TFG: Detección semi-automática de Sistemas de Estrellas Dobles

Trabajo de Fin de Grado

David Antuña Rodríguez
Daniel Gutiérrez Gertrúdix
Javier Carrión García

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2017/2018

Documento maquetado con TEXIS v.1.0+.

TFG: Detección semi-automática de Sistemas de Estrellas Dobles

Trabajo de Fin de Grado

Dirigida por el Doctor
Rafael Caballero Roldán

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2017/2018

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, nosotros queremos agradecer al Word de Microsoft el habernos forzado a utilizar LaTeX. Cualquiera que haya intentado escribir un documento de más de 150 páginas con esta aplicación entenderá a qué nos referimos. Y lo decimos porque nuestra andadura con LaTeX comenzó, precisamente, después de escribir un documento de algo más de 200 páginas. Una vez terminado decidimos que nunca más pasaríamos por ahí. Y entonces caímos en LaTeX.

Es muy posible que hubíeramos llegado al mismo sitio de todas formas, ya que en el mundo académico a la hora de escribir artículos y contribuciones a congresos lo más extendido es LaTeX. Sin embargo, también es cierto que cuando intentas escribir un documento grande en LaTeX por tu cuenta y riesgo sin un enlace del tipo "*Author instructions*", se hace cuesta arriba, pues uno no sabe por donde empezar.

Y ahí es donde debemos agradecer tanto a Pablo Gervás como a Miguel Palomino su ayuda. El primero nos ofreció el código fuente de una programación docente que había hecho unos años atrás y que nos sirvió de inspiración (por ejemplo, el fichero `guionado.tex` de TeX!S tiene una estructura casi exacta a la suya e incluso puede que el nombre sea el mismo). El segundo nos dejó husmear en el código fuente de su propia tesis donde, además de otras cosas más interesantes pero menos curiosas, descubrimos que aún hay gente que escribe los acentos españoles con el `\'{\i}`.

No podemos tampoco olvidar a los numerosos autores de los libros y tutoriales de LaTeX que no sólo permiten descargar esos manuales sin coste adicional, sino que también dejan disponible el código fuente. Estamos pensando en Tobias Oetiker, Hubert Partl, Irene Hyna y Elisabeth Schlegl, autores del famoso "The Not So Short Introduction to LaTeX 2_ε" y en Tomás

Bautista, autor de la traducción al español. De ellos es, entre otras muchas cosas, el entorno `example` utilizado en algunos momentos en este manual.

También estamos en deuda con Joaquín Ataz López, autor del libro "Creación de ficheros LaTeX con GNU Emacs". Gracias a él dejamos de lado a WinEdt y a Kile, los editores que por entonces utilizábamos en entornos Windows y Linux respectivamente, y nos pasamos a emacs. El tiempo de escritura que nos ahorramos por no mover las manos del teclado para desplazar el cursor o por no tener que escribir `\emph` una y otra vez se lo debemos a él; nuestro ocio y vida social se lo agradecen.

Por último, gracias a toda esa gente creadora de manuales, tutoriales, documentación de paquetes o respuestas en foros que hemos utilizado y seguiremos utilizando en nuestro quehacer como usuarios de LaTeX. Sabéis un montón.

Y para terminar, a Donal Knuth, Leslie Lamport y todos los que hacen y han hecho posible que hoy puedas estar leyendo estas líneas.

Resumen

...

...

...

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Introducción

...

Notas bibliográficas

...

Capítulo 2

Recolección de datos

2.1. Aladin

Aladin Sky Atlas es un programa interactivo desarrollado por el **CDS**, Centro de Datos Astronómicos de Estrasburgo, que permite acceder recursos astronómicos digitalizados.

Esta herramienta no solo permite visualizar imagenes de cuerpos celestes a lo largo de los años, también permite realizar una superposicion de las mismas para facilitar su comparación.

