Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Nombre del Centro

Trabajo Fin de Grado

AUTOMATIZACIÓN DE CALIBRACIÓN DE IMÁGENES ASTRONÓMICAS DE LARGAS SERIES TEMPORALES HETEROGÉNEAS

Alumno: Álvaro Miguel Anguita Palomino

Tutor: Prof. D. Manuel José Lucena López

Dpto: Departamento de Informática

Septiembre, 2021

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Manuel José Lucena López, tutor del Proyecto Fin de Carrera titulado: Automatización de calibración de imágenes astronómicas de largas series temporales heterogéneas, que presenta Álvaro Miguel Anguita Palomino, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2021

El alumno:

Firmado por ANGUITA PALOMINO ALVARO MIGUEL - ***7085** el día 06/09/2021 con un certificado emitido por AC FNMT Usuarios

Álvaro Miguel Anguita Palomino

Los tutores:

Firmado digitalmente por LUCENA LOPEZ, MANUEL JOSE.
Fecha: 06.09.2021 - 13:34:07.

Manuel José Lucena López

Agradecimientos

Dado que se acaba esta etapa en mi vida, me gustaría agradecer a mis padres y mis hermanas por el gran esfuerzo económico realizado, el haberme soportado en épocas de exámenes. Además, agradecerles la confianza que depositaron en mi cuando otras personas no lo hicieron y gracias también por animarme a seguir y no tirar la toalla.

También me gustaría agradecer el trabajo y la dedicación de algunos profesores, los cuales se implican con los alumnos y pierden tiempo de sus vacaciones o incluso su propio tiempo libre los findes de semana para resolver dudas y para ayudar en la manera de lo posible.

Por último, me gustaría agradecerle el apoyo que me ha dado siempre a Almudena, una de las personas más importantes que como he dicho siempre me ha apoyado y siempre ha confiado en que yo podía acabar este grado.

Indice

Indice de Figuras:

Índice de Tablas:

1. Introducción

En el presente documento se va proporcionar una herramienta, que a partir de una serie temporal de imágenes astronómicas y con distintas características tanto de la instrumentación como de software de adquisición, sea capaz de calibrarlas de manera masiva y de forma automática y como resultado devuelva las imágenes organizadas por objeto astronómico, filtro y telescopio. Las imágenes utilizadas para la realización de este proyecto han sido proporcionadas por el grupo de Cuerpos Menores del Instituto de Astrofísica de Andalucía (IAA), centro Severo Ochoa perteneciente al Consejo Superior de Investigaciones Científicas (CSIC).

Este documento seguirá un orden cronológico del trabajo realizado, exponiendo:

- Motivación
- Objetivos
- Historia
- Descripción del problema
- Estándar FITS y calibración de imágenes astronómicas
- Ingeniería de requisitos
- Estudio del lenguaje y librerías a utilizar
- El diseño del algoritmo y problemas encontrados

2. Motivación

Este proyecto surge con el objetivo de intentar realizar la calibración de imágenes astronómicas de grandes series de datos tomados con diferentes tecnologías y estándares a lo largo de los últimos años, de manera que permita automatizar una tarea que debido a la heterogeneidad de los telescopios, instrumentos y estándares no podía realizarse de manera masiva y rápida.

Como se mostrará a continuación, es posible solucionar este problema gracias a las nuevas librerías y métodos que existen en la actualidad.

3. Objetivos del proyecto

Los objetivos que se plantean para realizar el proyecto son los siguientes:

- Realizar la calibración de imágenes astronómicas
- Conocer las diferentes tecnologías y estándares que han ido surgiendo en los últimos años
- Diseñar e implementar una aplicación que para automatizar el proceso de calibrado de las imágenes.

4. Historia

Una de las cosas que más llama la atención desde hace algunos años, son las imágenes espaciales. Estas imágenes se utilizan en diferentes proyectos de todo el mundo para conocer más el universo. Estas imágenes se realizan a través de telescopios.

Uno de los centros que utilizan estas imágenes en sus investigaciones es el Instituto de Astrofísica de Andalucía (IAA) [1] del Consejo Superior de Investigaciones Científicas (CSIC).

La misión de este instituto es la de profundizar en el conocimiento del cosmos, además de acercarlo a la sociedad realizando investigaciones en el campo de la Astrofísica y la Ciencia Espacial de vanguardia.

Gracias a esta profundización e investigación, se fomenta el desarrollo tecnológico mediante la construcción de nueva instrumentación y diseminando la investigación entre la comunidad científica y el público en general por medio de actividades divulgadoras.

"*El IAA-CSIC se funda en 1975 con la intención de crear un centro de excelencia en investigación en Astrofísica, Ciencias del Espacio y tecnologías asociadas a dichas actividades. Desde entonces, el IAA se ha afianzado como centro de referencia nacional e internacional en investigación en Astrofísica, siendo hoy en día uno de los mayores centros de investigación del CSIC con casi 200 integrantes.*

Por su producción científica, es el segundo centro español en el área de Astrofísica y el séptimo entre los centros del CSIC en cualquiera de sus áreas de investigación.

En el IAA, se investiga en todas y cada una de las principales áreas de la Astrofísica moderna, desde la gravedad cuántica al sistema solar, pasando por la evolución de las galaxias, la cosmología, los componentes de nuestra Galaxia y los planetas extrasolares.

Esta investigación se apoya en los tres pilares fundamentales de la Astrofísica moderna: la observación de fenómenos astrofísicos con los medios más sofisticados, el desarrollo de nueva instrumentación y el estudio teórico y desarrollo de simulaciones numéricas.

Esta amplitud de conocimientos y técnicas, la interconexión entre ellos, hace al IAA destacar entre centros de investigación Astrofísica nacionales e internacionales. Pocos centros cubren tantos campos de la Astrofísica y Ciencia del Espacio y están involucrados en observaciones desde Tierra y desde observatorios y misiones espaciales en los que haya realizado desarrollos tecnológicos. Por ello, no es sorprendente que el IAA lidere grandes proyectos internacionales de investigación Astrofísica" (IAA-CSIC, 2021).

Uno de los grupos de dicho centro es el de Cuerpos Menores, que pertenece al departamento de Sistema Solar, y basa su ciencia en el estudio de cuerpos menores de nuestro Sistema Solar, especialmente Objetos Transneptunianos y Centauros.

El grupo de Cuerpos Menores está muy consolidado desde hace muchos años, financiado con varios proyectos nacionales e internacionales y en continua colaboración con instituciones científicas de varias partes del mundo.

Parte de la investigación que realiza este grupo, se obtiene de imágenes astronómicas obtenidas con telescopios de distintos observatorios del planeta. Esto ha permitido analizar y publicar a lo largo de los últimos 20 años decenas de artículos científicos en revistas de alto impacto.

[2][3] Este tipo de investigaciones que lleva a cabo el IAA se realizan utilizando distinto instrumental distribuido por todo el planeta, con distinta tecnología donde los telescopios más usados se encuentran en el Observatorio de Sierra Nevada (OSN) y

el Observatorio Astronómico Hispano-Alemán de Calar Alto. Estos dos observatorios se encuentran situados en la estación de esquí de Sierra Nevada y en la Sierra de Los Filabres respectivamente.

El OSN es operado y gestionado únicamente por el IAA-CSIC, mientras que el Observatorio de Calar Alto es gestionado por CSIC a través del IAA y la Junta de Andalucía que ha entrado en el consorcio hace dos años sustituyendo al DLR alemán, el cual ha sido socio del observatorio desde sus inicios hasta ahora.

En estos observatorios cuentan con diferentes telescopios: el OSN tiene como telescopios principales, uno de 0.90-m y otro de 1.50-m de apertura. Los tres telescopios principales del observatorio de Calar Alto, son aquellos de aperturas 1.23m, 2.2m y 3.5m.

5. Descripción del problema

Los datos se han ido analizando a lo largo de los años, pero la tecnología actual permite reanalizarlos con técnicas actuales aplicadas tanto a las últimas imágenes como a las que se tomaron hace 20 años. Para ello se necesita calibrar todas las imágenes de manera que puedan analizarse de nuevo usando técnicas y catálogos actuales.

Es tan variado el número de observatorios involucrados y la variación de tecnología en los últimos 20 años, que se necesita automatizar el proceso de calibración de imágenes astronómicas al máximo y búsqueda de los objetos celestes observados en ellas, para poder analizarlas de nuevo con técnicas actuales, ya que de otra manera se tardaría muchísimo en calibrar todos los objetos.

Los detectores utilizados en astronomía visual, convierten los fotones procedentes del cielo y amplificados por la óptica del telescopio, en señales eléctricas que son almacenadas de forma digital. Este proceso, lleva intrínsecos efectos de ruido que provocan falsas lecturas las cuales hacen que la imagen ideal que debería obtenerse, se convierta en una imagen real, alejada de la original que hay que corregir a posteriori con datos de calibración.

5.1. Actualidad

El proceso de observación de un objeto astronómico, no sólo consta de la imagen que se toma del cielo, sino que comprende otra serie de ficheros de calibración que una vez aplicados a esa imagen, permiten limpiarla de artefactos debido a ruido electrónico y corriente de oscuridad, defectos o suciedad en los filtros o en los chip, así como corregirla de gradientes que harían que las imágenes no sirvieran para analizarlas al nivel que necesitamos, corregir fallos en píxeles o líneas de píxeles de los detectores.

Por distintos motivos, no de todas las imágenes que se han ido ya analizando en el pasado y presente se guarda una copia de esos ficheros calibrados, así que lo ideal es volver a calibrar todas las imágenes que se tienen.

Puesto que el número de imágenes que se tienen desde hace 20 años es tan grande, así como el número de observatorios involucrados y dado que en la actualidad hay nuevos métodos que de forma masiva permiten analizar grupos de imágenes, se necesita algún método que, de forma rápida, pueda calibrar las distintas imágenes que se tienen, ya que a lo largo de los años esta calibración se hacía de manera más local, por ejemplo basándose en las imágenes tomadas solo por un año o simplemente basándose en un objeto celeste.

Por último, también nos encontramos con el problema de que, a lo largo del tiempo, las cabeceras de las imágenes han cambiado, las versiones del estándar de almacenamiento FITS han ido cambiando y en algunos países se han aplicado otros estándares y por lo tanto hay una variedad que hay que salvar.

6. Estándar FITS

[4] Los telescopios toman imágenes del Universo, y puesto que es necesario almacenar junto a ellas numerosos y variados tipos de datos, no pueden ser simples imágenes JPG o PNG. Por ello, es indispensable utilizar un formato de imagen adecuado, como es el formato FIT (Flexible Image Transport System), que emplea la extensión .fit y es uno de los formatos más utilizados en el campo de la, yendo mucho más allá que un simple JPEG o GIF.

Avalado por la NASA y por la Unión Astronómica Internacional, el formato de archivo FIT, además de admitir métodos de compresión de última generación, se emplea para transportar y almacenar conjuntos de datos científicos de todo tipo, como:

- Matrices multidimensionales: espectros 1D, imágenes 2D, cubos de datos 3D +.
- Tablas que contienen filas y columnas de información.
- Las palabras clave de encabezado que proporcionan información descriptiva sobre los datos.

