Desarrollo de una plataforma astroinformática para la administración y análisis inteligente de datos a gran escala

Búsqueda Semántica

Mauricio Solar, Marcelo Mendoza, Gabriel Candia,
Jorge Ibsen, Lars Nyman, Eduardo Vera, Diego Mardones, Guillermo Cabrera,
Paola Arellano, Karim Pichara, Paulina Troncoso, Ricardo Contreras,
Victor Parada.

Santiago, 11 de abril de 2014

Resumen

En este documento se informa el estado de avance del entregable "Buscador semántico", lo que corresponde a la documentación para el hito 2 de este producto. Se detallan avances en relación a operaciones de consultas y búsquedas sobre la base de datos de objetos recuperados desde imágenes, así como pruebas preliminares que permiten ilustrar propiedades de los métodos propuestos. Se valida el uso de estructuras de datos R-tree para indexar los objetos permitiendo condición de balance por altura por regiones con densidad de objetos variable, lo que garantiza tiempos de procesamiento computacional independientes de los parámetros de búsqueda de la consulta usada.

Palabras Claves: Thresholding, binarización, connected components, Ra/DEC, R-trees

1. Resumen Ejecutivo

El método propuesto indexa objetos que son registrados en imágenes FITS. En síntesis, las imágenes FITS corresponden a cortes de los cubos de datos ALMA, recuperados desde el UV con IFFT2D. Cada uno de estos cortes es procesado con un algoritmo de *adaptive thresholding* que permite obtener una versión binarizada de la imagen. La imagen binarizada es procesada con un algoritmo de detección de objetos conexos (*labels connected objects*), que permite etiquetar cada objeto de la imagen. Usando *convex hulls*, los objetos son regularizados en sus contornos. Finalmente, a estos objetos se les extraen características relevantes para el índice, entre ellas, coordenadas del centroide y excentricidad.

Las coordenadas extraídas con el proceso anterior son relativas a las dimensiones de la imagen. Se implementa un proceso que permite expresar las coordenadas relativas en el sistema *Right Ascension and Declination*, RA/DEC, las cuales son las que finalmente se consideran para el proceso de indexamiento. Luego, estas alimentan un índice R-tree que introduce la condición de balance por altura en todas las regiones indexadas del cielo. garantizando que el costo computacional de recuperación es el mismo independientemente de la región del cielo sobre la que se realiza la búsqueda. Se hacen pruebas que permiten concluir que el método es adecuado al problema ya que tiene buenas propiedades para el problema de recuperación de objetos en grandes bancos de imágenes como las generadas por ALMA.

2. Definición del Problema

ALMA realiza observaciones sobre el cielo generando datos en coordenadas UV y con una tercera dimensión de representación que corresponde a frecuencia. Debido a esto, cada observación libera datos en forma de cubos, los cuales pueden ser analizados por slices de frecuencia, cada uno de ellos representable por un FITS. Luego, un cubo puede ser considerado como un stack de FITS.

Nuestro indexador semántico trabaja a nivel de slices FITS indexando los objetos del cubo. Sin perder generalidad, los métodos estudiados y reportados en este informe operan directamente sobre las imágenes, lo cual les permite ser extendidos a otros dominios de observación que no están basados en radioastronomía, como por ejemplo, observación en el rango óptico.

Una observación puede contener registros de uno o más objetos. Es de interés para el ChiVO, el poder realizar consultas sobre un sistema de coordenadas, las cuales debieran retornar un listado de fuentes (cubos o slices) que contienen los objetos astronómicos que satisfacen las condiciones de búsqueda. Por lo tanto, se requiere de un procedimiento que permita caracterizar a los objetos de las observaciones, detectando de forma automática coordenadas u otras características descriptivas de interés. La caracterización de los objetos debe ser completamente automática, para favorecer la implementación de un algoritmo que pueda ser colocado en producción y escale a grandes bancos de imágenes.

3. Solución propuesta

3.1. Síntesis Hito 1

La solución explorada requiere de librerías que permitan trabajar con imágenes de observaciones astronómicas. Para ello, hemos considerado librerías que trabajen sobre FITS (Flexible Image Transports System), el cual es el formato más comúnmente usado para registro de imágenes astronómicas. Se mostrarán en este informe resultados obtenidos usando las librerías FITSi, biOps y EBImage, disponibles para el lenguaje R. Son pruebas de concepto que permiten validar la solución propuesta. Una vez se genere el pipe definitivo a colocar en producción, será necvesario integrar estos códigos con CASA (Common Astronomy Software Application), el cual está desarrollado en C++ pero provee interfaces para Python. Es posible integrar CASA y R sobre Python, ya que R posee características que lo hacen compatibles con códigos py. Finalmente, es posible ejecutar los códigos R en forma stand alone, pudiendo ser posible en la etapa de puesta en producción que los procesos de CASA y R operen en secuencia, sin acoplamiento.

Se ilustra la solución propuesta sobre una imagen recuperada desde el das [1].

