Trabajo Fin de Máster

Máster Universitario en Ciencia de Datos e Ingeniería de Computadores

PREPROCESAMIENTO DE IMÁGENES EN CIENCIA DE DATOS PARA LA BÚSQUEDA DE NUEVOS OBJETOS EN EL SISTEMA SOLAR

Autor:	Manuel Montero Santana
ETSIIT Director:	Salvador García López
IAA Director:	René Duffard

Escuela Técnica Superior de Ingenierías Informáticas y de Telecomunicación

Agradecimientos

A Rafael Morales por sus ideas e incansable ayuda durante todo el proceso y desarrollo del trabajo.

A mis tutores, René Duffard y Salvador García López, por su ayuda y consejos tanto en el trabajo como en la elaboración de este documento.

Resumen

En este Trabajo Fin de Máster se realizará un algoritmo de preprocesamiento y soporte para la búsqueda de nuevos objetos transneptunianos (TNOs) en el Sistema Solar. Para ello se crearán imágenes sintéticas usando los datos de la sonda espacial GAIA, la cual nos proporciona la posición y brillo de las fuentes dentro de la región de cielo objeto de estudio. Para la creación de la imagen sintética se usará un algoritmo genético que aproxime gaussianas artificiales en las posiciones proporcionadas por GAIA, para finalmente realizar un algoritmo para la obtención de una imagen limpia de objetos conocidos donde realizar la detección de candidatos. Por último, se comentará un análisis preliminar del diseño para Spark, con el que posteriormente se pueda procesar todas las imágenes disponibles de manera más eficiente.

Palabras Clave: Preprocesamiento, algoritmos genéticos, imágenes astronómicas, transneptuniano.

Yo, **Manuel Montero Santana**, alumno del **Máster Universitario en Ciencia de Datos e Ingeniería de Computadores** de la **Escuela Técnica Superior de Ingenierías Informáticas y de Telecomunicación** de la **Universidad de Granada**, con DNI 45779546-R, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la Biblioteca del Centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Manuel Montero Santana

Granada a de Septiembre del 2017

D. **Salvador García López**, Profesor del Área de Informática del departamento de **Ciencias de la Computación e Inteligencia Artificial** de la Universidad de Granada, y D. **René Duffard**, Investigador del departamento de **Sistema Solar** del Instituto de Astrofísica de Andalucía,

Informan:

Que el presente trabajo, titulado **Preprocesamiento de imágenes en ciencia de datos para la búsqueda de nuevos objetos en el Sistema Solar**, ha sido realizado bajo su supervisión por **Manuel Montero Santana**, y autorizo la defensa de dicho trabajo ante la Comisión de Evaluación que corresponda.

Y para que conste, expide y firma el presente documento en Granada a de Septiembre de 2017.

ETSIIT Director: **IAA Director:**

Salvador García López **René Duffard**

Yo, **Manuel Montero Santana**, alumno del **Máster Universitario en Ciencia de Datos e Ingeniería de Computadores** de la **Escuela Técnica Superior de Ingenierías Informáticas y de Telecomunicación** de la **Universidad de Granada**, con DNI 45779546-R, declaro explícitamente que el trabajo presentado es original, entendido en el sentido de que no he utilizado ninguna fuente sin citarla debidamente.

Fdo: Manuel Montero Santana

Granada a de Septiembre del 2017

Índice general

Página

Índice de figuras

Figura **Página**

Capítulo 1

Introducción

Contenidos

1.1. Motivación

Este proyecto se centra en una interesante aplicación de la Ciencia de Datos como es la resolución de problemas de búsqueda y optimización aplicado sobre bases de datos astronómicas de gran tamaño.

En los últimos años, la necesidad de almacenar cada vez más cantidad datos en observatorios y centros de investigación para su posterior estudio hace imposible el análisis eficaz de toda la información, por lo que es de vital importancia disponer de algoritmos y sistemas que procesen grandes cantidades de datos.

Normalmente, en astronomía, grandes cantidades de datos se refiere a un gran número de imágenes, que son la principal herramienta de la que dispone la comunidad científica para analizar los fenómenos astrofísicos. A través de las imágenes podemos obtener desde un tránsito planetario hasta la composición atmosférica de algunos exoplanetas, todo esto observando la luz en diferentes longitudes de

onda que proviene de las estrellas. La luz que nos llega en diferentes longitudes de onda representa fenómenos diferentes, es decir, a más corta longitud de onda, observaremos fenómenos mucho más violentos y energéticos, como pueden ser las supernovas. Las imágenes obtenidas en este trabajo se han obtenido observando la banda óptica, la cual podemos observar totalmente desde la superficie terrestre.

Actualmente, el análisis de las imágenes astronómicas es muy tedioso, el simple proceso de corregirlas de efectos instrumentales ya implica un gasto de tiempo importante para comenzar el análisis. Una vez se comienza el estudio, en el caso de descubrimiento de objetos transneptunianos, debemos ir imagen por imagen aplicando de técnicas de detección, lo que implica un gasto de tiempo enorme.

Recientemente se ha publicado una aplicación desarrollada por la Universidad Politécnica y el Instituto de Astrofísica de Canarias llamada "Cazasteriodes", la cual está disponible para Android de manera gratuita. Esta aplicación tiene el mismo objetivo que este trabajo, aunque usando otros medios. Los usuarios deben detectar dentro de una secuencia de imágenes que objetos se mueven respecto al fondo plano de estrellas, las cuales están fijas durante toda la secuencia. En este caso, sirve para detectar asteroides, pero la filosofía es prácticamente la misma, analizar qué objetos no conocidos se están moviendo de la manera que buscamos.

En este trabajo se pretende crear algo similar pero utilizando técnicas de ciencia de datos y machine learning que sean capaces de analizar las imágenes como una persona lo haría. Se desarrollará un algoritmo para realizar un preprocesamiento sobre imágenes astronómicas, las cuales, hay que tratar para limpiarlas de fuentes conocidas para favorecer el descubrimiento de nuevos objetos transneptunianos en el sistema solar. Este algoritmo será muy novedoso en la comunidad astronómica y capaz de reproducir el aspecto de una imagen astronómica, simulando las estrellas conocidas en ella.

Dicho algoritmo creará una imagen sintética usando datos y posiciones de estrellas obtenidos por la sonda espacial GAIA. La imagen sintética simulará fielmente cómo sería una región de cielo "perfecta", es decir, contendrá solo las fuentes conocidas en esa zona, de manera que posteriormente podremos analizar la imagen es busca de posibles candidatos a objetos transneptunianos mucho más rápidamente. Las técnicas para la creación y uso de imágenes que simulan el cielo

para fines de descubrimiento de objetos es totalmente novedosa, dotándola de un gran interés, sobretodo observando la enorme utilidad que esta podría tener en un futuro para el análisis masivo de datos.

El desarrollo de técnicas implica un plus de complejidad, lo que significa que se deben implementar métodos y afrontar problemas no vistos hasta ahora. En la actualidad, casi cualquier tecnología pionera que se disponga a resolver problemas complejos, usa ciencia de datos, y en este caso no será diferente. En el caso de este trabajo, se utilizarán algoritmos de búsqueda y optimización, los cuales han dado soporte a innumerables campos como las finanzas, ecología, ingeniería, estudio de redes sociales, aeronáutica, etc. En concreto, se utilizará un algoritmo genético para crear las estrellas artificiales que compondrán la imagen sintética. Esta idea es nueva y permite conseguir resultados muy aproximados a la realidad.

La cantidad de datos que actualmente se tienen para el análisis de algunas regiones ronda los 2 Terabytes de información, lo que supone una cantidad enorme para poder examinarla de manera eficaz por personas. El objetivo final de un futuro programa que use esta tecnología será la detección automática de estos objetos, de esta forma las personas encargadas de hacer este proceso actualmente podrían dedicar su labor a otras tareas.

En una última etapa, se realizará un análisis preliminar del diseño para Spark para el uso del algoritmo de forma paralela en el conjunto total de las imágenes, las cuales, en última instancia contendrán los candidatos a nuevos descubrimientos.

1.2. Objetivos generales

Con la motivación presentada se pretende crear un algoritmo Big Data que permita preprocesar y eliminar las fuentes estelares conocidas de imágenes astronómicas reales. Además, para agilizar el procesamiento de las imágenes, es importante reducir su tiempo de ejecución y consumo de memoria en la medida de lo posible.

Para llevar a cabo la implementación se han diferenciado varias etapas que engloban todo el proyecto:

En primer lugar tendremos una etapa de toma de decisiones inicial, donde se plantean y se valoran diferentes ideas para dar solución al problema en cuestión. En esta etapa se realizan reuniones presenciales para aclarar conceptos y poner en común las bases del trabajo. Seguidamente se hará una revisión bibliográfica, consultando algunas fuentes de interés que permitan abordar de la mejor manera posible el trabajo.

Previo desarrollo del algoritmo habrá una etapa de creación de funciones para facilitar el manejo tanto de librerías en Python como de algunos programas externos. Los ficheros astronómicos tienen una configuración singular, por tanto, es necesario disponer de funciones que permitan acceder y modificar datos de manera rápida y eficaz.

Tras ello, se diseñará e implementará un algoritmo genético que permita aproximar gaussianas 2D en imágenes. El objetivo en esta etapa es que el algoritmo proporcione, mediante una configuración de parámetros inicial, gaussianas 2D lo más parecidas posibles a las fuentes estelares reales que encontramos en las imágenes.

Una vez diseñado el algoritmo genético se procederá a realizar el cálculo de la imagen sintética que posteriormente servirá para obtener la imagen final sin las fuentes conocidas proporcionadas por GAIA. A partir de la imagen final se pretende obtener los candidatos a nuevos objetos haciendo uso del programa Sextractor y las funciones previamente desarrolladas.

Por último, se realizará un análisis preliminar del diseño para Spark.

1.3. Estructura de este documento

Este documento se estructura en: introducción (la presente parte), estado del arte y conceptos, desarrollo de funciones para manejo de librerías y ficheros

astronómicos, desarrollo del algoritmo genético, implementación del programa, resultados y conclusiones.