6.1. Historia del estándar

El formato FIT nace a raíz de la necesidad del intercambio de datos entre los distintos observatorios astronómicos. [5][6] A finales de los años 70, D.C. Wells que trabajaba para el observatorio nacional de Kitt Peak, situado en el desierto de Sonora en Arizona, y R.H. Harten, que pertenecía a la fundación de radio astronomía de Países Bajos, definieron en 1976 por primera vez este formato.

Esta necesidad de un formato que permitiera el intercambio de imágenes entre los distintos observatorios mundiales, se planteó en una reunión de la sección de astronomía de la fundación nacional de ciencias de estados unidos, lo que llevo a la formación de un grupo de trabajo para el problema. Aunque la mayoría de los detalles técnicos del formato fueron desarrollados en 1979, no se estandarizó hasta 1981, cuando en la revista europea Astronomy and Astrophysics, D.C. Wells, R.H. Harten y E.W. Greisen publicaron los primeros artículos que definían el formato FIT. Gracias a esto, el formato FIT se convirtió rápidamente en el estándar para el intercambio de datos de la comunidad astronómica.

En 1982, el formato FIT fue aprobado oficialmente por la Unión Astronómica Internacional, adoptando la mayoría de proyectos y organizaciones posteriormente este formato para compartir y archivar sus datos científicos.

6.2. Definición del estándar FITS

Se puede definir este formato como un formato para el almacenamiento transmisión y procesamiento de datos astronómicos, el cual fue diseñado

específicamente para dicha función. Además, Incluye disposiciones como la descripción de información de calibración espacial y fotometría, así como los metadatos del origen de la imagen. Éstos se almacenan en la cabecera de la imagen en formato ASCII en tablas, estas se utilizan en FITS para permitir una fácil interpretación de los registros de encabezado por parte de humanos y ordenadores.

Tal y como viene definido en la última versión del estándar:
“Cada registro de palabra/clave de la cabecera, de 80 caracteres, constará de un nombre de palabra/clave, un indicador de valor (sólo necesario si hay un valor) un valor opcional y un comentario opcional. Las palabras clave pueden aparecer en cualquier orden, excepto cuando se indique específicamente lo contrario en esta norma. Se recomienda que el orden de las palabras clave en los archivos FITS se conserve durante las operaciones de procesamiento de datos porque los diseñadores del archivo FITS pueden haber utilizado convenciones que otorguen un significado particular al orden de ciertas palabras clave (por ejemplo, agrupando secuencias de palabras clave de comentario en lugares específicos de la cabecera, o añadiendo palabras clave HISTORY en orden cronológico de los pasos de procesamiento de datos o utilizando palabras clave CONTINUE para generar valores de palabras/clave de cadena larga).”

Varios Autores: FITS Working Group, International Astronomical Union.

Por lo tanto, se puede decir que el estándar FITS consta de dos cosas, por un lado, almacena la imagen y por otra una cabecera donde se muestra distintas etiquetas que dan información variada sobre las condiciones técnicas con la que se ha tomado esa imagen.

6.3. Estructura de una imagen FIT

Las imágenes FITS están compuestas de una o más unidades de datos (HDU) la primera HDU es una matriz de pixeles N-dimensional con valores de 0 a 2 elevados a 16-1, de ahí que normalmente las imágenes sean de 16 bit. Adicionalmente pueden contener extensiones que pueden ser: tablas ASCII con filas y columnas de datos en

formato de caracteres ASCII (cabeceras) de 80 bytes o tablas binarias de datos con los mismos tipos de datos numéricos admitidos para la matriz principal [5].

```
Filename: Desktop/TFG/archivos e imagenes/2008QD4-035Cle.fit
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      77   (1024, 1024)   int16 (rescales to uint16)
```

Figura 1: Estructura información archivo fit

La resolución soportada por las imágenes FITS no está definida ni delimitada, aunque si está delimitada la cuantización de los pixeles, ya que las imágenes son de 16 bit la cuantización máxima es de 65632 cuentas.

El magic number (primeros bytes de un fichero que sirve para identificar el tipo de fichero) de los archivos FITS es HEX: 53 49 4d 50 4c 45. Para identificar estos ficheros en este proyecto se va a usar las distintas extensiones que podría tener.

6.4. Cabecera FITS

La cabecera de un archivo fit contiene datos de la observación y del archivo, incluyendo por ejemplo el instrumento con el que se obtuvo la imagen, coordenadas, fecha, los filtros utilizados, etc. Normalmente las cabeceras comienzan con la clave “SIMPLE” y acaban con la clave “END”.

```
[6]: SIMPLE  =                    T
     BITPIX  =                   16 /8 unsigned int, 16 & 32 int, -32 & -64 real
     NAXIS   =                    2 /number of axes
     NAXIS1  =                 1024 /fastest changing axis
     NAXIS2  =                 1024 /next to fastest changing axis
     BSCALE  =   1.0000000000000000 /physical = BZERO + BSCALE*array_value
     BZERO   =   32768.000000000000 /physical = BZERO + BSCALE*array_value
     DATE-OBS= '2013-02-17T05:04:56.85' /YYYY-MM-DDThh:mm:ss observation start, UT
     EXPTIME =   500.00000000000000 /Exposure time in seconds
     EXPOSURE=   500.00000000000000 /Exposure time in seconds
     SET-TEMP=  -110.00000000000000 /CCD temperature setpoint in C
     CCD-TEMP=  -110.00000000000000 /CCD temperature at start of exposure in C
     XPIXSZ  =   27.000000000000000 /Pixel Width in microns (after binning)
     YPIXSZ  =   27.000000000000000 /Pixel Height in microns (after binning)
     XBINNING=                    2 /Binning factor in width
     YBINNING=                    2 /Binning factor in height
     XORGSUBF=                    0 /Subframe X position in binned pixels
     YORGSUBF=                    0 /Subframe Y position in binned pixels
     FILTER  = 'Clear   ' /           Filter used when taking image
     IMAGETYP= 'LIGHT   ' /           Type of image
     OBJCTRA = '09 32 52' /           Nominal Right Ascension of center of image
     OBJCTDEC= '+45 43 29' /          Nominal Declination of center of image
     OBJCTALT= ' 32.6065' /           Nominal altitude of center of image
     OBJCTAZ = '305.5162' /           Nominal azimuth of center of image
     OBJCTHA = '  5.2572' /           Nominal hour angle of center of image
     SITELAT = '37 03 51' /           Latitude of the imaging location
     SITELONG= '-03 23 05' /          Longitude of the imaging location
     JD      =   2456340.7117690970 /Julian Date at start of exposure
     JD-HELIO=   2456340.7193757351 /Heliocentric Julian Date at exposure midpoint
     AIRMASS =               1.7913 / Airmass at exposure start
     TRAKTIME=   1.0000000000000000 /Exposure time used for autoguiding
     FOCALLEN=   12000.000000000000 /Focal length of telescope in mm
     APTDIA  =   1500.0000000000000 /Aperture diameter of telescope in mm
     APTAREA =   1608102.7843058109 /Aperture area of telescope in mm^2
     SWCREATE= 'MaxIm DL Version 5.12' /Name of software that created the image
     OBJECT  = '        '
```

Figura 2: estructura cabecera

7. Calibración de imagines astronómicas.

El calibrar una imagen astronómica significa corregirla de ruido, viñeteo, artefactos o manchas del sensor en estas 2 imágenes vamos a ver alguno de los ejemplos mencionados

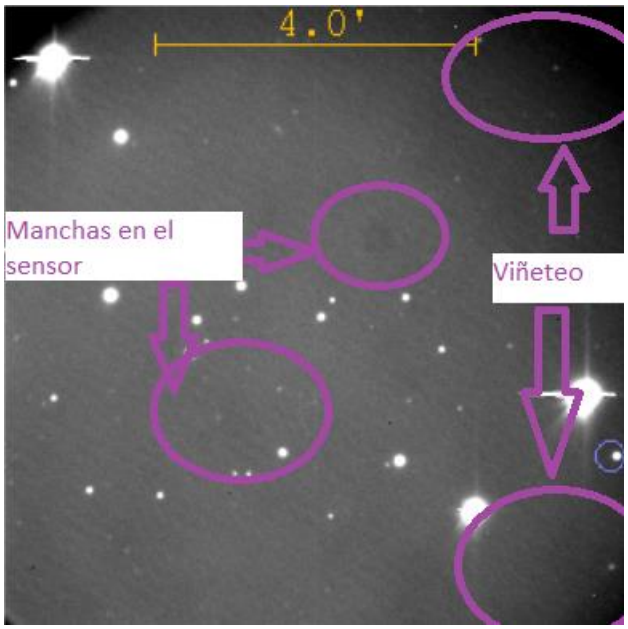

Figura 3: Imagen astronómica sin calibrar

En esta imagen se pueden observar distintos gradientes, sobre todo en las esquinas y pequeñas partículas de polvo, todo esto será corregido gracias a la calibración

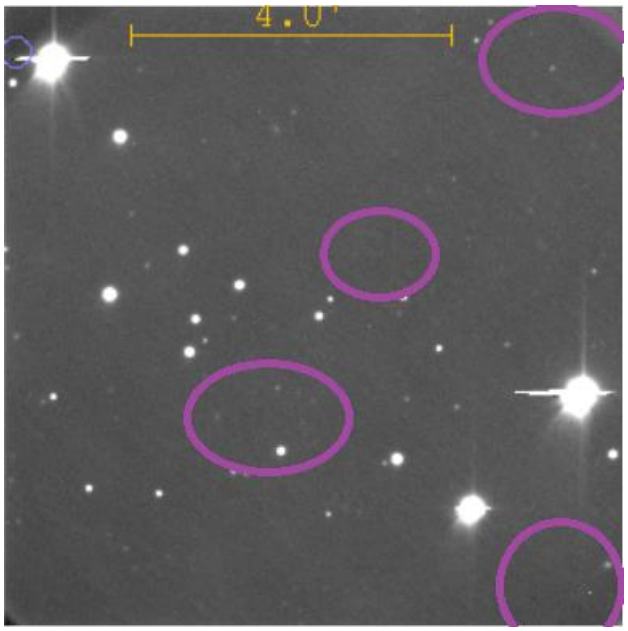

Figura 4: Imagen astronómica calibrada

Como se puede observar las manchas en el sensor y el viñeteo ha sido corregido gracias a la calibración.

Antes de profundizar en el tema de la calibración de imágenes, es necesario hablar de las imágenes que se toman y las cámaras que hacen posibles estas imágenes.

La mayoría de telescopios, y en particular los utilizados por el IAA, los cuales he mencionado anteriormente, utilizan CCD de sus siglas en inglés “charge-coupled device” (dispositivo de carga acoplada) como sensores en sus cámaras, de ahí que se les llame a las cámaras CCD.