La imagen ilustra lo desarrollado hasta el hito 1 del buscador semántico. Cada FITS es procesado en relación con la intensidad de sus pixels, sobre lo cual se aplica un algoritmo de adaptive thresholding,el cual binariza la imagen separando el background del foreground. Un algoritmo de connected componenets etiqueta las regiones conexas de la imagen, asignándoles un ID, que es finalmente usado como ID de cada objeto. Un algoritmo de convex hull se aplica sobre la imagen binarizada para proceder a la extracción de

[1] http://das.sdss.org/

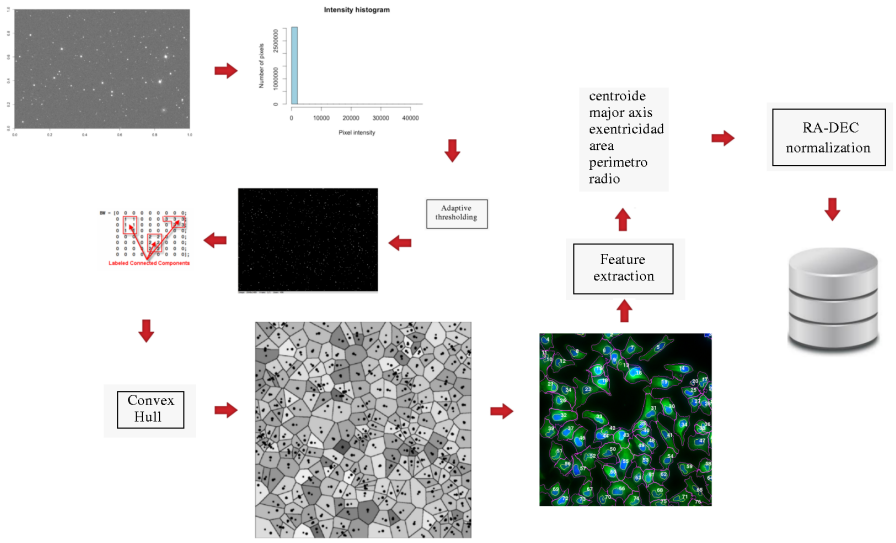

Figura 1: Proceso de extracción de información desde FITS para crear registros de objetos en la base de datos

características, recuperando centroide, eje mayor y otras propiedades de interés. Un proceso de normalización tranforma las coordenadas relativas a la imagen en coordenadas absolutas RA-DEC. Luego, las características del objeto y su ID son registrados en una base de datos.

3.2. Base de Datos

La elección de base de datos tiene un papel preponderante en el proyecto, dado que toda la información obtenida al procesar las imágenes debe ser almacenada en una base de datos para su posterior uso. De esta misma forma esta base de datos debe ser capaz de implementar índices espaciales los cuales son requeridos para realizar las consultas de una manera más rápida y eficiente.

Por lo general la mayoría de las base de datos que se usan actualmente permiten implementar un índice espacial, pero claramente la forma de implementación y funcionamiento son muy distintas.

La información a guardar en la base de datos en primera instancia es la obtenida mediante el procesamiento de las imágenes, es decir las coordenadas astronómicas de cada objeto, así como también el nombre del archivo FITS desde donde fueron obtenidos como también alguna otra característica si fuera necesaria.

El tipo de base de datos a utilizar será una base de datos relacional. En la actualidad es uno de los modelos más utilizados en base de datos dado que es utilizado para modelar problemas reales y administrar datos de forma dinámica. Este modelo se basa en el uso de relaciones, estas relaciones se conocen también como conjuntos de datos denominados *tuplas*. Cada relación se considera una tabla donde las filas representan los registros de la tabla y las columnas son los campos de la tabla. A diferencia de otros modelos jerárquicos de base de datos la forma y el lugar donde se almacenan los datos no tiene mayor importancia, dado que la información es recuperada mediante consultas que ofrecen gran flexibilidad para administrar la información.

El lenguaje más habitual de consultas es el SQL[1] que a su vez es muy utilizado en la astronomía. En astronomía también utilizan bases de datos relacionales por el hecho de mantener grandes cantidades de tablas y relaciones, donde el correcto uso del SQL es vital para lograr consultas eficientes y no sobrecargar más de lo necesario las bases de datos.

Por lo general en astronomía los grandes archivos de datos permiten el uso de SQL, claro que las consultas están limitadas y poseen ciertas reglas para asegurar su buen funcionamiento, un ejemplo son la cantidad de filas que se pueden extraer. Un ejemplo de un sitio que permite realizar consultas en SQL para obtener información del observatorio *Sloan Digital Sky Survey* es Casjobs.

3.2.1. Elección de Base de Datos

SQLite SQLite es un motor de base de datos que soporta consultas SQL. Es de uso público para cualquier clase de proyectos ya sean públicos o privados. A diferencia de otras base de datos que necesitan de un servidor, SQLite permite leer y escribir directamente en un archivo del disco duro, lo que facilita su instalación y utilización para casos donde se necesita algo rápido y sencillo, mediante solo un archivo en el disco duro se pueden almacenar, las tablas, índices, triggers, etc. SQLite funciona tanto en 32 bits como en 64 bits, con solo copiar el archivo de una plataforma a otra sin tener que hacer algún cambio. SQLite es una buena solución cuando se busca software abierto y algo simple, posee una comunidad bien amplia de desarrolladores que trabajan en nuevas funciones y en solucionar los bugs.

SQLite posee el índice espacial R-Tree. La forma en como funciona es primero teniendo una tabla normal que posea datos de los objetos además de poseer un cuadro que pertenece a una región, es decir posee 2 coordenadas máximas y 2 coordenadas mínimas que generan una región en el espacio. Luego sobre esa tabla es necesario generar un índice que puede ser de 2 e incluso 3 dimensiones. Para realizarlo es necesario generar una tabla virtual que contendrá solo los datos de ubicación, es decir los valores a agregar por cada objeto al índice son el id (el cuál sirve para ligar los demás datos de características pertenecientes en otra tabla) y 4 valores que corresponden a las coordenadas mínimas y máximas. Para utilizar las capacidades del índice es necesario realizar el query mediante la tabla virtual y luego hacer un join sobre la otra tabla para recuperar los detalles del objeto.

