Procesamiento y almacenamiento distribuido de imágenes basados en grafos: Aplicación para recuperación de información en imágenes muy grandes

Raúl Alonso Calvo

Facultad de Informática

Universidad Politécnica de Madrid

Tesis Doctoral para la obtención del:

Doctorado en Software y Sistemas

2012

Quiero dedicar esta Tesis Doctoral

a Milagros por estar siempre a mi lado.

A mis padres que me lo han dado todo.

A mis hijos Daniel y Ana que lo son todo.

Agradecimientos

En primer lugar quiere agradecer a mi tutor José Crespo del Arco, por su constante apoyo, esfuerzo y dedicación. Gracias por sus conocimientos, seriedad, responsabilidad, motivación e insistencia sin los cuales no hubiera podido llevar a buen término la presente Tesis Doctoral.

También quiero agradecer los consejos recibidos de otros profesores y compañeros como Victor Maojo y Miguel García Remesal que han influenciado enormemente en mi formación como investigador.

En último lugar, quiero agradecer a mi mujer Milagros, a mi familia y amigos, por sus ánimos y ayuda.

Resumen

El almacenamiento y tratamiento de señales digitales es un campo muy importante de la informática. Dichas señales contienen información valiosa que ha de ser extraída y transformada para poder ser utilizada. En la presente tesis doctoral se han creado métodos para almacenar, procesar y recuperar información de las regiones contenidas en una imagen, en especial en imágenes de gran tamaño.

Como base del trabajo se ha diseñado una estructura de datos de tipo grafo para poder almacenar todas las regiones contenidas en una imagen. En esta estructura de datos se pueden guardar tanto los descriptores de bajo nivel de las regiones como las relaciones estructurales entre las distintas regiones de la imagen. En los sistemas de almacenamiento de imágenes es una práctica habitual distribuir las imágenes para mejorar el rendimiento. Más allá de este tipo de distribución, una característica distintiva y novedosa de la estructura de datos creada en la presente investigación es que puede funcionar de forma distribuida de manera que una imagen grande puede ser dividida en varias sub-imágenes, y dichas sub-imágenes pueden ser almacenadas de forma separada en varios servidores.

También se han adaptado algunos métodos y algoritmos pertenecientes a la Morfología Matemática para trabajar directamente sobre la estructura de datos distribuida. De esta manera, se pueden procesar todas las sub-imágenes de una misma imagen sin necesidad de reconstruir la imagen inicial.

Finalmente, haciendo uso de la estructura de datos y de los métodos desarrollados se ha creado un prototipo de sistema multi-agente capaz de almacenar y procesar imágenes grandes. Este prototipo permite realizar consultas para recuperar información perteneciente a regiones de una imagen almacenada en el sistema sin necesidad de volver a ser procesada.

En la experimentación realizada, resumida en los resultados presentados, se muestra que la división y distribución de una imagen en varias sub-imágenes reduce los tiempos de almacenamiento, procesamiento y recuperación de la información.

Abstract

The storage and processing of digital images is an important field of Computer Science. These images contain valuable information that must be extracted and processed to be used. In this thesis some methods have been proposed and developed to store, process and retrieve information from the regions contained in an image, especially in large images.

Images are stored using a region-oriented and graph-based data structure specifically designed for this work. This data structure stores low-level region descriptors and the structural relationships between different regions of the image.

In image management systems it is a common practice to distribute the images to improve performance. Beyond this kind of distribution, an original and novel aspect of the data structure is that it can operate in a distributed manner so that a large image can be divided into several sub-images, and these sub-images can be stored separately on multiple servers.

Some methods from Mathematical Morphology have been adapted for acting directly on the distributed data structure. This way, these methods can be applied directly on sub-images stored in remote data structures without the need of rebuilding the whole initial image.

Finally, based on the developed data structure and methods, a multi-agent system prototype has been created that is capable of storing and processing very large images. This prototype allows retrieving region information belonging to stored images in the system without rebuilding the original image.

In the thesis' experimental results, it is shown that the division and distribution of an image into several sub-images reduces the time of storage, processing and information retrieval for the regions in the original image.

Índice

Lista de Figuras

LISTA DE FIGURAS

Lista de Tablas

Capítulo 1

Introducción y Objetivos

1.1. Motivación

El almacenamiento y gestión de grandes colecciones de imágenes es un tema de investigación importante en muchos campos de la informática. En sectores como la informática médica o la bioinformática, por ejemplo, se generan diariamente una gran cantidad de imágenes. Estas imágenes contiene información de gran valor para profesionales e investigadores. Por esta razón las imágenes son procesadas para extraer la información que contienen y son almacenadas para poder ser consultadas en cualquier momento.

Comúnmente, las instituciones y empresas que poseen grandes colecciones de imágenes, utilizan sistemas de almacenamiento, indexación y recuperación de imágenes. Por ejemplo, en el campo de la informática médica estos sistemas se denominan '*Picture Archiving and Connectivity Systems*' o PACS. La adquisición e instalación de estos sistemas de administración de imágenes es, por lo general, muy costosa tanto en el aspecto económico como en el de recursos computacionales. Esto es debido a que estos sistemas requieren grandes servidores o clusters y profesionales expertos para su instalación y adaptación. Por consiguiente, el mantenimiento y actualizaciones de estos sistemas es también costoso.

En muchas ocasiones los sistemas de gestión de imágenes necesitan ser ampliados o complementados con funcionalidades que permitan llevar a cabo las tareas que se requieren sobre las colecciones de imágenes. Incluso dentro del mismo cam-

po de investigación, la misma colección de imágenes necesita distintos métodos añadidos al sistema de gestión de imágenes para poder ser utilizada de forma eficiente. Por ejemplo, en una colección de imágenes biomédicas para ayuda al diagnóstico, se necesitan generalmente métodos de extracción de regiones significativas y segmentación para intentar localizar un tumor en la imagen; también se necesitarán métodos de búsqueda de información por contenido (de la imagen) para obtener de la colección imágenes similares ya diagnosticadas.

Estos grandes sistemas se limitan normalmente a almacenar imágenes, y recuperar dichas imágenes por su identificador. Posteriormente, se utiliza un programa de visualización de imágenes para procesar y extraer información de las imágenes recuperadas. Para cada consulta de la información contenida en la imagen se ha de procesar la imagen completa, lo cual es una limitación para imágenes de gran tamaño, donde el procesamiento requiere más tiempo de computación.

A la hora de la implementación de un sistema de administración de imágenes o de un sistema PACS, la pieza elemental sobre la que construir dicho sistema es un sistema gestor de bases de datos. Dicho sistema gestor de bases de datos almacena las imágenes indexadas.

Una imagen contiene gran cantidad de información dentro de los píxeles y regiones de la imagen. Dichas regiones contienen descriptores de color, de forma, y de textura, así como relaciones existentes entre dichas regiones. Incluso, algunas imágenes, en algunos dominios, contienen información semántica contenida dentro del formato de la imagen, como el estándar DICOM en la informática bio-médica.

Un punto clave abordado por el presente trabajo es el tratamiento de imágenes de gran tamaño. En la actualidad gracias a los avances tecnológicos en los dispositivos de captura de señal, el tamaño de las imágenes es cada vez mayor. Y como es lógico, cuanto mayor es una imagen, mayor es la cantidad de información que contiene y mayor es el tiempo necesario para procesar, almacenar y recuperar información de la imágenes del sistema de administración de imágenes. Algunos tipos de estas imágenes muy grandes son:

- Imágenes capturadas con sistemas de alta resolución que generan imágenes de gran tamaño.

  Dentro de esta clasificación están las imágenes de satélite de alta resolu-

ción. Por ejemplo, en el proyecto *Blue Marble* del catálogo *Visible Earth* de la NASA, contiene imágenes tomadas por satélite de la Tierra, que pueden alcanzar tamaños en formato RAW de 86.400×43.200 píxeles, esto es 3.732.480.000 píxeles (Wolfe, Roy, y Vermote, 1998) (N.A.S.A., 2012).

- Imágenes artificiales.

  Existen proyectos, como por ejemplo *GigaPixel*, en los que se crean imágenes de muy alta resolución a partir de una colección de fotografías más detalladas de algún objeto o paisaje.

Existen otros campos en los que se generan imágenes de gran tamaño, como pueden ser las imágenes médicas, o las imágenes de anatomía patológica (García-Rojo, Bueno-García, González-García, y Carbajo-Vicente, 2005).

A la hora de almacenar y recuperar información de grandes colecciones de imágenes en una base de datos, diversos trabajos demuestran los beneficios de distribuir estas grandes colecciones de imágenes (Gudivada y Raghavan, 1995) (Gardels, 1996) (Burl, Fowlkes, Roden, Stechert, y Mukhtar, 1999) (Hastings, Langella, Oster, y Saltz, 2004). Dicha distribución ayuda a reducir el tiempo necesario para recuperar información de la colección.

1.2. Hipótesis de partida

Esta tesis doctoral aborda la creación de una estructura de datos y métodos asociados, para estudiar la validez de la siguiente hipótesis:

La división de imágenes de gran tamaño en sub-imágenes, y la distribución de estas sub-imágenes, mejora el tiempo de procesamiento morfológico basado en grafos y de recuperación de información de las regiones en las sub-imágenes distribuidas.

1.3. Objetivos

Para estudiar la validez de la hipótesis de partida, se han definido los siguientes objetivos, compuestos de una serie hitos a cumplir:

1. Creación de una estructura de datos distribuida de tipo grafo, implementada en una base de datos relacional, para almacenar imágenes.

   Hito:

   - Diseño de un esquema de base de datos capaz de contener la estructura de datos diseñada.

2. Desarrollar algoritmos y métodos para la división, almacenamiento, procesamiento distribuido de una imagen y recuperación de información de regiones de la imagen.

   Hitos:

   - Desarrollo de un nuevo algoritmo de división, que preserve las vecindades del grafo de la imagen, de tal manera que cada sub-imagen pueda ser almacenada y tratada independientemente del resto, como si fuera una imagen completa.
   - Desarrollo de métodos de almacenamiento que disminuyan el tiempo necesario para guardar imágenes de gran tamaño en la estructura de datos.
   - Adaptación de métodos y operaciones de la Morfología Matemática para poder aplicarse de forma distribuida sobre la estructura de datos. Dichos métodos evitarán en la medida de lo posible el trasiego de información de la imagen por la red ya que se trata, en general, de imágenes de gran tamaño. Se facilita de esta manera el procesamiento y recuperación de imágenes muy grandes.
   - Facilitar la recuperación de la información de regiones de las imágenes almacenada en la estructura de datos.

3. Implementar un sistema que facilite el uso y evaluación de los métodos distribuidos que sean desarrollados.

   Hitos:

   - Creación de un sistema distribuido multi-agente capaz de manejar colecciones de imágenes de gran tamaño.

- El sistema debe ser accesible y ofrecer facilidades para su uso. También ofrecerá métodos para la evaluación de los métodos desarrollados en la presente tesis.

4. El sistema debería poder ser mantenido y ampliado de una forma sencilla. Asimismo, sería deseable que el sistema no necesitara grandes recursos computacionales para su funcionamiento.

1.4. Estructura del documento

La organización del presente documento comienza en este capítulo 1 de introducción. En este capítulo se establece una hipótesis y unos objetivos para investigación llevada a cabo.

- En el capítulo 2 se realiza una introducción al 'estado del arte' del proyecto realizado, comentando las tecnologías utilizadas para la consecución de la tesis.

- A continuación, en el capítulo 3 se presentan los resultados del trabajo de investigación de la tesis: (i) Una estructura de datos de tipo grafo original utilizada como base para almacenar las imágenes en el presente trabajo. (ii) Un novedoso algoritmo de división, creado para dividir las imágenes preservando vecindades de las regiones. (iii) La adaptación para trabajar de forma distribuida y directamente sobre la estructura de datos creada de métodos y operaciones provenientes de la Morfología Matemática. (iv) Las posibilidades ofrecidas por la estructura de datos para recuperar información de las regiones de las imágenes almacenadas.

- La estructura y funcionamiento del sistema creado, usando la estructura de datos y métodos desarrollados en la investigación de la presente tesis doctoral, se detallan en el capítulo 4. Este sistema, que utiliza la tecnología de agentes, posee novedosas técnicas de balanceo de carga. Así, este capítulo explica la composición interna de los agentes del sistema y de dicho balanceo de carga implementado.

- En el capítulo 5 se recogen los resultados numéricos experimentales obtenidos de aplicar de una batería de pruebas sobre el sistema multi-agente. Se han realizado pruebas con distintos tipos de operaciones en el sistema. A la vez, para cada operación realizada, se ha variado el número de agentes presentes en el sistema.

  Todas estas pruebas se han realizado con un juego de imágenes de distintos tamaños, para estudiar la variación de los tiempos, también, con respecto al tamaño de la imagen.

- En el capítulo 6 de Conclusiones, se revisa la consecución de los objetivos iniciales que se querían alcanzar, comentando cómo se han logrado y las dificultades encontradas. También se hace referencia a las posibles mejoras que se podrían realizar. En la sección 6.3 se comentan las dificultades encontradas para la consecución del trabajo realizado. Y en la sección 6.4 de futuras líneas de trabajo, del mismo capítulo, se tratan los nuevos frentes de investigación abiertos tras la realización de la presente tesis doctoral.

  El capítulo 7 es la traducción al inglés del capítulo 6.

- Para concluir, la bibliografía incluye las referencias bibliográficas de los artículos y los libros consultados durante el desarrollo de la investigación de la presente tesis doctoral.

Capítulo 2

Estado del arte

En este capítulo se estudia el estado actual de la cuestión. En él se describen las bases y los principales enfoques existentes en la actualidad de las tecnologías utilizadas en la investigación y el desarrollo de la presente tesis doctoral.

En la sección 2.1 se analizan los diferentes métodos y arquitecturas existentes en el campo de la integración de bases de datos heterogéneas. La siguiente sección 2.2 realiza una breve explicación del amplio campo de la Morfología Matemática comentando sus fundamentos básicos y los métodos que se han usado en la presente tesis doctoral. Por último, la sección 2.3 presenta el paradigma de computación denominado *Cloud Computing*, y se describen sus principales características.

2.1. Métodos de integración y acceso a fuentes de datos distribuidas

La integración de información de orígenes distribuidos es un área muy extensa de la informática. La información puede ser almacenada de diversas formas, como ficheros de texto planos, ficheros anotados, ficheros binarios, hojas de cálculo o en bases de datos. A menudo, la integración de información se utiliza para intercambiar y relacionar distintas fuentes de datos heterogéneas y en distintas localizaciones.

Siguiendo la revisión hecha en (Sujansky, 2001), se podrían distinguir dos

métodos principales para realizar la integración de información:

1. Traducción los datos.

   En este método los datos son obtenidos de sus orígenes y almacenados en un repositorio centralizado que tiene un esquema de datos predefinido. Los usuarios acceden al repositorio obteniendo datos de las distintas fuentes de datos de forma totalmente transparente. El ejemplo más común de este tipo de integración son los *data warehouses*.

   Este tipo de acceso a los datos tiene como principal ventaja la rapidez de obtención de los datos para el usuario final, ya que todos los datos se encuentran almacenados en el repositorio central. Al almacenar los datos en el repositorio central algunos datos necesitan ser traducidos para almacenarlos usando unas escalas y vocabularios comunes, por eso a este método se le denomina de traducción de datos, ya que se traducen los datos de las fuentes de datos originales al esquema y medidas del repositorio central para almacenarlos.

   También es destacable la autonomía que se consigue al tener una copia de los datos originales, con lo que no es necesario que exista conectividad con las fuentes de datos originales en todo momento. Este mismo aspecto presenta como desventaja que los procesos de carga del repositorio central suelen ser costosos, y el repositorio central debe ser actualizado con frecuencia para que los datos sean consistentes con las fuentes de datos originales.

2. Traducción de las consultas.

   En este segundo método las fuentes de datos e información se encuentran separadas y se interconectan mediante redes de datos, como por ejemplo Internet.

   Dentro de la traducción de consultas se pueden distinguir cuatro categorías diferenciadas por la forma de acceder a la información de las fuentes de datos:

   - Los primeros sistemas y métodos creados que usaban traducción de consultas para acceder a datos distribuidos y heterogéneos eran sis-

temas software especializados que actuaban puramente como **mediadores** y **envoltorios**, llamados en la bibliografía en inglés *mediators* y *wrappers* respectivamente (Wiederhold, 1992). Estos mediadores y envoltorios actúan como *brokers* entre los usuarios y las fuentes datos.

Los mediadores ofrecen a los usuarios la posibilidad de preguntar al usuario y generar consultas. Los envoltorios o *wrappers* contienen información acerca de la estructura real de las fuentes de datos y conocen la manera de traducir las consultas enviadas por los mediadores al lenguaje propio de las fuentes de datos.

En este tipo de traducción de consultas, cada fuente de datos suele necesitar un envoltorio específico. Algunos sistemas de ejemplo que utilizan este tipo de acceso a la información para la integración son *TSIMMIS* (Chawathe y cols., 1994), y *DIOM* (L. Liu y Pu, 1995).

- Otro enfoque se basa en el uso de un **esquema conceptual virtual único**. Este esquema virtual provee a los usuarios de un modelo semántico único que integra la información contenida en todas las fuentes de datos conectadas al sistema. Las consultas se crean con objetos pertenecientes a este esquema conceptual y se envían a un módulo capaz de dividir una consulta en un conjunto de sub-consultas que son enviadas a las bases de datos reales que componen el sistema. Este mismo módulo es el responsable de obtener los resultados de las consultas, transformar y mezclar dichos resultados para presentárselos a los usuarios.

  Ejemplos de este enfoque son *SIMS information mediator* (Arens, Hsu, y Knoblock, 1998), y su adaptación a la web *ARIADNE* (Knoblock y cols., 2001). Existen otros sistemas más específicos que siguen este enfoque, por ejemplo, en el campo de la biomedicina, los sistemas *BACIIS* (Ben Miled, Li, Kellett, Sipes, y Bukhres, 2002), y *TAMBIS* (Baker y cols., 1998).

- Existen otros sistemas que se basan en el uso de **esquemas conceptuales virtuales múltiples** donde cada fuente de datos se representa con un esquema conceptual propio. Este modelo soporta la inclusión,

exclusión y modificación de fuentes de datos de una manera más sencilla que los anteriores.

La principal dificultad de estos sistemas es el crear las relaciones semánticas entre entidades de distintas fuentes de datos para poder conseguir la integración. *OBSERVER* (Mena, Illarramendi, Kashyap, y Sheth, 2000) es un ejemplo de sistema que usa esta filosofía.

- Por último cabe mencionar los llamados sistemas **híbridos**. Estos sistemas son similares a los sistemas con esquemas conceptuales virtuales múltiples, con la salvedad de que utilizan un vocabulario, ontología o tesauro compartido con el que se construyen los distintos esquemas virtuales de las fuentes de datos. De este modo la interconexión de conceptos de las distintas fuentes de datos queda solucionado.

  Algunos ejemplos de sistemas híbridos son *PICSEL* (Goasdoué, Lattes, y Rousset, 2000), *COIN* (Goh, 1997), *BUSTER* (Stuckenschmidt, Harmelen, Fensel, Klein, y Horrocks, 2000) y *ONTOFUSION* (Pérez-Rey y cols., 2006).

El enfoque de traducción de consultas ofrece una mayor coherencia en el acceso a los datos frente al enfoque de traducción datos, ya que al obtener los datos directamente de la fuente de información original, estos siempre están actualizados. Pero también cuenta con algunas desventajas como que el rendimiento y la seguridad pueden ser más bajos, ya que en una sola consulta podría ser necesario acceder a varias fuentes de datos separadas geográficamente y posteriormente habría que realizar el paso de integración de los datos obtenidos de cada origen de datos.

Como se ha comentado en párrafos anteriores, para poder integrar información de distintas fuentes de datos se han de definir los conceptos que contiene cada una de estas fuente de datos. También es importante inter-relacionar los conceptos de los distintos orígenes de datos para poder lograr la integración.

Existen muchos aspectos de gran dificultad para la integración de datos, como la homogeneización de los datos, ya que el mismo concepto puede usar distintas unidades de medida en distintas fuentes, o puede ocurrir que el mismo objeto posea distintos identificadores en distintas bases de datos.

En la presente tesis doctoral se utiliza el enfoque de integración de bases de datos distribuidas usando traducción de consultas, donde las distintas bases de datos poseen un esquema conceptual virtual único.

2.2. Breve introducción a la Morfología Matemática

La Morfología Matemática es un amplio campo de técnicas no lineales de procesamiento de imágenes basada en operaciones de conjuntos (Serra, 1982) (Serra, 1988) (Beucher y Meyer, 1993) (Serra y Salembier, 1993) (Crespo, Serra, y Schafer, 1995) (Salembier y Serra, 1995) (Soille, 2003). La estructura básica (matemática) de la Morfología Matemática es el retículo completo.

Como se comenta en (Ronse, 2005), para la Morfología Matemática una imagen no es una combinación de frecuencias sinusoidales, ni el resultado de un proceso de Markov sobre puntos individuales. La Morfología Matemática considera que una imagen está compuesta por formas geométricas que tienen unas características específicas de color (o luminosidad). Por lo tanto, estas regiones pueden ser estudiadas por su interacción con otras formas con sus propias características de color.

Para realizar la comparación de los objetos que existen dentro de la imagen se utiliza un objeto con forma conocida. A este objeto de forma conocida se llama *elemento estructurante*.

Algunos elementos estructurantes básicos pueden verse en la figura 2.1.

Si se considera una imagen como una matriz de píxeles, el elemento estructurante define la vecindad de cada píxel dentro de la imagen. Tomando la figura 2.1, para los distintos elementos estructurantes mostrados, la vecindad del píxel marcado con una 'x' son todos aquellos píxeles con color gris en el elemento estructurante.

El comportamiento de los filtros morfológicos depende en gran parte de la forma y tamaño del elemento estructurante. Es decir, el resultado de una operación sobre una imagen puede tener distintos resultados dependiendo del elemento estructurante seleccionado para realizar dicha operación.

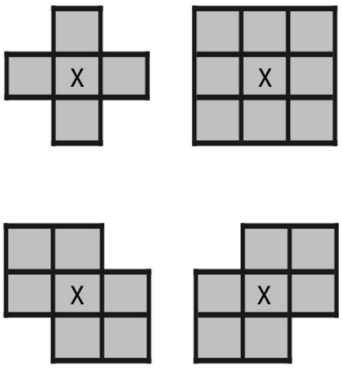

Figura 2.1: Ejemplos de elementos estructurantes

Para llevar a cabo el procesamiento de una imagen muchas operaciones morfológicas involucran una traslación del elemento estructurante sobre todos los píxeles que componen la imagen, y la imagen resultado, será la obtenida de aplicar la operación sobre la vecindad de cada píxel.

En las siguientes secciones se van a comentar los operadores básicos de la Morfología Matemática, la erosión y la dilatación.

2.2.1. Operadores básicos de la Morfología Matemática

- Dilatación: Para una imagen binaria (conjunto X), la dilatación según el elemento estructurante B, o $\delta_B(X)$ es la unión de las traslaciones de X según B:

$$\delta_B(X) = \{x | B_x \cap X \neq \emptyset\}$$

  Para una imagen en escala de grises la dilatación para un píxel x dado de la imagen f según el elemento estructurante B, o $[\delta_B(f)](x)$, será el máximo valor de la imagen en la ventana definida por el elemento estructurante B cuando tiene su origen en x:

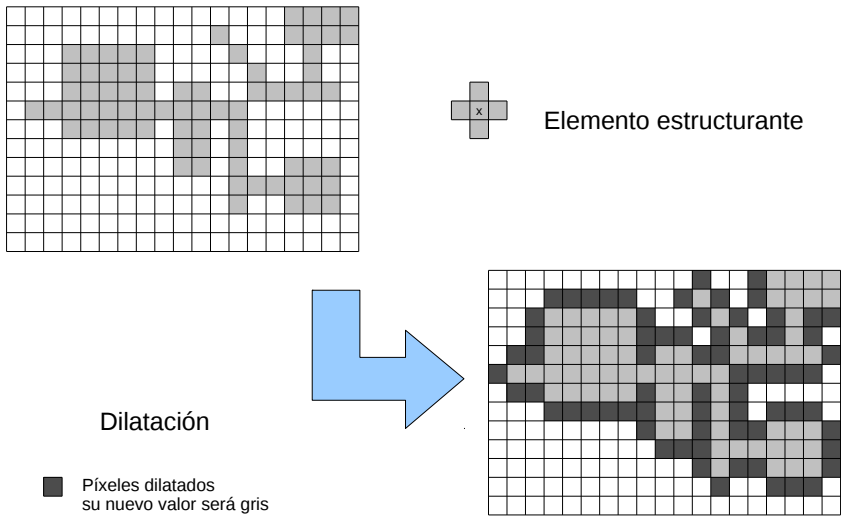

Figura 2.2: Ejemplo de dilatación morfológica

$$[\delta_B(f)](x) = \text{máx}\{f(x+b)|b \in B\}$$

En la presente tesis se utilizan operadores sobre grafos. La dilatación morfológica sobre un vértice v del grafo G de la imagen f se define como el máximo de los valores de los vértices vecinos de v y el propio v. Donde la vecindad de un vértice en el grafo $N_G(v)$ está definida por aquellos vértices que están unidos a v por un arco del grafo:

$$[\delta_G(f)](v) = \text{máx}\{f(v')|v' \in N_G(v)\}$$

- Erosión: Para una imagen binaria (conjunto X), la erosión según el elemento estructurante B, o $\epsilon_B(X)$ se define como el lugar de los puntos x tal que B_x está incluido en X, cuanto está situado en ese punto x:

$$\epsilon_B(X) = \{x | B_x \subseteq X\}$$

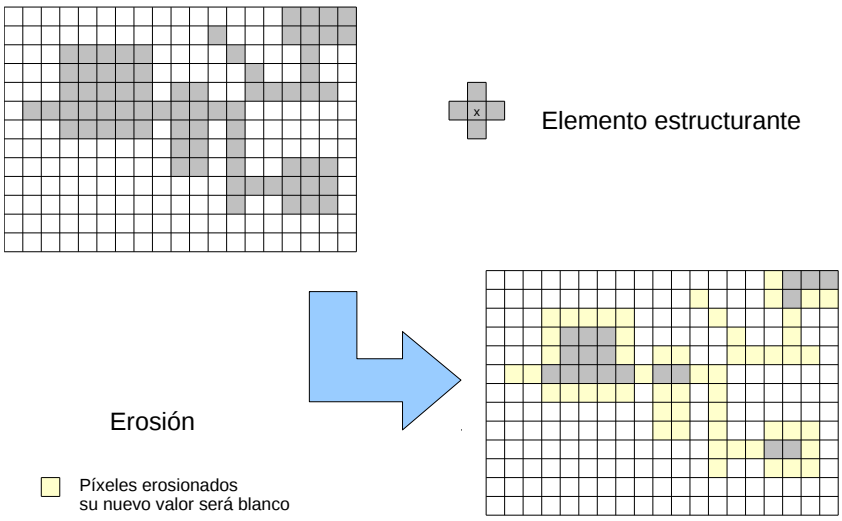

Figura 2.3: Ejemplo de erosión morfológica

Para una imagen en escala de grises la erosión para un píxel x dado, de la imagen f según el elemento estructurante B, o $[\epsilon_B(f)](x)$, será el mínimo valor de la imagen en la ventana definida por el elemento estructurante B cuando tiene su origen en x:

$$[\epsilon_B(f)](x) = \text{mín}\{f(x+b) | b \in B\}$$

Así mismo, la erosión morfológica sobre un vértice v del grafo G de la imagen f se define como el mínimo de los valores de los vértices vecinos de v y el propio v:

$$[\epsilon_G(f)](v) = \text{mín}\{f(v')|v' \in N_G(v)\}$$

Para una imagen multicanal f la dilatación morfológica $\delta_B(f)$ según el elemento estructurante B, será el resultado de aplicar una dilatación a cada uno de los canales de la imagen f_i:

$$\delta_B(f) = [\delta_B(f_1), \delta_B(f_2), ..., \delta_B(f_n)]$$

Análogamente, la erosión morfológica $\epsilon_B(f)$ de la imagen f según el elemento estructurante B, será el resultado de aplicar una erosión a cada uno de los canales de la imagen a cada uno de los canales de la imagen f_i:

$$\epsilon_B(f) = [\epsilon_B(f_1), \epsilon_B(f_2), ..., \epsilon_B(f_n)]$$

2.2.2. Otros operadores característicos de la Morfología Matemática

A partir de las operaciones básicas de erosión y dilatación es posible definir otras operaciones más complejas. Entre estas operaciones las más conocidas son:

- Apertura: La apertura morfológica γ de la imagen f según el elemento estructurante B se define como el resultado de aplicar una erosión morfológica ϵ_B sobre la imagen f y seguidamente una dilatación morfológica usando el elemento estructurante dual $\delta_{\hat{B}}$. Donde $\hat{B}$ es el resultado de trasponer el elemento estructurante B:

$$\gamma_B(f) = \delta_{\hat{B}}[\epsilon_B(f)]$$

- Cierre: El cierre morfológico ϕ de la imagen f según el elemento estructurante B se define como el resultado de aplicar una dilatación morfológica δ_B sobre la imagen f y seguidamente una erosión morfológica usando elemento estructurante dual $\epsilon_{\hat{B}}$. Donde $\hat{B}$ es el resultado de trasponer el elemento estructurante B:

$$\phi_B(f) = \epsilon_{\hat{B}}[\delta_B(f)]$$

Existen gran variedad de operaciones y algoritmos morfológicos para el tratamiento de imágenes, pero solamente se van a comentar aquellos que se usan en la presente tesis.