En el capítulo de estado del arte y conceptos se describirán todos aquellos componentes y conocimientos necesarios para entender el presente trabajo.

El capítulo de desarrollo de funciones contiene tanto la explicación de algunas librerías y programas astronómicos usados, como algunas funciones implementadas para facilitar el manejo de dichas librerías en este trabajo.

El posterior capítulo se centrará en explicar como se ha desarrollado un algoritmo genético para aproximar las gaussianas creadas por las fuentes astronómicas en imagenes.

Seguidamente se detallará la implementación del código para obtener la imagen final, donde se pretende obtener los candidatos a nuevos descubrimientos.

Finalmente se comentarán los resultados obtenidos, para seguidamente, en las conclusiones, dar una valoración de los mismos, un análisis de los problemas surgidos durante el desarrollo del trabajo y las vías de desarrollo futuro que se ven para el sistema.

Capítulo 2

Estado del arte y conceptos

Contenidos

2.1. Análisis de imágenes astronómicas

Para el desarrollo de este trabajo será fundamental dar algunas nociones de cómo se obtienen y tratan las imágenes astronómicas que se usarán. Dentro de la astronomía, la principal forma que tienen los astrónomos para analizar y estudiar el comportamiento de fenómenos astrofísicos, son las imágenes. Estas se obtienen mediante fotografías usando cámaras CCD. Un sensor CCD (Charge Coupled

Device) posee un circuito integrado que tiene una cara fotosensible, la cual es capaz de detectar luz.

Normalmente, los sensores CCD (Figura 2.1) tienen entre 2000 y 4000 píxeles, por lo que cada lado mide sobre 1-3 centímetros. Los píxeles son tan pequeños que normalmente la imagen de una estrella abarca varios píxeles adyacentes.

Figura 2.1: Sensor CCD

Los sensores CCD recogen la luz visible, en forma de fotones, y generan un carga eléctrica que es almacenada para una posterior lectura de datos. La cantidad de carga es proporcional al número de fotones que inciden en el CCD, por tanto, a mayor iluminación, mayor será la carga. El número que registra la cantidad de carga acumulada se denomina "número de cuentas".

Las imágenes obtenidas por las cámaras CCD, como las utilizadas en este trabajo, deben ser corregidas en lo que es llamado el proceso de reducción de datos[14], donde se elimina cualquier efecto que se produzca de manera artificial o por el sistema instrumental. Dicho proceso consta de tres etapas:

Sustracción del bias: Es una señal de fondo, introducida por la electrónica, la cual se debe restar.

Sustracción de la contribución térmica (o Dark Frames): Restar a la imagen de electrones que se generan en ausencia de luz como consecuencia de la excitación térmica. Normalmente este paso no es necesario puesto que las CCD trabajan a temperaturas muy bajas, donde es poco probable que se de este fenómeno.

Calibración de campo plano (flat-field): Debido a pequeñas variaciones, cada píxel puede tener una sensibilidad distinta, que varía normalmente entre el 1 y el 2 % de la señal media. En este paso se divide la imagen tomada entre una imagen de calibración llamada Flat-field, la cual refleja estas diferencias y las elimina. Esta imagen es tomada apuntando el telescopio hacia una zona de iluminación constante o al amanecer o atardecer.

Finalmente y una vez corregida la imagen, dependiendo de nuestros objetivos necesitaremos otras correcciones como pueden ser el cálculo de la extinción atmosférica o calibraciones astrométricas.

En las imágenes astronómicas podemos encontrar fuentes de luz, la mayoría proveniente de estrellas o galaxias. Debido a la atmósfera y efectos en el sistema óptico (telescopio) las fuentes generan una señal gaussiana como la observada en la Figura 2.2.

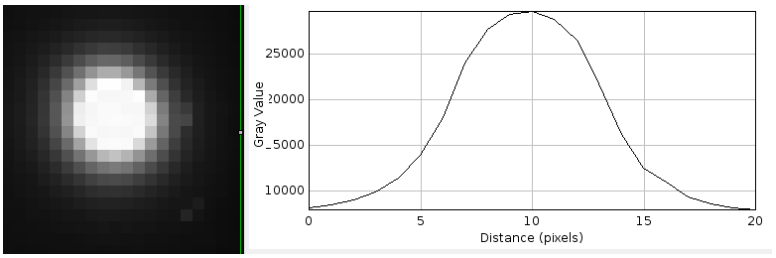

Figura 2.2: Perfil estelar

En este trabajo se describirá un algoritmo capaz de eliminar la mayoría de estas fuentes, dejando visibles aquellas no reconocidas (estrellas o galaxias débiles) o posibles candidatos a objetos transneptunianos.

2.1.1. TNO's (Objetos transneptunianos)

Los objetos transneptunianos[11] (TNOs, del inglés "trans-neptunian objects") son los cuerpos que pueblan la región llamada cinturón de Kuiper[20], un disco circunestelar situado entre 30 y 55 unidades astronómicas del Sol. La unidad astronómica es una unidad de longitud aproximadamente igual a la distancia media entre el planeta Tierra y el Sol (Unos 150 millones de kilómetros).

Los TNOs[10] son objetos de hasta unos 2000 km de diámetro compuestos básicamente de hielo de agua y de otros volátiles como metano (CH4), nitrógeno (N2), además de silicatos. Estos cuerpos se crean en el origen del sistema solar, cuando las partículas heladas y silicatos colisionaron y se agregaron hasta formar objetos helados que quedaron orbitando alrededor del Sol. Muchos de estos cuerpos fueron y son absorbidos por los planetas, pero aún quedan en órbita una gran cantidad que puede ser detectada desde la tierra.

Entonces, ¿Por qué buscar TNOs y no asteroides o NEAs (Asteroides cercanos a la tierra)?. Estos objetos son más difíciles de encontrar ya que provienen de más lejos y son más lentos, es por ello que aún puede existir un gran número sin descubrir.

Para la búsqueda de estos objetos analizaremos imágenes que han sido procesadas para eliminar fuentes conocidas obtenidas desde la sonda espacial GAIA. Una vez tengamos dichas imágenes, buscaremos movimientos que indiquen la presencia de este tipo de objetos.

2.1.2. Sonda espacial GAIA

Para obtener los objetos conocidos y posteriormente eliminarlos de la imagen real acudiremos a la base de datos de la sonda espacial GAIA[8].

GAIA es una misión de la Agencia Espacial Europea (ESA)[16] que tiene por objetivo determinar las posiciones, distancias, movimientos y características físicas de más de mil millones de estrellas de nuestra galaxia, con una precisión sin precedentes. A partir de los datos obtenidos se creará un mapa 3D y dinámico de la galaxia. El objetivo científico es proveer las claves sobre cómo se formó y evolucionó nuestra galaxia. Además, GAIA también realizará detección y clasificación orbital de sistemas planetarios, cartografiado de objetos y pruebas de la posible relación entre la relatividad general y la cosmología.

La misión GAIA actualmente está generando un catálogo de estrellas, el más preciso hasta la fecha. En dicho catálogo encontraremos la posición y brillo de todas las estrellas del cielo de forma muy precisa. Además, GAIA dispone de una

página web donde buscar y obtener los datos de los objetos en una región espacial indicada, lo cual es muy funcional y de vital importancia para el desarrollo de este trabajo.

Figura 2.3: Sonda espacial GAIA

La web de GAIA nos proporciona una lista de estrellas en una región determinada del cielo. De cada una podemos obtener numerosos parámetros, como se puede ver en la Figura 2.4.

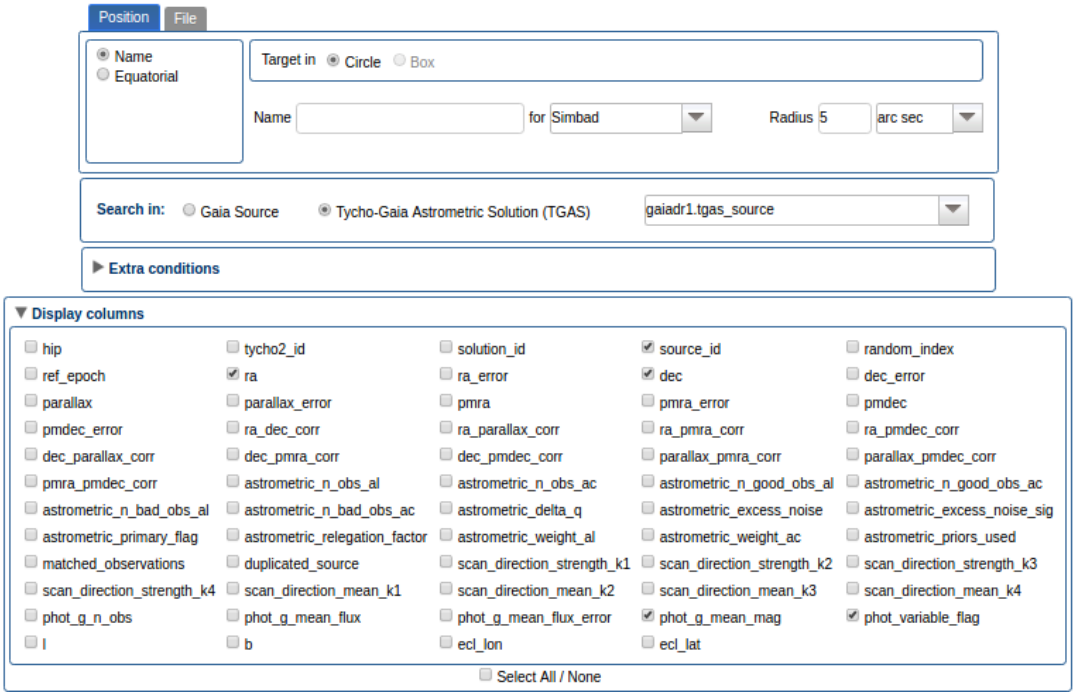

Figura 2.4: Web de GAIA

De toda la lista de parámetros nos interesan dos en concreto, RA (Ascensión recta) y DEC (Declinación). Estos dos parámetros hacen referencia a las coorde-

nadas ecuatoriales, un sistema de posicionamiento de astros en el cielo. Dichas coordenadas deberán ser convertidas posteriormente para la correcta proyección de las fuentes en las imágenes.