2.2. Obtención de imagenes

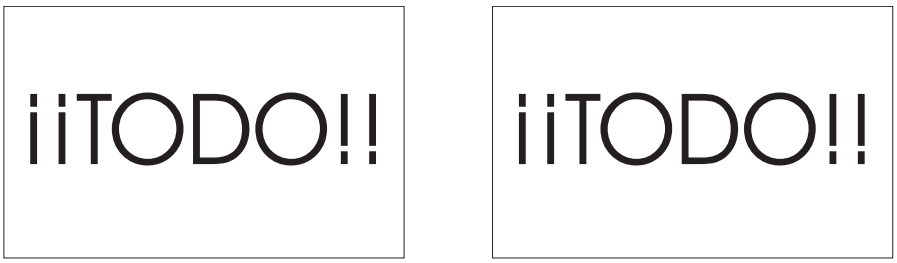
Como se describe la sección 2.1 es posible obtener imagenes que muestren la evolucion de los cuerpos celestes en coordenadas concretas con años de diferencia.

Como se puede apreciar, figura 2.2, Aladin no solo superpone las imagenes, también las rota para tratar de encajarlas y colorea cada imagen de un tono de modo que las estrellas pueden ser rojas o azules, dependiendo de la imagen en que aparezcan, si el cuerpo coincide, no se ha movido, será blanca.

Para facilitar el proceso de descarga aladin permite emplear scripts en combinación con ficheros de parametros. Puesto que no existe ninguna fuente que tenga fotos de todas las coordenadas celestes son necesarios dos scripts que emplean las siguientes fuentes.

- **POSSI** y **POSSII** para descargar imagenes de coordenadas positivas.
- **POSSI** y **SERC** para las negativas.

Además de ejecutar la superposición el script hace zoom en la coordenada proporcionada como parámetro y almacena la imagen resultante.



(a) (b)

Figura 2.1: Imagenes descargadas de las bases astronómicas

Figura 2.2: Superposición de las imagenes

2.3. WDS

La obtención de recursos de trabajo ya ha sido resuelta en la seccion 2.2, sin embargo, es necesario obtener un listado de sistemas de estrellas dobles ya reconocidas que poder analizar con el fin de semi-automatizar su reconocimiento.

WDS, Washington Double Star Catalog, es un catalogo mantenido por el Observatorio Naval de los Estados Unidos que recopila información sobre sistemas de estrellas múltiples.

El problema es que el catalogo almacena gran cantidad de estrellas, y nosotros queremos casos de sistemas de estrellas dobles de movimiento rápido. Con el fin de obtener la información que nos interesa hemos creado un programa que filtra dichos datos basandose en los siguientes criterios:

- La última vez que se vió el sistema debe ser como mínimo 1975, esto asegura que en los catalogos existiran almenos dos imagenes, una relativamente actual y una de los años 50.
- La magnitud de las estrellas que componen el sistema deben ser a lo sumo 19, de este modo las estrellas seran apreciables a simple vista.

- La separación debe pertenecer al rango entre 2 y 180, las estrellas con valores mayores están demasiado alejadas y desvirtuarian los resultados.
- El desplazamiento ha de ser superior a 60 para que el movimiento sea apreciable.

Las coordenadas de todos los sistemas que sigan dichos criterios serán almacenadas en un fichero que puede usarse como parametro de los scripts descritos en la seccion 2.2.

Notas bibliográficas

...

Capítulo 3

Procesamiento

3.1. Recoloreado

A simple vista las imagenes obtenidas en la sección 2.2 están coloreadas y nos permiten apreciar las diferencias fácilmente. El problema surge cuando queremos que el programa haga las misma extrapolaciones que hace nuestro cerebro.

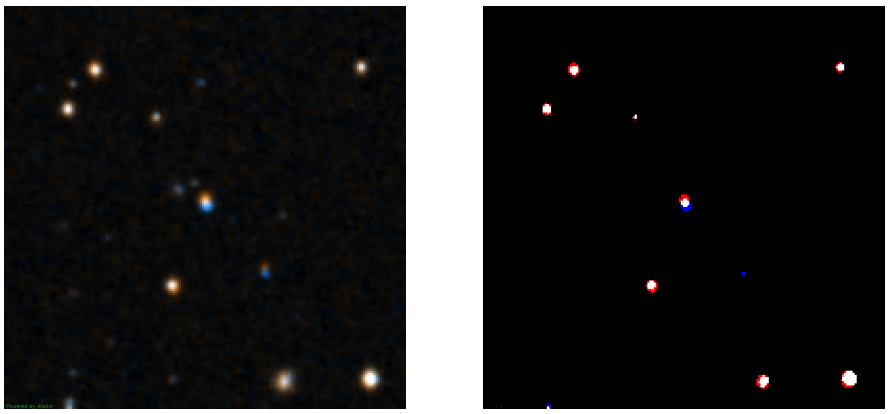
Con el fin de facilitar la comparación de los pixels la imagen pasa por un proceso de recoloreado que transforma la imagen pixel a pixel.