[7] Cuando un fotón golpea a cada pixel del sensor hace que se liberen electrones, esto es el efecto fotoeléctrico, cada pixel y sus puertas electrónicas asociadas actúan como si fueran un condensador, el cual recoge estos electrones de los pixeles cuando la luz incide sobre ellos. También es recogida una carga que se acumula hasta que el sensor es leído.

Durante la lectura, el procesador mide la carga recogida en cada pixel, por lo tanto, lo que finalmente se envía desde el sensor CCD al ordenador es la posición del pixel y su carga como una representación digital, creándose así las imágenes que se obtienen del sistema.

Como se ha mencionado anteriormente, para que una imagen de este tipo sea útil, es necesario que tenga una buena etiqueta o cabecera, para así poder tener información de la imagen en sí.

Una de las claves para poder analizar las imágenes que se obtienen a través de los observatorios, es su calibración, para que así se represente con más precisión la señal que se ha obtenido de los cuerpos celestes. Las fuentes de señales no astrofísicas deben cuantificarse y eliminarse en la medida de lo posible para que no contaminen los datos obtenidos.

Debemos tener en cuenta tres tipos de archivos para la correcta calibración y así eliminar el ruido de lectura de la cámara, el ruido de corriente oscura de la cámara (debido al movimiento de los electrones en el sensor) y la diferente iluminación del chip de la cámara, así como la presencia de suciedad en el chip y/o en las ópticas.

[8] La cámara CCD y la electrónica que tiene, hace que tengan un ruido propio, esto hace que se agregue una señal a cada fotograma tomado, independientemente de los tiempos de exposición, para corregir este ruido se han de tomar imágenes de calibración *bias* que compensan el ruido de lectura, las alteraciones provenientes del ordenador y el ruido eléctrico. Se deben hacer en la oscuridad con el obturador cerrado y/o la entrada de luz cubierta, y el tiempo de exposición debe ser lo más corto posible. Para poder calibrar las imágenes con *bias* se han de hacer un numero variado de tomas para así poder hacer una media de los datos *bias* obtenidos y tener mejor precisión, obteniendo un *Master Bias*.

Es muy importante que estas tomas se hagan con la misma temperatura que las imágenes astronómicas, y tan baja como sea posible.

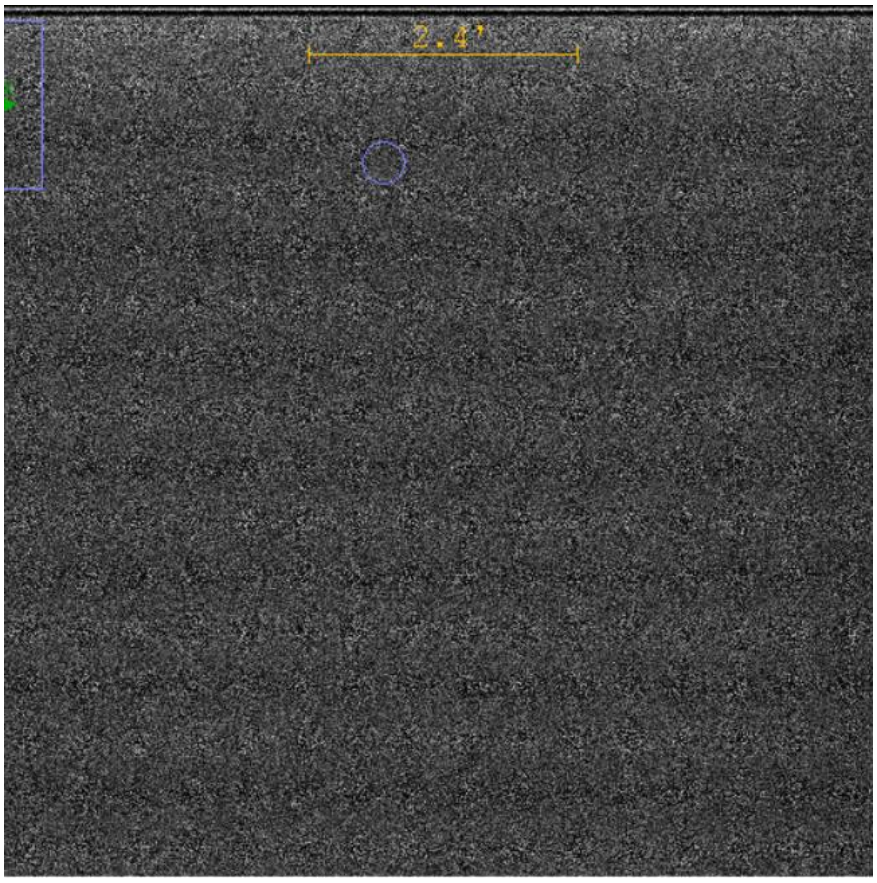

Figura 5: Imagen Bias

Una vez obtenido el *Master Bias* hay que realizar una serie de tomas con el obturador cerrado y/o con la entrada de luz cubierta, y el tiempo de exposición debe ser igual o superior que el de la imagen astronómica. Estas tomas son llamadas imágenes *dark*. Se debe de hacer porque los movimientos de los electrones en el chip

generan corriente oscura. Estas tomas, sirven para cuantificar la corriente oscura, y así poder restarlo al de la imagen astronómica.

Por último, se realizarán tomas de una fuente de luz uniforme o del cielo crepuscular, con el mismo enfoque que el de la imagen astronómica a calibrar, y el tiempo de exposición debe de ser aproximadamente la mitad de la capacidad total del pixel. Esto nos dará una imagen *flat* para cada filtro utilizado. Al igual que con las imágenes *bias*, se ha de hacer una mediana de todas las imágenes *flat* obtenidas cada noche astronómica para así tener más precisión a la hora de calibrar y obtener un master *flat*. Se ha de dividir la imagen de ciencia entre este *Master Flat* normalizado para no interferir en la señal de la imagen de ciencia al aplicar los ficheros de calibración. Para normalizar los *flat* hay que dividirla entre la mediana de los valores de los pixeles. Además de lo anterior, a cada uno de los flat hay que restarle el *Master Bias* correspondiente antes de normalizarlo y realizar la mediana. Al ser aplicada a la imagen astronómica, hará que se compensen los problemas como la suciedad, el polvo en las superficies ópticas o los reflejos en los deflectores.

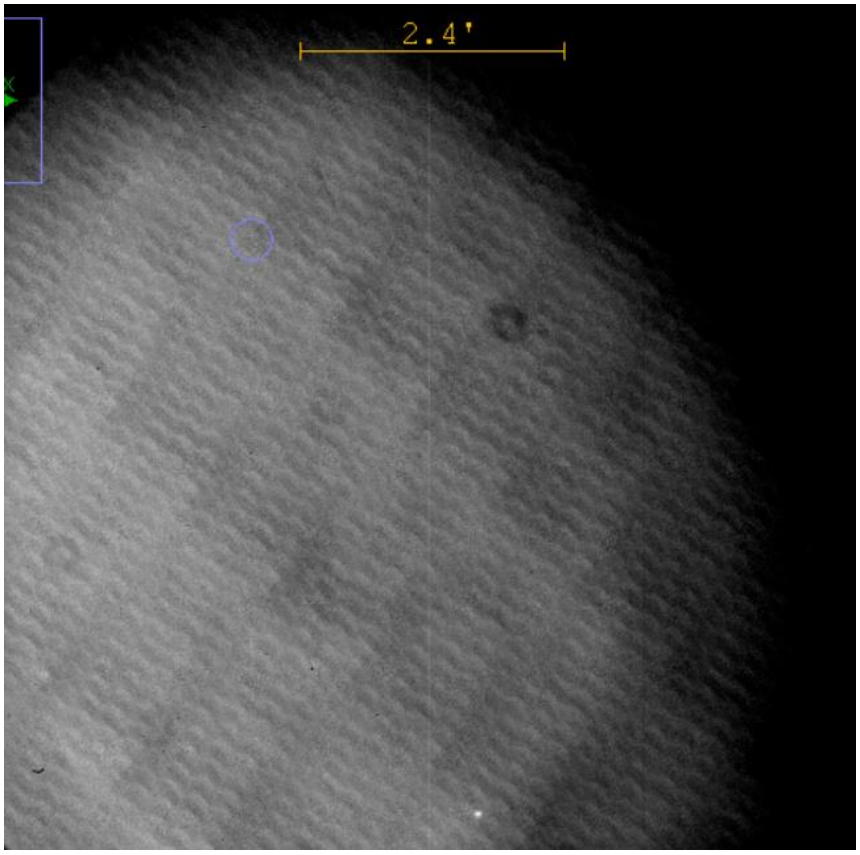

Figura 6: Imagen Flat

Aunque se menciona anteriormente que para calibrar bien las imágenes de los cuerpos celestes se han de calibrar con tres tipos de imágenes distintas: *Bias*, *Dark* y *Flat*, no se va a tener en cuenta las imágenes de calibración *Dark*, ya que estas solo se aplican si la cámara no está enfriada a muy bajas temperaturas, sino que es enfriada con técnicas electrónicas usando el denominado efecto Peltier. Teniendo en cuenta que las cámaras de los telescopios de Calar Alto y del OSN se enfrían con nitrógeno líquido, la temperatura de las cámaras está siempre por debajo de -100ºC, por lo tanto, no haría falta aplicar las imágenes *dark* a la calibración y solamente se usarán *bias* y *flat*.

La calibración, podría ser resumida como:

$$(IC-MB)\ /MF \tag{1}$$

Donde: IC=Imagen de ciencia

MB=*Master Bias*

MF= *Master Flat* ya normalizado y habiéndole restado el *bias*

8. Metodologías de desarrollo de software

8.1. Metodología a elegir

Para el desarrollo de este proyecto se han valorado 2 metodologías distintas: el modelo en cascada y una metodología ágil, Scrum, esta se ha tenido en cuenta ya que es la metodología más empleada en la actualidad en las empresas.

La metodología que se va a seguir en este proyecto va a ser la del modelo en cascada, esto se debe a que el modelo en cascada es más adecuado para proyectos

en los que solo hay un desarrollador, mientras que en Scrum se necesitaría que hubiera mínimo cuatro personas, además dadas las especificaciones y lo concreto que es el proyecto los requisitos de este no van a cambiar.

8.2. Metodología en Cascada

[10] En ingeniería de software, el desarrollo en cascada o modelo en cascada, es la metodología que ordena rigurosamente las etapas de desarrollo de software, de tal manera que una etapa debe esperar a que la anterior esté finalizada completamente. Es por esto, que se denomina desarrollo en cascada, por esta manera de ordenar las etapas. Fue el primer modelo en originarse y es la base del resto de modelos que se han ido desarrollando después.

La primera versión del modelo en cascada, fue propuesta por Winston W. Royce en 1970, y en 1980 y 1985 fue revisada por Barry Boehm e Ian Sommerville respectivamente.