Para más información acerca de la misión consultar la web https://gaiaverse.eu/inicio/.

2.1.3. Coordenadas Celestes

Como hemos comentado anteriormente, las posiciones proporcionadas por GAIA hacen referencia al sistema ecuatorial de posicionamiento celeste[15], en esta sección se comentará brevemente cómo se localiza un objeto en el espacio.

Para poder localizar un objeto en el espacio necesitamos tres números, dos que simulan las coordenadas X e Y en un plano y un número L indicando la distancia hasta el objeto. Para dar la posición de los objetos celestes basta con utilizar dos ángulos, que nos proporcionan la dirección en la que se encuentran, sin incluir la distancia.

Dependiendo de los planos de referencia que se utilicen, para definir dichos ángulos y los criterios o convenios de medida tendremos distintas coordenadas celestes: horizontales, ecuatoriales, eclípticas o galácticas. El sistema más utilizado y fundamental es el denominado sistema ecuatorial, cuyas coordenadas son la declinación o distancia al ecuador y la ascensión recta que es ángulo contado desde el punto Aries en el sentido de la rotación terrestre.

En este trabajo usaremos las coordenadas ecuatoriales[4], que son equivalentes a la latitud y longitud terrestres proyectadas en el cielo. Este sistema de coordenadas es fijo y por ende rota al rotar la Tierra. Cada estrella tiene una ascensión recta (longitud) y una declinación (latitud) en el cielo. Cuando se obtiene una imagen del cielo, este sistema está proyectado en ella.

En la Figura 2.5 se puede observar cómo localizar un objeto usando coordenadas ecuatoriales. Vemos como la tierra está ligeramente girada hacia los polos norte y sur celestes, esta rotación es debida al llamado eje del mundo, el cual permite el movimiento de rotación de la tierra. El ecuador celeste es un círculo

imaginario que cruza perpendicularmente el eje del mundo, que además interseca con el plano de eclíptica, el cual indica el camino del sol durante un año. Uno de los puntos de cruce es el llamado punto Aries (o punto Vernal), que permite medir la ascensión recta dirección Este sobre el plano del ecuador, con el objetivo de identificar en horas, minutos y segundos un astro en el cielo. A partir de la ascensión recta, la elevación del objeto (equivalente a latitud) es medida mediante la declinación. Ambas coordenadas nos posicionan un objeto de forma unívoca en el cielo.

Figura 2.5: Sistema ecuatorial de coordenadas

2.2. Algoritmos genéticos

Los algoritmos genéticos[6][9] son una técnica de búsqueda que basa su funcionamiento en la teoría de la evolución de Darwin. Son métodos adaptativos que se pueden usar para resolver problemas de búsqueda y optimización, basándose en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Un algoritmo genético consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuáles de ellos se adaptan mejor al problema, a su vez, también son seleccionados para la nueva generación. Tras un ciclo que repite este proceso obtendremos el ejemplar que mejor responda al problema que hemos planteado.

Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano.

2.2.1. Historia

Los primeros algoritmos genéticos datan de principios de los cincuenta y fueron desarrollados por biólogos especializados en evolución, los cuales pretendían simular el comportamiento natural.

La aplicación de estos algoritmos en ingeniería no tardó en llegar. Durante la década de los 60 distintos investigadores desarrollaron de forma independiente, algoritmos inspirados en la evolución. En 1965 el profesor Ingo Rechenben, de la Universidad Técnica de Berlín, introdujo la que llamó Estrategia Evolutiva. Esta técnica no introducía población ni cruzamiento, los padres mutaban para producir descendientes y se conservaba el mejor. Posteriormente Fogel desarrolla la Programación Evolutiva, donde se introduce por primera vez el concepto de población, lo que permitía que la salida de resultados no dependiera solo de la entrada de datos actuales. Sin embargo, en estas dos metodologías se obvia el cruce, un paso fundamental en el algoritmo actual.

Holland fue el primero en proponer explícitamente los cruces. Con la publicación del libro “Adaptación en Sistemas Naturales y Artificiales”, basado en sus investigaciones, se presenta el concepto de sistemas digitales adaptativos utilizando la mutación, la selección y el cruzamiento, surgiendo así, los algoritmos genéticos.

A mediados de los 80, los algoritmos genéticos se aplicaron en una gran cantidad de campos, desde las matemáticas a problemas de optimización. Hoy en día,

están resolviendo problemas de áreas de estudio tan diversas como la predicción en la bolsa, ingeniería aeroespacial, etc.

2.2.2. Metodología

Normalmente, los algoritmos genéticos basan su funcionamiento en 5 pasos: Inicialización, evaluación, selección, cruce y mutación.

Lo primero, y uno de los pasos más importantes en el desarrollo de un algoritmo genético, es el diseño del cromosoma y inicialización de la población. En primer lugar se debe establecer los parámetros con los cuales evaluaremos a los individuos, lo que se denomina cromosoma. Estos parámetros deben ir cambiando para adaptarse al problema y evolucionar.

Una vez diseñado el cromosoma debemos inicializar la población de individuos que lucharan por la evolución. El tamaño de la población depende de la naturaleza del problema, pero normalmente contiene varios cientos de posibles soluciones. A menudo, la población inicial se genera aleatoriamente, permitiendo toda la gama de posibles soluciones (el espacio de búsqueda). Ocasionalmente, las soluciones pueden ser sembradas en áreas donde es probable encontrar soluciones óptimas.

Otra de las funciones más importantes dentro del funcionamiento de los algoritmos genéticos es la función de fitness, la cual nos permite evaluar a los individuos. Esta función tiene por objetivo puntuar a los individuos con el fin de distinguir cuáles de ellos son mejores posibilidades.

Una vez calculadas las puntuaciones para cada individuo nos disponemos a seleccionar a los más aptos. Ciertos métodos de selección evalúan la aptitud de cada solución y preferentemente seleccionan las mejores soluciones. Otros métodos califican sólo a una muestra aleatoria de la población, ya que el proceso anterior puede llevar mucho tiempo.

El siguiente paso es crear nuevos individuos a través de una combinación de operadores genéticos: entrecruzamiento cromosómico (también llamado crossover o cruce) y mutación. Para cada nueva solución se seleccionan un par de soluciones

padres y se combinan para formarla. Se seleccionan nuevos progenitores para cada nueva solución, y el proceso continúa hasta que se genere una nueva población de soluciones de tamaño apropiado.

Tras la etapa de cruce se procede a mutar algunos genes del cromosoma. Solo se realizará la mutación si al calcular un valor aleatorio este está por debajo de una determinada cantidad llamada tasa de mutación. Si esta cantidad es muy pequeña puede conducir a la deriva genética, lo cual nos indica poblaciones poca variadas. Por contra, si es muy alta, puede conducir a la convergencia prematura del algoritmo genético.

El algoritmo puede terminar de dos formas, tras un número de generaciones o la detección de convergencia del mismo, en ambos casos, se seleccionará al mejor individuo de la última generación como mejor solución.

2.3. Ciencia de datos y Big Data

Actualmente la evolución de las tecnologías, las redes sociales, la capacidad de almacenamiento, etc., permiten que cada vez se intercambian y generen más datos. Toda esta información se puede recibir a alta velocidad y su procesamiento normalmente requiere un análisis muy costoso en términos temporales. La gran cantidad de datos disponibles junto con las herramientas capaces de analizarlos y procesarlos de manera eficaz es lo que se conoce como Big Data.

Dicho concepto engloba infraestructuras, tecnologías y servicios que dan soporte al procesamiento de estas grandes cantidades de datos. Podríamos diferenciar las aplicaciones Big Data como aquellas que trabajan con volúmenes de datos que superan la capacidad del software habitual para ser manejados y gestionados. Debido a la continua evolución y progreso tecnológico dicho concepto se va actualizando debido al incremento en la potencia de cómputo y almacenamiento cada vez mayores.

Con grandes volúmenes de datos no referimos a Terabytes o Petabytes, es decir, cantidades de datos que hace unos años era impensable procesar de manera

eficaz. Además, estos datos, son obtenidos de manera muy variada, ya que hoy por hoy podemos recibir información de multitud de fuentes diferentes como pueden ser Redes Sociales, dispositivos electrónicos y móviles, sensores, internet de las cosas, etc.

En aplicaciones convencionales la información recibida y procesada es información estructurada que ha pasado por numerosos filtros de calidad para poder garantizar que la información de salida tiene una precisión y una exactitud determinada. Sin embargo, cuando hablamos de Big Data nos referimos a información que puede estar semiestructurada o no tener ninguna estructuración. La gestión de esta información desestructurada precisa de una tecnología diferente y permite tomar decisiones basadas en información que tiene importantes grados de inexactitud.

En Big Data también es muy importante el concepto de velocidad, el cual se refiere a la rapidez con que los datos se reciben, se procesan y se toman decisiones a partir de ellos. A la mayoría de los sistemas tradicionales les es imposible analizar de forma inmediata los grandes volúmenes de datos que les llegan, por ello, es un concepto muy presente en las tecnologías Big Data.

Una vez estamos en disposición de analizar los datos, debemos definir qué es el valor de los mismos y para qué vamos usarlos. El valor es la importancia del dato para el negocio, lo que en última instancia permite hacer ofertas enfocadas a un sector de la población o lanzar productos cuando el mercado es más susceptible de comprarlos.

El reto de procesar grandes cantidades de información de forma precisa ha estado siempre presente. La mayor parte del tiempo se ha trabajado con pocos datos debido a que las herramientas para obtener, organizar, almacenar y analizar eran muy pobres. Hoy el entorno técnico ha cambiado por completo. Siempre hay, y habrá, un límite en cuanto a los datos que podamos manejar, pero es mucho menos limitante de lo que solía ser y se volverá incluso más manejable con el tiempo.

2.4. Spark

Apache Spark es un framework para realizar computación de forma paralela en clústeres. Fue desarrollado en la Universidad de California y proporciona una interfaz entre código y clústeres para la implementación de paralela de aplicaciones.

