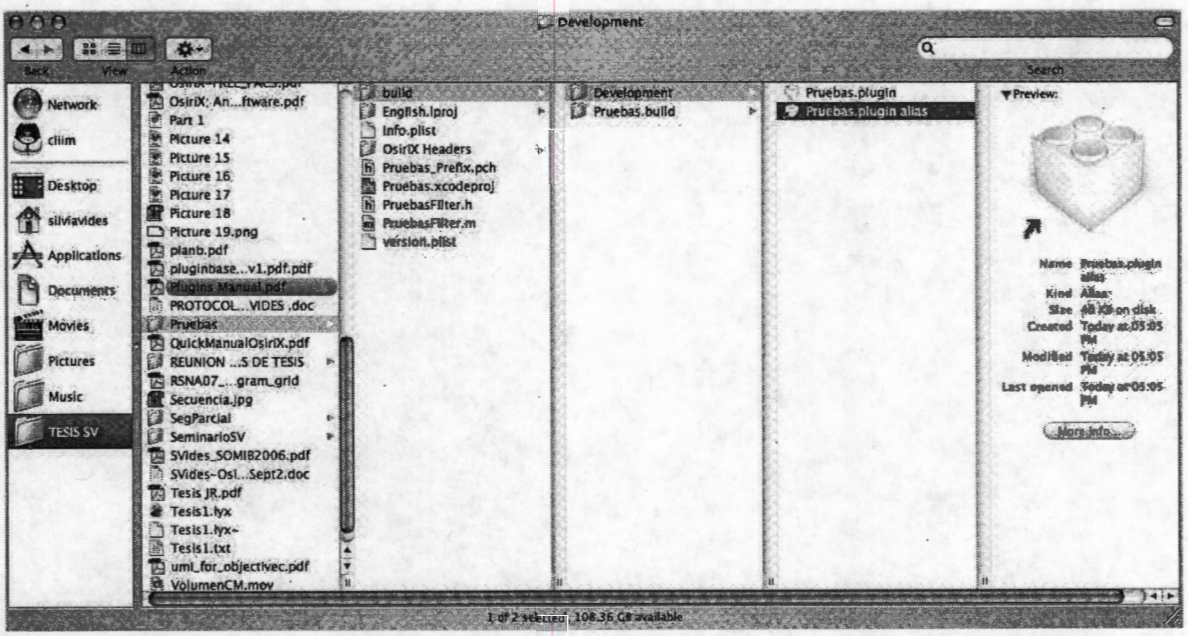


Figura C.9: Plugin compilado dentro del directorio *Build*

8. En la ventana de Xcode, hacer doble click sobre Info.plist y aparece lo que se muestra en la Figura C.10. Allí se muestra el nombre que se va a desplegar en el menú de Plugins de OsiriX, en este caso es *Pruebas*; el tipo de plugin, que por definición es *Image Filter* (Filtros de imagen) pero puede ser *roitool* o puede ser *others*. También aparece la información sobre la clase prinicipal, que en este caso es PruebasFilter.

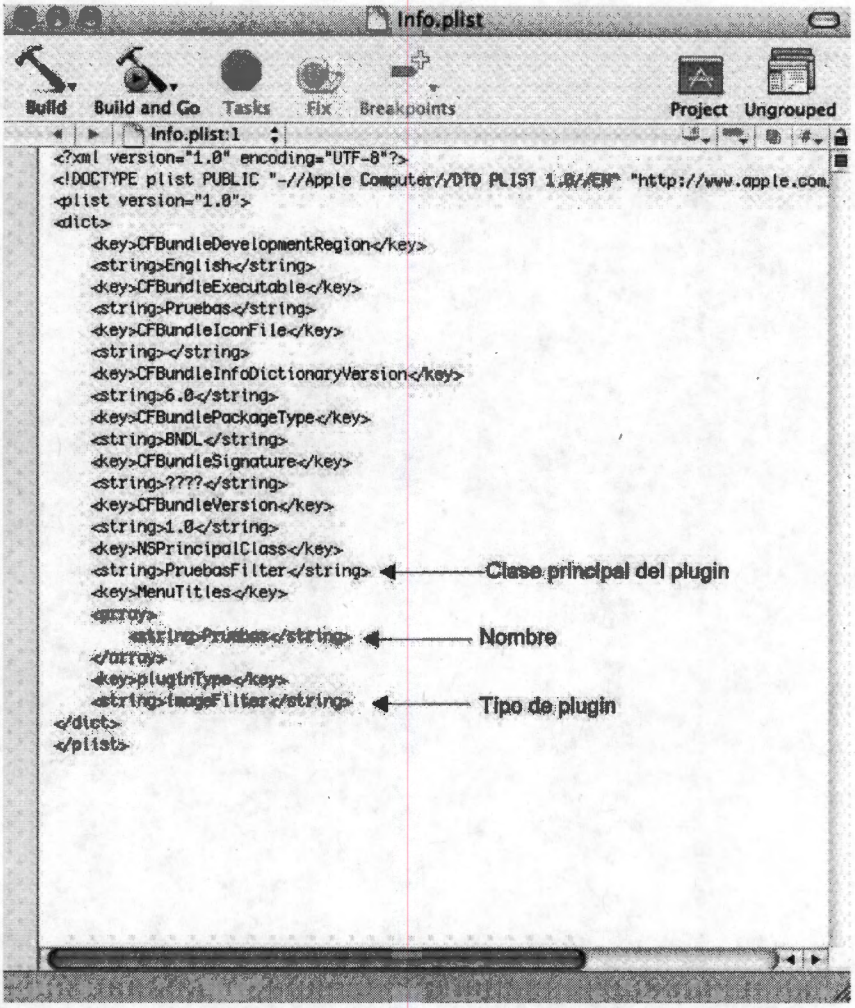


Figura C.10: Info.plist del plugin Pruebas.

9. En la ventana de Xcode oprimir el ícono de *Build and Run* (Figura C.8 (B)) y Xcode ejecutará automáticamente a OsiriX (si no hay errores, de lo contrario éstos aparecerán en la ventana de Xcode en *Errors and Warnings*). El plugin Pruebas debe aparecer en el menú de Image Filters en la opción de Plugins de la barra de OsiriX, como se muestra en la Figura C.11, en este momento está listo para aplicarse sobre la imagen desplegada.

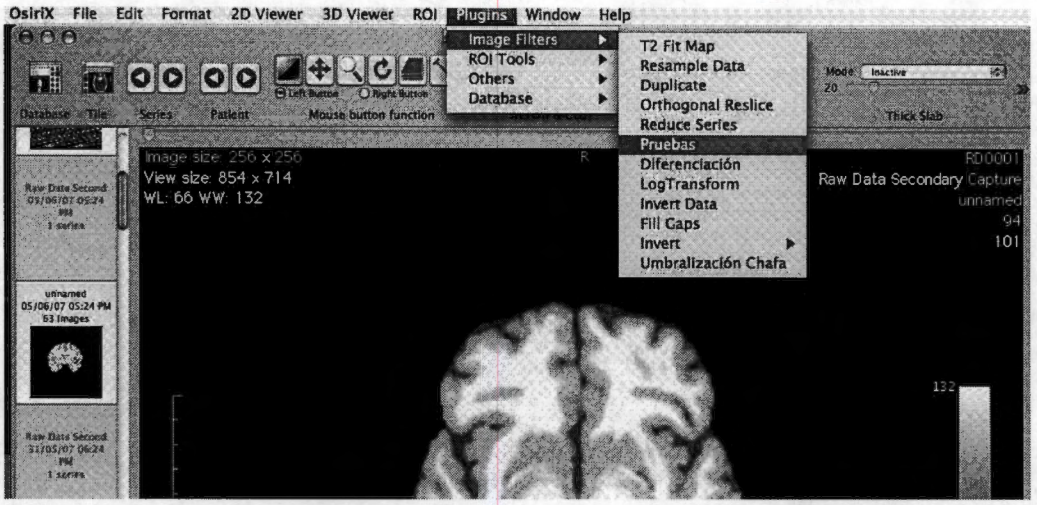


Figura C.11: Plugin Pruebas en la barra de opciones de OsiriX

C.1.2. Como construir un plugin a partir de otro ya existente

Otra forma de crear un plugin puede ser duplicar [19] un plugin existente y para que funcione, sustituir el nombre por el del nuevo plugin en todos los lugares donde aparece el nombre del plugin que se ha duplicado. Si no se tienen plugins para duplicar pueden obtenerse de la página [27] o de la página principal de OsiriX [16].

1. Duplicar un plugin ya existente.

2. Sustituir el nombre del antiguo plugin por el del nuevo. Por ejemplo, se tiene el plugin llamado "*pruebas*" que se muestra en la Figura C.12, se hace un duplicado de este plugin y se sustituye todo lo que dice "pruebas" por "HolaMundo" el nuevo nombre.
Los archivos a sustituir son:

- `pruebasFilter.h` por `HolaMundoFilter.h`
- `pruebasFilter.m` por `HolaMundoFilter.m`
- `pruebas.xcodeproj` por `HolaMundo.xcodeproj`
- `pruebas_Prefix.pch` por `HolaMundo_Prefix.pch`

Por lo tanto, la carpeta quedará como se muestra en la Figura C.13.

3. Después es necesario borrar el directorio "*Build*" porque allí se encuentra el plugin compilado y ahora es necesario crear uno nuevo.