2.2.3. Segmentación de imágenes usando Morfología Matemática

La segmentación es una operación usada para la extracción de regiones significativas de una imagen (Meyer y Beucher, 1990) (Crespo, Serra, y Schafer, 1993) (Marcotegui, 1996) (Salembier y Garrido, 2000) (Yu, Fritts, y Sun, 2002).

Las técnicas tradicionales de segmentación de imágenes suelen dividirse en dos tipos:

- Técnicas de crecimiento de regiones.

  Son aquellas técnicas donde los píxeles de una imagen son asignados a una región usando medidas de proximidad y de similitud. Se parte de un conjunto de regiones iniciales o semillas, y la segmentación termina cuando todos los píxeles de una imagen están asignados a alguna región. Los bordes de las regiones se definen en aquellos píxeles donde dos regiones distintas se encuentran.

- Técnicas de detección de bordes.

  Estas técnicas se basan en suponer que el interior de los objetos contenidos en una imagen sufren leves variaciones de color. A la vez, se supone que en los límites de los objetos existen grandes variaciones con respecto a sus objetos vecinos. La tarea que realizan estas técnicas es básicamente detectar esas variaciones en los niveles de color y realzarlas. Posteriormente, usando el gradiente, se localizan los bordes de los objetos presentes en la imagen. Un problema es que en general los bordes de una región tienen una definición desigual.

En la Morfología Matemática existen varios métodos para la segmentación de imágenes. El más extendido, y que combina técnicas de crecimiento de regiones y de detección de bordes, se denomina **watershed**.

- El algoritmo de la *watershed* detecta los mínimos locales de la imagen, y va agrupando píxeles a estos mínimos hasta definir unas líneas de división. Posteriormente usando un gradiente define las fronteras entre las distintas regiones obteniendo los bordes de la imagen.

  En esta técnica se trabaja con una imagen gradiente, y no con la imagen original. Si consideramos la imagen gradiente en niveles de gris como una representación topográfica de niveles de altitud, la *watershed* es la línea divisoria de las cuencas hidrográficas. Para calcular la *watershed* suelen utilizarse algoritmos de inundación de gradiente (Soille, 2003) (Vincent y Soille, 1991) (Beucher y Meyer, 1993) (Meyer y Beucher, 1990). Estos algoritmos inundan la imagen gradiente por sus mínimos locales y obtienen las líneas *watershed* en los píxeles donde aguas de cuencas adyacentes se encuentran.

  Una solución al problema de la 'sobre-segmentación' (una segmentación con un excesivo número de regiones, debido a que el gradiente tiene un elevado número de mínimos) es utilizar los denominados 'marcadores' que se utilizan para modificar la homotopía del gradiente, de forma que éste tenga como mínimos solamente los marcadores. Así, al inundar el gradiente, se obtendrán tantas regiones como marcadores. Dichos marcadores deben señalar regiones significativas de la imagen, y generalmente se calculan de forma semi-automática (Soille, 2003).

  En la figura 2.4 se muestra gráficamente los conceptos explicados del algoritmo de la *watershed*.

  En la presente tesis se ha implementado un algoritmo de segmentación de la *watershed* distribuido.

En este trabajo, para segmentar las regiones de una imagen se han utilizado técnicas de fusión de **zonas planas** (Crespo, 1993)(Crespo, Schafer, Serra, Gratin, y Meyer, 1997). Para realizar la segmentación, inicialmente se construye el

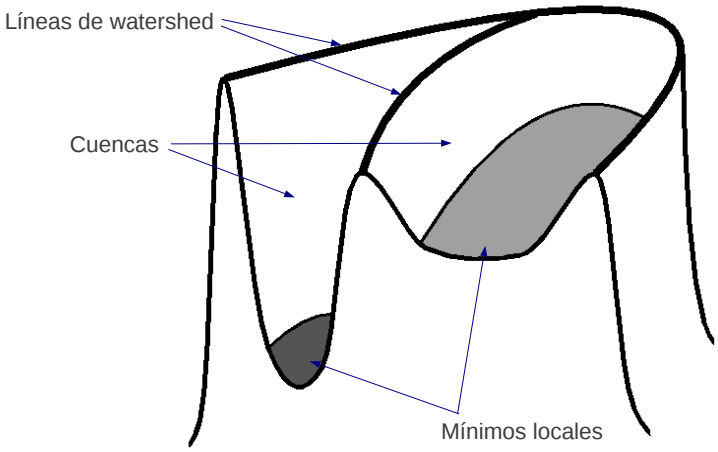

Figura 2.4: Watershed

grafo de regiones de la imagen, de forma que una región está compuesta por componentes conexas de píxeles que poseen un mismo nivel de color. Posteriormente, sobre este grafo inicial se aplican métodos de fusión de regiones y se obtienen como resultado las regiones significativas de la imagen.

2.3. Cloud Computing

Cloud Computing es un término tecnológico de gran actualidad, que se presenta como el nuevo paradigma de programación para el futuro. Aunque si tomamos como referencia la definición del NIST (*National Institute of Standards and Technology*) hecha por Peter Mell y Tim Grance en Octubre de 2009 Nos podemos dar cuenta de que el *Cloud Computing* lo estamos utilizando cotidianamente, por ejemplo usamos las redes sociales que pueden ser vistas como un Servicio Software Bajo demanda (*Software as a Service* - SaaS).

La definición formal del NIST es:

"Cloud computing es un modelo para habilitar acceso conveniente bajo demanda a un conjunto compartido de recursos computacionales configurables (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de los servicios."

Este modelo hace hincapié en la disponibilidad y posee cinco características esenciales, tres modelos de servicio y cuatro modelos de desarrollo, que se analizan en las siguientes sub-secciones.

2.3.1. Características esenciales del Cloud Computing

Como se ha indicado anteriormente, un servicio de *cloud computing* debe poseer las cinco características esenciales que se explican a continuación:

- *Auto-servicio bajo demanda.* Los clientes de los servicios*cloud computing* pueden acceder al servicio automáticamente. Es decir, no necesitan ninguna intervención de un proveedor humano.

- *Acceso por red de banda ancha.* Los servicios son accesibles por métodos estándar a través de internet. Es necesario un gran ancho de banda para muchos servicios, por lo que sería necesaria una conexión con el proveedor de servicios equivalente a una conexión de área local.

- *Batería (pool) de recursos.* El usuario no sabe exactamente de cuántos recursos dispone el proveedor, ni dónde se encuentran, lo único que necesita saber es que el proveedor le suministra tantos recursos como él necesita para completar su propósito.

- *Elasticidad y rapidez.* El sistema que implemente el *cloud computing* debe ser escalable de forma transparente al cliente. En el caso que un cliente necesite un aumento de recursos en un momento dado, el sistema debe ser fácilmente ampliable.

- *Medición del servicio.* La utilización de los recursos por parte del cliente es controlada y almacenada de forma transparente por el proveedor. De este

modo el proveedor puede ofrecer sus servicios de pago por uso.

2.3.2. Modelos de Servicios en Cloud Computing

Atendiendo al nivel de acceso a los recursos ofrecidos en los servicios ofertados, el NIST define tres modelos de servicios de *cloud computing*.

- *Nube de Software como Servicio (SaaS)*. El proveedor ofrece acceso al usuario a una aplicación propia. El usuario sólo puede utilizar las funcionalidades ofrecidas por dicha aplicación. Como se ha mencionado en la introducción de esta sección, un servicio SaaS podría ser, por ejemplo, Facebook.

- *Nube de Plataforma como Servicio (PaaS)*. Se le da acceso al cliente a una infraestructura computacional que le permite instalar y ejecutar aplicaciones propias. Un ejemplo de servicio PaaS podría ser el acceso a un escritorio remoto de un ordenador. En este tipo de servicios el usuario no tiene permitido administrar los recursos del sistema.

- *Nube de Infraestructura como Servicio (IaaS)*. El cliente puede administrar los recursos físicos e instalar software propio, incluso el Sistema Operativo.

2.3.3. Modelos de desarrollo de Nubes

En la definición de *cloud computing* se definen cuatro modelos de desarrollo:

- *Nube Privada*. La nube es usada exclusivamente por una organización.

- *Nube Comunitaria*. La nube es compartida por varias organizaciones con propósitos comunes.

- *Nube Pública*. Nube disponible para el público general. La nube pertenece a una organización que vende sus servicios.

- *Nube Híbrida*. Tipo de nube que mezcla varios de los modelos anteriores.

Capítulo 3

Métodos para el almacenamiento, procesamiento y recuperación distribuidos de imágenes de gran tamaño

El presente capítulo explica los métodos novedosos creados para almacenar y analizar imágenes de gran tamaño. Gracias a dichos métodos se ha creado un prototipo de sistema de almacenamiento, tratamiento y recuperación de imágenes, que también se explicará.

En primer lugar, la sección 3.1 detalla la estructura de datos implementada en una base de datos. Esta estructura de datos ha sido creada para almacenar la información contenida en una imagen. Dicha estructura de datos es la base de los métodos, operaciones y consultas de recuperación sobre las imágenes. A continuación, en la sección 3.2 se describe la optimización y modificación de la estructura de datos para poder ser utilizada de forma distribuida. Dentro de este apartado se explica el algoritmo de división de imágenes original que se ha creado.

3.1. Estructura de datos para almacenamiento de imágenes de gran tamaño

El núcleo principal para realizar el almacenamiento y tratamiento de imágenes es una estructura de datos para manejar las imágenes. Dicha estructura de datos

debe proporcionar una forma eficiente para el almacenamiento y posterior acceso a toda la información obtenida de las imágenes.

El primer paso para definir la estructura de datos es saber qué tipo de información se va a almacenar. En nuestro caso queremos almacenar información significativa de las imágenes, es decir obtendremos la información de regiones de las imágenes (Crespo y cols., 1993) (Crespo y cols., 1995) (Crespo y cols., 1997) (Crespo y Maojo, 1998). Además se busca que esta estructura de datos esté orientada para favorecer el uso de las operaciones y filtros de la Morfología Matemática (Serra, 1982) (Serra, 1988) (Serra y Salembier, 1993).

Una región de una imagen se define por diversas características – color, tamaño, forma, textura, orientación, borde, etcétera –, pero también es muy importante su posición dentro de la imagen y las regiones con las que limita. Es decir, las relaciones que existen entre las distintas regiones en la imagen son también importantes. Además cada región puede estar compuesta de varias regiones más pequeñas.

La tabla 3.1 muestra los descriptores de características que se han seleccionado para guardar en la estructura de datos, para cada región de una imagen. se puede observar que existen descriptores de color, forma, textura, posición y relaciones entre las distintas regiones. Este conjunto inicial de características se puede ampliar fácilmente. También se puede usar un subconjunto de estos descriptores para poder aumentar la eficiencia del sistema y reducir el tiempo de procesamiento, ya que el conjunto inicial de descriptores ofrece una amplia variedad de descriptores físicos, y algunos de ellos requieren un calculo complejo como por ejemplo la estimación del número de lados de un polígono que aproxime la región.

Cabe destacar que algunas de las relaciones que se pueden almacenar en la estructura de datos no pueden ser extraídas a priori de la imagen. Es decir, las relaciones 'relacionado-con' y 'es-parte-de' pueden ser extraídas de la imagen de una manera automática, sin embargo las relaciones 'relacionado-con' y 'disjunto-de' son relaciones que deben ser deducidas y anotadas por los usuarios, ya que requieren un conocimiento del significado de la imagen. Igualmente la descripción de la región ('descripción-textual') debe ser dado por un usuario.

A la hora de implementar la estructura de datos, para poder almacenar las

Descriptores de Color	Descriptores de Forma	Relaciones
Media Rojo	Área	adyacente-a
Media Verde	Tamaño eje mayor	es-parte-de
Media Azul	Tamaño eje menor	relacionado-con
Moda Rojo	Orientación eje mayor	disjunto-de
Moda Verde	Orientación eje menor	descripción textual
Moda Azul	Centroide	
Varianza Rojo	Puntos de la región	
Varianza Verde	Puntos de Borde	
Varianza Azul	Longitud de los píxeles de borde	
Máximo valor Rojo	Medida de compacidad	
Máximo valor Verde	Estimación del número de lados	
Máximo valor Azul		
Mínimo valor Rojo		
Mínimo valor Verde		
Mínimo valor Azul		

Tabla 3.1: Clasificación de los descriptores obtenidos para una región de una imagen.

relaciones entre regiones existentes en una imagen de entrada, se ha escogido una estructura de tipo grafo. De esta manera, cada región presente en la imagen se representa en la estructura de datos como un nodo. Y cada relación entre regiones de la imagen puede verse como un arco del grafo.

Todos los métodos desarrollados para el sistema se basan en regiones, al igual que la estructura de datos, y usan esta estructura de datos jerárquicas y de tipo grafo. Existen trabajos previos que usan estructuras de datos basadas en regiones. (Tanimoto y Pavlidis, 1975) (Marcotegui, 1996) (Salembier y Garrido, 2000)(Ostermann, 2002) (Yu y cols., 2002). Las citadas estructuras de datos están diseñadas para contener información de regiones de una imagen y mantenerlas almacenadas en memoria. Esto significa que dichas estructuras de datos trabajan sobre memoria volátil y no serían válidas para imágenes de gran tamaño debido a los grandes recursos que éstas requieren. Además, la estructura de datos creada en el presente trabajo es capaz de almacenar mayor cantidad de información de las regiones de una imagen y de sus relaciones. Por ejemplo, la estructura de datos

propuesta en (Ostermann, 2002) es un árbol construido usando las relaciones de inclusión entre las regiones, de manera análoga a los *quad-trees*, por lo que sólo almacena este tipo de relación.

Adicionalmente, como la estructura de datos se ha diseñado para poder usarse para el tratamiento y análisis de la información contenida en las imágenes, la estructura de datos es capaz de manejar varios niveles interrelacionados para una misma imagen. Al aplicar un filtro a una imagen se puede guardar el estado actual de la imagen, generando de esta manera una nueva versión de la imagen (con el filtro aplicado). Por lo que, cada uno de estos niveles de una imagen contiene la imagen completa, y se corresponden a la aplicación de operadores y filtros sobre las regiones de la imagen. Es decir, el nivel n es el resultado de aplicar un tratamiento al nivel anterior, el nivel $n-1$.

Para una imagen inicial dada (véase 3.1) después de obtener las 12 regiones que componen la imagen (véase 3.2).

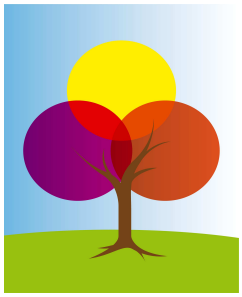

Figura 3.1: Imagen inicial

Figura 3.2: Regiones de la imagen inicial

Si a las regiones iniciales de la imagen (nivel 0), por ejemplo, le aplicáramos una fusión de regiones por similitud de color en la banda roja, obtendríamos un segundo nivel (nivel 1). Las regiones del nivel 0 y del nivel 1 estarían relacionadas entre ellas como se indica en la figura 3.3.

Por lo tanto, como se aprecia en la figura 3.3, guardando los distintos niveles para una imagen, el sistema permite el registro de los filtros y operaciones aplicados sobre dicha imagen en el sistema.

Cada nivel puede guardar la información completa de la imagen con los valores de los descriptores de las regiones, o simplemente los parámetros de los filtros

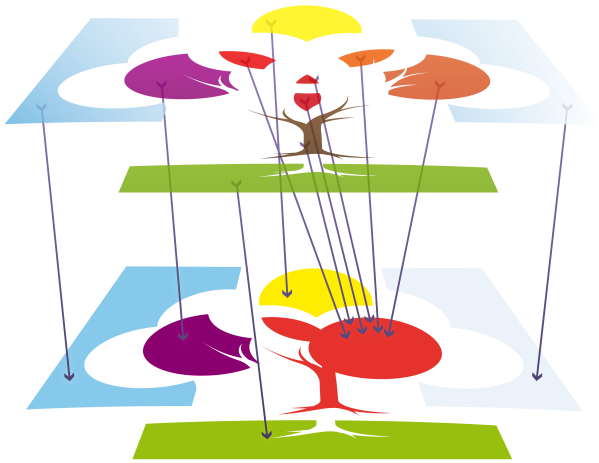

Figura 3.3: Niveles de procesamiento de una imagen

aplicados sobre el nivel anterior. Al disponer de los distintos niveles completos, se puede navegar a través de las simplificaciones de una imagen.

3.1.1. Esquema de la implementación de la estructura de datos en un SGBD relacional

Como ya se ha comentado, el tipo de imágenes que se quieren tratar en la presente investigación son imágenes de gran tamaño, y por lo tanto con gran cantidad de información contenida. Debido al tamaño de dichas imágenes, mantener una imagen en la estructura de datos, conteniendo todas sus regiones y características, en la memoria del ordenador es generalmente complicado. Más aún cuando lo que se quiere es almacenar varias imágenes.

Además, la intención es que toda esta información obtenida sea accesible, tanto para realizar análisis sobre ella como para la realización de posteriores búsquedas.

Por estas razón se ha implementado la estructura de datos explicada en 3.1 en un esquema de base de datos relacional. Dicha implementación se puede ver en la figura 3.4.

El esquema de base de datos está preparado para que se pueda almacenar la

Figura 3.4: Esquema de base de datos relacional de la estructura de datos

información completa de la imagen, sus regiones y todos los descriptores de dichas regiones. Además se distinguen distintos niveles de una misma imagen como se ha explicado en el apartado anterior.

Para facilitar el uso de la estructura de datos se han desarrollado métodos de acceso a la base de datos, de tal manera que sea totalmente transparente el hecho de que esté trabajando sobre un SGBD y no sobre un fichero de imagen.

3.1.1.1. Búsqueda de regiones en imágenes por contenido

Uno de los objetivos principales de la presente tesis es la realización de búsquedas y recuperar regiones de una imagen por sus características. Se quiere comprobar si el uso de la estructura de datos creada mejora la extracción de regiones significativas de una imagen usando criterios de las características de las regiones.

La implementación de la estructura de datos usando una base de datos faci-

lita en gran medida la búsqueda de regiones de imágenes por sus descriptores. Muchos de los sistemas de CBIR desarrollados usan bases de datos para almacenar las imágenes indexadas y posteriormente recuperarlas, (Carson, Thomas, Belongie, Hellerstein, y Mallik, 1999) (Hauptmann y Witbrock, 1997) (Smith y Chang, 1996) (Smith y Chang, 1997) (Niblack y cols., 1993) (Veltkamp y Tanese, 2000) (Datta, Joshi, Li, James, y Wang, 2006) (Y. Liu, Zhang, Lu, y Ma, 2007). En el presente trabajo no sólo se indexan las imágenes, sino que además se indexan las regiones de las imágenes almacenadas.

Gracias a la gran cantidad de descriptores e información almacenada de las regiones en la base de datos, la función de búsqueda y de comparación entre regiones de las imágenes es fácilmente configurable por el usuario. Para dicha función se puede usar la información mencionada de descriptores de región y de relaciones de las regiones almacenadas 3.1.

En el tratamiento de imágenes tradicional, para extraer de una imagen las regiones que cumplan ciertos criterios, la imagen es procesada entera, obteniendo como resultado otra imagen que contiene sólo las regiones buscadas. Si después se quisieran obtener de la misma imagen las regiones que cumplen otro criterio distinto, habría que volver a procesar la imagen por completo para obtener las nuevas regiones resultado. Por el contrario, usando la estructura de datos creada, y realizando una única pasada sobre los píxeles de la imagen en el procesamiento, toda la información de las características de las regiones de dicha imagen es indexada y almacenada en la estructura de datos. Por lo que se pueden realizar consultas sobre las regiones de dicha imagen, que están indexadas, sin tener que volver a procesar la imagen. El beneficio para este tipo de procesamiento será mayor cuanto mayor sea la imagen a procesar.

3.2. Creación de la estructura de datos distribuida y los métodos de tratamiento distribuidos para imágenes de gran tamaño

Las imágenes de gran tamaño tienen muy grandes dimensiones y gran cantidad de información contenida. Por ejemplo del proyecto *Blue Marble* del catálogo *Visible Earth* de la NASA, que contiene imágenes tomadas por satélite de la

Tierra, que pueden alcanzar tamaños en formato RAW de 86.400×43.200 píxeles, esto es 3.732.480.000 píxeles (Wolfe y cols., 1998) (N.A.S.A., 2012).

Como es de suponer, el principal problema con estas imágenes es que se necesitan servidores muy potentes y una gran cantidad de tiempo para ser procesadas, debido a su gran tamaño. Más aún, cuando las operaciones son sobre regiones, como es nuestro caso, el coste es mayor, ya que la imagen debe ser pre-procesada para extraer las regiones.

Por esta razón se han adaptado los métodos y la estructura de datos para poder ser utilizados de forma distribuida. No sólo dividiendo la colección de imágenes en varios ordenadores, ya que esta técnica no presenta gran novedad, sino que una imagen de entrada puede ser dividida en varias sub-imágenes y cada una de estas sub-imágenes puede ser almacenada y tratada de forma separada al resto.

3.2.1. Algoritmo de división de imágenes de gran tamaño

Se ha diseñado un algoritmo para poder dividir la imagen inicial en distintas sub-imágenes. Posteriormente se distribuirán dichas sub-imágenes para ser almacenadas en distintas bases de datos.

Una condición muy deseable para poder procesar una imagen de forma distribuida o paralela es el expresado informalmente en la fórmula 3.1.

$$F(Imagen) \equiv \cup_i [F(Imagen_i)] \equiv F(\cup_i Imagen_i) \tag{3.1}$$

Esto es, que el resultado del procesamiento de la imagen completa sea el mismo que el resultado de aplicar el mismo procesamiento sobre todas las sub-imágenes por separado y unir todos los resultados parciales. Sin embargo esto no es posible conseguirlo para todas las operaciones sobre imágenes. Más concretamente, se ha creado un algoritmo de división, para conseguir esta propiedad para el tratamiento distribuido usando operadores de la **Morfología Matemática basados en grafos**, es decir, que operan sobre regiones.

Para nuestro sistema se van a usar operadores y filtros de la Morfología Matemática (explicada en la sección 2.2). Dichos filtros deben ser adaptados para poder usarse distribuidamente. Dado que la Morfología Matemática está basada

en el uso de elementos estructurantes, que definen vecindades, hay que preservar, en general, las vecindades que aparecen en la imagen. Es decir al dividir la imagen en varias sub-imágenes hay que mantener la información de las vecindades de la imagen original.

Un segundo punto a tener en cuenta, es que el sistema usará operaciones y filtros basados en regiones. Para las operaciones basadas en píxeles la división y procesamiento distribuido es más sencillo, pues sólo deben tenerse en cuenta los bordes de separación entre las sub-imágenes para preservar las vecindades como se aprecia en la figura 3.5.

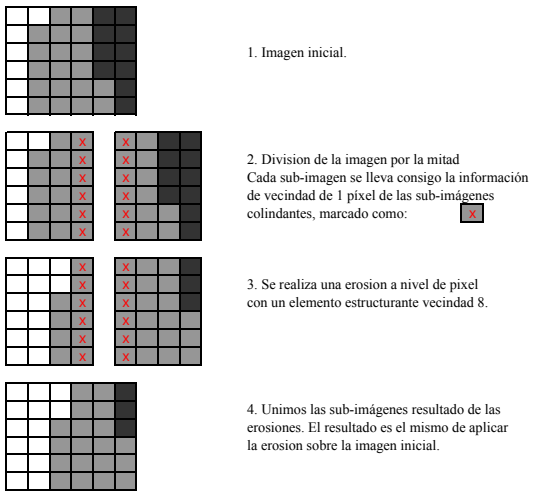

Figura 3.5: División a nivel de píxel de una imagen y erosión distribuida

En la presente tesis, como se ha comentado anteriormente, se van a utilizar operaciones sobre regiones. Se tomarán como regiones iniciales de una imagen las zonas planas dicha imagen. La erosión sobre las regiones de la imagen se realiza sobre el grafo de las regiones como se puede apreciar en la figura 3.5. La definición de la erosión basada en grafo se puede encontrar en la sección 2.2.1.

Al dividir la imagen, debemos evitar la creación de nuevas regiones. Por ello no podemos usar simplemente una división lineal de la imagen ni los conocidos *quad-trees*, ya que se podría dividir una zona plana inicial en varias sub-imágenes y procesarlas obteniendo que son distintas regiones, ver figura 3.8.

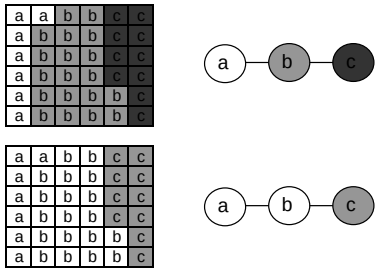

Figura 3.6: División a nivel de píxel de una imagen y erosión distribuida

Por esta razón es necesario dividir la imagen teniendo en cuenta el grafo de regiones de la imagen, ver figura 3.8.

A continuación se describe el algoritmo de división implementado, el cual recibe una imagen de entrada y la divide por su dimensión más larga (ancho o alto), obteniéndose de esta manera tres sub-imágenes I_A, I_B e I_C.

La parte central de la imagen (sub-imagen I_B) es tratada y se obtienen todas las regiones que contiene durante el proceso de división. Por lo tanto I_B está lista para ser insertada en la base de datos. Por otro lado, las partes derecha e izquierda de la imagen (sub-imágenes I_A e I_C) son imágenes sin procesar, y serán enviadas a otros nodos de procesamiento para ser tratadas.

Algoritmo:

```
FOR EACH region IN lineaDivision DO
   IF (NOT estaTratada(region)) THEN
      conjuntoRegiones = obtenerRegionesSimilares (region)
      FOR EACH region_r IN conjuntoRegiones DO
          marcarComoTratada(region_r)
      END FOR
      unionRegiones = ∪ conjuntoRegiones
      // unionRegiones : union de regiones en conjuntoRegiones
      I_A = I_A \ unionRegiones
      I_B = I_B ∪ unionRegiones
      I_C = I_C \ unionRegiones
   END IF
END FOR
```

`lineaDivision` es la línea central de píxeles a lo largo de la dimensión más grande

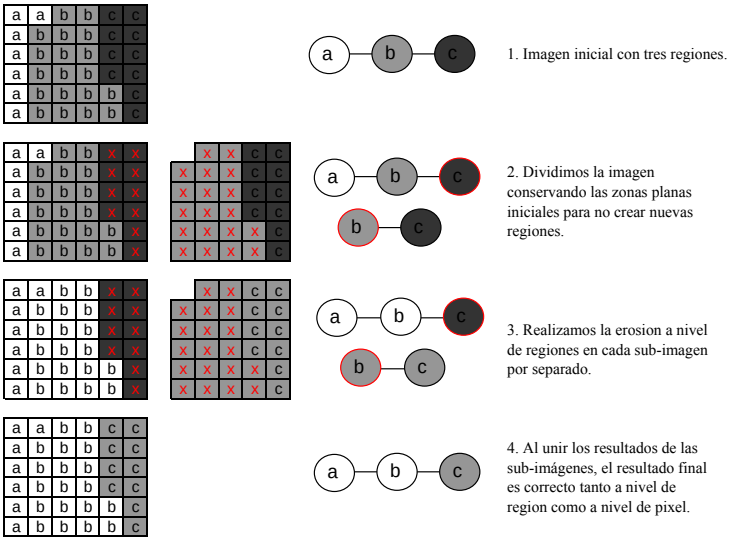

Figura 3.7: División a nivel de región de una imagen y erosión distribuida

de la imagen.