Apache Spark proporciona una interfaz de programación de aplicaciones centrada en una estructura de datos denominada RDD, la cual mantiene los datos distribuidos en modo de sólo lectura. Se desarrolló como consecuencia a las limitaciones en la computación de cluster MapReduce[12][13], que obliga a una estructura de flujo de datos lineal en programas distribuidos.

Los RDDs de Spark funcionan como un espacio de trabajo para programas distribuidos, los cuales acceden de una forma común para compartir la memoria distribuida. Los RDDs facilitan la implementación de algoritmos iterativos y mejoran el análisis de datos exploratorio, ya que hace más eficaz la consulta repetida de datos en una misma ejecución. La latencia de las aplicaciones desarrolladas en Spark puede verse reducida muy notablemente en comparación con implementaciones anteriores a Spark. Entre la clase de algoritmos iterativos comúnmente implementados están los algoritmos de entrenamiento para sistemas de aprendizaje de máquina, que fueron una de las motivaciones iniciales para el desarrollo de Apache Spark.

Apache Spark tiene un sistema de almacenamiento distribuido con el cual puede interactuar con una amplia variedad de sistemas de archivos, incluyendo Hadoop Distributed File System (HDFS), Sistema de archivos MapR (MapR-FS), Cassandra, OpenStack Swift, Amazon S3 o Kudu. Spark también proporciona un modo local pseudo-distribuido, el cual es usualmente utilizado para propósitos de desarrollo o pruebas, donde el almacenamiento distribuido no es necesario y se puede usar el sistema de archivos local. En tal escenario, Spark se ejecuta en una sola máquina con un ejecutor por núcleo de CPU. En ambos casos el proceso de ejecución suele seguir el esquema de la Figura 2.6.

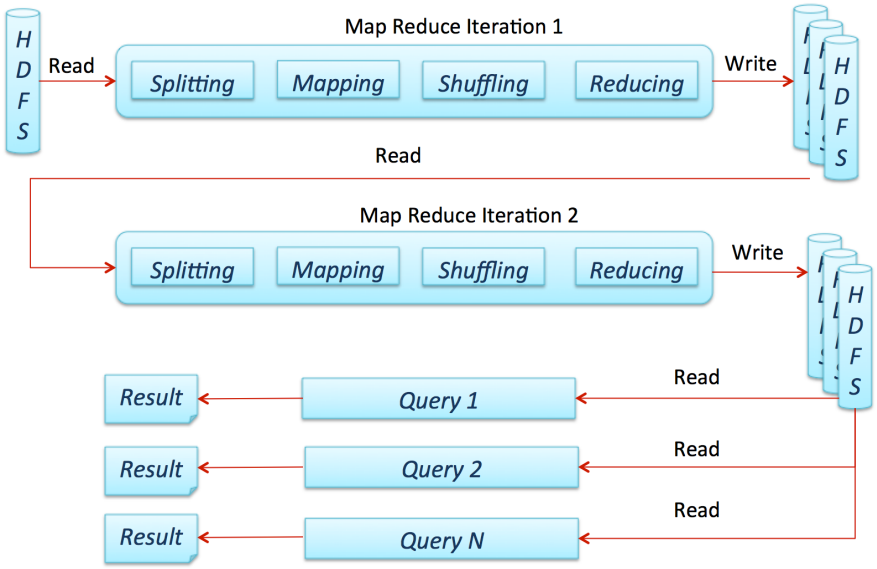

Figura 2.6: Proceso de ejecución usando Spark

Capítulo 3

Desarrollo de funciones para manejo de librerías y ficheros astronómicos

Contenidos

Para el desarrollo de este trabajo se utilizarán algunas librerías y programas externos, por lo que en este capítulo se comentará tanto la necesidad como la funcionalidad de cada uno de ellos. Además, también se expondrá alguna función desarrollada.

3.1. Ficheros FITS

FITS es el formato de archivo más comúnmente usado en el mundo de la astronomía. Este formato permite almacenar una gran variedad de datos, como

pueden ser imágenes, espectros electromagnéticos, listas de fotones, etc. En el caso de las imágenes el fichero se divide en dos: datos y cabecera.

Una de las ventajas de los ficheros FITS es que la información de las cabeceras es legible en ASCII, de modo que un usuario puede examinar las cabeceras para investigar un archivo de procedencia desconocida. Las cabeceras contienen secuencias de 80 cadenas de caracteres fijos que llevan pares de valores, interpolados entre los bloques de datos. Los pares de valores proveen información como son el tamaño, origen, formato binario de los datos, comentarios, historia de los datos y cualquier otra información que el creador desee.

Para la lectura de ficheros FITS se utiliza en primera instancia el paquete *"astropy.io.fits"*, el cual permite el acceso a los ficheros y la carga de las diferentes partes que lo integran. Con el objetivo de ahorrar código se implementan funciones para la lectura y escritura de ficheros de FITS usando el paquete comentado anteriormente.

Las posiciones de las estrellas, dentro de los ficheros FITS, se representan mediante dos coordenadas (X,Y) en píxeles. Esto representa un problema ya que GAIA proporciona dichas estrellas usando coordenadas ecuatoriales (Ascensión recta, Declinación), las cuales se deben convertir usando programas destinados para ello y que se comentarán en la siguiente sección.

3.2. Cambio de coordenadas: sky2xy

El fichero de datos que obtenemos de GAIA tiene un formato CSV donde tenemos dos columnas que nos indican la ascensión recta y la declinación para un determinado objeto. La ascensión recta se mide en horas, minutos y segundos, mientras que la declinación es un grado con valor [-90, 90] dependiendo de la latitud del observador. A partir de estas coordenadas debemos localizar nuestra región espacial y posteriormente realizar una conversión a pixel (X,Y).

Para la correcta conversión de coordenadas se hace uso del programa *"sky2xy"*, el cual, a partir de un fichero FITS, permite la transformación de ascensión recta

y declinación a píxeles (X,Y) dentro de un tamaño de imagen definido por el fichero FITS pasado al programa.

El programa *"sky2xy"* puede ser encontrado en su página web oficial (http://tdc-www.harvard.edu/wcstools/sky2xy/). Su uso se realiza a través de consola, utilizando el comando *"sky2xy"* seguido de una imagen FITS, de la cual obtendrá los tamaños y coordenadas para la conversión, y dos posiciones de cielo (Ascensión recta, Declinación). El programa nos devolverá la posición (X,Y) donde se encontraría ese objeto en la imagen.

A continuación se muestra un pequeño código donde se implementa un tokenizador de cadenas de caracteres recibidas desde consola BASH para la ejecución del programa *"sky2xy"*. A partir de los tokens se actualiza la base de datos de objetos para cambiar las coordenadas.

```
def RaDec2Pixels():
  for i in range(database.shape[0]):
    cmd = ['sky2xy', headerName, str(database[i, 0]), str(database[i, 1])]
    output = subprocess.Popen(cmd, stdout=subprocess.PIPE).communicate()[0]
    output = output[0:len(output)-2]
    items = [x for x in output.split(' ') if x]
    database[i, 0] = int(float(items[4]))
    database[i, 1] = int(float(items[5]))
```

Tras la conversión se tienen las posiciones de todas las estrellas correctamente para formar la imagen sintética, que posteriormente usaremos para calcular la imagen final sin fuentes conocidas, que debemos analizar para comprobar que realmente se han ido eliminando dichas estrellas.

3.3. Detección de fuentes con SExtractor

La detección de fuentes no es tarea fácil puesto que se tienen que dar varias circunstancias para clasificar un objeto como fuente. Para ello haremos uso del programa *"SExtractor"*, el cual construye un catálogo de objetos a partir de una imagen astronómica. Está particularmente orientado a la reducción de los

datos de galaxias a gran escala, pero también funciona bien en campos estelares moderadamente poblados.

El programa *"SExtractor"* puede ser encontrado en su página web oficial (https://www.astromatic.net/software/sextractor) y basa su comportamiento en una red neuronal que es capaz de clasificar si un determinado objeto es una fuente. Una vez se configuran los parámetros de comportamiento, lo cual puede ser bastante complejo, su uso es sencillo, basta con ir al directorio donde lo tenemos configurado y ejecutar en un terminal *"sex fichero.fits"*, esto crearía un fichero *"a.out"* donde ha escrito todos los detalles de la extracción de fuentes para su posterior análisis con programas destinados a la observación de ficheros FITS.

3.4. AstroImageJ y MACROS

Existen diferentes programas para observar las imágenes astronómicas de forma correcta. Estos programas proporcionan muchas funcionalidades muy necesarias a la hora de estudiar y comprender los fenómenos presentes en las imágenes.

En el caso de este trabajo se ha utilizado *"AstroImageJ"*, que permite usar un gran número de funcionalidades y posibilita la ejecución de MACROS. Las MACROS son un conjunto de instrucciones que se ejecutan de manera secuencial desde el programa. En este caso contendrán comandos que formarán elipses en determinadas posiciones de la imagen, las cuales serán candidatos a TNOs. Estas posiciones las obtenemos del fichero *"a.out"* que *"SExtractor"* crea en el proceso de detección.

"AstroImageJ" se puede encontrar en su web oficial y es multiplataforma, por lo tanto podemos lanzarlo desde Mac, Windows y Linux. Tiene una interfaz interactiva que permite la lectura y escritura de ficheros FITS, así como todo un conjunto de herramientas para el análisis de imágenes astronómicas.

En la Figura 3.1 podemos obsevar las fuentes detectadas por *"SExtractor"* con elipses dibujadas alrededor de ellas usando *"AstroImageJ"*.