4. Se inicia Xcode haciendo doble click sobre el archivo `HolaMundo.xcodeproj`.

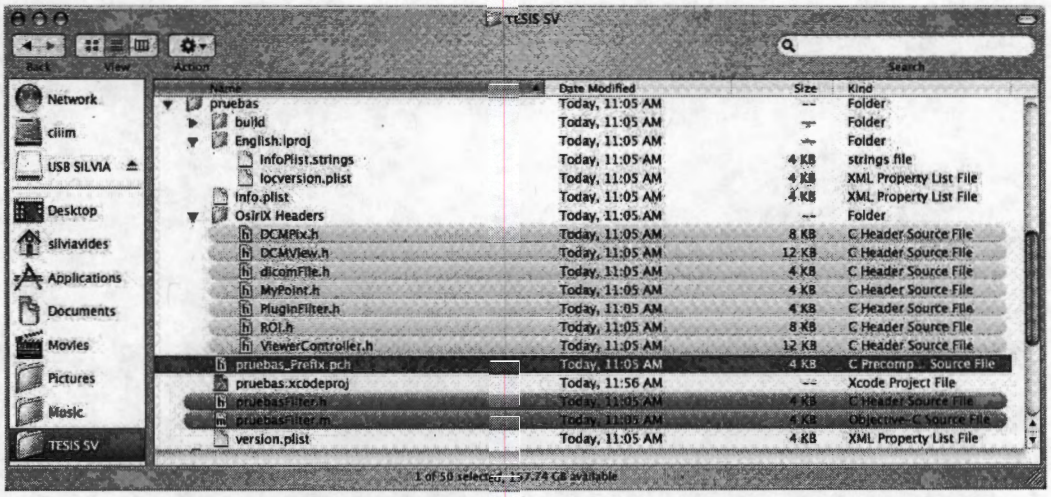


Figura C.12: Directorios y archivos que pertenecen al plugin "pruebas"

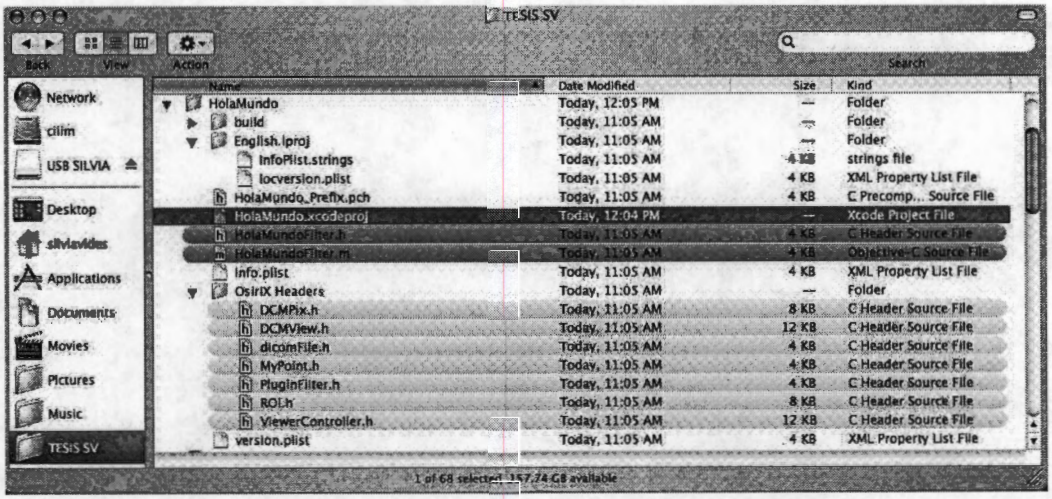


Figura C.13: Nuevo plugin "Hola Mundo" con los nombres de los archivos sustituidos

5. Ya dentro del ambiente Xcode es necesario sustituir el nombre de `pruebasFilter.h`, `pruebasFilter.m` y `pruebas_Prefix.pch` así como definir la nueva ruta hacia los archivos. Esto se hace en la opción de *Get Info* de menú *File*. Esto se muestra en la Figura C.14.

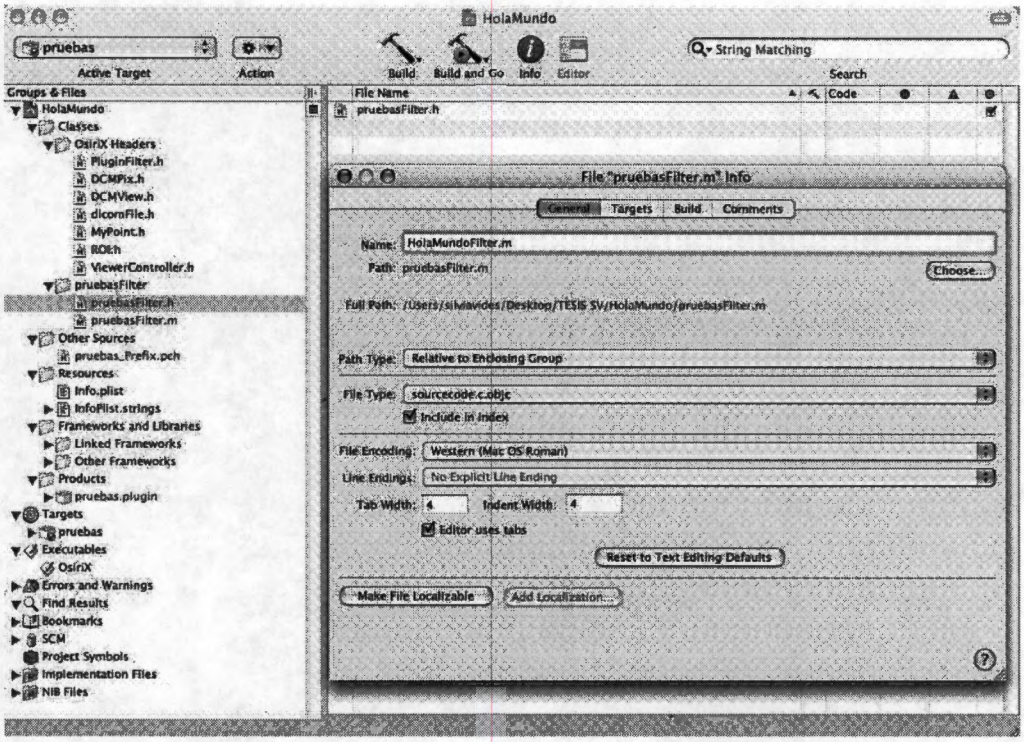


Figura C.14: Sustitución de nombres y rutas para generar el nuevo plugin

6. Borrar "*pruebas.plugin*" del directorio *"Products"* del mismo ambiente de Xcode. Luego se selecciona *Get Info* sobre "pruebas" en Targets y se sustituye en las pestañas "*General*", "*Build*" y "*Properties*" cualquier cosa que diga "pruebas" por "HolaMundo" como se muestra en la Figura C.15.

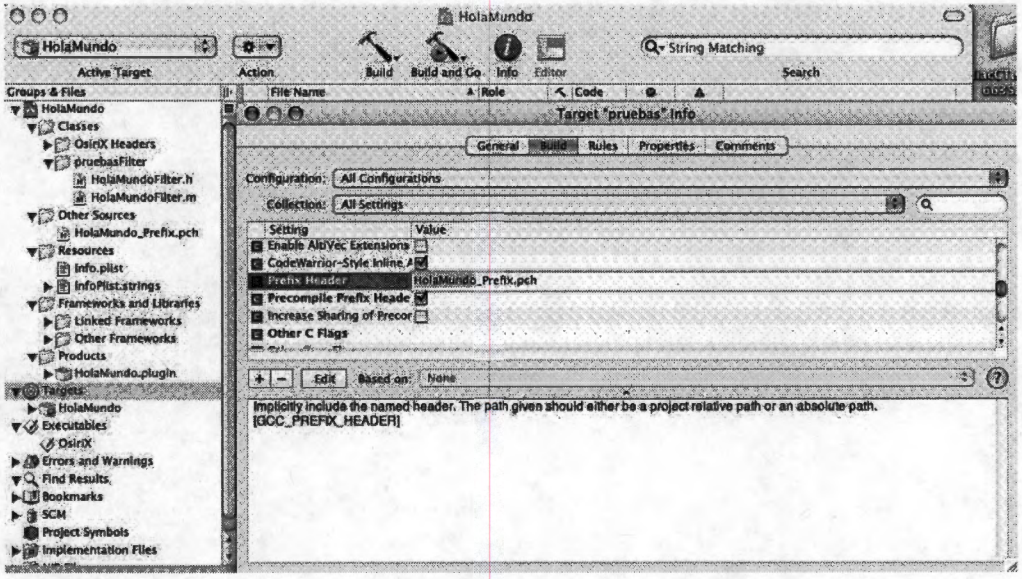


Figura C.15: "HolaMundo" Target en el ambiente Xcode

7. Definir el Ejecutable, tal como se mencionó anteriormente y se muestra en la Figura C.5.
8. Ahora hay que definir dónde y cómo aparece el nuevo plugin en el menú de OsiriX, para esto, se edita el `Info.plist` y se introducen uno o más nombres en "MenuTitles" por que es o son los nombres que van a ser desplegados en el menú. También hay que seleccionar qué tipo de plugin es, las opciones son:

- **imageFilter:** para que el nuevo plugin aparezca en el menú de '*Image Filters*'.
- **roiTool:** el plugin aparece en el menú de herramientas para Regiones de Interés ('*ROIs tools*').
- **other:** para que esté en el menú de '*Others*'.

9. Después hay que editar el archivo "*InfoPlist.string*" para los detalles de versión y nombre.
10. Editar los archivos *HolaMundo.h* y *HolaMundo.m* en el encabezado y en el código fuente.
11. Después de todos estos cambios, se construye el plugin ejecutando el "*Build*" (el martillo) en la barra principal de Xcode, entones en el directorio "*Build*" aparece (si no hay errores) el plugin compilado, es decir "holamundo.plugin". Ahora es necesario crear un "alias" de este plugin y colocarlo en */Library/Application Support/OsiriX/Plugins* porque es en este directorio donde OsiriX busca al arrancar la aplicación [19].
12. Ahora puede ejecutarse el plugin al hacer click en el botón de "*Build and Debug*" (martillo y círculo verde) para comprobar que OsiriX puede encontrar el plugin y Xcode es capaz de depurarlo.
Después de esto, pueden modificarse los códigos fuente (.m y .h) y generarse más clases para que el plugin haga lo que el usuario necesita.

C.1.3. Cómo instalar un plugin

Debido al concepto de software libre y de código abierto, existe una comunidad OsiriX, donde es posible intercambiar y evaluar plugins hechos por otros desarrolladores, por lo que en la página principal de OsiriX [16] es posible descargar plugins (es decir, archivos xxxxx.plugin) y para instalarlos es necesario colocar el archivo xxxxx.plugin en */Library/Application Support/OsiriX/Plugins* y se inicia OsiriX, el plugin debe aparecer en el menú correspondiente.

Bibliografía

[1] J. Azpíroz-Leehan, M. Cadena-Méndez, and F. Martínez-Licona, "Analysis of the medical imaging services at three hospitals of mexican national health system," *Journal of Medical Systems*, 2007. aceptado.

[2] J. R. Jiménez-Alaniz, V. Medina-Bañuelos, and O. Yáñez-Suárez, "Data-driven brain MRI segmentation supported on edge confidence and a priori tissue information," *IEEE Trans. Med. Imag.*, vol 25, pp. 74–83, January 2006.

[3] J. R. Jiménez-Alaniz, *Segmentación No paramétrica de Imágenes Cerebrales de Resonancia Magnética.* PhD thesis, Universidad Autónoma Metropolitana - Iztapalapa, 2006.

[4] J. R. Jiménez, V. Medina, and O. Yáñez, "Nonparametric density gradient estimation for segmentation of cerebral MRI," in *Proc. of the Second Joint EMBS/BMES Conference*, (Houston, Texas, USA), pp. 1076–1077, october 2002.