`obtenerRegionesSimilares` aumenta la región inicial, con aquellos píxeles adyacentes que son similares al recibido como parámetro. Formalmente, dos píxeles a y b son similares si cumplen las tres expresiones siguientes:

$$\text{máx}(|r_a - r_b|, |g_a - g_b|, |b_a - b_b|) < th_{max} \tag{3.2}$$

$$\text{mín}(|r_a - r_b|, |g_a - g_b|, |b_a - b_b|) < th_{min} \tag{3.3}$$

$$\frac{|r_a - r_b| + |g_a - g_b| + |b_a - b_b|}{3} < th_{avg} \tag{3.4}$$

Donde: (r_a, g_a, b_a) y (r_b, g_b, b_b) denotan los valores RGB de las regiones a y b, respectivamente. El algoritmo de división obtiene regiones por crecimiento de semillas ayudándose de una pila al igual que en (Brunner y Soille, 2005). Los píxeles se exploran en profundidad, no en anchura. Esto se hace así para acceder al fichero de la imagen lo más secuencialmente posible.

Como se ha comentado anteriormente, se necesita la información de vecindades

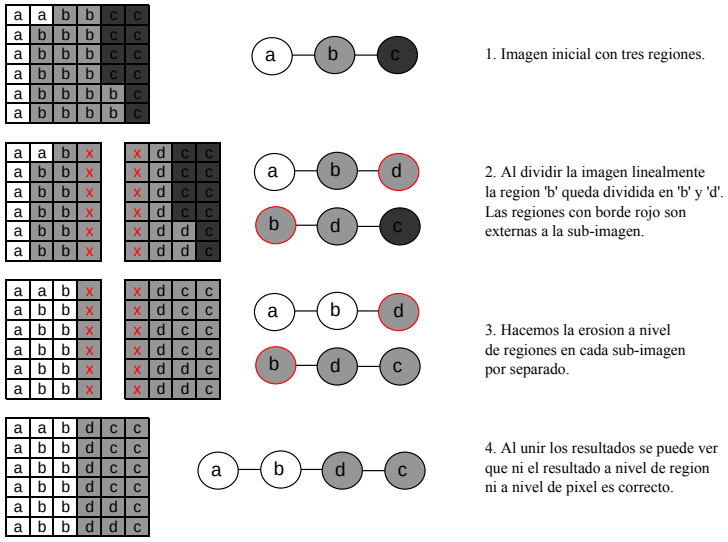

Figura 3.8: División a nivel de región de una imagen y erosión distribuida

de los píxeles I_A e I_C que delimiten con la sub-imagen procesada I_B. Por lo que además de los píxeles sin tratar se añade parte de la sub-imagen I_B para ser enviada junto con las sub-imágenes I_A e I_C. De esta manera, cuando estas dos sub-imágenes sean procesadas se obtendrán sus vecindades con las regiones pertenecientes a la sub-imagen central.

Para evitar problemas en el nombrado de las regiones de la imagen, las etiquetas que se escojan para cada región debe ser única en toda la imagen. Para este fin, las regiones de las imágenes son etiquetadas con el *id* de la sub-imagen dentro del sistema seguido del *id* numérico local que recibe la región al ser procesada la imagen.

Es de destacar que un aspecto importante del algoritmo, es que la imagen se recorre solamente una vez para ser procesada y almacenada en la base de datos. Esta característica es muy beneficiosa debido al gran tamaño de las imágenes que se van a tratar.

3.2.2. Distribución de la estructura de datos para almacenamiento de imágenes de gran tamaño

Gracias al algoritmo de división creado, una imagen de entrada puede ser dividida para ser procesada y almacenada en distintos ordenadores. De la misma manera, se ha modificado la estructura de datos explicada en la sección 3.1 para que se permita almacenar distintas sub-imágenes de una misma imagen en varias estructuras de datos de forma distribuida. Es decir una imagen podrá estar guardada distribuidamente en varias bases de datos, porque, como se ha comentado, se usa un Sistema Gestor de Bases de Datos Relacionales para implementar la estructura de datos.

La estructura de datos debe permitir tanto el almacenamiento distribuido, como el procesamiento de la imágenes contenidas también de forma distribuido y la recuperación de las regiones dichas imágenes.

Para que el procesamiento de las imágenes se realice sobre la estructura de datos de una forma óptima, se ha de mantener información de vecindad de las regiones de cada imagen. Por ello cada sub-imagen deberá mantener información actualizada de las regiones que están fuera de ella y son limítrofes con alguna región que le pertenece. Para lograr este objetivo se ha ampliado el esquema de la base de datos añadiendo una tabla para la información externa de la sub-imagen, guardando en ella el agente donde se encuentra guardada cada región externa.

3.2.3. Distribución de los métodos de tratamiento de las imágenes de gran tamaño

Como se ha comentado en apartados anteriores las estructura de datos definida como base para el sistema es de tipo grafo y está centrada en las regiones. Por lo tanto, las operaciones de Morfología Matemática que se han implementado son operaciones basadas en grafos (Serra, 1982) (Serra, 1988) (Soille, 2003). Para dichas operaciones se utilizan las vecindades básicas del grafo, es decir, una región y sus regiones colindantes.

Todos los métodos y operaciones se han implementado para trabajar directamente sobre la estructura de datos. Es decir, sobre la base de datos donde se encuentra implementada. Por lo tanto cada operación sobre las imágenes es tra-

ducida a un conjunto de código SQL. Esta manera de implementar las operaciones consigue que no sea necesario extraer las imágenes ni las sub-imágenes de la base de datos para aplicarle algún filtro u operación.

Teniendo en cuenta que la estructura de datos se ha modificado para ser usada de forma distribuida, para aplicar un filtro u operación sobre una imagen hay que aplicarlo sobre todas las sub-imágenes que la componen. Esto es, hay que aplicar la operación o filtro en aquellos agentes donde se encuentren almacenadas las sub-imágenes en el sistema. Por lo que varios agentes y sus bases de datos han de ser consultados y cada agente obtiene un sub-resultado de la operación. De forma habitual los agentes deben interactuar para poder actualizar información de las distintas vecindades de las regiones que son externas a su propia sub-imagen.

3.2.3.1. Erosión y Dilatación

Como es sabido una dilatación de grafo para una región dada (o una erosión, respectivamente) con tamaño de vecindad 1 debe obtener el máximo (el mínimo, respectivamente) del valor de la región y todas sus regiones vecinas.

El siguiente código SQL realiza una erosión sobre la media de color de cada canal de la imagen. Dichos descriptores son parte de los disponibles en las imágenes almacenadas. No obstante, se podría usar cualquiera de los descriptores de color de las regiones que se encuentran almacenados en la estructura de datos, como por ejemplo, el máximo de la región. Los descriptores disponibles en la estructura de datos están comentados en la tabla 3.1

Para explicar cómo funcionan las consultas sobre la estructura de datos cabe explicar que la tabla *Vecindad* incluye las relaciones de vecindad del grafo de la imagen, y la tabla *DescriptoresColorRegion* incluye todos los descriptores y características de color de las regiones de una imagen.

El resultado de esta consulta genera un registro para cada región de la imagen y los nuevos valores de sus tres bandas de color, conteniendo como resultado el mínimo de entre todos sus vecinos y la misma región.

```
INSERT INTO "Erosion__ImagenEntrada"
SELECT  Vec.idRegion as idRegion,
   MIN(Regs.MediaValorRojo) as nuevoValorRojo,
   MIN(Regs.MediaValorVerde) as nuevoValorVerde,
```

```
   MIN(Regs.MediaValorAzul) as nuevoValorAzul
FROM Vecindad Vec
JOIN DescriptoresColorRegion Regs
  ON Vec.idVecino = Regs.idRegion
     OR Vec.idRegion = Regs.idRegion
WHERE Vec.imagen='ImagenEntrada'
GROUP BY Vec.idRegion
```

Para obtener una dilatación bastaría con cambiar las funciones de agregado MIN de la consulta anterior por la función de agregado MAX, que obtiene los valores máximos.

A efectos prácticos, en el sistema los registros generados por la consulta son almacenados en una tabla temporal de la base de datos para poder ser consultados posteriormente o utilizados para otros filtros u operaciones.

3.2.3.2. Apertura y Cierre

Las operaciones de apertura y cierre son junto con la erosión y la dilatación, las operaciones más representativas de la Morfología Matemática. Como se comentó en la sección 2.2.1, la apertura morfológica es una erosión seguida de una dilatación; y un cierre morfológico es una dilatación seguido de una erosión. Gracias a esta definición, la implementación de estas dos operaciones es inmediato, en tanto ya tenemos definidas las operaciones de erosión y dilatación morfológicas.

Existe un pequeño problema que viene dado por la naturaleza distribuida de los métodos creados. Cuando se aplica una erosión o dilatación, cada agente obtiene como resultados la erosión o dilatación de las regiones pertenecientes a las sub-imágenes almacenadas en el propio agente. Pero, para poder aplicar otra operación sobre dichos resultados necesitamos actualizar los valores para las regiones vecinas de dichas sub-imágenes, que pertenecen a otras sub-imágenes fuera de el agente. Así pues es necesario un paso de sincronización para obtener los valores de las regiones vecinas que son externas a dicho agente. Tras este paso de sincronización se podría aplicar la siguiente operación para completar la apertura o cierre morfológico.

3.2.3.3. Watershed

Dentro de la multitud de métodos de segmentación que existen, la watershed es el método más conocido y representativo (Meyer y Beucher, 1990) (Soille, 2003) dentro de la Morfología Matemática. Por esta razón la paralelización de la watershed basada en regiones ha sido estudiada en numerosas ocasiones (Meijster y Roerdink, 1995) (H. Zhou y Yang, 2002) (Zhou, Yang, Tang, y Xiao, 2004). Todos estos autores se centran en la paralelización y no en la distribución del algoritmo de la watershed. Es decir, el algoritmo se ejecuta de forma paralela en varios procesadores de un mismo ordenador y teniendo una memoria compartida. Esto implica que la información de las regiones y sus vecindades en un algoritmo paralelo es compartida por los distintos procesadores.

En el caso del procesamiento distribuido cada ordenador tiene sólo parte de la información, esto es un sub-dominio de la imagen, y se necesita tener información actualizada de las regiones o sub-dominios colindantes. Este requisito para el procesamiento distribuido se logra gracias a la estructura de datos explicada en la sección 3.1 y al algoritmo de división explicado en 3.2.1, ya que dicha información existe y está actualizada en las distintas sub-imágenes.

A continuación explicaremos en detalle el algoritmo distribuido para la watershed basada en regiones.

Tras tener la imagen distribuida, se procede a realizar el algoritmo de inundación por separado en sub-dominio. Este paso se conoce como inundación local. En este paso hay regiones para las que no se puede calcular su etiqueta porque necesitan información de otros sub-dominios.

A diferencia de otros operadores, la watershed tiene una naturaleza iterativa. Debido a esta naturaleza, es posible que, en cada iteración n de la inundación, la información externa al sub-dominio debe actualizarse, pues es necesaria para la iteración $n + 1$ del algoritmo. Si alguna región tiene una región vecina que está fuera del sub-dominio, dicho valor debe estar actualizado para poder calcular correctamente el valor de las etiquetas.

El paso de inundación local se ejecuta de forma distribuida e independiente en cada sub-dominio. El siguiente pseudo-código describe dicho algoritmo de :

```
WHILE (NOT isEsVacia(colaJerarquica))
  cRegion = siguiente(colaJerarquica)
  labels[] = getEtiquetasVecinos(cRegion)
  IF (size(labels)=1) THEN
    setEtiqueta (cRegion,labels[0])
  ELSE
     IF contieneEtiquetasTemporales(labels) THEN
        setEtiqueta (cRegion,U)
     ELSE
        marcarWatershed (cRegion)
     END IF
  END IF
  IF (tieneVecinosExternos(cRegion)) THEN
    actualizarInformacion(cRegion)
  END IF
  candidatos = getVecinosSinTratar(cRegion)
  insertarEnCola (colaJerarquica,candidatos)
END WHILE
```

Inicialmente, `colaJerarquica` se inicializa con las regiones mínimas locales. Estas regiones mínimas locales son aquellas cuya etiqueta es menor que la de todos sus vecinos. A la hora de calcular los mínimos locales se han de tener en cuenta todas las regiones vecinas de una región, incluyendo aquellas regiones que están fuera del sub-dominio donde se está calculando la inundación. Aquellos mínimos locales que sean exteriores al sub-dominio son marcados con una etiqueta temporal quedando a la espera de saber su valor definitivo, ya que el valor final de dichas regiones se calculará en su sub-dominio correspondiente.

Las regiones que son procesadas en cada iteración se obtienen de la `colaJerarquica`, usando la función `siguiente`, que devuelve aquellas regiones cuyo valor de etiqueta sea menor.

Las regiones son etiquetadas sólo en el sub-dominio en el que pertenecen. Cuando una región es etiquetada con una etiqueta permanente, si tiene regiones vecinas fuera de su sub-dominio, la nueva etiqueta de la región es propagada a aquellos sub-dominios donde existan regiones vecinas a dicha región. Esto se efectúa usando la función `actualizarInformacion`.

La función `getEtiquetasVecinos` devuelve todas las etiquetas diferentes de los vecinos de la región dada, a excepción de las etiquetas de watershed. Las etiquetas de watershed no son devueltas, porque no han de ser propagadas. A la hora de evaluar una región para etiquetarla pueden ocurrir tres casos:

(a) En el caso que sólo exista una única etiqueta entre todos los vecinos procesados de la región, entonces se etiqueta dicha región con el valor de esa única etiqueta.

(b) Cuando una región puede ser inundada por diferentes etiquetas y una o más de ellas es una etiqueta temporal (es decir, que pertenece a otro sub-dominio), entonces se etiqueta la región con la etiqueta de indeterminado 'U'. Esta etiqueta 'U' quiere decir que no se puede determinar el valor de inundación de esa región sin información adicional de otros sub-dominios. Aquellas regiones etiquetadas con 'U' se re-evaluarán cuando la información de todos los sub-dominios adyacentes se haya recibido.

(c) Si los vecinos de una región poseen más de una etiqueta definitiva (no temporal), entonces la región se etiquetará como watershed. Es decir con la etiqueta 'W'.

Por lo tanto, cuando en un sub-dominio, se han actualizado todas las etiquetas temporales, se puede re-evaluar aquellas regiones etiquetadas con la etiqueta 'U', y determinar así su watershed completa.

La actualización en nuestra estructura de datos las etiquetas temporales con la información recibida de otro sub-dominio es relativamente sencilla, debido a que está implementada usando una base de datos. Por lo que una actualización de una etiqueta temporal en n regiones se traduce a una simple consulta *UPDATE* de SQL.

Con respecto a los algoritmos paralelos citados anteriormente, en (H. Zhou y Yang, 2002), para dividir la imagen, ésta se escanea dos veces: la primera vez para obtener las regiones, y una segunda para dividir las regiones en distintos sub-dominios. Gracias al algoritmo de división de imágenes que hemos desarrollado, en nuestro caso la imagen se distribuye en distintas sub-imágenes, guardando las regiones y sus descriptores, de tal manera que la imagen se recorre una única vez. Para la aplicación de la watershed desarrollada con imágenes de gran tamaño es muy conveniente minimizar el número de iteraciones sobre los píxeles de la imagen.

3.2.4. Recuperación de información de imágenes de gran tamaño

Un método básico e imprescindible para cualquier estructura de datos es el de consulta y extracción de la información. Como ya se ha comentado, se han creado métodos para realizar búsquedas por contenido sobre las regiones de las imágenes almacenadas. Para realizar estas búsquedas se puede usar la información de descriptores de región y de relaciones de las regiones almacenados 3.1 en la estructura de datos.

Debido a la característica distribuida de la estructura de datos al realizar una búsqueda se obtendrán resultados de diferentes nodos. Por ello, es necesario mezclar los sub-resultados provenientes de cada nodo para obtener los resultados finales.

De una manera similar al sistema de integración de bases de datos basado en ontologías: Ontofusion (Pérez-Rey y cols., 2006) (Alonso-Calvo y cols., 2007), una consulta sobre una sola imagen almacenada en la estructura de datos distribuida (por ejemplo la obtención de una ventana dentro de una imagen), obtendrá información de diferentes nodos.

En el caso concreto del presente trabajo, la integración de los esquemas de bases de datos distribuidos usa un esquema conceptual virtual único ya que todas las bases de datos en el sistema poseen el mismo esquema, el de la estructura de datos explicado en la sección 3.1.

De igual modo no es necesario traducir los resultados provenientes de las diferentes fuentes de datos, ya que todos son almacenados con la misma escala de valores. En consecuencia, para obtener el resultado final de una consulta sobre una imagen distribuida en la estructura de datos, (i) la consulta ha de ser dividida y enviada a aquellos nodos que contengan información de dicha imagen, y (ii) el resultado final será el la unión de los sub-resultados obtenidos de los distintos nodos.

3. MÉTODOS DISTRIBUIDOS DE TRATAMIENTO DE IMÁGENES

Capítulo 4

Creación de un sistema distribuido para imágenes de gran tamaño

Para poder manejar la estructura de datos distribuida y poder almacenar todas las sub-imágenes que una imagen inicial dada pueda contener, se ha desarrollado un sistema distribuido. El sistema, explicado en este capítulo 4, es capaz de mantener los distintos sub-dominios en los que la imagen inicial es dividida e interconectar por medio de mensajes esos sub-dominios. Dicho sistema está internamente implementado usando agentes colaborativos, cuyo diseño, comportamiento e interacción se comentan en la secciones 4.1, 4.2 y 4.3.

En la Figura 4.1 se muestra el esquema del sistema que se ha creado.

En las sub-secciones que siguen se explican los distintos roles o tipos de agentes existentes en el sistema en la sección 4.1. En la sección 4.2 se comenta cómo las tareas son asignadas y ejecutadas por dichos agentes en el sistema usando un planificador y un balanceo de carga creado específicamente para las necesidades del sistema durante la investigación de la presente tesis doctoral.

La implementación del sistema multi-agente se ha realizado usando el lenguaje de programación Java de *Sun Microsystems*. Gracias a las características de Java los agentes se pueden ejecutar sobre máquinas con hardware y sistema operativo heterogéneos. Por otro lado, el sistema puede ser configurado para utilizar sistemas gestores de bases de datos heterogéneos.

Así mismo, para facilitar el uso de dicho sistema a terceros se ha implementado

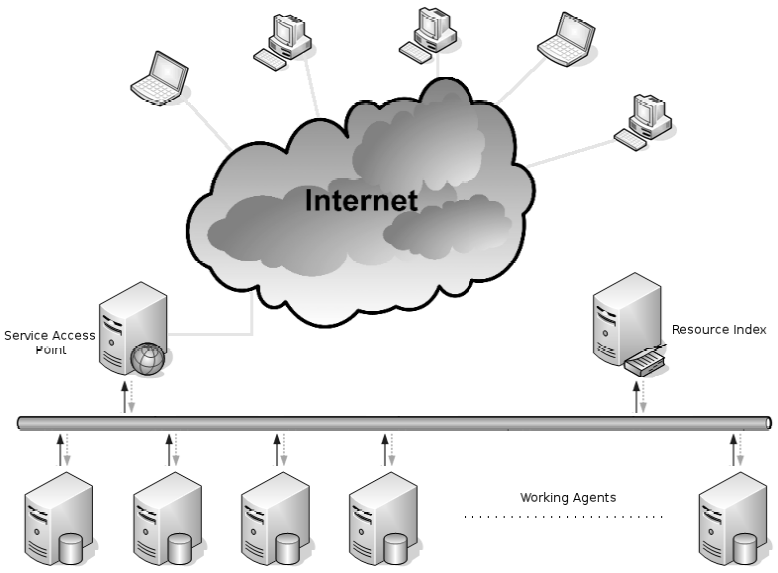

Figura 4.1: Esquema del Sistema de Agentes Distribuido para almacenamiento y tratamiento de imágenes

un Servicio de *Cloud Computing* como se comentará en la sección 4.4. Un ejemplo de cliente que usa dicho servicio de *Cloud Computing* puede verse en el anexo A.

4.1. Roles de los agentes

El prototipo del sistema se ha implementado usando el paradigma de orientación a agentes. Se pueden distinguir tres roles o tipos de agentes principales mostrados en la figura 4.1:

- El *agente punto de acceso al servicio de Cloud Computing*. Es un agente único en el sistema. Es el punto de entrada para acceder al sistema por parte de los usuarios autorizados. Este agente ofrece métodos a los usuarios finales, para (a) almacenar imágenes en el sistema, (b) aplicar operaciones sobre las imágenes almacenadas, y (c) consultar las imágenes del sistema.
- El *agente índice de recursos*. Es un agente único en el sistema. Puede considerarse como un servicio de directorio para el sistema. Este agente contiene información acerca de la localización, carga computacional actual, estado

del procesamiento de las imágenes y predicción de tiempos de ejecución de todos los agentes existentes en el sistema.

- Los *agentes trabajadores*. Son la parte principal o núcleo del sistema. Estos agentes implementan todas las funcionalidades y métodos ofrecidos por el sistema. Cuando reciben una imagen, comprueban si hay que dividirla o procesarla, realizando la operación necesaria. Son capaces de manejar la base de datos que implementa la estructura de datos distribuida. Estos agentes además pueden almacenar, filtrar y consultar las regiones de las imágenes usando la estructura de datos que se encuentra en la base de datos.

  Cada agente trabajador tiene la capacidad de examinar el sistema físico donde se está ejecutando. Usando estas medidas del sistema físico, cada agente puede calcular la carga de dicho sistema y así puede estimar el tiempo necesario que supondrá realizar las tareas que tiene pendientes. Esta recogida de datos por parte de los agentes del entorno en el que se ejecutan, es vital para realizar el balanceo de carga del sistema. El balanceo de carga se explica en profundidad en la sección 4.2.

Adicionalmente, estos agentes poseen funcionalidades para comunicarse con otros agentes trabajadores para enviar sub-imágenes para ser procesadas en otros agentes. También se comunican para actualizar la información de la vecindad de las regiones almacenadas en su base de datos.

4.2. Balanceo de carga de las tareas

El balanceo de carga es una técnica utilizada en sistemas distribuidos para reducir el tiempo de ejecución de las tareas. Son estrategias de distribución de las tareas entre los diferentes nodos que componen el sistema distribuido. Los beneficios de su utilización en sistemas muti-agente y grid han sido ampliamente estudiados (Leinberger, 2000)(Cao, Spooner, Jarvis, y Nudd, 2005)(Yagoubi y Sulimani, 2007).

En concreto para nuestro sistema, se ha desarrollado una estrategia de balanceo de carga para poder reducir el coste del procesamiento inicial y el almacena-

miento de las imágenes en las bases de datos administradas por los agentes que componen el sistema. Su implementación se ha basado en tres conceptos básicos:

(i) La *publicación y descubrimiento de servicios* (service advertisement and discovery). Es el índice del sistema. Los diferentes agentes en el sistema lo usan para localizar otros agentes y para registrar su propia localización.

(ii) La *predicción de rendimiento*. Es la herramienta utilizada para asignar las nuevas tareas que entran en el sistema entre los agentes. Es necesaria para equilibrar y optimizar los tiempos de ejecución de los agentes. Cada agente registra sus tiempos de ejecución clasificándolos por el tipo de tarea, para poder estimar de forma probabilística el tiempo necesario para una tarea nueva. Por supuesto no todas las tareas requieren el mismo tiempo, aunque sean del mismo tipo. Por lo tanto se necesita almacenar también algunos parámetros de las tareas que inciden directamente sobre el tiempo de ejecución como pueden ser: el número de píxeles de la sub-imagen o el número de regiones iniciales.

(iii) Las *colas locales de tareas* empleadas por los agentes para almacenar las tareas pendientes y usadas para la planificación de dichas tareas. Estas colas son necesarias porque cuando un agente recibe una tarea puede que se esté ejecutando otra recibida anteriormente.

Gracias a estos tres conceptos se puede realizar la planificación de tareas y el balanceo de carga en el sistema. Dicha implementación se encuentra dentro de los *agentes trabajadores* del sistema. Vamos a explicar en detalle el funcionamiento e implementación de estos agentes 4.1 en el paso de tratamiento y almacenamiento de una imagen. Este proceso está ilustrado en la figura 4.2.

En el momento que un *agente trabajador* recibe una imagen sin procesar, es capaz de comprobar si el número de píxeles sin procesar imagen es mayor o menor que el umbral fijado en el algoritmo de división 3.2.1.

En el caso de que el número de los píxeles sin procesar en la imagen sea mayor al umbral para la división, la imagen es procesada extrayendo las regiones que la componen y las características y relaciones de dichas regiones. Posteriormente la imagen procesada es enviada a un agente trabajador para ser almacenada.

En el caso de que el número de los píxeles sin procesar en la imagen sea mayor al umbral para la división, la imagen es dividida, obteniendo así una sub-imagen procesada y las regiones que contiene (I_B) y otras dos sub-imágenes que siguen

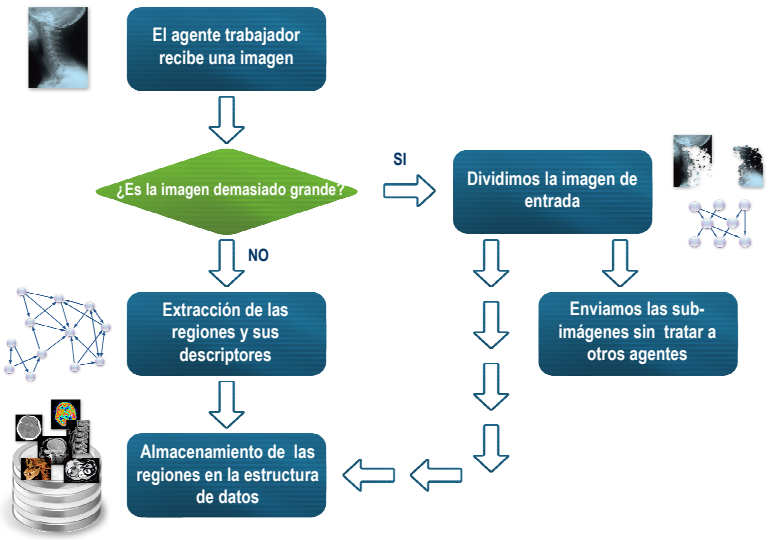

Figura 4.2: Algoritmo de procesamiento y almacenamiento de una imagen por los agentes trabajadores

sin estar procesadas I_A e I_C. I_A e I_C se envían a otros agentes para ser procesadas y almacenadas. Mientras I_B es enviada a otro agente para ser almacenada.

Por lo tanto como se puede suponer existen dos tipos de balanceo de carga implementados en los agentes trabajadores. El primero es el encargado de encontrar al agente en el sistema con menos carga para procesar una imagen. El segundo tipo es para encontrar el agente que va a almacenar una imagen ya procesada.

- Balanceo de carga en el procesamiento de imágenes. Se usa en el sistema para minimizar el tiempo de procesamiento de las imágenes introducidas. Para ello se usa el parámetro de *número de píxeles sin procesar* de las sub-imágenes recibidas. Gracias a este parámetro se consigue estimar el tiempo de ejecución que se necesita para procesar la sub-imagen. De esta manera también se puede estimar la carga pendiente de los agentes trabajadores y así seleccionar al agente con menos carga de trabajo.

  Internamente los agentes trabajadores se componen principalmente de dos hilos o threads como se puede apreciar en la figura 4.3.