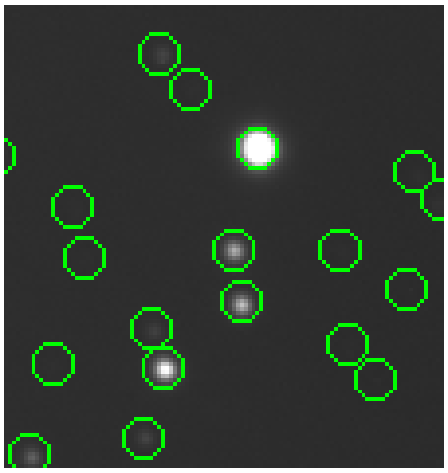

Figura 3.1: Ejemplo de fuentes identificadas con SExtractor

Finalmente, en la Figura 3.2 podemos ver un diagrama de procesos de como es el análisis y tratamiento de una imagen astronomica de inicio a fin.

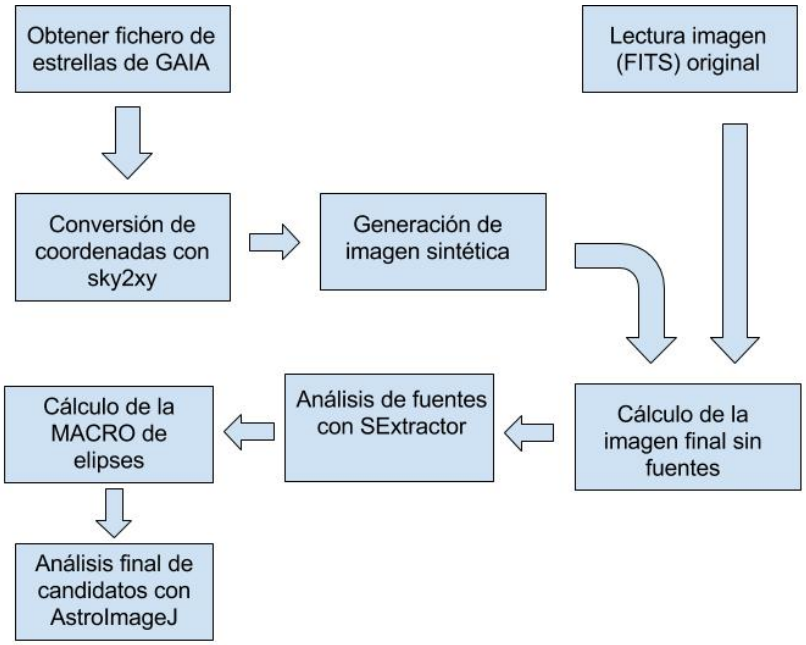

Figura 3.2: Diagrama de tratamiento de una imagen astronómica

Capítulo 4

Desarrollo de un algoritmo genético para aproximar gaussianas 2D en imágenes

Contenidos

Como se ha comentado previamente, las estrellas generan formas gaussianas en las imágenes y el objetivo final es eliminarlas de la mejor forma posible, es decir, buscar la configuración óptima de una gaussiana que se adapte a la fuente. Esto es exactamente el objetivo de los algoritmos de búsqueda y optimización, entre ellos los algoritmos genéticos, que ofrecen mucha facilidad de diseño y desarrollo para este tipo de problemas.

La generación de gaussianas no es tarea sencilla ya que los parámetros deben variar para adaptarse de la mejor manera a fuentes influidas por distintos factores en las diferentes imágenes. Si observamos una fuente puntual, en dos imágenes distintas (Figura 4.1), la forma que tendrá será diferente (dependiendo de muchos factores), por tanto, aunque existe una configuración de parámetros que aproximan la gaussiana en función del brillo, estos no son los mejores y deberán adaptarse en cada imagen.

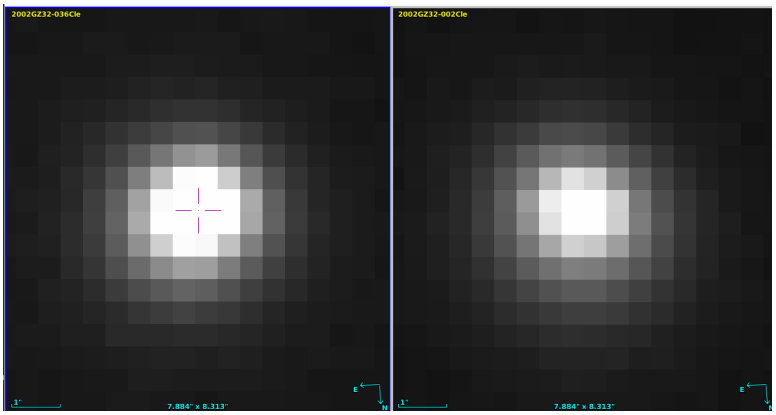

Figura 4.1: Diferencia entre una misma fuente en diferentes imágenes

En este capítulo se detallará el diseño del algoritmo genético que eliminará las fuentes conocidas de la región de cielo presente en las imágenes astronómicas.

4.1. Creación de gaussianas en 2D

La creación de la gaussiana[18][19] se realizará sobre una región cuadrada, donde muy próximo del centro estará el máximo valor y irá disminuyendo a medida que nos alejamos de él, llegando a ser 0 en los bordes de la región. En la Ecuación 4.1 podemos observar la fórmula que permite crear gaussianas en una región, se forman partiendo de valor 1 en su máximo y disminuyendo hacia 0, por lo que habrá que multiplicar por el máximo valor ($MaxValue$) que queramos conseguir.

"Reduction" es un parámetro que indica el valor con el cual se va a ir reduciendo la intensidad de la gaussiana, es muy importante puesto que ya no se

depende del flujo o magnitud de la estrella, el propio algoritmo es capaz de ajustarlo. Adicionalmente tenemos dos parámetros *"xFact"* y *"yFact"* que varían para formar gaussianas más ovaladas. Ambos parámetros se multiplican por la resta del tamaño de la región y la posición inicial de la gaussiana en ese eje.

Por último tenemos el parámetro Sigma, que indica la llamada Anchura a Media Altura (FWHM del ingles Full Width at Half Maximum). Este valor es la distancia entre puntos de una curva en la que la función alcanza la mitad de su valor máximo.

$$Star = MaxValue * e^{Reduction * \frac{xFact*(x-x0)^2 + yFact*(y-y0)^2}{2\sigma^2}} \tag{4.1}$$

El algoritmo intentará ajustar gaussianas a fuentes reales, cuando obtenga la mejor solución posible, este devolverá la negativa de la estrella. Esto es debido a problemas con el fondo del cielo, el cual también tiene un valor de brillo y fuentes muy débiles próximas a ese número que podemos perder si no usamos el negativo posteriormente.

4.2. Preprocesamiento previo de la región

Para realizar el ajuste de la gaussiana se recorta la región de la imagen real que deja la fuente en el centro (o lo más cercano a él). Si esa fuente esta en borde y la imagen no permite que el recorte sea cuadrado se rellena dicha caja con el valor mediana en esa región de cielo. Una vez hecho el recorte, nos quedará una matriz donde tendremos una fuente en el centro (quizás alguna mas por otros lugares) y lo demás tendrá valor de cielo.

El valor de cielo oscila en un rango sobre toda la imagen, esto quiere decir que en zonas donde no hay fuentes el valor entre los píxeles no es el mismo. Esto puede complicar el funcionamiento del algoritmo genético ya que si el objetivo es obtener una gaussiana que minimice la suma entre la fuente real y la sintética (En negativo), un valor de cielo alto causa que la gaussiana se forme más grande. Esto es un efecto lógico ya que la fuente sintética tenderá a suprimir también ese valor

de más en el cielo. En la Figura 4.2 podemos observar qué ocurre si usamos una región con el fondo de cielo intacto para la búsqueda de la gaussiana, creándose más grande de lo necesario.

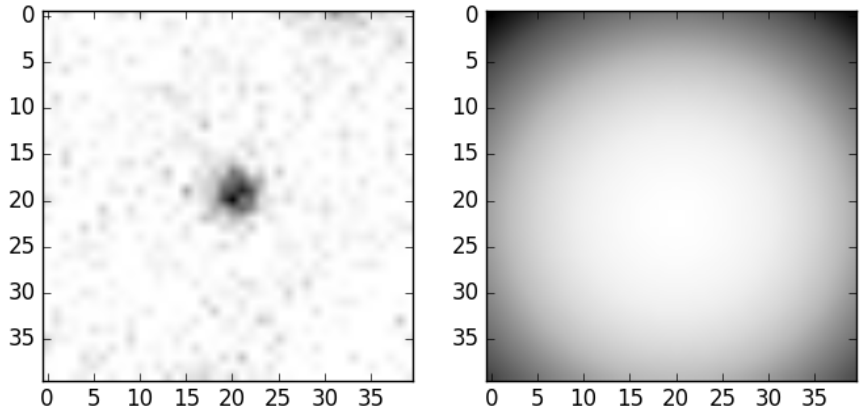

Figura 4.2: Gaussiana creada sin tener en cuenta el valor de cielo

La solución a este problema es eliminar, en la medida de lo posible, la influencia de ese valor de más en el fondo de la imagen. Para ello basta con calcular el valor más repetido dentro de esa región y restárlselo. En la figura 4.3 podemos observar la misma región de antes pero con el cielo sustraído, comprobando cómo se forma una gaussiana mucho más acorde al tamaño original de la fuente.

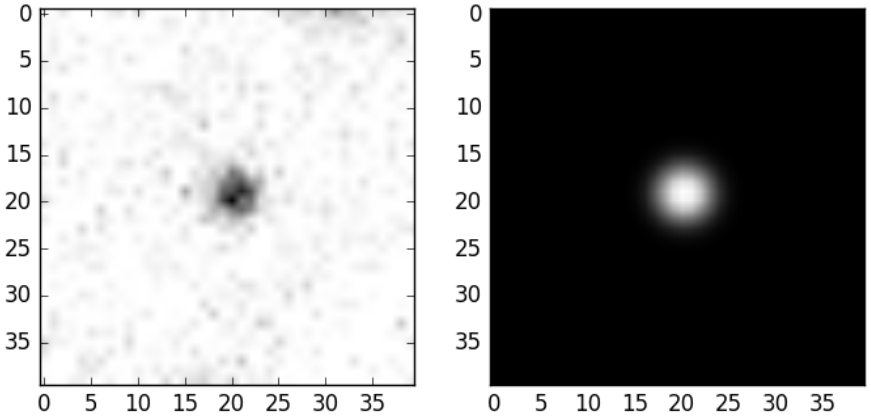

Figura 4.3: Gaussiana creada teniendo en cuenta el valor de cielo

4.3. Descripción del algoritmo genético

En las siguientes secciones se describirá como se ha decidido codificar el cromosoma que configura a los individuos, como se inicializa la población en el espacio de soluciones y finalmente se comentarán tanto los operadores de selección, cruce y mutación, como la función de fitness que evalúa a los individuos.