Un punto en contra que posee SQLite es que para el índice R-Tree necesita que se archiven regiones y no puntos, cosa que puede ser un problema para archivar los objetos dado que se tienen las coordenadas del centroide de cada objeto (es decir un punto, no una región). Otra cosa que llama la atención es que las consultas no varían mucho con respecto a una consulta por rangos de un valor y rangos de otro valor que se puede hacer normalmente sobre una tabla, ahora esto no quiere decir que el índice no funcione sino que traduce esa consulta para poder utilizar de forma correcta el índice.

MySQL MySQL es uno de los motores de base de datos más usados actualmente, a pesár de que es un software de tipo open source, de igual forma es necesario pagar muchas de las funcionalidades que éste motor posee, sobre todo las más complejas. A diferencia de SQLite, MySQL es una organización mucho más grande y con gente más dedicada al tema.

[1] Structured Query Language.

Al investigar acerca de los índices espaciales que ofrece MySQL se deja ver de inmediato que posee mayor cantidad de herramientas para generar consultas espaciales que SQLite. Las consultas a diferencia de SQLite pueden ser por regiones geométricas, que pueden ser cuadrados por ejemplo o polígonos más complicados. A diferencia de SQLite donde los campos utilizados para las búsquedas espaciales son números aquí pueden corresponder a guardar figuras geométricas que no necesariamente son cuadrados (como en SQLite) y además permiten guardar puntos, que es lo que se busca para el índice de objetos, dado que se indexarán mediante su centroide.

PostgreSQL PostgreSQL es otro motor de base de datos basado en el modelo relacional. Es público para el que quiera usarlo y provee buenas herramientas para instalar e iniciar sus servicios. La instalación de PostgreSQL viene con un idea llamado pgAdmin que permite crear bases de datos, realizar consultas sobre las tablas presentes entre otras cosas.

PostgreSQL debe ser un punto intermedio entre SQLite y MySQL donde SQLite es muy simple por lo que algunas funcionalidades no las tiene y MySQL siendo más complejo y completo. Posee una gran cantidad de herramientas para interactuar con las bases de datos que pueden ser utilizadas en casi todos los sistemas operativos.

En términos de búsquedas espaciales la configuración de PostgreSQL se vuelve un poco más tediosa, dado que hay que instalar un paquete extra denominado PostGIS que a su vez requiere de otras librerías. Dentro de esta librería están los índices espaciales y los objetos geométricos parecidos a los que utiliza MySQL, donde las consultas por región son del estilo este punto está dentro de esta región que puede ser un cuadrado o polígonos más complejos acorde a las necesidades.

El paquete de PostGIS está separado de la configuración básica de PostgreSQL dado que posee una gran cantidad de herramientas que no son tan básicas para utilizar, que a su vez permiten gran flexibilidad para realizar y responder consultas de manera rápida y eficiente. PostGIS dice ser el complemento más completo en búsquedas espaciales y tiene recursos dedicados al buen funcionamiento del paquete lo que lo hace una herramienta muy poderosa con nuevas actualizaciones.

En la Tabla 1 se muestra una comparación de los tres posibles motores de bases de datos a utilizar, tomando en cuenta los puntos expuestos anteriormente. Cabe destacar que en funcionalidades, se analizan las que son necesarias para llevar a cabo el índice de objetos astronómicos, así como también en complementos pagados.

Al momento de elegir entre una y otra base de datos es necesario saber las características que esta va a cumplir, el modelo de datos que ocupará la cantidad de tablas y relaciones que pueda tener para lograr un funcionamiento óptimo. Para el proyecto la decisión esta entre SQLite y PostgreSQL dado que MySQL requiere ciertas licencias para utilizar las búsquedas espaciales. Por otra parte SQLite al parecer es demasiado básico en búsquedas espaciales por lo que no se podría sacar demasiado provecho de éstas, la elección más aceptada parece ser PostgreSQL que no requiere licencias para la utilización de PostGIS y permite realizar búsquedas más complejas.

Competencia	SQLite	PostgreSQL	MySQL
Instalación	Sencilla	Sencilla	Normal
Configuración avanzada	Limitada	Extensa	Extensa
Índice espacial	Si soporta	Si soporta	Si soporta
Tipo de índice	R	GiST	R
Funcionalidades	No las cumple	Si las cumple	Si las cumple
Complementos extra	No necesita	PostGIS	Librerías de geometría
Escalabilidad	Limitada	Si	Si
Flexibilidad	Limitada	Gran flexibilidad	Flexible
Complementos pagados	No	No	Si

Cuadro 1: Comparación de los puntos importantes de las bases de datos.

3.2.2. Base de Datos Espacial

Las bases de datos espaciales pueden estar basadas solo en datos de ubicación espacial o como en el caso de este proyecto poseer algunos campos especiales en una tabla que permitan fijar la ubicación del objeto en un espacio previamente definido. Para el caso de astronomía y como se mencionó anteriormente se usan las coordenadas de tipo RA-DEC que tienen como unidad los grados, se pueden asimilar mucho a las coordenadas geográficas que utiliza cualquier sistema de posicionamiento en la tierra por lo que no es necesario procesarlas de alguna forma para ubicarlas en el espacio.

En este estilo de bases de datos es necesario definir un cuadro de referencia que permite definir la localización y relación entre los objetos, ya que los datos tratados en este tipo de bases de datos tienen un valor relativo al cuadro referencia. Los sistemas de referencia espacial pueden ser georreferenciados como no georreferenciados, los georreferenciados se establecen sobre la superficie terrestre.