  (i) El *hilo de planificación* contiene la cola local de tareas pendientes de eje-

Figura 4.3: Composición interna de un agente trabajador

cutar por el agente, y las tareas que están actualmente siendo ejecutadas. El número de tareas en ejecución de forma concurrente es configurable, y es controlado por este thread. Cuando el agente recibe una tarea nueva, el planificador la introduce en la cola de tareas pendientes de ejecución. Cuando es posible ejecutar una nueva tarea de las que se encuentran almacenadas en la cola, el hilo planificador crea un nuevo hilo de java para ejecutar la tarea.

El planificador almacena la información relativa a la ejecución de cada tarea. Esto es, almacena la fecha de comienzo, el número de píxeles de la imagen a procesar y la fecha de finalización de la tarea. Gracias a estos datos almacenados, un agente puede estimar el tiempo de ejecución necesario para completar una tarea dada. Se utiliza una función de regresión logística, donde el parámetro de entrada, como se ha comentado anteriormente, es el *número de píxeles sin procesar* de las sub-imagen, y los valores de la función de regresión son calculados gracias al historial de ejecución de tareas del agente. De esta manera se puede calcular el tiempo estimado para ejecutar una tarea nueva en los agentes trabajadores, y por lo tanto la

carga del agente (teniendo en cuenta las tareas pendientes encoladas).

Cuando la cola de tareas pendientes de un agente cambia, porque se ha completado una tarea o bien porque le llega una nueva, el hilo planificador del agente trabajador actualiza la información comunicando al *agente índice de recursos* el tiempo estimado que le queda al agente para terminar todas sus tareas pendientes. De esta manera cuando un agente necesita enviar una tarea (una sub-imagen nueva para ser procesada por un agente trabajador), se consulta al agente índice de recursos para obtener al agente menos ocupado en ese instante.

(ii) El *hilo de monitorización de carga*, es el responsable de registrar el consumo por parte de los hilos de Java de los recursos físicos del ordenador al ejecutar una tarea. Se registran sólo aquellos hilos que ejecutan tareas de procesamiento de imágenes.

En la implementación de los hilos de la máquina virtual java de Sun todos los hilos se ejecutan dentro del proceso de la máquina virtual. Para poder registrar el consumo de CPU por cada hilo que ejecuta una tarea se ha utilizado Sun JMX Beans. El registro de este consumo de recursos por cada hilo ayuda enormemente a refinar la función de estimación de los tiempos de ejecución para una tarea.

- Balanceo de carga de las regiones en el almacenamiento de imágenes

  Para reducir el tiempo de ejecución para aplicar un filtro u operación sobre las imágenes almacenadas es importante que las regiones de la imagen sean distribuidas de la forma más homogénea posible. Esto es debido a que tanto los filtros como las operaciones se aplican sobre las imágenes en el sistema, directamente sobre la estructura de datos almacenada en la base de datos. Por lo que si un agente posee muchas mas regiones de una imagen dada, que el resto de agentes, el tiempo de ejecución en dicho agente será mayor que en el resto, ya que el tiempo de ejecución es, en general, directamente proporcional al número de regiones en el agente. Igualmente si las regiones se distribuyen de forma homogénea entre los agentes del sistema, el tiempo de consulta sobre las imágenes se reduce también, siendo similar en todos

los agentes que contienen la imagen.

El principal escollo a salvar para distribuir de forma homogénea las regiones de una imagen, es que el número total de las regiones que contiene una imagen no se puede saber a priori, al contrario de lo que ocurre con el número de píxeles sin tratar. Por este motivo cuando un agente inserta las regiones de una sub-imagen en su estructura de datos local, actualiza, en el índice de recursos, el número de regiones de la imagen que están almacenadas en dicho agente. De esta manera, cuando se va a enviar una sub-imagen ya procesada proveniente del algoritmo de división, se elegirá aquel agente que contenga el menor número de regiones de la misma imagen.

En resumen, el balanceo de carga implementado en el sistema explota los beneficios de usar un índice de recursos centralizado, manteniendo la información del sistema actualizada y disponible para todos los agentes. A la vez se reduce el tráfico de paquetes de red (el mayor inconveniente de esta técnica) siendo los agentes trabajadores los encargados de asignar las nuevas tareas a otros agentes trabajadores gracias a las colas locales de tareas pendientes. Como añadido, las colas locales permiten el re-balanceo de tareas cuando un agente queda muy ralentizado, ya que puede reenviar sus tareas pendientes a otros agentes trabajadores.

4.3. Mensajería entre agentes

La comunicación entre los distintos agentes software es una pieza básica en la creación de un sistema multi-agente. Debido a que los agentes pueden estar situados en sitios remotos se ha desarrollado el servicio de mensajería a través de Servicios Web de Java.

Los mensajes que usa el sistema se pueden dividir en cinco tipos de mensajes diferenciados:

- *Mensajes de operación.* Usados para aplicar un filtro u operación sobre una imagen, como pueden ser las explicadas anteriormente, erosión, dilatación, watershed, o el borrado de una imagen del sistema. Estos mensajes son originados por un usuario y el *agente punto de acceso al servicio de Cloud*

Computing traslada la petición a todos los agentes trabajadores que poseen información de la imagen afectada.

- *Mensajes de recuperación.* Este tipo de mensajes es generado también por el usuario y recibe una serie de parámetros de búsqueda devolviendo como resultado un fichero binario, ya puede ser con partes de una imagen, o con nombres de una imagen que cumple los parámetros de entrada.

- *Mensajes de actualización o sincronización de regiones.* Gracias a estos mensajes la información externa contenida en cada agente puede ser actualizada correctamente, después de realizar operaciones sobre las regiones. Ya que cuando una operación modifica una región que tiene alguna región vecina fuera de su sub-dominio o agente, se manda un mensaje al agente remoto con la nueva información de la región modificada.

  Este tipo de mensajes no esperan ninguna respuesta, pero sí que transmiten un fichero binario de datos con la nueva información.

  Este tipo de mensajes son automáticos, es decir, son generados por las operaciones y filtros no necesitan la intervención del usuario.

- *Mensajes de administración de agentes.* Se utilizan para obtener información de los agentes o actualizar su estado en el *agente índice de recursos* para poder efectuar el balanceo de carga. No actúan sobre la estructura de datos y se generan por los propios agentes cuando reciben o completan tareas.

- *Mensajes de inserción de una imagen.* Este tipo especial de mensajes es generado por los agentes cuando procesan (o dividen) una imagen. Gracias al uso del balanceo de carga, estos mensajes se envía una sub-imagen procesada para almacenarla en la estructura de datos a aquel agente menos ocupado en ese momento. En el caso de la división, se envían las sub-imágenes que quedan por procesar a otro agente para que las procese.

4.4. Implementación de un servicio de Cloud Computing

Los métodos desarrollados y explicados en los apartados anteriores 3.2 y 3.2.3 han dado lugar a la creación de un sistema multi-agente distribuido como se explica en la sección 4. Dichos métodos son útiles para el procesamiento y almacenamiento de imágenes muy grandes. El almacenamiento y procesamiento de estas imágenes necesita sistemas complejos y grandes recursos computacionales.

Por otra parte, el *Cloud Computing*, según la definición hecha por el NIST (NIST, 2010), es una herramienta para ofrecer recursos virtuales bajo demanda. Dichos recursos virtuales pueden ser físicos o servicios de software y son controlados por el propietario del servicio.

Por ello se ha implementado un servicio software de *Cloud Computing* para que pueda ser usado por profesionales e instituciones que no posean los suficientes recursos técnicos, capacidad de almacenamiento o de computación. Gracias a este servicio software de *Cloud Computing* se pueden crear aplicaciones basadas en el sistema desarrollado.

El servicio creado es privado, y sólo puede ser accedido por clientes con cuenta en el sistema y que posean autorización de acceso. Un cliente puede introducir y almacenar imágenes en el sistema. Dichas imágenes son privadas y sólo accesibles por el propietario de las mismas. Además, el servicio ofrece métodos para aplicar los operadores distribuidos desarrollados y explicados en la sección 3.2.3. Los usuarios tienen la posibilidad de hacer públicas sus imágenes si lo desean, para compartirlas con otros usuarios.

Usando este servicio software, un usuario u organización podría crear su propio CBIR privado de una manera relativamente sencilla y sin necesidad de costosos recursos informáticos.

Como se comenta en la sección 2.3, la implementación de un servicio software de *Cloud Computing* debe ser escalable y eficiente. De manera que dicha implementación permita reducir los costes de procesamiento de tareas complejas y sea fácilmente ampliable. El sistema desarrollado cumple dichos requisitos, ya que está implementado mediante un sistema multi-agente distribuido y balanceado. Estas características del sistema permiten escalar el sistema simplemente añadien-

do nuevos agentes, como ya veremos en la sección de resultados y conclusiones 5.1.

Capítulo 5

Resultados

En el presente capítulo se muestra, usando como ejemplo algunas imágenes grandes de distintos tamaños, el funcionamiento de los métodos y algoritmos creados.

En primer lugar, en la sección 5.1 se muestran pruebas de la obtención de regiones de una imagen y almacenamiento de dichas regiones en el sistema. La sección está dividida en varias partes:

- En la sección 5.1.2 se estudia la selección del valor óptimo para el valor umbral del algoritmo de división. También se muestra la manera en que éste valor influye en el rendimiento de la extracción de regiones de una imagen.

- La sección 5.1.3 explica con un ejemplo básico el funcionamiento del algoritmo de división basado en regiones usado en el sistema.

- En la siguiente sección, 5.1.4, se muestran el comportamiento del prototipo del sistema al variar el número de agentes en el sistema para el almacenamiento de las imágenes de ejemplo.

Las secciones 5.2.1 y 5.2.2 muestran los resultados de aplicar una erosión morfológica sobre imágenes almacenadas en el sistema y la ejecución de una consulta de recuperación de información sobre las mismas imágenes, respectivamente.

Por último, en la sección 5.3 se resumen y explican los resultados mostrados en las secciones 5.1.4, 5.2.1 y 5.2.2 del presente capítulo.

5.1. Almacenamiento de imágenes

5.1.1. Bases del algoritmo de extracción de regiones

El novedoso algoritmo de división y extracción de regiones, que se ha desarrollado, ha sido diseñado con el propósito de ser utilizado para procesar imágenes de gran tamaño. Por esta razón su principal característica es que no carga el contenido de la imagen procesada en la memoria principal. Por ejemplo si quisiéramos procesar una imagen de 21.601×10.801, el tamaño de dicha imagen en memoria sería de 890 MB. Y si quisiéramos procesar una imagen muy grande, de 86.400×43.200 a color, su tamaño en memoria (tomando 4 bytes por píxel) sería de 11,8 GB. Por lo tanto, se hace inviable el uso de las estructuras comúnmente usadas en herramientas y librerías de visión artificial como *openCV* (Intel, 2011) o *CVIPtools* (siue.edu, 2011). Ya que estas herramientas almacenan la imagen completa en memoria para procesarla.

Para solventar los problemas expuestos en el párrafo anterior debidos al tamaño de las imágenes, sólo se almacenan en memoria las regiones obtenidas, sus características, relaciones y una representación raster de la región, así como algunas estructuras de datos necesarias para procesar la imagen. Pero el acceso a la información de la imagen se hace por *streaming* sobre el fichero almacenado directamente en el disco duro.

Normalmente, para procesar imágenes de gran tamaño, se divide la imagen sub-imágenes que son procesadas separadamente. En nuestro caso, como se explica en la sección 3.2.1, al tratarse de procesamiento a nivel de regiones, no podemos dividir la imagen de forma lineal 'ciega' ya que se podría dividir alguna de las regiones iniciales y por lo tanto se crearían nuevas regiones.

El algoritmo creado se puede ver ralentizado comparándolo con otros algoritmos que usan bibliotecas de visión artificial para imágenes, ya que para obtener los píxeles de la imagen se accede al fichero en el disco, porque no se mantiene la imagen en memoria principal. Pero como se ha comentado anteriormente esta característica es necesaria para la extracción de regiones de imágenes muy grandes.

La tabla 5.1 muestra los tiempos necesarios para extraer la información, y escribir en un fichero de datos todas las regiones pertenecientes a imágenes de

distintos tamaños. Para las pruebas se ha limitado la pila de ejecución a un tamaño máximo de 2GB.

Tamaño	PNG Regiones	PNG Tiempo	JPG Regiones	JPG Tiempo
9.000.000	248.851	127,4	5.338.954	OutOfMemory
4.000.000	221.028	66,57	2.790.473	OutOfMemory
3.000.000	79.125	33,23	1.937.468	2.709
2.000.000	48.861	24,19	1.437.544	1.511
1.500.000	18.298	16,22	1.129.317	163,4
1.000.000	14.843	10,8	726.259	94,25

Tabla 5.1: Comparación de tiempos por tipo de imágenes

Otra de las características intrínsecas del algoritmo es que está orientado a regiones. En consecuencia, el algoritmo es más efectivo con imágenes cuyo número de regiones es bajo independientemente de su tamaño en píxeles. Y como se puede apreciar en la tabla 5.1 para aquellas imágenes con un gran número de regiones como pueden ser imágenes JPG, donde el algoritmo de compresión genera muchas regiones en los bordes, su funcionamiento requiere mucha memoria (ya que las regiones sí se almacenan en memoria). En el apartado de futuras líneas de trabajo se comentan algunas posibles mejoras del algoritmo para reducir el tiempo necesario para procesar este tipo de imágenes.

5.1.2. Configuración del umbral del algoritmo de división

Como se comentó en la sección 3.2.1, el algoritmo de división que se ha desarrollado, posee un parámetro configurable, que es el valor umbral para la división de las imágenes. Dicho algoritmo divide las imágenes recibidas si el número de píxeles sin tratar que contiene la imagen es mayor que este valor umbral. En esta sección vamos a ver las consecuencias sobre el rendimiento del algoritmo que se producen al variar dicho valor umbral.

Para realizar la configuración se ha utilizado una imagen de dimensión 7.001×7.001 píxeles, y se ha procesado usando 5 agentes en el sistema.

La tabla 5.2 y la figura 5.1 muestran los tiempos necesarios para almacenar dicha imagen en el sistema, variando el valor del umbral de división desde 120.000

píxeles a 4.000.000 píxeles. Tanto en el gráfico como en la tabla, los tiempos están desglosados en las distintas etapas de segmentación y almacenamiento.

- El tiempo de **procesamiento** es el usado para segmentar una imagen, esto es, obtener todas las regiones de una imagen, así como todas las características de cada región y las relaciones entre las distintas regiones.
- Los tiempos de **almacenamiento temporal**, y **almacenamiento** son los necesarios para almacenar en la estructura de datos, dentro de la base de datos, las regiones, relaciones y características de una sub-imagen.
- El tiempo de **transmisión** es el utilizado para enviar las sub-imágenes de un agente a otro, para ser almacenada o segmentada.
- Por último, el tiempo de **sincronización** es el tiempo necesario para sincronizar las vecindades de las regiones en el sistema.

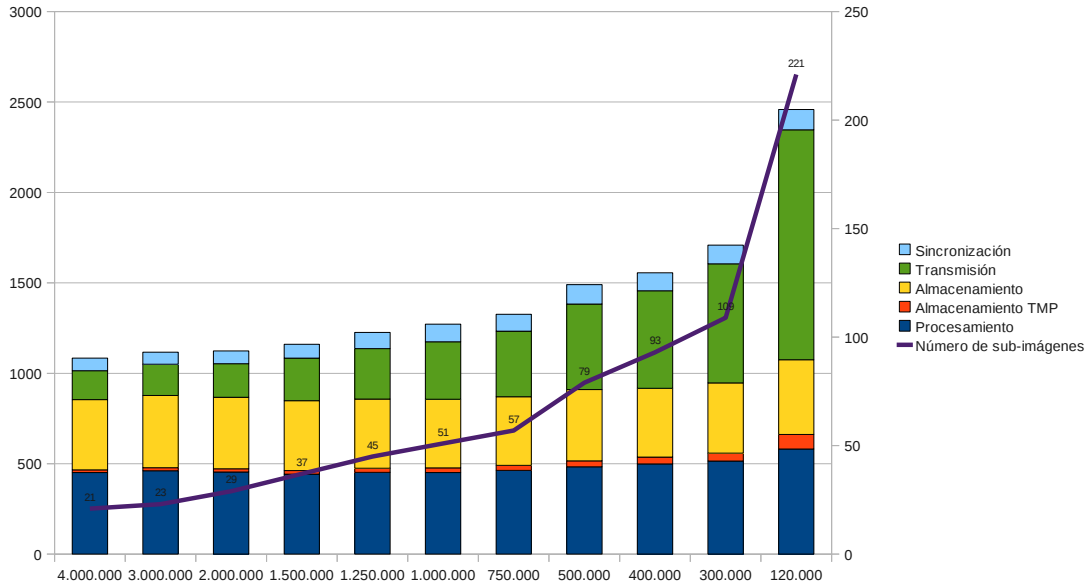

Figura 5.1: Tiempos de procesamiento modificando el umbral de división

Umbral	Procesamiento	Almacenamiento TMP	Almacenamiento	Transmisión	Sincronización	Total	Número sub-imágenes
4.000.000	451,48	15,34	387,77	159,73	69,25	1.083,57	21
3.000.000	461,08	16,64	399,92	173,1	66,4	1.117,14	23
2.000.000	454,33	18,08	395,24	185,2	70,78	1.123,63	29
1.500.000	441,79	20,63	385,68	236,37	75,78	1.160,25	37
1.250.000	452,52	22,9	381,92	279,59	88,88	1.225,81	45
1.000.000	451,58	25,59	379,15	318,08	97,27	1.271,67	51
750.000	463,55	27,77	379,45	361,85	93,61	1.326,23	57
500.000	482,61	32,95	394,56	472,94	107,34	1.490,4	79
400.000	498,98	37,97	380,63	538,41	99,78	1.555,77	93
300.000	514,52	44,18	387,66	659,3	103,6	1.709,26	109
120.000	581,3	80,75	412,78	1271,6	111,92	2.458,35	221

Tabla 5.2: Configuración del umbral del algoritmo de división

En la tabla 5.2 se puede observar que el tiempo de procesamiento de la imagen se reduce para valores de umbral grandes. Al mismo tiempo, como describe la figura 5.2, el tiempo de transmisión de las sub-imágenes depende directamente del número de sub-imágenes generadas por el paso de división. Por tanto, el tiempo de transmisión se ve afectado directamente por el valor del umbral de división: cuanto menor es el umbral, mayor número de sub-imágenes es creado, y más tiempo se necesita para su envío entre agentes.

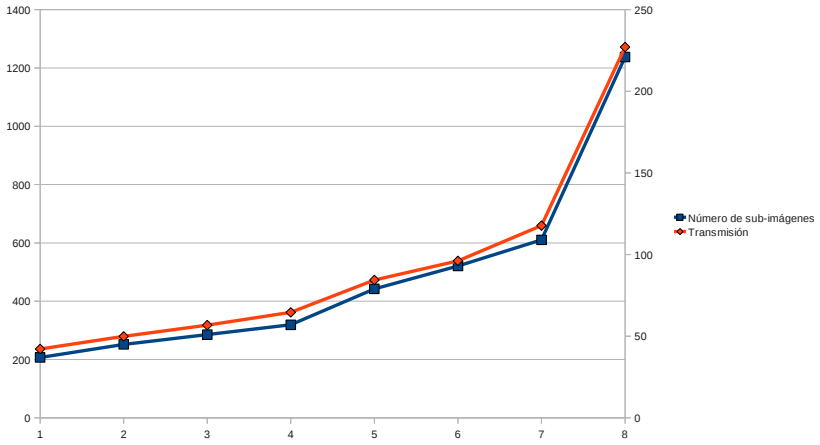

Figura 5.2: Tiempos de transmisión de las sub-imágenes cambiando el umbral de división

Por otra parte el tiempo de inserción de las regiones y las relaciones en la base de datos no varía mucho al cambiar el valor umbral de división en el algoritmo.

El objetivo es seleccionar aquel valor de umbral que nos permita reducir lo máximo posible el tiempo de procesamiento y el tiempo de transmisión, pero debemos tener en cuenta los resultados de la sección 5.1.1 donde se evalúa el

algoritmo de procesamiento para imágenes con muchas regiones (como eran las imágenes JPG), porque el número de regiones de una imagen no se conoce a priori.

Para que el sistema pueda soportar imágenes con muchas regiones, se ha escogido un valor umbral de 1.500.000 píxeles.

5.1.3. Funcionamiento básico de la división y almacenamiento de una imagen

En esta sección se explica, usando una imagen sencilla de ejemplo, el funcionamiento del algoritmo de división.

La imagen inicial, es una imagen artificial de color, con un tamaño de 1.850×2.468 píxeles. La imagen se muestra en la figura 5.3.

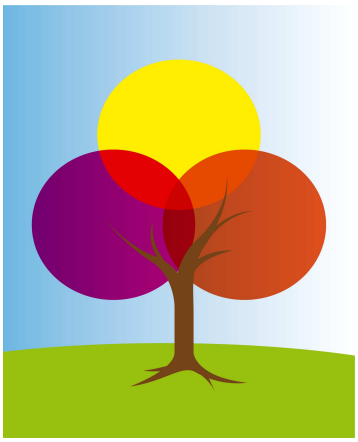

Figura 5.3: Imagen inicial de prueba

Si aplicáramos un paso del algoritmo de división sobre esta imagen, lo primero que hace el algoritmo de división es seleccionar la linea media de la dimensión más larga de la imagen, que en este caso es la altura.

Así recorreríamos esa línea de división obteniendo, por medio de un algoritmo de crecimiento de regiones, todas las regiones que componen esa línea divisoria. Al finalizar este paso, habríamos obtenido tres sub-imágenes a partir de la imagen inicial, a las que llamaremos I_1, I_2 e I_3.

I_2 se refiere a la sub-imagen del medio, que ya está procesada, y está compuesta de las regiones que forman la línea central de la imagen. De I_2 se obtiene un fichero de datos que contiene todas las características de las regiones, las relaciones entre ellas, y una representación raster de cada región. Este fichero se obtiene en un formato tabular para que toda esta información pueda ser almacenada en la base de datos que implementa la estructura de datos. De esta manera se almacena todo tipo de información de las regiones de la imagen.

Por otra parte, se obtienen otras dos sub-imágenes I_1 e I_3 que aún están sin procesar. Corresponden, en este caso, a la parte superior y la inferior de la imagen original, separadas por la sub-imagen I_2.

Si a estas sub-imágenes I_1 e I_3 se les aplica nuevamente el algoritmo de división (a cada una de ellas por separado), se obtendrán otras tres sub-imágenes de cada una de ellas.

Al igual que en el paso anterior I_{1-2} e I_{3-2} serán sub-imágenes tratadas de las que obtenemos ficheros de datos.

Y tendremos cuatro sub-imágenes, I_{1-1}, I_{1-3}, I_{3-1} e I_{3-3}, que han de ser procesadas.

Finalmente se procesan estas cuatro sub-imágenes, I_{1-1}, I_{1-3}, I_{3-1} e I_{3-3}, por separado, obteniéndose los ficheros de datos que contienen la información de las regiones de las sub-imágenes.

Los resultados del proceso de división están descritos gráficamente en la figura 5.4. Las líneas grises que aparecen en cada sub-imagen son los bordes de las regiones de dicha sub-imagen.

En todo este proceso, cada imagen puede ser tratada en un agente distinto del sistema. Tanto los ficheros de datos obtenidos, como las sub-imágenes sin procesar son enviadas a otros agentes para ser almacenados y procesadas, respectivamente.

El fichero de datos resultado de procesar una imagen se almacenará en aquel agente que indique el planificador del sistema, de manera que se mantengan distribuidas lo más equitativamente posible el número de regiones en cada agente.

La tabla 5.3 muestra la distribución de píxeles y regiones existentes en cada sub-imagen, así como el tiempo requerido para procesar (incluyendo escribir el fichero de datos en disco) y el tiempo necesario para introducir el fichero de datos a la estructura en la base de datos.

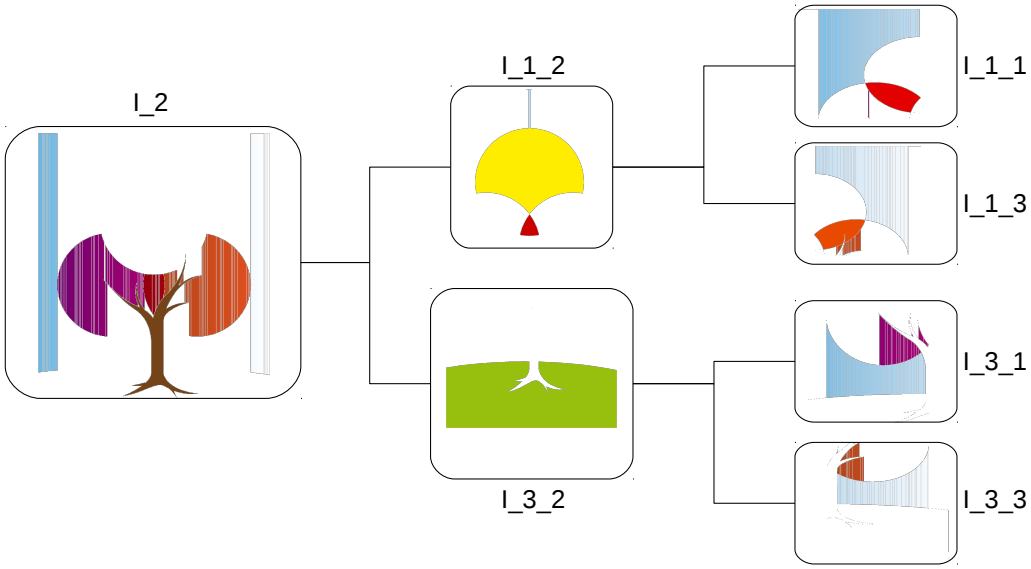

Figura 5.4: División de la imagen inicial

Sub-Imagen	Píxeles	Regiones	Procesamiento	Almacenamiento
I_2	1.425.657	147	5,5	1,6
I_{1-2}	416.928	9	2,1	0,3
I_{3-2}	927.759	6	2,4	0,44
I_{1-1}	577.395	2.352	5,68	4,4
I_{1-3}	603.373	3.029	7,75	6,5
I_{3-1}	325.526	3.315	4,51	8,9
I_{3-3}	289.162	3.111	5,31	6,98
Total	4.565.800	11.969	33,25	29,12

Tabla 5.3: Estadísticas del procesamiento de la imagen

5.1.4. Estudio del almacenamiento de las imágenes del sistema

Para estudiar el rendimiento de los métodos distribuidos creados y del prototipo de sistema, en esta sección se muestran los resultados de almacenar tres imágenes de distintos tamaños, desde 4.631×3.025 píxeles una imagen mediana, pasando por una imagen grande de 7.001×7.001 píxeles, a una más grande de 21.601×10.801 píxeles. Este almacenamiento en el sistema se va a realizar variando el número de agentes que componen dicho sistema distribuido y comprobar si el rendimiento mejora al aumentar el número de agentes.

En la tabla 5.4 se pueden ver las características de dichas imágenes utilizadas para las pruebas.

ID Imagen	Dimensión	Regiones	Relaciones	Píxeles (en millones)	Tamaño (en MB)
Imagen 1	4.631×3.025	8.718	47.540	14	40,1
Imagen 2	7.001×7.001	176.263	1.003.185	49	489,5
Imagen 3	21.601×10.801	821.800	4.777.768	233,3	906,2

Tabla 5.4: Imágenes utilizadas para las pruebas del sistema

A continuación se muestran los resultados obtenidos para las tres imágenes. Intentaremos deducir si la división de una imagen en varias sub-imágenes para ser procesadas distribuidamente mejora los tiempos para realizar operaciones morfológicas basadas en regiones.

5.1.4.1. Imagen 1

Para la Imagen 1, los tiempos de almacenamiento usando distinto número de agentes, de uno a seis, son los que se muestran en la figura 5.5.

Se puede observar que la reducción de tiempo de procesamiento producida con la inclusión de nuevos agentes en el sistema no es muy significativo.

Esto es debido a que la imagen, al ser tamaño mediano, solamente es divida dos veces por el algoritmo de división, y las sub-imágenes obtenidas por el algoritmo de división son muy desiguales.