[5] J. R. Jiménez, V. Medina, and O. Yáñez, "Nonparametric MRI segmentation using mean shift and edge confidence maps," *SPIE Progress in Biomedical Optics and Imaging*, vol. 4, pp. 1433–1441, February 2003.

[6] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, pp. 32–40, January 1975.

[7] V. A. Epanechnikov, "Nonparametric estimation of a multivariate probability density," *Theory Prob. Appl.*, vol. 14, pp. 153–158, (URSS) 1969.

[8] J. Azpíroz-Leehan, V. Medina-Bañuelos, and J. Lerallut, *Procesamiento Digital de Imágenes.* Coordinación de Extensión Universitaria, UAM-Iztapalapa, 2003.

[9] P. Meer and I. Weiss, "Smoothed differentiation filters for images," *J. Visual Comm. and Image Representation*, vol. 3, pp. 58–72, 1992.

[10] R. Koekoek and R. F. Swarttouw, "The Askey-scheme of hypergeometric orthogonal polynomials and its q-analogue," Tech. Rep. 98-17, Delft University of Technology, Faculty of Information Technology and Systems, Department of Technical Mathematics and Informatics, 1998. http://aw.twi.tudelft.nl/ koekoek/askey.html.

[11] C. Ferreira and E. Mainar, "Estudio asintótico de polinomios ortogonales en la tabla de askey," *Rev. Real Academia de Ciencias Zaragoza*, vol. 57, pp. 147–188, 2002.

[12] Free Software Foundation http://www.fsf.org/licensing/essays/free-sw.html.

[13] GNU www.gnu.org/copyleft/gpl.html.

[14] H. Cervantes and S. Charleston, "Using a workflow engine in a plugin-based product line architecture," in *Lecture Notes in Computer Science LNCS (9th International Symposium on Component-Based Software Engineering - CBSE)*, vol. 4063, 2006.

[15] A. Rosset, L. Spandola, and O. Ratib, "OSIRIX: Open-source software for navigating in multidimensional DICOM images," *Journal of Digital Imaging*, vol. 17, no. 3, 2004.

[16]

[17] A. Hillegass, *Cocoa Programming for Mac OS X.* Addison-Wesley, 2004.

[18] S. Kochan, *Programming in Objective-C.* Sams Publishing, 2004.

[19] A. Rosset, "Plugin developers toolkit," 2007.

[20] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide.* Addison-Wesley, 1999.

[21] A. Martínez-Martínez, J. R. Jiménez-Alaniz, A. González-Márquez, and N. Chávez-Avelar, "DICOM static and dynamic representation through unified modeling language," *SPIE PACS and Imaging Informatics*, vol. 5371, pp. 79–89, 2004.

[22] I. Jacobson, G. Booch, and J. Rumbaugh, *Unified Software Development Process.* Addison Wesley Longman, Inc., 1999.

[23] Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California, 95054, U.S.A., *Sun Educational Services, Basic Track II Java*, 2003.

[24] IBSR, *Internet Brain Segmentation Repository.* www.cma.mgh.harvard.edu/ibsr.

[25] *McConnell Brain Imaging Centre.* http://www.bic.mni.mcgill.ca/brainweb/.

[26] A. Rosset, L. Pysher, L. Spandola, and O. Ratib, "OSIRIX: open source multimodality image navigation software," *SPIE PACS and Imaging Informatics*, vol. 5748, pp. 20–27, 2005.

[27] SourceForge OsiriX http://osirix.svn.sourceforge.net/viewvc/osirix/.

PUBLICACIONES

Plugin para OsiriX: Detección de Bordes por Magnitud de Gradiente.

S.Vides Cañas, J. Azpíroz Leehan, J. R. Jiménez Alaniz

Laboratorio de Investigación en Neuroimagenología y
Centro de Investigación de Instrumentación e Imagenología Médica
Departamento de Ingeniería Eléctrica, Universidad Autónoma Metropolitana, Iztapalapa, México

Resumen —**OsiriX es una herramienta flexible que permite una interacción de colaboración entre aplicaciones dedicadas al manejo, procesamiento y visualización de imágenes médicas. En este artículo se describe un ejemplo de la integración y extensión que permite OsiriX, al implementar un algoritmo de detección de bordes por cálculo de gradiente. OsiriX provee un marco de desarrollo abierto para la integración de métodos de procesamiento de imágenes, y ofrece una serie de ventajas para la programación de plugins. Esta arquitectura de software, permite ampliar y enriquecer la herramienta para aplicaciones específicas, de tal manera que los desarrollos en laboratorios de investigación en procesamiento y segmentación de imágenes puedan tener sus aplicaciones en el medio ambiente clínico.**

Palabras clave— **Gradiente, imágenes médicas, OsiriX, plugin, procesamiento de imágenes.**

I. Introducción

OsiriX, es un programa de código abierto dedicado al procesamiento de imágenes en formato DICOM, siendo el actual estándar en imagenología médica. Presenta la ventaja de tener una arquitectura extensible con módulos o programas que añaden una característica o función específica a un sistema mayor, en este caso OsiriX. Estos módulos o "plugins" permiten adecuar la herramienta a las necesidades particulares de un proyecto o investigación en imagenología médica.

El programa ha sido implementado como una aplicación de OS X (sistema operativo de los sistemas Apple Macintosh), basada en Cocoa, por lo que la construcción de un plugin tiene acceso inmediato a las clases de Cocoa; la codificación se realiza a través de Objective C, que es un lenguaje de programación orientado a objetos.

Algunas de las ventajas de OsiriX:

-Manejo y visualización de imágenes multimodales y multidimensionales.
-Visualización 3D y 4D, ofreciendo reconstrucción multiplanar (MPR), representación de superficies, representación de volúmenes, y máxima proyección de intensidad (MIP).
-Fusión de imágenes.
-Número arbitrario de imágenes desplegadas simultáneamente.
-Parámetros de visualización (opacidad, tamaño, visibilidad, etc.) controlados por el usuario.
-Filtros para procesamiento de imágenes y funciones disponibles a través de plugins.
-OsiriX además funciona como una estación DICOM/PACS para imágenes médicas.

OsiriX se distribuye bajo la Licencia General Pública de GNU, por lo tanto su código fuente esta disponible para todos, y puede modificarse y distribuirse libremente. Este software para desarrollo de estaciones de visualización se aplica en la actualidad en más de 50 hospitales e institutos en el mundo entero.

Parte del interés de desarrollar este módulo computacional para la estación de visualización proviene del hecho de que se trata de una primera aplicación de métodos de procesamiento de imágenes y de reconocimiento de patrones que se han desarrollado dentro del Laboratorio de Investigación en Neuroimagenología (LINI) del Departamento de Ingeniería Eléctrica de la UAM Iztapalapa. Una vez adquirida la experiencia en el desarrollo de este plugin, se desarrollarán otros que aplicarán la experiencia en el análisis estadístico de imágenes para la segmentación de las mismas. El objetivo final es transportar los resultados de aplicación básica desarrollados en LINI al entorno clínico dentro de una plataforma de software libre para lograr que estos desarrollos tengan trascendencia para el cuidado de los pacientes.

II. Descripción

En la implementación de un plugin en OsiriX, se deben utilizar las siguientes herramientas:

El lenguaje de programación Objective C, que se basa en SmallTalk-80 y algunas librerías de C. Objective C es parte del ambiente de desarrollo de la Fundación para el Software Libre de GNU. [6]

Cocoa, es una colección de clases divididas en: Foundation, que es el conjunto elemental de utilidades (caracteres, datos, listas, etc.) y AppKit, que se refiere a interfaces gráficas de usuario (botones, ventanas, dibujos, etc.).

Otra herramienta, es el ambiente de desarrollo, Xcode, que es una aplicación que permite compilar, revisar y ejecutar programas en forma sencilla. [5]

Para desarrollar un plugin en OsiriX, deben generarse una serie de archivos, que contienen objetos de las clases fundamentales o superclases del plugin. Es decir, que el

plugin que se esta creando será una subclase de uno de estos "OsiriX Headers" u objetos que por tanto, son indispensables. Ver Fig. 1

En la implementación del plugin se deben incluir los archivos version.plist e Info.plist que contienen información sobre el nombre y la ubicación del plugin dentro del menú de OsiriX (ImageFilter, ROI, others), el número de versión, etc.

Los archivos .m y .h son la clase principal del plugin. Es ahí donde esta el algoritmo para ejecutar.

El archivo .xcode, contiene el proyecto completo y el .pch, el índice del archivo, no es necesario modificarlo.

El directorio build contiene el plugin compilado.

Es importante que un alias de este plugin esté en la siguiente ruta */library/Application Support/OsiriX/plugins*, debido a que es allí donde OsiriX buscará, al iniciar el procedimiento requerido por el plugin. El plugin puede utilizar todas las funciones que tiene OsiriX.

Los "OsiriX Headers" son siete [3]:

-PluginFilter.h, el plugin es una subclase de este archivo. Aquí se encuentran muchos métodos y variables de instancia básicos para el manejo de la imagen.

-ViewerController.h, es un objeto de la clase NSWindowController, que maneja el despliegue en 2D.

-DCMViewer.h, contiene la imagen desplegada.

-DCMPix.h, es el objeto que maneja los píxeles de la imagen.

-dicomFile.h, contiene la información del archivo DICOM.

-ROI.h, es el objeto donde esta contenida una región de interés (ROI) de la imagen seleccionada.

-MyPoint.h, es el objeto que describe un punto en 2D.

Fig. 1. Directorio del plugin "Diferenciación", mostrando los archivos y directorios que utiliza.

III. Metodología

Un problema de importancia fundamental en el análisis de imágenes es la detección de bordes. En el LINI se ha trabajado bastante en el tema; ahora se plantea la necesidad de llevar el algoritmo implementado en un ambiente de trabajo que pueda ser utilizado en un entorno clínico.

Los contornos caracterizan las fronteras de los objetos, y por tanto son de gran utilidad para segmentar e identificar límites de objetos.

Los puntos de contorno son las zonas de píxeles en las que existe un cambio abrupto de nivel de gris. Si pensamos en una imagen como una función continua f(x,y), vemos que su derivada tiene un máximo local en la dirección del contorno.

Es por esto que las técnicas más usadas en la detección de contornos se basen en la medida del gradiente. [2]

La estimación del gradiente de una imagen, se lleva a cabo mediante un filtrado espacial empleando una máscara de diferenciación, W, especificada en una ventana de (m x m) definiendo un subespacio de gradiente en $R^{m \times m}$ [4].

El gradiente en un punto de la imagen se obtiene, entonces, mediante el producto interior de una ventana (m x m) de datos de la imagen por dos máscaras de diferenciación en las dos direcciones perpendiculares (x, y) de la cuadrícula de la imagen.