4.3.1. Descripción del cromosoma

El cromosoma de los individuos estará formado por un vector de 8 posiciones, donde los primeros 7 representarán los parámetros de ajuste para la gaussiana y el último el valor de aptitud de dicho individuo. Los parámetros que forman el cromosoma son los siguientes:

1. MaxValue: Valor máximo en el centro de la fuente real.
2. FWHM o Sigma: Agrandar o decrementar el tamaño de la gaussiana.
3. xOffset: Pequeño valor sumado a la posición inicial en el eje X.
4. yOffset: Pequeño valor sumado a la posición inicial en el eje Y.
5. xFactor: Valor para crear gaussianas más ovaladas en el eje X.
6. yFactor: Valor para crear gaussianas más ovaladas en el eje Y.
7. ReductionFactor: Valor con el cual se reduce la intensidad de la gaussiana en cada píxel.

Los dos parámetros (*"xOffset"* e *"yOffset"*) se añaden con el fin de suprimir pequeñas diferencias de píxeles después de realizar la conversión de coordenadas entre sistema ecuatorial y píxeles en la imagen.

4.3.2. Inicialización de la población

Previo uso de los operadores de selección, cruce y mutación, es necesario comentar cómo se realiza la inicialización de la población y sus individuos. Los parámetros que conforman los cromosomas de los individuos se inicializan directamente en un rango de valores donde la creación de gaussiana es acorde al tamaño de la región disponible.

Para comenzar el proceso se realiza la inicialización de la población en una matriz de 250 filas (valor por defecto de individuos) y 8 columnas en los siguientes rangos o valores:

- MaxValue: Valor máximo de la fuente real.
- FWHM o Sigma: Valor aleatorio entre [1, 90]
- xOffset: Valor aleatorio en el rango [-8, 8]
- yOffset: Valor aleatorio entre [-8, 8]
- xFactor: 1 si yFactor es mayor que 1 y aleatorio entre [1, 3] en caso contrario.
- yFactor: 1 si xFactor es mayor que 1 y aleatorio entre [1, 3] en caso contrario.
- reductionFactor: -0.00008.

4.3.3. Selección

Para realizar la elección de individuos más aptos se usa la selección por torneo determinístico, donde aleatoriamente se seleccionan un número p de individuos y pasa a la siguiente generación el que más aptitud tenga de ellos. En el caso de este algoritmo se seleccionará un número de individuos igual a la mitad del tamaño actual de población, estos serán los elegidos para pasar a la siguiente generación. Por último, para evitar problemas de convergencia del algoritmo, se añade la selección elitista de los dos mejores individuos de la población anterior.

4.3.4. Cruce

Para realizar el cruce que creará nuevos individuos se utiliza el llamado cruce uniforme, donde cada gen de la cadena del nuevo individuo pertenece con una misma probabilidad al padre o a la madre. El objetivo del cruce es generar una nueva población de individuos del mismo tamaño que la anterior, para ello, se copian los dos mejores individuos elegidos en la etapa anterior y posteriormente se van extrayendo aleatoriamente dos para realizar el proceso de cruce hasta completar la actual población.

4.3.5. Mutación

La mutación es un proceso muy importante para garantizar la diversidad de la población. Si a un individuo le toca mutar una vez establecida una probabilidad (0.1 en este caso) de mutación, se selecciona aleatoriamente un gen y se muta para modificarlo. En este algoritmo, como los valores se mueven en rangos es importante acotarlos para evitar errores en la generación de la gaussiana. Los genes se mutan de la siguiente manera:

- MaxValue: Se le suma un valor entero aleatorio en el rango (-1000,1000)
- FWHM o Sigma: Se le suma un valor aleatorio en el rango (-1,1) teniendo en cuenta no violar rango original (1,90).
- xOffset: Se le suma un valor aleatorio en el rango (-1,1) teniendo en cuenta no violar rango original (-8,8).
- yOffset: Se le suma un valor aleatorio en el rango (-1,1) teniendo en cuenta no violar rango original (-8,8).
- xFactor: Se le suma un valor aleatorio en el rango (-0.1,0.1) teniendo en cuenta no violar rango original (1,3).
- yFactor: Se le suma un valor aleatorio en el rango (-0.1,0.1) teniendo en cuenta no violar rango original (1,3).

- reductionFactor: Se le suma o resta de forma aleatoria un valor en el rango (-0.00008,-0.00007).

4.3.6. Función de fitness

La función de fitness define cómo se va a comportar y evolucionar nuestro algoritmo genético. Esta función pretende minimizar el error cometido al sumar la gaussiana artificial y la región real de la imagen. En la Ecuación 4.2 vemos como la función consta de dos sumatorios, ambos sobre el tamaño de la región sobre la cual se ha hecho el recorte (Filas y columnas). Para cada píxel se calcula la suma, teniendo en cuenta que la imagen sintética es negativa, por tanto actúa como una resta. Una vez calculada la aptitud de cada individuo, la función devuelve la población ordenada de más apto (menos error cometido) a menos apto.

$$Fitness = \sum_{I=0}^{Row} \sum_{J=0}^{Columns} (RealStar[i,j] + SyntheticStar[i,j])^2 \qquad (4.2)$$

Esta ecuación es bastante simple, lo que permite entender muy fácilmente que se está haciendo. Además, se puede realizar directamente sobre toda la región, lo que hace que se calcule de forma muy eficiente frente a otras opciones más complejas y costosas en tiempo de ejecución.

Capítulo 5

Propuesta de preprocesamiento de imágenes

Contenidos

En este apartado se define la propuesta a partir de la cual se va a realizar la implementación. Se definirá como se realizará el cálculo de la imagen sintética y la posterior obtención de la imagen sin las fuentes conocidas.

5.1. Creación de la imagen sintética

Haciendo uso de la web de GAIA obtenemos el fichero con el cual vamos a crear la imagen sintética que nos servirá de ayuda a la hora de realizar el procesamiento para obtener una imagen "limpia" de objetos conocidos.

El fichero GAIA se utilizará para realizar la conversión entre sistema ecuatorial y pixeles en la imagen haciendo uso de la utilidad *"sky2xy"*. Dicho resultado nos actualizará la posición de cada fuente, que aún puede estar un poco desplazada respecto a la original. Además, si la posición actualizada quedara fuera de la

región delimitada por el cabecera de la imagen real, el cálculo de dicha fuente se omitiría.

Una vez comprobado el rango de la estrella, se realiza el cálculo de la gaussiana para dicha fuente, utilizando 250 individuos, 20 generaciones máximas y 5 generaciones como límite para que el algoritmo continue convergiendo, si en ese plazo el mejor individuo es el mismo, se considera terminado el proceso. Esta configuración de parámetros ha sido escogida de forma empírica, ya que se alcanzan buenas soluciones en un tiempo acorde. Aun así, en una futura implementación de forma paralelizada se tendría que realizar un pequeño análisis de cuál es el la configuración más beneficiosa en ambos términos.

El cálculo nos devuelve el individuo con más aptitud, es decir, la mejor configuración de parámetros para formar la gaussiana de una determinada fuente. Haciendo uso de la función vamos generando las fuentes artificiales y añadiendolas a una matriz inicializada a ceros.

Al final del proceso (Figura 5.1) tendremos una imagen con todas las fuentes conocidas sobre la región que estamos tratando para utilizarla con el fin de eliminar dichas fuentes en la imagen real. En la Figura 5.2 se puede apreciar como es el resultado de una imagen sintética.