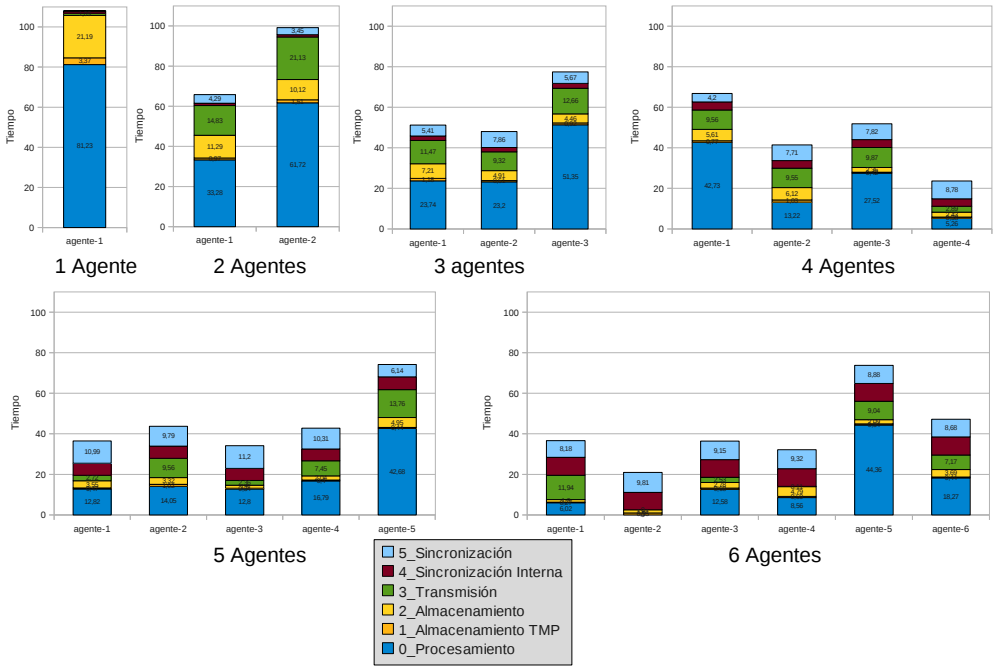

Figura 5.5: Almacenamiento de la Imagen 1

En las figuras 5.6 y 5.7 se muestran los gráficos de distribución de las regiones extraídas de la imagen entre los distintos agentes, y de los píxeles, respectivamente.

Se puede apreciar que la distribución de regiones no es uniforme entre los distintos servidores. Esto es consecuencia, como se ha explicado anteriormente de que el tamaño de la imagen no es suficientemente grande.

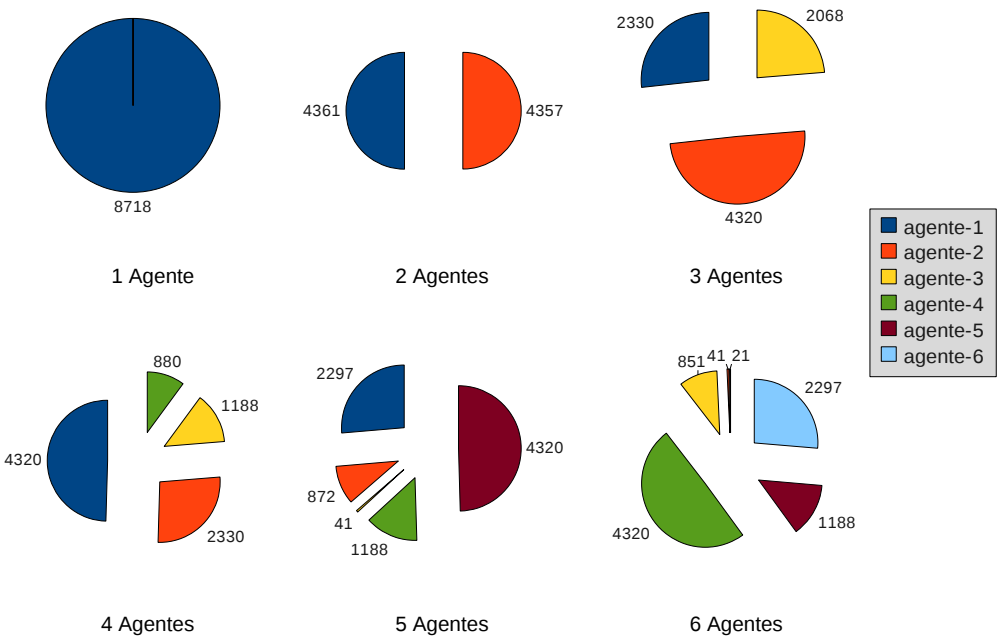

Figura 5.6: División de las regiones de la Imagen 1

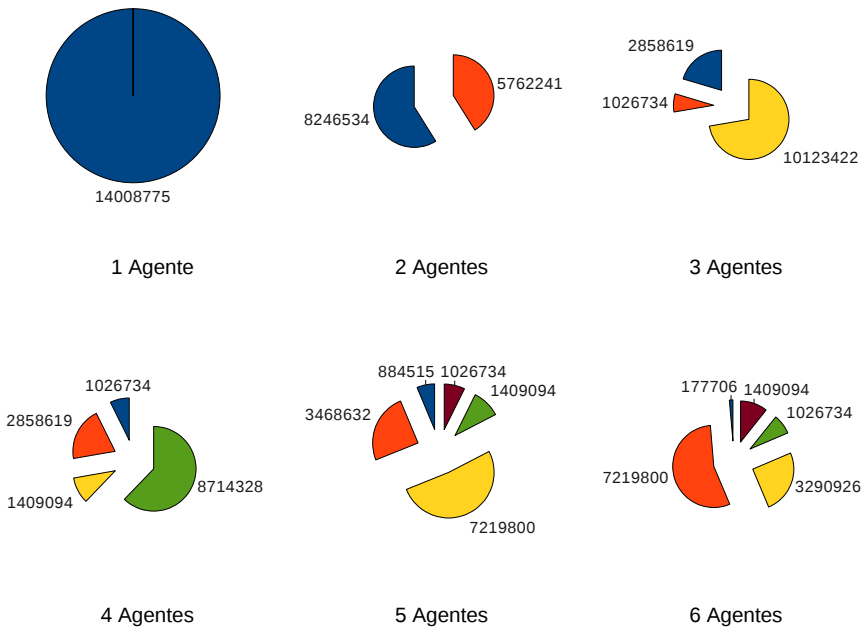

Figura 5.7: División de los píxeles de la Imagen 1

5.1.4.2. Imagen 2

Para la Imagen 2, los tiempos de almacenamiento al utilizar distinto número de agentes en el sistema se muestran en la figura 5.8.

Para esta imagen, que es casi cuatro veces mayor que la imagen 1, sí que se puede observar una reducción del tiempo necesario para segmentar la imagen, a medida que se incrementa el número de agentes en el sistema.

Se puede apreciar en el gráfico 5.8, que el tiempo para almacenar la imagen usando dos agentes (tomando como valor, el tiempo de aquel agente que más tiempo emplea en almacenar la imagen), se reduce con respecto al requerido utilizando solamente un agente.

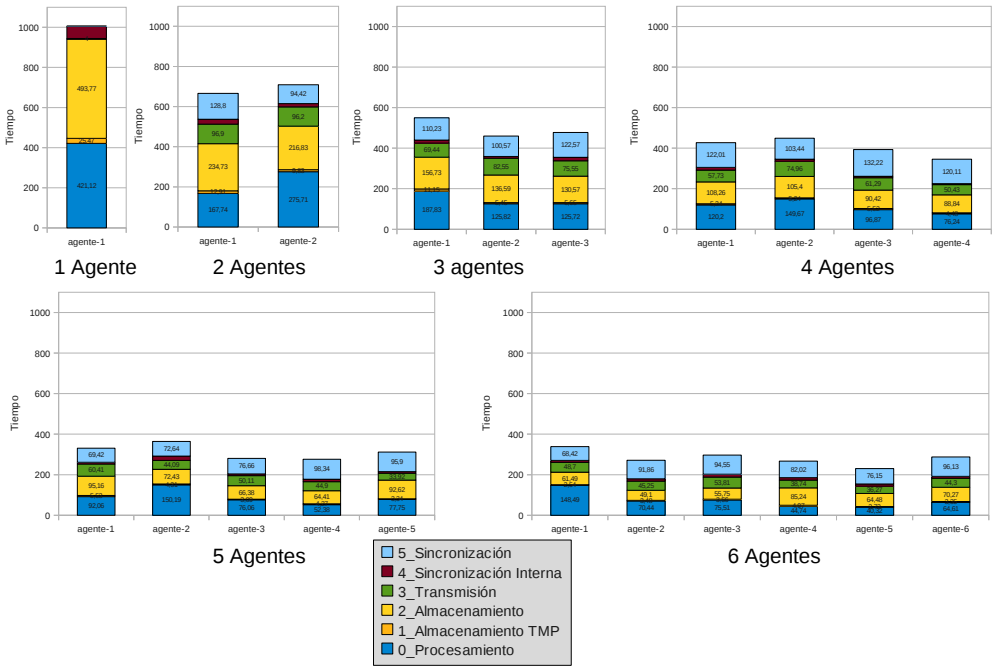

Figura 5.8: Almacenamiento de la Imagen 2

A su vez en la figura 5.9 se aprecia cómo las regiones de esta imagen se almacenan de una forma homogénea en los distintos agentes. Este no es el caso de los píxeles (ver figura 5.10), ya que como se ha explicado en capítulos anteriores, lo que interesa es que las regiones estén bien repartidas, pues las operaciones morfológicas se realizan a nivel de región. Así, este reparto equitativo de las regiones entre los agentes del sistema ayudará a distribuir homogéneamente el tiempo para aplicar una operación sobre la imagen.

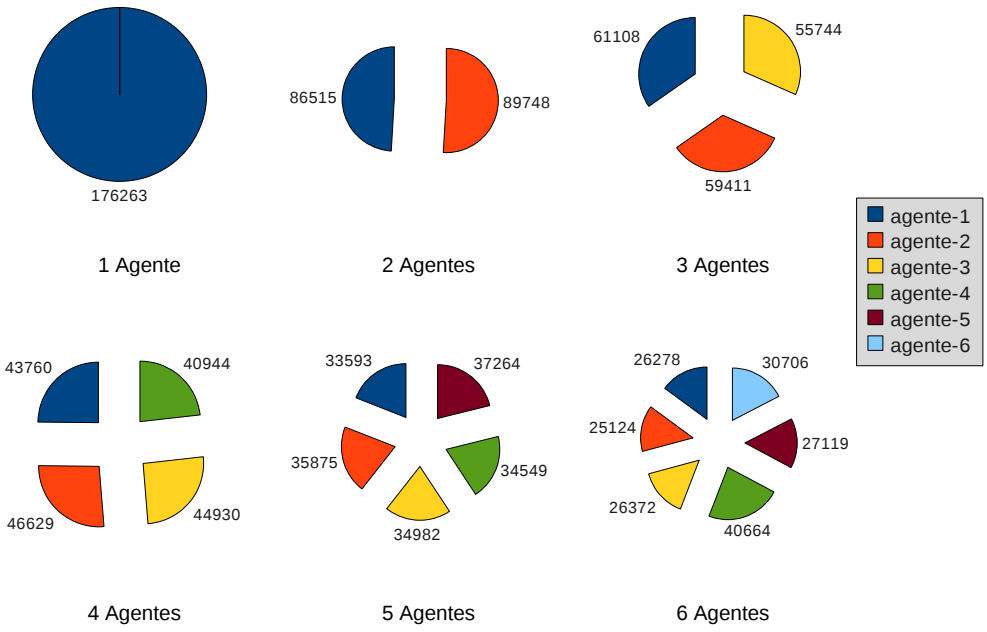

Figura 5.9: División de las regiones de la Imagen 2

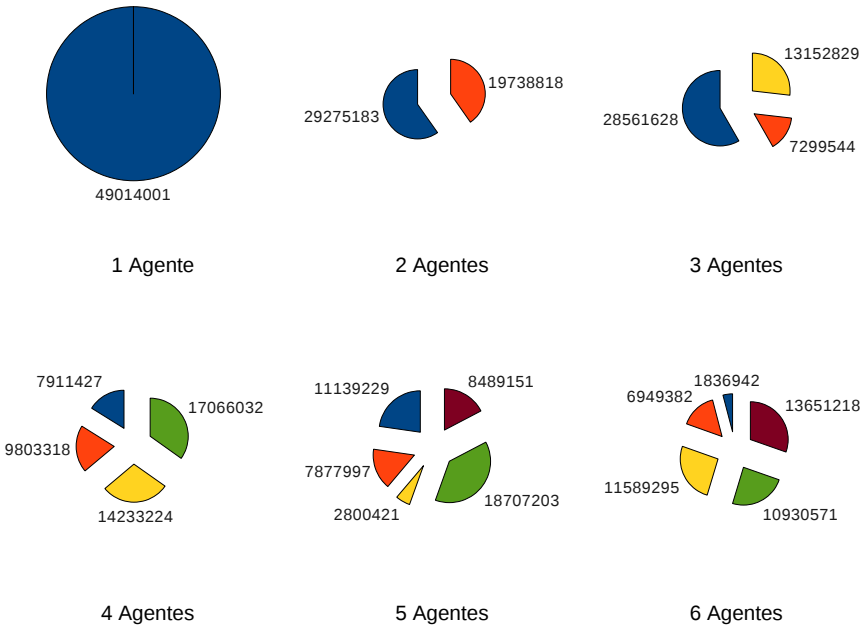

Figura 5.10: División de los píxeles de la Imagen 2

5.1.4.3. Imagen 3

Los tiempos de almacenamiento de la imagen 3, usando distinto número de agentes en el sistema, son los que se muestran en la figura 5.11. Para esta imagen, el tiempo de segmentación y almacenamiento también se reduce cuando el número de agentes se aumenta en el sistema, con respecto al necesario para almacenar la misma imagen utilizando solamente un agente en el sistema.

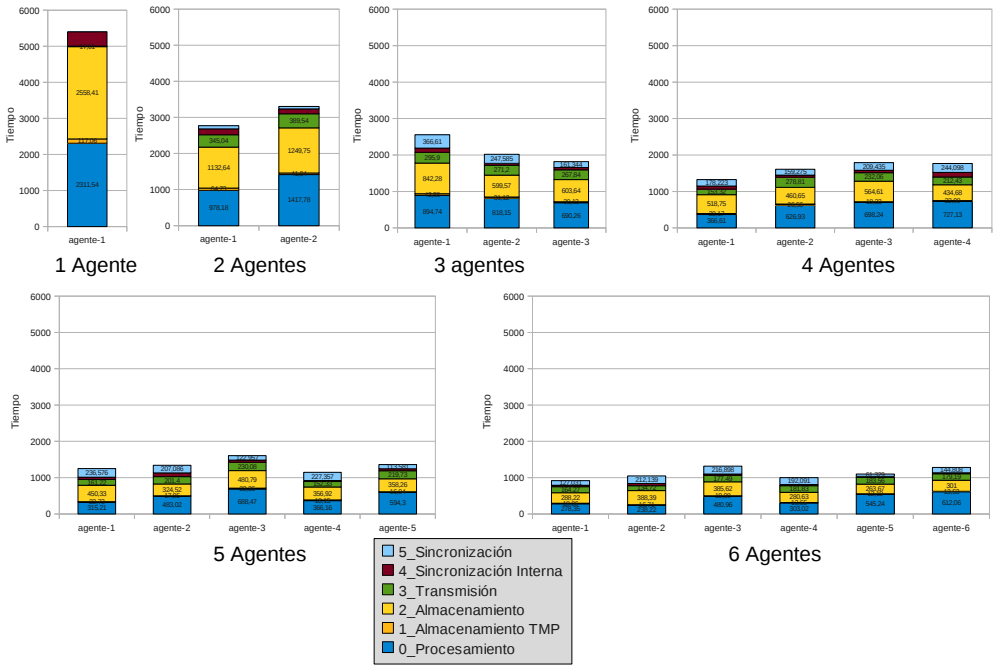

Figura 5.11: Almacenamiento de la Imagen 3

En esta imagen, al igual que ocurre con para imagen 2, la distribución de las regiones al ser almacenadas en la estructura de datos, implementada en una base de datos, se realiza de forma casi homogénea. En la figura 5.9 se puede ver el reparto de las regiones entre los distintos agentes que componen el prototipo.

En este caso, como se ve en la figura 5.10), el almacenamiento de los píxeles es más equitativo que en la imagen anterior.

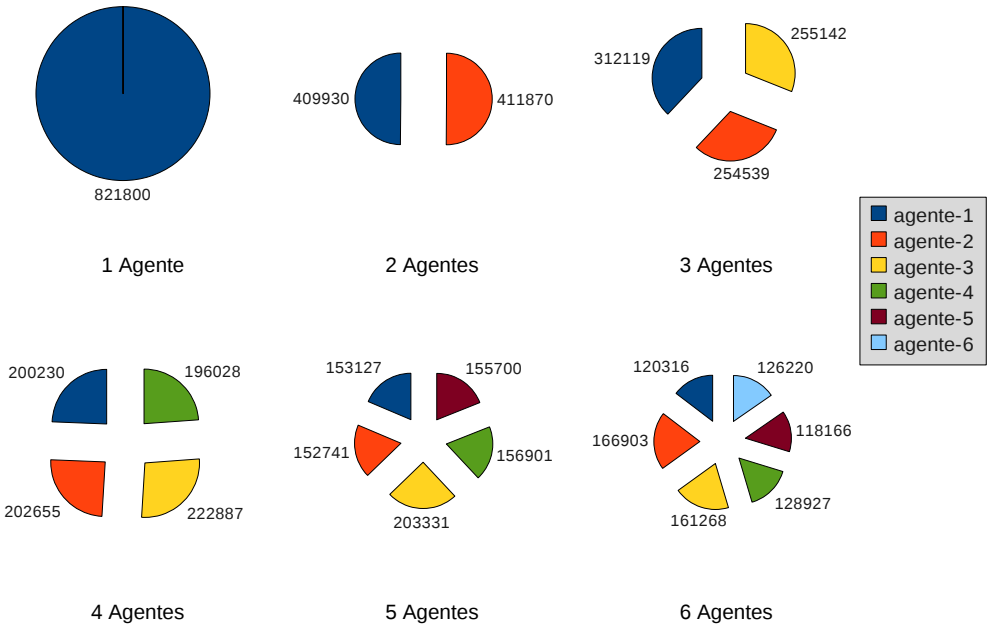

Figura 5.12: División de las regiones de la Imagen 3

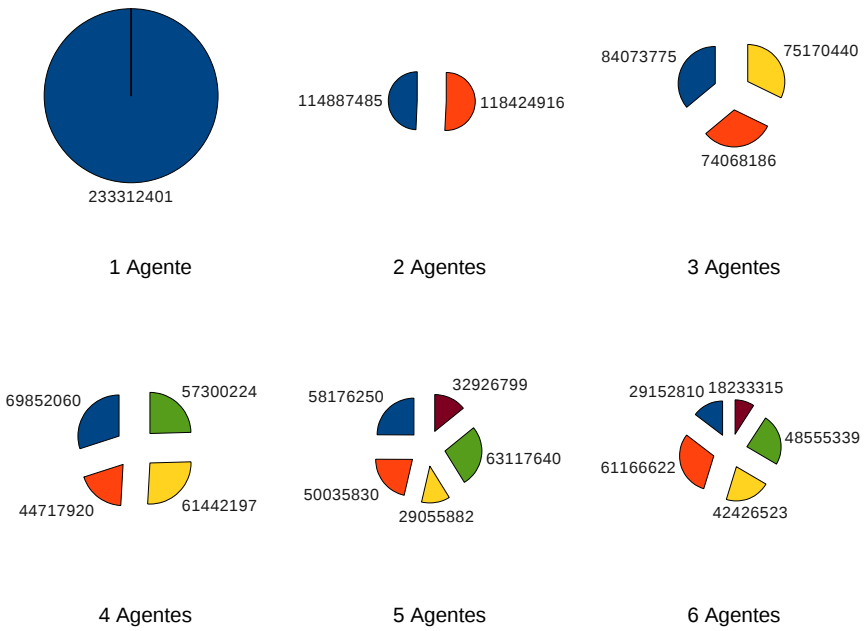

Figura 5.13: División de los píxeles de la Imagen 3

5.2. Filtros y recuperación de las imágenes

5.2.1. Erosión morfológica sobre imágenes almacenadas

En este apartado se procederá a aplicar, como ejemplo de uno de los procesamientos morfológicos distribuidos desarrollados, una erosión morfológica. Dicha erosión, como ya se ha explicado, y al igual que los otros métodos desarrollados se ha implementado para ser ejecutada directamente sobre la imagen almacenada en la estructura de datos. Es decir, se ejecuta directamente sobre la imagen almacenada en una base de datos, sin tener que recomponer la imagen.

Se probará a realizar una erosión sobre las imágenes de test mostradas en la tabla 5.4, y variando el número de agentes en los que cada imagen se encuentra almacenada.

De esta manera, se intenta comprobar si el tiempo de procesamiento para la operación morfológica de erosión se reduce cuando el número de agentes aumenta en el sistema.

5.2.1.1. Imagen 1

Para la primera imagen, como ya se comentó en la sección 5.1.4, en la figura 5.6, se muestra que las regiones de la imagen no están bien distribuidas entre los distintos agentes del sistema. Esto es debido a que el tamaño de la imagen es mediano, y no lo suficientemente grande para ser dividida más veces por el sistema. Por ello, siempre existe un agente que tiene almacenadas la mayoría de las regiones de la imagen.

Así, se puede apreciar en el gráfico de la figura 5.14, comparándolo con el gráfico de distribución de regiones de la figura 5.6, se distingue que aquel agente que más tarda en realizar la erosión morfológica sobre las regiones que contiene de la imagen, es aquel que posee mayor número de regiones almacenadas su propia base de datos.

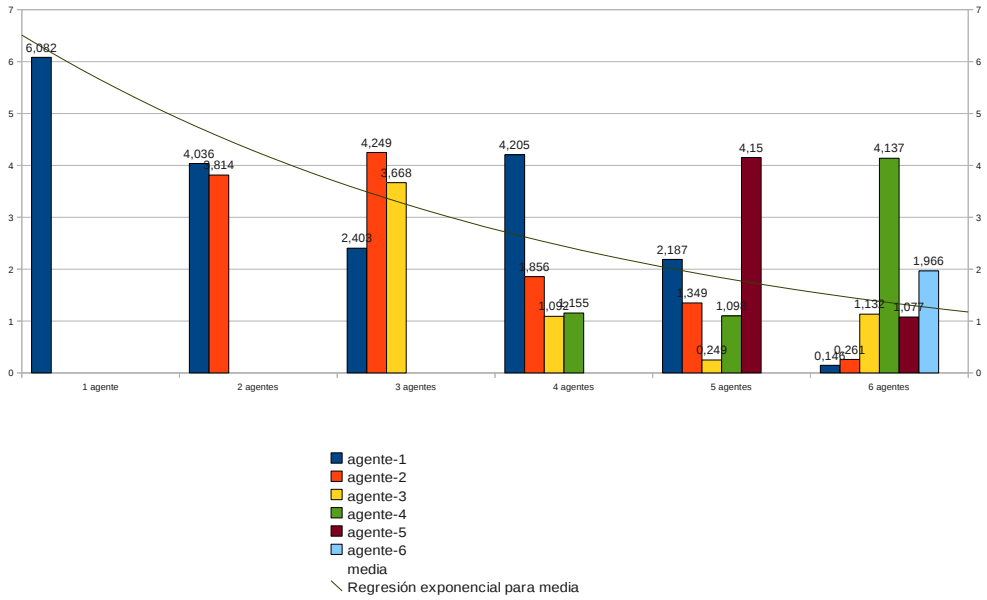

Figura 5.14: Erosión sobre la Imagen 1

5.2.1.2. Imagen 2

Para la segunda imagen de test, que tiene unas dimensiones de 7.001×7.001 píxeles, como se muestra en la figura 5.9, el reparto de las regiones entre los distintos agentes que componen el sistema es más homogéneo. Por este motivo, el tiempo de procesamiento de la erosión morfológica sobre las regiones de la imagen también se reparte mejor entre los distintos agentes del sistema. Este reparto del procesamiento de la erosión, se puede ver en el gráfico de la figura 5.15. De esta manera al repartirse mejor el número de regiones, y por lo tanto el tiempo de procesamiento, cuanto mayor es el número de agentes en el sistema, menor es el tiempo necesario para realizar la erosión morfológica de las regiones de la imagen.

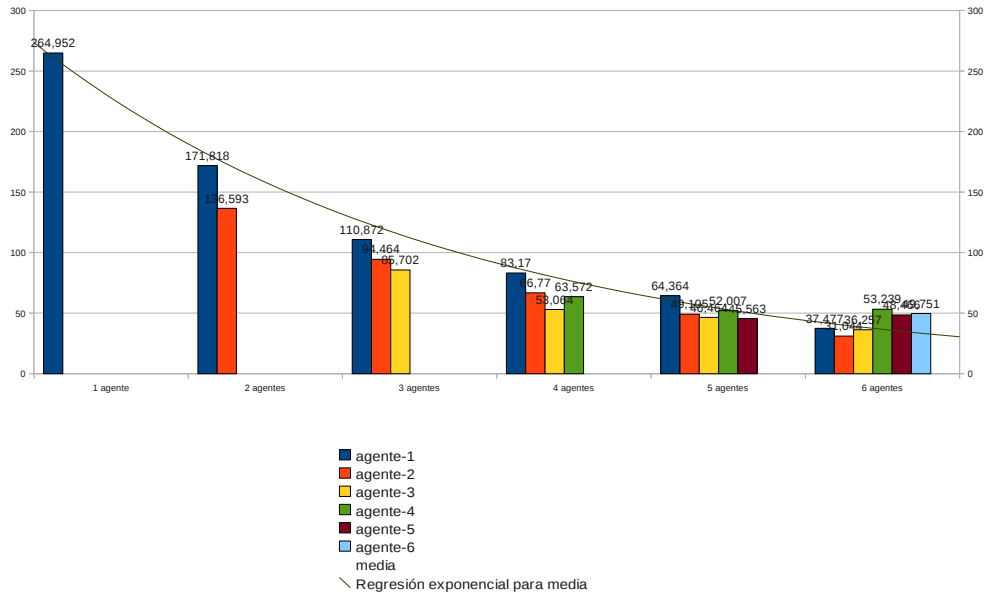

Figura 5.15: Erosión sobre la Imagen 2

5.2.1.3. Imagen 3

En la imagen 3, los tiempos de ejecución de una erosión morfológica, usando distinto número de agentes para almacenar la imagen en el sistema, se muestran en el gráfico de la figura 5.15. Al igual que sucede para la imagen 2, el tiempo se reduce cuando el número de agentes que componen el sistema aumenta, ya que también en esta imagen se han logrado almacenar las regiones de la imagen distribuyéndolas entre los distintos agentes de forma casi equitativa. Dicho reparto de regiones se puede ver en el gráfico de la figura 5.12.