La estructuración de la información espacial procedente del mundo real en capas conlleva cierto nivel de dificultad. En primer lugar, la necesidad de abstracción que requieren los computadores implica trabajar con primitivas básicas de dibujo, de tal forma que toda la complejidad de la realidad ha de ser reducida a puntos, líneas o polígonos. En segundo lugar, existen relaciones espaciales entre los objetos geográficos que el sistema no puede obviar; la topología, que en realidad es el método matemático-lógico usado para definir las relaciones espaciales entre los objetos geográficos puede llegar a ser muy compleja, ya que son muchos los elementos que interactúan sobre cada aspecto de la realidad.

Un dato espacial es una ubicación en el espacio que posee una forma particular, estos datos por lo general son de forma vectorial los cuales se construyen y pueden ser expresados por tres tipos de objetos espaciales.

- **Puntos** Definidos por un par de coordenadas, que en este caso son RA-DEC que denotan un hito en el espacio.
- **Líneas** Objetos abiertos que cubren una distancia dada, se usan generalmente para calcular rutas entre un punto y otro, mediante varias líneas es posible trazar un polígono también por lo que aunque lo de rutas no es necesario para el índice, trazar líneas quizás si.

- **Polígonos** Figuras cerradas que son la unión de varios objetos abiertas por ejemplo líneas, cubren un área determinada.

De esta forma la información contenida en puntos, línea o polígonos se almacena como un conjunto de pares coordenados. La ubicación puntual de un objeto se define como una coordenada (RA,DEC), las características lineales se almacenan como un conjunto de coordenadas y los polígonos también se almacenan como un conjunto de coordenadas solo que son cerrados.

3.3. Índice de Objetos

3.3.1. Acceso espacial

Para evitar recorrer los datos de forma secuencial lo que genera una carga innecesaria, se crean índices que reducen la cantidad de elementos a visitar para responder una consulta. En este caso particular el índice tiene que ser de tipo espacial es decir tiene que estar basado en algún dato espacial presente en la tabla a indexar.

La indexación clásica de datos mediante B-Tree no sirve para índices espaciales dado que los valores no tienen un orden único. Por este motivo es necesario la utilización de algun método de acceso espacial. Existen tres categorías de métodos de acceso espacial, las PAM (Point Access Method), los árboles R-Tree y las SAM (Spatial Access Method), el punto de acceso a utilizar depende del tipo de dato espacial por el cual se está consultando.

Aunque hay un montón de benchmarks frente a los diferentes tipos de acceso espacial, ninguno es muy concluyente pero lo que si se recomienda al menos utilizar uno, dado que al realizar consultas sin índice espacial tomará mucho más tiempo del necesario.

B-Tree Los árboles B son estructuras de datos que se encuentran comúnmente en las implementaciones de bases de datos y de sistemas de archivos. Son árboles balanceados donde cada nodo puede poseer más de dos hijos.

La búsqueda se realiza de forma similar a los árboles binarios donde se comienza buscando por la raíz y se va recorriendo el árbol hacia abajo dependiendo del valor del subnodo. El árbol B se utiliza para indexar un valor de datos de la tabla, por lo que para las búsquedas espaciales se descarta.

kd-Tree Un árbol kd se refiere a un árbol de k dimensiones, es una estructura de datos que organiza los puntos en un espacio Euclídeano. Un árbol kd emplea solo planos perpendiculares a uno de los ejes del sistema de coordenadas. Todos los nodos del árbol almacenan un punto, desde el nodo de la raíz hasta las hojas.

La construcción del árbol se hace de forma de que todos los ejes de coordenadas vayan iterando, por ejemplo en un árbol de dos dimensiones el nodo de la raíz representa al eje x, los nodos hijo de la raíz representan al eje y, los nietos de la raíz vuelven a representar el eje x y así sucesivamente. En caso de tener más dimensiones funciona de la misma forma.

En la Figura 2 se muestra un ejemplo de kd tree en dos dimensiones. Se puede observar que todos los nodos guardan información sobre un punto y no es necesario llegar a la raíz para encontrar todos los puntos, esto puede acortar la consulta sobre el árbol pero también puede extenderla más de la cuenta si el árbol queda demasiado desbalanceado. También se puede observar que el kd-tree abarca toda el área a pesar de que hay partes donde no hay puntos u objetos de interés. Se puede observar también que los puntos que están sobre una línea vertical corresponden a una partición en el eje x y eso se puede ver reflajado gráficamente en la imagen y también en el árbol.

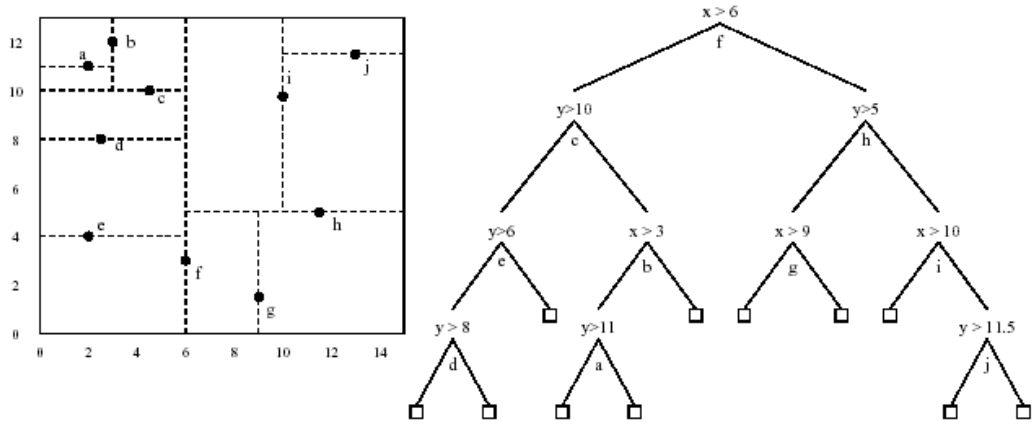

Figura 2: Ejemplo de kd-Tree en dos dimensiones.