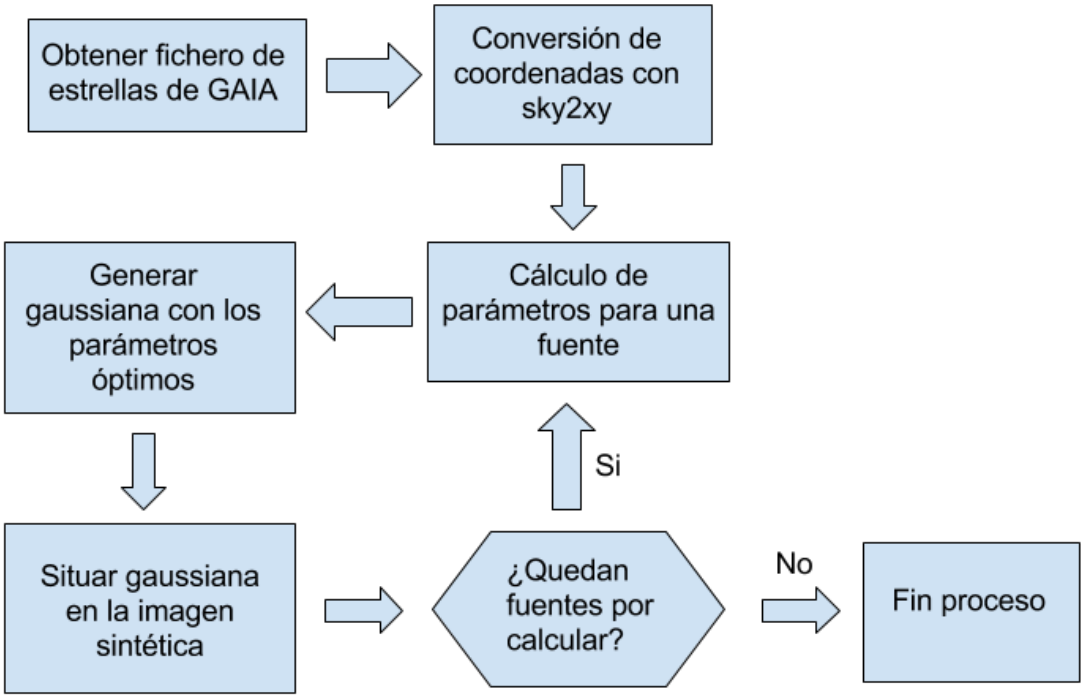

Figura 5.1: Proceso de cálculo de la imagen sintética

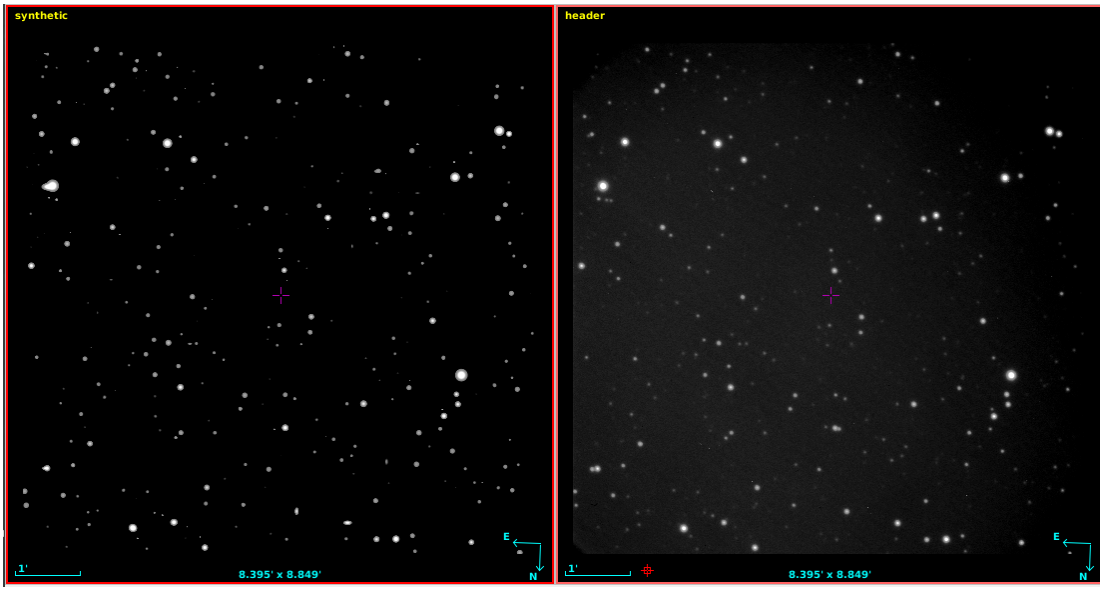

Figura 5.2: Comparación entre imagen sintética y real

5.2. Cálculo de la imagen final

Una vez obtenida la imagen sintética, la cual tiene las mismas dimensiones que la real, se debe realizar un proceso iterativo observando si cada valor de píxel supera un umbral, con el fin de establecerlo o cambiarlo de forma definitiva en esa imagen final.

En una primera versión del algoritmo se calculaba la imagen sintética positiva y se restaba de la real, tras ello, quedaban rastros en la resta que continuaban detectandose como fuentes. Para solucionar ese problema se añadió un umbral que sólo permitía que los objetos más brillantes permanecieran en la imagen final, eliminando así el problema del ruido de las restas. Esta solución añadió un segundo problema, la pérdida de objetos débiles y por tanto, posibles candidatos.

Para remediar este problema se optó por usar la imagen en negativo de manera que, donde el valor calculado para la gaussiana fuese negativo, indicaría una zona con valor superior al fondo. De esta forma, se eliminan los problemas con los objetos con un brillo débil.

Para mejorar el funcionamiento del algoritmo, a la hora de sustituir el valor negativo de la gaussiana en la imagen real, se calcula en el número de cuentas más repetido en un cuadrado alrededor del píxel en cuestión, usándolo para crear un valor aleatorio entre (ValorRepetido-80,ValorRepetido+80) y sustituirlo en la imagen real. De esta manera, cuando se suprime una fuente, la región resultante parece menos alterada.

Finalmente, se le suman a la imagen 32.768 cuentas para situar los valores en el rango que trabajan los ficheros FITS.

Capítulo 6

Implementación

Contenidos

En este capítulo se definirá la implementación del algoritmo que realizará todo el proceso en la obtención de la nueva imagen, utilizando el lenguaje de programación Python. En la segunda sección se comentará de forma teórica la implementación en Spark.

6.1. Implementación iterativa

El primer paso para la creación de la imagen sintética es la lectura del fichero de GAIA con una función creada para ello. A la función le debemos especificar las columnas del fichero CSV donde están la ascensión recta y declinación, además del nombre del fichero. Dicha función almacena en un objeto la matriz que contiene ambas coordenadas.

```
1 import GAIAFile as GDB
2 DBobject = GDB.GAIAFile("result.csv", 1, 2)
```

Una vez tenemos el objeto con la matriz, hacemos la llamada a la función que crea la imagen sintética. Dicha función tiene como objetivo exportar dos imágenes, la sintética y la final sin fuentes, partiendo de un archivo FITS, el nombre del archivo a exportar, la matriz de posiciones y un tamaño de imagen.

```
1 import SyntheticImage as VI
2 VI.createSyntheticImage("header.fit", "negative.fit",
3             DBobject.database, sizeX=1024, sizeY=1024)
```

El proceso comienza leyendo el FITS, es decir, la imagen original. De dicho fichero se obtienen los datos (la imagen) y la cabecera (utilizada posteriormente para crear el nuevo fichero). Para comenzar el proceso se crea una matriz del tamaño especificado en los parámetros de la función más un "offset", esta matriz contendrá las gaussianas que vamos creando. Dicho "offset", por los 4 lados de la imagen, se utiliza para conseguir pegar gaussianas (las cuales se crean en una región cuadrada) justo en los bordes, pudiendo así exceder los límites sin problemas con el tratamiento de matrices. Obviamente, en el proceso final, dichos "offset" serán eliminados, quedándonos solo con la zona deseada.

Previa etapa de cálculo de gaussianas, se hace el cambio de coordenadas usando la función descrita en la sección "Cambio de coordenadas: sky2xy" del capítulo "Desarrollo de funciones para manejo de librerías y ficheros astronómicos" de este documento.

El proceso de creación de gaussianas se realiza en bucle, utilizando cada una de las filas del dataset que recibimos por parámetro. Si la posición en píxeles cae dentro del tamaño de la región para crear la imagen (puede que no ocurra), se hace la llamada al algoritmo genético que aproxima la gaussiana y se crea la misma para luego situarla en su correspondiente posición.

Tras todos los cálculos, para todas las fuentes, se recorta el "offset" comentado anteriormente, previo cálculo de la imagen final. Para el cálculo de la imagen final se hace uso de una función que busca valores en la imagen sintética menores que un valor muy cercano a cero , es decir, valores negativos creados por las gaussianas. Una vez encontrado un valor, se realiza un cálculo del valor de cielo en una zona alrededor del píxel en cuestión, de forma que luego se crea un número aleatorio en un rango en torno al valor y se sitúa en esa posición de la matriz.

Finalmente, sumamos una valor fijo (32.768) a la imagen resultante. Esta cantidad se codifica como 0 en los archivos FITS y debe ser sumada para la correcta codificación del fichero. En la lectura de cualquier fichero FITS en Python este valor ha sido restado y es por ello que debe devolverse al rango original de codificación.

Unas vez, tenemos correctamente codificadas las imágenes, se abrirá en un nuevo objeto la imagen original y se sustituirá la parte de datos por los calculados en el algoritmo, para posteriormente crear el nuevo fichero FITS.

```
1 def createFits(data, headername, name='finalImage.fits'):
2     inhdulist = pyfits.open(headername)
3     inhdulist[0].data = data
4     hdulist = pyfits.HDUList(inhdulist)
5     if os.path.exists(name): os.remove(name)
6     hdulist.writeto(name)
```

6.2. Paralelización en Spark

Una vez desarrollada la versión iterativa, es posible comentar brevemente cuáles son los pasos a seguir para una implementación de forma paralela en Spark, ya que en futuras versiones del algoritmo es de vital importancia su ejecución en paralelo debido a la cantidad enorme (2 Terabytes) de datos de los que disponemos.

Actualmente, el cálculo de una imagen de 21 MB en versión iterativa sobre un ordenador con procesador Intel Core i3, tarda sobre los 240 segundos. Esto significa que procesar 2 Terabytes de información nos llevaría alrededor de 132 días. Realizando un cálculo informal de una posible mejora utilizando un cluster de computación con 10 nodos, veríamos reducida la cantidad de 132 a 13 días.

Partiendo del algoritmo iterativo comentado en la sección anterior se puede establecer cuales son las etapas más susceptibles de paralelizar debido a su coste computacional. Sin duda, la parte más costosa computacionalmente, es el cálculo de las gaussianas, por lo que sería el primer punto a paralelizar.

Para realizar el proceso de paralelización se debe definir de qué manera se van a distribuir los nodos del cluster, en este caso, la paralelización se lanzaría por cada imagen, utilizando los nodos para ir calculando las gaussianas que necesitamos. Es por ello que tendríamos que realizar un cálculo previo para dividir en particiones todas las fuentes entre la potencia total disponible de la forma más óptima posible.

A este proceso se le llama mapReduce, que es un modelo de computación que permite la programación de sistemas paralelos donde se diferencian dos etapas; etapa de división del trabajo en los nodos (MAP) y etapa de unión (Reduce) de resultados calculados en los nodos.

En el caso del algoritmo para este trabajo, la etapa Reduce es sencilla, solo necesitamos conocer o los parámetros óptimo de cada gaussiana y luego generarla, o directamente la gaussiana en una matriz. Puede ocurrir que el desarrollo de esta etapa se complique más si pretendemos hacer sistemas que calculen parcialmente soluciones en cada nodo o se dividan el trabajo.

Capítulo 7

Resultados

7.1. Resultados

La imagen original de la que disponemos contiene 597 fuentes detectadas por SExtractor y el objetivo es que se vean reducidas en la medida de lo posible. De las 597 fuentes detectadas, muchas pertenecen a falsos positivos, como pueden ser pixeles muertos, rayos cósmicos, ruido del fondo del cielo o estrellas de bajo brillo que no aparecen en GAIA.