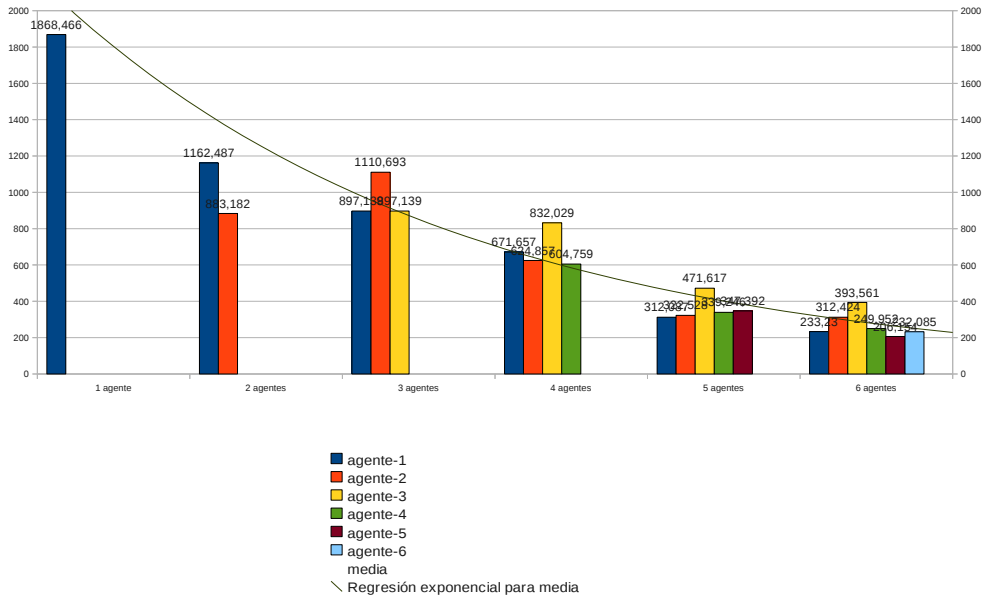

Figura 5.16: Erosión sobre la Imagen 3

5.2.2. Recuperación de información de regiones de una imagen

Un aspecto reseñable del prototipo de sistema que se ha desarrollado en la presente tesis es la opción de recuperar regiones de las imágenes por sus características. El prototipo se ha creado siguiendo el enfoque de integración de bases de datos distribuidas usando traducción de consultas (véase sección 2.1). Dicho prototipo se ha diseñado usando una variación del sistema ONTOFUSION (Alonso-Calvo y cols., 2007) (Pérez-Rey y cols., 2006), donde no es necesario el uso de ontologías para la integración de diferentes esquemas relacionales. Esto es debido a que todas las bases de datos de donde se va a obtener la información siguen un único esquema, el de la estructura de datos diseñada.

Para ver si el tiempo de procesamiento de las consultas de recuperación sobre las imágenes almacenadas mejora al crecer el número de agentes, se va a ejecutar una consulta sobre las imágenes seleccionadas para las pruebas y mostradas en la tabla 5.4.

Dicha consulta se realizará sobre cada una de las imágenes, variando el número de agentes en los que está almacenada dicha imagen. Para de esta manera comprobar si el tiempo necesario para completar la consulta se reduce al aumentar el número de agentes en el sistema.

La consulta de ejemplo que se va a realizar, es la siguiente:

Obtener de la imagen, aquellas regiones que cumplan todas de las siguientes condiciones:

- (a) Regiones que tengan un área menor que un valor umbral dado
- (b) Regiones que tengan una etiqueta de color mayor que un valor umbral dado
- (c) Regiones cuyos ejes mayor y menor sean más pequeños que el 1 % de la dimensión de la imagen

Esta consulta podría ser, para por ejemplo, obtener aquellas regiones que no son relevantes en la imagen o que podrían considerarse como ruido.

5.2.2.1. Imagen 1

Los resultados de la realización de dicha consulta sobre la imagen 1 se muestran en la figura 5.17.

Se puede observar que, para esta imagen, no existe reducción de tiempo de ejecución para cuando se utilizan más de dos agentes.

Al igual que se comentó anteriormente y que ocurre en el los pasos de segmentación, almacenamiento y en el procesamiento (ejecución de la erosión morfológica)(secciones 5.1.4 y 5.2.1). Esto es debido al tamaño de la imagen, al ser una imagen de tamaño mediano no contiene muchas regiones, por lo que en el almacenamiento de las mismas no está equilibrado entre los distintos agentes del sistema. Como consecuencia, siempre hay un agente que almacena la mayoría de las regiones y es el que más tarda en realizar la consulta sobre su base de datos.

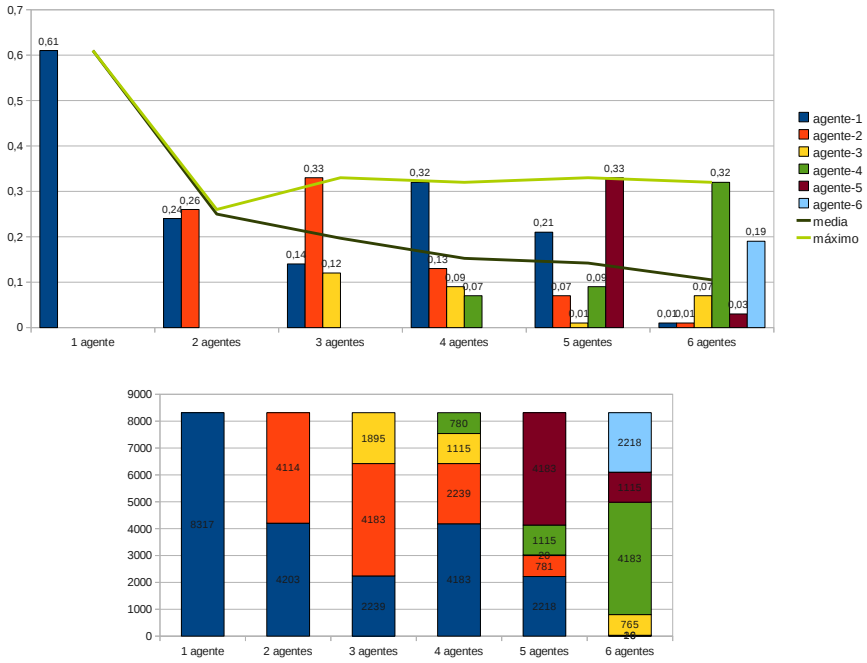

Figura 5.17: Consulta sobre la Imagen 1

5.2.2.2. Imagen 2

Para la imagen 2 se aprecia un mejor comportamiento del sistema al realizar la consulta de recuperación de regiones. Esto es debido a que las regiones están almacenadas de forma más distribuida entre los agentes del sistema. Sin embargo, como en este tipo de consultas el número de regiones afectadas y su ubicación no se conoce a priori, la distribución en esta imagen de dichas regiones afectadas no está repartida. En la ejecución de la consulta con 4 y con 5 agentes en el sistema, los tiempos del agente que más tarda son similares ya que dicho agente recupera el mismo número de regiones de su base de datos en ambos casos.

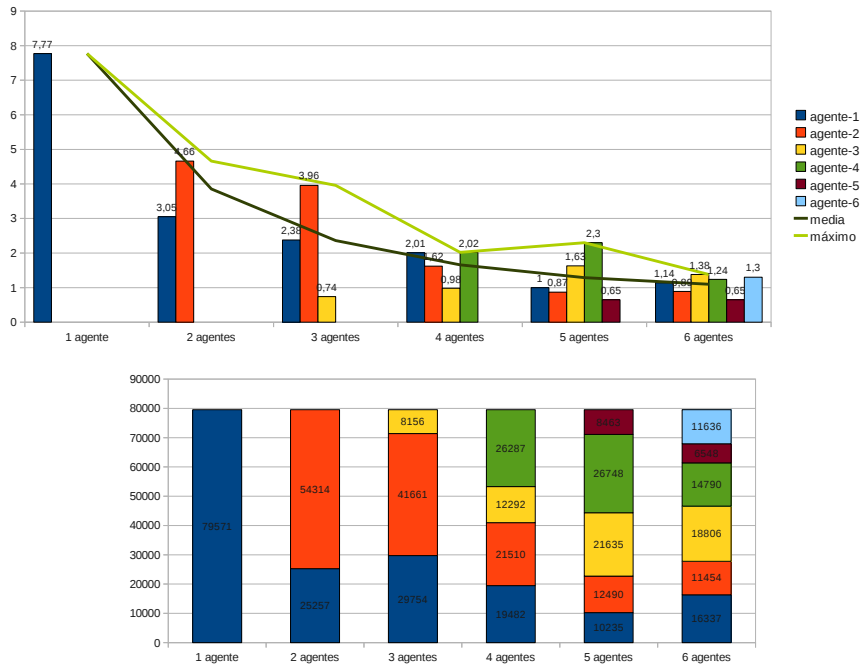

Figura 5.18: Consulta sobre la Imagen 2

5.2.2.3. Imagen 3

Por último, la imagen 3, que es la más grande de las imágenes de test, al existir más regiones que en las otras dos imágenes, las regiones que cumplen los criterios de la consulta están más repartidas entre los distintos agentes. Por lo que el tiempo de procesamiento para la consulta decrece al aumentar el número de agentes en el sistema.

Esto se puede observar en la segunda parte del gráfico 5.19, donde se muestran la distribución de regiones recuperadas por cada agente en el sistema al ejecutar la *query* de recuperación.

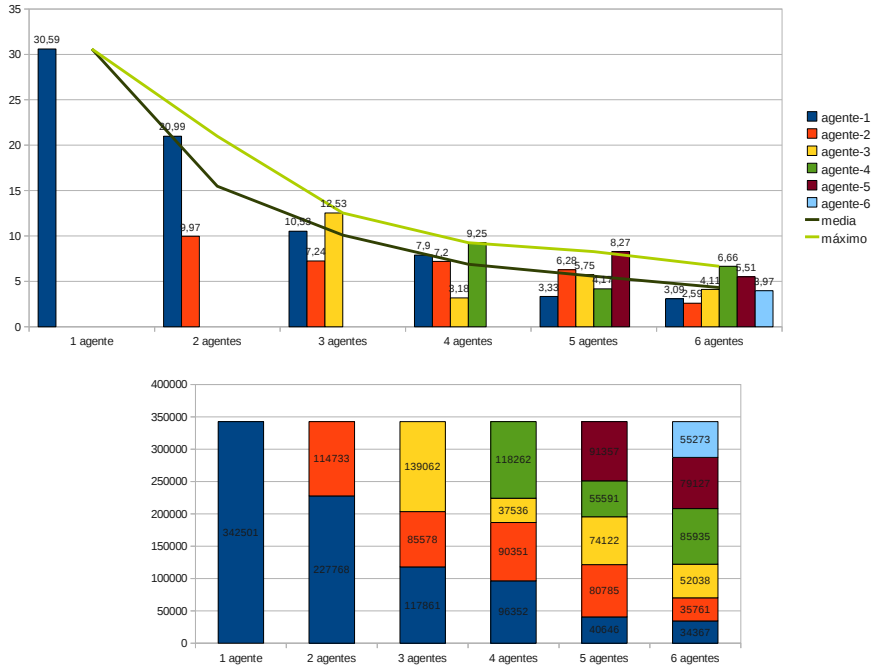

Figura 5.19: Consulta sobre la Imagen 3

5.3. Discusión de los resultados obtenidos

En la presente sección se hace un resumen y se comentan los resultados obtenidos de las secciones anteriores, para cada una de las tres imágenes del juego de pruebas.

Para ver el comportamiento de los métodos desarrollados se va a utilizar el porcentaje de ahorro o ganancia de tiempo de cada test realizado. Dicho porcentaje se calcula siguiendo la formula mostrada a continuación 5.1.

$$\%Ganancia_{Op} \equiv (1 - \frac{max(t_{op-N}(Imagen_i))}{t_{op-1}(Imagen_i)}) \times 100 \tag{5.1}$$

Donde:

- $Imagen_i$ es la imagen sobre la que se realiza la operación Op que puede ser, (i) Segmentación y almacenamiento, (ii) erosión morfológica, o (iii) recuperación de información.
- $t_{op-1}(Imagen_i)$, es el tiempo necesario para completar la operación correspondiente usando solamente un agente en el sistema.
- $max(t_{op-N}(Imagen_i))))$, es el mayor de los tiempos de los agentes, resultantes de completar una operación usando n agentes en el sistema. En otras palabras, el tiempo del agente que más tarda en completar la operación.

5.3.1. Imagen 1

Para la imagen 1, que tiene un tamaño de 4.631×3.025, la ganancia al variar el número de agentes del sistema es la que se muestra en la tabla 5.5.

Operación	1 agente	2 agentes	3 agentes	4 agentes	5 agentes	6 agentes
Almacenamiento	0,00	8,22	28,34	38,15	31,28	31,73
Erosion	0,00	33,64	30,14	30,86	31,77	31,98
Query	0,00	57,38	45,90	47,54	45,90	45,90

Tabla 5.5: % de ganancia para la imagen 1 (4.631×3.025)

En esta tabla tabla 5.6 se puede descubrir la razón de que las operaciones

sobre la imagen 1 no reduzcan sus tiempos cuando el sistema tiene más de 3 agentes.

Aún así, la ganancia en esta imagen en al utilizar tres servidores o más en el sistema es del 30 % en las operaciones de segmentación y almacenamiento y en la erosión morfológica. Esta ganancia aumenta hasta un 46 % en la consulta de recuperación de regiones por características.

Sub-imagen	regiones	pixeles	tiempo de procesamiento
I	41	7.219.800	46,80
I_1	21	177.706	4,54
I_{1-1}	4.320	1.026.734	6,91
I_{1-3}	1.188	1.409.094	11,52
I_3	12	1.796.398	12,22
I_{3-1}	2.297	884.515	4,54
I_{3-3}	839	1.494.528	11,07
Total	8.718	14.008.775	97,6

Tabla 5.6: Desglose del procesamiento de la imagen 1 (4.631×3.025)

El hecho de que no se produzca una ganancia mayor en esta imagen se debe a que hay una sub-imagen (I_{1-1}) que contiene más o menos la mitad de las regiones de la imagen. Así mismo, para dividir la imagen inicial, se requiere la mitad del tiempo total de procesamiento de la imagen. Hay que tener en cuenta que estos tiempos incluyen no solo la obtención de las regiones, sino también la escritura en disco de las tres sub-imágenes resultantes y los ficheros de datos.

Por lo tanto, la ganancia en esta imagen, que no es lo suficientemente grande para la configuración del prototipo del sistema, se podría mejorar disminuyendo el valor umbral del algoritmo de división. Con esto, la imagen generaría más sub-imágenes y por lo tanto se repartirían más las regiones entre los distintos agentes.

5.3.2. Imagen 2

En los resultados de la imagen 2, que tiene unas dimensiones de 7.001×7.001 píxeles, se ve que la ganancia en el sistema, crece al aumentar el número de

agentes en el sistema.

Este aumento de ganancia se produce para las tres operaciones, segmentación y almacenamiento, erosión morfológica y para la consulta de recuperación de regiones por características.

Los datos cuantitativos de la ganancia se pueden ver en la tabla 5.7.

Operación	1 agente	2 agentes	3 agentes	4 agentes	5 agentes	6 agentes
Almacenamiento	0,00	29,51	45,33	55,36	63,77	66,34
Erosion	0,00	35,15	58,15	68,61	75,71	79,91
Query	0,00	40,03	49,03	74,00	70,40	82,24

Tabla 5.7: % de ganancia para la imagen 2 (7.001×7.001)

Como se comentó en la sección 5.2.2 para esta imagen, en la consulta de recuperación, cuando el sistema tiene cuatro y cinco agentes, la ganancia es prácticamente la misma. Esto se debe como muestra el gráfico 5.18 a que el número de regiones obtenidas del agente que más tarda es el mismo, por lo que el tiempo requerido para ese agente es similar.

Para esta segunda imagen la ganancia al utilizar seis servidores en el sistema llega hasta el 66 % en la operación de segmentación y almacenamiento, que es la más costosa.

Además en la erosión morfológica la ganancia llega hasta el 80 % y hasta un 82 % en la consulta de recuperación de regiones por características.

5.3.3. Imagen 3

Para la tercera imagen, que es la más grande, con unas dimensiones de 21.601×10.801 píxeles, las ganancias con respecto al uso de un sólo agente en el sistema se muestran en la tabla 5.8.

Operación	1 agente	2 agentes	3 agentes	4 agentes	5 agentes	6 agentes
Almacenamiento	0,00	27,51	59,47	66,85	70,26	75,63
Erosion	0,00	37,78	40,56	55,47	74,76	78,94
Query	0,00	31,38	59,04	69,76	72,97	78,23

Tabla 5.8: % de ganancia para la imagen 3 (21.601×10.801)

Esta imagen, al ser la más grande, es la que más homogéneamente tiene repartido el número de regiones entre los agentes del sistema. Por ello, en la erosión morfológica la ganancia llega hasta casi el 79 % al igual que en la consulta de recuperación de regiones por características.

También el paso de segmentación morfológica y de almacenamiento en la estructura de datos se distribuye mejor entre los agentes, llegando a una ganancia con seis agentes en el sistema del 76 %.

Como se observa, en la tabla de resultados 5.8, en esta imagen la ganancia es creciente, para las tres operaciones, es decir, la ganancia crece a medida que el número de agentes en el sistema va aumentando.

5.3.4. Comparación de resultados con otros algoritmos

Para poder evaluar el rendimiento de la estructura de datos se han implementado tres algoritmos, la erosión y dilatación morfológica basadas en regiones y un algoritmo de filtrado de regiones. Dichos algoritmos se han desarrollado utilizando el lenguaje de programación Java. Para la experimentación, dichos algoritmos han sido ejecutados utilizando el mismo hardware y sistema operativo que los agentes del sistema creado en la tesis. Estos algoritmos siguen la filosofía tradicional del procesamiento de imágenes, es decir, reciben una imagen de entrada y devuelven como resultado otra imagen distinta realizando el procesamiento de la imagen de entrada en memoria.

Para comparar los nuevos métodos creados en la presente tesis doctoral y que se ejecutan directamente sobre la estructura de datos se utilizarán los tiempos del sistema compuesto con un solo agente.

- Para la erosión y la dilatación morfológica basadas en regiones, el algoritmo implementado recibe una imagen de entrada y devuelve como resultado la erosión o dilatación morfológica en una imagen resultado. El algoritmo implementado recorre los píxeles de la imagen de entrada y, usando un algoritmo de crecimiento de regiones, obtiene las regiones que componen la imagen. Para cada región que se obtiene, se guarda las etiquetas máxima y mínima de los píxeles vecinos de la región obtenida. De esta manera, una vez que se obtiene una región, se dispone de los valores necesarios para

realizar la erosión y dilatación de la región.

En la tabla 5.9 se muestran los tiempos requeridos para realizar la erosión basada en regiones de las imágenes de prueba 1 y 2 de la tabla 5.4. Dichos tiempos no incluyen el tiempo de carga de la imagen en memoria, ni el de escritura del resultado en disco, es decir, solamente se tiene en cuenta el tiempo de procesamiento de la imagen en memoria.

ID Imagen	Dimensión	t erosion estructura de datos	t algoritmo erosión morfológica
Imagen 1	4.631x3.025	6,082	114,865
Imagen 2	7.001x7.001	264,952	416,556

Tabla 5.9: Comparación de una erosión morfológica de regiones en la estructura de datos frente a un algoritmo

La complejidad del algoritmo implementado, y por lo tanto el tiempo de procesamiento, depende directamente del número de píxeles de la imagen de entrada. Por otro lado, la estructura de datos está orientada a regiones, por lo que el procesamiento se ve afectado por número de regiones y de relaciones entre estas regiones. Como se puede observar en la tabla 5.9 los tiempos para la erosión morfológica basada en regiones son significativamente menores para las imágenes almacenadas en la estructura de datos frente al procesamiento usando el algoritmo en memoria.

- Para la recuperación de regiones sobre una imagen de entrada, se ha implementado un algoritmo que recorre la imagen de entrada seleccionando solo aquellas regiones que cumplen ciertas condiciones. La imagen de resultado la componen aquellas regiones seleccionadas.

  En la tabla 5.10 se muestran los tiempos empleados para la ejecución del filtrado de regiones de las imágenes de prueba 1 y 2 de la tabla 5.4. Dichos tiempos no incluyen el tiempo de carga de la imagen en memoria, ni el de escritura del resultado en disco, es decir, solamente se tiene en cuenta el tiempo de procesamiento de la imagen en memoria.

  De igual modo que para el algoritmo de erosión morfológica, la complejidad del algoritmo depende directamente del número de píxeles de la imagen

ID Imagen	Dimensión	t filtrado estructura de datos	t algoritmo filtrado de regiones
Imagen 1	4.631x3.025	0,61	120,156
Imagen 2	7.001x7.001	7,77	428,351

Tabla 5.10: Comparación de recuperación de regiones desde la estructura de datos frente a un algoritmo

de entrada. Esto se debe a que la imagen es procesada por completo para extraer las regiones que la componen, devolviendo como resultado algunas de las regiones de la imagen inicial. Sin embargo, la estructura de datos recorre las regiones de la imagen seleccionando solo aquellas que cumplan las condiciones marcadas por el filtro. La tabla 5.10 muestra los tiempos para la recuperación de regiones usando el algoritmo y la imagen almacenada en la estructura de datos. Para este segundo método los tiempos son mucho menores.

Como observación a los resultados anteriores cabe aclarar que los tiempos obtenidos usando los algoritmos de procesamiento creados son similares al paso de división y extracción de características de las regiones de una imagen. Por este motivo, se obtendrá mayor beneficio de la utilización de la estructura de datos cuando se realicen varios filtros o consultas recuperación de información sobre la misma imagen.

5.3.5. Resumen de los resultados

Tanto el algoritmo de división, como la estructura de datos y el mismo prototipo de sistema han sido diseñados para realizar operaciones morfológicas basadas en regiones sobre imágenes muy grandes. Por esta razón, no debe sorprender que su comportamiento, o ganancia, al aumentar los agentes en el sistema sea mejor cuanto mayor sea la imagen que se esta procesando.

Para el paso de división, segmentación y almacenamiento en la base de datos, hemos visto que la reducción de los tiempos es considerable para imágenes grandes como las imágenes 2 y 3 de ejemplo. Pero, ¿qué pasaría si usáramos un número ilimitado de agentes en el sistema?, ¿cuál es el mínimo de tiempo al

que se podría reducir el procesamiento en el prototipo creado?. La respuesta se encuentra en los test mostrados, al igual que ocurre con la imagen 1: el tiempo mínimo al que se puede reducir el procesamiento de la imagen es aquel paso de división o procesamiento que más tarde. Es decir, en el caso que se tuviera un número ilimitado de agentes y, por ejemplo, una imagen que se divida en 120 sub-imágenes, habrá uno de esos 120 procesamientos que será el tiempo máximo de procesamiento en un agente, como pasa con la imagen de ejemplo 1 con su paso de división inicial. Así mismo, en tiempo lineal, el procesamiento será reducido a la suma de los tiempos máximos desde la imagen inicial hasta el último nivel de división.

Como se ha comprobado en los tests, y se muestra en este capítulo con los ejemplos de las tres imágenes de distinto tamaño, las operaciones morfológicas de erosión, dilatación, apertura, cierre y watershed dependen directamente del número de regiones almacenadas y de las relaciones entre las diferentes regiones. Por esto, se implementó el balanceo de carga en el sistema para el paso de almacenamiento, para equilibrar en número de regiones almacenadas en los diferentes agentes. De esta manera se intenta equilibrar al máximo el tiempo de procesamiento en los distintos agentes en el sistema. Se ha demostrado que dicho balanceo a la hora de realizar el almacenamiento homogéneo de regiones en los distintos agentes mejora el comportamiento de los filtro sobre la imagen. Esto se puede explicar, ya que al estar todas estas operaciones morfológicas implementadas como procedimientos que trabajan directamente sobre las regiones almacenadas en una base de datos, trabajan por lo tanto con unas tablas mayores cuanto mayor sea el número de regiones y relaciones existentes.

De igual modo sucede para la recuperación de información de regiones de una imagen almacenada en el sistema. Para la ejecución de una consulta con restricciones sobre las características de las regiones, aquel agente que posee un mayor número regiones que cumplen los criterios, es lógico que tarde más en recuperarlas. Pero es importante que las regiones estén balanceadas entre los distintos agentes, para que los tiempos necesarios para recorrer los índices buscando y recuperando aquellas regiones que cumplen los criterios sea parecido en todos los agentes que contienen la imagen.

Por último, cabe mencionar que los tiempos de segmentación y procesamien-

to de las imágenes pueden ser mejorados, a pesar del buen comportamiento del prototipo. Como se ha comentado anteriormente en este libro, el algoritmo de división utiliza la imagen desde el fichero en el disco duro directamente. El trabajar directamente sobre el fichero de la imagen en el disco duro, es una limitación considerable, ya que la velocidad de lectura de un disco duro de un ordenador de sobremesa, actualmente 7.200 r.p.m., es considerablemente menor que la velocidad de lectura de la memoria RAM o que un disco duro de un servidor.

No obstante, trabajar con imágenes tan grandes en memoria RAM es prácticamente inviable, como también se ha comentado anteriormente. Por ejemplo la imagen 3 de test (21.601×10.801 píxeles), ocuparía en memoria, usando un entero para cada píxel, 890 MegaBytes (solo la imagen). Para su tratamiento, además de mantener en memoria la imagen necesitaríamos memoria para mantener las estructuras de datos comunes para el tratamiento de imágenes morfológico, por ejemplo, para saber si un píxel ha sido tratado. Si para este fin utilizáramos un bit para marcar si un píxel esta tratado o no, en esta imagen necesitaríamos casi 223 MB. Además, en nuestro caso que queremos obtener las regiones de la imagen con sus etiquetas, raster y características, etc., por lo que se necesitarían otras estructuras de datos extra.

Otro factor que ralentiza un poco el proceso de segmentación y almacenamiento desarrollado es que en el paso de división se incluye la tarea de serializar las regiones para almacenarlas en la base de datos. Teniendo que generar ficheros con el formato tabular correspondiente.

Igualmente para los pasos de almacenamiento y recuperación, es determinante la velocidad del disco duro de los ordenadores que contienen la base de datos. Esto es debido a que, aunque los SGBD actuales están optimizados gracias al uso de índices y cachés, necesitan trabajar sobre el disco duro asiduamente.

Capítulo 6

Conclusiones y futuras líneas de trabajo

En el presente capítulo de conclusiones, en primer lugar, en la sección 6.1, se enumeran los objetivos fijados al comienzo de la tesis doctoral para examinar cómo se han llevado a cabo su cumplimiento.

A continuación, se exponen en la sección 6.2 las conclusiones de la presente tesis doctoral. Las secciones 6.3 y 6.4 comentan las mayores dificultades encontradas durante el desarrollo de la tesis y las futuras líneas de trabajo propuestas, respectivamente.

Finalmente, en la sección 6.5 se enumeran las publicaciones en congresos y revistas realizadas por el autor y que están relacionadas con la presente tesis doctoral.

6.1. Consecución de los objetivos

Se van a recapitular los objetivos planteados en la introducción de la presente tesis doctoral para examinar su cumplimiento. Estos objetivos iniciales fueron fijados para probar la validez la hipótesis de partida.

Los objetivos fijados fueron:

1. Creación de una estructura de datos distribuida, implementada en una base de datos relacional, para almacenar imágenes.

Cumplimiento de los hitos:

- Se ha desarrollado una estructura de datos de tipo grafo implementada sobre una base de datos relacional. Dicha estructura de datos se presenta en las sección 3.1, y, como se comenta, es capaz de almacenar las regiones de una imagen de entrada, así como una amplia variedad de descriptores de dichas regiones. Las relaciones existentes entre las distintas regiones de la imagen también son guardadas en la estructura de datos (ver tabla 3.1).

2. Desarrollar algoritmos y métodos para la división, almacenamiento y procesamiento distribuido de una imagen.

   Cumplimiento de los hitos:

   - Se ha implementado un novedoso algoritmo de división, explicado en la sección 3.2.1. El algoritmo está diseñado para preservar el número de regiones iniciales de la imagen original, de tal manera que sea el mismo número que la suma de las regiones de todas las sub-imágenes resultantes. Además, mantiene las vecindades del grafo de la imagen inicial en las sub-imágenes separadas. Estas propiedades son generalmente necesarias para poder aplicar filtros morfológicos basados en grafo de forma distribuida.
   - La estructura de datos, comentada anteriormente, ha sido adaptada para poder almacenar una imagen de forma distribuida (sección 3.2). Así las sub-imágenes obtenidas en el algoritmo de división pueden ser almacenadas de forma separada en distintas bases de datos que implementan la estructura de datos. Gracias a esta adaptación de la estructura de datos, el tiempo necesario para almacenar una imagen de entrada completa se reduce cuando se utiliza más de una base de datos. Datos más concretos de esta reducción del tiempo de almacenamiento se pueden consultar en la sección 5.1.
   - Se han adaptado algunos operadores básicos y algoritmos de la Morfología Matemática para ser ejecutados de forma distribuida. Dichos

operadores se explican en la sección 3.2.3. Una característica novedosa es que estos algoritmos y operadores es que pueden ser aplicados directamente sobre la estructura de datos. Esta característica original implica que no es necesario recuperar la información de la imagen almacenada en la estructura de datos (en la base de datos), para poder realizar las operaciones. Esta característica facilita la aplicación de estos operadores y algoritmos sobre imágenes de gran tamaño, ya que se reducen los requisitos de memoria volátil para dichos métodos.