R-Tree Los árboles R son estructuras de datos similares a los árboles B, pero a diferencia de los árboles B que indexan una sola variable, los árboles R permiten almacenar variables multidimensionales por lo que sirven para las búsquedas espaciales. La estructura de datos divide el espacio en forma jerárquica en conjuntos que podrían estar sobrepuestos.

Cada nodo interno del árbol R almacena dos datos, uno es la identificación del nodo hijo y además guarda los límites de coordenadas que puede tener el nodo, con el fin de que todos los hijos estén dentro de esos límites para ser guardados en ese nodo. La cantidad de hijos que puede contener cada nodo se define al generar el árbol. Los algoritmos para insertar y borrar datos de los árboles se fijan en los conjuntos límites de los nodos para asegurar que todos los elementos cercanos pertenezcan a la misma hoja. De esta misma forma al momento de insertar un nuevo punto o región al árbol se verifica cuál nodo sufriría el menor aumento en su conjunto límite para poder insertarlo. De forma similar a la inserción, los métodos de búsqueda sobre árboles R se fijan en los conjuntos límites de los nodos, por lo que requieren recorrer una menor parte para encontrar la región buscada, por lo que facilitan mucho el uso de base de datos con búsquedas espaciales.

Los árboles R no garantizan un buen rendimiento en los peores casos, pero si logran facilitar las búsquedas comúnmente hechas como encontrar todas las farmacias en una región definida por cuatro pares de coordenadas [19].

En la Figura 3 se muestra un ejemplo de un R-tree donde la parte superior muestra las regiones que corresponden a los conjuntos límites de cada nodo y en la parte inferior se muestra como un árbol. Se puede observar de manera gráfica que la región número 3 posee dentro las regiones 8,9 y 10, y en la parte inferior se puede observar que corresponde que estas 3 regiones son nodos hijo de la región número 3.

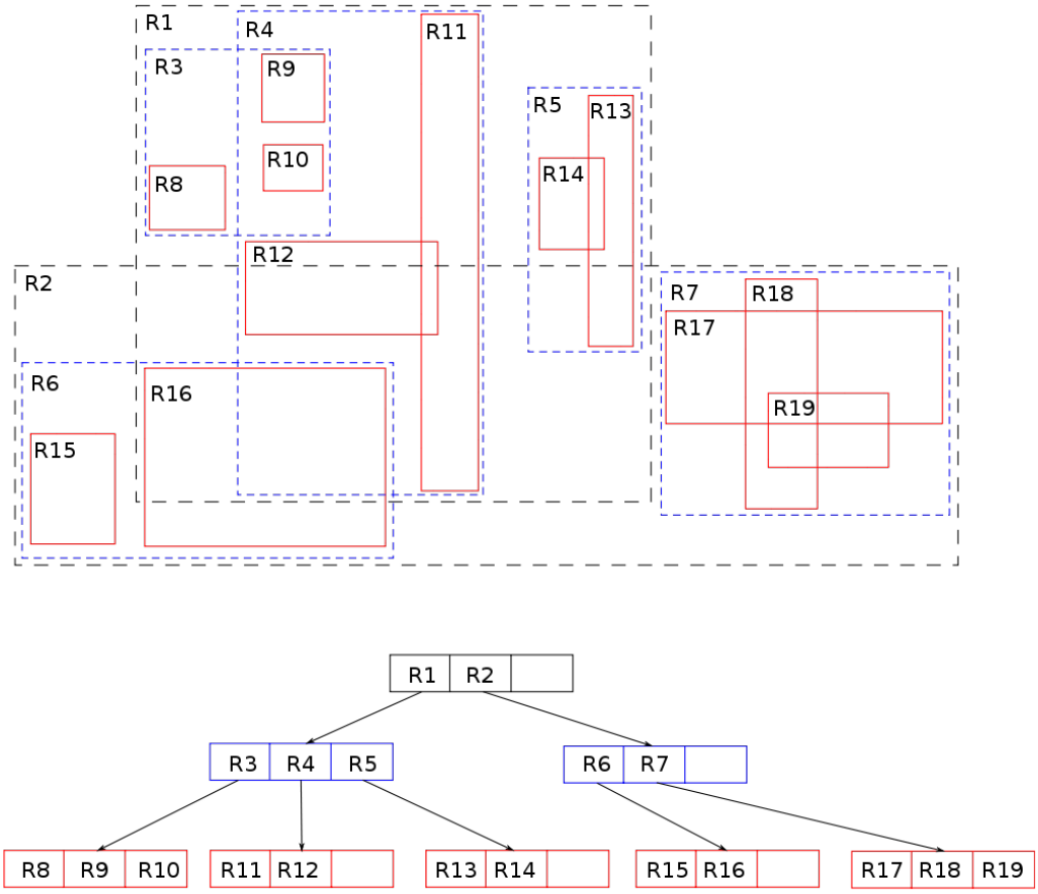

Figura 3: Ejemplo de R-Tree en dos dimensiones.

Como se puede observar en el ejemplo esta clase de índice se usa cotidianamente cuando por ejemplo se busca algún restaurante cercano a la posición actual de una persona con alguna aplicación que tenga geolocalización. De forma similar permite establecer regiones de interés en el espacio, aunque para la indexación de los objetos astronómicos se archivará solo un punto en el índice, el cuál corresponde a la posición del centroide del objeto.