Las diferentes etapas del desarrollo en cascada son:

1. Análisis de requisitos
2. Diseño del sistema
3. Diseño del programa
4. Codificación
5. Pruebas
6. Verificación
7. Mantenimiento

Aunque actualmente, es más común encontrar estas fases dividas solamente en 5 etapas, fusionando algunas de ellas, se encuentran:

1. Análisis
2. Diseño
3. Implantación
4. Verificación
5. Mantenimiento

Cualquier error en el diseño e implementación, es detectado en la etapa de verificación, llevando a un rediseño y a una nueva implantación.

Análisis: Es la primera etapa del proyecto, la etapa de preparación. En esta fase, se tiene que determinar cuáles son las necesidades y objetivos que el proyecto tiene que cumplir. Posteriormente, se han de reunir todos los requisitos que se deben diseñar e implementar. Por lo general si trabajas para un cliente, en esta fase es cuando se le presenta al cliente la propuesta de proyecto.

Diseño: En esta fase se empezará a diseñar el software del proyecto, se darán los primeros esbozos, y se debe definir la estructura de todos los elementos necesarios para el desarrollo. También es importante describir la relación entre los distintos elementos.

Implementación: En la etapa de implementación se realizará una traducción del diseño a código, se deben realizar pruebas para verificar que no existan errores e ir dándole poco a poco forma al producto final.

Verificación: Como se menciona anteriormente, en esta fase se ejecuta el código y se comprueba su funcionalidad, comparando los resultados con los objetivos establecidos en la fase del análisis. También es conveniente presentar el resultado al cliente para que de una valoración.

Mantenimiento: esta es la última fase del modelo en cascada, donde se han de analizar los resultados obtenidos y realizar los cambios que se vean oportunos para dar por finalizado el proyecto.

Esta fase no solo se realiza una vez, sino que se ha de volver a ella para comprobar que se va adaptando a los cambios del entorno.

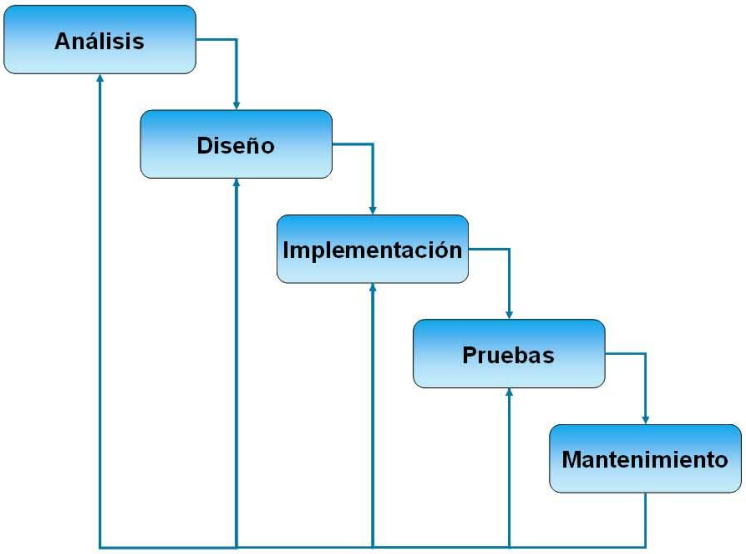

Figura 7: Fases modelo en cascada

9. Ingeniería de requisitos

9.1. Análisis

El objetivo principal del sistema es el de calibrar una cierta cantidad de imágenes de cielo de forma automática, teniendo en cuenta unos ciertos pasos previos a la calibración. Estos pasos mencionados nacen, como se menciona anteriormente, de la necesidad de salvar las distintas diferencias que puede haber en las imágenes obtenidas a lo largo de los años.

Por ello se explicará más adelante en el apartado del algoritmo, cómo se han solventado los distintos problemas que he ido detectando.

Para la accesibilidad de la aplicación, he tenido en cuenta en un primer momento realizar una interfaz simple, pero dado que el uso de esta aplicación va destinada a un grupo de personas muy concreto y puesto que estas personas tienen conocimientos de uso de la línea de comandos (CMD) y conocimientos básicos de programación, se propone para ejecutar la aplicación el uso del CMD.

9.2. Requisitos funcionales

Después de varias reuniones con el experto, y dejando claros los requisitos del proyecto, los podemos resumir en:

1. El usuario podrá elegir un directorio para que se analicen todos los archivos que hay, posteriormente se crearán archivos *Master Bias*, *Master Flat* y se calibrarán todas las imágenes de ese directorio.
2. El usuario podrá elegir un directorio y solo se analizarán los *Master Bias* y *Master Flat*, posteriormente se calibrarán todas las imágenes del directorio.

9.3. Planificación temporal

Para hacer una planificación temporal se estima el tiempo que se le va a dedicar a cada uno de los requisitos, puesto que este sistema solo tiene dos requisitos y dado que comparten la mayoría de funciones, la planificación temporal va a constar de una tabla con la estimación total del proyecto y a continuación, en otra tabla se desglosará una estimación del tiempo empleado en la generación del algoritmo.

Tarea	Estimación Temporal
Búsqueda de información	2 semanas
Diseño y Estudio	3 semanas
Implementación del código	3 semanas
Pruebas	1 semanas
Elaboración de la documentación	4 semanas
Total	**13 semanas**

Tabla 1: Planificación temporal

Búsqueda de información: En estas dos primeras semanas, se empezará buscando información sobre qué son los archivos FITS, cómo manejarlos, como crearlos etc. También se realizará una búsqueda de información sobre la historia del estándar, así como sobre los telescopios que se utilizaron para la obtención de las imágenes. Además, se va a estudiar cual será el entorno de desarrollo óptimo a usar, así como las distintas librerías y paquetes que se necesitarían para la implementación del código.

Diseño y estudio: En estas tres semanas se va a tener una primera toma de contacto con imágenes .fit, las cuales han sido obtenidas por medio del IAA, en

concreto han sido proporcionadas por un miembro del grupo de Cuerpos Menores. También se estudiarán los distintos fallos que se pueden obtener a la hora de leer las cabeceras de las imágenes y encontrar una posible solución a estos fallos. Además, va a empezar a diseñar la estructura general del código, y los diferentes y numerosos pasos previos antes de poder hacer una calibración automática de imágenes.

Implementación: Durante estas tres semanas se va a implementar el código, basándose en el diseño y en los estudios previos realizados, intentando solventar los distintos problemas encontrados y consiguiendo realizar una calibración automática de imágenes astronómicas, el objetivo del proyecto.

Pruebas: En esta semana lo que se va a hacer es proporcionarle el código a un experto del grupo de Cuerpos Menores del IAA para que pruebe su funcionamiento y busque posibles fallos, dándome así una valoración de este, y posteriormente procediendo a corregir los fallos encontrados.

Elaboración de la documentación: Durante cuatro semanas va a redactar la memoria en la cual se encuentra toda la información necesaria para comprender la necesidad de la realización de la aplicación, así como información sobre todo el proyecto.

En la siguiente tabla, se muestra un desglose del apartado de diseño e implementación del código:

Tarea	Estimación temporal
Estudio y unificación de los diferentes tipos de imágenes	20 horas
Estudio y unificación de los diferentes tipos de telescopios	15 horas
Estudio y unificación de los diferentes tipos de filtros	15 horas
Estudio y transformación de fecha en fecha astronómica	5 horas
Estudio e identificación de parámetros en el path	15 horas

Clasificación de las imágenes y creación del log principal	20 horas
Agrupación y creación de archivo de información y log de *Bias* y *Master Bia*s	20 horas
Agrupación y creación de archivo de información y log de *Flat* y *Master Flat*	25 horas
Calibración de las imágenes de ciencia.	20 horas
Total	**150 horas**

Tabla 2: Desglose horas Diseño e Implementación

La estimación total de tiempo consta de 13 semanas, de las cuales los sábados y domingos no se trabaja, y el resto de días se dedican 3 horas por la mañana y 2 por la tarde, siendo el cómputo total de horas dedicadas al proyecto de 325 horas en total, de las cuales 150 horas serán exclusivas del diseño y la implementación.

Tareas /Semana	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9
Búsqueda de información									
Información Telescopios	✓								
Información Fit	✓	✓	✓						
Entorno de desarrollo			✓	✓					
Librerías				✓	✓				
Diseño y estudio									
Estudio y unificación de tipo, telescopios, filtros y fecha			✓	✓	✓	✓	✓		
Identificación parámetros en path					✓	✓	✓		
Implementación del código									
Creación log principal						✓	✓	✓	
Creación Master Bias y log							✓	✓	✓
Creación Master Flat y log							✓	✓	✓
Calibración de las imágenes								✓	✓
Pruebas									✓

Tabla 3: Diagrama de Gantt 1

Tareas /Semana	Semana 10	Semana 11	Semana 12	Semana 13
Elaboración de la documentación	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	

Tabla 4: Diagrama de Gantt 2

10. Detalles de la implementación

10.1. Lenguaje de programación a usar

En este proyecto se va a usar el lenguaje Python como lenguaje principal de programación debido a que es una herramienta multiplataforma usada ampliamente en entornos científicos, a su facilidad de uso, y a la gran disponibilidad de bibliotecas para múltiples propósitos. Python fue creado por Guido Van Rossum, un informático holandés en 1989, aunque la primera versión publica, la 0.9.0, fue en 1991.

A raíz de esto, el lenguaje fue evolucionando, llegando tener la versión 2.0 en el año 2000 y la versión 3.0 en 2008. En su primera versión, ya incluía clases con herencia, excepciones, funciones, y su principal característica: un funcionamiento modular, para que así la gente con pocos conocimientos informáticos supiera manejarlo, haciendo el código más limpio y accesible, esta característica se mantiene a día de hoy. Las versiones han ido mejorándose unas a otras sin dejar de funcionar las anteriores, y la última actualización que se tiene es de octubre de 2020 [12].

10.2. Ventajas y desventajas

[13][14]Ventajas:

- Lenguaje multiplataforma que funciona en cualquier sistema
- Tiene una amplia flexibilidad en cuanto a la declaración, por ejemplo, de los tipos de datos, incluso no hay por qué declararlos, todo lo contrario, pasa en Java o C++
- En el entorno de Python encontramos IDEs (Entorno de Desarrollo Integrado) como Jupyter notebook, Vs code o Pycharm Comunity, todos ellos gratuitos y muy potentes, en concreto Jupyter, el cual puede ser usado para ciencia de datos
- Dada la simpleza ya comentada, la creación de script en Python es muy sencilla por lo que es muy fácil de aprender como lenguaje y puede ayudarnos en tareas automáticas
- Tiene un enfoque orientado a objetos

- Usa un intérprete
- Es el lenguaje líder para el análisis de datos y el aprendizaje automático
- Es posible realizar comprobaciones en el código mientras se está ejecutando
- Tiene una amplia cantidad de librerías, sobre todo algunas muy potentes para la astronomía como es Astropy.