3. Implementar un sistema que facilite el uso y evaluación de los métodos distribuidos que sean desarrollados.

   Cumplimiento de los hitos:

   - Se ha desarrollado un sistema distribuido multi-agente que usa: (i) la estructura de datos distribuida, (ii) el algoritmo de división de imágenes, (iii) y los operadores y métodos distribuidos de Morfología Matemática, que se han desarrollado en la presente tesis.

     Este sistema se presenta en el capítulo 4. Gracias al uso de todos estos componentes y métodos desarrollados, el sistema es capaz de almacenar y procesar imágenes de gran tamaño usando ordenadores de sobremesa.

     Para el sistema desarrollado se ha diseñado un planificador de tareas y de balanceo de carga diseñado específicamente para mejorar el tiempo de procesamiento de las tareas sobre imágenes muy grandes.

   - Para que el sistema y los métodos creados sean accesibles y puedan ser usados en un futuro por otras personas e instituciones se ha creado un servicio software de *Cloud Computing* descrito en la sección 4.4. Este servicio ofrece acceso a las funcionalidades creadas a través de un interfaz de programación.

     Para facilitar el uso a clientes que no sean desarrolladores de software, se ha creado un portal web como cliente del servicio de *Cloud Computing* (véase la sección 7.6). Este cliente web implementado facilita la evaluación del sistema y de los métodos creados, ya que, gracias a dicho

interfaz se pueden inspeccionar el estado de las imágenes almacenadas y los tiempos de procesamiento requeridos para aplicar distintas filtros y operaciones sobre las imágenes.

4. El sistema debería poder ser mantenido y ampliado de una forma sencilla. También, sería deseable que el sistema no necesitara grandes recursos computacionales para su funcionamiento.

   Cumplimiento de los hitos:

   - El sistema ha sido probado usando 6 PCs clónicos de sobremesa cuyas capacidades de procesamiento, almacenamiento y memoria son las de un ordenador de gama media de la actualidad.
   - Gracias a que el sistema ha sido implementado usando agentes distribuidos, como se ha comentado en el capítulo 5, es un sistema escalable. Es decir, el sistema es fácilmente ampliable: simplemente se han de añadir nuevos agentes al sistema.
   - Se ha creado un 'ordenador virtualizado' para ejecutar un agente de los que forman el sistema. La virtualización se ha realizado usando la aplicación *Oracle VirtualBox*. Se el ordenador virtual creado se ha instalado *Ubuntu 11.04 Server* como sistema operativo y se ha instalado el software necesario para poder ejecutar un agente y la base de datos que implementa la estructura de datos. De esta manera, para crear un nuevo agente, bastaría con (i) cambiar la dirección IP y nombre de red del sistema operativo (de la máquina virtual), (ii) dar un identificador nuevo al agente para el sistema y la dirección del agente índice de recursos (esto se hace editando un fichero de configuración). Con estos dos sencillos pasos se puede registrar y usar el nuevo agente en el sistema.

     Por lo tanto, gracias a la virtualización, la ampliación del número de agentes del sistema se puede realizar añadiendo una nueva máquina virtual. De la misma manera, la copia de seguridad de los agentes del sistema es fácilmente realizable.

6.2. Conclusiones

Durante la investigación de presente tesis doctoral se han desarrollado nuevos métodos y algoritmos para la división, almacenamiento y procesamiento de imágenes de gran tamaño de forma distribuida. Como marco de prueba para la utilización de dichos métodos, se ha creado un sistema multi-agente que, utilizando dichos métodos y algoritmos, es capaz de dividir una imagen inicial en varias sub-imágenes, y almacenar estas sub-imágenes de forma distribuida.

La distribución de las sub-imágenes que componen una imagen en distintos agentes, como se comenta en el capítulo 5, ayuda a reducir el tiempo necesario para la extracción de regiones y su almacenamiento. Así mismo, se ha comprobado en la fase de experimentación que el rendimiento de las operaciones morfológicas basadas en grafo implementadas (erosión, dilatación, apertura, cierre y watershed) depende directamente del número de regiones de la imagen sobre la que se aplican. Como consecuencia de esto, cuando la imagen se distribuye para ser procesada, los tiempos de procesamiento se reducen. Esto es debido a que el tiempo de procesamiento se reparte entre los distintos agentes.

De igual modo sucede para la recuperación de información de regiones de una imagen almacenada en el sistema. Cuando se realiza una consulta para obtener regiones de una imagen que posean ciertas características, el tiempo necesario para recuperar esas regiones se reduce cuanto mayor sea el número de agentes en los que se ha distribuido la imagen. No obstante, tanto para el procesamiento como para la recuperación de información, es importante que las regiones estén balanceadas entre los distintos agentes.

Un objetivo adicional de la presente tesis doctoral era poder ofrecer los métodos de almacenamiento, análisis y recuperación de imágenes creados, a otras organizaciones e investigadores que no dispongan de recursos computacionales. Por ello, el sistema ofrece las funcionalidades a través de un servicio de computación en nube. Este servicio ofrece la posibilidad de almacenar y procesar grandes colecciones de imágenes y, más específicamente, imágenes muy grandes. Gracias a estas características, una organización podría usar el sistema como un sistema virtual de gestión de imágenes, para administrar su colección de imágenes privadas, manteniéndola indexada y accesible para sus usuarios.

Adicionalmente, el sistema permite a un usuario publicar y compartir sus imágenes con otros usuarios del sistema. Gracias a esta característica, se podrían crear colecciones de imágenes compartidas y públicas. Esta colecciones, en campos como por ejemplo la medicina, son de gran utilidad para enseñanza y ayuda al diagnóstico.

6.3. Complicaciones encontradas durante el desarrollo de la tesis

Las principales complicaciones encontradas en el transcurso de la investigación son consecuencia principalmente del objeto de estudio, las imágenes de gran tamaño.

Para realizar el tratamiento de una imagen de gran tamaño, usando un ordenador de características convencionales, nos encontramos inicialmente varios problemas:

- Para procesar una imagen muy grande, un ordenador de sobremesa convencional no posee suficiente memoria para mantener la estructura de datos que contiene toda la información de la imagen completa para ser procesada.

  Solución:

  Se ha decidido crear la estructura de datos usando un Sistema Gestor de Bases de Datos Relacional. De tal manera que la información de la imagen no está en la memoria volátil del ordenador, sino que está almacenada en el disco duro. Una vez almacenada la imagen, los métodos de procesamiento creados actúan directamente sobre la base de datos, evitando el tener que mantener la información en la memoria volátil.

- En muchos campos, como la medicina, no se pueden preprocesar las imágenes para almacenarlas ya que se perderían detalles de la imagen. Debido a la gran cantidad de información a almacenar en la base de datos, sobre todo a la hora de insertar los píxeles de la imagen, el proceso de almacenamiento requiere tiempo. Además, la velocidad de un disco duro convencional (en la

actualidad 7200 r.p.m.) es una limitación, ya que la base de datos trabaja sobre ficheros en el disco duro.

Solución:

Se configura el SGBD para trabajar con división de tablas, para que los ficheros de las tablas de píxeles y regiones de las imágenes tengan un tamaño menor y sean más manejables. Aún así, debido a que todos los ficheros se encuentran en el mismo disco duro, el proceso sigue siendo algo lento.

Por ello, se modifica la estructura de datos para trabajar de forma distribuida. De esta manera, una imagen es dividida en sub-imágenes y éstas son distribuidas en varios servidores. Así mismo, el almacenamiento y la recuperación de los datos de una misma imagen se realiza de forma paralela en distintos servidores. Esto ayuda a reducir el tiempo necesario para el almacenamiento y procesamiento de una imagen.

- Al realizar el almacenamiento distribuido, las imágenes tienen que ser divididas. Se pretende realizar el tratamiento morfológico de las sub-imágenes a nivel de región y a la vez reducir el tráfico de información por la red al mínimo. Por ello, la división lineal de una imagen no es válida, ya que usando este tipo de división sólo mantienen las vecindades de la imagen a nivel de píxel, no a nivel de región como necesitamos para el procesamiento morfológico.

  Solución:

  Se ha desarrollado un algoritmo de división de imágenes, capaz de dividir una imagen de entrada en varias sub-imágenes, manteniendo las vecindades a nivel de región (necesarias generalmente para el procesado morfológico a nivel de región). Dicho algoritmo se ha diseñado de forma que la imagen se recorre una sola vez para ser procesada por completo, al igual que se haría usando una división lineal. Este algoritmo original se ha desarrollado ya que no se encontró ninguno que cumpliera las condiciones requeridas en la bibliografía.

- A la hora de realizar comprobaciones, sobre las bases de datos, para comprobar la validez de los métodos distribuidos creados, el proceso de com-

probación se complica cuantos más agentes (con su propia base de datos) existan. También la recopilación de los tiempos de ejecución se hace más compleja al incrementarse el número de agentes en el sistema.

Solución:

Se ha creado el sistema de agentes con utilidades para recavar información de los distintos agentes. Dicha información proveniente de todos los agentes es almacena en el agente índice de recursos.

También se ha desarrollado un interfaz gráfico gracias al cual se pueden inspeccionar la información relativa a los resultados del procesamiento de las imágenes.

- A la hora de realizar pruebas con el sistema, no ha sido fácil encontrar imágenes de tamaño muy grande.

  Solución:

  Se han utilizado partes de imágenes de la NASA, como imágenes de test, así como digitalizaciones de alta calidad usando escáneres.

6.4. Futuras líneas de trabajo

Actualmente el sistema es capaz de almacenar imágenes de gran tamaño, una posible ampliación del sistema es su utilización para el almacenamiento de secuencias de imágenes. Estas secuencias, pueden ser vistas como imágenes con tres dimensiones. Ya sea la tercera dimensión de carácter temporal, como en los vídeos, o espacial, como por ejemplo en secuencias de imágenes de tomografías.

Se podría aprovechar la característica multi-agente del sistema para la creación de un sistema de *streamming* de representaciones 3D. Cada agente mandaría los datos que posee y el cliente reproduciría la secuencia, de una forma similar a la que actúan los canales *p2p* en internet.

El sistema es capaz de regenerar las imágenes a partir de los datos almacenados en la estructura de datos, pero este apartado del sistema podría mejorarse, ya que no siempre es necesario generar la imagen con su tamaño original. Podrían

adaptarse los conceptos de representación piramidal de vistas previas, como las que se usan en sistemas GIS, para reducir el tiempo de visualización de las imágenes.

En este mismo sentido, la representación y almacenamiento de los píxeles de la imagen en la estructura de datos se podría optimizar. Para ello se podrían eliminar aquellos píxeles que pertenezcan al fondo de la imagen (es decir, que sean de un determinado color) de la estructura de datos. Otra posibilidad sería obtener una representación vectorial de las regiones iniciales que contienen muchos píxeles. De esta manera la cantidad de información a almacenar se reduciría, ya que el almacenamiento de los millones de píxeles de una imagen es el paso que más tiempo requiere en el paso de almacenamiento de una imagen en el sistema.

Otra posibilidad podría ser extraer sólo regiones de interés que sigan determinado patrón de características, y dejando sin almacenar en la base de datos el resto de regiones que no cumplen las condiciones de similitud dadas.

La presente tesis doctoral se ha centrado principalmente en la implementación de métodos de tratamiento morfológico a nivel de región, usando estructuras de tipo grafo. Pero la estructura de datos puede ser usada para aplicar otros tipos de filtros como por ejemplo filtros lineales, o filtros morfológicos a nivel de píxel. Se podría ampliar el sistema implementado nuevos filtros y operaciones para imágenes de gran tamaño.

Por otro lado, los avances en los distintos sistemas gestores de bases de datos han dado lugar a nuevos paradigmas de bases de datos distintos al modelo relacional como pueden ser bases de datos orientadas a objetos o bases de datos XML. A la hora de almacenar instancias de objetos provenientes de lenguajes orientados a objetos, como es nuestro caso con Java, existen estudios comparativos que indican que las bases de datos orientadas a objetos son más rápidas que las relacionales (jpab, 2011). Una posible línea de trabajo podría ser el adaptar el sistema y la estructura de datos para usar un sistema gestor de base de datos orientada a objetos, usando por ejemplo *Hibernate* de Java. De esa manera se podrían comparar los tiempos de los tiempos de ejecución con ambas bases de datos y ver si realmente se mejora.

A la hora de crear un sistema de gestión de imágenes en la nube (*Cloud Computing*), las primeras cuestiones que se plantean siempre son acerca de la seguridad

del sistema. Aunque estas cuestiones generalmente tienen más connotación legal que técnica, una posible ampliación del sistema sería el uso de certificados para encriptar las transmisiones de datos entre los distintos agentes. Otros aspectos legales para el uso del sistema, como la ubicación de la información o la protección de los datos personales, varía según las leyes de cada país.

6.5. Publicaciones

1. R. Bahillo, M. García-Remesal, D. Pérez, R. Alonso-Calvo, B. Romero, J.C. Llorente, G. Martínez, V. Barbado, C. Moral, A. Martínez-Agra, F. Martín-Sanchez, A. Sousa, I.C. Oliveira, y V. Maojo. **Uso de tecnologías de Agentes para la integración de Bases de Datos Cínicas y Genéticas**. En Actas de INFORSALUD 2004. Madrid (España), 2004.

2. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez, M.C. Zarcero, A. Sousa, J.L. Oliveira, I. Castro-Oliveira, M. Santos, y A. Babic. **Designing New Methodologies for Integrating Biomedical Information in Clinical Trials**. En Proceedings de EUROMISE 2004. Praga (República Checa), 2004.

3. D. Pérez, V. Maojo, M. García-Remesal y R. Alonso-Calvo. **Biomedical Ontologies in Post-Genomic Information Systems**. En Proceedings de IEEE BIBE 2004. T'aichung (Taiwán), 2004. Core Ranking: C.

4. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez y A. Sousa. **ARMEDA II: Suporting Genomic Medicine through the Integration of Medical and Genetic Databases**. En Proceedings de IEEE BIBE 2004. T'aichung (Taiwán), 2004. Core Ranking: C.

5. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez, A. Sousa-Pereira, y A. Babic. **ARMEDA II: Integrated Access to Heterogeneous Biomedical Databases**. En

Proceedings de MEDINFO 2004. San Francisco, CA (Estados Unidos de América), 2004. Core Ranking: B.

6. D. Pérez-Rey, V. Maojo, M. García-Remesal, R. Alonso-Calvo, H. Billhardt, F. Martín-Sanchez, y A. Sousa. **ONTOFUSION: Ontology-based integration of genomic and clinical databases**. Computers in Biology and Medicine. 2006. Publicación JCR: factor de impacto: 1.07 (Q1).

7. V. Maojo, M. García-Remesal, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez. **Designing New Methodologies for Integrating Biomedical Information in Clinical Trials**. 2006. Methods of Information in Medicine. Publicación JCR: factor de impacto: 1.68 (Q1).

8. R. Alonso-Calvo, V. Maojo, M. García-Remesal, F. Martín-Sanchez, H. Billhardt y D. Pérez-Rey. **An Agent and Ontology-based System for integrating Public Genomic Databases**. 2007. Journal of Biomedical Informatics. Publicación JCR: factor de impacto: 2.00 (Q1).

9. R. Alonso-Calvo, R., J. Crespo, M. García-Remesal, A. Anguita, V. Maojo. **On distributing load in cloud computing: A real application for very-large image datasets** En Proceedings de International Conference on Computational Science 2010. Core Ranking: A.

10. R. Alonso-Calvo, J. Crespo, V. Maojo, A. Muñoz, M. García Rojo, L. Pérez, J. Azpiazu **Cloud Computing en Salud: Sistema para administrar imágenes Biomédicas**. En Actas de INFORSALUD 2011. Madrid (España), 2011.

11. R. Alonso-Calvo, J. Crespo, V. Maojo, A. Muñoz, M. García-Remesal, D. Pérez-Rey. **Cloud Computing Service for Managing Large Medical Image Data-Sets Using Balanced Collaborative Agents**. En Book Series: Advances in Intelligent and Soft Computing. De 9th International Conference on Practical Applications of Agents and Multi-Agent Systems. Core C.

6.6. Agradecimientos

Este trabajo ha sido financiado en parte por el "Ministerio de Ciencia e Innovación" de España (Ref.: TIN2007-61768).

Capítulo 7

Conclusions

In this final chapter, Section 7.1 summarizes the objectives set at the beginning of the thesis and comments how they have been satisfied.

Then, Section 7.2 presents the conclusions of the thesis research. Sections 7.3 and 7.4 discusses the major difficulties encountered, and future lines of work research, respectively.

Finally, Section 7.5 lists in journal papers and conference communications published by the author and that are related with this thesis' work.

7.1. Achievement of objectives

In this section we are going to recapitulate the goals outlined in the introduction of this thesis. These initial objectives were set to test the validity of the initial hypothesis.

The objectives established were:

1. *A distributed data structure has to be developed. This data structure must be implemented in a relational database and it has to be capable of storing images.*

    - A graph-based data structure has been designed using a relational database schema. This data structure is explained in Section 3.1. It is capable of storing regions of an input image and a wide variety of descriptors for these regions. Relationships between different regions

of the image also are stored. More detailed information can be found in Table 3.1.

2. *It is necessary to develop methods and algorithms for the division, storage and processing of distributed images.*

   - A new dividing algorithm has been implemented, as explained in Section 3.2.1. This algorithm divides an input image into three sub-images. A key feature is that it is designed to preserve the number of initial flat-zone regions from the image. Thus, the sum of regions in all sub-images is the same that the number of regions in the original image. Besides, it also keeps the neighborhood of the graph of the initial image in separate sub-images. These two properties are usually necessary for applying distributed graph-based morphological filters.
   - The data structure discussed above has been adapted in order to be able to store an image in a distributed manner (see Section 3.2). Thus, the sub-images obtained from the division algorithm can be stored separately in the different databases that implement the data structure. Thanks to this adaptation of the data structure the time required to store an entire input image is reduced when using more than one database. More specific details of the reduction of storage time can be found in Section 5.1.
   - Some basic operations from mathematical morphology have been implemented to be executed in a distributed manner. These operators are explained in Section 3.2.3. A novel feature is that these algorithms and operators can be applied directly to the data structure database. This feature implies that, it is not necessary to recover the image information stored in the data structure (database) in order to perform operations. Moreover, It facilitates the application of these operators and algorithms on very large images, because volatile memory requirements are significantly reduced.

3. *A system prototype has to be created in order to facilitate the use and evaluation of the distributed methods that has been developed.*

- A distributed multi-agent system has been created that uses: (i) the distributed data structure, (ii) the dividing algorithm and, (iii) the distributed methods and operators from the mathematical morphology that have been developed in this thesis.

  The architecture and functionalities of the system are presented in Chapter 4. By using the proposed components and methods, the system prototype is able of store and process very large images using desktop computers.

  The system prototype possesses a load balancing and task scheduler that has been designed and developed for the prototype, and that reduces the processing time needed for storing and processing very large image.

  For the developed prototype, a novel scheduling and load balancing sub-system have been designed specifically in order to enhance storing and processing tasks when dealing with very large image.

- A Cloud Computing Software Service (SaaS) has been created for making methods and algorithms accessible to other people and institutions. This software service is explained in Section 4.4, and provides access to features developed in this thesis through a programming interface.

  To facilitate the utilization of the service by users that are not software developers, a web portal has been created. This web portal acts as an interface to the Cloud Computing Service; further details can be found in Section 7.6. This Web client facilitates the evaluation of implemented system and methods, because, the state of the stored images and the time required to apply different filters and operations on images can be easily inspected.

4. *The prototype system should be upgraded and maintained easily. It would be desirable that the system does not require large computational resources.*

   - The prototype has been deployed and tested using six desktop PCs. These computers have medium-range computational resources. Spe-

cifically, they have an Intel CoreQuad processor with 4 GB of RAM memory.

- A consequence of using distributed agents technology in the system prototype development is that this system can be upgraded easily, just adding new agents to the system, as discussed in Chapter 5.
- Virtualization technology has been used for creating a generic 'virtual computer' that contains a system agent. Virtualization has been achieved using Oracle VirtualBox. The virtual computer has Ubuntu 11.04 Server as operating system, and the software needed for deploying a system agent.

  Thus, the process to upgrade the system by adding a new agent is quite simple. Using the generic virtual machine, we have (i) to assign an IP address to the virtual machine operating system, and, (ii) to give a new ID to the agent for registration in the system resource index agent (this is done by editing a configuration file). After these two steps the new agent is included in the system.

  Therefore, thanks to virtualization, increasing the number of agents of the system can be achieved by adding a new virtual machine. Similarly, the system backup is easily attainable.

7.2. Conclusions

This work presents a new distributed data structure and novel methods and algorithms for dividing, segmenting, storing and processing very large images in a distributed manner. As a test framework, a distributed multi-agent system has been created as well, using these methods and algorithms. This system is capable of dividing one input image into several sub-images, and storing the regions contained in every sub-image distributedly into several different agents.

The sub-image distribution into different agents in the system, as commented in Chapter 5, helps to reduce the time required for the region extraction and storage steps. Also, it has been shown in the experimentation that the performance

of the implemented graph-based morphological operations (erosion, dilation, opening, closing and watershed) depends directly on the number of regions of the image in which they are applied. As a result, when one image is distributed to be processed, the processing time are reduced. This is so because the processing time is divided among the different agents.

This applies similarly to information retrieval from regions of an image stored in the system. The time required to retrieve some regions of an image that accomplish some constraints is reduced when the number of agents where the image is distributed is increased. But, for both processing and information retrieval, it is important that regions are balanced between the different agents.

An additional objective of this thesis was to provide the created storage, processing and retrieval methods to other institutions and researchers that do not have enough computational resources. Therefore, the system offers its functionalities through a cloud computing service. So, by using this service, users could manage, store and process large image collections (and especially composed of very-large images) and to perform posterior region-information retrieval.

Additionally, the system allows users to publish and share their own images with other users of the system. With this feature, shared public image collections can be created. These collections in fields such as medicine are very useful, for example for teaching or diagnostic purposes.

7.3. Difficulties during the PhD research

The focus of the thesis' work on very large images imposes difficult constraints when personal computers are used.

The main difficulties encountered in this thesis' research have been the following:

- Usually, processing very large images using personal computers is not possible because this kind of computers does not have enough RAM memory for maintaining the whole image within a data structure.

  Solution:

To solve this problem, the data structure has been implemented using a relational database management system. Thus, the information from an image is stored in the hard disk into the database instead of being stored in main memory.

Once images are stored in the data-structure inside a database, in order to process them, methods capable of applying filters and operations directly on the database have been created. This feature solves the problem of needing a great amount of volatile memory for the processing of very-large images.

- In some areas, such as in medicine, images should not be pre-processed in order to preserve all information and details available. There exists an enormous amount of information contained in the regions of the images that have to be inserted in the data-structure. Therefore, it is necessary some time for storing all these information, moreover, when using personal computers with conventional hard drives (nowadays at 7200 r.p.m.) that limit database performance.

  Solution:

  The database management system has been configured for using the capability of working with table partitioning. By using table partitioning, one table in the database is stored in several sub-tables, and each table is a different physical file in the hard disk. Hence, files in the hard disk that contain tables become smaller and easier to manage. However, in the other hand, all these sub-tables are still stored in the same physical hard disk, and this affects negatively to the performance when storing data.

  Consequently, the data-structure has been modified for working in a distributed manner. One image could divided in several sub-images and those sub-images could be stored in different distributed computers. Thus, storing and retrieving information from images is done parallelly among different computers. This fact aids to reduce the time needed for storing, processing and retrieving information from images.

- Input images have to be divided for storing them in a distributed manner. In this thesis, graph-based mathematical morphology filters and operators

for processing images are used. Thus, neighbor information has to be preserved when images are divided. But, lineal dividing algorithms only preserves neighbor information at the pixel-level, so a new region-based dividing algorithm has to be created.

Solution:

A novel dividing algorithm has been designed since no similar algorithm has been found in literature. This new dividing algorithm divides one input image into multiple sub-images preserving the region-level neighborhood and the original number of regions in the image.

- The process for testing the created methods gets more complex when the number of agents in the system augments. Obtaining execution times from different agents is also a complex task.

  Solution:

  A multi-agent system has been created. One component of this system is a directory index where information from the agents in the system is stored. This information includes the execution times of algorithms.

  In order to inspect the results and execution times, a graphical web interface has been developed.

- It is not easy to find open-access to collections of very-large images for testing data structure, methods and algorithms that have been proposed.

  Solution:

  Some images from the NASA image gallery have been used for testing the developed methods and algorithms developed. Also, some high-resolution digitized images from scanners have been used.

7.4. Future work

Currently, the proposed system prototype is capable of storing very large images. A possible extension of the system and the data-structure is to adapt

them for storing image sequences. These image sequences could be seen as three-dimensional images where the third dimension is referred, for example, to the time such as in videos, or to the space as in digital tomography.

Also, the multi-agent system could be upgraded for creating a 3-D projection streaming system where each agent sends relevant data contained in its database. Thus, the client will receive data from several different agents and it will be able to reproduce 3-D sequences. This extension would be similar to peer-to-peer (p2p) television channels in the Internet.

The prototype created in this thesis is capable of regenerating the original image from the stored region information. However, it is not always necessary to rebuild the original image. In order to reduce the time needed for visualizing images, some concepts from Geographical Information Systems, such as pyramidal representation, could be implemented. Following this line of work, pixel representation and storage in the data-structure could be optimized in different ways, such as: (i) deleting pixels belonging to the image background (or simply, not storing them); (ii) changing the current raster representation of regions and use a vectorial representation instead, for reducing the amount of data stored and speeding up region visualization; and (iii) storing in the database only those regions that accomplish some pre-defined patterns or restrictions, limiting this way the amount of regions stored in the system.

Although the data structured has been created mainly focused in supporting graph-based methods mathematical morphology, it could be used for developing and applying other new filters on very-large images, such as, for example lineal filters, or pixel-level mathematical morphology filters.

Regarding database insertion optimization, some new database paradigms like object oriented databases or XML databases could be tested. Some comparisons exist, that intend to demonstrate that, when storing objects from object-oriented programming languages, object-oriented databases are generally faster than traditional relational databases (jpab, 2011). Thus, a possible upgrade of the system could be adapting the data-structure and implementing it inside an object-oriented database for using it from Java (for example using hibernate). After that, a performance comparison could be made between different data structure implementations (relational versus object-oriented).

When talking about Cloud Computing and transferring private information over the Internet, questions about security commonly appear. Some of those issues are legal, and they often vary for each country.

7.5. Publications

1. R. Bahillo, M. García-Remesal, D. Pérez, R. Alonso-Calvo, B. Romero, J.C. Llorente, G. Martínez, V. Barbado, C. Moral, A. Martínez-Agra, F. Martín-Sanchez, A. Sousa, I.C. Oliveira, y V. Maojo. **Uso de tecnologías de Agentes para la integración de Bases de Datos Cínicas y Genéticas**. In 'Actas de INFORSALUD 2004'. Madrid (España), 2004.

2. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez, M.C. Zarcero, A. Sousa, J.L. Oliveira, I. Castro-Oliveira, M. Santos, y A. Babic. **Designing New Methodologies for Integrating Biomedical Information in Clinical Trials**. In Proceedings of EUROMISE 2004. Praga (República Checa), 2004.

3. D. Pérez, V. Maojo, M. García-Remesal y R. Alonso-Calvo. **Biomedical Ontologies in Post-Genomic Information Systems**. In Proceedings of IEEE BIBE 2004. T'aichung (Taiwán), 2004. Core Ranking: C.

4. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez y A. Sousa. **ARMEDA II: Suporting Genomic Medicine through the Integration of Medical and Genetic Databases**. In Proceedings of IEEE BIBE 2004. T'aichung (Taiwán), 2004. Core Ranking: C.

5. M. García-Remesal, V. Maojo, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez, A. Sousa-Pereira, y A. Babic. **ARMEDA II: Integrated Access to Heterogeneous Biomedical Databases**. In Proceedings of MEDINFO 2004. San Francisco, CA (Estados Unidos de América), 2004. Core Ranking: B.