Diferencias entre kd-tree y R-tree

- Los árboles R son balanceados, lo que hace que todas las consultas se demoren el mismo tiempo, por otra parte los árboles kd no lo son lo que requiere aplicar técnicas extras de optimización para utilizarlos.
- Los árboles R están orientados para su uso en disco, por lo que organizan la información de tal

forma que al querer guardarlas en una base de datos por ejemplo se torna fácil. Por otra parte los árboles kd están orientados a trabajar en memoria, por lo que escribir en disco puede ser tedioso.

- Los árboles R no cubren todo el espacio si es que no hay objetos presentes en alguna área remota de este, por otra parte los árboles kd siempre cubren todo el espacio.
- Los árboles R particionan los datos en rectángulos, mientras que los árboles kd hacen una partición binaria. Por esto mismo los réctangulos del árbol R pueden sobreponerse, lo que no es necesariamente un problema, pero se trata de minimizar.
- Los árboles kd son fáciles de implementar en memoria, el cual es su enfoque. A diferencia de los árboles R.
- Los árboles R pueden guardar rectángulos y polígonos, mientras que los árboles kd solo guardan puntos dado que no soportan la sobreposición.
- Los árboles R vienen con varias estrategias de optimización como diferentes estrategias a la hora de insertar, reinsertar o borrar algún punto.

Es por estos motivos recién descritos que la elección entre árbol R y árbol kd se inclina hacia el árbol R. Ambos podrían funcionar perfectamente pero el que se desempeña de manera óptima en las circunstancias del proyecto es el árbol R, esto por que requiere escribir continuamente en base de datos, requiere optimizar las consultas y el hecho de ser desbalanceado puede perjudicar mucho la implementación del índice, requiere poder guardar formas más complejas si en un futuro los objetos se caracterizan de forma más completa, entre otras.

GiST GiST viene de la sigla Generalized Search Tree y es una estructura de datos que permite crear diferentes tipos de árboles. GiST es una generalización del árbol B que provee una herramienta de búsqueda mediante un árbol balanceado, sin realizar asunciones del tipo de dato a guardar o del tipo de consulta que se está realizando. GiST puede ser utilizado para implementar los índices más comúnmente usados como lo son el árbol B, el árbol R, el árbol kd, entre otros. De la misma forma que permite la utilización de distintos tipos de índices y tipos de datos por el cual indexar, también permite realizar índices especiales sobre nuevos tipos de datos, tiene como restricción que los árboles a implementar deben ser balanceados en cuanto a la altura, por lo que no sirve para generar índices como quad tree [1]. GiST puede ser usado con cualquier tipo de data siempre y cuando pueda ser ordenada de forma jerárquica [20].

GiST es una estructura de datos extensible en términos de tipos de dato y tipos de árboles, también permite un manejo sobre las consultas a realizar donde estas también pueden realizarse especialmente para un caso particular. Como se muestra es una herramienta que extiende de manera óptima el manejo de base de datos, permite la correcta evolución de las bases de datos para soportar nuevos tipos de árboles.

El mayor uso de GiST en la actualidad está en las bases de datos relacionales PostgreSQL, esta implementación incluye soporte para largos variables de los datos, datos compuestos, concurrencia y

[1] Árbol basado en la descomposición recursiva.

recuperación, éstas están presente en todas las extensiones de GiST. Además de esto hay una gran cantidad de módulos siendo desarrollados con GiST y distribuídos mediante PostgreSQL.

3.3.2. Consultas espaciales

En relación al documento de requerimientos del ChiVO, se requieren implementar tres tipos de búsqueda sobre la bases de datos:

- **Búsqueda por región:** La búsqueda por región queda definida por cuatro valores, dos valores inferior y superior de coordenada RA y dos valores inferior y superior de coordenada DEC. Con esto se forma una región cuadrada donde se buscarán todos los objetos presentes en esta región. Tiene como obligación los cuatro datos de entrada.
- **Búsqueda circular:** La búsqueda circular queda definida mediante un par de coordenadas RA-DEC y un radio medido en grados al igual que las coordenadas. Con esto se forma una región circular donde se buscan todos los objetos dentro de ésta. Cabe destacar que para este tipo de búsqueda no existe un objeto círculo listo para consultar, sino que es necesario generar una región mediante un cálculo de distancias hacia el centro, esto quiere decir que puede ser más costoso en términos de ejecución que la búsqueda por región.
- **Búsqueda de vecinos cercanos:** La búsqueda de los vecinos cercanos lo que busca es darle como datos de entrada un par de coordenadas que permitan generar un punto en el espacio y sobre ese punto trazar un círculo de un largo definido con un radio no muy largo con el fin de encontrar los objetos pertenecientes a el que se podrían catalogar como vecinos por cumplir una distancia acotada. La búsqueda es parecida a la búsqueda circular solo que el radio queda definido previamente.

Las tres consultas expuestas anteriormente pueden ser realizadas usando PostgreSQL y procesadas de forma eficiente usando un R-tree de GIST.

3.4. Índice de objetos

El índice de objetos en PostgreSQL debe ser sobre una columna de tipo geométrica, esto quiere decir que esa columna puede ser un punto, como también puede ser un rectángulo o un polígono más complejo. Para el caso particular del índice de objetos astronómicos se guardó una variable centroid que es de tipo geométrica que corresponde al punto del centroide de cada objeto. Para la vista el valor que posee esta columna no es fácil de entender dado que esta expresado en muchos números, es por esto que igual se guardan las posiciones RA y DEC del centroide en otras columnas.