Los colores que reconoce este proceso se definen mediante un diccionario que almacena el nombre y composición del mismo. Esto facilita la modificación de los grupos reconocidos de modo que un cambio en el coloreado inicial de las imagenes no supone un problema.

El problema de este proceso es definir cómo ha de interpretar la tonalidad del pixel, para atajarlo hemos empleado la distancia Manhanttan. Se trata de medir la distancia entre la composición del pixel y la del grupo, asumiendo que ambas tonalidades son RGB y que están almacenadas en triplas, de la forma (R, G, B), tan solo hemos de restar cada componente con su homóloga y sumar los resultados para obtener un valor. El grupo que resulte con menor valor será aquel al que pertenezca el pixel.

Tal y como muestra la imágen 3.1b hay información que se pierde durante el proceso debido a que las estrellas están muy apagadas haciendo que la distancia al negro sea menor que al azul o rojo. Para poder atajar este problema se realiza una comprobación posterior, si el grupo asignado es negro pero la distancia al azul o al rojo es inferior a un cierto umbral se reasigna el grupo del pixel, imagen

Cuando se ha obtenido el grupo del pixel se pinta del color de grupo en un canvas, una vez se han procesado todos los pixel se almacena el resultado en un fichero png.



(a) (b)

Figura 3.1: Recoloreado usando la distancia Manhattan

3.2. Comprobación

Aún cuando el detector acepta un sistema de estrellas dobles existe un problema, puede no tratarse de un sistema nuevo. Para solventar esto existe una fase de comprobación que se puede activar para todos los sistemas aceptados.

Si se ha solicitado esta fase se realiza un primer paso que consiste en conectarse a WDS y descargar los datos de los sistemas conocidos, este paso solo se hace una vez por ejecución, al comienzo, para no saturar los servidores. El hecho de emplear los mismos datos durante una ejecución completa no supone un problema puesto que la base de datos no se altera habitualmente.

Cuando un sistema es aceptado esta fase toma sus coordenadas y busca coincidencias en todas las entradas descargadas, si no hay ninguna se ignora el sistema detectado. Si por el contrario existen datos asociados al sistema se almacenan en un archivo json junto con los datos proporcionados por el detector y el calculo del error entre ambos.

Si bien esta fase es una primera comprobación sus resultados no son definitivos, en el caso del no, puesto que se comprueban las coordenadas centrales de la imagen. Si la estrella no se encuentra en el centro es imposible para este programa determinar si el sistema ya se conocía o no.

Notas bibliográficas

...

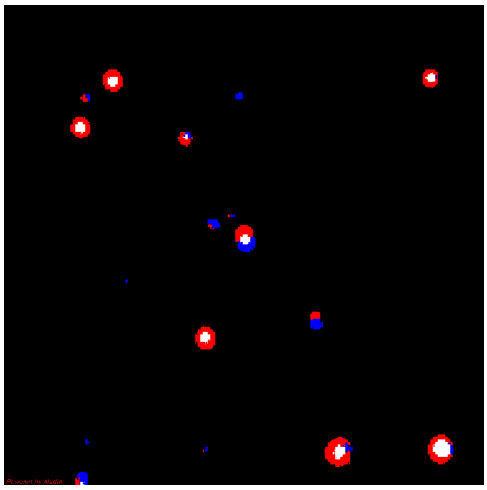


Figura 3.2: Recoloreado de la imagen 3.1a con un umbral de 150

Capítulo 4

Parametrización del detector

4.1. Introducción

...

Notas bibliográficas

...

Capítulo 5

Workflow

5.1. Introducción

En los capitulos 3 y 4 se describe diferentes procesos por los que una fotografía ha de pasar para poder detectar sistemas de estrellas dobles, cada una de estas etapas viene definida por un numero arbitrario de entradas y salidas y un evento que al abortar la ejecución permita al programa salir de forma segura.