Desventajas:

- Es más lento con respecto a otros lenguajes, a la hora de ejecutarse debido al uso del intérprete
- Hay muchos servidores hosting que no tienen soporte para Python

11. Entorno de desarrollo

Figura 8: Logo Anaconda

[15] Como ya he mencionado, el lenguaje que voy a usar para desarrollar la aplicación que resuelva el problema propuesto va a ser Python, para empezar a programar es necesario instalar Anaconda. Con esto estaremos instalando 4 sectores: **Anaconda Navigator, Anaconda Project,** las **librerías de ciencia de datos** y **Conda.** Para instalarlo solo hay que descargar el ejecutable de la página web oficial.

[16]Anaconda es una distribución libre y de código abierto, por lo tanto, es gratuita, que nos sirve para simplificar el despliegue y administración de paquetes

software. En general, es un gestor de entornos y paquetes que contiene una gran colección de librerías y paquetes de código abierto.

Aparte nos hará falta un editor de texto que soporte Python y que pueda ejecutarlo, Anaconda Navigator nos trae ya instalados algunos como Jupyter Notebook y algunos más, pero yo voy a instalar JupyterLab, que, aunque es muy similar a Jupyter Notebook, es más completo.

Tal y como viene definido en [16]:" JupyterLab es un entorno de desarrollo interactivo basado en web para cuadernos, código y datos de Jupyter. JupyterLab es flexible: configure y organice la interfaz de usuario para admitir una amplia gama de flujos de trabajo en ciencia de datos, informática científica y aprendizaje automático. JupyterLab es extensible y modular: escribe complementos que agregan nuevos componentes y se integran con los existentes."

Para su instalación solo tenemos que abrir el terminal de anaconda y ejecutar la siguiente orden: "conda installa -c conda-forge jupyterlab".

Para su ejecución, se puede abrir desde el terminal ejecutando "jupyter-lab" o simplemente podemos usar la interfaz de Anaconda Navigator y darle a lanzar la aplicación. Al darle se nos abrirá un entorno basado en web con un cuaderno donde podemos empezar a programar.

12. Biblioteca a usar

12.1. Diferentes bibliotecas

Haciendo un estudio sobre las diferentes librerías y paquetes posibles que se pueden utilizar para el procesamiento de las imágenes astronómicas, destacan los siguientes:

1. [17] Siril: Es una herramienta de procesamiento de imágenes astronómicas, capaz de convertir, preprocesar imágenes, ayudar a alinearlas automática o manualmente, apilarlas y mejorar las imágenes finales.
2. [18] Astropy: "El paquete Astropy contiene funciones clave y herramientas comunes necesarias para realizar astronomía y astrofísica con Python. Es el

núcleo del Proyecto Astropy, que tiene como objetivo permitir que la comunidad desarrolle un ecosistema sólido de paquetes afiliados que cubran una amplia gama de necesidades de investigación astronómica, procesamiento de datos y análisis de datos."

3. [19] Amuse: Una biblioteca para realizar simulaciones astrofísicas de diferentes dominios físicos y escalas.

Después de haber revisado estas tres librerías y debido a que el problema que tengo que resolver es algo muy específico, la librería que más se ajusta a las necesidades del software a diseñar es Astropy.

12.2. Astropy

Figura 9: Logo Astropy

[20] El proyecto Astropy nace de la comunidad astrofísica, para el desarrollo de un paquete centralizado para la astronomía, y utiliza el lenguaje Python. Gracias a este proyecto se mejora la usabilidad, interoperabilidad y colaboración entre los muchos paquetes de astronomía que usan este lenguaje de programación.

El paquete central de Astropy contiene funcionalidades que van dirigidas a astrónomos y astrofísicos, aunque también pueden servir a cualquier persona que quiera implementar algún software sobre astronomía.

Este proyecto incluye también paquetes afiliados, que son paquetes que aun no habiendo sido desarrollados por el equipo que desarrollo Astropy, han sido incluidos, estos paquetes afiliados comparten el objetivo de Astropy y se basan en la infraestructura del paquete central.

Aparte del código en sí, Astropy es una comunidad de desarrolladores que están de acuerdo en compartir su código de manera abierta, ya que esto es saludable para la comunidad de la astronomía y la astrofísica.

12.2.1. Utilidades de Astropy

[21]Con Astropy podemos hacer muchas cosas, entre ellas destacan las siguientes:

1. Convertir una velocidad radial al sistema de reposo local.
2. Determinar y trazar la altitud de un objeto celeste.
3. Transformar posiciones y velocidades desde/hacia un marco galactocéntrico.
4. Crear archivos FITS.
5. Acceder a los datos almacenados como tablas de multiextensión de los archivos FITS.
6. Leer y trazar una imagen FITS.
7. Editar encabezados FITS.

Para este proyecto nos vamos a centrar en cómo acceder a los datos, crear un archivo FITS, y editar sus cabeceras.

Figura 10: Reconocimiento Astropy

12.2.2. Instalación

Para instalar Astropy, tan solo se necesita abrir JupyterLab desde Anaconda Navigator y escribir en el Notebook “conda update astropy”. Una vez escrito, se ejecuta la celda y se reinicia el kernel para que se complete la instalación. También se puede ejecutar ese comando desde el terminal propio de Jupyter.

Tras realizar esto solo deberemos importar fits de astropy.io cada vez que abramos de nuevo Jupyter Lab: “from astropy.io import fits”

13. Diseño del algoritmo y problemas detectados.

Una vez visto el entorno de desarrollo a usar y la librería principal del proyecto es la hora de diseñar el problema, el primer paso que se debe dar es como leer un archivo FITS, una vez realizado esto nos encontramos con el primer problema, los archivos FITS pueden tener más de una cabecera, este problema no ha sido solventado, ya que las imágenes FITS del OSN y de Calar Alto solo tienen una cabecera, aunque no se haya arreglado, en un futuro en la fase de mantenimiento, una vez entregado el proyecto es posible modificar el código para arreglarlo.

El proceso a seguir viene definido en el siguiente diagrama y será explicado más adelante.

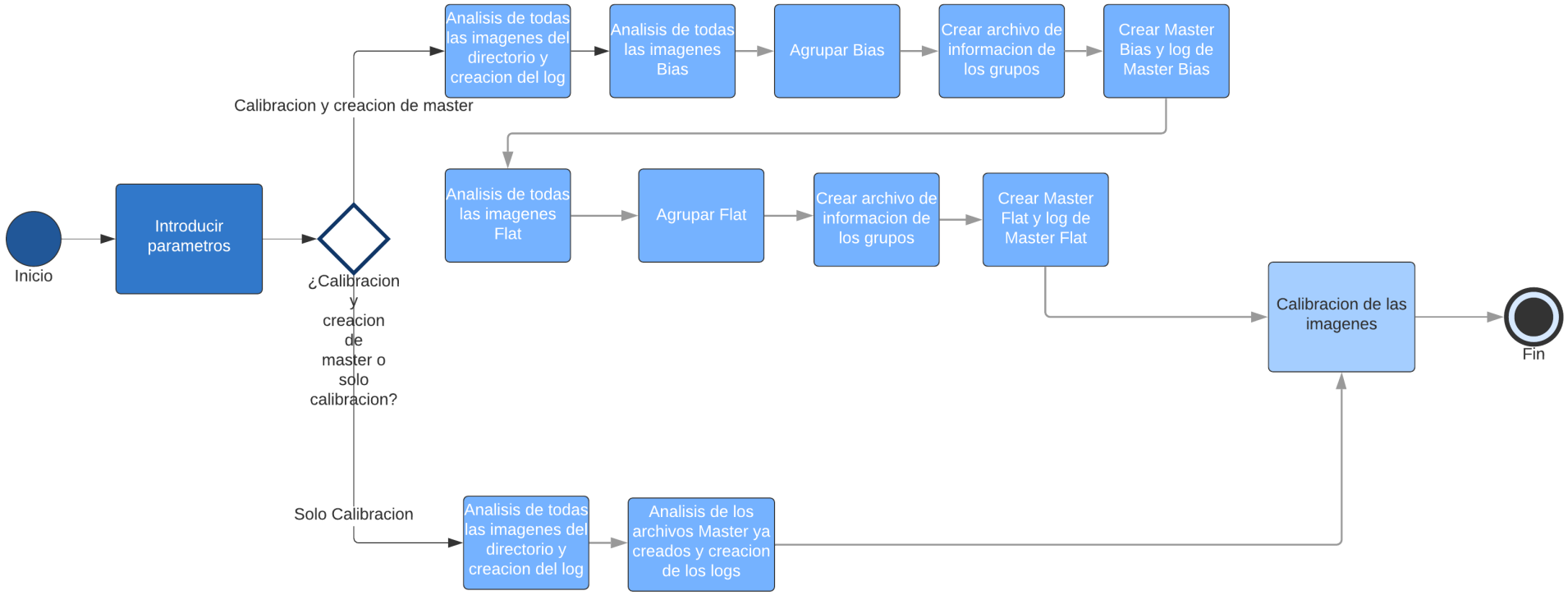

Figura 11: Diagrama de bloques.

Una vez leída la cabecera, el siguiente paso es obtener de ella los distintos parámetros que harán que se pueda clasificar una imagen, estos parámetros son: el telescopio, el filtro, la fecha, numero de pixeles en el eje X, numero de pixeles en el eje Y, el binning (técnica que se usa para registrar en vez de un pixel, los n^2 pixeles adyacentes y asignarlos como un único pixel), y el tipo de imagen.

Como se ha mencionado anteriormente, puesto que las imágenes llevan tomándose desde hace muchos años y las cabeceras y el estándar FITS ha ido cambiando con el tiempo, es posible que haya etiquetas de las cabeceras que sean incorrectas, que tengan otro nombre o incluso que no existan, y por esto nos encontramos el siguiente problema: es posible que el tipo de imagen tenga un nombre distinto cada vez, por ejemplo en las imágenes del OSN en la etiqueta IMGTYPE viene una imagen de cielo como *science* y en las de calar alto viene como *Sky*. Este problema se ha solventado haciendo una lista de posibles casuísticas con los distintos tipos de imágenes y unificando cada uno de ellos en una sola palabra clave. Además, es posible una imagen puede tener en la etiqueta IMGTYPE la palabra *science* y diríamos que esa imagen es de ciencia, es decir de cielo, pero al ver por ejemplo el tiempo de exposición de esta, nos damos cuenta que es muy cercano a 0, por lo tanto, no es posible que sea una imagen de ciencia sino una imagen *bias*. Para corregir este problema lo que se ha hecho es comprobar que el tiempo de exposición es 0. También es posible encontrarse con el caso de que en IMGTYPE venga el tipo *ligh*, con lo cual sería una imagen de cielo, pero en la ruta o el nombre del archivo venga otro tipo de imagen, si se da este caso nos quedaremos con el tipo que viene en la ruta o nombre ya que un 90% de las veces es correcto. En cualquier otro caso la imagen se marcará como no leída.