En la Figura 4 se muestra la tabla utilzada para almacenar los objetos astronómicos. Por el hecho de darle énfasis al índice no se guardaron otras características de cada objeto pero agregar estas caracetrísticas a la tabla es trivial y depende de qué clase de datos se quieren extraer de estos. Las columnas que tiene la tabla son id, centroid que es de tipo geométrico, ra, dec y fitsimage. Fitsimage corresponde al nombre del FITS de donde se extrajo el objeto, se guarda este dato con el fin de que si quiere indagar más en el objeto pueda ir a la fuente a obtener mayor información.

id integer	centroid geometry	ra real	dec real	fitsimage character varying(80)
1	0101000020E6100000760798016981684 0D03296B05868EF3F	196.044	0.981488	drC-000752-g6-0350.fits
2	0101000020E61000005772E3076981684 0253422523C51ED3F	196.044	0.916166	drC-000752-g6-0350.fits
3	0101000020E6100000A346015B6481684 028EC816EEDC4ED3F	196.044	0.930289	drC-000752-g6-0350.fits
4	0101000020E61000003D9CB2976381684 04A570C53CBDFEB3F	196.043	0.871069	drC-000752-g6-0350.fits
5	0101000020E6100000E4948488628168407C3789A663C6EE3F	196.043	0.961717	drC-000752-g6-0350.fits
6	0101000020E61000009BA02AEC5E816840F83123628 3B4EE3F	196.043	0.959535	drC-000752-g6-0350.fits
7	0101000020E6100000BDA3BFE0568168405FBF44097D21ED3F	196.042	0.910338	drC-000752-g6-0350.fits
8	0101000020E61000001916AD7C4281684088B00535406BF03F	196.039	1.02618	drC-000752-g6-0350.fits
9	0101000020E61000001C2C69D940816840A7C203EEDA26EC3F	196.039	0.879743	drC-000752-g6-0350.fits
10	0101000020E61000005A2E01A54081684078E650168497EA3F	196.039	0.830996	drC-000752-g6-0350.fits
11	0101000020E6100000265047B34816840866986197FE6ED3F	196.038	0.934387	drC-000752-g6-0350.fits

Figura 4: Tabla de objetos.

Consultas por regiones rectangulares Para realizar una consulta mediante una región rectangular y aprovechar el índice generado sobre la columna geométrica centroid es necesario no utilizar la claúsula ¿= o ¡= sobre el valor RA o DEC sino que se evalúa si un objeto pertenece al rectángulo en cuestión mediante el operador &&. Para realizar una consulta rectangular es necesario generar un rectángulo mediante dos puntos, el punto inferior izquierdo (que corresponde al RA y DEC menor) y el punto superior derecho (que corresponde al RA y DEC mayor), para esto se utiliza el método ST_MakeBox2D que recibe como parámetro 2 puntos y genera un rectángulo.

Consultas por regiones circulares Para realizar una consulta circular se utiliza la función ST_Buffer(geom,distancia) donde el geom corresponde a un punto geométrico y la distancia vendría siendo el radio. Esta función no retorna un círculo exacto, el círculo es generado mediante un polígono aproximado de 8 lados, la precisión de este círculo se puede aumentar agregando un tercer parámetro a la función que corresponde a la cantidad de lados del polígono. En la Figura 5 se muestra como se realiza el círculo mediante un polígono [22].

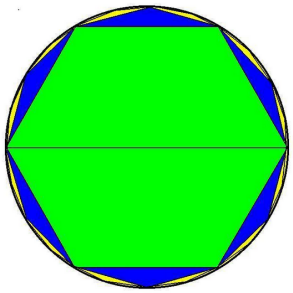

Figura 5: Generación de una región circular.

La consulta circular requiere como parámetros el punto central mediante coordenadas RA-DEC y el radio del círculo. Para la consulta de los vecinos cercanos se utiliza algo similar solo que el radio está previamente definido y tiene un valor más pequeño.

3.5. Tiempo de Procesamiento de Consultas

Para probar el funcionamiento del índice y de los algoritmos de procesamiento, se utilizaron imágenes obtenidas desde observatorios ópticos. Para validar este comportamiento es necesario obtener una mayor cantidad de imágenes para poner a prueba los algoritmos, así como también la base de datos bajo una carga mayor, con el fin de observar el desempeño del índice de objetos.

Para obtener grandes cantidades de imágenes, es necesario realizar un *mass download* desde algún observatorio. El observatorio escogido fue el SLOAN, dado que sus imágenes son *surveys* del cielo donde cada imagen posee grandes cantidades de objetos. El SLOAN tiene una herramienta que permite buscar archivos FITS dada unas ciertas coordenadas RA-DEC. Dado que es un trabajo tedioso ingresar una a una todas las coordenadas de los objetos que se quieren descargar, se recurrió a generar pares de coordenadas de forma aleatoria, lo que se traduce en 800 puntos distintos en la búsqueda. Se obtuvo un total de 835 archivos FITS que pesan 4.8 Gb, los cuales fueron descargados en aproximadamente 5 horas. Cada archivo pesa alrededor de 6 MB. La ejecución del algoritmo de extracción de características tardó 5 horas y dió como resultado 314.000 objetos encontrados y archivados en la tabla objetos de la base de datos.

Para poder validar el desempeño del índice, se realizaron los tres tipos de consulta previamente expuestos, obteniendo distintas magnitudes de objetos en cada una, lo que permite verificar los distintos tiempos de respuesta obtenidos. En la Tabla 2 se muestra el detalle de las 12 consultas realizadas. En la columna consulta se muestra el tipo de consulta, en la columna parámetros se muestran los parámetros utilizados, que son distintos para cada tipo de consulta. En la columna número de objetos se muestran los objetos obtenidos al realizar la consulta, en la columna tiempo se expresa el tiempo que tomó realizar la consulta medida en milisegundos y por último en la última columna se muestran las columnas que fueron pedidas mediante la consulta. Como se puede observar los mayores tiempos ocurrieron al consultar por todas las columnas y abarcando todos los objetos en el índice, éste tiempo tiene una baja considerable al preguntar por sólo una columna, lo cuál se podría aprovechar consultando por solo las columnas necesarias.