6. D. Pérez-Rey, V. Maojo, M. García-Remesal, R. Alonso-Calvo, H. Billhardt, F. Martín-Sanchez, y A. Sousa. **ONTOFUSION: Ontology-based integration of genomic and clinical databases. Computers in Biology and Medicine**. 2006. JCR Impact factor: 1.07 (Q1).

7. V. Maojo, M. García-Remesal, H. Billhardt, J. Crespo, R. Alonso-Calvo, D. Pérez, F. Martín-Sanchez. **Designing New Methodologies for Integrating Biomedical Information in Clinical Trials**. 2006. Methods of Information in Medicine. JCR Impact factor: 1.68 (Q1).

8. R. Alonso-Calvo, V. Maojo, M. García-Remesal, F. Martín-Sanchez, H. Billhardt y D. Pérez-Rey. **An Agent and Ontology-based System for integrating Public Genomic Databases**. 2007. Journal of Biomedical Informatics. JCR Impact factor: 2.00 (Q1).

9. R. Alonso-Calvo, R., J. Crespo, M. García-Remesal, A. Anguita, V. Maojo. **On distributing load in cloud computing: A real application for very-large image datasets**. In Proceedings of International Conference on Computational Science 2010. Core Ranking: A.

10. R. Alonso-Calvo, J. Crespo, V. Maojo, A. Muñoz, M. García Rojo, L. Pérez, J. Azpiazu **Cloud Computing en Salud: Sistema para administrar imágenes Biomédicas**. In 'Actas de INFORSALUD 2011'. Madrid (España), 2011.

11. R. Alonso-Calvo, J. Crespo, V. Maojo, A. Muñoz, M. García-Remesal, D. Pérez-Rey. **Cloud Computing Service for Managing Large Medical Image Data-Sets Using Balanced Collaborative Agents**. On Book Series Advances in Intelligent and Soft Computing. From 9th International Conference on Practical Applications of Agents and Multi-Agent Systems. Core C.

7.6. Acknowledgements

This work has been supported in part by “Ministerio de Ciencia e Innovación” of Spain (Ref.: TIN2007-61768).

Anexo A : Implementación de un portal web como cliente del servicio de Cloud Computing

Con el fin de ofrecer a usuarios finales el acceso al sistema de administración de imágenes de gran tamaño, se ha creado un interfaz gráfico basado en la tecnología web. Como se ha comentado en la sección anterior el servicio software de *Cloud Computing* es privado, por lo que sólo pueden acceder al él, a través del interfaz gráfico, aquellos usuario con clave en el sistema, como se muestra en la figura 1.

Una vez que un usuario se ha autenticado, se le ofrece el poder utilizar el sistema multi-agente de almacenamiento y procesamiento de imágenes de gran tamaño, a través del servicio software de *Cloud Computing*. El primer paso para usar el sistema es introducir sus imágenes en el servidor del interfaz gráfico. El usuario puede subir sus imágenes a través de un formulario (en la parte superior del listado en la imagen 2) o si tienen un tamaño demasiado grande, a través de una conexión sftp. El usuario autenticado puede ver el listado de sus imágenes y puede eliminarlas o pedir que sean procesadas para ser introducidas en el sistema multi-agente, y por lo tanto, en la estructura de datos distribuida. En la figura 2 se puede ver el listado de imágenes subidas por un usuario y los botones de eliminar y procesar (a la derecha del nombre de las imágenes).

El usuario también puede obtener el listado de todas las imágenes que ha almacenado en el sistema multi-agente. Dichas imágenes ya están insertadas en

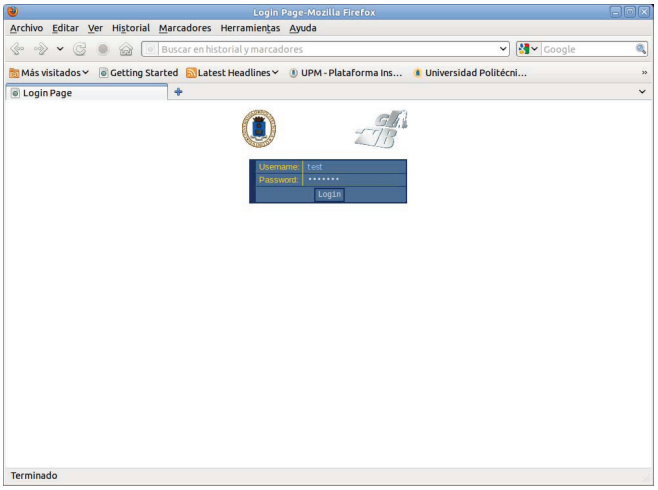

Figura 1: Pantalla de inicio para entrar al servicio de administración de imágenes de gran tamaño.

la estructura de datos desarrollada y están listas para aplicarles algún filtro u operación. Un listado de imágenes almacenadas de ejemplo se muestra en la figura 3.

Las imágenes que ya están introducidas en el sistema pueden ser accedidas por medio del listado mostrado en la imagen 3. Accediendo a una imagen concreta, se da la posibilidad de aplicar los filtros y operaciones disponibles sobre la imagen en el sistema.

Así mismo, se obtiene toda la información de las sub-imágenes componen la imagen, dónde se encuentran almacenadas y se muestran dichas sub-imágenes.

También se obtiene información de los filtros y operaciones que se han efectuado sobre la imagen, mostrando los tiempos de ejecución de los filtros en los distintos agentes.

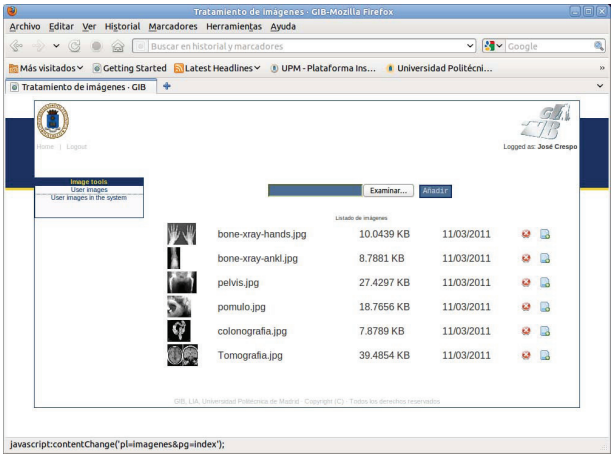

Figura 2: Imágenes que el usuario ha introducido en el sistema pero que están aún sin procesar.

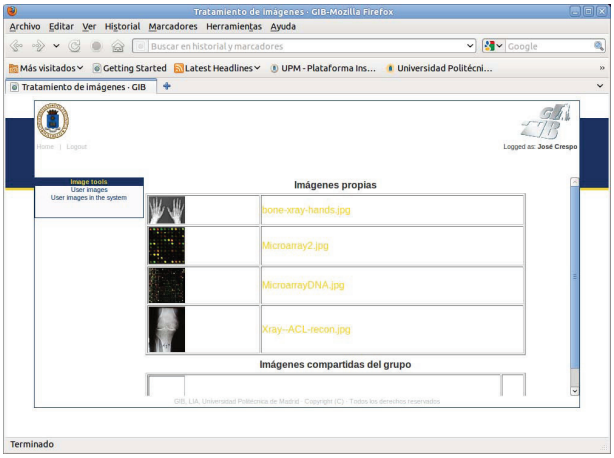

Figura 3: Imágenes que han sido procesadas y almacenadas distribuidamente en el sistema.

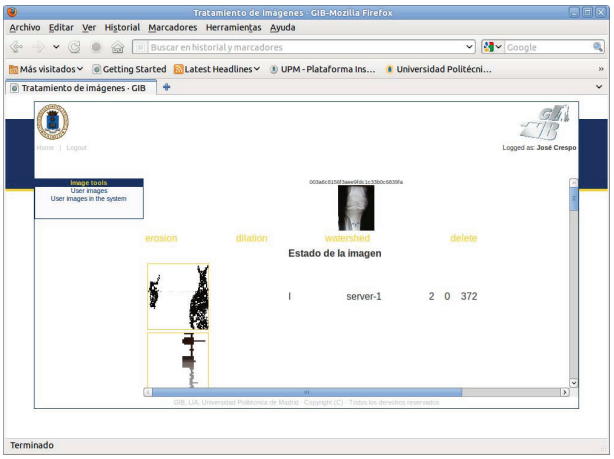

Figura 4: Descripción de la imagen de usuario y posibles operaciones a ejecutar sobre ella.

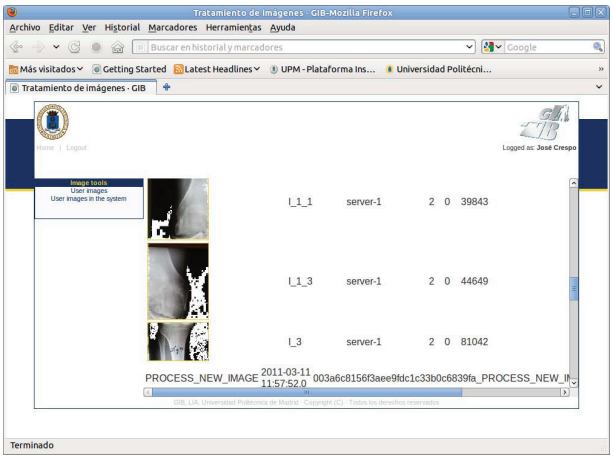

Figura 5: Conjunto de sub-imágenes que componen la imagen original.

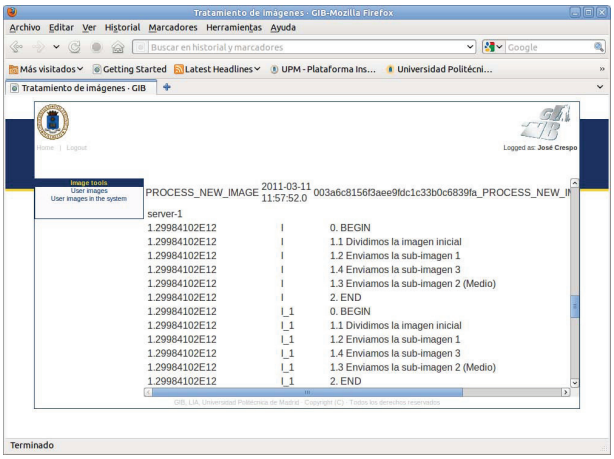

Figura 6: Desglose de tiempos de ejecución de la imagen.

. ANEXO A : IMPLEMENTACIÓN DE UN PORTAL WEB COMO CLIENTE DEL SERVICIO DE CLOUD COMPUTING

Bibliografía

Alonso-Calvo, R., Crespo, J., García-Remesal, M., Anguita, A., y Maojo, V. (2010). On distributing load in cloud computing: A real application for very-large image datasets. *Procedia Computer Science*, *1*(1), 2669 - 2677. Disponible en http://www.sciencedirect.com/science/article/B9865-506HM1Y-BN/2/7e2436f6872fe813a66952ee80d76bc3 (ICCS 2010)

Alonso-Calvo, R., Maojo, V., García-Remesal, M., Martín, F., Billhardt, H., y Pérez-Rey, D. (2007). An agent and ontology-based system for integrating public gene, protein, and disease databases. En (Vol. 40(1), pp. 17–29). 39, 77

Arens, Y., Hsu, C., y Knoblock, C. (1998, San Francisco, CA.). Query processing in the sims information mediator.
9

Bach, J. R., Fuller, C., Gupta, A., Hampapur, A., Horowitz, B., Humphrey, R., y cols. (1996). Virage image search engine: an open framework for image management. *Storage and Retrieval for Still Image and Video Databases IV*, *2670*(1), 76–87.

Baker, P., Brass, A., Bechhofer, S., Goble, C., Paton, N., y Stevens, R. (1998, Montreal). Tambis: Transparent access to multiple bioinformatics informa-

tion sources. an overview. *Proceedings of the Sixth Int. Conf. on Int. Syst. for Mol. Biol., ISMB98*. 9

Ben Langmead, J. L. M. P., Michael C Schatz, y Salzberg, S. L. (2009). Searching for SNPs with cloud computing. *Genome Biology*.

Ben Miled, Z., Li, N., Kellett, G., Sipes, B., y Bukhres, O. (2002, Nov.). Complex life science multidatabase queries. *Proc. of the IEEE*, *90*. 9

Beucher, S., y Meyer, F. (1993). The morphological approach to segmentation: the watershed transformation. En E. Dougherty (Ed.), *Mathematical morphology in image processing* (pp. 433–481). New York: Marcel Dekker. 11, 17

Brunner, D., y Soille, P. (2005). *Iterative area seeded region growing for multichannel image simplification* (Vol. 30; C. Ronse, L. Najman, y E. Decencière, Eds.). Springer Netherlands. Disponible en http://dx.doi.org/10.1007/1-4020-3443-1_36 (10.1007/1-4020-3443-1_36) 31

Burl, M. C., Fowlkes, C., Roden, J., Stechert, A., y Mukhtar, S. (1999). *Diamond eye: A distributed architecture for image data mining.* 3

Cao, J., Spooner, D. P., Jarvis, S. A., y Nudd, G. R. (2005). Grid load balancing using intelligent agents. *Future Generation Computer Systems*, *21*(1), 135-149. 43

Carson, C., Thomas, M., Belongie, S., Hellerstein, J., y Mallik, J. (1999, June). Blobworld: A system for region based image indexing and retrieval. En *Visual information and information systems, proceedings of the third international conference visual'99, amsterdam, the netherlands* (p. 173-187). Springer. 27

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou,

Y., Ullman, J., y cols. (1994, Tokyo, Japan, Oct.). The tsimmis project: Integration of heterogeneous information sources. En (pp. 7–18). 9

Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, *1*(1), 9–36. Disponible en citeseer.nj.nec.com/519283.html

Codd, E. (1970). A relational model for large shared databanks. *Storage and Retrieval for Media Databases conference, SPIE Electronic Imaging, Santa Clara*, *13*(6), 377-387.

Crespo, J. (1993). *Morphological connected filters and intra-region smoothing for image segmentation*. Tesis Doctoral no publicada, Atlanta, GA, USA. (UMI Order No. GAX94-15632) 17

Crespo, J., y Maojo, V. (1998, April). New results on the theory of morphological filters by reconstruction. *Pattern Recognition*, *31*(4), 419–429. 22

Crespo, J., Schafer, R., Serra, J., Gratin, C., y Meyer, F. (1997). The flat zone approach: A general low-level region merging segmentation method. *Signal Processing*, *62*(1), 37–60. 17, 22

Crespo, J., Serra, J., y Schafer, R. (1993, Barcelona, May). Image segmentation using connected filters. En J. Serra y P. Salembier (Eds.), *Workshop on mathematical morphology* (pp. 52–57). 16, 22

Crespo, J., Serra, J., y Schafer, R. (1995, November). Theoretical aspects of morphological filters by reconstruction. *Signal Processing*, *47*(2), 201–225. 11, 22

Cross, A., y Hancock, E. (1998, November). Graph matching with a dual-step em algorithm. *IEEE Transactions on pattern analisys and Machine intelligence*, *20*(11).

Dancyger, K. (1999). *Techniques of film and video editing.* Barcelona: Gedisa.

Date, C., y Darwen, H. (1997). *A guide to the sql standard.* Addison-Wesley, New York.

Datta, R., Joshi, D., Li, J., James, y Wang, Z. (2006). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, *39*, 2007. 27

García-Rojo, M., Bueno-García, G., González-García, J., y Carbajo-Vicente, M. (2005). Preparaciones digitales en los servicios de anatomía patológica (i). aspectos básicos de imagen digital. *Revista Española de Patología*, *38*. Disponible en http://www.pgmacline.es/revpatologia/volumen38/vol38-num2/38-2n02.htm 3

Gardels, K. (1996). The open gis approach to distributed geodata and geoprocessing.
3

Goasdoué, F., Lattes, V., y Rousset, M. C. (2000). The use of carin language and algorithms for information integration: The picsel project. *International Journal of Cooperative Information Systems*, *9*, 383–401. 10

Goh, C. (1997). Representing and reasoning about semantic conflicts in heterogeneous information sources. *PhD Dissertation, MIT*. 10

Gonzalez, R., y Wintz, P. (1978). *Digital image processing.* Wiley and Sons.

Gormley, G. (1999, April). Scene break detection & classification in digital video sequences. Technical Report, School of Computer Applications Dublin City University.

Gudivada, V., y Raghavan, V. (1995, sep). Content based image retrieval systems. *Computer*, *28*(9), 18 -22. 3

Haralick, R., y Shapiro, L. (1992a). *Computer and robot vision. vol. I.* Reading, Massachusetts: Adison-Wesley Publishing Company.

Haralick, R., y Shapiro, L. (1992b). *Computer and robot vision. vol. II.* Reading, Massachusetts: Adison-Wesley Publishing Company.

Hastings, S. L., Langella, S., Oster, S., y Saltz, J. H. (2004, December). Distributed data management and integration framework: The mobius project. *Proceedings of the Global Grid Forum 11 (GGF11) Semantic Grid Applications Workshop*, 20–38. 3

Hauptmann, A., y Witbrock, M. (1997). Informedia news on demand: Multimedia information acquisition and retrieval. in Maybury, M. T., Ed, Intelligent Multimedia Information Retrieval, AAAI Press/MIT Press, Menlo Park, CA. Disponible en citeseer.ist.psu.edu/hauptmann97informedia.html 27

Hauptmann, A., y Witbrock, M. (1998, April, 22-24). Story segmentation and detection of commercials in broadcast news video. En (Vol. ADL98, p. 168-179). Advances in Digital Libraries. Disponible en citeseer.ist.psu.edu/hauptmann98story.html

H. Zhou, Y. J., y Yang, X. (2002). An improved parallel watershed algorithm for distributed memory system. En *Ica3pp '02: Proceedings of the fifth international conference on algorithms and architectures for parallel processing* (p. 310). Washington, DC, USA: IEEE Computer Society. 36, 38

Intel. (2011, Dec). Opencv home: http://opencv.willowgarage.com/wiki/. Disponible en http://opencv.willowgarage.com/wiki/ 54

Jacobson, I., Booch, G., y Rumbaugh, J. (1998). *The unified software development process.* Addison-Wesley.

Jacobson, I., Booch, G., y Rumbaugh, J. (1999). *The unified software development process*. Addison-Wesley.

Jain, A. (1989). *Fundamentals of digital image processing, prentice hall information and system sciences series (series editor: T. kailath)*. Englewood Cliffs: Prentice Hall.

jpab. (2011, Jun). Openjpa vs mysql comparison: http://www.jpab.org/openjpa/mysql/server/objectdb/objectdb/server.html. Disponible en http://www.jpab.org/OpenJPA/MySQL/server/ObjectDB/ObjectDB/server.html 99, 110

Klein, P., Sebastian, T., y Kimia, B. (2001). Shape matching using edit-distance: an implementation. En *Symposium on discrete algorithms* (p. 781-790). Disponible en citeseer.nj.nec.com/klein01shape.html

Knoblock, C., Minton, S., Ambite, J., Ashish, N., Muslea, I., Philpot, A., y cols. (2001). The ariadne apprach to web-based information integration. *International Journal of Cooperative Information Systems*, *10*, 145–169. 9

Leinberger, G. K. V. B. R., W. Karypis. (2000). Load balancing across near-homogeneous multi-resource servers. En *Heterogeneous computing workshop, 2000. (hcw 2000) proceedings. 9th* (p. 60-71). 43

Liu, L., y Pu, C. (1995, New York). The distributed interoperable object model and its application to large-scale interoperable database systems. 9

Liu, Y., Zhang, D., Lu, G., y Ma, W.-Y. (2007, January). A survey of content-based image retrieval with high-level semantics. *Pattern Recogn.*, *40*, 262–282. Disponible en http://dx.doi.org/10.1016/j.patcog.2006.04.045 27

Maojo, V., García-Remesal, M., Billhardt, H., Crespo, J., Alonso-Calvo, R., Pérez-Rey, D., y cols. (2006). Designing new methodologies for integrating biomedical information in clinical trials. *Methods of Information in Medicine 2006*, *2*(45), 180–185.

Marcotegui, B. (1996). *Segmentation de sequences d'images en vue du codage*. Ph. D. Thesis, École Nationale Supérieure de Mines de Paris. 16, 23

McGrath, D. (2001). *Editing and post-production*. Barcelona: Océano.

Meijster, A., y Roerdink, J. B. T. M. (1995). A proposal for the implementation of a parallel watershed algorithm. En *In proceedings of caip'95* (pp. 790–795). Springer Verlag. 36

Mena, E., Illarramendi, A., Kashyap, V., y Sheth, A. (2000). Observer: An approach for query processing in global information systems based on interoperation between pre-existing ontologies. *Distributed and parallel databases*, *8*, 223–271. 10

Meyer, F., y Beucher, S. (1990). Morphological segmentation. *J. Visual Commun. Image Repres.*, *1*(1), 21–45. 16, 17, 36

Miene, A., Dammeyer, A., Hermes, T., y Herzog, O. (2001, September 8). Advanced and adaptive shot boundary detection. ECDL 2001 WS Generalized Documents. Disponible en http://citeseer.nj.nec.com/miene01advanced.html

Miene, A., Hermes, T., Ioannidis, G., Fathi, R., y Herzog, O. (2002). Automatic shot boundary detection and classification of indoor and outdoor scenes. Proceedings of NIST Special Publication 500-251: The Eleventh Text REtrieval Conference (TREC 2002). Disponible en http://citeseer.nj.nec.com/588653.html

N.A.S.A. (2012). Visible earth project: Blue marble collection. *Official Website*. Disponible en http://visibleearth.nasa.gov/view_set.php?categoryID=2363 3, 28

Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., y cols. (1993, February, 2-3). The qbic project: Quering imagen by content using color, texture and shape. En (p. 173-187). San Jose, CA, Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases. 27

NIST. (2010, Dec). Definition of cloud computing (v. 15): http://csrc.nist.gov/groups/sns/cloud-computing/. Disponible en http://csrc.nist.gov/groups/SNS/cloud-computing/ 50

Novotni, M., y Klein, R. (2003, June). 3D Zernike descriptors for content based shape retrieval. En *The 8th acm symposium on solid modeling and applications*. Disponible en citeseer.nj.nec.com/novotni03zernike.html

O'Connor, N., Marlow, S., Murphy, N., Smeaton, A., Browne, P., Deasy, S., y cols. (2001). Físchlár: An on-line system for indexing and browsing broadcast television content. Disponible en citeseer.nj.nec.com/445633.html

Ostermann, L. G. (2002). *Hierarchical region based processing of images and video sequences: Application to filtering, segmentation and information retrieval*. Ph. D. Dissertation, Departament of Signal Theory and Communications. Universitat Politécnica de Catalunya. 23, 24

Pass, G., Zabih, R., y Miller, J. (1996). Comparing images using color coherence vectors. En (p. 65-73). Proceedings of ACM Multimedia. Disponible en citeseer.nj.nec.com/article/pass96comparing.html

Pérez-Rey, D., Maojo, V., García-Remesal, M., Alonso-Calvo, R., Billhardt, H.,

Martín, F., y cols. (2006). Ontofusion: Ontology based integration of genomic and clinical databases. *Computers in Biology and Medicine 2006*, *7-8*(36), 712-730. 10, 39, 77

Picard, D., Cord, M., y Revel, A. (2006). CBIR in distributed databases using a multi-agent system. En *Ieee international conference on image processing (icip'06)*. Disponible en http://publi-etis.ensea.fr/2006/PCR06

Pratt, W. (1991). *Digital image processing*. New York: John Wiley and Sons, 2nd Edition.

Rangarajan, A., Chui, H., y Mjolsness, E. (1999). A new distance measure for non-rigid image matching. En *Energy minimization methods in computer vision and pattern recognition* (p. 237-252). Disponible en citeseer.nj.nec.com/rangarajan99new.html

Ronse, C. (2005). Guest editorial. *Journal of Mathematical Imaging and Vision*, *1*(22), 103 – 105. 11

Rumbaugh, J., Jacobson, I., y Booch, G. (1998). *Unified modeling language reference manual*. Addison-Wesley.

Sadlier, D., Marlow, S., O'Connor, N., y Murphy, N. (2001). Automatic tv advertisement detection from mpeg bitstream. Proc. Int. Conf. on Enterprise Information Systems, ICEIS.

Salembier, P., y Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions in image processing*, *9*(4), 561–576. 16, 23

Salembier, P., y Serra, J. (1995). Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on Image Processing*, *4*(8), 1153–1160. 11

Serra, J. (1982). *Mathematical morphology. Volume i.* London: Academic Press. 11, 22, 33

Serra, J. (Ed.). (1988). *Mathematical morphology. Volume ii: theoretical advances.* London: Academic Press. 11, 22, 33

Serra, J., y Salembier, P. (1993, July). Connected operators and pyramids. En *Proceedings of spie, non-linear algebra and morphological image processing, san diego* (Vol. 2030, pp. 65–76). 11, 22

Shapiro, L., y Brady, J. M. (1992). Feature-based correspondence: An eigenvector approach. *Image and Vision Computing*, *10*(5), 283-288.

siue.edu. (2011, Dec). Cviptools home: http://www.ee.siue.edu/cviptools/. Disponible en http://www.ee.siue.edu/CVIPtools/ 54

Smeaton, A., Gilvarry, J., Gormley, G., Tobin, B., Marlow, S., y Murphy, N. (1999). An evaluation of alternative techniques for automatic detection of shot boundaries in digital video. School of Electronic Engineering and School of Computer Applications Dublin City University Glasnevin, Dublin , IRELAND. Disponible en citeseer.nj.nec.com/article/smeaton99evaluation.html

Smeaton, A., Lee, H., O'Connor, N., Marlow, S., y Murphy, N. (2002). Tv news story segmentation, personalisation and recommendation. American Association for Artificial Intelligence (www.aaai.org). Disponible en citeseer.nj.nec.com/557094.html

Smith, J., y Chang, S. (1996). Visualseek: A fully automated content-based image query system. In ACM International Conference on Multimedia, 87-98. 27

Smith, J., y Chang, S. (1997). Quering by colour regions using the visualseek content-based visual query system. M.T. Maybury, editor, Intelligent mul-

timedia information retrieval, AAAI Press. 27

Soille, P. (2003). *Morphological image analysis* (2nd ed.). Heidelberg: Springer-Verlag. 11, 17, 33, 36

Stentiford, F. (2003). An attention based similarity measure with application to content based information retrieval. *Storage and Retrieval for Media Databases conference, SPIE Electronic Imaging, Santa Clara*. Disponible en citeseer.nj.nec.com/stentiford03attention.html

Stuckenschmidt, H., Harmelen, F. van, Fensel, D., Klein, M., y Horrocks, I. (2000). Catalogue integration: A case study in ontology-based semantic translation. *Technical Report IR-474, Computer Science Department, Vrije Universiteit Amsterdam*. 10

Sujansky, W. (2001). Heterogeneous database integration in biomedicine. *Journal of Biomedical Informatics*, *34*, 285–298. 7

Tanimoto, S., y Pavlidis, T. (1975). A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, *4*(2), 104 - 119. 23

Veltkamp, R., y Tanese, M. (2000). Content-based image retrieval systems: A survey. Department of computing Science, Utrecht University. 27

Vincent, L., y Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE PAMI, 1991*, *13*(6), 583–598. 17

Wan, X., y Kuo, C. (February, 1996). Color distribution analysis and quantization for image retrieval. En (Vol. 2670). SPIE proceedings.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, *25*, 38–49. 9

Wolfe, R. E., Roy, D. P., y Vermote, E. (1998, July). MODIS land data storage,

gridding, and compositing methodology: Level 2 grid. *IEEE Transactions on Geoscience and Remote Sensing*, *36*, 1324-1338. 3, 28

Yagoubi, B., y Sulimani, Y. (2007). Task load balancing strategy for grid computing. *Journal of Computer Sciences*, *3*(3), 186-194. 43

Yu, W., Fritts, J., y Sun, F. (2002, Lausanne, Switzerland, Aug). A hierarchical image segmetation algorithm. En *Ieee international conference on multimedia and expo.* 16, 23

Zhou, H., Yang, X., Tang, Y., y Xiao, N. (2004). Further optimized parallel algorithm of watershed segmentation based on boundary components graph. En *Npc* (p. 498-501). 36