Hecho esto, pasamos a la obtención del filtro que solo se buscará si el tipo de imagen es *SKY* o *FLAT*, y nos vamos a encontrar con el mismo problema que antes, existe una variedad inmensa de palabras que se pueden referir a un mismo filtro, por lo tanto, se han hecho varias listas con las distintas casuísticas posibles que podríamos encontrarnos, para así unificarlos en uno solo. Además, es posible que si la imagen es muy antigua no tenga la etiqueta FILTER, para este problema lo único que tenemos que hacer es buscar en el path y en el nombre del archivo si existe alguna referencia al filtro.

Realizado esto, lo siguiente es encontrar con que telescopio se realizó la imagen, para ello se buscará la etiqueta TELESCOPE y tal y como pasa con el filtro y el tipo de imagen, se han de unificar todas diferentes etiquetas en una sola.

A continuación, debemos de obtener la fecha, uno de los parámetros más importantes, lo primero es obtenerla de la etiqueta DATE-OBS, pero esta fecha no nos sirve, ya que para calibrar las imágenes debemos obtener la fecha astronómica, aquí tenemos el siguiente problema. Por convenio, la fecha astronómica va desde las 12 del mediodía hasta las 12 del mediodía del día siguiente, por lo tanto, una imagen tomada a las 22:00 del día 5 de febrero pertenece a la misma noche astronómica que una tomada a las 2:00 del día 6 de febrero. Para solventar este problema debemos de hacer uso de la librería DATETIME, con esta librería podremos pasar la fecha obtenida en formato *string* a formato *date*, y una vez realizado esto podremos restarle una hora si la hora que tiene la imagen es antes de las 12 del mediodía, para así obtener la fecha astronómica.

El resto de parámetros son obtenidos sin ningún problema de las distintas etiquetas.

Realizados todos estos pasos, se creará un log inicial con los datos de todas las imágenes que se encuentran en la carpeta pasada como parámetro, el cual será útil para no tener que ir imagen por imagen a la hora de calibrarlas. Si se ha encontrado alguna contrariedad, por ejemplo, en el path se hace referencia al tipo de imagen y en la etiqueta IMGTYPE viene un tipo de imagen distinta a la del path, se le dará más importancia a lo que ponga en el path ya que el 90% de las veces los resultados son correctos, y se marcará en el log como que se ha encontrado una contradicción, para que el experto al verlo determine si es acertado o no.

```
funcion crearLogs(ruta)
    pathCreationLog<-'log & info/'
    CrearDirectorio(pathCreationLog)
    CrearArchivoExcel("fits_calibrator_log.xlsx")
    RellenarCabecerasExcel
    Para todos los elementos de ruta:
        Para todos los archivos:
            Si archivo == Fit:
                archivo_fit<-abrirArchivo(archivo)
                header<-archivo.header
                Si tamaño archivo == 1:
                    type<-buscar tipo en path
                    Si type== vacio:
                        type<-header.tipo
                        Si type != "Unknown":
                            time<-archivo.tiempoDeExposicion
                            Si time == 0 :
                                rellenar log con type "Bias"
                                rellenar log warning
                            Si no:
                                rellenar log con type
                        Si no
                            rellenar log con archivo no leido
                    Si no
                        rellenar log con type
                    rellenar log con header.nombre
                    rellenar log con header.path
                    pX<-header.pixelX
                    pX<-header.pixelY
                    rellenar log con pX y pY
                    Intentar:
                        binning<-header.naxis
```

Figura 12: creación log 1

```
                    Excepcion:
                        Intentar:
                            binning<-header.binning
                        Excepcion:
                            binning<-"Unknown"
                    rellenar log con binning
                    Si type == "Sky" o type == "Flat":
                        filter<-buscarFiltroEnPath()
                        Si filter no esta vacio:
                            rellenar log con filter
                        Si no:
                            filter<-header.filter
                            rellenar log con filter
                        Fin Si
                    Si no:
                        rellenar log con "None"
                    Fin Si
                    date<-obtenerFechaAstronomica(header.date)
                    rellenar log con date
                    telescope<-obtenerTelescopio(header.telescope)
                    rellenar log con telescopio
                    Si type == "Sky":
                        Si telescope == "OSN":
                            object<-buscar objeto en ruta
                        Si no:
                            object<-header.object
                            rellenar log con object
                        Fin Si
```

Figura 13: creación log 2

```
                    Fin Si
                Si no:
                    rellenar log con "None"
                Fin Si
            Si no:
                    rellenar log con archivo no leido
            Fin sI
            Si no:
                rellenar log con archivo no leido
            Fin Si
            Fin Si
        Fin Para
    Fin Para

```

Figura 14: creación log 3

El siguiente paso será el de crear los *Master Bias*, para esto se leerá el log creado con anterioridad, se buscará cuáles son las imágenes de tipo *bias* y se obtendrá el telescopio, la fecha, el número de pixeles en el eje X, el número de pixeles en el eje Y, y el binning, con estos datos se irán agrupando las imágenes según su telescopio y fecha, y se irán anotando los distintos path, fechas y telescopios ya agrupados en un archivo de información, que se utilizará para la creación de los *Master Bias*. A continuación, se leerá el archivo de información con los diferentes path y se irán creando los *Master Bias*, además se creará un archivo log con la información de los *Master Bias* creados para dar cierta información al experto una vez ejecutada la aplicación.

```
funcion obtenerBias()
    pathCreationLog<-'log & info/'
    hoja<-Leer log("fits_calibrator_log.xlsx")
    listaFinal<-[]
    Para todas las filas de hoja:
        type<-obtener tipo
        Si type == "BIAS":
            pX<-obtener pixelesX
            pY<-obtener pixelesY
            date<-obtener fecha
            binning<-obtener binning
            telescope<-obtener telescopio
            lista<-[date,telescope,pX,pY,binning]
            Si lista no esta en listafinal:
                listaFinal.añadir(lista)
            Fin Si
        Fin Si
    Fin Para
    devolver listaFinal

```

Figura 15: obtener Bias

```
funcion AgruparBias(listaBias)
    pathCreationLog<-'log & info/'
    hoja<-Leer log("fits_calibrator_log.xlsx")

    Para todas las filas de hoja:
        type<-obtener tipo
        pX<-obtener pixelesX
        pY<-obtener pixelesY
        date<-obtener fecha
        binning<-obtener binning
        telescope<-obtener telescopio
        path<-obtener path
        datos<-[date,telescope,pX,pY,binning]
        Para todos los elementos de listaBias:
            Si type == "BIAS" y coinciden los demas parametros con algun elemento de listaBias:
                añadimos a listaBias el path
        Fin Para
    Fin Para
    devolver listaBias
```

Figura 16: agrupar Bias

```
funcion crearInfoBias(listaBias)
    pathCreationLog<-'log & info/'
    crearArchivoExcel('info_master_bias.xlsx')
    Para todas elementos de listaBias:
        rellenar log con elemento
    Fin Para
```

Figura 17: crear información Bias

```
funcion crearMasterBias()
    pathCreationLog<-'log & info/'
    CrearDirectorio(pathCreationLog)
    hoja<-Leer log(""info_master_bias.xlsx")
    CrearArchivoExcel("log_master_bias.xlsx")
    RellenarCabecerasExcel
    pathCreationFiles<-"resultados/master_bias/"
    masterBias<-[]
    Para todas las filas de hoja:
       masterBias<-[]
       pX<-obtener pixelesX
       pY<-obtener pixelesY
       date<-obtener fecha
       binning<-obtener binning
       telescope<-obtener telescopio
       name<-"master_bias_"+date+"_"+telescope+".fits"
       Para todas las columnas de la fila:
           path<-obtener path
           ccd<-(fits.getdata(path),fits.getheader(path))
           masterBias.añadir(ccd)
       Fin Para
       combiner<-combinar(masterBias)
       master_bias<-combiner.mediana
       master_bias.header<-masterBias.header
       master_bias.header.filename<-name
       master_bias.header.history<- historia de la creacion del archivo
       escribirArchivoFit(master_bias)
       rellenar log con datos
    Fin Para
```

Figura 18: crear master Bias

Realizado esto se crearán los *Master Flat* de la misma manera que los *Master Bias*, leyendo el log de todas las imágenes y obteniendo la fecha y telescopio, además

aquí se deberá obtener el filtro que se usó para tomar la imagen *flat*, ya que también será una de las características que se usarán para agrupar las distintas imágenes *flat*. Una vez agrupadas, se procederá de la misma manera que con los *Master Bias*, se leerán todos los path de las imágenes *flat* para la creación de los *Master Flat* pero con un unas características añadidas, una vez agrupadas las imágenes *flat* habrá que restarle el *Master Bias* correspondiente a cada una de las imágenes *flat*, el cual será asociado por medio de la fecha y el telescopio, si no se encontrara ningún *Master Bias* con la misma fecha que un grupo de imágenes Flat, se debería buscar un *Master Bias* en los días posteriores o anteriores a la fecha astronómica de las imágenes *flat*, eligiendo la más cercana a dicha fecha. Los descrito anteriormente es otro de los problemas encontrados, ya que se ha de ir buscando un *Master Bias* en el siguiente orden: día siguiente, día anterior, dos días después, dos días antes…, así sucesivamente hasta un número de días que será determinado por el usuario. A continuación, se deberán normalizar todas las imágenes *flat*, una vez hecho todo esto se procederá a la creación de los *Master Flat*. Tal y como se hizo con los *Master Bias*, se creará un log con la información de los *Master Flat* para su posterior uso.

```
funcion obtenerFlat()
    pathCreationLog<-'log & info/'
    hoja<-Leer log("fits_calibrator_log.xlsx")
    listaFinal<-[]
    Para todas las filas de hoja:
        type<-obtener tipo
        Si type == "FLAT":
            pX<-obtener pixelesX
            pY<-obtener pixelesY
            date<-obtener fecha
            binning<-obtener binning
            telescope<-obtener telescopio
            filter<-obtener filtro
            lista<-[date,telescope,filter,pX,pY,binning]
            Si lista no esta en listafinal:
                listaFinal.añadir(lista)
            Fin Si
        Fin Si
    Fin Para
    devolver listaFinal

```

Figura 19: obtener Flat

```
funcion AgruparFlat(listaFlat)
    pathCreationLog<-'log & info/'
    hoja<-Leer log("fits_calibrator_log.xlsx")

    Para todas las filas de hoja:
        type<-obtener tipo
        pX<-obtener pixelesX
        pY<-obtener pixelesY
        date<-obtener fecha
        binning<-obtener binning
        telescope<-obtener telescopio
        filter<-obtener filtro
        path<-obtener path
        datos<-[date,telescope,filter,pX,pY,binning]
        Para todos los elementos de listaBias:
            Si type == "FLAT" y coinciden los demas parametros con algun elemento de listaFlat:
                añadimos a listaFlat el path
        Fin Para
    Fin Para
    devolver listaFlat
```

Figura 20: agrupar Flat

```
funcion crearInfoFlat(listaFlat)
    pathCreationLog<-'log & info/'
    crearArchivoExcel('info_master_flat.xlsx')
    Para todas elementos de listaFlat:
        rellenar log con elemento
    Fin Para
```