Consulta	Parámetros	N° Objetos	Tiempo	Columnas
Region Query	RA:(-30,36), DEC:(-30,360)	314459	15257 ms	Todas
Region Query	RA:(-30,36), DEC:(-30,360)	314459	2317 ms	id
Region Query	RA:(-30,36), DEC:(-30,360)	314459	3953 ms	id,ra
Region Query	RA:(100,150), DEC:(50,100)	11077	646 ms	Todas
Region Query	RA:(0,200), DEC:(30,50)	25178	1417 ms	Todas
Radial Query	Punto:(180,180), Radio:210	314459	16367 ms	Todas
Radial Query	Punto:(50,100), Radio:40	19884	1023 ms	Todas
Radial Query	Punto:(50,100), Radio:40	19884	458 ms	id,ra,dec
Radial Query	Punto:(200,50), Radio:10	1620	178 ms	Todas
Neighbour Query	Punto:(190,52)	1064	146 ms	Todas
Neighbour Query	Punto:(78,62)	3209	247 ms	Todas
Neighbour Query	Punto:(161,64)	961	137 ms	Todas

Cuadro 2: Tiempos de procesamiento de consultas.

4. Trabajo futuro

Las pruebas realizadas sobre FITS de SLOAN ilustran que la propuesta usada en este proyecto es adecuada. Sin embargo, se requieren pruebas sobre datos tipo ALMA para verificar la correctitud de la propuesta bajo las condiciones de operación que ALMA tiene. Para esto, se planifica para el siguiente hito probar y extender estos logros en los siguientes puntos:

- Pruebas con datos tipo ALMA: Se requieren slices de cubos tipo ALMA para probar nuestros algoritmos. Para ello, se trabajo en la generación de datos sintéticos y también se cuenta con algunos cubos que permiten pruebas preliminares.
- Robustez del algoritmo adaptive threholding: No se han realizado pruebas de robustez en relación con SNR. Los datos SLOAN pueden tener propiedades en SNR que favorezcan nuestra propuesta y por tanto, la evaluación y eventual extensión de nuestros algoritmos para trabajar en condiciones diversas de SNR es un punto importante de abordar.
- Pruebas con objetos de geometrías sofisticadas: La mayor parte de las imágenes de SLOAN corresponden a surveys sobre fuentes puntuales. Esto favorece al algoritmo labeled connected componentes y podría signficar un deterioro en precisión ante objetos de geometrías más complejas como galaxias. Se abordará la integración de nuestros algoritmos con métodos que favorezcan el reconocimiento de objetos distintos a fuentes puntuales. En este sentido, la tesis de Rodrigo Gregorio (USM), que se encuentra en desarrollo, ofrece una línea de alternativas basadas principalmente en GaussClumps que puede permitir abordar este problema de forma exitosa.

Referencias

[1] R project. ¡http://www.r-project.org/¿. [en línea] [consulta: 30-12-2012].

[2] Casa: Common astronomy software application. ¡http://casa.nrao.edu/¿. [en línea] [consulta: 11-05-2013].

[3] Virtual observatory. ¡http://www.virtualobservatory.org/faq.aspx¿. [en línea] [consulta: 05-11-2012].

[4] Fits format. ¡http://fits.gsfc.nasa.gov/¿. [en línea] [consulta: 05-03-2012].

[5] Sloan digital sky survey. ¡http://www.sdss.org/¿. [en línea] [consulta: 25-03-2012].

[6] Celestial equatorial coordinate system. ¡http://astro.unl.edu/naap/motion1/cec_units.html¿. [en línea] [consulta: 05-04-2013].

[7] Oleg Skylar. Adaptive thresholding, ebimage r package, 2007.

[8] Gaussian smoothing. ¡http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm¿. [en línea] [consulta: 10-05-2013].

[9] Gregoire Pau. Binary segmentation, ebimage r package, 2009.

[10] G. Pau and O. Skylar. Fill holes in objects, ebimage r package, 2009.

[11] K. Haralick, R. Shanmugan and Deinstein H. Textural features for image classification, ieee transactions on systems, man and cybernetics, 1973. 3(6):610-621.

[12] Casjobs. ¡http://casjobs.sdss.org/dr7/en/¿. [en línea] [consulta: 20-06-2013].

[13] Sqlite. ¡http://sqlite.org/¿. [en línea] [consulta: 25-06-2013].

[14] Sqlite r-tree módulo. ¡http://sqlite.org/rtree.html¿. [en línea] [consulta: 25-06-2013].

[15] Postgresql. ¡http://www.postgresql.org/¿. [en línea] [consulta: 15-07-2013].

[16] Postgis. ¡http://postgis.net/¿. [en línea] [consulta: 15-07-2013].

[17] Ralf Hartmut G. Spatial database systems tutorial. Oct. 1994. Special issue on spatial database systems of the VLDB Journal (Vol. 3, No4).

[18] B-trees definition. ¡http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch13.html¿. [en línea] [consulta: 25-07-2013].

[19] Antonin Guttman. R-trees: A dynamic index estructure for spatial searching. Jun. 1984. University of California, ACM New York, p 47-57.

[20] Gist indexes in postgresql. ¡http://www.postgresql.org/docs/8.1/static/gist.html¿. [en línea] [consulta: 01-08-2013].

[21] Andrew Harris. Fitsio: Fits (flexible image transport system) utilities, 2013.

[22] Circle in postgis. ¡http://theworldofapenguin.blogspot.com/2008/06/circles-in-postgis.html¿. [en línea] [consulta: 20-11-2013].