Partiendo de esa premisa y sabiendo que en dicha zona tenemos 322 fuentes conocidas, la imagen resultante del algoritmo obtiene entorno a las 330 fuentes dependiendo de la ejecución, sobre 50 menos que el objetivo óptimo. En la Figura 7.1 se puede apreciar, en la imagen izquierda, como queda la región tras el uso del algoritmo sobre la de la derecha, apareciendo solo las fuentes más débiles y desapareciendo las estrellas.

Este resultado es muy prometedor ya que bastantes estrellas no tienen un posicionamiento exacto entre la imagen y GAIA, por tanto, la generación de la gaussiana aún deja ruido que es detectado como fuente. Uno de los objetivos principales por tanto es mejorar el previo cálculo de la posición de las fuentes, para poder eliminar más eficazmente la región.

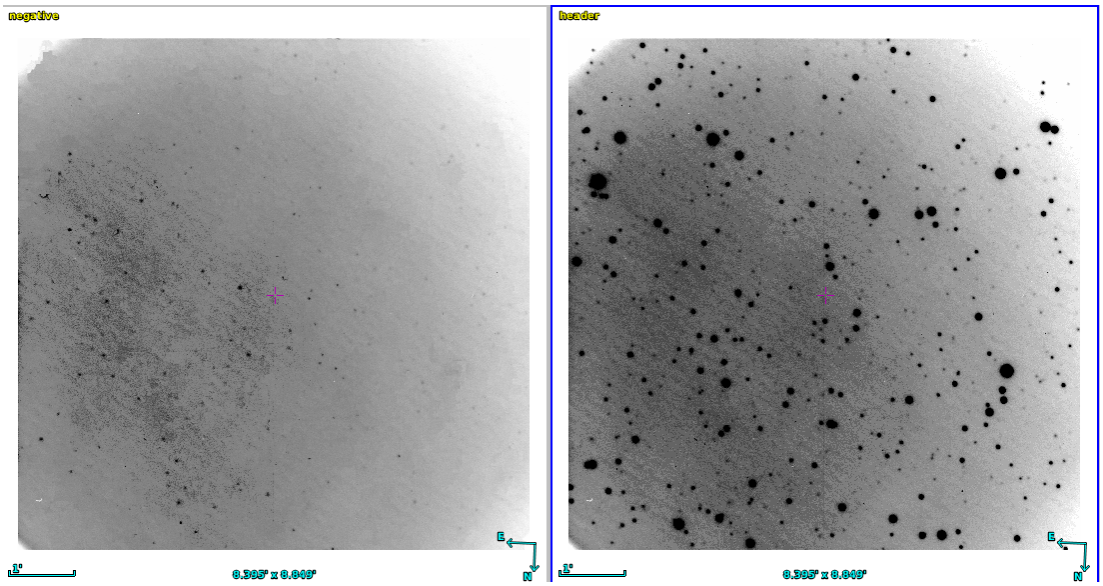

Figura 7.1: Imagen final limpia de fuentes conocidas

Si usamos el algoritmo sobre dos imágenes pasadas 3 horas y 10 minutos podemos observar como una fuente de las no conocidas se ha movido ligeramente en la Figura 7.2, lo cual indica un verdadero candidato a TNO. Esta imagen ha sido seleccionada por contener dicho objeto y poder mostrar cual es el objetivo futuro en la búsqueda automática de estos objetos.

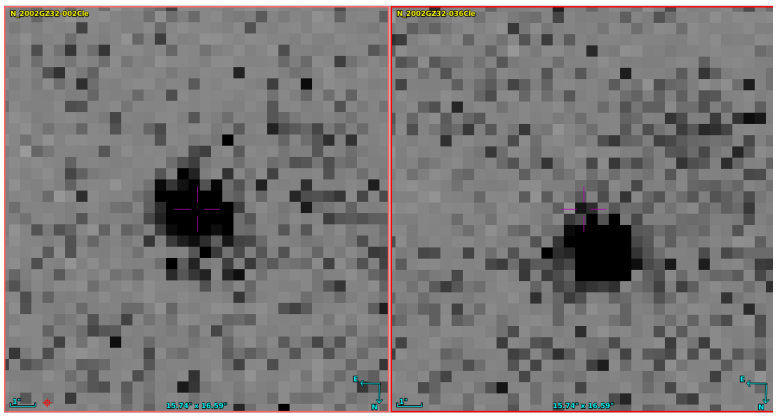

Figura 7.2: Candidato a TNO

Capítulo 8

Conclusiones

Contenidos

En este último capítulo se ofrece un resumen de las conclusiones alcanzadas en los capítulos anteriores. Además, se plantearan propuestas de trabajo futuro que continúen en la línea de los objetivos marcados en este proyecto.

8.1. Conclusiones

El algoritmo implementado constituye un buen punto de partida para continuar con el desarrollo de un sistema automatizado para la detección de TNOs. La idea de implementar un algoritmo genético para aproximar las gaussianas de las estrellas cumplió con las expectativas y puede ser mejorado para adaptarse aún mejor en imágenes más complejas en un futuro.

Usando la imagen final, en periodos de tiempo diferentes, somos capaces de observar si alguna fuente no conocida se mueve de posición, lo cual significa que estamos ante un verdadero candidato a TNO. Usando esta técnica podemos realizar el cálculo de múltiples imágenes y buscar las fuentes que se mueven en ellas.

Otro punto clave es la "limpieza" de imágenes de falsos positivos y configuración del SExtractor, el cual en su configuración por defecto aún sigue detectando un gran número de candidatos falsos y proporciona un número de fuentes muy grande.

La imagen final obtenida es muy interesante, gracias a la mejora de relleno con cielo de fondo obtenemos un resultado muy llamativo e intuitivo de lo que está ocurriendo sobre la imagen. A partir de ella sería mucho más fácil la detección o el análisis de cualquier objeto fuera de los conocidos.

8.2. Trabajo futuro

A partir de los resultados obtenidos se abren nuevas vías tanto de mejora como investigación.

Uno de los principales puntos de mejora es el posicionamiento de fuentes a partir de GAIA, el cual aún comete pequeños errores que producen que el cálculo de las gaussianas esten un poco desplazado, generando pequeños errores.

Otro punto de mejora posible es la limpieza de las zonas donde hay estrellas muy brillantes, las cuales suponen un problema a la hora de calcular los valores de fondo en tiempo de ejecución.

Una vez corregidos esos pequeños detalles se abren vías como puede ser el cálculo automático de TNOs a partir de las imágenes obtenidas y la versión paralelizada de este. Para ello primero habría que configurar correctamente SExtractor para la detección solo de fuentes, puesto que las imágenes contienen otros defectos que son detectados como tales.

El objetivo final sería el descubrimiento de TNOs en zonas muy pobladas de estrellas, las cuales son las más complicadas de tratar y limpiar.

Bibliografía

[1] P. G. Benavidez A. Dell'Oro, A. Campo Bagatin and R. A. Aleman. Statistics of encounters in the trans-neptunian region, AyA 558, A95 (2013), DOI: https://doi.org/10.1051/0004-6361/201321461.

[2] D. Ashlock. Evolutionary computation for modeling and optimization, Springer, ISBN 0-387-22196-4.

[3] Goldberg D.E. Booker, L.B. and J.H. Holland. Classifier systems and genetic algorithms, Artificial Intelligence, 40, 1989, pp. 235-282.

[4] Elisa de Castro. Coordenadas celestes, Sociedad española de astronomía. https://www.sea-astronomia.es/drupal/node/159. 6/09/2016.

[5] The Astropy Developers. Convolution and filtering, Astropy.docs. http://docs.astropy.org/en/stable/convolution/.

[6] S. Forrest. Genetic algorithms: Principles of natural selection applied to computation, Science, Vol. 261, No. 5123, August 13, 1993, pp. 872-878.

[7] S. Forrest. Genetic algorithms: Principles of natural selection applied to computation, Science, Vol. 261, No. 5123, August 13, 1993, pp. 872-878.

[8] D. Hestroffer. Preparing gaia for the solar system, EAS Publications Series, 2 (2002) 359-364, DOI: https://doi.org/10.1051/eas:2002036.

[9] J.H. Holland. Adaptation in natural and artificial systems, University of Michigan Press, 1975, 211 p.

[10] P. Lacerda. The sizes of kuiper belt objets, SPICA Workshop 2009, DOI: https://doi.org/10.1051/spica/200902004.

[11] Javier Licandro. Objeto transneptuniano, Sociedad española de astronomía. http://www.sea-astronomia.es/drupal/node/283. 6/09/2016.

[12] T. Das A. Dave J. Ma M.McCauley M. J. Franklin S. Shenker I. Stoica M. Zaharia, M. Chowdhury. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, University of California, Berkeley. NSDI 2012, April 2012.

[13] Jameel Mohammed. Is apache spark going to replace hadoop?, APTUZ. Published: 20, March 2015 07:33. http://aptuz.com/blog/is-apache-spark-going-to-replace-hadoop/.

[14] European Southern Observatory. Basic image processing, http://www.eso.org/ ohainaut/ccd/. 6/09/2016.

[15] University of Nebraska-Lincoln. Celestial equatorial coordinate system, http://astro.unl.edu/naap/motion1. 6/09/2016.

[16] M.A.C. Perryman. Gaia: An introduction to the project, EAS Publications Series, 2 (2002) 3-26, DOI: https://doi.org/10.1051/eas:2002001.

[17] Urmila Pol. Big data analysis: Comparision of hadoop mapreduce and apache spark, International Journal of Engineering Science and Computing. 06-2016, pp: 6389-6391.

[18] A. Walker R. Fisher, S. Perkins and E. Wolfart. Gaussian smoothing, University of Edinburgh. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[19] Herbert Raab. Detecting and measuring faint point sources with a ccd, Astronomical Society of Linz, Sternwarteweg 5, A-4020. Linz, Austria.

[20] C. A. Trujillo. Future surveys of the kuiper belt, University of Arizona Press, Tucson, pp. p.573-585.