Figura 21: crear información Flat

```
funcion crearMasterFlat()
    pathCreationLog<-'log & info/'
    CrearDirectorio(pathCreationLog)
    hojaFlat<-Leer log("info_master_flat.xlsx")
    hojaBias<-Leer log("log_master_bias.xlsx")
    CrearArchivoExcel("log_master_flat.xlsx")
    RellenarCabecerasExcel
    pathCreationFiles<-"resultados/master_flat/"
    masterFlat<-[]
    Para todas las filas de hojaFlat:
        masterFlat<-[]
        pX<-obtener pixelesX
        pY<-obtener pixelesY
        date<-obtener fecha
        binning<-obtener binning
        telescope<-obtener telescopio
        filter<-obtener filtro
        name<-"master_bias_"+date+"_"+telescope+".fits"
        Mientras no se haya asignado Bias y numero de dias sea menor que 5:
            Para todas las filas de hojaBias:
                pXBias<-obtener pixelesX
                pYBias<-obtener pixelesY
                dateBias<-obtener fecha
                binningBias<-obtener binning
                pathBias<-obtener path
                telescopeBias<-obtener telescopio
```

Figura 22: crear Master Flat 1

```
                Si pX == pXBias y pY == pYBias y date == dateBias y binning == binningBias y telescope == telescopeBias:
                biasAsignado<-true
                    Para todas las columnas de la fila:
                        path<-obtener path
                        ccdF<-(fits.getdata(path),fits.getheader(path))
                        ccdB<-(fits.getdata(pathBias),fits.getheader(pathBias))
                        ccd<-ccdF-ccdB
                        cdd<-cdd/mediana(cdd)
                        masterFlat.añadir(ccd)
                    Fin para
                Fin SIF
            combiner<-combinar(masterFlat)
            master_flat<-combiner.mediana
            master_flat.header<-masterFlat.header
            master_flat.header.filename<-name
            master_flat.header.history<- historia de la creacion del archivo
            escribirArchivoFit(master_flat)
            rellenar log con datos
            Fin para
        Si biasAsignado == False:
            Aumentar date o reducir date
        Fin Si
        Fin Mientras
    Fin para
```

Figura 23: crear Master Flat 2

Por último, después de realizar todos los pasos previos se procede a la calibración de las imágenes, para ello se volverá a leer el log que se creó inicialmente, para obtener todas las imágenes de tipo Sky, así como sus parámetros asociados: fecha, filtro, telescopio, numero de pixeles en el eje X, numero de pixeles en el eje Y, y binning. Obtenida toda esa información primero se deberá restar el *Master Bias* correspondiente que se asocie por cercanía temporal, el cual se buscará en el log de los *Master Bias*, se asociarán imágenes con *Master Bias* sí coinciden en telescopio, fecha, numero de pixeles en el eje X, numero de pixeles en el eje Y, y binning, y tal y como pasaba a la hora de crear los *Master Flat*, se deber buscar un *Master Bias* en los días anteriores y posteriores a la fecha astronómica de la imagen. Una vez encontrado se procederá a restar el *Master Bias* a la imagen de ciencia.

A continuación, se buscará un *Master Flat* que pueda ser asociado a la imagen de ciencia, buscando en el log de *Master Flat* uno que coincida en fecha, telescopio, filtro, numero de pixeles en el eje X, numero de pixeles en el eje Y, y binning. Al igual que en el paso anterior, hay que buscar de nuevo un *Master Flat* los días anteriores y posteriores a la fecha astronómica de la imagen. También se creará un log de todas las imágenes calibradas para que el experto tenga información sobre las imágenes y pueda corroborar que la calibración ha sido correcta.

Si la imagen no se pudiera calibrar debido a que no se encuentre ningún *Master Flat* o ningún *Master Bias* la imagen se clasificaría en el directorio correcto según el objeto celeste al que hace referencia, al igual que el resto de imágenes calibradas, pero se le añadiría al final del nombre "*_notCalibrated*" y no se guardaría en el log de imágenes calibradas.

```
funcion calibrarImagenes()
    pathCreationLog<-'log & info/'
    pathCreationFiles<-'resultados/calibrated_images/'
    CrearDirectorio(pathCreationLog)
    hoja<-Leer log("fits_calibrator_log.xlsx")
    hojaFlat<-Leer log("log_master_flat.xlsx")
    hojaBias<-Leer log("log_master_bias.xlsx")
    CrearArchivoExcel("fits_calibrator_log.xlsx")
    RellenarCabecerasExcel
    Para todas las filas de hoja:
        object<-obtener object
        pX<-obtener pixelesX
        pY<-obtener pixelesY
        date<-obtener fecha
        binning<-obtener binning
        telescope<-obtener telescopio
        filter<-obtener filtro
        name<-obtener name
        fileRead<-obtener fileRead
        path<-obtener path
        Si type == "Sky" y fileRead != "*" y path no contiene "_c":
            Mientras no se haya asignado Bias y numero de dias sea menor que 5:
                Para todas las filas de hojaBias:
                    pXBias<-obtener pixelesX
                    pYBias<-obtener pixelesY
                    dateBias<-obtener fecha
                    binningBias<-obtener binning
                    pathBias<-obtener path
                    telescopeBias<-obtener telescopio
                    Si pX == pXBias y pY == pYBias y date == dateBias y binning == binningBias y telescope == telescopeBias:
                        ccd<-(fits.getdata(path),fits.getheader(path))
                        ccdB<-(fits.getdata(pathBias),fits.getheader(pathBias))
                        ccd<-ccdF-ccdB
                        biasAsignado<-true
```

Figura 24: calibrar imágenes 1

```
                biasAsignado<-true
                Mientras no se haya asignado Flat y numero de dias sea menor que 5:
                    Para todas las filas de hojaFlat:
                        pXFlat<-obtener pixelesX
                        pYFlat<-obtener pixelesY
                        dateFlat<-obtener fecha
                        binningFlat<-obtener binning
                        filterFlat<-obtener filtro
                        pathFlat<-obtener path
                        telescopeFlat<-obtener telescopio
                        Si pX == pXFlat y pY == pYFlat y date == dateFlat y binning == binningFlat y telescope == telescopeFlat y filtro == filtroFlat:
                            ccdF<-(fits.getdata(pathFlat),fits.getheader(pathFlat))
                            image<-ccd/ccdF
                            image.header<-ccd.header
                            image.header.filename<-name
                            image.header.history<- historia de la creacion del archivo
                            flagAsignado<-True
                            pathCreationImage<-'resultados/calibrated_images/'+object+"/"
                            escribirArchivoFit(image)
                            rellenar log con datos de image
                        Fin Si
                    Fin Para
                    Si flagAsignado == False:
                        Aumentar date o reducir date
                    Fin Si
                Fin Mientras
            Fin Si
        Fin Para
```

Figura 25: calibrar imágenes 2

```
                Fin Para
                Si biasAsignado e==s False:
                    Aumentar date o reducir date
                Fin Si
            Fin Mientras
            Si bias Asignado == False o flatAsignado == False:
                escribirArchivoFit(image+"_notCalibrated")
            Fin Si
        Fin Si
    Fin Para
```

Figura 26: calibrar imágenes 3

El menú principal de uso de la aplicación es muy sencillo, si se le pasan 2 parámetros por el CMD, el directorio y cualquier otro parámetro, la aplicación calibrará las imágenes del directorio, pero no creará nuevos archivos *Master Bias* ni *Master Flat*, solamente analizará los ya creados para ver si se han añadido a mano nuevos o se han borrado y con esos creará un log el cual usará para calibrar las imágenes del directorio. En cambio, si se le pasa un solo parámetro, el directorio, analizará todas las imágenes y creará los *Master Bias* y *Master Flat* correspondientes, a continuación, calibrará todas las imágenes.

```
funcion crearMasterBias()
    pathCreationLog<-'log & info/'
    CrearDirectorio(pathCreationLog)
    CrearArchivoExcel("log_master_flat.xlsx")
    CrearArchivoExcel("log_master_bias.xlsx")
    RellenarCabecerasExcel
    Para todos los elementos de ruta:
        Para todos los archivos:
            Si archivo == Fit:
                archivo_fit<-abrirArchivo(archivo)
                header<-archivo.header
                type<-header.tipo
                time<-header.tiempoDeExposicio
                telescope<-header.telescope
                date<-header.date
                pX<-header.pixelesX
                pY<-header.pixelesY
                Intentar:
                        binning<-header.naxis
                Excepcion:
                    Intentar:
                        binning<-header.binning
                    Excepcion:
                        binning<-"Unknown"
                Si time == "0.0"
                    type<-"BIAS"
                Si type == "BIAS":
                    rellenar log con datos
                Si no:
                    Si type == "FLAT":
                        filter<-obtenerFiltroDelNombre
                        rellenar log con datos
                    Fin si
                Fin Si
                Fin si
            Fin Si
        Fin Para
    Fin Para
```

Figura 27: analizar Master

```
def menu():
    Si numVariables == 2:
        programaCompleto(directorio)
    Si no:
        calibrarSolo(directorio)
    Fin Si
```

Figura 28: menú principal

```
def programaCompleto(ruta):
    createLogs(ruta)
    listaBias=groupBias()
    listaBiasPath=groupPathBias(listaBias)
    createInfoMasterBias(listaBiasPath)
    createMasterBias()
    listaFlat=groupFlat()
    listaFlatPath=groupPathFlat(listaFlat)
    createInfoMasterFlat(listaFlatPath)
    createMasterFlat()
    calibrate()
```

Figura 29: programa completo

```
def calibrarSolo(ruta):
    createLogs(ruta)
    analizarMaster()
    calibrate()
```

Figura 30: calibrar solo

14. Librerías auxiliares usadas

1. Pathlib: Esta librería servirá para crear el directorio de carpetas para poder guardar los archivos sé que irán creando durante la ejecución de la aplicación.
2. Numpy: La librería Numpy es bastante importante en este proyecto, ya que se usará para hacer la división de imagen entre el *Master Flat*, las imágenes *flat* entre el *Master Bias* y hacer la normalización de las imágenes *flat*, además servirá para elegir el tipo de número, float30, int16… que tendrán las matrices de datos de las imágenes creadas.
3. Openpyxl: Gracias a esta librería es posible leer archivos Excel y manejarlos a nuestro gusto, obteniendo los distintos valores de cada celda.
4. Xlsxwriter: Con esta librería es posible escribir archivos Excel desde cero, gracias a esto podremos crear los distintos archivos log y de información que nos simplificarán las cosas a la hora de implementar el código de la aplicación.
5. Os: Esta librería es muy simple, aunque tiene una funcionalidad muy grande, ya que gracias a ella podremos obtener todos los path, archivos y rutas de esos archivos separados en listas, de una ruta especifica.
6. Ccdproc: Con esta librería podremos hacer las distintas cuentas que nos hace falta para calibrar las imágenes, a parte podremos combinar las distintas imágenes *flat* y *bias* para crear los *Master Flat* y *Master Bias*.

15. Valoración del experto

La calibración de imágenes astronómicas obtenidas con instrumentación asociada a distintos telescopios es un paso imprescindible que hay que realizar previamente a su reducción científica.

A pesar de ser un sencillo algoritmo matemático el que hay que aplicar para la calibración, el proceso se complica cuando son grandes series de datos lo que hay que calibrar, tomadas con diferentes telescopios y en distintas épocas, ya que los estándares utilizados por los equipos de diferentes grupos de investigación, con distinta tecnología y a lo largo de largos intervalos temporales no siempre tienen los mismos criterios y en consecuencia a veces la tarea de localización de los diferentes grupos de ficheros de calibración hace que un proceso sencillo se convierta en manual, laborioso y que puede incrementar la planificación temporal de cualquier reducción científica de datos.

Existe software de distinto tipo que es capaz de realizar la calibración de forma automática, pero ninguno de ellos cumple con las especificaciones requeridas al trabajo desarrollado en este TFG, en el sentido que no se adaptan a la heterogeneidad de datos con los que pueden contar los grupos de investigación.

Hemos evaluado el software desarrollado en el presente TFG con imágenes de diferentes telescopios, instrumentos, tecnología y años y se ha comprobado no sólo que la aplicación de los algoritmos finales de calibración los realiza de forma correcta, sino que también el sistema se adapta a una gran variedad de posibilidades que puede encontrarse debido a la ausencia de un estándar único que identifique unívocamente cada uno de los ficheros involucrados en una reducción científica de las imágenes con las que se ha probado.

El software ha sido probado con imágenes antiguas y nuevas de telescopios situados en distintos observatorios como el 1.5m de Sierra Nevada, 1.23m de Calar Alto en Granada, 2.5m Nordic Telescope de la Palma y algunos otros, cumpliendo los requerimientos y es capaz de identificar de forma correcta, rápida, eficaz y agrupar de forma correcta, tanto los diferentes ficheros involucrados en el proceso de calibración como los distintos objetos celestes que han sido observados en las diferentes

imágenes. El software proporciona los datos ya calibrados y agrupados por los nombres de los objetos observados, lo que lo hace muy útil para acelerar una tarea que a veces no puede hacerse sin abrir con programas de visualización, las diferentes imágenes.

Además de la utilidad que ya tiene, el sistema permite ser escalado en un futuro de manera que se puedan añadir de forma sencilla, posibles casos de otros telescopios que no se hayan contemplado en el presente trabajo. El ser multiplataforma es un punto a favor ya que no siempre todos los miembros de los distintos equipos de investigación poseen los mismos sistemas operativos, en este caso, eso no sería un problema.

En un futuro, se podrían realizar mejoras tales como el añadido de nuevas características de imágenes no contempladas, diferentes filtros de los instrumentos no añadidos en esta primera versión o se podría usar en cámaras no refrigeradas con Nitrógeno Líquido sino con métodos electrónicos de efecto Peltier, las cuales no han sido contempladas en este trabajo ya que la inmensa mayoría de los instrumentos situados en observatorios profesionales no se refrigeran electrónicamente. Del mismo modo, aunque a priori no es necesario, añadir una interfaz gráfica podría ser interesante de cara a un futuro.

Se valora muy positivamente el trabajo realizado en este TFG que a buen seguro se va a usar en grupos de investigación como el nuestro de manera habitual a partir de ahora y va a permitir tanto analizar nuevos datos como reanalizar antiguos que aunque ya se hicieron y publicaron en el pasado, ahora se pueden evaluar con distintas técnicas nuevas no usadas en su momento y que serán objeto de publicaciones científicas en revistas de alto impacto y presentadas en congresos internacionales a los que habitualmente se presenta este tipo de ciencia.

Granada, 5 septiembre de 2021

Nicolás Morales

Instituto de Astrofísica de Andalucía.

16. Manual de instalación y uso de la aplicación

Para que los usuarios puedan usar la aplicación deberán tener instalado Python pudiendo descargarlo de la página oficial. Una vez instalado, para la ejecución se deberá abrir el CMD y moverse al directorio donde este el archivo .py, hecho esto solo habrá que ejecutarlo con el comando:

Python [archivo.py] [directorio] [parámetro]

[archivo.py] Nombre del archivo .py

[directorio] Directorio del que se van a analizar todas las imágenes. fit

[parámetro] parámetro opcional para que el programa se ejecute en su versión simple.

Dado que es posible una ejecución de la aplicación sin que genere nuevos archivos *Master Bias* y *Master Flat* lo único que se deberá hacer es introducir un parámetro opcional, sea el que sea hará posible esta ejecución más simple del programa.

Realizado todo esto se generarán 2 carpetas nuevas en el directorio donde este el archivo .py: una carpeta de log & info donde se encontrarán todos los logs creados y una carpeta llamada resultados que contendrá a su vez 3 carpetas: una con las imágenes calibradas y ordenadas según el objeto celeste observado, otra con los archivos *Master Bias* y otra con los *archivos Master Flat.*

El programa se puede ejecutar en cualquier plataforma y es funcional, el sistema usado por el experto es Linux y el funcionamiento es correcto también ahí.

17. Conclusiones

Para concluir este proyecto vamos a realizar una comparación entre los objetivos y su grado de consecución.

- **Realizar la calibración de imágenes astronómicas**

  - Este objetivo se ha alcanzado, pues se ha realizado una aplicación que funciona correctamente, tal y como hemos podido ver en la "Valoración del experto".

- **Conocer las diferentes tecnologías y estándares que han ido surgiendo en los últimos años**

  - Se ha alcanzado este objetivo ya que hemos podido comprobar que, aunque existe una tecnología para realizar la calibración de imágenes, esta tiene que ser manual. en cuanto a los estándares, se ha conocido el estándar FITS y su funcionamiento.

- **Diseñar e implementar una aplicación que para automatizar el proceso de calibrado de las imágenes.**

  - Este objetivo se ha alcanzado pues se ha creado una aplicación con la que conseguimos automatizar dicho proceso. (Ver anexos 1)

17.1. Valoración global.

Considero que este proyecto me ha aportado una serie de conocimientos, tanto astronómicos como conocimientos del lenguaje Python y me ha resultado:

- **Motivador:** sido muy motivador pues el estándar estudiado y el uso que se puede dar a la aplicación es actual y de utilidad.
- **Estimulante:** pues conforme se iba conociendo más acerca del formato FIT, me permitía tener más conocimientos para el objetivo del proyecto.
- **Complejo:** ha resultado complejo pues se ha tenido que estudiar un estándar específico de imágenes para saber cómo se procesan dichas imágenes y poder manejarlas.
- **Desafiante:** se ha realizado una amplia búsqueda de información. Esta información no ha sido fácil de encontrar, pero gracias a estas búsquedas he aprendido no solo acerca del estándar FITS, sino también sobre librerías y uso del lenguaje de programación Python.

18. Bibliografía

[1] *Presentación | Instituto de Astrofísica de Andalucía - CSIC*. (s. f.). IAA. Recuperado de https://www.iaa.csic.es/presentacion

[2] *Presentación | observatorio de sierra nevada*. (s. f.). OSN. Recuperado de https://www.osn.iaa.csic.es/general/presentacion

[3] *Introducción*. (s. f.). CAHA. Recuperado de http://www.caha.es/es/sobre-caha/introducci%C3%B3n

[4] PALAUGEA COMUNICACIÓN, S.L. (2021, 6 mayo). *FITS, el formato de archivo que utiliza la NASA para sus imágenes astronómicas*. Gràffica. https://graffica.info/fits-formato-de-archivo/

[5] IAUFWG & International Astronomical Union. (2016, Julio). *Definition of the Flexible Image Transport System (FITS)*. https://fits.gsfc.nasa.gov/standard40/fits_standard40aa-le.pdf

[6] WELLS, D. C., GREISEN, E. W., & HARTEN, R. H. (1981). FITS: A FLEXIBLE IMAGE TRANSPORT SYSTEM. *Astronomy & Astrophysics Supplement Series.*, 363.

[7] AAVSO. (2014). *Guía de Fotometría CCD de la AAVSO Versión 1.1* (Versión 1.1). https://www.aavso.org/sites/default/files/publications_files/ccd_photometry_guide/CCDPhotometryGuide.pdf

[8] Howell, S. (2006). *Handbook of CCD Astronomy* (Vol. 4). Cambridge University Press. https://doi.org/10.1017/CBO9780511807909.006

[9] ScrumGuides, Schwaber, K., & Sutherland, J. (2020, noviembre). *La Guía Scrum La Guía Definitiva de Scrum: Las Reglas del Juego*. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

[10] *Unidades*. (2014, 4 diciembre). Fundamentos de Ingeniería de Software. https://fundingsoft.wordpress.com/unidades/

[11] Carranza, A. (s. f.). *¡Conoce el modelo en cascada y escala tus proyectos de software a pasos agigantados!* https://www.crehana.com. https://www.crehana.com/es/blog/desarrollo-web/modelo-en-cascada/

[12] School, T. (2021, 29 abril). *La historia de Python. Las versiones de un lenguaje único*. Tokio School. https://www.tokioschool.com/noticias/historia-python/

[13] Ruiz, M. (2020, 25 junio). *Python VS Java: Comparativa 2019*. OpenWebinars.net. https://openwebinars.net/blog/python-vs-Java/

[14] GeeksforGeeks. (2020, 31 marzo). *Difference between Python and C++*. https://www.geeksforgeeks.org/difference-between-python-and-c/

[15] Toro, L. (2017, 15 septiembre). *Anaconda Distribution: La Suite más completa para la Ciencia de datos con Python*. Desde Linux. https://blog.desdelinux.net/ciencia-de-datos-con-python/?_gl=1%2A177efa4%2A_ga%2AYW1wLWw2R3pqRno3ZVhGcVJTUU5Zelh2dG5qQlhjaGRFcVl2NDE5Ym1KZzdHbGFqQ25uU2RrUzJsMU0xOFlHYUxENmw.

[16] *Project Jupyter*. (s. f.). Home. Recuperado de https://jupyter.org/

[17] *Siril - FreeAstro*. (s. f.). Siril. Recuperado de https://free-astro.org/index.php/Siril

[18] *Astropy*. (s. f.). Astropy. Recuperado de https://www.astropy.org/

[19] EDP Sciences. (2013). The Astrophysical Multipurpose Software Environment. *Astronomy & Astrophysics*, 1. https://www.aanda.org/articles/aa/pdf/2013/09/aa21252-13.pdf

[20] *Astropy*. (s. f.). Astropy. Recuperado de https://www.astropy.org/about.html

[21] *Astropy examples*. (s. f.). Example Gallery. https://docs.astropy.org/en/stable/generated/examples/index.html

19. Anexos.

Junto con la memoria se va a entregar:

1. Código fuente de la aplicación en un archivo .py