El gradiente en x, se obtiene por diferenciación de renglones y suavizado de columnas simultáneo. Esto se hace a través del producto exterior de dos secuencias unidimensionales s(i) y d(i), i = -m, ...0,...m.

La máscara en x, será:

$$W_{dx} = sd^T$$

Donde s y d son secuencias de suavizado y diferenciación respectivamente definidos por los polinomios de Krawtchouk que producen filtros para datos con pesos binomiales [7] obtenidos de:

$$w(i) = \frac{1}{2^{2m}}\binom{2m}{m+i} = \frac{1}{2^{2m}}\frac{(2m)!}{(m-i)!(m+i)!}$$

para $i = -m, \ldots 0, \ldots m$.

Con estos pesos se calcularan las secuencias de la siguiente manera:

$$s(i) = w(i)$$
$$d(i) = \frac{2i}{m} w(i)$$

El gradiente en y, se obtiene diferenciando en columnas y suavizando los renglones, por lo tanto la máscara en y será la transpuesta de la máscara en x [1].

$$W_{dy} = W_{dx}{}^T = ds^T$$

Con estas máscaras de diferenciación numérica se obtienen las derivadas en x y en y para cada píxel de la imagen:

$$G_x = W_{dx} A$$
$$G_y = W_{dy} A$$

Donde A es una ventana de datos de m x m centrada en el píxel.
Luego, el módulo del gradiente estimado en el píxel es:

$$G = \sqrt{G_x^2 + G_y^2}$$

que corresponde al gradiente de la imagen.

"Diferenciación", es el nombre del nuevo plugin que calculará los bordes de la imagen a través del gradiente. Ver Fig.2.

En la implementación se utilizó Xcode, se hizo la liga a OsiriX (en la carpeta de "Executables") y se han incluido todos los archivos necesarios para que OsiriX interprete que el nuevo plugin debe correr dentro de su ambiente.

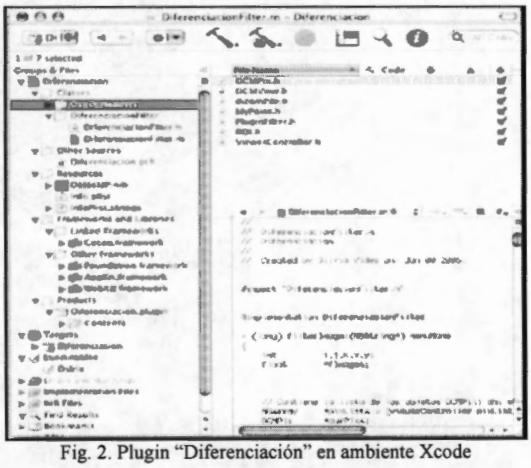


Fig. 2. Plugin "Diferenciación" en ambiente Xcode

En el archivo DiferenciacionFilter.h se declara la interface de la clase "Diferenciación", es decir, se definen las variables de instancia y los métodos que van a utilizarse y en el archivo DiferenciacionFilter.m se realiza la implementación de los métodos declarados, es decir, aquí escribimos el cuerpo del plugin.

En esta etapa se utilizaron algunas clases definidas por Cocoa y las facilidades del lenguaje orientado a objetos Objective C para codificar las fórmulas y los procedimientos para el cálculo del gradiente. Se utilizó DCMPix, para el manejo de los píxeles de la imagen, y se utilizó PluginFilter como superclase de este nuevo plugin. Varios de los métodos ya definidos en estos objetos de OsiriX ayudan en la adecuada manipulación de la imagen y en los cálculos pertinentes.

Xcode construye el plugin, pero el éxito total de la implementación se observará directamente en el tratamiento que el plugin hará de la imagen.

IV. Resultados

El cálculo del gradiente puede corroborarse, por ejemplo, utilizando como modelo la imagen de un fantoma el cual esta compuesto de dos esferas concéntricas, ya que el resultado correcto de la diferenciación debe ser dos círculos concéntricos. Vemos en la Fig. 3 que el plugin "Diferenciación" esta dentro del ambiente de OsiriX y se eligió que estuviera en el submenú de "Image Filters" por su naturaleza. En la misma figura se muestra la imagen original a la que se le extraerán los bordes.

El resultado de esta "Diferenciación" se observa en la Fig.4. Se ha obtenido la imagen de 2 círculos. Se puede apreciar que los valores de intensidad del borde del círculo exterior son mayores que los del círculo interior y esto es lo esperado, dado que las diferencias de contraste en la imagen original, son mayores entre el fondo y la esfera exterior.

Fig. 3. Plugin "Diferenciación" dentro del menú de OsiriX e imagen original

Se utilizó el plugin "Diferenciación" en otro tipo de imágenes, por ejemplo, en la detección de bordes en imágenes cerebrales. Se procesó un estudio de RM de un sujeto sano, proporcionado por el Hospital General de Massachussets y disponible en el sitio IBSR (Internet Brain Segmentation Repository) de internet. El estudio son imágenes T1 en corte coronal de dimensión 256 x 63 x 256 y con voxel de tamaño 1 x 3 x 1.

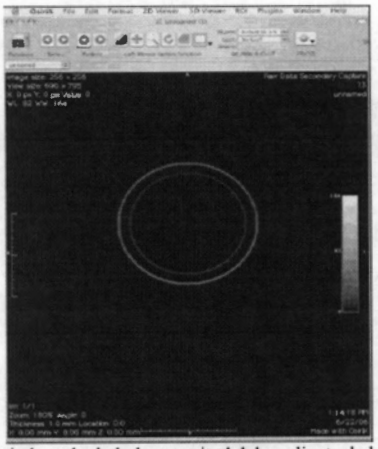


Fig. 4. El plugin ha calculado la magnitud del gradiente de la imagen y se observan solamente dos bordes que corresponden a las esferas del fantoma.

El plugin "Diferenciación" permite la lectura completa del volumen y extrae los bordes de cada una de las 63 imágenes del estudio. La Fig. 5 muestra los resultados obtenidos de una de las imágenes cuando se procesa el volumen, rebanada por rebanada, con el gradiente como detector de bordes.

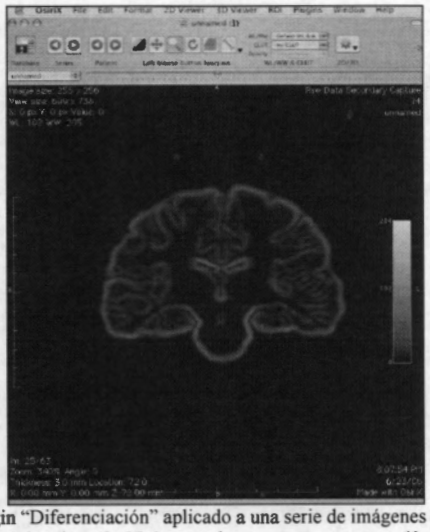


Fig.5. Plugin "Diferenciación" aplicado a una serie de imágenes de cerebro. Observamos la rebanada 25 de un volumen compuesto por 63 rebanadas.

Para comparaciones de rendimiento computacional se ejecutó el algoritmo de detección de bordes por magnitud del gradiente en Matlab [8], en una computadora con procesador dual de 1 GHz, Xeon, y 4 GB de RAM, el tiempo de procesamiento para las imágenes de RM cerebrales antes mencionado, fue 19.3177 segundos, siendo el tiempo para una rebanada significativa de 0.3647 segundos. El tiempo para el mismo estudio, utilizando el plugin de OsiriX en una computadora G5 con procesador dual de 2.3 GHz y 512 MB en RAM, fue 9.804 segundos y el tiempo para la misma rebanada significativa fue de 0.245 segundos.

V. CONCLUSIONES

La filosofía del software de código abierto permite no solo acceder, sino también modificar y adaptar el software a las necesidades particulares en diferentes líneas de investigación.

OsiriX, ofrece grandes ventajas en el desarrollo de herramientas para investigación en el área de imágenes médicas. Un ejemplo de esto es el manejo de gran variedad de formatos de imágenes, lo que facilita la implementación de cualquier plugin, sin preocuparse por la lectura de estos diferentes formatos.

Xcode y Cocoa ofrecen facilidades para implementar algoritmos ya desarrollados en laboratorios de investigación. Esto es muy importante, ya que hay programas que son el resultado de mucha investigación, que sin embargo no se utilizan fuera del laboratorio de desarrollo y con este tipo de software de código abierto, es posible llevarlos a un nivel de utilidad mayor en la investigación o incluso fuera del ambiente universitario, todo esto fundamentado en uno de los principios básicos del software libre, que es la libre distribución y utilización del software.

La arquitectura extensible de OsiriX (y de otros programas de código abierto) a través de plugins, fortalece en gran medida la continuidad y la comunicación de resultados entre desarrolladores. La utilización de un lenguaje orientado a objetos y de clases de Cocoa, permite la modularidad en la construcción de un proyecto y la fácil comprensión para futuros programadores.

El ejemplo de programación anteriormente presentado, muestra cómo es posible adaptar desarrollos de software propios, a un sistema extensible de código abierto para hacer accesible nuevos desarrollos en el área de la segmentación y el procesamiento de imágenes al campo clínico. La unión entre desarrollos en los centros de investigación y la estación de visualización basada en OsiriX permitirá a muchos hospitales iniciar una migración de sus sistemas de radiografía convencional a sistemas de almacenamiento y comunicación de imágenes sin tener que depender exclusivamente de un proveedor que los convierta en clientes cautivos.

Referencias

[1] J.R. Jiménez Alaniz , "Segmentación No Paramétrica de Imágenes Cerebrales por Resonancia Magnética", Tesis de Doctorado en Ingeniería Biomédica, Universidad Autónoma Metropolitana, México D.F., 2006.

[2] J. Azpíroz, V. Medina, J.F. Lerallut, "Procesamiento de imágenes biomédicas", Universidad Autónoma Metropolitana, 2000, cap. 5, pp.176-183.

[3] Rosset, A. "OsiriX Plug-ins Developer Toolkit" , version 1.0, UCLA, California, 2004.

[4] Jiménez, J.R., Medina, V., Yáñez O.; "Nonparametric MRI Segmentation using mean shift and edge confidence maps", Medical Imaging 2003: Image Processing, Milan Sonka, J. Michel Fitzpatrick, Editors, Proc. of SPIE, vol. 5032, pp. 1433-1441, 2003.

[5] A. Hillegrass, "Cocoa Programming for Mac OS X", 2nd. Edition, 2004.

[6] S. Kochan, "Programming in Objective C", 1st. Edition, 2004.

[7] P. Meer, B. Georgescu, "Edge detection with embeded confidence", IEEE Transactions Pattern Análisis Machine Intell., vol. 23, No. 12, pp. 1351-1366, 2001.

[8] http://www.mathworks.com/products/matlab

VIII Simposio de Cirugía Asistida por Computadora
y Procesamiento de Imágenes Médicas
UAM-Iztapalapa
Septiembre de 2007

SEGMENTACIÓN POR CORRIMIENTO DE MEDIA PONDERADO CON MAPAS DE CONFIANZA DE BORDES, COMO PLUGIN PARA OSIRIX

Silvia Vides Cañas, Joaquín Azpíroz Leehan, Juan Ramón Jiménez Alaniz.

Universidad Autónoma Metropolitana, UAM-Iztapalapa

I. INTRODUCCIÓN

El desarrollo de software de código abierto (Open Source Software, OSS) para aplicaciones en imagenología médica permite la colaboración de individuos y grupos para producir herramientas de alta calidad que satisfagan las necesidades del usuario. En nuestro caso particular, OSS permite la implementación de algoritmos y procedimientos desarrollados en universidades, centros de investigación y hospitales. Este proceso puede llevarse a cabo utilizando OsiriX, un programa de visualización de imágenes en formato DICOM de Apple Macintosh. OsiriX ofrece funciones para manipulación de imágenes tal como son la proyección multiplanar, filtros de convolución, ajuste de grosor de rebanadas, reconstrucción volumétrica, proyección de intensidad máxima y mínima, y reconstrucción de superficies [5], [6]. Una ventaja importante es que OsiriX es extensible con módulos o plugins. Éstos son unidades de extensión binarias para una aplicación cuya arquitectura permite a usuarios finales añadir funcionalidad en sitios definidos una vez que la aplicación ya ha sido desarrollada, a esto se le conoce como extensibilidad tardía. Los Plugins son opcionales y la aplicación a la que extienden es funcional aún si no hay plugins instalados. Sin embargo, ellos si dependen de la aplicación a la que extienden y no pueden funcionar sin ella. Como se mencionó anteriormente, OsiriX tiene una arquitectura basada en plugins. OsiriX se distribuye como software de código abierto bajo la Licencia General Pública de GNU (GPL de GNU) [2] y puede descargarse de la página web de OsiriX [10].

Este trabajo es sobre la implementación del método de segmentación conocido como Mean Shift como un plugin para OsiriX. Este método de estimación noparamétrica de densidad fue reportado como una técnica de segmentación muy robusta para imágenes por resonancia magnética de cerebro [1]. Para desarrollar el plugin son necesarias herramientas orientadas a objetos [3] para el modelado. Con este propósito se presentan diagramas UML. El artículo esta organizado de la siguiente forma: las herramientas para el modelado del proyecto y para la construcción del plugin se mencionan en la Sección II. Los diagramas UML y los resultados del plugin implementado en MRI reales se presentan en la Sección III, y finalmente, algunas conclusiones en la Sección IV.

II. METODOLOGÍA

A. Herramientas en OS X

Las herramientas en OS X para desarrollar un plugin en OsiriX son:

Xcode es el ambiente de desarrollo integrado (Integrated Development Environment, IDE). Xcode reúne todos los recursos para editar, compilar, depurar y ejecutar la aplicación. Xcode utiliza el compilador para C de GNU (GCC) y también el depurador de GNU (GDB). Para la construcción de interfaces gráficas o GUIs, Xcode trabaja con Interface Builder, otra herramienta de desarrollo en OS X.

Objective-C es un lenguaje dinámico y orientado a objetos, basado en un lenguaje llamado SmallTalk-80. Fue mejorado añadiendo extensiones del lenguaje C para crear un nuevo lenguaje de programación que permite crear y manipular objetos. Objective C, ahora es parte de la Fundación para el Software Libre (Free Software Foundation's GNU) por lo tanto es de dominio público. [8]

Cocoa, es un marco de trabajo o "framework" es decir, es

una colección de clases que están previstas para ser usadas juntas. Esto significa que las clases son compiladas juntas en una librería reusable de código. Todas las aplicaciones de Cocoa usan dos de estos frameworks:

- *Foundation Kit*: Todos los lenguajes de programación necesitan clases para manejar valores estándar. Elementos como cadenas, datos, listas y timers se encuentran aquí.
- *Application Kit* o *AppKit*: Todo lo relacionado con la interfaz de usuario esta contenido en este framework. Esto incluye clases para ventanas, botones, campos de texto, eventos y dibujos. [4].

B. Algoritmo de Mean Shift

Según [1] la segmentación de imágenes cerebrales por resonancia magnética es efectuada aplicando un estimador de densidad noparamétrico, esto, usando el algoritmo de Corrimiento de Media (CM) o Mean Shift (MS). La calidad de las fronteras entre clases es mejorada al incluir un mapa de confianza de bordes, el cual representa la confianza de estar ante la presencia de un borde entre regiones adyacentes. Un grafo de adyacencia se construye con las regiones etiquetadas, que luego se analizan y podan para poder fusionar las regiones.

Uno de los métodos más utilizados para calcular bordes está basado en la orientación del gradiente, y los pasos para su cálculo son la estimación de la magnitud y orientación del mismo. La estimación del gradiente se lleva a cabo utilizando máscaras de diferenciación, **W**, definida en una ventana de m x m píxeles, y que define un subespacio del gradiente en R^{mxm}. Este proceso también permite calcular la orientación del gradiente y puede ser usado para definir una plantilla de borde ideal, t, con la misma estimación de la orientación del gradiente. La medida de confianza de encontrarse ante la presencia de un borde puede definirse como $\eta = |t^T a|$, lo que corresponde, en el domino de las imágenes, al coeficiente de correlación entre las plantillas de borde ideal y la ventana de datos.

El mapa de confianza de bordes φ es calculado para cada voxel como la combinación lineal de la función de distribución acumulada empírica de la magnitud del gradiente ρ y la medida de confianza, η

$$\varphi = \rho\beta + (1-\beta)\eta \tag{1}$$

donde β es una constante entre 0 y 1 que controla la mezcla de la magnitud del gradiente ρ y la información del patrón local de η.

El procedimiento de estimación noparamétrica que se basa en el algoritmo de MS usa las modas locales de la función de densidad del espacio conjunto espacio-rango para definir los centros de los grupos o "clusters". Este algoritmo ha probado ser robusto ante diferentes niveles y tipos de ruido y no asume ninguna forma para la función de densidad de probabilidad. La calidad de las fronteras entre clases es mejorada pesando cada voxel de la región por una función de confianza de bordes, por lo tanto, los voxeles que estén cerca de un borde (mapa de confianza ≈1) tienen menor influencia en la determinación de un nuevo centro de cluster. La ecuación de MS que incluye la ponderación por Mapa de Confianza es

$$M_h(x) = \frac{1}{\sum(1-\varphi_i)} \sum_{x_i \in S_h(x)} (1-\varphi_i) X_i - x \tag{2}$$

donde $S_h(x)$ es una hiperesfera de radio h, Xi es el dato que va a ser incorporado al cálculo de $M_h(x)$.

La medida de confianza también es utilizada para la fusión de regiones que tienen bordes débiles entre si, a través de la aplicación iterativa de la cerradura transitiva en una matriz de adyacencia de regiones (MAR). El paso de la fusión se completa con un proceso de podado, el cual elimina regiones muy pequeñas con respecto a un parámetro dado por el usuario.

C. Proceso de Desarrollo de Software

Para transformar requerimientos de usuario en programas de software es necesario establecer un orden de actividades, es decir, utilizar Proceso de Desarrollo de Software (Software Development Process, SDP). Los "SDPs" orientados a objetos son considerados como centrados en la arquitectura, iterativos incrementales y guiados por casos de uso. En caso de los plugins, la arquitectura ya esta definida por la aplicación a la que extienden. Las arquitecturas basadas en plugins definen lugares y condiciones específicas para incorporar los plugins. El núcleo de la aplicación principal controla el ciclo de vida general y se comunica con el plugin por interfaces específicas, por lo tanto, la arquitectura no era una actividad a realizar en este proyecto en particular. El progreso es iterativo e incremental lo que se refiere al desarrollo de productos de software en pasos pequeños y manejables. Entre cada etapa, hay retroalimentación que permite reajustar el foco para la siguiente fase. Cuando todos los pasos planeados han sido bien alcanzados; el producto esta completo y listo. Cuando un SDP es guiado por casos de uso, cada requerimiento debe ser expresado como un caso de uso y luego todos deben ser validados como requisitos del sistema. Los productos se prueban para garantizar que satisfagan los casos de uso [7].

D. Diagramas UML

Un modelo ayuda a comprender mejor el sistema que estamos desarrollando. Los modelos ayudan a visualizar un sistema tal como es o como quisiéramos que fuera. Permiten especificar la estructura y el comportamiento del sistema y proporcionan un plano que orienta en la construcción y documenta las decisiones que se han tomado [9]. Si se aplica SDP y se hace un modelo los resultados sirven para visualizar el diseño y verificarlo contra los requerimientos antes de que sea generado código [7].
El Lenguaje de Modelado Unificado (Unified Modeling Language, UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software completo o alguna de sus partes [9].
A través de UML es posible capturar la información sobre clasificación de las estructuras, comportamiento dinámico y manejo del modelo de un sistema de software [7]. Cada diagrama de UML permite a los desarrolladores y a los usuarios finales visualizar el sistema desde diferentes puntos de vista y niveles de abstracción.
En el desarrollo de este proyecto en particular se utilizaron cuatro diagramas UML, tres de comportamiento (diagramas de casos de uso, colaboración y secuencia) y uno estructural (diagrama de clases).
Diagrama de Casos de Uso Es la descripción de una secuencia de acciones e inclusive las variaciones que el sistema lleva a cabo para entregar un resultado de valor para el usuario. Este diagrama captura la funcionalidad del sistema.

Diagramas de Interacción son de dos tipos:

Diagramas de Colaboración muestra la interacción y la colaboración entre objetos. En este diagrama no se incluye el tiempo como dimensión, sino que las relaciones son etiquetadas con números para mostrar secuencia. Enfatiza la organización estructural de los objetos que envían y reciben mensajes.
Diagramas de Secuencia representa el orden temporal o de secuencia de los mensajes entre los objetos. Tienen una dimensión vertical (tiempo) y una dimensión horizontal (los objetos). Estos diagramas ayudan a definir los métodos necesarios para la implementación. Enfatizan el orden temporal de los mensajes. Un diagrama de secuencia muestra una serie de objetos y mensajes entre ellos, por lo tanto representan también la visión dinámica del sistema.
Diagramas de Clases son diagramas estructurales que contienen clases, interfaces, colaboraciones y las relaciones entre ellos (dependencia, generalización y asociación). Dentro de los diagramas de clases se muestran métodos y atributos de cada clase e incluso pueden tenerse paquetes de subsistemas. Un diagrama de clases ilustra el diseño estático de un sistema [9].

E. Construcción de un plugin en OsiriX

Como se mencionó anteriormente, OsiriX puede ampliarse con la adición de unidades de extensión o plugins. Estos plugins son opcionales y tienen un lugar determinado dentro de la barra del menú de OsiriX. El plugin es un proyecto en Xcode que está vinculado a la librería de clases, Cocoa y que necesita algunos archivos importantes para poder formar parte del ambiente de OsiriX.
Para la implementación del modelo de Segmentación por Corrimiento de Media como un plugin de OsiriX es necesario crear un directorio con los siguientes documentos y subdirectorios:

- Todos los archivos .m y .h que contienen el código fuente de cada clase definida en el diagrama de clases.
- Version.plist e Info.plist son archivos que contienen información acerca del plugin: nombre, versión y el sitio exacto donde va aparecer dentro del menú reservado para Plugins en la barra de opciones de OsiriX (ImageFilter, ROI, others), etc.
- English.lproj contiene información sobre el idioma del plugin.
- xcodeproj es el proyecto completo en Xcode.
- pch contiene el archivo prefijo.
- OsiriX Headers es un subdirectorio que contiene los objetos que OsiriX utiliza para comunicarse con el plugin. Los OsiriX Headers son siete:
    - PluginFilter: el plugin es una subclase de este objeto. Contiene métodos y atributos de gran utilidad para el procesamiento de imágenes.
    - ViewerController: es un objeto del tipo NSWindowController y controla el despliegue de la ventana de visualización en 2D.
    - DCMView: es un objeto del tipo NSOpenGLView que maneja la imagen desplegada.
    - DCMPix: es un objeto que maneja los píxeles de la imagen.
    - dicomFile: contiene los datos respecto al archivo DICOM.
    - ROI: es un objeto que maneja la información sobre una ROI.
    - MyPoint: es un objeto que maneja la información de un punto en 2D.
- El directorio 'Build' contendrá al plugin compilado.

Para que el proyecto pueda funcionar como plugin, es necesario definir a OsiriX como ejecutable del proyecto en Xcode. También es muy importante generar un "alias" del plugin compilado dentro de la carpeta de Plugins localizada en *\Library\ApplicationSupport\OsiriX\Plugins,* porque cuando OsiriX inicie, buscará en ese directorio aquellos archivos que tengan la extensión '.plugin' y que sean una subclase del objeto 'PluginFilter'. Luego identifica si hay un método llamado 'filterimage', ya que éste método determina la clase principal del proyecto y por lo tanto, el inicio del plugin [3].

III. RESULTADOS

Las figuras 1 y 2 muestran dos de los diagramas UML utilizados para modelar el procedimiento de Segmentación por Corrimiento de Media ponderado con Mapas de Confianza. El diagrama de Casos de Uso (Fig. 1) muestra la interacción de los actores (el usuario y la aplicación, OsiriX) con el plugin.

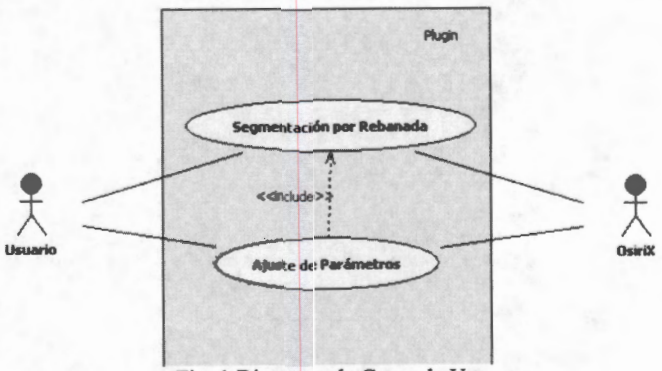


Fig. 1 Diagrama de Casos de Uso

Los casos de uso para este plugin son dos: La Segmentación por Corrimiento de Media completa que calcula todo el procedimiento para una o varias rebanadas, y la segmentación por Ajuste de Parámetros que sirve para elegir el mejor conjunto de parámetros para un tipo de dato, antes de se aplique a un estudio completo.
Las clases para cada caso de uso fueron definidas después de las etapas de análisis y diseño. El diagrama de colaboración en la Fig. 2 representa la relación o interacción colaborativa entre clases. Primero, el usuario inicia la aplicación y carga los datos porque si los datos no están en el ambiente de OsiriX el menú de plugins esta deshabilitado. Una vez que el plugin "Segmentación" es seleccionado, el proceso inicia a través de la clase principal llamada SegmentationFilter que maneja el flujo de datos y mensajes y solicita los parámetros al usuario. La clase SegmentationFilter controla que cada etapa del procedimiento de segmentación se lleve a cabo. Tal como lo sugiere la metodología del corrimiento de media, el mapa de confianza de la imagen se calcula primero. La clase Mapa de Confianza necesita datos de la clase de Plantillas la cual tiene una relación de colaboración exclusiva con esta clase. Cuando el mapa ha sido calculado, la clase de Filtrado por CM solicita los parámetros, la rebanada que esta siendo procesada y el mapa de confianza, para calcular el CM. El resultado es una imagen con las modas que convergieron hacia los valores calculados.
Para la clase de Fusión, son necesarias la imagen filtrada y el mapa de confianza. En la clase de Fusión la imagen filtrada es etiquetada para identificar el número de regiones que se buscará fusionar. Para fusionar se utilizan dos criterios, el primero, de adyacencia por lo que es necesario generar una Matriz de Adyacencia de Regiones (MAR) y el segundo, de intensidad de los bordes entre regiones (mapa de confianza); con esto, la fusión se realiza siempre y cuando sea posible. La fusión se hace en forma iterativa, cuando concluye, se hace un podado para poder fusionar pequeñas regiones. Cada región es comparada con un número mínimo de píxeles, y si la región esta bajo este umbral, se une a la región más conveniente. La clase Segmenta sustituye los píxeles de las regiones finales con las modas calculadas. Cuando esto ocurre, la clase Despliegue muestra el resultado.
Para comprobar la implementación del plugin, se utilizaron imágenes reales. Las imágenes fueron obtenidas del sitio de Internet Brain Segmentation Repository (IBSR) [11]. El estudio es el identificado como 1_24 y consiste en datos de un cerebro normal proporcionado por El Hospital General de Massachusetts.

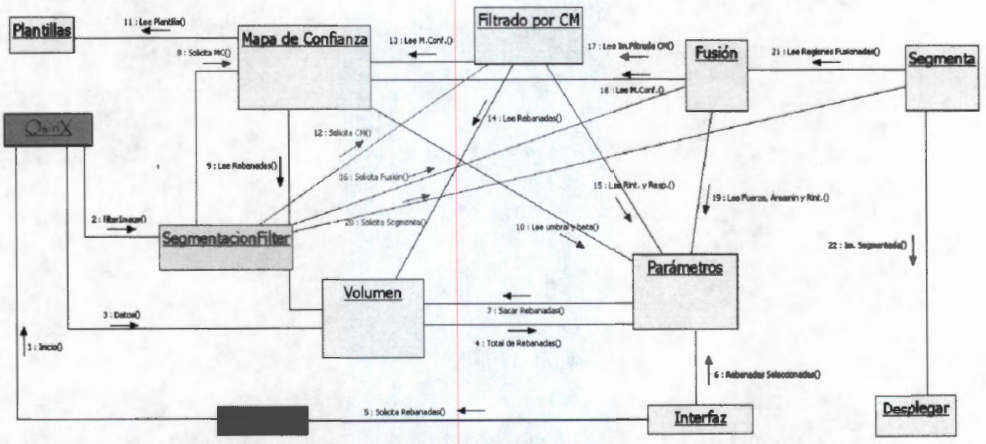


Fig. 2 Diagrama de colaboración del caso de uso de Segmentación por Rebanada.

Los datos consisten en imágenes por resonancia magnética en modalidad T1 de tamaño 256 x 63 x 256 y con una resolución de voxel de 1 x 3 x 1 mm. Los parámetros utilizados son los sugeridos por [1] para este tipo de dato. Los resultados del plugin se muestran en la Fig. 3. Después de que el plugin es aplicado, todas las herramientas de OsiriX como son reconstrucción 3D, mejoras en la visualización o cambio de formato pueden utilizarse.

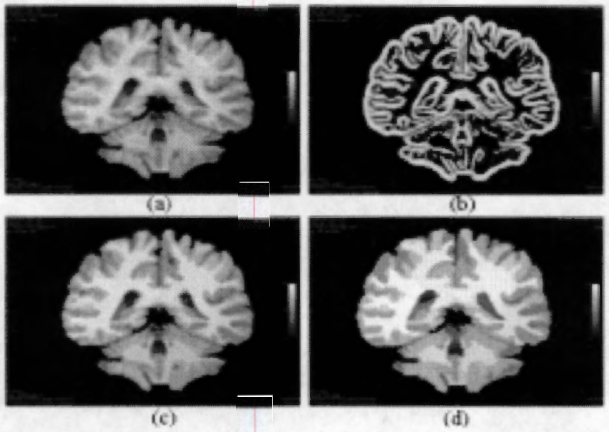


Fig. 3 muestra resultados de las clases implementadas en el plugin. (a) imagen original; (b) el mapa de confianza correspondiente; (c) la imagen filtrada por CM; (d) resultado final después de fusión y podado.

IV. CONCLUSIONES

A través de SDP y diagramas de UML se ha modelado la metodología de la Segmentación por Corrimiento de Media ponderada por Mapas de Confianza, independientemente del lenguaje de codificación. La implementación como un plugin de OsiriX requiere modelado orientado a objetos para facilitar la comprensión y la solución del problema, asimismo facilita futuras extensiones del plugin desarrollado.

Las arquitecturas basadas en plugins ofrecen muchas posibilidades para la comunicación y utilización en aplicaciones de la vida real de diferentes algoritmos desarrollados en universidades y centros de investigación. OsiriX ofrece muchas ventajas para el desarrollo de herramientas en el área de investigación de imágenes médicas; por ejemplo, OsiriX tiene la capacidad de manejar estudios con grandes cantidades de imágenes y con diferentes formatos, por lo que se simplifica la extensión de opciones de procesamiento ya que el problema de la lectura de datos esta solucionado en gran medida.

El ejemplo de aplicación que se presenta, muestra como es posible adaptar los desarrollos internos de una institución al ambiente clínico. También hace posible la colaboración interinstitucional para ayudar a los hospitales en la migración de estudios hechos en película convencional al uso de PACS (Picture Archiving and Communication System) sin la necesidad de depender de un proveedor o marca específica que puede convertir al hospital en un cliente cautivo.

REFERENCIAS

[1] J. R. Jiménez-Alaniz, V. Medina-Bañuelos, Oscar Yáñez-Suárez: "Data-driven brain MRI segmentation supported on edge confidence and a priori tissue information". IEEE Trans. Med. Imaging 25(1): 74-83 (2006).
[2] GNU Public Licence, www.gnu.org/copyleft/gpl.html
[3] Plugin Developers Toolkit, Osirix Documentation, OsiriX Wiki, en.wikibooks.org/wiki/Online
[4] A. Hillegass, Cocoa Programming for Mac OS X, Addison Wesley, 2nd Ed., 2004.
[5] A. Rosset M.D., L. Pysher M.D., L. Spadola M.D., O. Ratib M.D., "OSIRIX: open source multimodality image navigation software", in Proc. SPIE Conf. Medical Imaging 2005: PACS and Imaging Informatics; Osman M. Ratib, Steven C. Horii; Eds., vol. 5748, pp. 20-27 (2005).
[6] A. Rosset, L. Spadola, O. Ratib: OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images, Journal of Digital Imaging, vol 17(3), 2004.
[7] A. Martinez-Martinez, J.R. Jimenez-Alaniz, A. Gonzalez-Marquez, N. Chavez-Avelar: "DICOM static and dynamic representation through unified modeling language", in Proc. SPIE Conf. Medical Imaging 2004: PACS and Imaging Informatics; Osman M. Ratib, H. K. Huang; Eds., vol. 5371, pp. 79-89 (2004).
[8] S. Kochan, Programming in Obective C, Sams Publishing, 2004.
[9] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Object Technology, Addison Wesley, 1999. pag. 6, 13, 25, 97 and 107.
[10] OsiriX homepage, homepage.mac.com/rossetantoine/osirix/Index2.html
[11] Internet Brain Segmentation Repository (IBSR); http://neuro-www.mgh.harvard.edu/cma/ibsr/

Proceedings of the 29th Annual International
Conference of the IEEE EMBS
Cité Internationale, Lyon, France
August 23-26, 2007.

Plugin for OsiriX: Mean Shift Segmentation

Vides Cañas S., Azpíroz Leehan J. *, *Senior Member, IEEE*, and Jiménez Alaniz J.R., *Member, IEEE*

Abstract— **Open source and plugin-based software provide the possibilities for a flexible and free transportation of algorithms to final users. In medical image processing, OsiriX is an instance of the fulfillment of this potential. This work presents the plugin implementation of the mean shift segmentation algorithm. It presents the modeling of the methodology using Object-Oriented tools, as a part of software development process. The resultant models have been represented through Unified Modeling Language (UML). The model parts have been implemented and proved as an OsiriX plugin, helping bridge the divide between algorithms devised in research laboratories and the application of these tools in the clinical setting.**

I. Introduction

OPEN source software (OSS) development for medical imaging enables collaboration of individuals and groups to produce high-quality tools that meet user needs. In our particular case, OSS allows the implementation of algorithms and procedures developed at universities, research centers and hospitals. This process is reviewed and illustrated with OsiriX, a fast DICOM viewer program for the Apple Macintosh. The OsiriX program offers image manipulation functions such as multiplanar projection, convolution filters, variable slice thickness adjustments, volume rendering, minimum and maximum intensity projections, and surface rendering [6]. An important advantage is that OsiriX is extensible with modules or plugins. They are binary extension units for an application whose architecture allows functionalities to be introduced by end-users at well-defined places once the application has been deployed. Plugins are optional as the application they extend is functional even if no plugins are present. However, they depend on the application they extend and cannot function without it. As mentioned before, OsiriX has a plugin-based architecture. OsiriX is distributed as open source software under the GPL [2] and can be downloaded at the OsiriX web site [10].

This work is about the implementation of the Mean Shift segmentation method as an OsiriX plugin. This nonparametric density estimation was reported as a robust segmentation technique in magnetic resonance brain images [1]. To develop the plugin, object-oriented tools [3] for modeling are necessary. For this purpose UML diagrams are presented. The paper is organized as follows: Tools for modeling the project and constructing the plugin are mentioned in Section II. UML diagrams and results of the implemented plugin in real MRI data are presented in Section III, and finally, some conclusions in Section IV.

Asterisk indicates corresponding author

S. Vides Cañas is with the CIIIM and the Neuroimaging Laboratory, Department of Electrical Engineering (e-mail: silvia.vides@gmail.com).

* J. Azpíroz Leehan is with the CIIIM, Department of Electrical Engineering. (e-mail: jazp@xanum.uam.mx).

J. R. Jiménez-Alaniz is with the Neuroimaging Laboratory, Department of Electrical Engineering. (e-mail: jajr@xanum.uam.mx).

Universidad Autónoma Metropolitana—Iztapalapa, Av. San Rafael Atlixco 186, Col. Vicentina, México, D. F., 09340, México

II. Methods

A. OS X Tools

The OS X tools to develop an OsiriX plugin are:

Xcode is the integrated development environment (IDE). Xcode tracks all the resources to edit, compile, debug and launch the application. Xcode uses GNU C compiler (GCC) and de GNU debugger (GDB). For the construction of GUIs, Xcode works with Interface Builder, another OS X developer tool.

Objective-C is a dynamic and object-oriented language, based on a language called SmallTalk-80. It was layered on top of the C language, meaning extensions were added to C to create a new programming language that enables *objects* to be created and manipulated. Objective C support was added to the Free Software Foundation's GNU development environment, i.e., it is in the public domain. [8]

Cocoa is a framework or a collection of classes that are intended to be used together. That is, the classes are compiled together into a reusable library of code. All Cocoa applications use two frameworks:

- *Foundation Kit*: Every object-oriented programming language needs the standard values, collection and utility classes. Things like strings, dates, lists, threads, and timers are here.
- *Application Kit or AppKit*: All things related to the user interface are in this framework. These include windows, buttons, text fields, events, and drawing classes. [4].

B. Mean Shift

As suggested by [1] the segmentation of brain magnetic resonance images is carried out by applying a nonparametric density estimation, using the Mean Shift (MS) algorithm. The quality of the class boundaries is improved by including an edge confidence map, which represents the confidence of truly being in the presence of a border between adjacent regions; an adjacency graph is then constructed with the labeled regions, and analyzed and pruned to merge adjacent regions.

The most frequently used edge detection methods are based on gradient orientation, and the steps for this calculation are the estimation of gradient magnitude and orientation. Gradient estimation is carried out using a differentiation mask, **W**, defined in an m x m window, and defining a gradient subspace in $R^{m \times m}$. This process permits to

calculate the orientation of gradient too, and it can be used to define an ideal edge template, t, with the same estimated gradient orientation. A measure of confidence for the presence of and edge in the analyzed data can be defined as $\eta = |t^T a|$, which corresponds in the image domain, to the correlation coefficient between the normalized template and the data.

The edge confidence map φ is computed for each voxel as a linear combination of the cumulative distribution function of the gradient magnitude ρ and the measure of edge confidence η

$$\varphi = \rho\beta + (1-\beta)\eta \quad (1)$$

where β is a constant between 0 and 1 that controls the blending of gradient magnitude ρ and the local pattern η information.

Nonparametric estimation strategy based on the MS algorithm, which use the local modes of the underlying joint space-range density function to define the cluster centers. This algorithm has proven to be robust under the presence of different levels and types of noise and doesn't assume a data probability density function form. The quality of the class boundaries is improved by weighting each voxel within the region by a function of its edge confidence, so that voxels that lie close to an edge (edge confidence ≈1) are less influential in the determination of the new cluster center. The MS that includes weighted edge confidence is

$$M_h(x) = \frac{1}{\sum(1-\varphi_i)} \sum_{x_i \in S_h(x)} (1-\varphi_i) X_i - x \quad (2)$$

where $S_h(x)$ is a hyper sphere of radius h, Xi is the data point that is been incorporated to $M_h(x)$ calculation.

The confidence measure is also used to merge regions with weak edges, through the iterative application of transitive closure operations in a region adjacency graph (RAG). The fusion step is completed with a pruning process to eliminate very small regions.

C. Software Development Process

In order to transform user requirements into software programs it's necessary to establish some activities, which mean to use a software development process (SDP). The object-oriented SDPs are considered as architecture-centric, iterative and incremental, and use-case driven. In the plugin case, the architecture is already defined by the application (OsiriX). The plugin-based architectures provide well-defined places and conditions for the plugins. The core of the basic application controls de life-cycle and communicates with the plugin in specific interfaces. So, the architecture is not an activity of this particular project. The progress is iterative and incremental and is related to the development of software products in small manageable steps. Between each step, feedback is obtained and permitted to adjust the focus for next stage. When all the planed phases have been finished the complete product is released. When a SDP is driven by use cases, the requirements should be expressed in use cases and validated as true requirements to the system. The products are tested and they have to guarantee that fulfill the use cases [7].

D. UML Diagrams

A model provides a better understanding of the system we are developing. Models help to visualize a system as it is or as we want it to be. They permit to specify the structure or behavior of a system and give a template that guides in constructing, and documenting the decisions we have made [9]. Applying a SDP and modeling the results are the way to visualize the design and verify it against requirements before some code can be generated [7].

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting system software and its parts [9].

Through UML it is possible to capture information about structural classification, dynamic behavior, and model management of a software system [7]. A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). The diagram provides a big picture of a software development project and helps to organize the contributions of each individually. Every UML diagram lets to developers and final users to visualize the system from different views and different abstract levels.

For the development of this Project four UML diagrams were used, three behavioral (Use Case, Collaboration and Sequence diagrams) and one structural (class diagram).

Use Case Diagrams are very important in organizing and modeling the behaviors of the system. A use case illustrates the static view of the behavior of a system or a part of it. It is a description of a set of sequences of actions, including variants that system performs to yield an observable result of value to an actor.

Interaction Diagrams are of two types:

Collaboration Diagrams show the interaction and the collaboration between objects. In this diagram the time is not included as a dimension. Instead, the relations are labeled with numbers to show the sequence. It emphasizes the structural organization of the objects that send and receive messages. A collaboration diagram illustrates the dynamic view of a system.

Sequence Diagrams represent the temporal order or the sequence of messages. They have a vertical dimension (time) and a horizontal one (the objects). This diagram helps to determine the methods needed for implementation. It emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and received by those objects. A sequence diagram illustrates the dynamic view of a system.

Class Diagrams are a structural diagram that contains the classes, interfaces, collaborations and their relationships (dependency, generalization, and association). Within class diagrams the methods and attributes of each class are shown and may also contain package or subsystems. A class diagram illustrates the static design view of a system [9].

E. Construction of a plugin in OsiriX

As mentioned before, OsiriX can be improved by adding extension units or plugins. The plugin developed in this work is placed in the "Plugins" option in the OsiriX menu panel, because all the plugins added to OsiriX must be in this option. The plugin it's an Xcode project linked to Cocoa framework and that needs some important files to be part of the OsiriX environment.

For the implementation of the Mean Shift Segmentation model as an OsiriX plugin it's necessary to create a directory with some specific files and subdirectories:

- All the .m and .h files that contain the source code of each class of the class diagram.
- Version.plist and Info.plist are files that contain information about the plug-in name, icon, version number, the place of plugin in the menu (ImageFilter, ROI, others), etc.
- English.lproj contains localized data if you want to create a multi-language plugin.
- xcodeproj file contains the Xcode project.
- pch file, contains the prefix file.
- OsiriX Headers subdirectory, which contains the objects OsiriX uses for the communication with the plugin. The OsiriX Headers are:
  - PluginFilter: the plug-in is a sub-class of this object; it contains some very useful methods and attributes in image processing.
  - ViewerController: it's an object of type NSWindowController that controls the '2D Viewer' window.
  - DCMView: NSOpenGLView that contains the displayed image.
  - DCMPix: an object that contains the pixel data of an image.
  - dicomFile : an object that contains data about the DICOM file.
  - ROI: an object that contains information about a ROI.
  - MyPoint: an object that describes a 2D point.
- The 'Build' subdirectory will contain the compiled plugin.

It's very important to define that the Executable of this Xcode project will be OsiriX and it's also necessary to create an alias of the compiled plugin in the "OsiriX plugin" directory, located in Application Support' of the 'Library' directory. When OsiriX starts, it will search in the 'plug-in' directory for files with the '.plugin' extension. It will then try to find the plugin that contains a sub-class of the 'PluginFilter' object and if the object has a 'filterimage' method; OsiriX will then run once the 'initPlugin' method and start the plugin [3].

III. Results

Figures 1 & 2 show two of the UML diagrams that were used to model the Mean Shift Segmentation procedure weighted by confidence maps. The Use Case diagram (Fig.1) shows the interaction among actors (the user and the application, OsiriX) and the plugin.

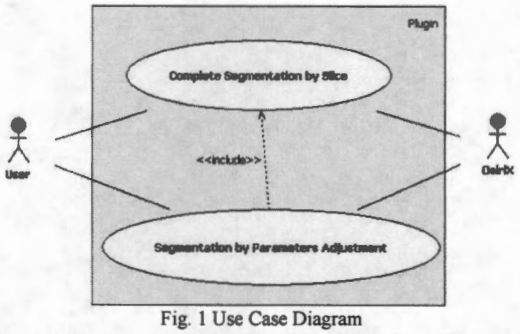


Fig. 1 Use Case Diagram

There are two use cases: the Complete Mean Shift Segmentation that calculates the procedure for a single slice or a complete data set, and the Segmentation by Parameters Adjustment that serves to choose the better parameters combination of the procedure, before the plugin is applied to the complete data set.

The classes for each use case were defined after the analysis and design steps. The collaboration diagram in Fig. 2 represents the relationship and collaborative interaction between these classes. First, the user runs the application and opens the data in OsiriX because if data are not present, the plugin menu is disabled. Once the developed plugin "Segmentation" is selected, the process begins through the main class SegmentationFilter that manages the flow of data and requests the parameters to use from the user. The SegmentationFilter class requests each step of the segmentation procedure to be carried out. Just as it has been suggested by the methodology for the mean shift segmentation, the confidence map of the image is calculated first. The Edge Confidence Map class needs the data from the Templates class, which has a collaboration relationship with the Edge Confidence Map Class. Once the map has been computed, the MS Filtering class that needs the data, the parameters of the process and the confidence maps carries out the mean shift filtering. The result is an image with the modes converging towards the values of calculations. For the Fusion class, the filtered image and the confidence map are needed. In the Fusion class the filtered image is labeled in order to identify the number of regions that will be tried to fuse using the adjacency and edge intensity criteria (confidence maps) between regions; the fusion is carried out whenever possible. After these procedures have been concluded, pruning is carried out to fuse small regions. Each region is compared with a minimum number of pixels, and if the region is under this threshold, it is fused with the most convenient region. The Segmenting class substitutes the final region pixels with the calculated modes. When this happens, the Display class provides the end result. In order to test the plugin implementation, real images were used. This images where obtained from Internet Brain Segmentation Repository (IBSR) [11]. The study is the number 1_24 and consists of a normal brain data set provided by The Massachusetts General Hospital.

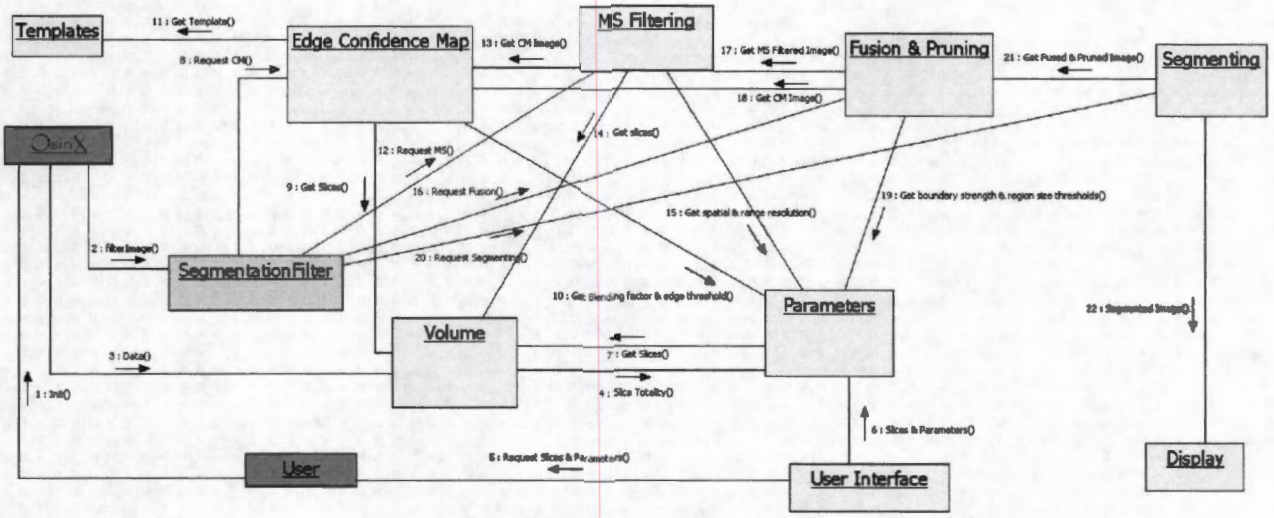


Fig. 2 Collaboration Diagram of the Segmentation by Slice Use Case

The data set consists of T1 weighted images of 256 x 63 x 256 size, 1 x 3 x 1 mm voxel resolution. The processing parameters used are those suggested by [1]. The results of developed plugin are shown in Fig. 3. After the plugin is applied, all tools could be used for 3D reconstruction, enhace, improve visualization or export images.

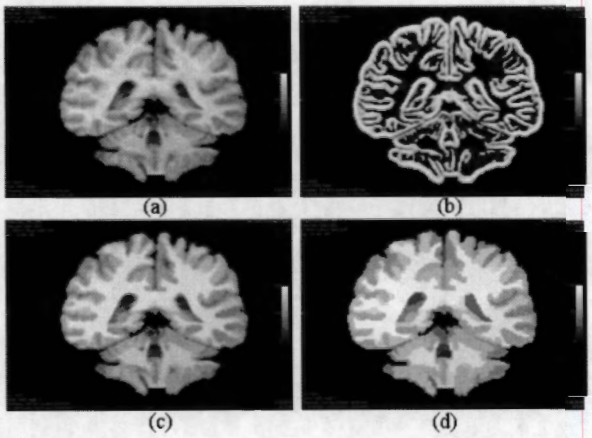


Fig. 3 shows intermediate results of the classes implemented in the plugin. (a) is the original image; (b) presents the corresponding confidence map, (c) contains the mean-shift filtered image; (d) shows the final result of fuson class.

IV. CONCLUSION

Applying a SDP and representing the results through UML has allowed obtaining the model of the Mean Shift Segmentation procedure supported on edge confidence maps. The implementation as an OsiriX plugin required object-oriented modeling to facilitate problem understanding, problem solution and future extensibilities of the developed plugin.

Plugin-based architectures offer many possibilities for the communication and real-life use of different algorithms that have been developed at universities and research centers.

OsiriX offers many advantages for the development of tools in the medical imaging research area. For example, the OsiriX ability to deal with a great number of image formats, simplifies the implementation of plugins that do not need to deal with this issue.

The application example that has been presented shows how it is possible to adapt each institution's internally developed tools to a clinical setting. The interinstitutional collaboration made possible with this tool will help many hospitals to migrate from conventional film imaging to the use of PACS without the need to depend on a single provider that might turn the hospital into captive clients.

REFERENCES

[1] J. R. Jimenez-Alaniz, V. Medina-Bañuelos, Oscar Yáñez-Suárez: "Data-driven brain MRI segmentation supported on edge confidence and a priori tissue information". IEEE Trans. Med. Imaging 25(1): 74-83 (2006).
[2] GNU Public Licence, www.gnu.org/copyleft/gpl.html
[3] Plugin Developers Toolkit, Osirix Documentation, OsiriX Wiki, en.wikibooks.org/wiki/Online
[4] A. Hillegass, Cocoa Programming for Mac OS X, Addison Wesley, 2nd Ed., 2004.
[5] A. Rosset M.D., L. Pysher M.D., L. Spadola M.D., O. Ratib M.D., "OSIRIX: open source multimodality image navigation software", in Proc. SPIE Conf. Medical Imaging 2005: PACS and Imaging Informatics; Osman M. Ratib, Steven C. Horii; Eds., vol. 5748, pp. 20-27 (2005).
[6] A. Rosset, L. Spadola, O. Ratib: OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images, Journal of Digital Imaging, vol 17(3), 2004.
[7] A. Martinez-Martinez, J.R. Jimenez-Alaniz, A. Gonzalez-Marquez, N. Chavez-Avelar: "DICOM static and dynamic representation through unified modeling language", in Proc. SPIE Conf. Medical Imaging 2004: PACS and Imaging Informatics; Osman M. Ratib, H. K. Huang; Eds., vol. 5371, pp. 79-89 (2004).
[8] S. Kochan, Programming in Obective C, Sams Publishing, 2004.
[9] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Object Technology, Addison Wesley, 1999. pag. 6, 13, 25, 97 and 107.
[10] OsiriX homepage, homepage.mac.com/rossetantoine/osirix/Index2.html
[11] Internet Brain Segmentation Repository (IBSR); http://neuro-www.mgh.harvard.edu/cma/ibsr/