Cada etapa por tanto puede ser configurada por sepado y lanzada puesto que para conectarlas solo es necesario que la salida de una etapa sea la entrada de la siguiente, de esta forma todas las etapas pueden trabajar de forma simultanea. Sin embargo, creemos que esta tarea puede simplificarse mediante la creacion de un controlador que se encarge de lanzar las etapas en diferentes hilos, para que sigan ejecutandose a la vez, y pararlas cuando el usuario lo requiera.

5.2. Estructura

La idea es generar una estructura que permita al usuario definir el orden de las etapas de manera sencilla, y una vez definidas las lance en diferentes hilos. Una posible representación del workflow sería un grafo acíclico dirigido, es importante que sea acíclico puesto que de otro modo la información nunca llegaría a un estado definitivo en el cual se acepte o rechace.

Podemos entonces definir cada fase como una entrada en un diccionario en la cual podemos almacenar la entrada y la salida, de este modo conectar una etapa con otra tan solo implica igualar la entrada de una a la salida de la otra. A la hora de lanzar el hilo es necesario sabar a que función llamar para lo cual nos encontramos con dos opciones: llamar siempre a una funcion con un nombre determinado o simplemente incluir otro dato en el diccionario. En este caso se ha optado por que además de lo que ya guardaba se almacene

la función a la que se debe llamar, esta opción da libertad a la hora de crear los pasos además de facilitar la posterior creación de los hilos.

Existe un último problema, algunas etapas del workflow pueden tener más de una entrada o salida por lo que en el diccionario se almacenara un array de entradas y otro de salidas, debido a esto también será necesario especificar el numero de entrada o salida a la que se hace referencia al conectar los pasos.

Una entrada del diccionario tendría por tanto el siguiente formato.

```
flow['step'] = {
  'input': [
    flow['other']['output'][0],
    settings.directory
  ],
  'output': [
    settings.directory2
  ],
  'callback': step.run,
}
```

Además se hace uso de una etapa *dummy* que tan solo define la entrada del primer paso en su salida. Una vez creadas las entradas en el diccionario para todas las etapas solo será necesario ejecutar el workflow.

5.3. Etapas

Si bien es cierto que el workflow esta pensado para dar libertad a la hora de implementar las etapas tiene dos requisitos que se han de cumplir. El primero es a la hora de definir el metodo que se utilizara como callback en el diccionario, sus parametros deben estar ordenados de modo que reciba primero todas las entradas, luego todas las salidas y por ultimo un evento de las clase *threading*, este orden ha de ser tenido en cuenta a la hora de crear la entrada del diccionario descrita en la sección 5.2. En segundo lugar se debe asegurar que la etapa se mantendrá a la espera de nuevos datos que procesar hasta que se active el evento que se recibe por parámetro y que una vez recibido no se abortara la ejecución del paso hasta que sea seguro.

Para poder saber que está pasando el workflow también monta un sistema de log, utilizando la libreria *logging* de python, al cual las etapas pueden acceder para escribir mensajes de error o información de debug que le pueda ser de utilidad al usuario para conocer que ha ocurrido.

Debido a que las etapas se encuentran en continua ejecución podría darse el caso en el que los mismos datos se procesen varias veces, para prevenir esto las etapas que hemos implementado mantienen un log de historia que solo modifican ellas en el cual almacenan el identificador de los datos al acabar de procesarlos, una de las primeras comprobaciones al recoger datos de la

entrada es comprobar si el id se encuentra en la historia del paso, de ser así los datos no se procesan.

5.4. Uso

Para utilizar el workflow son necesarios tres pasos.

En primer lugar modificar el fichero de configuración, *settings.py*, en el cual se encuentran almacenados todos los directorios de los pasos que hemos creado y los parametros de procesamiento que pueden modificarse de cada etapa.

Ir al workflow, *workflow.py*, y editar la función **define_flow()** para crear el diccionario de etapas tal y como se ha explicado en la sección 5.2.

Por último, ejecutar el workflow. Para parar el procesamiento basta con enviar una interrupción, Ctrl+C, en la consola en la cual se haya lanzado.

Notas bibliográficas

...

Capítulo 6

Problemas

6.1. Introducción

...

Notas bibliográficas

...

Parte I

Apéndices

Apéndice A

Así se hizo...

...

...

Resumen: ...

A.1. Introducción

...

Bibliografía

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes