Fotometría absoluta y brillo de fondo de cielo con AstMon-UCM

Trabajo académicamente dirigido por:

Jaime Zamorano - Dpto. Astrofísica y CC. De la Atmósfera
José Luis Contreras - Dpto. Física Atómica, Molecular y Nuclear

Miguel Nievas Rosillo

31 de octubre de 2012

Abstract

All-Sky images have proven to be a powerful tool for determining the astronomical sky quality, nowadays their use is spreading both for planned and existing observatories. The Universidad Complutense de Madrid's astronomical observatory (Observatorio UCM) is an urban observatory located at Campus of the University. It provides an excepcional laboratory to study the effects of human activity such as light pollution, air pollutants and aerosol concentration in the astronomical quality of the sky. In order to study these effects, we set up two years ago the all-sky monitor AstMon-UCM.

During the last year we have been developing an open-source software package to automate the analysis of all-sky cameras images. It characterizes sky quality for astronomical observations through parameters like Sky Brightness and Atmospheric Extinction. We have applied it to images from the AstMon-UCM, other all-sky monitors and common digital cameras equipped with fisheye lenses. Results have been compared with data derived from SQM photometers. These tools open the door to several exciting possibilities.

Resumen

Las imágenes de todo el cielo han demostrado ser una potente herramienta para determinar la calidad astronómica del cielo, y su uso se está extendiendo hoy en día tanto ubicaciones de futuros observatorios como en los ya existentes. El observatorio astronómico de la Universidad Complutense de Madrid (Observatorio UCM) es un observatorio urbano situado en el Campus Universitario que constituye un excepcional laboratorio para estudiar efectos de la actividad humana como la contaminación lumínica, la polución del aire y la concentración de aerosoles en la calidad astronómica del cielo. Para estudiar estos efectos, instalamos hace dos años el monitor astronómico AstMon-UCM.

Durante el pasado año hemos desarrollado un paquete de programas de código abierto para automatizar el análisis de las imágenes de cámaras de todo el cielo, caracterizando la calidad del cielo para observaciones astronómicas a través de parámetros como el Brillo de Fondo de Cielo o la Extinción Atmosférica. Este paquete se ha aplicado tanto en imágenes de AstMon-UCM como de otros monitores de todo el cielo y cámaras digitales equipadas con objetivos ojo de pez, comparando los resultados obtenidos con datos provenientes de fotómetros SQM. Estas herramientas abren la puerta a numerosas y excitantes posibilidades.

Agradecimientos

Quiero agradecer en primer lugar a los directores de este trabajo, Jaime Zamorano Calvo (Dpto. de Astrofísica y CC. de la Atmósfera) y José Luis Contreras González (Dpto. de Física Atómica, Nuclear y de Partículas) por el apoyo, motivación, consejos y ayuda prestada en todo este tiempo, además de por haber puesto en contacto nuestro grupo de trabajo con los responsables de AstMon-Izaña; a Pablo Ramírez Moreta por introducirme el funcionamiento de AstMon, la inestimable ayuda prestada (sobre todo en los comienzos), y por haber dejado en marcha todo el sistema necesario para el funcionamiento de AstMon-UCM; y a Francisco Ocaña González y Alejandro Sánchez de Miguel por su apoyo puntual y "evaluación" y posteriores consejos sobre los resultados que se iban obteniendo.

Índice

1. Introducción

El observatorio astronómico de la Universidad Complutense de Madrid (Observatorio UCM) se encuentra enclavado en Ciudad Universitaria. Es por lo tanto un observatorio urbano afectado por la contaminación lumínica.

Con el fin de monitorizar la calidad astronómica del cielo sobre Madrid y estudiar los efectos de la contaminación lumínica en el brillo de fondo de cielo se adquirió en 2010 un dispositivo astronómico (AstMon-UCM) que ha estado tomando datos y determinando parámetros astronómicos desde entonces.

En el Trabajo Académicamente Dirigido de Pablo Ramírez Moreta ("Brillo de fondo de cielo con AstMon-UCM"[Mor11]) del curso 2010-2011 se consiguió que AstMon-UCM funcionara de forma autónoma y se apuntaron los trabajos necesarios para lograr un mejor aprovechamiento del monitor. Durante este curso hemos contribuido a la puesta a punto de la cámara para mejorar su rendimiento científico. Se han establecido rutinas de uso y almacenamiento de datos en bruto para posterior tratamiento.

Estos pasos eran necesarios para abordar los siguientes objetivos:

1. Desarrollo de una serie de rutinas que automaticen el proceso de análisis de las imágenes AllSky que toma AstMon-UCM.

2. Comparación entre AstMon-UCM y otros dispositivos más sencillos de medida de brillo de fondo de cielo.

3. Estudio del uso de cámaras fotográficas como monitores de cielo.

En esta memoria se muestran los trabajos realizados y los resultados obtenidos. Se describe sucintamente el monitor astronómico AstMon-UCM en la sección 2 para ayudar a situar el tema de investigación. En la sección 3 se muestra el desarrollo de los programas para realizar la calibración de AstMon-UCM y la fotometría que permite obtener los parámetros astronómicos de calidad del cielo y los mapas de brillo de fondo de cielo. Otros instrumentos más sencillos han sido probados junto a AstMon-UCM y los resultados se describen en la sección 4. La sección 5 se dedica a los resultados más importantes y por último en la sección 6 y 7 se listan las conclusiones y los trabajos pendientes de realizar.

2. Monitor astronómico del observatorio UCM

AstMon, del acrónimo en inglés AllSky Transmission MONitor, es un instrumento científico cuyo objetivo es realizar un monitoreo continuo del cielo nocturno de un cierto lugar. Como producto de dichas observaciones, este dispositivo estima la medida de brillo de fondo de cielo y la extinción. Dado que se encuentra continuamente recogiendo datos, permite además hacer estudios en el tiempo de estos parámetros, lo que lo convierte en un instrumento idóneo para poder evaluar la calidad astronómica del cielo de un cierto lugar.

Otras funciones de AstMon son la generación de mapas de brillo de fondo de cielo (que permiten visualizar la variación espacial de la iluminación del cielo nocturno) y la generación de mapas de nubes.

El desarrollo del aparato fue llevado a cabo por la empresa iTec Astronomica S.L, existiendo en España 4 ejemplares de este dispositivo: AstMon-UCM, AstMon-Doñana, AstMon-OT (Izaña, Tenerife) y AstMon-CAHA (en el observatorio astronómico hispanoalemán de Calar Alto).

Figura 1: AstMon-Doñana

Figura 2: AstMon-CAHA

AstMon-UCM se encuentra en la posición más elevada de la Facultad de Ciencias Físicas de la Universidad Complutense de Madrid, esto es, en la azotea del edificio, junto a las cúpulas, los sistemas de detección de bólidos y estaciones meteorológicas y forma parte del equipo del Observatorio UCM.

El sistema funciona de manera autónoma tomando imágenes científicas en los filtros astronómicos en la secuencia programada con los tiempos de exposición seleccionados. Almacena las observaciones y calcula para cada una de ellas los coeficientes de extinción a lo largo de la noche. Con estos resultados proporciona casi en tiempo real los mapas de brillo de fondo de cielo. Se listan a continuación las salidas del programa interno de AstMon-UCM.

- Ficheros con las imágenes en las bandas B. V, R. En formato astronómico estándar FITS. Tamaño de 2500×2500 píxeles. Actualmente la secuencia programada es que tome imágenes en series espaciadas $200s$ con tiempos de exposición de $40s$ en cada filtro (en total, cada serie se completa en $320s$. Cada noche se almacenan $1{,}35Gb/hora$.

- Mapas de brillo de fondo de cielo en cada banda. En formato FITS y en unidades de $mag/arcsec^2$
- Ficheros de extinción, para cada noche. Contiene valores de extinción para cada estrella suponiendo una constante instrumental fija.
- Ficheros con la medida de brillo de fondo de cielo en posiciones seleccionadas (dentro de la interfaz del programa), para cada noche y cada filtro un fichero de texto.

Figura 3: Localización de AstMon-UCM en la azotea de la facultad de Físicas. Al fondo se ve la cúpula Oeste del observatorio. Coordenadas: 40º.450941 N; 3º.726065 O

Una descripción completa de AstMon-UCM puede encontrarse en la memoria del trabajo de Pablo Ramírez Moreta[Mor11]. Se mencionan a continuación sus características ópticas fundamentales,

Cámara	QSI 583 WS (refrigerada y con rueda de filtros)
CCD	KAF 8300. Tamaño de píxel: $5{,}4 \times 5{,}4 \mu m$.
Objetivo	Sigma 4.5mm F2.8 EX DC HSM Circular Fisheye
Filtros fotométricos	Sistema Johnson: B,V,R

2.1. Elementos ópticos

- Objetivo ojo de pez: f=4.5mm F/2.8 Sigma EX DC HSM Circular Fisheye

  Este objetivo fue seleccionado por la gran calidad óptica que ofrece, libre prácticamente de distorsión[1].

[1]Medidas realizadas por J. Aceituno muestran que esta distorsión (vease [iAS11]), de existir, es pequeña y se encuentra en límites muy tolerables. Esta observación se discutirá, con imágenes test, a lo largo de este documento.

El ojo de pez seleccionado utiliza una proyección ZEA (*Zenithal Equal Area*), muy útil para los objetivos del instrumento ya que se trata de una proyección que conserva áreas como más adelante veremos, de modo que el flujo medido por píxel es directamente proporcional al correspondiente por $arcsec^2$ para cualquier posición en la imagen y no es necesario efectuar la corrección que tenga en cuenta la diferente escala espacial.

- Filtros:

  AstMon-UCM trabaja, desde que fue instalado, con 3 filtros del sistema Johnson-Cousins: B, V, R.

Filtro	$\lambda_0(\mathring{A})$	$\Delta\lambda(\mathring{A})$
B	4361	890
V	5448	840
R	6407	1580

Cuadro 1: Sistema UBVRI de Johnson-Cousins

- Cúpula semiesférica transparente. Fabricada en metacrilato. Su función es proteger el objetivo de los agentes meteorológicos. Periódicamente se hace necesario limpiar este elemento, ya que tiende a acumular bastante polvo y a mancharse en días de lluvia o noches en las que la humedad es muy elevada y condensa. El objetivo se encuentra aproximadamente en el centro de curvatura de la semiesfera para evitar en lo posible distorsiones de este elemento óptico.

2.2. Cámara

AstMon-UCM viene equipado con una cámara CCD astronómica QSI 583ws de la casa Quantum Scientific Imaging. El obturador es mecánico, con un rango de exposiciones de 0.03s a 240 minutos.

La refrigeración se realiza mediante un módulo Peltier capaz de regular la temperatura de trabajo en intervalos de 0.1ºC desde $T = 0°C$ hasta $T = -40°C$. De acuerdo con [ea11], la temperatura de trabajo de AstMon son -15ºC. Es de suponer que la elección de esta temperatura no es arbitraria, sino que supone un equilibrio entre reducción de ruido de origen térmico, capacidad de los sistemas anti-condensación de evitar que la humedad relativa crezca y se forme rocío (o incluso hielo) sobre el sensor y posibilidades reales de la célula Peltier (seguramente en verano no sea capaz de alcanzar los -40ºC teóricos[2], de modo que se busca una temperatura aceptable que se pueda alcanzar con facilidad casi todo el año).

El sensor es un Kodak KAF-8300, que tiene las siguientes características[3] de interés para AstMon:

[2]Los sistemas de refrigeración Peltier consiguen bajadas de temperatura diferenciales, esto es, de una cierta cantidad de grados respecto a la temperatura ambiente. Son en general sistemas poco eficientes, debiendo disipar una cantidad de calor en el foco caliente mucho mayor al correspondiente calor que absorben en el foco frío. El Peltier que monta la QSI 583ws, de acuerdo a las especificaciones que figuran en la web de QSI: `http://qsimaging.com/583-details.html` puede disminuir la temperatura entre 38 y 45ºC.

[3]extraídas de `http://qsimaging.com/583-details.html`

Tamaño del pixel	$5{,}4 \times 5{,}4 \mu m$
Ganancia	$0{,}5e^-$ y $1{,}1e^-$
Rango dinámico	16 bits
Ruido de lectura:	$8e^-$ típicamente
Corriente de oscuridad	$1e^-/pix/s$ (a $0°C$)
Tiempo de lectura	$21s$, depende del ordenador
Conectividad	USB 2.0.

Cuadro 2: Características del CCD KAF-8300

La rueda de filtros motorizada tiene 5 posiciones. Utiliza filtros de 1.25", actualmente 3 posiciones en uso y dos vacías.

2.3. Electrónica de control y sistemas antihumedad

- Diferencial y magneto-térmico: su función es proteger frente a subidas de tensión en la red.
- Reloj-programador: En origen se utilizaba para "hibernar" el sistema. Tras tener problemas con la regleta de enchufes e intercambiar correos con el diseñador de AstMon (J. Aceituno) para volver a ajustar el reloj, se dejó en posición de siempre encendido.
- Ventiladores:

  Uno de mayor tamaño en la caja de AstMon que salta cuando se superan los $20°C$ de temperatura en el habitáculo, su función es evitar la entrada de aire húmedo que pueda dañar los componentes electrónicos.

  Otro de menor tamaño que junto a una resistencia introduce aire caliente en el interior de la cúpula de metacrilato para evitar la condensación.
- Sistemas pasivos para reducir la humedad: se han introducido 2 vasijas con compuestos diseñados para reducir la humedad dentro del habitáculo (por ejemplo, silica gel y/o desecantes por precipitación). Complementan a la caja con Zeolita en la base del habitáculo que se encarga de filtrar el aire que entra en éste y reducir su humedad en noches frías.

2.4. Otros sistemas y modificaciones

1. **Computadora de control.** Originalmente, existía dentro del receptáculo de AstMon un ordenador portátil que se encargaba de manejar todos los dispositivos, realizar la captura y análisis de datos, etc. Tras varios problemas con el mismo, se decidió instalar un dispositivo que permitiera canalizar varios USB's a través de un cable de red y poder llevarlos de esta forma a la sala entre cúpulas, donde un PC mejor equipado (con discos duros de mayor capacidad y por razones de construcción más fiable para esta labor) pudiera realizar la gestión de AstMon-UCM.
2. **Enlace Cámara-PC** El dispositivo clave en la migración del portátil al PC de la sala entre cúpulas (que se produjo a principios de 2011) es el Anywhere USB. Básicamente, es un sistema capaz de enviar las señales propias de varios USB a través de un cable de red, mejorando notablemente los límites en cuanto a alcance de los cables USBs[4]

[4]Estos generalmente tienen problemas, salvo que se utilicen repetidores, cuando se intenta usar cables de más de 5m

Desafortunadamente, el funcionamiento de este dispositivo deja bastante que desear porque el driver que lo maneja tiende a bloquear AstMon-PC entero con bastante frecuencia, siendo necesario a día de hoy acceder físicamente a AstMon-PC para reiniciarlo[5].

Además, se trata de un aparato cuyo soporte actualmente no es multiplataforma[6].

3. **Discos de almacenamiento.**

   De igual forma, el traslado de la gestión de AstMon-UCM ha abierto la posibilidad de introducir varios discos duros en AstMon-PC, aumentando enormemente las capacidades de almacenamiento de imágenes (actualmente hay 6TB instalados).

2.5. El software de AstMon

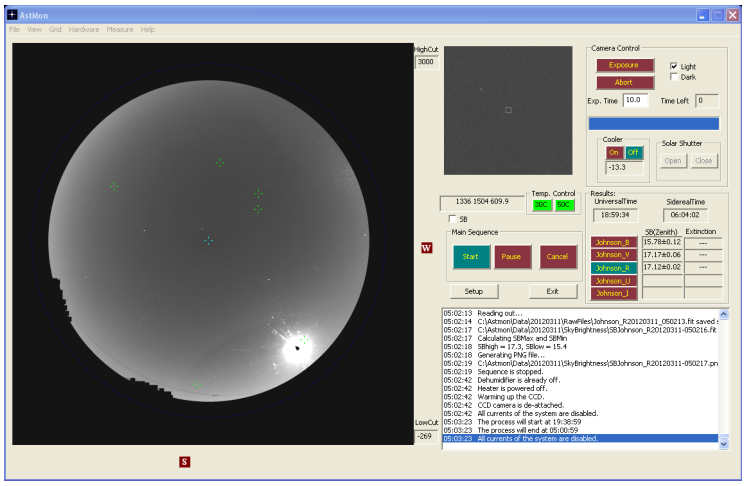

Figura 4: Pantallazo del software AstMon UCM

AstMon es un monitor automatizado, dispone de un completo software que se encarga tanto de la toma de imágenes y control de la rueda de filtros como del tratamiento y procesado de las mismas. En el siguiente esquema se muestra gráficamente (de forma simplicada) el proceso de toma de datos y análisis que realiza:

[5]Se está estudiando la posibilidad de introducir algún mecanismo, tipo interruptor por red, que permita reiniciar el equipo remotamente.

[6]La empresa creadora, Digi, sólo ha publicado los controladores para el sistema operativo Windows

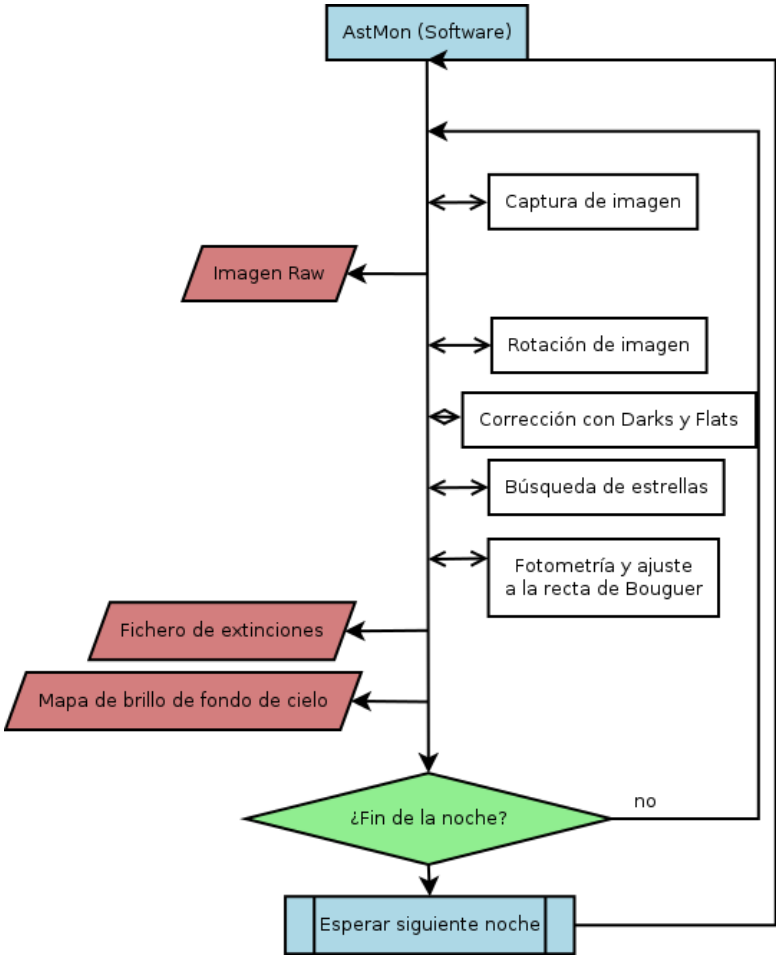

Figura 5: Diagrama de funcionamiento de AstMon (Software)

2.5.1. Calibración inicial

La calibración inicial del dispositivo fue realizada durante el curso 2010/2011 (ver [Mor11]). Básicamente, y tal y como se describe en [ea11], el software tiene que realizar una primera calibración astrométrica, que no es completamente automática (requiere de la intervención del usuario) para que el programa sepa la orientación y el eje óptico del sistema y pueda estimar correctamente la posición de las estrellas en la imagen.

Otra de las cosas que pide el programa es la generación de una máscara para la imagen, en otras palabras, se genera un patrón para estudiar únicamente una porción de la imagen, descartando así zonas astronómicamente poco útiles como son la posición de las cúpulas o zonas con fuerte contaminación lumínica en el horizonte. Afortunadamente, la aplicación de esta máscara de estrellas parece hacerse a posteriori de la toma de imágenes, con lo que siempre tenemos disponibles las imágenes crudas en las cuales esta máscara no se aplica.

Por último, se fijan las horas en que AstMon ha de medir cada día del año en el fichero *ObservingTimes.dat*. Dicho fichero tiene 3 columnas, la primera se compone de 4 dígitos (DDMM, con DD el día, MM el mes). La segunda columna y la tercera contienen otros 4 dígitos hhmm, con hh la hora y mm los minutos.

2.5.2. El catálogo estelar de AstMon

AstMon utiliza el catálogo de Ducati (2002), que incluye la fotometría en 11 bandas de Johnson para estrellas de hasta magnitud 16 además de sus coordenadas ecuatoriales J1950. Presenta la ventaja de disponer de un número suficientemente amplio de estrellas de diferentes tipos espectrales y de incorporar estrellas brillantes.

2.5.3. Preparando una sesión

Al principio de cada sesión, AstMon-PC manda una señal a AstMon-UCM que activa los mecanismos de refrigeración activa. Con ello se consigue reducir la temperatura del sensor hasta unos -15ºC. Éste es un paso obligado para poder disminuir la corriente de oscuridad que afecta a las imágenes. Alcanzada esta temperatura, comienza la toma de DARKS. Con esto se genera un MASTERDARK, que se restará de cada una de las imágenes de luz. Dado que el termostato mantiene una temperatura más o menos constante en el CCD a lo largo de la noche, podemos en principio utilizar este MASTERDARK durante toda la sesión sin problemas (la corriente de oscuridad debería permanecer inalterada). Los ficheros con las imágenes de DARK no se almacenan y sólo se emplean esa noche para realizar el procesado. Éste es uno de los trabajos pendientes ya que disponer en la base de datos de estos ficheros permitiría repetir por completo el procesado de las imágenes en un futuro.

Este proceso no se puede aplicar para generar un FLATFIELD, la otra de las correcciones que serían precisas. Generar un FLATFIELD en un dispositivo que cubre 180º en el cielo (una semiesfera completa) no es sencillo. No se pueden, por razones evidentes, aplicar FLATS de cielo, puesto que el cielo al atardecer y amanecer no tiene una luminosidad homogénea a gran escala. Se hace necesario utilizar una esfera integradora o dispositivo similar, capaz de generar una luz uniforme proyectada en una semiesfera que envolvería el sistema óptico de AstMon por completo, incluyendo la cúpula de metacrilato. Este proceso se realizó durante el montaje inicial de AstMon por parte del Dr. J. Aceituno colocando una cúpula blanca uniformemente iluminada en azimut sobre la cúpula transparente. Estudiando la variación de la luminosidad respecto al ángulo cenital del sistema, se generó un FlatField sintético que es el que viene utilizándose hasta hoy[7].

2.5.4. Toma de imágenes

Comienza la toma de imágenes. En este punto, se harán exposiciones de un cierto tiempo de exposición en cada uno de los filtros disponibles. Este tiempo de exposición, así como el lapso de tiempo entre cada serie de tomas, está predefinido en el fichero *Parameters.dat*. Actualmente, se usan tiempos de 40s para los 3 filtros en uso en AstMon-UCM, aunque anteriormente estuvo tomando imágenes con 10s de exposición. Incrementar más el tiempo de exposición tiene sentido si se está midiendo en cielos poco contaminados, donde el fondo de cielo es muy oscuro y hace falta prolongar las exposiciones para obtener una señal suficiente.

En el caso de AstMon-UCM, donde el cielo nocturno es muy brillante, utilizar exposiciones muy prolongadas puede provocar que la imagen se vele. Además, al aumentar el tiempo de exposición el movimiento de las estrellas se hace más evidente, y estas dejan de ser puntuales. Ello complica sensiblemente la medida de su flujo e introduce errores. Hay que llegar pues a un compromiso entre la señal de fondo que se captura y estos efectos indeseados.

[7] La metodología utilizada se describe con detalle en [ea11].

Las imágenes que se van tomando se guardan tanto en formato FITS (del inglés *Flexible Image Transport System*), como en PNG no lineal (para un uso no científico), en directorios que siguen la siguiente pauta

- `C:\AstMon\Data\YYYYMMDD\RawFiles`
- `Johnson_FYYYYMMDD_hhmmss.formato`

donde YYYY es el año, MM el mes y DD el día. Ha de notarse que actualmente AstMon almacena los ficheros según el día oficial, no según la noche astronómica. Esto supone un pequeño inconveniente al analizar las imágenes, puesto que parte de las imágenes (las correspondientes al anochecer) se almacenan en un directorio, y el resto (madrugada) se almacenan en otro diferente. En un futuro se cambiará este método para que la misma noche quede completamente almacenada en un sólo directorio.

De igual forma, F hace referencia al filtro utilizado (V,R,B para AstMon-UCM), YYYYMMDD es el año, mes y día. hhmmss es la hora, minuto y segundo. Formato puede ser, como hemos anticipado, png o fits.

2.5.5. Análisis de las imágenes

Las imágenes tomadas en el apartado anterior se calibran utilizando un MASTERFLAT ($FF(i,j)$ en notación matricial) y el MASTERDARK generado al inicio de la sesión ($DARK(i,j)$), de acuerdo con la expresión

$$SCI(i,j) = \frac{IMA(i,j) - DARK(i,j)}{FF(i,j)} \tag{2.1}$$

A continuación, AstMon aplica la máscara almacenada para evitar medir sobre zonas astronómicamente poco útiles de la imagen. Con esto, se llega a una imagen que contiene toda la información necesaria para determinar la constante instrumental del dispositivo (su valor depende del filtro en uso, y podría contener una ligera variación temporal como más adelante estudiaremos), el coeficiente de extinción y el brillo de fondo de cielo para ese momento y el filtro que se está utilizando.

Antes de nada, AstMon hace una identificación de las estrellas que aparecen en la imagen. Utiliza el catálogo Ducati (2002) descrito anteriormente. La identificación de las estrellas que allí aparecen es automática. Basándose en unas transformaciones previamente calculadas durante la calibración inicial del aparato, conociendo la posición geográfica de AstMon-UCM, la fecha y hora de la observación y mediante rutinas astronómicas se hace una estimación de la posición en la imagen que debería tener cada una de las estrellas que son visibles en el momento de la toma de la imagen.

Aunque la distorsión del ojo de pez que utiliza AstMon es muy pequeña, se aplica una pequeña corrección sobre las posiciones en la imagen que tiene en cuenta este efecto. Con esto, se buscan estrellas brillantes en los alrededores y se aplica el método de fotometría de apertura, es decir, se introducen 3 circunferencias concéntricas sobre la posición de la estrella. La apertura de las mismas queda definida en el fichero Parameters.dat (actualmente, los valores son 6,9 y 11 píxeles).

Sobre el círculo más interno, se mide el flujo de la estrella. Como las estrellas no aparecen puntuales en la imagen, sino que tienen una cierta distribución que, por simplicidad, podríamos entender como una gaussiana; se hace necesario dejar un gap (el anillo intermedio) que contenga

la posible cola de la distribución. Con esto evitamos que al medir el cielo (sobre el anillo más externo) podamos estar introduciendo parte del flujo de la estrella.

2.5.6. Determinación de coeficientes de extinción y constantes instrumentales

Medidos los flujos, se trata de resolver el sistema de ecuaciones:

$$m_\lambda = m_{\lambda_0} + K_\lambda \chi \tag{2.2}$$
$$m_\lambda = C_\lambda - 2{,}5 \log_{10} F(c/s) \tag{2.3}$$

En la expresión anterior, C_λ es la constante instrumental en la banda fotométrica λ. K_λ es el coeficiente de extinción en λ. χ es la masa de aire, que en primera aproximación es $\sec z$. También aparece m_{λ_0} sería la magnitud que está almacenada en el catálogo, es decir, aquella que mediríamos si pudieramos eliminar la atmósfera terrestre, y por tanto, cualquier absorción de flujo por parte de esta. Por su parte, m_λ es la magnitud que podríamos medir desde Tierra. Como no es conocida a priori, es conveniente eliminarla y dejar todo en función de magnitudes conocidas. Así, llegaríamos a la expresión

$$m_{\lambda_0} + 2{,}5 \log_{10} F(c/s) = C_\lambda - K_\lambda \chi \tag{2.4}$$

Esta ecuación nos invitaría a ajustar a una recta de la forma $y = mx + n$ los datos disponibles. En efecto, tendríamos $y = m_{\lambda_0} + 2{,}5 \log_{10} F(c/s)$, $x = \chi$, $m = -K_\lambda$ y $n = C_\lambda$. El problema de esto es que precisamente uno de los objetivos de AstMon es estudiar noches que no son fotométricas. En ellas, la relación anterior puede dar unos resultados bastante malos.

Lo que hace AstMon es, en primer lugar, determinar para cada par de estrellas detectadas un valor de la extinción. En principio, si la noche fuera fotométrica los resultados deberían distribuirse con poca dispersión en torno a un valor central. Se calcula acto seguido la mediana de los valores de extinción determinados y se descartan los valores que difieren en más de un 10 % de la mediana (sigma clipping). Este "descarte" de valores permite evaluar la bondad de los datos antes de realizar un ajuste. En efecto, AstMon realiza un ajuste lineal sólo en aquellos casos en que el 75 % de la masa de valores cae dentro de ese rango de ± 10 % de la mediana. Esto serviría para obtener valores tanto para la constante instrumental como para la extinción.

El coeficiente de correlación obtenido a través de este ajuste se utiliza a su vez para decidir si se debe actualizar la constante instrumental. El criterio que utiliza es que para modificar las constantes instrumentales, hace falta que $R^2 > 0{,}95$. Por desgracia para nosotros, desde que AstMon-UCM fue instalado hace ya dos años, ninguna imagen de las tomadas ha cumplido este requisito tan exigente, con lo que hoy día siguen estando definidas las constantes instrumentales que se derivaron de la noche del 1 de Noviembre de 2009.

2.5.7. Cálculo del brillo de fondo de cielo

Dejar fija la constante instrumental y actualizarla cuando se den una serie de condiciones tiene algunas ventajas importantes. Como hemos dicho, uno de los objetivos principales de AstMon es ser un monitor de brillo de fondo de cielo que pueda trabajar a lo largo del año, de forma que se pueda estudiar la evolución del mismo a largo plazo, incluso en noches no fotométricas.

Calcular el brillo de fondo de cielo en una determinada posición del cielo es un procedimiento sencillo si utilizamos la expresión

$$SB = C_\lambda - 2{,}5 \log_{10}(F_{cielo}(c/s)/A) \tag{2.5}$$

siendo F_{cielo} el flujo en cuentas/s que se registra en un píxel de imagen y A el área en $arcsec^2$ del cielo proyectada en 1 píxel de la cámara. AstMon es capaz de generar imágenes en las que el brillo de fondo de cielo se mide para toda la imagen (en realidad, parece que utiliza una serie de puntos convenientemente distribuidos y a partir de los valores obtenidos hace una interpolación). Estas imágenes se almacenan actualmente en el disco duro, en la siguiente ruta y nombre

- `C:\AstMon\Data\YYYYMMDD\SkyBrightness`
- `SBJohnson_FYYYYMMDD_hhmmss.formato`

donde F hace referencia al filtro utilizado (V, R, B para AstMon-UCM), YYYYMMDD es el año, mes y día. hhmmss es la hora, minuto y segundo. El formato puede ser, como hemos anticipado, png o fits.

También muestra el valor correspondiente al cénit (el cénit calculado durante la calibración inicial de AstMon-UCM está en las coordenadas de imagen 1220,1268[8]). En este caso, se guardan los datos en un fichero de texto plano en el mismo directorio que las imágenes, `SBJohnson_FYYYYMMDDPos0.dat` (la sintaxis es la misma que hemos descrito anteriormente). Dentro de ese fichero, encontramos 7 columnas:

Fecha y Hora	X	Y	Brillo de fondo de cielo	Incertidumbre	?	?

Estos ficheros permiten analizar de forma rápida y sencilla la evolución del brillo de fondo de cielo a lo largo de una noche en los filtros seleccionados.

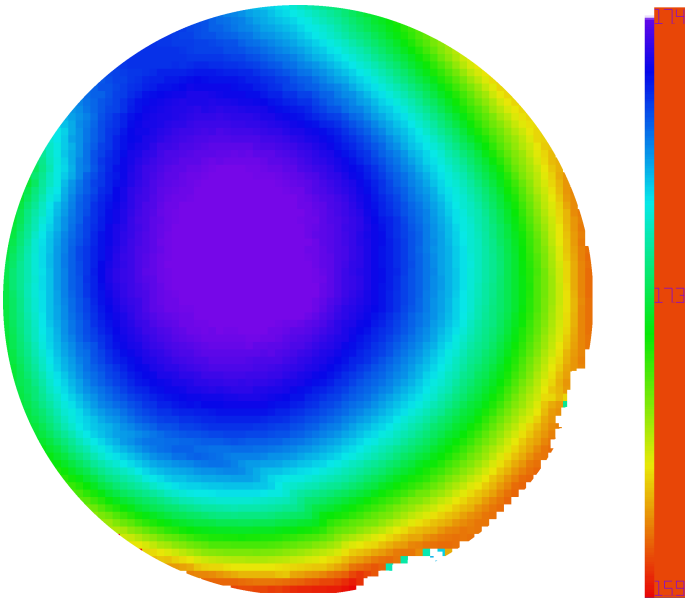

Figura 6: Mapa de brillo de fondo de cielo obtenido con AstMon

[8]Cuando hemos realizado la astrometría por nuestra cuenta, se ha encontrado como solución 1227,1271

2.5.8. Otras herramientas de AstMon-UCM

Hemos utilizado las funciones principales del software AstMon para generar datos de extinción y mapas de brillo de fondo de cielo, pero no son las únicas. Una función que se utilizó al principio, pero que actualmente está desactivada, es la generación de mapas de nubes.

Al medir el flujo de las estrellas que hay en el campo en una noche parcialmente nubosa, podemos ver que en ciertas regiones del cielo el flujo medido es mucho menor respecto al esperable que en otras regiones. Éste es por tanto un buen indicador de la cobertura de nubes existente.

Hasta aquí hemos hecho un breve repaso del funcionamiento interno de AstMon. El objetivo fundamental de este trabajo, aparte de estudiar cómo funciona tanto el hardware como el software AstMon, es estudiar la posibilidad de replicar ambas partes e implementarlas de forma independiente.

3. Procedimientos de fotometría absoluta

3.1. Introducción

La autonomía rutinaria de AstMon-UCM tiene grandes ventajas. AstMon-UCM dispone de una completa suite de software que se encarga de prácticamente todo, tanto de la toma de imágenes como de su análisis en tiempo real, guardando los resultados en el disco duro. Sin embargo, este automatismo plantea inconvenientes. Aunque la mayoría de los parámetros de funcionamiento son seleccionables, los programas internos trabajan de forma que el usuario no dispone de todo el control ni se sabe en muchos puntos cómo se realizan ciertos tratamientos de interés astrofísico. Por ello, pretendemos migrar el control y el procesado de datos a un sistema Linux ya que hemos comprobado que AstMon-UCM presenta problemas de interrupciones no deseadas en el sistema operativo MS Windows, donde trabaja actualmente. Con este fin pretendemos reproducir todas las rutinas de procesado de forma que sean parametrizables. De esta forma tendremos el premio adicional de poder exportarlo a otros dispositivos similares más sencillos.

Se muestra a continuación la reprodución de los pasos que sigue AstMon durante el análisis de las imágenes, con nuestras aportaciones originales en los puntos que se ha estimado conveniente,

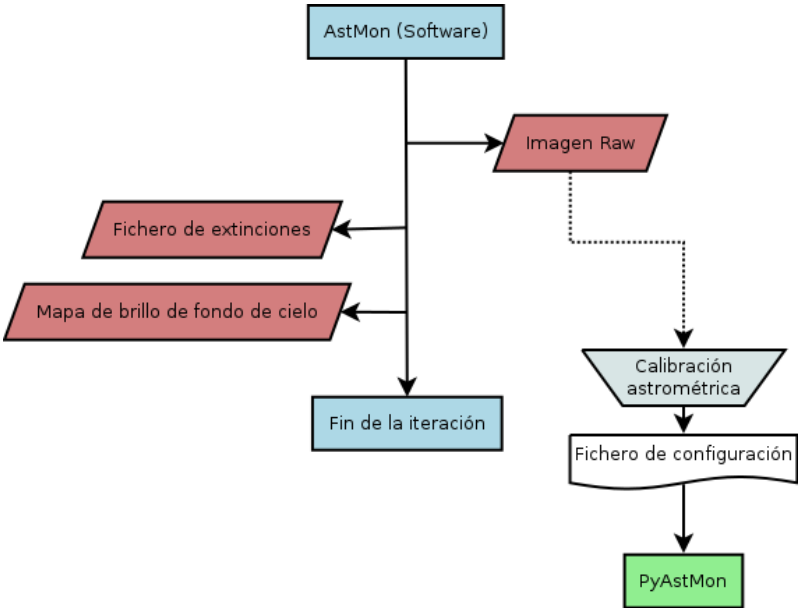

Figura 7: Diagrama de operación de AstMon. A partir de la línea discontínua, análisis de las imágenes con procedimientos alternativos a AstMon presentado en este trabajo. Nótese que por el momento, la calibración inicial astrométrica se ha de realizar manualmente.

Una de las primeras decisiones a tomar al abordar el proyecto era elegir el lenguaje de programación a utilizar para construir los módulos y algorítmos que analizarían las imágenes. Tras sopesar si empleábamos Matlab, C, C++ o Python nos decantamos por Python (concretamente, la versión 2.7) por ser un lenguaje sencillo, potente, altamente portable (hasta el punto de que la mayor parte del código se ha escrito y probado desde un sistema operativo GNU/Linux, mientras que el análisis como tal se proyectaba realizar sobre el propio AstMon-PC, que lleva

instalado el sistema Windows XP). Python dispone además de una gran cantidad de paquetes y módulos que han sido muy útiles y nos han ahorrado bastante trabajo, como por ejemplo *Pyephem*, *Numpy*, *Scipy* o *Pyfits*.

El objetivo será generar un paquete de módulos que permitan analizar tanto las imágenes de AstMon-UCM como otras imágenes AllSky de distinta procedencia. Para empezar, nos conformaremos con intentar reproducir los resultados a los que llega AstMon.

Todo el software ha sido convenientemente documentado y parametrizado de forma que sea fácilmente exportable a dispositivos similares. Para facilitar la lectura de la memoria los diferentes módulos se han incorporado en el Apéndice 1 salvo quizá aquellos suficientemente cortos como para no interrumpir la lectura.

3.2. Descripción de los módulos

3.2.1. Parametros del observatorio UCM

Éste es el primer paso de nuestro tratamiento. Por simplicidad en el trabajo con ciertas coordenadas celestes, utilizaremos muy a menudo el paquete *Pyephem*. Éste nos exige que definamos las coordenadas (latitud y longitud) de nuestro observatorio, así como la fecha y hora (parámetros necesarios para el paso de coordenadas ecuatoriales a horizontales y viceversa).

Utilizamos entonces dos módulos en este punto. El primero de ellos únicamente fijará los datos de geolocalización de nuestro observatorio `observatorio_UCM.py` [A.1] y el otro carga la imagen de AstMon utilizando `lectura_fits_calibracion` [A.2] del módulo `cargar_imagen.py` (véase Apéndice 1).

Las imágenes generadas por AstMon están en formato FITS (imágenes crudas). Para manejarlos, hacemos uso del paquete PyFITS, separando la cabecera de la imagen (con parámetros como filtro, fecha o tiempo de exposición que pasaremos en forma de una clase *Imagen* a nuestros módulos) del resto de la imagen.

Código 3.1: Ejemplo de cabecera FITS generada por el software AstMon

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                  -32 / number of bits per data pixel
NAXIS   =                    2 / number of data axes
NAXIS1  =                 2500 / length of data axis 1
NAXIS2  =                 2500 / length of data axis 2
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
EXPOSURE=                 10. / Total Exposure Time
FILTER  = 'Johnson_V'          / Filter
UT      = '02:36:16'           / Universal Time
DATE    = '20111108_023616'    / YYYY-MM-DD
END
```

Como vemos la información contenida en la cebecera es mínima. No contiene datos sobre la ganancia del CCD, ni del observatorio, ni aparecen en ellos reflejada una solución astrométrica o fotométrica. Solucionar esta carencia es otro de los trabajos que se proponen para el siguiente curso ya que la cabecera FITS es la fuente de información que permite extraer de los datos almacenados los resultados científicos.

Parte de esta información es esencial para trabajar con el paquete de módulos propuestos en este trabajo. Tal es el caso de *EXPOSURE*, *FILTER* y *DATE*, información que afortunadamen-

te queda representada en los ficheros que genera el software de captura de AstMon. Cuando tratemos de analizar imágenes que no son de AstMon, por ejemplo provenientes de algún dispositivo móvil como una cámara Réflex con ojo de pez, tendremos previamente que completar la cabecera fits de forma adecuada para que nuestros programas trabajen.

El otro cometido de la función `lectura_fits_calibracion` es separar los datos de AstMon. La CCD utilizada tiene una profundidad de bits de 16 (enteros). Pese a ello, AstMon almacena en el fichero una serie de valores flotantes con precisión de 32bits, probablemente con la intención de no perder información tras el proceso de calibrado (recordemos que durante el procesado, AstMon aplica internamente DARKS y FLATS, estos últimos implican divisiones entre números y es natural que obtengamos decimales). La estructura de la parte de datos del fichero fits es la de una matriz $N \times N$ de valores flotantes. Leer esa matriz para operar con ella desde Python es realmente sencillo, como se muestra en el siguiente ejemplo[9]

```
\$ python
```
Cargamos el paquete pyfits

```
>>> import pyfits
>>> fichero_fits=pyfits.open("Imagen_de_astmon.fit")
>>> matriz_datos_fits = fichero_fits[0].data
```
Mostramos el valor de alguna celda y el tipo de matriz que estamos tratando

```
>>> type(matriz_datos_fits)
<type 'numpy.ndarray'>
>>> matriz_datos_fits[1560][2213]
1186.0
```
Hay que notar que estamos ante una matriz de datos, la forma de acceder a ellos es indicar primero la fila (píxel Y) y después la columna (píxel X). Es decir, en el ejemplo propuesto `matriz_datos_fits[1560][2213]` indica que estamos tomando el valor en $Y = 1561$ y en $X = 2214$ (los vectores en Python van desde 0 a $N - 1$, de ahí ese desfase de una unidad).

3.2.2. Solución astrométrica

El primer paso es familiarizarnos con las imágenes que genera AstMon, tratar de determinar donde se situa el cénit de la imagen y hallar las transformaciones necesarias para pasar de coordenadas X,Y (píxeles) en la imagen a coordenadas astronómicas (primero *Azimut* y *Altura*, después *Ascensión recta* y *Declinación*).

El objetivo utilizado presenta, como ya hemos comentado, una proyección *Zenithal Equal Area* (ZEA), cuyas ecuaciones de transformación son [cG02],

$$x = R_\theta \sin\phi \qquad y = -R_\theta \cos\phi \tag{3.1}$$

$$R_\theta = \frac{180^\circ}{\pi}\sqrt{2(1-\sin(\theta))} = \frac{360^\circ}{\pi}\sin(\frac{90-\theta}{2}) \tag{3.2}$$

Es evidente que para poder utilizar estas expresiones (o sus inversas), nos hace falta definir condiciones iniciales. Si las caracterizamos en coordenadas polares planas, estas dos condiciones

[9] En sistemas UNIX/Linux abrimos una consola y ejecutamos python (el binario debe estar en el PATH)

iniciales son la posición del cénit de nuestra imagen (a partir de donde se mide la altura θ y por tanto R_θ), y un 0° para ϕ.

Esta calibración AstMon la hace de forma semiautomática como hemos visto, y hay planes de intentar automatizar la búsqueda en nuestro paquete de software (de conseguirlo, podríamos trasladar nuestro programa a otro tipo de cámaras de forma realmente simple). Sin embargo, y como primera aproximación, se realizó la calibración manualmente.

Para ello, se construyó un script que leía las condiciones iniciales que nosotros fijábamos y pintaba sobre la imagen de AstMon las estrellas en las posiciones que se estimaban a partir de esas condiciones iniciales. Realizando varios ajustes sucesivos por prueba y error, se obtuvieron unas condiciones tales que las estrellas estimadas se aproximaban bastante a las reales, y en cualquier caso, cualquier residuo en el ajuste sería facilmente eliminable durante el procesado de las imágenes mediante el cálculo de centroides[10].

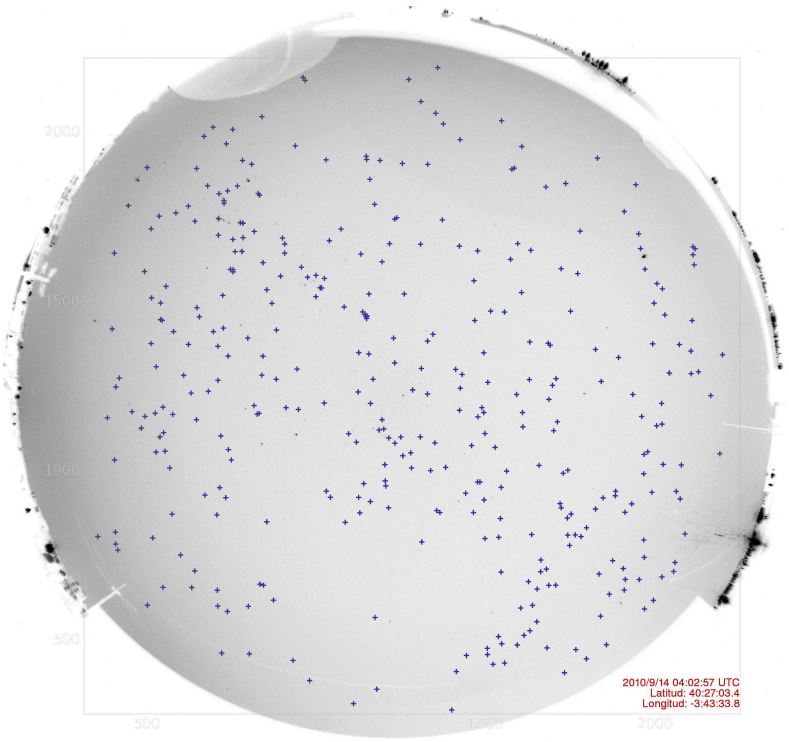

Figura 8: Posiciones estimadas de las estrellas

3.2.3. Conversión de coordenadas

Una parte clave en nuestro desarrollo es poder identificar una serie de estrellas en una imagen de ojo de pez. El catálogo del que partimos nos da coordenadas ecuatoriales (referidas a J1950.0) que tendremos que transformar en primera instancia a coordenadas horizontales (que dependen de posición del lugar de observación y de la hora sidérea) y después, pasar de esas coordenadas azimut-altura a coordenadas X,Y en nuestra imagen. Tenemos por tanto dos pasos (ó 4 si consideramos los inversos).

[10] En la sección XXXX se comenta cómo implementar esta idea en nuestro algoritmo.

Se crearon para esta tarea un par de módulos. El primero de ellos, `eq2horiz.py` se encarga del paso de coordenadas ecuatoriales a horizontales según las expresiones,

- Paso de Ecuatoriales a Horarias

$$AH = ST - \alpha \tag{3.3}$$

  siendo α la ascensión recta del astro, AH su ángulo horario (depende de la hora sidérea) y ST el tiempo sidéreo local del lugar de observación.

- Paso de Horarias a Horizontales

$$\sin ALT = \cos\phi \cdot \cos\delta \cdot \cos AH + \sin\phi \sin\delta \tag{3.4}$$
$$\cos ALT \cdot \cos AZ = \cos\phi \sin\delta - \sin\phi \cos\delta \cos AH \tag{3.5}$$
$$\cos ALT \cdot \sin AZ = -\cos\delta \sin H \tag{3.6}$$

  Para resolver estas expresiones, hay que tener presente que los signos asociados a sin y cos son importantes, pues según su valor estaremos en un cuadrante u otro (formalmente $\tan\theta = \tan(\theta + 180)$).

En el módulo encargado de estos cálculos, se ha definido por conveniencia una función que realiza el paso directo de ecuatoriales a horizontales[11]. Lo hace llamando sucesivamente a las funciones de paso de ecuatoriales->horarias y horarias->horizontales. De igual forma para el paso inverso (que no utilizaremos en nuestro desarrollo).

Una vez tenemos las coordenadas horizontales calculadas, y hecha la calibración inicial para poder pasar a las posiciones sobre la imagen en proyección ZEA, ya podemos pasar las coordenadas horizontales a coordenadas X,Y sobre la imagen. La transformación se reduce a aplicar las expresiones (3.1) y (3.2), que se implementan en el módulo `horiz2xy.py`.

De igual forma podríamos mostrar la inversa (en el Apéndice 2 daremos referencias para poder acceder al código completo del proyecto, omitiendo algunas partes en este punto por brevedad). También se han generado funciones que devuelven las coordenadas horizontales del eje óptico (el punto central del CCD) y la posición en el sensor del cénit.

3.2.4. Catálogo estelar

Para realizar todas las medidas sobre la imagen, tenemos que disponer de un catálogo donde aparezca una lista de estrellas susceptibles de ser estudiadas. Al igual que el software de AstMon, optamos por utilizar el catálogo Ducati 2002, un catálogo de estrellas hasta magnitud 16 (filtro V) en 11 bandas fotométricas de Johnson. Además de las distintas magnitudes en cada banda (con las que podemos extraer información de colores por ejemplo), muestra el número de catálogo, sus coordenadas (J1950) y algunos datos más.

El catálogo se generó utilizando la herramienta VizieR del *Centre de données astronomiques de Strasbourg*, pidiendo que generara un fichero de texto plano en formato TSV (*Tab Separated Values*), que facilitará enormemente la lectura de datos. Algunas particularidades de este catálogo generado son las siguientes

[11] Para realizar los cálculos, se precisa conocer con precisión el tiempo sidéreo local. Si bien es una magnitud que podemos conocer o determinar utilizando Pyephem, descubrimos que el propio Pyephem tiene algorítmos para esta transformación de coordenadas ya predefinidos, que son los que realmente se utilizan en nuestro procedimiento. Se conserva el código para la transformación de coordenadas por mera completitud.

- Se piden las siguientes columnas: *recno, Name, RAJ1950, DEJ1950, Vmag, U-V, B-V, R-V, I-V*.
- Se ordenan los datos por Vmag. Nos centraremos únicamente en las 80 o 100 estrellas que tengan una magnitud V menor, el resto son demasiado débiles para arrojar unas medidas de flujo aceptables en nuestro lugar de observación y con el CCD utilizado.

El fichero que bajamos contiene 27 líneas introductorias (comentarios acerca del catálogo y su contenido, descripción de columnas, etc) y después empiezan a aparecer los valores de las estrellas, en la forma

Código 3.2: Catálogo estelar Ducati 2002 utilizado

```
recno     Name     RAJ1950     DEJ1950     Vmag     U-V     B-V     R-V     I-V
           "h:m:s"     "d:m:s"      mag      mag     mag     mag     mag
--------    ------------    ----------    ---------    -----    -----    -----    -----    ------
    1279      HD  48915       06 42 56.7    -16 38 46    -1.46    -0.05     0.00     0.00     0.03
    1155      HD  45348       06 22 50.5    -52 40 03    -0.74     0.25     0.15    -0.22    -0.39
    2371      HD 124897       14 13 22.7    +19 26 30    -0.05     2.51     1.23    -0.98    -1.63
    2419      HD 128620       14 36 11.2    -60 37 49     0.01     0.95     0.71
    3145      HD 172167       18 35 14.6    +38 44 09     0.03     0.00     0.00     0.04     0.07
     803      HD  34029       05 12 59.4    +45 56 56     0.08     1.25     0.80    -0.60    -1.04
```

La lectura y análisis de este fichero se lleva a cabo con el módulo `cargar_catalogo.py`, que tiene dos funciones. La primera de ellas símplemente recorre el fichero y va almacenando en una matriz los diferentes valores de las variables antes descritas (columnas) para cada estrella. La otra función es quizá algo más compleja. Carga la matriz que nos da la función anterior para procesar algunos de sus valores. La idea es adaptar el formato al que acepta *Pyephem*. De paso, a partir de la magnitud en B y los índices de color B-V y R-V, calcula las magnitudes en los filtros B y R.

Por último, utilizando *Pyephem* realiza el cálculo de las coordenadas horizontales de la estrella, lo que nos servirá para registrar en una nueva tabla únicamente aquellas estrellas que son visibles en el cielo en ese momento (aquellas que tienen una altura mayor que una mínima, por defecto 0). Véase `cargar_catalogo_matriz.py` [A.4] en el Apéndice 1. Tenemos con esto toda la información astrométrica y fotométrica que necesitamos en nuestro análisis, y además ya habíamos descrito el procedimiento de carga en memoria de la imagen (cabecera por un lado y la matriz) de valores por otro.

3.2.5. Análisis de las imágenes

En este punto se desarrolló un algorítmo capaz de analizar la imagen y detectar las estrellas que aparecen en el catálogo en la misma. Para eso, utilizando la solución astrométrica del punto anterior (paso de coordenadas ecuatoriales del catálogo a las coordenadas XY de la imagen) se hace una primera estimación de las posiciones de las estrellas.

En primer lugar, se inicializan algunas variables y se realiza una copia sin manipular de la matriz de datos para las representaciones gráficas. Véase `deteccion_estrellas.py` en Apéndice 1.

Medidas sobre cada estrella A continuación, comenzamos un procedimiento iterativo que tomará ciertas medidas (flujo básicamente) para cada estrella. Usaremos para ello el código definido en `deteccion_estrellas.py`. Llamamos primero a la función `parametros_estrella`, encargada de recuperar de la matriz de datos una serie de parámetros de la estrella, tales como su

magnitud en el filtro en uso, la magnitud en V, las coordenadas azimut y altura, el número de catálogo HD, y las soluciones astrométricas (posiciones X,Y en la imagen). Se calcula asimismo la masa de aire (si llamamos $z = 90 -$ altura al ángulo cenital, la masa de aire vendría dada en primera aproximación por $\sec(z)$) y el índice de color. Toda esta información se guarda en una clase `Estrella` y se devuelve el control a la función `tabla_fotometria`.

Tenemos que realizar una primera estimación de su posición en la imagen. El tipo de fotometría que vamos a realizar se conoce como fotometría de apertura, en la cual se generan una serie de anillos de medida (pueden ser 1,2 ó 3, siendo la última la que utilizamos, por ser más completa y precisa). La función de cada una ya se describió cuando analizábamos el software AstMon (apartado 3.2.5) y no se repetirá por brevedad.

Selección de aperturas La experiencia ha demostrado que no es conveniente usar un radio fijo para todas las estrellas del campo. Al observar las imágenes, uno se da cuenta que el aspecto típico de las estrellas no es igual en todos los filtros, siendo en el filtro B mucho menos puntuales las estrellas, bien por la óptica utilizada o bien por la propia atmósfera.

De igual forma, a medida que nos alejamos del eje de la imagen el perfil de la estrella deja de ser circular, pasando a algo que se asemeja a un "cometa", como ilustra la siguiente imagen

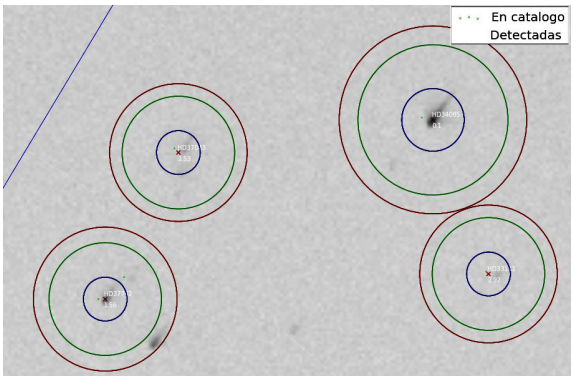

Figura 9: Deformación del perfil de la estrella cuando nos alejamos del eje de la imagen. Se observan asimismo los círculos de medida que se han estimado para 4 de las estrellas, la posición inicial calculada con la solución astrométrica (puntos verdes) y el ajuste por centroides (aspas rojas)

En nuestro caso, utilizamos tiempos de exposición muy cortos, de apenas 40s por filtro. Es de esperar que en una imagen de mayor tiempo de exposición pueda entrar en juego también la rotación de campo y deformarnos la estrella. Y por último, el propio brillo de la estrella hace que esta no sea un objeto puntual en la imagen, siendo más extenso en la misma cuanto más brillante es la estrella.

Con todo, es evidente que necesitaremos unos tamaños de los círculos adaptables, que tengan en cuenta todas estas posibilidades. Eso se consigue llamando a la función `radio_discos_mag` del módulo `grid_medidas.py`

Medida preliminar de los flujos Como vemos, se parte de un tamaño base (muy pequeño), y se van aplicando correcciones que tienen en cuenta el tiempo de exposición, la altura de la estrella (recordemos que en estas cámaras AllSky hablar de altura prácticamente equivale a hablar de distancia al eje de la imagen), la magnitud en el filtro considerado y la resolución de la imagen (en una imagen de muy baja resolución, como en una CCTV, no tiene sentido partir de tamaños

de discos de medida que cubran un porcentaje muy alto de la superficie de la imagen. De igual forma, en cámaras reflex modernas o CCDs de mucha resolución, lo normal es que la estrella tienda a ocupar más píxeles).

De todas formas, estos tamaños son simplemente el punto de partida para determinar el centroide de la estrella y dar una primera medida del flujo de la estrella (de cara a ver si se ha detectado o no). A la hora de medir flujos para ajustar, utilizaremos lo que se conoce como "fotometría de saturación". Partiendo de un tamaño base, vamos aumentando el radio del círculo interno de medida (el que corresponde al flujo de la estrella+fondo) hasta que el flujo neto de la estrella (una vez restado el fondo que se medía con el anillo más externo) crece en un porcentaje <2 %.

El siguiente paso es hacer una primera medida aproximada del flujo de la estrella, puesto que vamos a analizar una gran cantidad de zonas en cada imagen, nos interesa que el proceso sea rápido. Sólo se usará para intentar detectar si la estrella es visible o no en la imagen y si el flujo que da es suficiente. Usamos el módulo `deteccion_estrellas.py`.

Como vemos, en esta etapa no estamos tan interesados en acotar de forma precisa los flujos y sus incertidumbres, sino tener una cierta idea de si la estrella se muestra brillante o no en la imagen.

Pasemos al siguiente fragmento de código de la función `tabla_fotometria`. Esta parte de código se encuentra en `deteccion_estrellas.py` [A.10].

Se descartan en primer lugar las estrellas de magnitud elevada, no son buenas candidatas para realizar fotometría. La magnitud límite depende de varios factores, tales como el filtro utilizado (hay que ser más permisivos con el filtro B por ejemplo, dado que la señal de este filtro siempre tiende a ser más pobre y la detección de estrellas de calibración da una SNR más baja), la altura (es importante tener una masa de puntos razonable a alturas relativamente bajas en tanto en cuanto nos amplían la base en valores de masa de cielo para calcular la extinción) o el tiempo de exposición (un tiempo de exposición mayor nos permite evidentemente detectar estrellas menos brillantes). Estas dependencias pueden verse en `deteccion_estrellas.py` [A.11].

En `medidas_imagen.py` [A.12] nos encontramos la función que corrige de viñeteo. Las imágenes que se están analizando pueden no estar calibradas con imágenes de FLATS y DARKS. Al ser ópticas de muy amplio campo, lo normal es que estén sujetas a un fuerte viñeteo, efecto que tendremos que tener en cuenta de alguna forma, bien utilizando datos proporcionados por el propio fabricante (es el caso de algunas ópticas de Sigma, en las que se nos proporcionó una tabla de medidas que mostraban el viñeteo para distintos ángulos) o bien utilizando medidas propias.

AstMon-UCM utiliza hasta la fecha unos FLATS que se tomaron hace ya varios años, cuando el dispositivo contaba con el obturador solar entre otras piezas. Ese obturador introducía algunos efectos en la iluminación del chip que aparecen de forma muy clara en el FLAT. Esos FLATS se utilizan internamente en el programa (igual que los DARKS que toma en cada sesión), pero la imagen RAW que se almacena es una imagen sin calibrar.

Mientras se prepara el dispositivo experimental para crear unos nuevos FLATS que ya tengan en cuenta la eliminación del obturador solar (el proceso no es sencillo, pues hace falta iluminar de forma uniforme 180° de campo, y nuestra intención a priori es realizar la toma de FLATS sin tener que desplazar AstMon-UCM para evitar perder la propia calibración astrométrica del dispositivo) la corrección de viñeteo en AstMon se utiliza midiendo sobre el viejo FLAT en las zonas que creemos están menos afectadas por el obturador solar, obteniendo un conjunto de medidas similares a las de Sigma y corrigiendo de ellas.

Como vemos, básicamente toma los ángulos y luminosidad relativa (respecto al valor máximo) dados en dos listas en el fichero de configuración, realiza la interpolación entre estos valores

y lo aplica, siempre que existan las listas antes mencionadas, a los valores de flujo medidos. Esta función es muy importante de cara a poder construir correctamente la recta de Bouguer y generar el mapa de brillo de fondo de cielo.

Selección de estrellas para fotometría Las condiciones de detección dependerán de las propias características de la imagen, como ocurría con el radio de la estrella. Con esto buscamos ser más permisivos en la detección de estrellas por ejemplo a alturas bajas, en las cuales la estrella ocupa más píxeles y el flujo por píxel es sensiblemente menor. Tener medidas a relativamente baja altura es crítico para obtener una buena recta de Bouguer, puesto que cualquier variación de la extinción queda muy bien trazada aunque tengamos una incertidumbre algo más alta en flujos o en la medida de la masa de atmósfera que con estrellas cercanas al cénit. `deteccion_estrellas.py` [A.13] la estimación que se hace del flujo límite detectable,

$$F_{lim} = C^0_{lim} \cdot F_{fondo} \cdot \Sigma_a \cdot \sqrt{t_{exp}} \tag{3.7}$$

Esta parametrización se ha obtenido de forma empírica (da buenos resultados generalmente con el tipo de instrumentos que estamos utilizando).

- Partimos de un valor base C^0_{lim}, especificado en el fichero de configuración
- Se multiplica por el flujo de fondo (en este punto, para un valor $C^0_{lim} = 1$ estaríamos dando por buena un flujo de estrella *neto* igual al flujo de fondo)
- Multiplicamos por un parámetro relacionado con la altura. De nuevo, conviene relajar los requisitos para alturas bajas para evitar quedarnos sin estrellas con valores de *secz* elevados.
- Multiplicamos por último por $\sqrt{t_{exp}}$. La señal de la estrella aumenta con el t_{exp} mientras que el ruido lo hace con $\sqrt{t_{exp}}$. Cuando el tiempo de exposición aumenta, es más sencillo encontrar estrellas y con este parámetro seríamos más exigentes en la detección, evitando que se produzcan un número elevado de detecciones "débiles".

La última parte vista en la función que realiza el análisis de la imagen mostraba un `if`. Es una condición para descartar estrellas que no reúnen las condiciones para ser usadas en la fotometría por tener un flujo neto de estrella demasiado pequeño. Las que superan este test, pasan al siguiente punto, en el cual haremos las medidas de flujo de forma precisa.

Medida precisa de los flujos En primer lugar, se determina la posición de la estrella de forma más precisa calculando un centroide[12]

El objetivo de la función `discos_medida` es proporcionar una lista con los píxeles contenidos en cada disco de radio dado centrado en una cierta posición de la imagen y la función `calcular_centroide` del módulo `estadistica.py` utiliza esa lista de píxeles para calcular el centro de gravedad.

Este punto es bastante delicado y es conveniente describirlo con detalle. Imaginemos un caso sencillo, unidimensional, en el que pretendemos determinar el centro de los datos como en el ejemplo

[12] `deteccion_estrellas.py` [A.14]

Posición (X)	1	2	3	4	5	6	7	8	9	10
Brillo (B)	20	21	19	19	20	22	28	36	26	20

Tradicionalmente, los centros de gravedad se calcularían mediante la expresión

$$\bar{X} = \frac{\sum_{i=1}^{N} X_i \cdot B_i}{\sum_{i=1}^{N} B_i} \tag{3.8}$$

Según esto, el centroide se situaría en la posición 5.8 , que difiere bastante del aparente 8 que esperaríamos viendo los datos. Es evidente que hay que modificar ligeramente este método. En primer lugar, observamos que los datos del ejemplo tienen un "offset" de en torno a 20. Ese offset empeora notablemente los resultados obtenidos. Restando ese valor 20, obtenemos 7.9 como valor. Parece evidente que obtener un valor fiable para el fondo es crítico a la hora de obtener el centroide.

Otra posibilidad para mejorar el resultado es no considerar (3.8), sino una variante.

$$\bar{X} = \frac{\sum_{i=1}^{N} X_i \cdot B_i^2}{\sum_{i=1}^{N} B_i^2} \tag{3.9}$$

Con esto (y sin restar el fondo) obtendríamos un valor 6.2 . Nuevamente, la razón de utilizar esta variante se puede ver si observamos el ejemplo propuesto anteriormente. En forma lineal, el valor de 36 pesa obviamente más que un valor 20. Pero aun el dominio es más claro si calculamos el cuadrado, siendo $36^2 \gg 20^2$. Esto nos llevaría a plantear la posibilidad de usar una potencia aun mayor. Sin embargo, hemos de tener especial cuidado con valores anormalmente altos de ciertos píxeles (aquellos afectados por un rayo cósmico por ejemplo, y que tienen un valor en cuentas muy superior al de muchos píxeles con flujo estrella).

Una vez observamos los resultados, es evidentemente que el método más efectivo en este caso es restar el fondo (probablemente por tener un SNR bajo), pero en nuestro análisis utilizaremos ambas estrategias a la vez para lograr una mejor convergencia. Para su implementación, véase `deteccion_estrellas.py` [A.15].

Teniendo el centroide estimado, el siguiente punto es realizar una medida más precisa del flujo de la estrella y calcular con esto los parámetros necesarios $(2{,}5 \log F_\nu,\ m + 2{,}5 \log F_\nu)$ para hacer el ajuste a la recta de Bouguer. Véase `deteccion_estrellas.py` [A.16]

Medida precisa del fondo La medida precisa del fondo se realiza ajustando el brillo de fondo que rodea a cada estrella a una Gaussiana. Es una técnica más lenta durante el cálculo pero el resultado es más preciso que usar un estimador más simple como la mediana. Véase `deteccion_estrellas.py` [A.17]

Nuestro objetivo es determinar el flujo neto de la estrella, pero nosotros lo que medimos son flujos estrella+fondo. De esta forma, hay medir primero en torno a la estrella el flujo de fondo y restárselo del flujo integrado de la estrella+fondo. El cálculo de incertidumbres en flujos debe tener en cuenta varios factores

- **Ruido fotónico** (en inglés, *shot noise*). Es un ruido intrínseco de la señal, siendo su distribución poissoniana cuando medimos los flujos en e^- (no lo es si medimos en ADUs como es nuestro caso, debiendo hacer la transformación utilizando la ganancia).
- **Ruido de amplificador** (ruido de lectura), N_r. Es un parámetro que suministra el fabricante, pero se puede estimar en AstMon por diferentes vías como las propuestas en [QSI] y [Bui].

- **Ruido térmico** (corriente de oscuridad), N_t. Este parámetro se conoce también para la cámara QSI 583, resultando casi despreciable frente al ruido de amplificador a tiempos de exposición bajos y frente al ruido fotónico a tiempos de exposición elevados.

De esta forma, si estamos midiendo el flujo de una determinada fuente (suponemos en principio que ocupa varios píxeles en la imagen y que pretendemos determinar su fondo), su incertidumbre se puede estimar como

$$\Delta F_{total}(c) = \sqrt{\frac{F_{total}(c)}{g} + N_{pix}\frac{N_r^2}{g^2} + N_{pix}\frac{N_t^2}{g^2}} \tag{3.10}$$

Si estamos interesados en una medida por ejemplo de fondo, en que pretendemos calcular un valor promedio por píxel, la expresión anterior adopta la forma

$$\bar{F}(c/px) = \frac{F_{total}}{N_{pix}} \qquad \Delta\bar{F}(c/px) = \sqrt{\frac{\frac{\bar{F}(c/px)}{g} + \frac{N_r^2}{g^2} + \frac{N_t^2}{g^2}}{N_{pix}}} \tag{3.11}$$

Conviene explicar por qué se ha utilizado la mediana o un ajuste por gaussiana en vez de la moda o una media. La razón básica es que, al estar midiendo flujos relativamente pequeños para el fondo, un píxel cuyo valor se desvíe mucho (outlier) respecto al promedio del resto nos modificará el promedio del conjunto notablemente. La media es por tanto un estadístico muy sensible frente a valores puntuales altos, y no es por tanto apropiada en nuestro estudio del fondo de cielo.

En cuanto a la moda, estaríamos tentados a pensar que es el estadístico ideal para nuestro propósito ya que no es sensible a píxeles cuyo valor se desvíe mucho del valor promedio. Es de esperar que los píxeles de fondo sean mucho más abundantes, con lo que dominarán en el histograma de brillos. Por otra parte el ruido de lectura (recordemos, dominante a tiempos de exposición bajos y señales pequeñas) en la imagen en principio es de esperar que tenga una forma más o menos simétrica, dando una distribución tipo gaussiana. Esa gaussiana tendrá un máximo que coincide precisamente con la moda (el valor que más se repite).

Sin embargo, en la práctica tiene algunas desventajas ya que es un estadístico débil cuando el número de píxeles de medida es relativamente bajo (será nuestro caso). No es raro que aparezcan casos en que la muestra sea bimodal, o que la moda no coincida con el centro de la distribución. Además no es tan rápido de calcular como la mediana o la media, ni tan robusto como un ajuste por gaussiana.

La debilidad de la moda como estadístico cuando la muestra es pobre se puede poner de manifiesto con un par de ejemplos reales que se tomaron durante la construcción de los módulos, para dos estrellas diferentes,

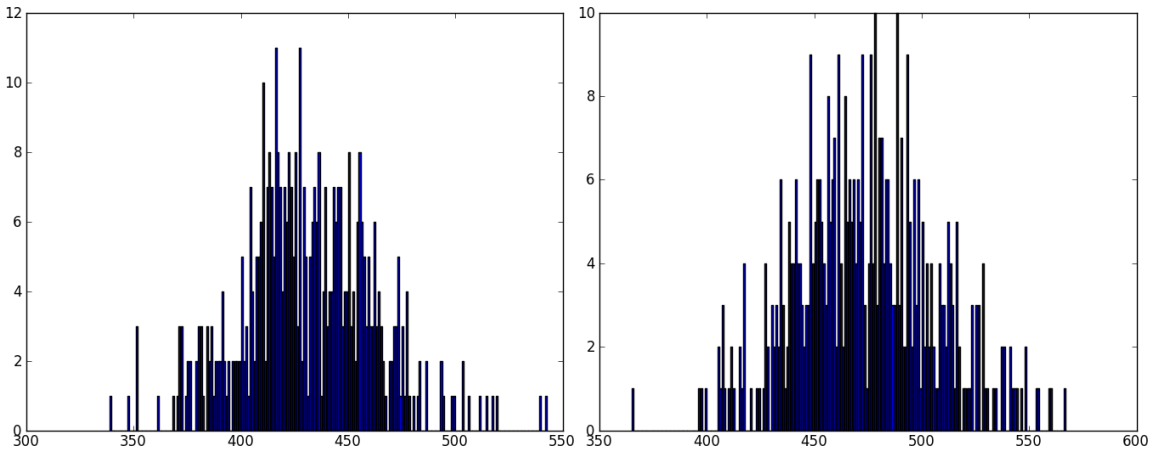

Figura 10: Distribución típica del número de píxel con un determinado flujo (en cuentas)

Evidentemente, se observa que la distribución a grandes rasgos podría aproximarse por una gaussiana, pero en ambos casos tenemos dos modas , y ambas están situadas algo alejadas del centro real de la distribución. La moda queda entonces como un estadístico a tener en cuenta cuando tenemos muchos píxeles medidos.

Algunos autores[Pat03], proponen unificar intervalos en el histograma (para hacer este más robusto) y a continuación aplicar ajustes que tengan en cuenta que en realidad, para muchos casos en los que tenemos fuentes que contaminan un porcentaje elevado de píxeles de los cuales queremos determinar el fondo, el ajuste gaussiano que estamos proponiendo no es tan apropiado, sino que la distribución tiende a tener una cola de flujos elevados más pronunciada.

[Pat03] menciona igualmente la posibilidad de dividir la muestra en fragmentos y hacer una estadística sobre los valores obtenidos en cada fragmento para descartar aquellos que contienen flujo no deseado.

En nuestro caso, la contaminación de objetos en las muestras de fondo es muy baja, con lo que el planteamiento propuesto es más simple. No obstante, en un futuro se podría pensar en mejorar este punto y hacerlo más robusto para, por ejemplo, poder descartar el flujo de algún elemento extenso como es la Vía Láctea si así se desease para alguna aplicación.

Todas estas formas de calcular flujos "medios" (usando la media, mediana, moda y un ajuste a una gaussiana), junto a las funciones que representan los histogramas se recogen en el módulo `estadística.py`. En el caso de la gaussiana, se puede calcular también la anchura típica de la distribución (desviación estándar), mientras que para el resto calcularíamos la desviación típica.

Esto nos sirvió inicialmente para tener una estimación de las incertidumbres (antes de tener caracterizado con precisión la ganancia, ruido térmico y ruido de lectura de nuestro dispositivo). Además, es un importante test para comprobar que el hecho de que las incertidumbres sean algo elevadas en el cálculo de brillo de fondo de cielo no es casual. Esto se discutirá más adelante.

Determinación de otros parámetros fotométricos Determinado el flujo, podemos calcular algunos parámetros que necesitaremos para hacer la calibración fotométrica (esto es, ajustar los datos a la recta de Bouguer). Estos cálculos se realizan a través de la función `parametros_bouguer` del módulo `deteccion_estrellas.py` [A.18]

Los cálculos de errores (por propagación normal) serían, suponiendo que no tenemos incertidumbre en la medida de magnitudes (es un dato que tomamos de las tablas) o que al menos

será despreciable frente a la incertidumbre asociada al flujo medido,

$$\Delta(M + 2{,}5 \log F) = \Delta(2{,}5 \log F) = \frac{2{,}5}{\ln 10} \frac{\Delta F_{estrella}}{F_{estrella}} \tag{3.12}$$

Representación gráfica Nos queda representar gráficamente, en caso de estar analizando una única imagen y que así se desee, el fits que se está analizando junto a las estrellas en catálogo, estrellas detectadas por nuestros procedimientos y los círculos de medida que se utilizan.

Este punto es muy interesante de cara a ver que realmente los tamaños de estos círculos que se estiman son adecuados para cubrir bien la estrella y están correctamente situados y es la forma actual de ajustar (por prueba y error) los parámetros de astrometría que deberemos introducir en el fichero de configuración para que resolver la imagen. Utilizamos para ello el paquete *Matplotlib* de *Python*. Véase `deteccion_estrellas.py` [A.19]

De paso, en el fragmento de código mencionado se introduce una llamada a una función que sustituye los píxeles de posiciones de estrellas por el valor de fondo. Con esto se pretende que las medidas de fondo de cielo en el último apartado sean más robustas. Una muestra de las imágenes que se obtienen con este procedimiento se muestra en la figura siguiente.

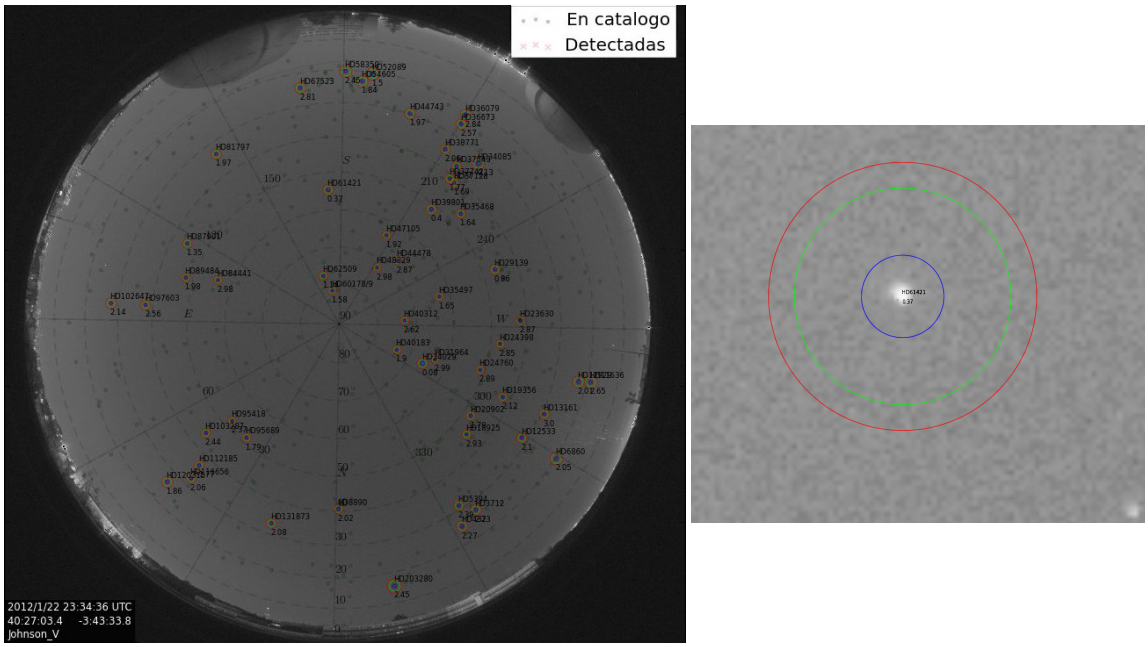

Figura 11: Imagen generada por el paquete PyAstMon-UCM en el que se observa la imagen cruda (en escala logarítmica). Superpuestos, los círculos de medida centrados en las estrellas que se han podido detectar y medir. A la derecha, un recorte centrado en una estrella.

Aquí concluye la función de búsqueda de estrellas y medida de flujos. Ya tenemos todos los datos necesarios en la tabla `tabla_fotometria` para representar y ajustar la recta de Bouguer y generar un mapa de brillo de fondo de cielo, tarea de la que se encarga el módulo `calibracion_fotometrica.py`

3.2.6. Calibración fotométrica

Ajuste de datos a la recta de Bouguer En el apartado 2.5.6 discutíamos las ecuaciones básicas que son necesarias para calibrar nuestro instrumento y estudiar la extinción atmosférica. Básicamente, se trata de ajustar la expresión (2.4) con los datos para cada una de las estrellas. Véase `calibracion_fotometrica.py` [A.20]

La estructura básica de este procedimiento es, como vemos:

1. Obtenemos de `tabla_fotometria` los valores de $y = m + 2{,}5 \cdot \log_{10} F_\nu$, Δy y $x = \chi = \sec(z)$ calculados en `deteccion_estrellas.py`.

2. Realizamos una regresión lineal con la función `theil_sen` del módulo `estadistica.py`.

3. Obtenemos los parámetros de la regresión (pendiente y ordenada en el origen, junto a sus incertidumbres), que se relacionan con el coeficiente de extinción ($-$ pendiente) y la constante instrumental de nuestro instrumento en esa banda (la ordenada en el origen).

4. Dado que durante el proceso se ha aplicado un sigma clipping (rechazo de valores atípicos que pudieran existir en `tabla_fotometria`), se vuelven a obtener los puntos que realmente se utilizan en el ajuste y que se mostrarán en la recta de Bouguer.

Es importante nota que al contrario de lo que es habitual, en nuestro análisis la regresión no se ha obtenido por mínimos cuadrados, sino que se ha intentado buscar un estimador más robusto frente a problemas en la muestra elegida, tales como la presencia de algún punto muy desviado.

En efecto, es muy típico cuando estamos utilizando estrellas estándar para realizar una calibración fotométrica, encontrarnos que el ajuste no es muy bueno, y que mejora notablemente cuando eliminamos alguna estrella. Este paso es fácil de ejecutar cuando el análisis se hace de forma interactiva, pero en un caso donde se busca automatizar lo máximo posible todo el proceso como el que nos preocupa en este trabajo, esto se vuelve más complejo. Podemos buscar estrategias como implementar un *sigma clipping* para eliminar datos muy desviados, pero en cualquier caso necesitaríamos un primer ajuste fiable para poder llevar a cabo este descarte de puntos.

De modo que uno de los problemas que se intentó atajar desde el principio es buscar algo que nos permita realizar ajustes lineales pero que al contrario que el método de mínimos cuadrados, sea robusto frente a la presencia de algún punto con una dispersión alta. Así, llegamos a un estimador muy sencillo que cumple estos requisitos: el estimador de *Theil-Sen*.

El método para determinar los parámetros de regresión es el siguiente[Wikc]:

1. Para cada par de puntos, determinamos un valor de la pendiente (nótese que esto sólo se puede hacer para parejas de puntos con distinto valor de x.

2. El valor de la pendiente ajustada será la mediana de todas las pendientes calculadas en el punto anterior.

3. Calculamos las ordenadas en el origen $y_i - m \cdot x_i$ (siendo m la pendiente).

4. El valor de la ordenada en el origen se toma como la mediana de las ordenadas en el origen determinadas anteriormente.

Nos hemos ajustado a este procedimiento, pero con algunas variaciones, a saber

- Intentamos estimar una incertidumbre tanto para la pendiente como para la ordenada en el origen tal y como se calculaba en el caso de una regresión lineal por mínimos cuadrados[Lab].
- Al calcular tanto medianas como ordenadas en el origen, se introduce un sigma clipping para descartar los puntos que más se desvían del estimador.

En [Wikc] realizan una pequeña comparativa en una muestra de puntos a los que se ha añadido ruido para ver que influencia tiene éste en la regresión

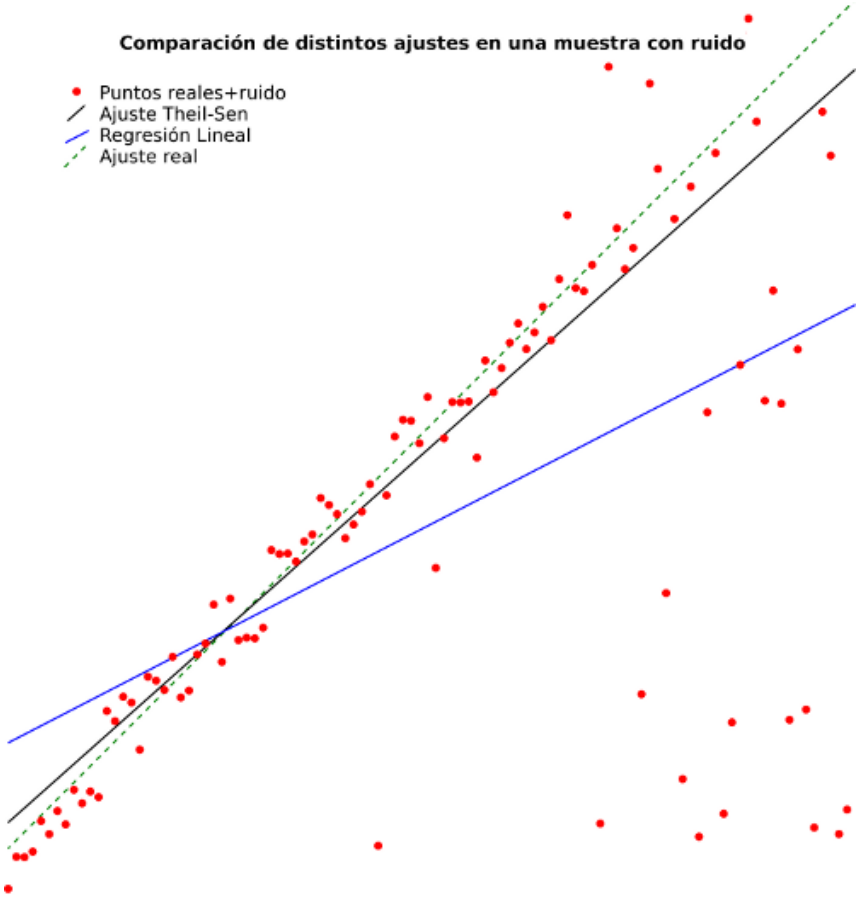

Figura 12: Comparativa de distintos tipos de estimadores en una regresión lineal presentada en [Wikc]

Ante la presencia de ruido, la regresión lineal por mínimos cuadrados, al intentar minimizar también el cuadrado de la desviación del ruido, deforma completamente la recta de regresión, siendo el resultado bastante malo. Theil-Sen por contra, aunque también se ve afectado, preserva un buen ajuste a la masa de puntos.

Igual que en el caso de la regresión por mínimos cuadrados, que tenemos un coeficiente de correlación (de Pearson) para tener una idea de la calidad del ajuste, en el caso de la regresión de Theil-Sen disponemos del coeficiente de correlación de Kendall-Tau como indicador (el coeficiente de correlación de Pearson no es aplicable tal cual, o al menos no tiene el significado a

que nos tiene acostumbrados, pudiendo tomar valores por encima de 1 para ciertos conjuntos de datos).

El procedimiento para el cálculo de este coeficiente (descrito en [[Wikb]] se basa en hacer un conteo de la diferencia entre el número de valores de pendiente positiva y el número de valores de pendiente negativa, todo divido entre el total de casos.

$$R_{Kendall-tau} = \frac{N_{pen>0} - N_{pen<0}}{N} \tag{3.13}$$

es evidente que en un ajuste perfecto, siempre que la pendiente no sean nula (en cuyo caso nos veríamos obligados a utilizar algunas de las variantes que se construyen sobre este coeficiente de correlación, conocidas como tau-b, tau-c, etc). Como trabajo futuro en este sentido, puede ser interesante investigar si alguna variante del método da resultados más optimistas cuando la pendiente es pequeña como es el caso del tipo de ajuste que nos concierne (las extinciones en noches fotométricas pueden ser del orden de 0.15-0.20 para las bandas V y R en el observatorio UCM, observándose que una pequeña cantidad de ruido en las medidas de flujo da lugar a unos valores del coeficiente de correlación aparentemente bastante pequeños).

Hecha la calibración fotométrica y determinadas la constante instrumental y el coeficiente de extinción, la función `recta_bouguer.py` representa opcionalmente los puntos experimentales, el ajuste de los puntos a la recta de Bouguer y los valores de constante instrumental y coeficiente de extinción en una gráfica, que puede representarse por pantalla o guardarse en un fichero (este punto dependerá de los parámetros de entrada que se introduzcan en el proceso, como veremos cuando mostremos el script que ejecuta todo el análisis).

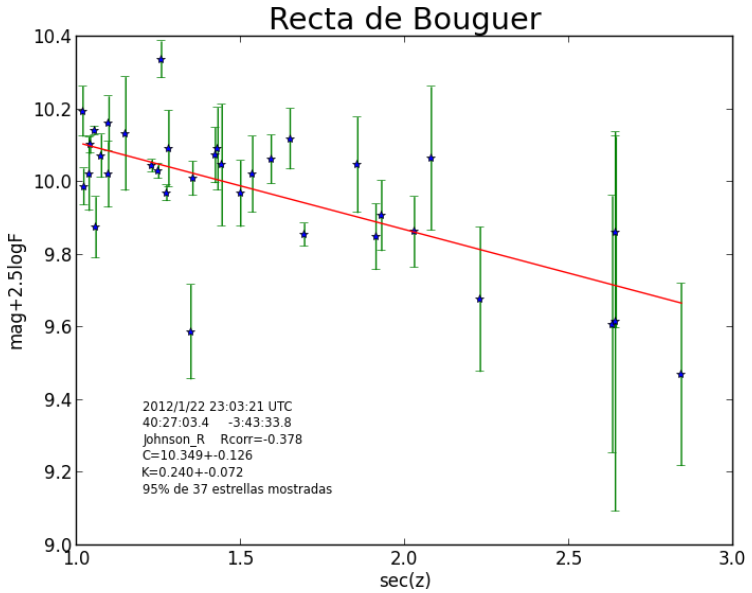

Figura 13: Ajuste a la recta de Bouguer (sin fijar la cte instrumental) generado por PyAstMon-UCM. Con objeto de poder medir en noches no fotométricas el valor de la extinción, se puede fijar el valor de la cte instrumental en el fichero de configuración.

3.2.7. Mapa de brillo de fondo de cielo

Llegamos al último punto de nuestro análisis, generar un mapa de brillo de fondo de cielo y dar valores para esta magnitud en determinadas posiciones de especial interés (actualmente se da en el cénit, con una apertura de unos 10°, pero podría ampliarse a otros puntos de forma sencilla).

En este caso, los flujos que se están estudiando son flujos de fondo, con lo que en principio no habría que hacer ninguna corrección. Veremos que esto no es así, estando obligados a restar una señal de BIAS presente en la imagen (se tomará en zonas no iluminadas de esta, fuera del círculo iluminado por el ojo de pez) que no es en absoluto despreciable. El hecho de que existiese una corriente de bias tan elevada fue uno de los motivos que nos hicieron sospechar que las imágenes que generaba AstMon no estaban calibradas con DARKS (de haberlo estado, uno esperaría que esa corriente de bias fuera más próxima a 0).

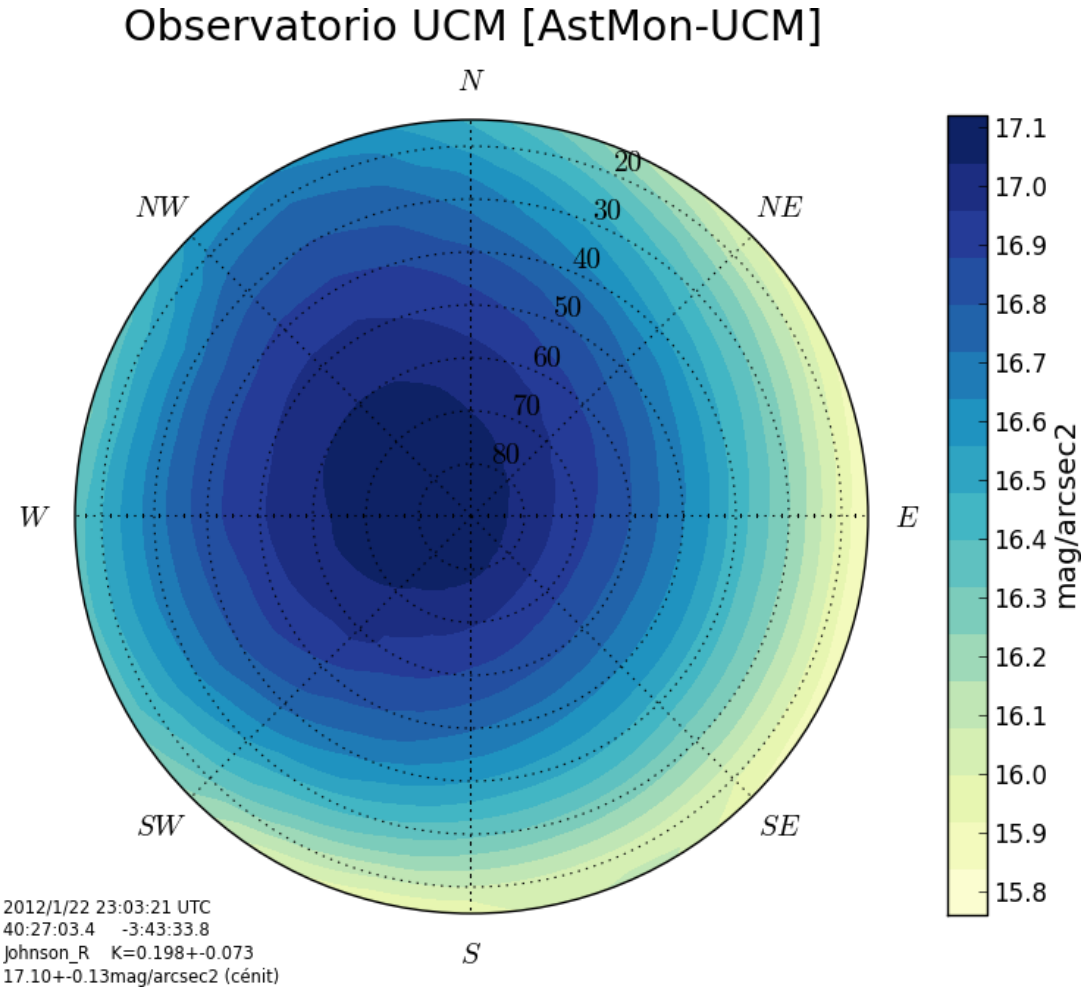

Figura 14: Ejemplo de mapa de brillo de fondo de cielo generado por PyAstMon-UCM. En la imagen, tomada en el filtro R, se observa claramente como el observatorio UCM está sometido a un tipo de contaminación lumínica que no es homogénea en Azimut, sino que presenta claramente una zona más oscura por la dirección NW.

El fundamento físico de los cálculos ya se describió en el apartado 2.5.7, y más concretamente con la ecuación [2.5]. Véase `calibracion_fotometrica.py` [A.21] para ver la implementación en PyAstMon-UCM.

El resto de la función básicamente repite ese cálculo para una serie de posiciones en el cielo. Esto nos da una masa de puntos razonablemente bien espaciada que nos permite, mediante interpolación, generar un grid equiespaciado en azimut y alturas para poder representar el mapa de brillo de fondo de cielo. Inicialmente, esta parte generaba mapas con la misma estructura que las propias imágenes de AstMon-UCM incluyendo únicamente el código de color.

Recientemente, se ha reformado para generar mapas similares a los que se crean en el proyecto NixNox[13], utilizando como referencia ya no la propia disposición del CCD y el objetivo, sino las coordenadas reales en el cielo. Véase `calibracion_fotometrica.py` [A.22]. Con esto, se abre la posibilidad a comparar diréctamente los resultados obtenidos a los que se pueden obtener por ejemplo con dispositivos como el fotómetro SQM. Véase la sección 4.2.2).

La representación del mapa de brillo de fondo de cielo se realiza utilizando el módulo `calibracion_fotometrica.py` [A.23].

3.2.8. Fichero de configuración

Uno de los objetivos de la creación de este paquete de herramientas es que no sólo sirvan para nuestro dispositivo en concreto, sino que puedan ser exportables de forma sencilla a otro tipo de imágenes tomadas con cámaras distintas al CCD utilizado y orientadas de forma diferente.

Por ello, es muy recomendable intentar separar lo máximo posible aquellas variables que identifiquen de alguna forma nuestro dispositivo o la configuración que este presenta. Nos referimos por ejemplo a la solución astrométrica, que es propia de nuestro dispositivo y que hará falta cambiar al intentar aplicar nuestros algoritmos en otras imágenes.

También las constantes instrumentales para los casos en que se realice un ajuste a una recta de Bouguer dejando la ordenada en el origen fija (este paso vimos que era necesario cuando tratamos de medir el brillo de fondo de cielo en noches no fotométricas, tratándose de un método muy utilizado por varios dispositivos de medida del fondo de cielo).

Al final, en vez de dejar estas constantes definidas dentro de los módulos, se optó por generar un fichero de configuración en claro (de forma que fuese sencillo de editar y visualizar de forma externa). Véase `fichero_configuracion.py` [A.24] y [A.25]

El módulo de lectura es realmente sencillo. Hace una llamada a una función auxiliar encargada de recorrer el fichero de configuración (dicha función no tiene en cuenta espacios ni aquellas líneas que comienzan con #, es decir, que están comentadas), almacenando cada opción junto a su valor en un vector. A continuación, introducimos un bucle que va comprobando si están presentes las opciones que son necesarias y va guardando su valor dentro de la clase Imagen. Esa clase se devuelve finalmente con parámetro de salida al programa principal.

El fichero de configuración que utilizamos se muestra en `config_astmon.py` [A.26]

Las variables que en él aparecen se describen a continuación:

`obs_lat` y `obs_lon`: Coordenadas geográficas del observatorio.

`despl_x_ptos` y `despl_y_ptos`: Si el centro de la imagen no coincide con el cénit, se puede indicar el desplazamiento o corrección que hay que aplicar.

`cte_AZ` Rotación que hay que aplicar a la proyección ZEA "canónica" para que coincida con la orientación de nuestra imagen.

[13] NixNox es un proyecto de la Sociedad española de Astronomía SEA para localizar lugares oscuros donde poder disfrutar de cielos estrellados

`cte_R` Factor de escala de la imagen.

`numero_max_estrellas`: Número máximo de estrellas que se leerán del catálogo. Cuanto mayor sea, mayor muestra de estrellas susceptibles de ser medidas, pero la búsqueda tardará más tiempo en completarse.

`altura_min`: Altura mínima para medir las estrellas, las que estén más bajas se descartan en la fotometría.

`radio_base`: Radio base de los círculos de medida. Sobre este radio se aplican varios factores de corrección (ya descritos) hasta llegar al tamaño real de los círculos que se utilizan. Hay que jugar con este parámetro, pues a mayor radio más probabilidad de encontrar la estrella (sobre todo a alturas bajas), pero más tiempo tarda el proceso de búsqueda y medida, además de aumentar las probabilidades de medir varias estrellas a la vez.

`lim_deteccion_base`. Factor por encima del fondo que ha de tener el flujo de la estrella para que se considere detectada. Nuevamente, es un factor base, nuestro programa aplica una serie de correcciones al respecto.

`bits_ccd`: nuestro programa utiliza este parámetro como forma de descartar estrellas que poseen píxeles saturados o próximos a la saturación (pérdida de linealidad) al medir flujos. Esas estrellas se descartan automáticamente puesto que su flujo no se puede determinar correctamente[14], sino que cualquier estimación lo infravalorará.

`ganancia`: parámetro propio del CCD, se utiliza para hacer la estimación de errores en la medida de flujos, en e^- / ADU.

`ruido_lectura`: Medida del ruido de lectura del CCD (de forma abreviada RON), en e^-.

`ruido_termico`: Ruido de caracter térmico. Aumenta con el tiempo de exposición, se ha de introducir en e^- / s.

`magnitud_maxima`: Magnitud máxima de las estrellas (filtro V) para ser utilizadas. Las estrellas más débiles se descartan en fotometría, aunque se tienen en cuenta para sustituir los valores de los píxeles que ocupan por el valor medio del fondo de cara al cálculo de brillo de fondo de cielo.

`lim_Kendall_tau`: Límite inferior en el coeficiente de correlación para que la noche sea considerada fotométrica, y por tanto, realizar un ajuste complejo sin fijar la constante instrumental.

`termino_color_#`: Valor, error. Determinados experimentalmente, son correcciones que hay que aplicar debido a las características propias de nuestros filtros. Estas correcciones no se aplican de forma automática en los mapas de fondo de cielo, siendo necesario corregir de este efecto a posteriori.

`cte_inst_#`: Valor, error. Se determinan experimentalmente analizando una o varias imágenes. Son los valores para la constante instrumental en cada filtro de nuestro dispositivo.

[14] Es evidente que se pueden buscar estrategias para dar una aproximación al mismo por ajuste a gaussianas en 2D del perfil de la estrella por ejemplo, pero no es un proceso sencillo y realmente aporta poco al análisis "global" que estamos planteando

`nivel_fondo_#`: Niveles de fondo que se desean incluir en el mapa de brillo de fondo de cielo. Si no se especifican (se recomienda hacerlo si se desean analizar noches enteras y generar mapas que sean "coherentes" en el código de colores entre sí), se estima de forma automática para representar lo mejor posible los datos.

`area_pixel`: Área proyectada de un píxel en el cielo. Valor calculado teniendo en cuenta la focal del objetivo y el tamaño físico de cada celda del sensor suponiendo que la proyección es tal que preserva áreas (el caso de los objetivos utilizados en nuestro análisis, Sigma 4.5mm y Sigma 8mm)[15]. Es necesario para estimar en $mag/arcsec^2$ el brillo de fondo de cielo.

`titulo_mapafondo`: Título que hay que darle al mapa de brillo de fondo de cielo (resultado final de nuestro análisis de imágenes individuales).

`angulo_interp` y `caidas_interp`: ángulo desde el eje de la imagen e iluminación relativa del sensor para corregir de viñeteo.

`flatfield_#` y `darkframe`: Ruta a los Flats y Dark que se utilizarán para reducir las imágenes. Si se especifican, no se aplica la corrección de viñeteo.

Algunas variables, tales como las referidas a la corrección de viñeteo son opcionales, sin embargo, la mayoría son necesarias para el correcto funcionamiento de nuestros procedimientos y se recomienda encarecídamente asignarles un valor lo más realista posible.

3.2.9. Script lanzador (programa principal)

Hasta ahora, todo lo descrito son módulos. Tienen una entrada y una salida, realizando una serie de operaciones entre medias. Por sí solos, estos módulos tienen poco sentido, no siendo utilizables de forma directa. Nos hace falta generar un script lanzador o programa principal, que será el que interactué con el usuario, tomando como entrada una fecha o imagen, realizando los cálculos que se pidan (que se indicarán como parámetros de entrada) y devolviendo por pantalla o almacenando en un fichero los resultados de dichos cálculos.

La creación de este script ha sido gradual, y aun hoy dedico tiempo a pensar cuál es la mejor forma de presentarlo. De hecho, en el pasado más que un script lanzador manejaba un conjunto de scripts, cada uno con un cometido distinto (uno para analizar una imagen única, otro para analizar una noche entera, otro para analizar un mes entero y otro para analizar todas las imágenes almacenadas en AstMon-PC). Esta aproximación, que surgió a medida que iban necesitándose realizar estos análisis, era poco práctica y compleja de mantener, de modo que en Marzo de 2012 se reemplazó parte del código para hacerlo más compacto. Aun así, dada su longitud, se pasará a explicar por partes.

En una primera parte, destinada a leer los parámetros de entrada, distinguimos a su vez varias secciones.

[15] En el futuro, será necesario buscar la forma de tener en cuenta casos en que esto no es así ya que otros objetivos pueden estar empleando proyecciones que que para cada posición cambien el área muestreada de cielo.

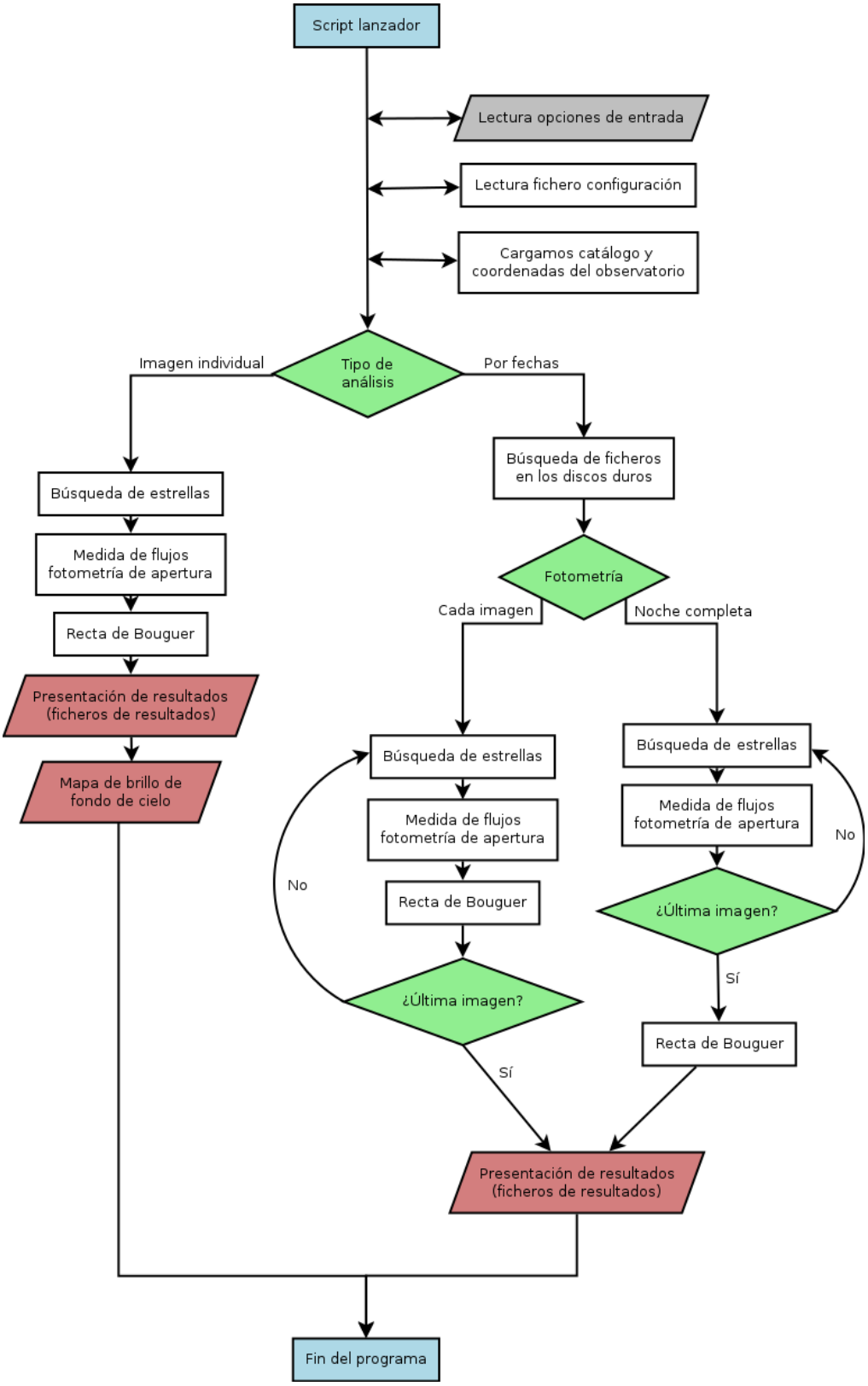

Figura 15: Diagrama del funcionamiento PyAstMon-UCM y el script lanzador

Ayuda del programa Todo programa debería disponer de una breve ayuda que identifique que parámetros de entrada pueden especificarse y que aparezca bien invocando dicha ayuda (se ejecuta el programa `lanzador_fotometria.py` con el parámetro `-h`) o bien cuando se introduce un parámetro de forma incorrecta. La función encargada de mostrar la ayuda está en `lanzador_fotometria.py` [A.27].

Detección de errores Tras el código que muestra la ayuda, encontramos tres funciones que se encargan de localizar errores típicos en la introducción de parámetros en el programa, avisando al usuario del error y mostrando la ayuda. El código asociado se puede encontrar en `lanzador_fotometria.py` [A.28].

Cálculo de fechas que se analizarán Se define a continuación la función que se encargará de, a partir de una fecha introducida, mostrar todas las fechas susceptibles de análisis en `lanzador_fotometria.py` [A.29].

Esta función, como vemos, es algo compleja. La razón de que así sea es que admite bastante libertad en la forma de expresar la fecha. Se puede indicar una fecha completa (en formato [año,mes,día]), de modo que se integraran las medidas de todas las imágenes de esa noche y se ajustarán de forma conjunta a la recta de Bouguer. Para dar mayor libertad en el formato de entrada, hay que resaltar que nuestro programa es capaz de procesar fechas en formato corto. Es decir, [2010,09,04] es equivalente a escribir [10,9,4].

Se puede también suprimir el día, dejando [año,mes], con lo que se recorrerán todos los días de ese mes, analizándose como antes los datos de cada noche de forma conjunta. Incluso podemos suprimir el mes y dejar todo sólo [año], con lo que estaremos indicando al programa que analice todos los datos de ese año, tratándolos noche por noche.

Por último, si en vez de introducir la fecha en formato [años,mes,día] introducimos "todo", le estaremos pidiendo al programa que analice todos los ficheros de todos los discos duros de AstMon-PC. El uso de esta opción no es aconsejable, en tanto en cuanto puede tardar varios días en completarse y consumir una gran cantidad de recursos en AstMon-PC.

Fichero de entrada Es posible que en algunos casos nos interese no analizar una determinada fecha, sino alguna imagen en particular, bien sea de AstMon u otra imagen fits tomada con otro dispositivo. En este caso, en vez de pasar al programa el parámetro `-f [anyo,mes,dia]` , lo que haremos será pasarle el parámetro `-i fichero_entrada`. Salvo que el fichero fits se encuentre en el mismo directorio que el script lanzador, será necesario especificar la *ruta completa*. En sistemas operativos Windows podría ser de la forma `C:\Documents and Settings\usuario\Mis documentos\fichero.fits`, mientras que sistemas tipo Unix, será algo de la forma `/home/usuario/fichero.fits`. Véase `lanzador_fotometria.py` [A.30]

Ficheros de salida El paquete de software que estamos presentando puede mostrar ciertos datos por pantalla, guardarlos en un fichero u omitirlos (incluso se pueden omitir ciertos cálculos no imprescindibles si no se van a mostrar[16]). La función que se presenta a continuación se encarga de tomar decisiones sobre si se muestra un cierto resultado y como se muestra en base a los parámetros que se introducen en la entrada. `lanzador_fotometria.py` [A.31]

[16]Un ejemplo es la generación del mapa de brillo de fondo de cielo, que consume mucho tiempo de cálculo y no siempre será necesario en nuestro análisis, siendo a veces suficiente con determinar el fondo de cielo en un punto, un objetivo mucho más modesto y sencillo

La forma de decidir como se muestran los datos es la siguiente. Supongamos por ejemplo el caso de la generación del mapa de brillo de fondo de cielo. Si acudimos a la ayuda del presente script, veremos que se invoca con el parámetro `-of`.

- Si no se especifica `-of`. El mapa de brillo de fondo de cielo no se muestra por pantalla, tampoco se guarda en un fichero.
- Especificando `-of` a secas, el programa nos mostrará por pantalla el mapa de brillo de fondo utilizando una herramienta de visualización que nos permite ampliar zonas (zoom), guardar la imagen o fragmentos de ella, etc. Este es un modo, digamos, interactivo. Es muy apropiado por su flexibilidad a la hora de visualizar las gráficas para analizar imágenes fits individuales, pero poco apropiado cuando tratamos de analizar un gran número de imágenes en modo no-interactivo.
- Si se indica `-of /ruta/al/fichero.formato`, nuestro programa no mostrará por pantalla el mapa, sino que lo guardará en el fichero especificado. Como es natural, el usuario que ejecuta el programa deberá tener permisos de escritura en ese directorio. De igual forma, el nombre del fichero deberá acabar en algún formato de imagen conocido (*png* o *pdf* son buenas opciones[17]). No hace falta indicar la fecha o el filtro utilizados, el programa detecta estos parámetros de forma automática viendo la cabecera del fits, y automáticamente completa el nombre del fichero para incluirlos, en la forma `/ruta/al/fichero_fecha_filtro.formato`.

[17] En caso de estar manejando la opción para generar tablas, `-or`, `-oft` o `-ot`, se puede usar como formato *txt* o *dat*

4. Dispositivos sencillos de medida de fondo de cielo

AstMon-UCM es un dispositivo muy completo pensado para ser utilizado en un observatorio astronómico de manera permanente. Debido a su diseño, complejidad y funcionalidades tiene un precio elevado. Aunque existen versiones transportables de precio algo menor, hemos querido explorar la posibilidad de utilizar sistemas más sencillos para medida de brillo de fondo de cielo que sean transportables para medidas de campo. Aprovechando AstMon-UCM podemos comparar los resultados de todos los equipos.

4.1. Fotómetro SSP3

El Observatorio UCM dispone de un fotómetro estelar Optec SSP-3 que se utiliza en las prácticas de observación astronómica. Dispone de filtros fotométricos de Johnson-Coussins y permite realizar fotometria absoluta. Este fotómetro integra en una apertura fija de 1 mm de diámetro que cubre sobre el cielo un área que depende de la escala de placa de la óptica que se le acople.
El prof. Jaime Zamorano realizó el 10 de febrero de 2012 un experimento utilizando el fotómetro SSP-3 sin utilizar óptica alguna. Apuntó el fotómetro hacia el cénit y midió en diferentes instantes al comienzo de la noche. Medidas de laboratorio en LICA demostraron que el ángulo de aceptación de este sistema es de 15 grados.

Figura 16: Fotómetro SSP3. Imagen del sitio web de Optec, `http://www.optecinc.com/astronomy/`

Hora (TU)	V	B	Dark	V Neto	B Neto	-2.5 log (B/V)
19:30	33,2	12,1	6,03	27,17	6.07	-1.63
19:55	24,2	11,0	6,23	17,97	4.77	-1.44
20:22	20,7	10,4	6,30	14,40	4.10	-1.36
20:35	20,9	10,4	6,28	14.62	4.12	-1.38

Cuadro 3: Flujos medidos con el SSP3 el día 10/02/2012

Para comparar con los valores de brillo de fondo de cielo en esos instantes se tomaron los resultados que generaba AstMon-UCM. Para ello, se construyó un script (`comparacion_ssp3.py`, [A.42]) que tomaba un fichero FITS de brillo de fondo de cielo de AstMon para calcular un promedio del brillo de fondo de cielo (promediando flujos) en círculos en torno al cénit.

Hora (TU) / radio (px)	100	150	200	250
Filtro B				
19:33	17.863	17.838	17.816	17.794
19:52	18.083	18.067	18.048	18.030
20:24	18.258	18.243	18.230	18.217
20:32	18.249	18.236	18.224	18.213
Filtro V				
19:34	16.663	16.668	16.669	16.669
19:53	17.107	17.060	17.018	16.992
20:19	16.992	17.275	17.260	17.246
20:32	17.290	17.272	17.257	17.242

Cuadro 4: Brillos de fondo de cielo medidos con AstMon para comparar con el SSP3

Con estos datos, se puede realizar una calibración del SSP3 para establecer una transformación directa que, partiendo del número de cuentas que de el SSP3, obtengamos inmediatamente el brillo de fondo de cielo en $mag/arcsec^2$ de la siguiente manera,

$$B = 19{,}77 - 2{,}5log_{10}F_B(c/s) \tag{4.1}$$

$$V = 20{,}14 - 2{,}5log_{10}F_V(c/s) \tag{4.2}$$

Esto convierte al SSP3 en un fantástico medidor de brillo de fondo de cielo en bandas fotométricas usadas en Astronomía. Pero hay que tener en cuenta las siguientes consideraciones:

- El SSP3 tiene una corriente de oscuridad no despreciable
- El ángulo de aceptación (ángulo en el cual el fotómetro es capaz de recibir flujo), medido con el adaptador de 1.25"del SSP3, es del orden de 15° (con una caída brusca en cuanto nos separamos de dicho ángulo). Este ángulo de aceptación puede disminuirse utilizando tubos alargadores sobre el adaptador de 1.25".
- El flujo de cuentas medido es suficiente para medir en cielos brillantes como el de Madrid pero se necesitará una óptica adicional para aumentar la señal registrada en cielos oscuros.

El fotómetro SSP-3 con esa óptica sencilla sería un sistema muy eficaz para medir brillo de fondo de cielo en el campo ya que dispone de *display* y funciona con una batería de 9 V.

4.2. Fotómetros SQM

4.2.1. Mapas de brillo de fondo de cielo con fotómetros SQM

La compañía Unihedron (`http://www.unihedron.com/projects/darksky/`) ha desarrollado los fotómetros SQM (*Sky Quality Meter*) que vienen calibrados de fábrica (precisión de 0.1 mag) y muestran directamente medidas de brillo de fondo de cielo en una banda que abarca las bandas B y V de Johnson. El ángulo de aceptación de los SQM es de $FWHM \simeq 20°$.

Figura 17: Fotómetro SQM-L montado sobre un trípode fotográfico nivelado apuntando a 20 grados de distancia cenital para un cierto acimut. Imagen tomada por el prof. Jaime Zamorano.

La SEA posee varios de estos instrumentos portátiles y los ha puesto a disposición de distintas asociaciones de aficionados para tener mapas de brillo de fondo de cielo a lo largo de toda la península dentro del proyecto NIXNOX[18]. En el protocolo de toma de datos de NIXNOX se obtienen medidas en toda la bóveda celeste en un muestreo a 20, 40, 60, 80 grados de altura y 12 posiciones de acimut además de una medida en el cénit. De esta forma se puede construir un mapa de brillo de fondo de cielo con un dispositivo que en principio no está pensado para obtener mapas.

El jueves 10 de Mayo de 2012 se realizaron medidas entre las 22:00 y 23:30 (hora local, UTC+2) utilizando el dispositivo Unihedron SQM-L propiedad del prof. Jaime Zamorano (número de serie 3298) desde la azotea de la facultad. Fue una noche con algunas nubes altas dispersas a lo largo de la noche, que se desaparecieron hacia la mitad de las medidas. Para comprobar la repetibilidad de las medidas, se realizaron dos series de medidas, correspondientes a alturas 20°, 40°, 60°, 80°, 90° y 30°, 50°, 70°, 90°.

En la siguiente tabla representamos los datos recogidos (se representa azimut de forma que 0° equivalga al Norte, 90° al este, 180° al sur y 270°). A la mitad de las medidas, se detectó una desviación en el origen de azimuts (0°) de 10° hacia el oeste. Los datos se siguieron tomando con este offset para evitar introducir errores no deseados en las medidas, pero ha de tenerse en cuenta este hecho ya que las imágenes generadas con el software NIXNOX no están corregidas de este efecto.

[18] `http://www.sea-astronomia.es/drupal/nixnox`

alt/az	180	210	240	270	300	330	0	30	60	90	120	150
90	17.68											
80	17.44	17.47	17.55	17.63	17.66	17.69	17.67	17.65	17.61	17.53	17.49	17.45
70	17.32	17.41	17.48	17.61	17.67	17.66	17.65	17.55	17.46	17.37	17.32	17.32
60	17.07	17.16	17.23	17.35	17.45	17.52	17.49	17.38	17.24	17.06	17.00	17.03
50	16.92	17.07	17.29	17.41	17.48	17.53	17.48	17.30	17.13	17.00	16.92	16.91
40	16.62	16.66	16.79	16.99	17.03	17.18	17.23	17.01	16.79	16.62	16.54	16.56
30	16.61	16.60	16.83	16.94	16.91	17.09	17.23	16.86	16.61	16.42	16.37	16.34
20	16.46	16.25	16.48	16.59	15.74	16.62	16.93	16.72	17.04	16.11	16.01	16.23

Cuadro 5: Medidas con SQM-L en el observatorio UCM

Representando estos datos utilizando el programa NIXNOX desarrollado por el prof. Jaime Zamorano, se obtiene el siguiente mapa de brillo de fondo de cielo (se muestra también la imagen generada por el propio PyAstMon-UCM como comparación)

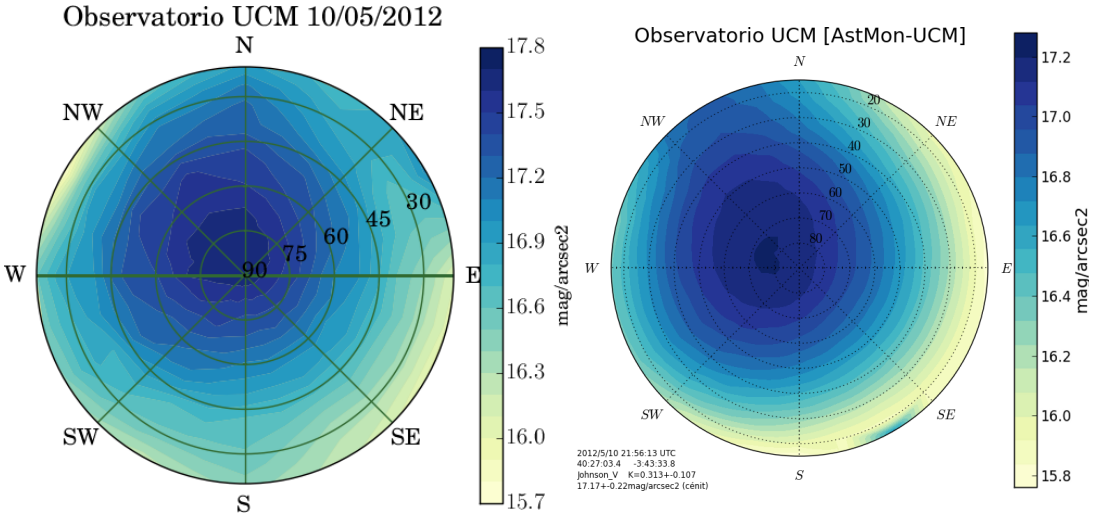

Figura 18: Mapa de brillo de fondo de cielo obtenido con el programa NIXNOX y las medidas del SQM-L. A su derecha, imagen de brillo de cielo generada por PyAstMon-UCM para el día 10/05/2012 a las 21:56 UTC (filtro Johnson V).

Nótese que los valores difieren de los que genera el SQM-L. Hay que tener en cuenta que la respuesta del SQM-L se ha diseñado de tal forma que equivale a una mezcla de filtros B y V, por tanto es de esperar que los valores que se obtengan sean algo intermedio entre los de bandas B y V de Johnson de AstMon.

4.2.2. Monitoreo de brillo de fondo de cielo con fotómetros SQM

Recientemente, Unihedron puso en el mercado dos nuevas variantes del SQM, ambas diseñadas para realizar medidas de forma más continuada y autónoma. Una de ellas se conecta a un PC por medio de un puerto USB; la otra por ethernet, siendo accesible vía internet sin necesidad de tener un PC conectado. El departamento de Astrofísica de la UCM posee uno de estos instrumentos (SQM-LE), pendiente de ser instalado en su posición definitiva.

Estos sistemas están diseñados para ser instalados en observatorios astronómicos u otros lugares donde se quiera monitorizar la variación de brillo de fondo de cielo en una determinada posición de cielo y trabajan durante toda la noche registrando y almacenando valores de fondo

de cielo. A pesar de que existen ya herramientas para leer los valores dados por el SQM (e incluso para subirlos a un servidor central: `http://www.sqm-network.com/index.php?lang=en`), el autor de estas lineas estuvo realizando algunas pruebas con ese SQM-LE y consideró interesante construir un pequeño y sencillo script que leyera los valores del SQM y los fuera almacenando en un fichero de texto para su posterior análisis[19]. El código está basado en un ejemplo que incluía el propio CD del SQM-LE, aunque posee modificaciones propias y correcciones.

El código que controla el SQM se puede encontrar en (`leer_sqmle.py`). Se divide en funciones de búsqueda [A.32] y lectura de valores [A.34 en adelante].

Si los datos generados se representan frente a la hora o a la altura del Sol (se ha generado otro script para ello, llamado `plot_fondocielo.py`, obtenemos lo siguiente

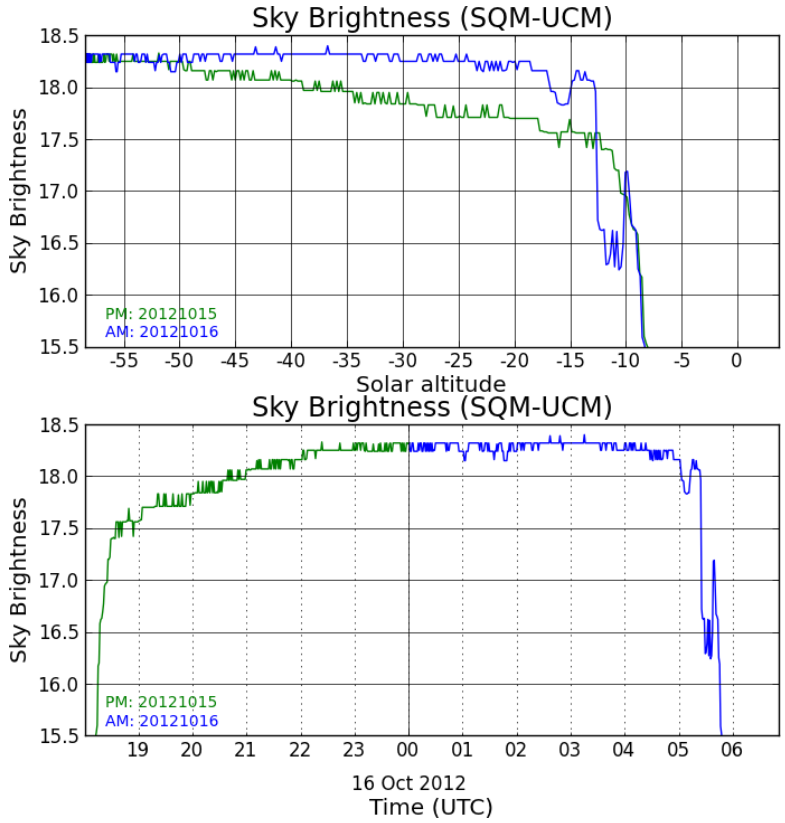

Figura 19: Medidas tomadas por el SQM-LE en la UCM, Madrid

Notamos que hay diferencias apreciables entre el brillo de fondo de cielo entre el anochecer y la madrugada. Así, aunque tracemos la magnitud observada frente a la altura del Sol (uno esperaría, en un sitio limpio y una noche fotométrica que ambas curvas colapsaran o fueran más próximas) vemos que el propio apagado de focos de iluminación de nuestra región a lo largo de la noche provoca variaciones en el brillo de cielo.

Este SQM-LE está actualmente instalado de forma provisional en una terraza de la facultad de CC. Físicas registrando datos de forma continua. En un futuro, y cuando se encuentre ya en su posición definitiva en la azotea, se automatizará la generación de estas gráficas cada noche.

[19] Actualmente, los 2 SQM-LE que posee el IAC en el OT y el ORM están funcionando con este mismo software.

Comparación con AstMon-UCM. 18 Mayo de 2012.

Aun estando en una ubicación temporal, podemos aprovechar que SQM-LE y AstMon-UCM están midiendo de forma simultánea para comparar los resultados de uno y otro instrumento. Esto se hizo en particular para la noche del 18 a 19 de Mayo (noche en la que además se tomaron medidas desde el observatorio de Yebes, Guadalajara, lo que puede ser interesante para una futura comparación de las condiciones en uno y otro sitio). El resultado de las medidas se resume en la siguiente gráfica elaborada por el prof. Jaime Zamorano,

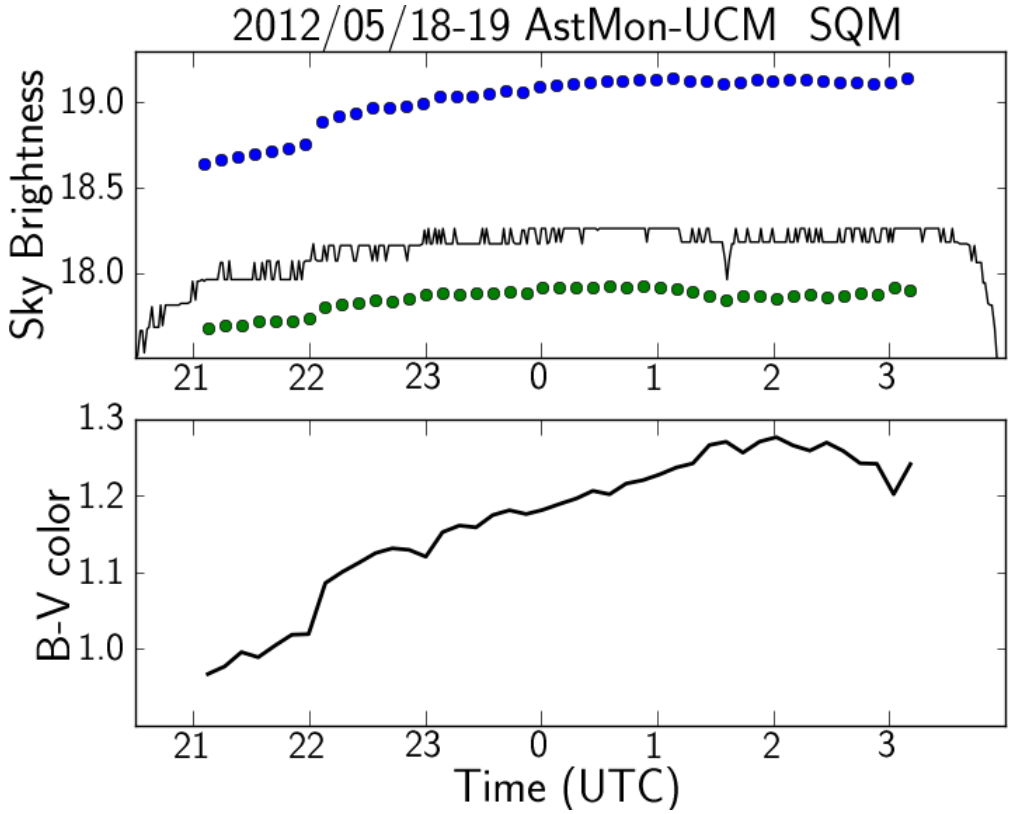

Figura 20: Valores del brillo de fondo de cielo en el cénit para registrados por SQM-LE y AstMon-UCM la noche del 18 al 19 de Mayo. Abajo se muestra el índice de color B-V correspondiente

Observemos lo siguiente,

- Las medidas del SQM caen entre las que se registran para los filtros B y V de AstMon-UCM (lógico ya que como hemos comentado la respuesta del SQM cubre las bandas B y V de Johnson), a pesar de lo cual la evolución que sigue es mucho más próxima a la que sigue el filtro V (el cielo en filtro B se oscurece entre 21 horas y 0 horas (TU) de forma notable).
- AstMon-UCM proporciona valores más precisos que el SQM (vemos que el SQM tiende a oscilar entre valores que distan más de $0{,}1^{mag}$.
- AstMon-UCM, por la forma en que está configurado, está preparado para medir tras el crepúsculo astronómico, no midiendo al anochecer. Tampoco mide durante el amanecer,

y durante el día se encuentra en un estado de "hibernación". Por contra, SQM mide de forma continua (todo el día), a pesar de lo cual sólo cuando existe una oscuridad suficiente es capaz de registrar valores válidos.

- La evolución del color no es simétrica, como notó el profesor Jesús Gallego. Además, presenta saltos a las 22^hTU y 23^hTU como pudieron observar el profesor Jaime Zamorano y los doctorandos Alejandro Sánchez y Francisco Ocaña. Existe una evolución hacia colores $B - V$ mayores a lo largo de toda la noche (el cielo como hemos comentado se hace más oscuro en banda B, pero no así en banda V, donde se mantiene casi constante).

  Las razones de esta evolución no están claras, puesto que si se debieran a apagado de alumbrado en la ciudad, debería haberse estabilizado el valor antes de lo que se observa (notamos aumento de brillo hasta pasadas las 01^h TU).

- SQM-LE posee mayor resolución temporal que AstMon-UCM, observación lógica dado que AstMon-UCM debe tomar imágenes (tiempos de exposición de $40s$ por filtro actualmente) en 3 filtros fotométricos, separados a su vez unos $200s$. Por contra, SQM-LE se puede programar para hacer medidas cada pocos segundos, con lo que cualquier evento transitorio (como el paso de una nube o un bólido por el cénit) tiene mayor posibilidad de ser trazado por SQM-LE que por AstMon-UCM.

4.3. Cámara réflex con objetivo ojo de pez

Una forma de fabricar un sistema parecido a AstMon pero más sencillo y barato consiste en utilizar una cámara digital dotada de objetivo ojo de pez que abarque toda la bóveda celeste. Las imágenes capturadas pueden ser analizadas de manera análoga a como se hace con las imágenes de AstMon para obtener parámetros de calidad astronómica del cielo y mapas de brillo de fondo de cielo.

Figura 21: Cámara fotográfica Canon con objetivo ojo de pez sobre trípode nivelado apuntando al cénit. Imagen cedida por el prof. Jaime Zamorano.

El principal problema de este sistema es que los filtros RGB que montan en la matriz de Bayer no se corresponden exactamente con los filtros típicos de Johnson (donde disponemos de estrellas calibradas). Habría que estudiar posibles correcciones que ayuden a paliar este problema. Esa matriz de Bayer, por contra, supone una cierta ventaja en tanto en cuanto permite capturar imágenes en 3 filtros diferentes de forma simultánea.

La noche del 24 al 25 de Marzo de 2012 Fernando Jáuregui, miembro del Planetario de Pamplona y AstroNavarra, utilizando una cámara Nikon D700 (cámara FullFrame) y un ojo de pez Sigma 8mm, tomó unas imágenes AllSky en formato .NEF dentro del proyecto NIXNOX

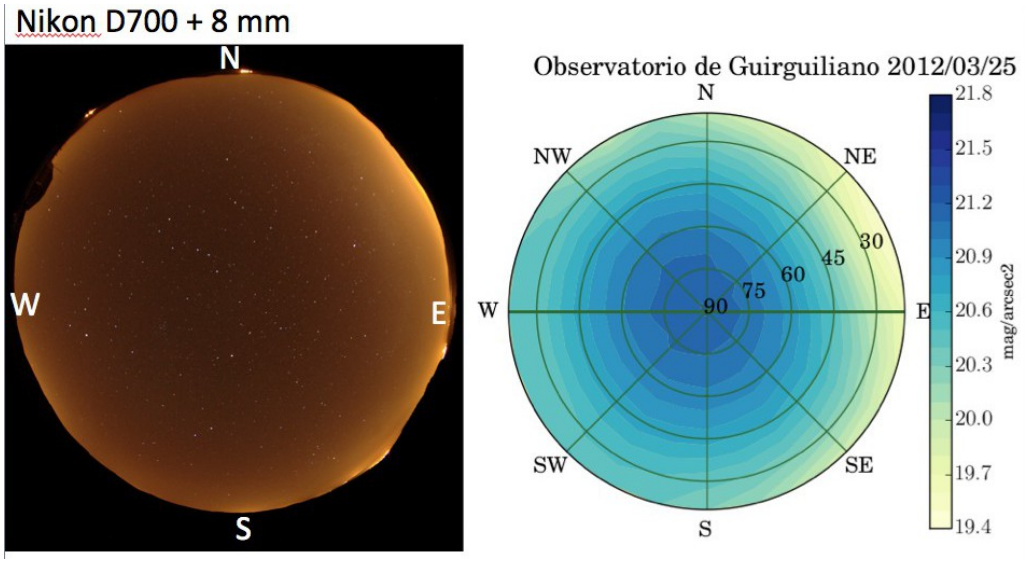

Figura 22: Fotografía de ojo de pez (izda) y representación de fondo de cielo obtenido con medidas discretas de un fotómetro SQM realizadas con el protocolo NixNox.

El calibrado de la imagen para la astrometría no fue sencillo, detectando una distorsión no despreciable en la imagen. Amablemente, Fernando Jáuregui tomó unas imágenes muy útiles para comprobar qué estaba ocurriendo utilizando como fondo un grid en el propio Planetario.

Una vez transformadas las imágenes a fits usando IRIS, se generó un mapa utilizando PyAstMon-UCM superponiendo el grid con las coordenadas horizontales sobre la propia imagen del observatorio. De aquí, se observó que efectivamente existía una ligera distorsión en la imagen entregada por la cámara, probablemente debida al hecho de que el sensor de la cámara utilizada es FullFrame, pero no suficiente para justificar las diferencias observadas en la imagen de campo.

Al final, el problema se encontraba en la hora de toma de datos. Justo esa noche se produjo el cambio de hora, y la cámara no estaba ni en UTC ni en UTC+2 (como correspondía a la hora local oficial), sino que estaba en UTC+1.

Solución astrométrica y estrellas detectadas .

Una vez visto y corregido el error en la fecha de la toma, pudimos determinar los parámetros astrométricos (esto es, desplazamiento en X,Y respecto al centro, rotación de la imagen y factor de escala) por prueba y error, llegando tras pocas iteraciones a un valor aceptable que permitiera encontrar estrellas:

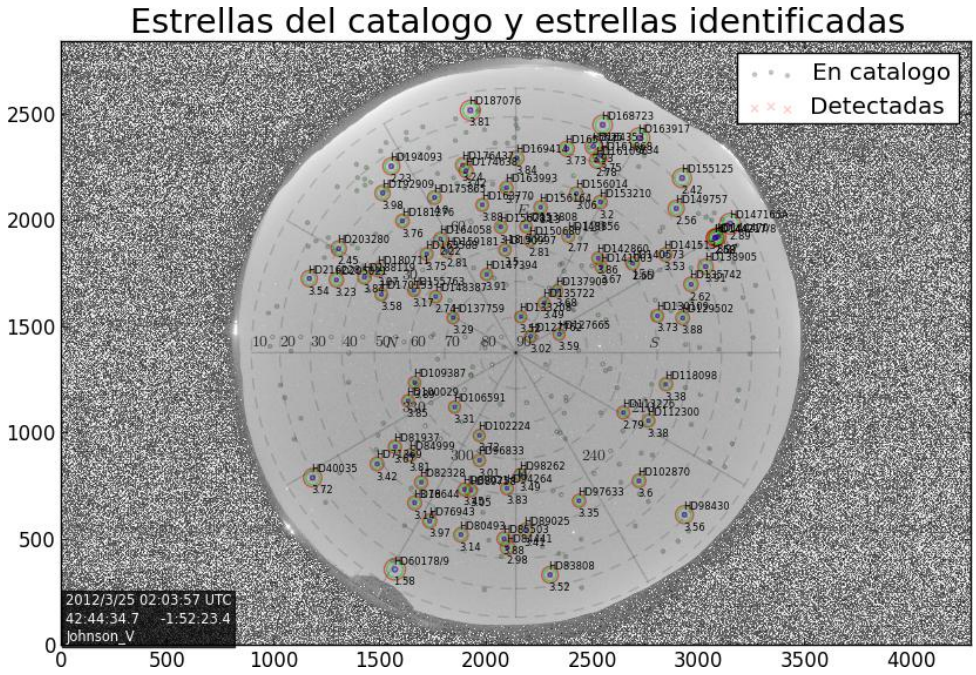

Figura 23: Astrometría resuelta con PyAstMon-UCM

Calibración fotométrica De cara a corregir los flujos medidos de efectos como viñeteo (recordemos que para la cámara utilizada no se dispone de FLATFIELDS), se barajó la posibilidad de medir con ese objetivo la respuesta siguiendo el sistema descrito por [Ina] y [Cor], pero Sigma nos ahorró el trabajo facilitándonos medidas internas de ese ojo de pez en el cual figuraba la respuesta a distintas inclinaciones. Con esto, fue posible corregir los flujos, obteniendo

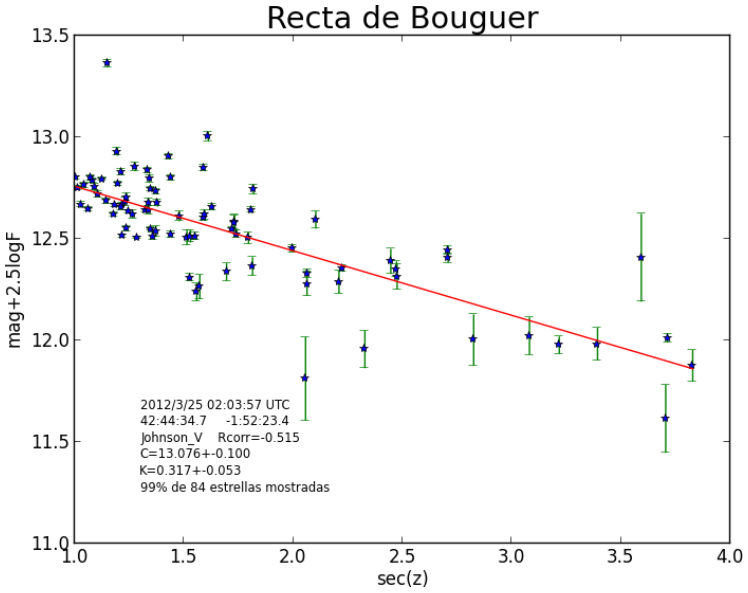

Figura 24: Calibración fotométrica de la imagen con con PyAstMon-UCM

De no haber tenido en cuenta esa corrección por respuesta del sistema, los flujos obtenidos a ángulos cenitales elevados habrían sido subestimados, obteniendo una extinción mayor de la real y medidas de brillo de fondo de cielo irreales en cuanto nos separamos del cénit.

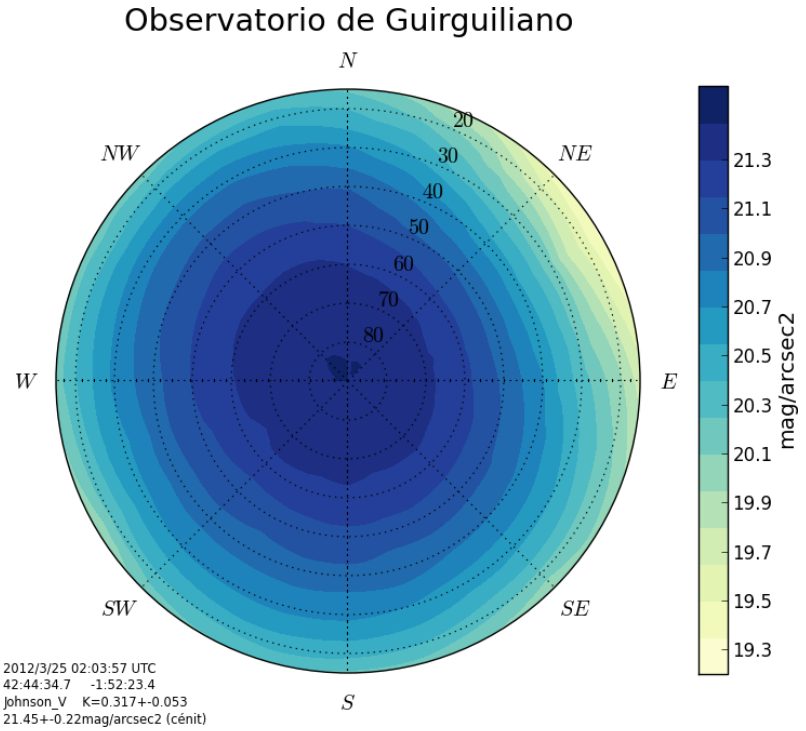

Figura 25: Mapa de fondo de cielo obtenido con PyAstMon-UCM a partir de la imagen RAW. Nótese comparando con la figura 22 que AstMon consigue una resolución espacial superior.

4.4. Otras cámaras de baja gama

4.4.1. Cámara de pequeño formato y objetivo ojo de pez

Las cámaras CCD/CMOS usadas por astrónomos aficionados como la QHY5[20] se emplean en el mundo de la astrofotografía para realizar autoguiado en monturas ecuatoriales. Son baratas por su sencillez y el tamaño de su detector. Entre otras ventajas mencionaremos:

- Permiten realizar tomas de un tiempo de exposición arbitrario.
- Se les pueden acoplar filtros, lo que permitiría comparar directamente con los datos fotométricos disponibles en catálogos.
- El tiempo de descarga de las imágenes es muy corto (al contrario de lo que ocurre con los CCD's de gran formato) y al carecer de obturador mecánico, están preparadas para aguantar un número elevado de disparos[21]. Esto hace que esta cámara sea muy útil para realizar tomas con mucha frecuencia sin miedo a que los componentes mecánicos se deterioren.

[20] En este caso, sensor CMOS MT9M001 de 8 bits, $5{,}2\mu m/pixel$ de tamaño de píxel y $1280 \times 1024 px$

[21] Están diseñadas para realizar tomas cada pocos segundos, tal y como requiere el guiado automático en astrofotografía

Sin embargo:

- La sensibilidad no es tan elevada como en una cámara CCD dedicada.
- Baja profundidad de bits, disminuyendo la precisión en la fotometría.
- Muy sensible en el IR, hace falta introducir un filtro que corte estas longitudes de onda.
- El ruido térmico y de lectura presente en estas cámaras es elevado. Sería conveniente estudiar la posibilidad de instalar un módulo Peltier para refrigerar el sensor.

4.4.2. Cámara de videovigilancia con tecnología de integración

Tradicionalmente, las cámaras de videovigilancia sólo permitían realizar tomas cortas. Hoy en día, se pueden encontrar en el mercado algunos modelos que permiten integrar exposiciones cortas para sintetizar una imagen cuya exposición efectiva puede ser de varios segundos. Esto permite capturar estrellas de cierta magnitud y podría permitir realizar la calibración fotométrica.

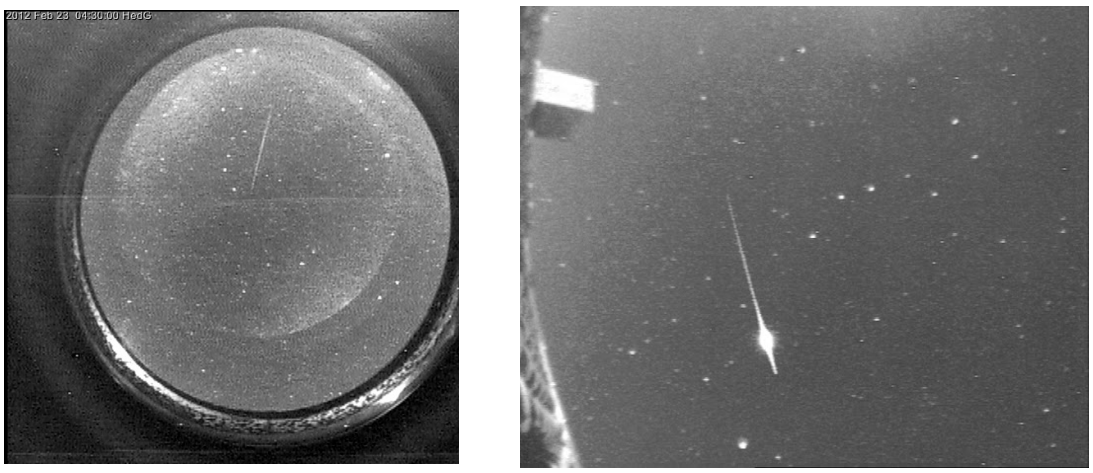

Figura 26: Cámara CCTV como AllSky. Imágenes tomadas por el prof. Jaime Zamorano.

5. Resultados experimentales

5.1. Origen de las incertidumbres en la medida de flujos y magnitudes

Un aspecto que merece la pena discutir es la razón de que se obtengan incertidumbres tan elevadas en el brillo de fondo de cielo. Es habitual obtener por ejemplo incertidumbres que se encuentran entre 0.1-0.2 magnitudes, incluso más si el ajuste no es muy bueno o tenemos poca señal de fondo (como veremos que ocurre en AstMon-OT).

La razón principal ya se podía intuir cuando comentábamos los histogramas típicos de los valores de flujo que toman píxeles al realizar estadística.

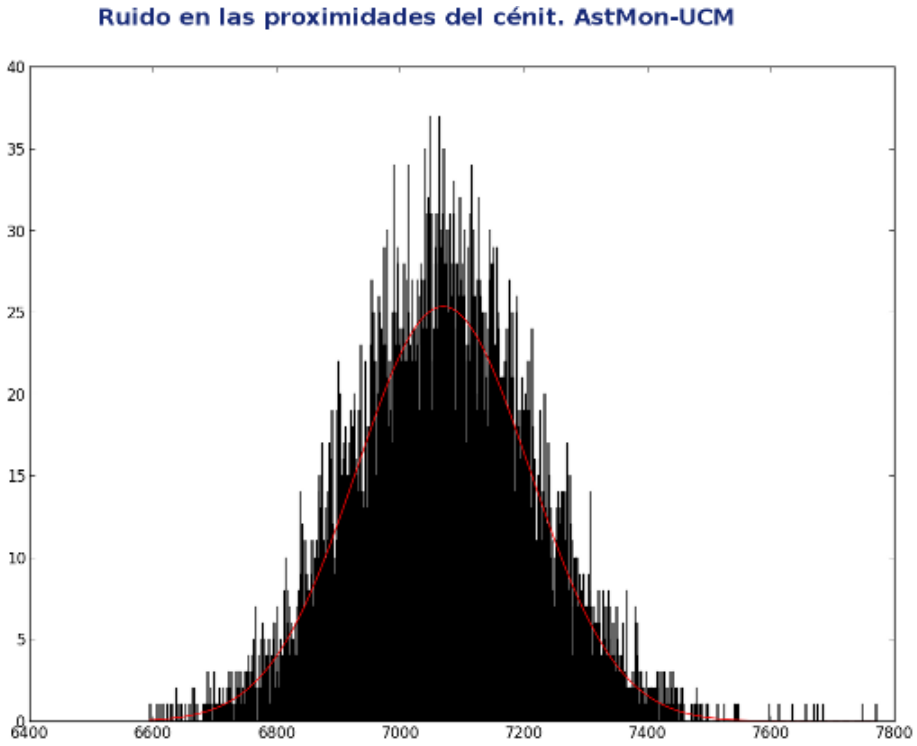

Figura 27: Ejemplo de distribución en AstMon-UCM de los valores de los píxeles cerca del cénit (círculo de medida de 55px de radio, tiempo de exposición de 40s en banda R

Observamos que, a pesar de que el CCD de AstMon-UCM se encuentra refrigerado a una temperatura aproximada de $-15°$, el ruido/granulado presente en la imagen no es en absoluto despreciable, estando fuertemente dominado por una componente de ruido de lectura. La distribución de valores se encuentra centrada en, aproximadamente, 7050 cuentas. Sin embargo, tenemos una anchura de 400 cuentas.

Esa dispersión de valores, que era aún mayor en valor relativo cuando teníamos configurado AstMon con tiempos de integración menores es la responsable, en buena medida, de que obtengamos unas incertidumbres elevadas en las medidas de flujo. Es evidente que, al tratarse de ruido de tipo fotónico (i.e. $\propto \sqrt{F}$) cuando la señal es suficientemente alta, la SNR de las medidas de flujo aumenta con $\sqrt{t_{exp}}$, ya que la señal aumenta de forma aproximadamente lineal con el tiempo de exposición, mientras que el ruido lo hace aproximadamente con $\sqrt{t_{exp}}$.

En cielos oscuros todo se complica, ya que por una parte el ruido de origen térmico deja de

ser despreciable, y por otra parte estamos sujetos a una fuerte componente de ruido de lectura.

Esto nos permite hacer una estimación de qué tiempos de exposición necesitamos para obtener una SNR determinada, aunque no debemos olvidar que tiempos excesivamente largos tienen también sus inconvenientes, como por ejemplo la deriva que presentan las estrellas (es más complicado medir su flujo sin entrar en problemas como que múltiples estrellas caigan en un mismo círculo de medida) o que buena parte de las estrellas más brillantes se estén perdiendo por saturación de píxeles.

5.2. Respuesta instrumental de AstMon-UCM

Analizando los datos obtenidos con nuestros procedimientos se observaban varios hechos:

- Durante el análisis, se observa claramente un offset en los valores de cada píxel (midiendo en las esquinas, donde no hay iluminación, se ve que el flujo no es 0). Es de esperar que en una imagen a la que hemos aplicado Darks correctamente (AstMon genera Darks en cada sesión, una vez la temperatura se estabiliza) esa corriente de oscuridad que resta en la imagen sea, en promedio, próxima a 0.
- Los valores de los píxeles son, a pesar de que la imagen se guarda en formato flotante de 32 bits, enteros. Esto da a entender que la imagen no ha sufrido ningún tipo de procesado. En particular, la corrección de FlatField implica divisiones entre números, lo que en general debería dar números decimales y no enteros como se observa.
- Los datos disponibles presentaban unos valores excesivamente altos cerca del cénit. Este hecho ya fue observado por Pablo Ramírez Moreta en su trabajo[Mor11],

  > *Como se ha comentado anteriormente, ha sido necesario introducir una corrección de color para mejorar el ajuste de estas rectas, y es que valores de estrellas cercanas al valor de masa de aire: sec(z) 1.1 tenían una dispersión demasiado elevada. El término de corrección de color introducido en el segundo miembro de la ecuación [4.3] ...*

  Sin embargo, no concretó la causa de esas observaciones, probablemente por tener una muestra de datos (medidos manualmente) bastante pobre. Al observar la forma del FlatField, este presenta ese mismo efecto de abrillantamiento cerca del cénit, lo que puede explicar perfectamente que nuestros flujos medidos sean elevados en esa región si consideramos que no han sido corregidos de FlatField.
- Las extinciones obtenidas eran bastante elevadas tanto para lo esperable en una ciudad como Madrid como lo que se medía en otros emplazamientos como Izaña (Tenerife). Además, cambiaban notablemente cuando extendíamos las medidas a valores de $\sec z$ más altos. Por último, y no menos importante, lejos de parecer una recta, los datos parecían ajustarse de forma polinómica.

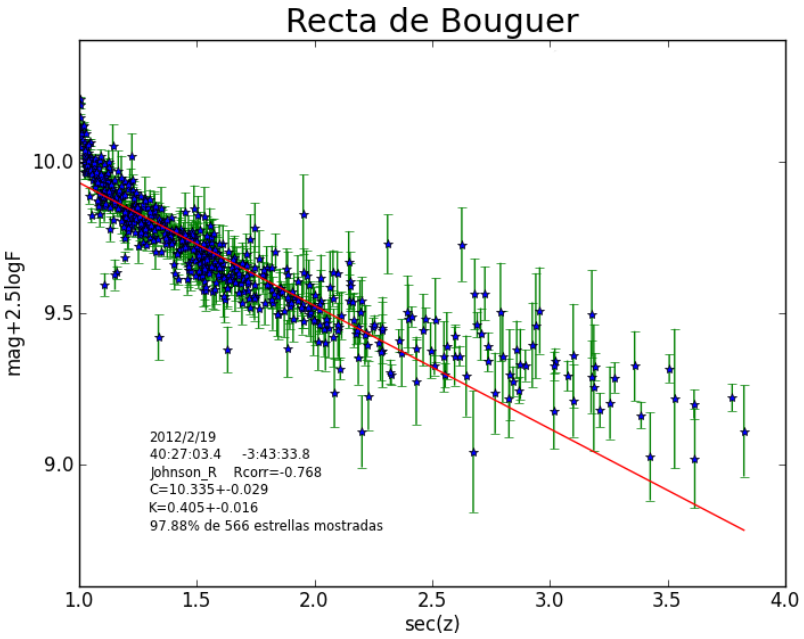

Figura 28: Ajuste a la recta de Bouguer de datos no corregidos de viñeteo/respuesta instrumental

Por desgracia, los FLATS disponibles se tomaron cuando el obturador solar estaba presente. Al retirarlo, si aplicamos esos mismos FLATS introducimos irremediablemente artefactos en ciertas partes de la imagen, con lo que esa solución no es apropiada mientras no generemos nuevos FLATS.

En vez de ello, se decidió utilizar las regiones menos afectadas de los FLATS para construir una curva de respuesta que, mediante interpolación, pudieramos aplicar a todos los flujos medidos en la imagen.

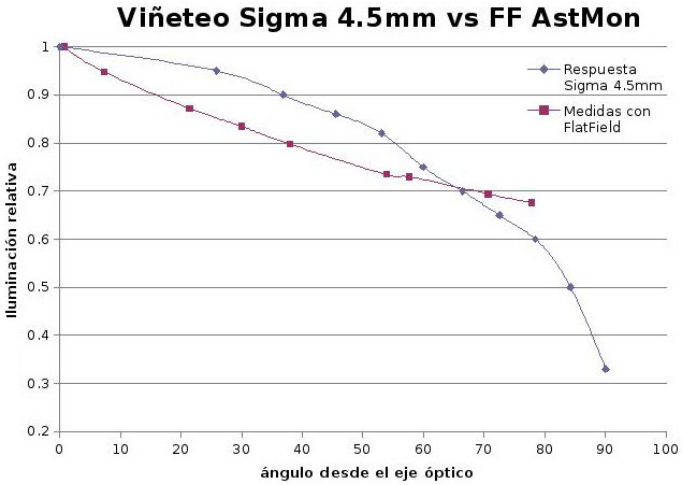

Figura 29: Curvas de respuesta teóricas del Sigma 4.5mm (tomadas de [Cor]) y medida utilizando el propio FLATFIELD

Como muestra de la influencia que tiene que no se estén aplicando los FLATFIELD en las imágenes, se midió una misma imagen tanto con como sin aplicar la corrección de respuesta instrumental, llegando a

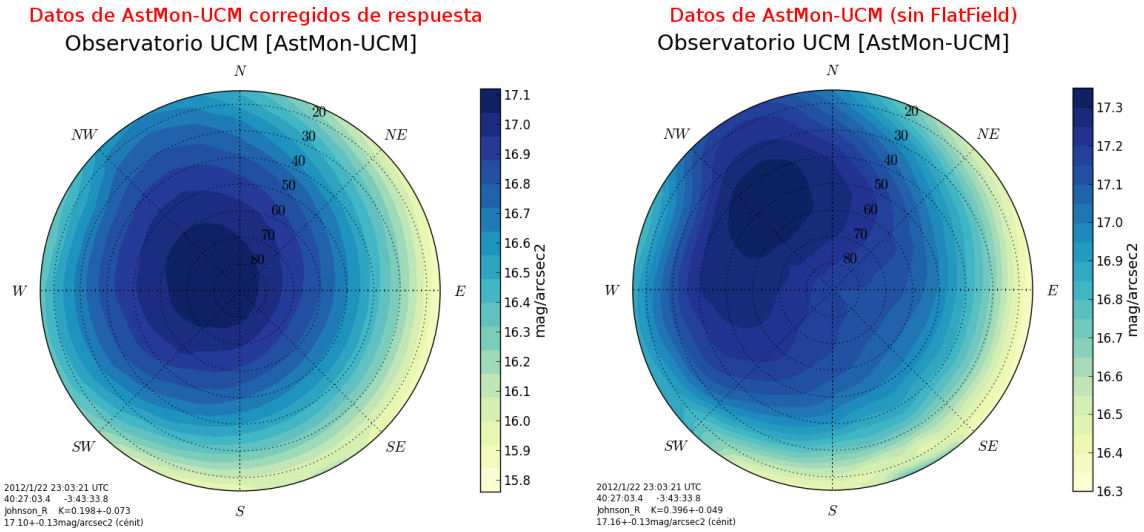

Figura 30: Comparación entre antes y después de aplicar la corrección de viñeteo/respuesta instrumental

Observemos que la región más oscura se ha desplazado hacia el cénit, coincidiendo con los resultados obtenidos con SQM.

5.3. Evolución del brillo de fondo de cielo en la Hora del Planeta 2012

AstMon toma imágenes cada pocos minutos todas las noches del año en filtros fotométricos. Esto permite no sólo estudiar cambios "a gran escala" (cambios estacionales), sino también cambios durante una determinada noche.

Esto se puso en práctica durante la Hora del Planeta[Wika], un evento promocionado por el WWF que se celebra el último sábado de cada Marzo. Muchos países se unen a este apagón simbólico de la iluminación de numerosos edificios, decoraciones, empresas y hogares para concienciar a la población sobre los peligros del cambio climático y la necesidad de tomar medidas al respecto.

En la UCM, se intentó ver si el efecto de apagado de luces que anunciaba la administración del estado y al que se supone se sumarían muchos particulares se observaba de alguna forma en las imágenes All Sky que se obtenían con AstMon.

Por desgracia, la hora del planeta cae en un momento en que la iluminación del Sol es aun muy evidente (al menos durante la primera media hora), con lo que el efecto neto en la disminución de luz no es tan apreciable. Además, en esta ocasión el cielo estaba cubierto de nubes, las cuales en principio eran más homogeneas, pero más tarde aparecieron nubes brillantes que distorsionaban las medidas.

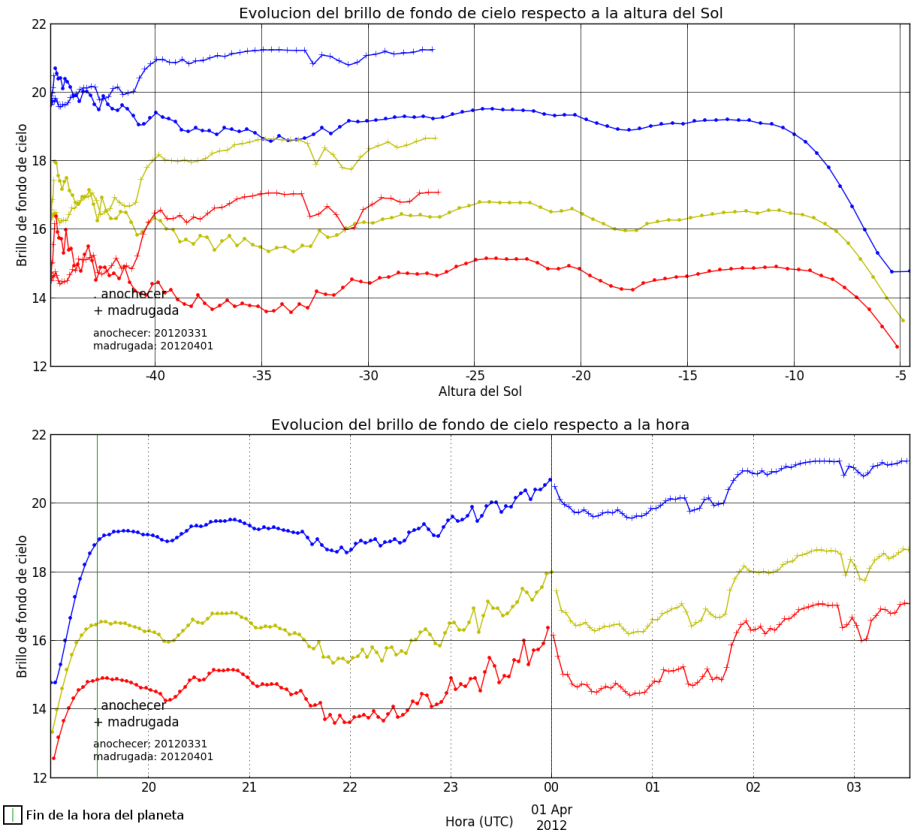

Figura 31: Medidas tomadas con AstMon durante la noche del 31 de Marzo al 1 de Abril. Marcado con una linea verde el final de la hora del planeta. Para mejorar la visualización de los datos, las medidas en V llevan un offset de +1, las de B llevan +2 de offset

Observamos que la gráfica es bastante irregular, característica de noches con nubosidad variable. Sí que se observa, quizá, un inicio más suave

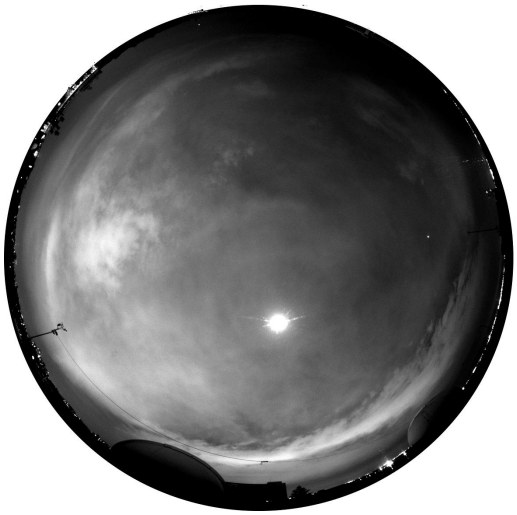

Figura 32: Imagen de las 19:40, 10 minutos después del fin de la hora del planeta. Es de esperar que la iluminación fuera encendiéndose de manera gradual, no instantánea. En la imagen de la izquierda, observamos como existe una nube bastante uniforme que ocupa casi todo el centro del cielo.

Por desgracia, pocos minutos después empezaron a surgir nubes más brillantes como la que se observa a la izquierda de la imagen. Esas nubes complicaron las medidas al provocar saltos en la medida de brillo de fondo de cielo.

5.4. Aplicación a las imágenes de AstMon-Izaña

Se están estudiando actualmente varias propuestas de lugares para albergar el proyecto CTA (Cherenkov Telescope Array, `http://www.cta-observatory.org/`), uno de los cuales es un valle situado en las inmediaciones del observatorio del Teide en Izaña (Tenerife), a 2390m sobre el nivel del mar. Con objeto de poder evaluar la calidad del emplazamiento para el proyecto y poder comparar con las condiciones presentes en otros de los lugares propuestos como Namibia o San Pedro Mártir (México), se ha instalado un AstMon controlado por el IAC.

El día 26 de Abril, el profesor José Luis Contreras consiguió acceso a las imágenes que obtenía ese AstMon. El acceso es a través de un repositorio ftp, y nuestro objetivo era poder probar nuestro código con unas imágenes cuyas condiciones de cielo son muy distintas a las disponibles en la UCM. En particular, el flujo de fondo de cielo es mucho menor (y por tanto su señal) y entran en juego otros posibles factores como el brillo propio de la Vía Lactea, que desde nuestro observatorio sencillamente no se puede observar.

Así, se descargaron varias imágenes para su posterior análisis, observando ciertas limitaciones.

- En el momento de elaborar este informe no se disponía de FLATFIELD ni medida alguna sobre la respuesta de su sistema, viéndonos obligados a utilizar las medidas de nuestro aparato. Aunque el hardware de ambos es el mismo (misma configuración de cámara-objetivo), no se puede descartar que existan pequeñas diferencias en la respuesta. Actualmente, si bien se dispone de FlatFields para cada filtro, es necesario estudiar si eliminan completamente el viñeteo del sistema.

- La señal de fondo es muy pobre. Una de las ventajas que tiene el observatorio UCM en ese sentido es la gran señal que tenemos de fondo. Con el CCD de AstMon-UCM bastan $30s$ de exposición para obtener flujos del orden de $3 - 5 \cdot 10^3$ cuentas para V y R (en el cénit), algo menos para B. Una estimación rápida de la SNR podría ser $SNR \approx \frac{S_0}{\sqrt{S_0/g}} = 39 - 50$ (V y R respectivamente, tomando $g = 0{,}5$ como es el caso de AstMon). En AstMon-IAC los márgenes respecto al bias/offset con que jugamos, al ser un cielo mucho más oscuro, no pasan de las 150 cuentas en el cénit para tiempos de 40s (de 250 cuentas de bias, obtenemos en torno a 350-400 de cielo) en V y R. La SNR en este caso es 9, con lo que ello supone en cuanto a incertidumbres en las medidas. La situación es si cabe más dramática en los filtros B y U. El cielo en estos filtros es aun más oscuro y el flujo que se mide es demasiado pequeño para tener medidas razonablemente precisas.

- A cambio de la pobre señal de fondo, el número de estrellas detectables y susceptibles de ser medidas es muy superior, pudiéndose medir de forma simultánea cientos de estrellas en los filtros de R y V, dando lugar a calibración fotométrica bastante robusta. Esto último no es aplicable al filtro U, para el cual la cúpula de metacrilato utilizada se muestra prácticamente opaca y la sensibilidad del CCD pequeña, obteniendo incluso para las estrellas más brillantes flujos muy pequeños, en el límite de detección (aun integrando 200s).

Se han seleccionado para la prueba las imágenes en filtro R de Johnson de los días 28 de Febrero de 2012 (sin Vía Láctea) y 2 de Mayo de 2012 (con Vía Láctea presente)

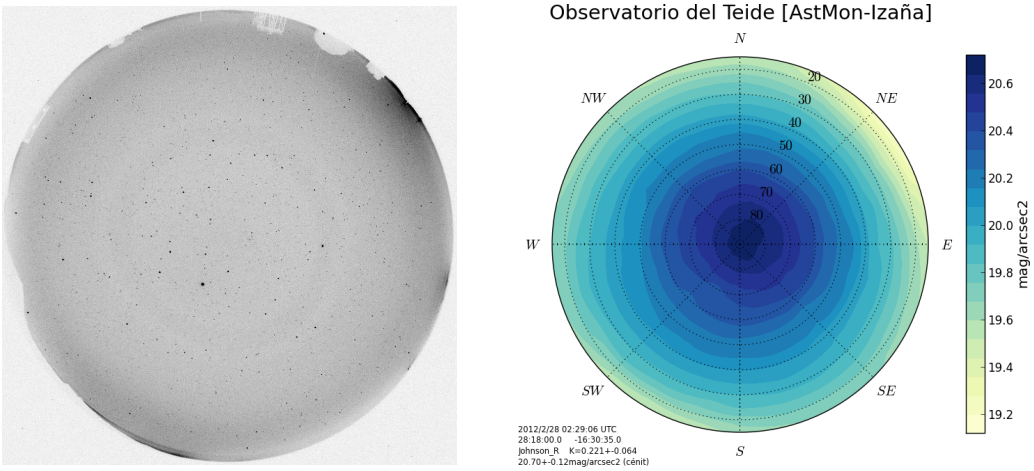

Figura 33: Imagen de AstMon-Izaña sin Vía Láctea presente en negativo. A la derecha, mapa de brillo de fondo de cielo obtenido tras el análisis con PyAstMon-UCM

Tal y como cabría esperar en un sitio razonablemente limpio como es Izaña de contaminación lumínica, el mapa de brillo de fondo de cielo tiene aproximadamente simetría radial (al menos hasta alturas medias) con pequeñas diferencias en el horizonte por la presencia de focos de contaminación (Santa Cruz de Tenerife y San Cristóbal de la Laguna).

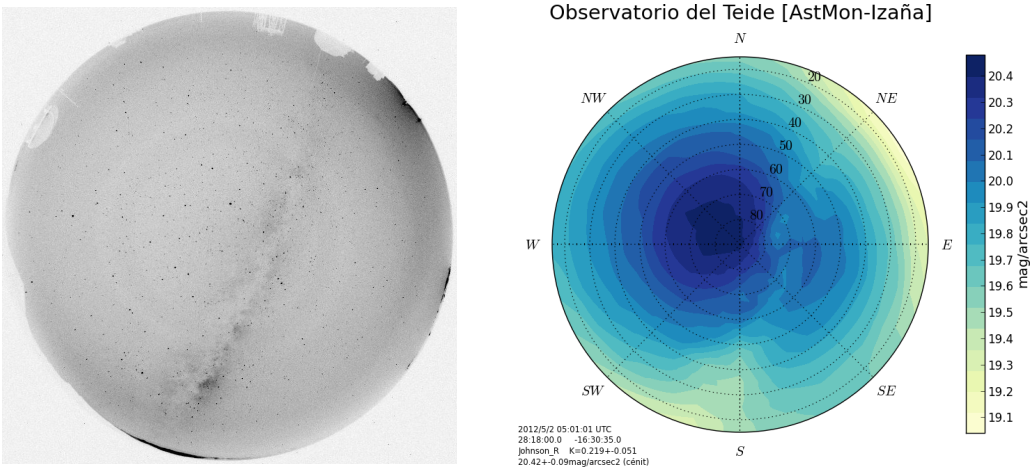

Figura 34: Imagen de AstMon-Izaña con Vía Láctea presente en negativo. A la derecha, mapa de brillo de fondo de cielo correspondiente. Nótese como se deforman las isofotas por la presencia de la Vía Láctea.

Se aprecia en la imagen de la izquierda perfectamente la Vía Láctea y, en una zona bastante más iluminada casi en el borde, el Centro de la Galaxia (dirección a Sagitario). Cuando calculamos la imagen de brillo de fondo de cielo (imagen de la derecha), se aprecia claramente la influencia de la Vía Láctea, que incluso desplaza el punto más oscuro de la imagen que en origen estaba muy próximo al cénit a una posición más alejada de éste, perdiéndose de paso la simetría radial que presentaba la imagen en ausencia de Vía Láctea.

6. Conclusiones

Con este trabajo se ha desarrollado todo un conjunto de herramientas para analizar imágenes AllSky similares a las que da AstMon, con el objetivo no sólo de poder calibrar el sistema para realizar fotometría, sino para poder hacer un estudio completo de las fuentes de iluminación que afectan al observatorio UCM en particular, siendo el análisis extensible en principio a cualquier otro observatorio con sólo ligeras variaciones.

6.1. Dificultades

En todo este proceso de construcción, nos hemos percatado de las dificultades que tiene realizar un análisis automatizado de datos "en crudo", por citar algunos ejemplos

- Cálculo preciso de posiciones de estrellas que figuran en un catálogo, ubicándolas en la imagen con no más de unos pocos píxeles de error.
- Corrección de efectos como viñeteo (diferencias radiales en la iluminación del sistema).
- Medida de flujos teniendo presente los tamaños de los círculos de medida, llegando a un compromiso (para cada estrella en particular y dependiendo de las características de la imagen) entre círculos demasiado grandes (uno se asegura de que está midiendo todo el flujo, pero por contra hay problemas adicionales en cuanto a que el riesgo de que entren varias estrellas en los círculos se multiplica) y unos círculos demasiado pequeños que pueden dejar parte de la estrella fuera y por tanto medir mal el flujo de la misma.
- Ajuste de las medidas fotométricas a una recta de Bouguer, tomando precauciones para tratar valores que se desvían de lo esperable bien por errores durante la medida, por problemas en la propia imagen o por saturación de píxeles. En este punto, cabe destacar el desarrollo realizado en base de estimadores robustos para la realización de regresiones lineales.

6.2. Procedimientos

Así, hemos llegado a una serie de procedimientos que permiten, entre otros objetivos:

- Caracterizar fotométricamente un determinado sistema AllSky a partir de la medida de flujos de estrellas estándar
- Medir de forma individualizada imágenes, así como analizar todas las imágenes disponibles de una determinada noche una a una, pudiendo generar:
  - ficheros con medidas adecuados para un posterior análisis, permitiendo realizar estadísticas con los datos.
  - gráficos con los datos en un diagrama $m_\nu + 2{,}5 \log_{10} F_\nu$ frente a *secz* (masa de aire) y el respectivo ajuste a la recta de Bouguer.
  - mapas de brillo de fondo de cielo, señalando en particular la medida en el cénit.
- Medir de forma conjunta imágenes de toda una noche, utilizando los valores fotométricos de todas las estrellas medidas de cada imagen para obtener una estadística robusta en noches fotométricas.

- Generar gráficas que muestren la evolución del brillo de cielo nocturno en función tanto de la hora y fecha como en función de la altura del Sol en ese momento.

Todo esto prácticamente sin intervención del usuario. A pesar de ello, el código se ha desarrollado de forma abierta para invitar a su posterior modificación y adaptación según los requisitos particulares de cada sistema. Y es esta la principal virtud del programa propuesto, el poder controlar en todo momento qué se hace y cómo se hace.

6.3. Resultados

Una vez comenzamos a analizar datos con nuestro sistema, se han podido obtener como resultados:

- Una comparación directa entre las condiciones presentes en un observatorio muy contaminado lumínicamente como es el observatorio UCM (un observatorio urbano) y las presentes en un lugar mucho más oscuro, como es el observatorio del Teide (Izaña, Tenerife) o el observatorio de Guirguiliano (Navarra). En particular, se ha presentado la posibilidad de analizar el efecto que la Vía Láctea sobre el brillo de fondo de cielo de un lugar oscuro como Izaña.
- La evolución del brillo nocturno en función de la altura solar, que muestra claras diferencias entre el periodo tras la puesta de Sol (el cielo se oscurece paulatinamente hasta pasada la culminación inferior) y la madrugada, relativamente plana hasta que empieza a observarse la luz matinal. Este hecho podría ser susceptible de un análisis futuro, puesto que uno esperaría que (salvo por efectos debidos a aerosoles) las diferencias presentes entre anochecer y madrugada fueran menores.
- Comparación con medidas de otros sistemas cuya forma de operar es diferente, tales como fotómetros o dispositivos como el SQM. Esto ha permitido valorar la fiabilidad y precisión de AstMon como sistema para caracterizar astronómicamente el cielo de un determinado lugar.

7. Trabajo pendiente

7.1. Mejoras en el código desarrollado

El trabajo que hemos realizado no acaba aquí. Se han propuesto en este sentido una serie de mejoras adicionales sobre lo ya existente (básicamente, depuración y simplificación de código) además de introducir nuevos aspectos como pueden ser:

- Análisis y solución astrométrica automática/semiautomática de la imagen, bien mediante el desarrollo de un algorítmo propio o bien mediante la implementación de algún sistema ya hecho como `Astrometry.net`. Por desgracia, el algorítmo actual depende de una solución astrométrica introducida ad-hoc por el usuario, siendo éste un punto débil para analizar imágenes de dispositivos móviles (requiere tiempo obtener una solución aceptable).
- Aumentar los grados de libertad posibles en la solución astrométrica, de cara a poder analizar imágenes con una mayor distorsión o imágenes cuyo eje difiera mucho del cénit[22].
- Contemplar la posibilidad de utilizar formas geométricas diferentes al círculo para las medidas de flujo, siendo este un punto interesante para analizar imágenes de tiempos de exposición muy largos, en las que las estrellas aparecen muy elongadas en ángulo horario.
- Estudiar la posibilidad de generar, desde el propio programa de análisis, mapas de color (B-V, R-V, etc).
- Permitir realizar estimación de términos de color desde el propio software.
- Desarrollo de una interfaz más cómoda para el manejo del software.

7.2. Desarrollo de un sistema de captura

Otro punto que merece mención a parte por su importancia es la construcción del software de captura para poder prescindir del software de AstMon por completo y tener un control total sobre el proceso, la reducción de imágenes con FLATS y DARKS, los tiempos de exposición que se utilizan, etc. Esto nos permitiría además

- Poder utilizar un sistema operativo más adecuado para estar funcionando continuamente, por ejemplo basado en Linux. Actualmente, una serie de factores (como son los drivers del dispositivo AnyWhereUSB, o incluso el propio software de AstMon) provocan bloqueos e interrupciones en el proceso de medida con bastante frecuencia. Esto hace que AstMon no sea un sistema con suficiente fiabilidad para ser utilizado en aplicaciones remotas, un objetivo muy interesante para estos aparatos AllSky[23].
- Aplicar el mismo sistema de captura y procesado no sólo a AstMon, sino a una amplia variedad de dispositivos que difieran en hardware. Para ello, será necesario diseñar una

[22] En estas, la distorsión del objetivo difiere mucho de la que contempla la proyección ZEA "canónica", a partir del cénit, de forma que el análisis mediante nuestro procedimiento pierde eficacia

[23] Sin ir más lejos, estos dispositivos se suelen instalar en lugares en los que se están evaluando las condiciones de cielo presentes para la posterior construcción de un observatorio Astronómico. Habitualmente son emplazamientos remotos, sin infraestructuras que permitan al investigador estar pendiente del sistema AllSky en todo momento. Tener en estos casos un sistema fiable y completamente autónomo es crítico para que sea un instrumento de utilidad.

interfaz de control común para todos los dispositivos, y hacer uso a continuación de distintos módulos que realicen la captura dependiendo de las características propias de cada sistema.

- Generar cabeceras en las imágenes FITS con una mayor cantidad de información que la que actualmente genera AstMon, así como la solución astrométrica en forma WCS.
- Almacenar, junto a las imágenes de cada noche, las imágenes de Dark para permitir que se puedan reducir a mano las imágenes si así se desea. Actualmente, AstMon toma estos Darks al inicio de la sesión y los aplica internamente, pero ni los almacena en el disco duro ni los aplica a las imágenes Raw que almacena.
- Almacenar las imágenes por noches en vez de por días. Puesto que el uso que se va a dar a los datos es fundamentalmente astronómico, no tiene sentido que las imágenes tomadas el día 21 de enero a las 23:50 estén en un directorio distinto a los tomados el día 22 de enero a las 00:05 por ejemplo.

7.3. Imágenes de FlatField

Centrándonos en AstMon-UCM, se han de tomar con relativa urgencia nuevas imágenes de FlatField, ya que las que se están utilizando actualmente no son satisfactorias por varias razones:

- Se han eliminado algunos elementos del sistema desde que AstMon-UCM fue instalado que tenían un efecto notable en la propia imagen de Flat, como por ejemplo el obturador solar. Esto introduce una sobre-corrección en las imágenes no deseable.
- Se tomaron hace ya mucho tiempo (años), no siendo descartable que la opacidad de la cúpula de metacrilato haya cambiado en este tiempo, de modo que las correcciones no sean las más adecuadas.

Por supuesto, en cuanto a resultados se refiere, con este trabajo sólo hemos explotado muy por encima las posibilidades que ofrece éste instrumento. A muchos de los objetivos que se marcó en su día para un posterior análisis Pablo Ramírez Moreta[Mor11], podemos añadir como objetivos interesantes:

- Estudio de la variación a largo plazo de las constantes instrumentales del sistema, así como la influencia que tiene la acumulación de polvo en la cúpula de metacrilato.
- Estadística de noches fotométricas en el Observatorio Astronómico UCM.
- Estudio de la evolución del brillo de fondo de cielo a largo plazo en el Observatorio Astronómico UCM.
- Modelización de la evolución del brillo de cielo a lo largo de una noche típica y estudio de componentes como son el apagado de fuentes de contaminación lumínica de la ciudad o el cambio en la concentración de aerosoles.
- Comparación entre las observaciones desde Tierra y observaciones espaciales (imágenes de la península tomadas desde la ISS).
- Diseño de medidores de brillo de fondo de cielo baratos.

7.4. Trabajo de campo/laboratorio

Dentro de este apartado, podemos destacar

- Calibración de cámaras digitales para su uso como monitores astronómicos. En particular, se podría investigar el funcionamiento de sistemas programables como el conocido *Magic Lantern*, `http://magiclantern.wikia.com/wiki/Unified` para las cámaras réflex digitales Canon, que eventualmente podría permitirnos obtener algunos resultados desde la propia cámara sin tener que recurrir a un computador.
- Selección de objetivos que sean atractivos para nuestro estudio, en particular que se ajusten a las proyecciones estándares (hemos visto por ejemplo que los Sigma de 4.5mm y de 8mm se ajustan razonablemente bien a la proyección ZEA)
- Desarrollo de sistemas ópticos para el fotómetro SSP3
- Pruebas de campo

A. Apéndice

A.1. Software desarrollado

A.1.1. Astrometría y ficheros de entrada

Código A.1: Función que carga las coordenadas del observatorio-UCM

```
def coordenadas_observatorio(Imagen):
    UCM = ephem.Observer()
    UCM.pressure = 0 # Que no tenga en cuenta efectos atmosfericos por ahora
    UCM.elevation = 700
    UCM.lat = Imagen.lat*pi/180# '40.450941' # Tomados de la pagina de cielosdemadrid
    UCM.lon = Imagen.lon*pi/180#'-3.726065'
    return UCM
```

Código A.2: Función que carga la imagen de AstMon

```
def lectura_fits_calibracion(fichero_imagen,UCM,Imagen):
    fichero_fits=pyfits.open(fichero_imagen) # Cargamos el fichero
    cabecera_fits=fichero_fits[0].header
    matriz_datos_fits = fichero_fits[0].data
    filtro = cabecera_fits['FILTER']
    if filtro in ['Jonhson_U','JohnsonU','Johnson_U','JonhsonU']:
        Imagen.filtro='Johnson_U'; cabecera_fits['FILTER'] = Imagen.filtro
    elif filtro in ['Jonhson_B','JohnsonB','Johnson_B','JonhsonB']:
        Imagen.filtro='Johnson_B'; cabecera_fits['FILTER'] = Imagen.filtro
    elif filtro in ['Jonhson_V','JohnsonV','Johnson_V','JonhsonV']:
        Imagen.filtro='Johnson_V'; cabecera_fits['FILTER'] = Imagen.filtro
    elif filtro in ['Jonhson_R','JohnsonR','Johnson_R','JonhsonR']:
        Imagen.filtro='Johnson_R'; cabecera_fits['FILTER'] = Imagen.filtro
    elif filtro in ['Jonhson_I','JohnsonI','Johnson_I','JonhsonI']:
        Imagen.filtro='Johnson_I'; cabecera_fits['FILTER'] = Imagen.filtro

    fecha_fits=cabecera_fits['DATE']
    anyo=fecha_fits[0:4]; mes=fecha_fits[4:6]; dia=fecha_fits[6:8]
    hora=fecha_fits[9:11]; minuto=fecha_fits[11:13]; segundo=fecha_fits[13:15]
    UCM.date=str(anyo)+"/"+str(mes)+"/"+str(dia)+" "+str(hora)+":"+str(minuto)+":"+str(segundo)
    Imagen.resolucion_xy=[int(cabecera_fits['NAXIS1']),int(cabecera_fits['NAXIS2'])]
    Imagen.texp=float(cabecera_fits['EXPOSURE']);
    if Imagen.texp==0.0: Imagen.texp=1.0
    Imagen.fecha=[anyo,mes,dia,hora,minuto,segundo]

    if Imagen.darkframe != False: # Calibracion de la imagen si hay darks y/o flats disponibles
        MasterDark = pyfits.open(Imagen.darkframe)[0].data
        print 'Aplicando MasterDark'
        matriz_datos_fits = matriz_datos_fits - MasterDark
    if Imagen.flatfield[Imagen.filtro]!=False:
        MasterFlat = pyfits.open(Imagen.flatfield[Imagen.filtro])[0].data
        print 'Normalizando y aplicando FlatField'
        MasterFlat = MasterFlat / np.median(MasterFlat)
        matriz_datos_fits = matriz_datos_fits/MasterFlat
    # Offset artificial de 1e-10 para evitar 0s
    matriz_datos_fits=matriz_datos_fits+1e-10*np.ones((len(matriz_datos_fits),len(matriz_datos_fits[0])))
    return cabecera_fits,matriz_datos_fits,UCM,Imagen
```

Código A.3: Ctes de transformación de coordenadas horizontales a XY

```
def horiz2xy(az,alt,Imagen):
    Imagen=ctes_xy_imagen(Imagen)
    # Corregimos refraccion
    alt = refraccion_atmosferica(alt,modo="directo")
    # Devuelvo las coordenadas en pixeles
    Rtheta=(180.0/pi)*sqrt(2*(1-sin(alt*pi/180.0)))
    x_ = -Imagen.cte_R*Rtheta*cos(az*pi/180.0-Imagen.cte_AZ*pi/180.0)
    y_ = Imagen.cte_R*Rtheta*sin(az*pi/180.0-Imagen.cte_AZ*pi/180.0)
    x = x_+Imagen.centro_pixel_x+Imagen.despl_x_ptos
    y = y_+Imagen.centro_pixel_y+Imagen.despl_y_ptos
    return x,y
```

Código A.4: Lectura catálogo Ducati

```
def leer_catalogo_tsv(nombre_catalogo):
    # Presupongo que existe un catalogo llamado: catalogo_estelar_astmon.tsv
    # Es un catalogo johnson 2002 en formato tab separated values
    # obtenido de: http://vizier.u-strasbg.fr

    try: nombre_catalogo
    except: nombre_catalogo='catalogo_estelar_astmon.tsv'
    catalogo_tabseparated=open(nombre_catalogo, 'r').readlines()

    # Quiero eliminar saltos de linea
    catalogo_tabseparated=[linea[:-1] for linea in catalogo_tabseparated]

    # Tenemos que dividir cada una de las lineas para fabricar la tabla
    print 'Leyendo catalogo ...'
    catalogo_matriz=([])
    linea_comienzo=32-1 #
    for linea in range(len(catalogo_tabseparated)):
        if linea>linea_comienzo:
            catalogo_matriz.append(catalogo_tabseparated[linea].split('\t'))

    numero_lineas=(linea-linea_comienzo-1)
    return catalogo_matriz, numero_lineas
```

Código A.5: Cargar catálogo en matriz de medidas estelares

```
def cargar_catalogo_matriz(catalogo_matriz,numero_lineas,UCM,altura_minima=0):
    # Cargamos los datos en una matriz con las coordenadas horizontales de pyephem
    estrellas=([])
    print 'Cargando estrellas en pyephem...'
    for i in range(numero_lineas):
        # El nombre de la estrella, es un vector -> quiero una cadena
        nombre_estrella_ = catalogo_matriz[i][1].split()
        nombre_estrella = ""
        for k in range(len(nombre_estrella_)):
            nombre_estrella=nombre_estrella+nombre_estrella_[k]

        # Coordenadas J1950, ojito -> misma jugada, pasar a formato pyephem
        asc_recta_ = catalogo_matriz[i][2].split()
        asc_recta = ""
        for k in range(len(asc_recta_)):
            if asc_recta=="":
                asc_recta=asc_recta+str(int(asc_recta_[k]))
            else:
                asc_recta=asc_recta+":"+asc_recta_[k]

        dec_ = catalogo_matriz[i][3].split()
        declinacion = ""
        for k in range(len(dec_)):
            if declinacion=="":
                declinacion=declinacion+str(int(dec_[k]))
            else:
                declinacion=declinacion+":"+dec_[k]

        # Magnitud V
        mov_propio_ar = '0'; mov_propio_dec='0';
        try: catalogo_matriz[i][4]
        except: magnitud_V = "20"
        else: magnitud_V = str(catalogo_matriz[i][4])

        # Mag. B. Nos dan B-V, de modo que
        try: float(catalogo_matriz[i][6])
        except: magnitud_B=str(float(magnitud_V))
        else: magnitud_B=str(float(magnitud_V)+float(catalogo_matriz[i][6]))

        # Mag. R. Nos dan R-V, de modo que
        try: float(catalogo_matriz[i][7])
        except: magnitud_R=str(float(magnitud_V))
        else: magnitud_R=str(float(magnitud_V)+float(catalogo_matriz[i][7]))

        # Declaramos la estrella en pyephem
        estrella_ephem_ = '"'+nombre_estrella+",f|S|A0,"+asc_recta+"|"+mov_propio_ar+\
            ","+declinacion+"|"+mov_propio_dec+","+magnitud_V+',1950,0"'
        estrella_ = ephem.readdb(estrella_ephem_)
        estrella_.compute(UCM)

        # Solo usaremos las que sean visibles en el cielo (altura >0)
        if estrella_.alt >altura_minima*pi/180:
            valores_procesados=nombre_estrella+","+magnitud_V+","+magnitud_B+','+magnitud_R+","
            valores_procesados+=str(estrella_.ra)+","+str(estrella_.dec)+","+str(estrella_.az)+","
            valores_procesados+=str(estrella_.alt)
            estrellas.append(valores_procesados)
    print "Lectura del catalogo finalizada. "+str(len(estrellas))+" estrellas cargadas."
    return estrellas
```

A.1.2. Búsqueda de estrellas

Código A.6: Análisis de la imagen y búsqueda de estrellas [parte1]

```
def tabla_fotometria(datos_fits,estrellas,UCM,Imagen,fichero_mapa=0):
    # Guardo los datos_fits sin tocar
    datos_fits_original=deepcopy(datos_fits)

    # Mensaje de debug
    print 'Buscando estrellas en la imagen ....'

    # Defino la tabla donde se guardaran todos los datos fotometricos
    tabla_fotometria=[]
    tabla_brillo_fondo=[]

    # Para el fondo de cielo habra que desechar algunos pixeles donde hay estrellas
    grid_borrar=[]; valor_fondo=[]

    num_estrellas=min([len(estrellas),numero_max_estrellas,Imagen.numero_max_estrellas])
    num_encontradas=num_estrellas

    if fichero_mapa!=0:
        figuramapa = figure(figsize=(10,10),dpi=100)
        imagenmapa = figuramapa.add_subplot(111)

    x_=[]; y_=[]; x_ok=[]; y_ok=[];
```

Código A.7: Análisis de la imagen y búsqueda de estrellas [parte2]

```
for estrella in range(num_estrellas):
    # Cargamos parametros de la estrella
    Estrella = parametros_estrella(estrellas[estrella],Imagen)
    # Disco de medida para estrella+fondo
    R1,R2,R3 = radio_discos_medida(Estrella.magnitud,texp=Imagen.texp,
        secZ=Estrella.secZ,resolucion=min(Imagen.resolucion_xy),
        radio_base=Imagen.radio_base)
    #R1=int(R1*1.5); R2=int(R2*1.5); R3=int(R3*1.5)
    pixeles_1,pixeles_2,pixeles_3=disco_medida(Estrella.X,Estrella.Y,R1,R2,R3,Imagen)
    if pixeles_1 == [] or pixeles_2 == [] or pixeles_3 == []:
        continue

    Estrella.cuentas_estrella,Estrella.cuentas_fondo = medir_cuentas_aprox(datos_fits,
            pixeles_1,pixeles_2,pixeles_3,texp=Imagen.texp,profbits=Imagen.bits_ccd)
```

Código A.8: Tamaño de los discos de medida

```
def radio_discos_medida(magnitud,texp,secZ,resolucion, radio_base):
    # Tamanyo base y correcciones
    R1 = radio_base; R2=R1*3; R3=R1*4
    factor_mag = 10**(-0.4*magnitud)
    factor_resol = 0.5*(resolucion/2500)
    factor_texp = 0.5*(texp/40) # a mayor tiempo, mayor elongacion de las estrellas
    factor_secz = 0.7*(secZ) # Aplicacion en camaras DSLR de alta resolucion

    R3 = int(R3*(1+factor_mag+factor_resol+factor_texp+factor_secz))
    R2 = int(R2*(1+factor_mag+factor_resol+factor_texp+factor_secz))
    R1 = int(R1*(1+factor_mag+factor_resol+factor_texp+factor_secz))
    return R1,R2,R3
```

Código A.9: Medida aproximada de flujos

```
def medir_cuentas_aprox(datos_fits,pixeles_1,pixeles_2,pixeles_3,texp,profbits):
    cuentas_fondo=medir_cuentas_fondo(pixeles_3,datos_fits,metodo='mediana')[0]
    cuentas_total,num_pixeles=medir_cuentas_estrella_mas_fondo(pixeles_1,datos_fits,profbits)
    cuentas_estrella = cuentas_total - num_pixeles*cuentas_fondo
    return cuentas_estrella/texp, cuentas_fondo/texp
```

Código A.10: Análisis de la imagen y búsqueda de estrellas [parte3]

```
# Pixeles cuyo valor no se debe tener en cuenta para el calculo de fondo
for pixel in pixeles_1+pixeles_2:
    grid_borrar.append(pixel)
    valor_fondo.append(Estrella.cuentas_fondo*Imagen.texp)

# Se utilizara para representar en el mapa las estrellas en catalogo
x_.append(Estrella.X); y_.append(Estrella.Y)

# No lo pongo antes para que me pinte los puntos y descarte del grid de fondo
if Estrella.magnitud>magnitud_limite(filtro=Imagen.filtro,secZ=Estrella.secZ,
        texp=Imagen.texp,magnitud_max=Imagen.magnitud_max) or Estrella.altura<Imagen.altura_min:
    #print 'Descartada por baja altura o mag alta'
    num_encontradas-=1
    continue

# Corregir respuesta del sistema
Estrella.cuentas_estrella = corregir_respuesta(Estrella.cuentas_estrella,pixeles_1,Imagen)
Estrella.cuentas_fondo = corregir_respuesta(Estrella.cuentas_fondo,pixeles_1,Imagen)

# Las condiciones de deteccion dependen de varios factores
min_deteccion=condiciones_deteccion(filtro=Imagen.filtro,
        valor_fondo=Estrella.cuentas_fondo,secZ=Estrella.secZ,
        base=Imagen.lim_deteccion_base,texp=Imagen.texp)

# Criterios de deteccion. Si el flujo es muy peq o muy grande, descartar (ruido, satura)
if Estrella.cuentas_estrella<min_deteccion:
    #print 'Estrella eliminada por pocas cuentas'
    num_encontradas-=1 # Esta estrella no la he encontrado
else:
```

Código A.11: Magnitudes límite para cada filtro

```
def magnitud_limite(filtro,secZ,texp,magnitud_max):
    # Se facilita la deteccion en filtros B (menor flujo), a alturas bajas y se corrige de texp
    try:
        base = magnitud_max
        extra_filtro = 0.25*int(filtro=='Johnson_B') -0.25*int(filtro=='Johnson_R')
        extra_altura = 3*log10(secZ) # Son flujos, uso log para pasar a mag
        extra_texp = 3*log10(texp/10 +1)
    except: return -5
    else:
        return base + extra_filtro+extra_altura+extra_texp
```

Código A.12: Corrección de viñeteo/respuesta del sistema

```
def corregir_respuesta(cuentas,pixeles,Imagen):
    try:
        angulo_interp=Imagen.angulo_interp
        caidas_interp=Imagen.caidas_interp
    except: pass
    else:
        if Imagen.flatfield[Imagen.filtro]==False:
            factor_interp=interpolate.interp1d(angulo_interp,caidas_interp,kind='linear')
            factor_cuentas=0; num_usados=0
            for num_pixel in range(len(pixeles)):
                # Interpolamos para varios de los pixeles y promediamos.
                # Ojo, al sumar offset puede dar error.
                if len(pixeles)<9 or num_pixel%(int(len(pixeles)/9)+1)==0:
                    pixel = pixeles[num_pixel]
                    try: angulo_eje = 90.-xy2horiz(pixel[0]+Imagen.despl_x_ptos,pixel[1]-\
                        Imagen.despl_y_ptos,Imagen)[1]
                    except: print 'Aviso: calculando correccion_respuesta de '+str(pixel)+\
                        ' me he salido de la imagen.'
                    else:
                        factor_cuentas += np.interp(angulo_eje,angulo_interp,caidas_interp);
                        num_usados+=1
            if num_usados>0:
                factor_medio = factor_cuentas*1./num_usados
                cuentas = cuentas/factor_medio
    return cuentas
```

Código A.13: Condiciones de detección de la estrella

```
def condiciones_deteccion(filtro,valor_fondo,secZ,texp,base):
    # Somos mas permisivos para alturas bajas
    factor_alturas = 1/secZ
    min_deteccion_estrella=base*valor_fondo*factor_alturas*sqrt(texp) # ~ noise -> sqrt
    return min_deteccion_estrella
```

Código A.14: Análisis de la imagen y búsqueda de estrellas [parte4]

```
else:
    # Calculamos el centroide de la estrella,
    # La estimacion fina solo se hace para estrellas que van a medirse
    pixeles_centroide=disco_medida(Estrella.X,Estrella.Y,int(R2),int(R2),int(R2),Imagen)[0]
    try: Estrella.X,Estrella.Y=calcular_centroide(pixeles_centroide,datos_fits,Estrella.X,
            Estrella.Y,Estrella.cuentas_fondo*Imagen.texp,min_deteccion)
    except:
        #print 'Descartada por no encontrar centroide'
        num_encontradas-=1
        continue
    else:
        # Si no se ha podido determinar un centroide, eliminamos la estrella.
        if [Estrella.X,Estrella.Y]==[False,False]:
            #print 'Descartada por no encontrar centroide'
            num_encontradas-=1
            continue
```

Código A.15: Cálculo del centroide de una estrella

```
def calcular_centroide(pixeles_estrella,datos_fits,X,Y,cuentas_fondo,min_deteccion):
    suma_lin=0; suma=0.; suma_pesx=0.; suma_pesy=0.
    #print cuentas_fondo
    for pixel in pixeles_estrella:
        # Ponerlo lineal + fondo por mediana es problematico
        cuentas_px = (datos_fits[pixel[1]][pixel[0]] - cuentas_fondo)**2
        suma_pesx+=cuentas_px *pixel[0]
        suma_pesy+=cuentas_px *pixel[1]
        suma+=cuentas_px
        suma_lin+=datos_fits[pixel[1]][pixel[0]] - cuentas_fondo
    # X,Y del centroide
    xcent=suma_pesx*1./suma; ycent=suma_pesy*1./suma
    #print ycent,xcent
    if datos_fits[ycent][xcent]-cuentas_fondo > (suma_lin+min_deteccion)/len(pixeles_estrella):
        # Debug
        #print 'Centroide OK'
        return xcent,ycent
    else:
        return False,False
```

Código A.16: Análisis de la imagen y búsqueda de estrellas [parte5]

```
    pixeles_1,pixeles_2,pixeles_3=disco_medida(Estrella.X,Estrella.Y,R1,R2,R3,Imagen)
    Estrella.cuentas_estrella,Estrella.inc_cuentas_estrella=medir_cuentas_preciso(datos_fits,
        pixeles_1,pixeles_2,pixeles_3,R1,R2,Imagen=Imagen)
    if Estrella.cuentas_estrella == False:
        #print 'Descartada por fallo en medida de cuentas estrella'
        num_encontradas-=1
        continue
    # Pixeles cuyo valor no se debe tener en cuenta para el calculo de fondo
    # Ahi hay estrellas
    for pixel in pixeles_1:
        grid_borrar.append(pixel)
        valor_fondo.append(Estrella.cuentas_fondo*Imagen.texp)

    # Corregir respuesta del sistema
    Estrella.cuentas_estrella = corregir_respuesta(Estrella.cuentas_estrella,pixeles_1,Imagen)
    Estrella.inc_cuentas_estrella = corregir_respuesta(Estrella.inc_cuentas_estrella,pixeles_1,Imagen)

    # Calculamos algunos parametros necesarios para la recta de Bouguer
    Estrella=parametros_bouguer(Imagen,Estrella)

    x_ok.append(Estrella.X); y_ok.append(Estrella.Y)
    # Construyo la tabla
    tabla_fotometria.append([Estrella.numcatalogo,Estrella.magnitud,Estrella.cuentas_estrella,
                Estrella.Z,Estrella.secZ,Estrella._25logF,Estrella.M_25logF,Estrella.INC_M_25logF,
                Estrella.indice_color,Estrella.azimut])
```

Código A.17: Análisis de la imagen y búsqueda de estrellas. Medida precisa de flujos

```
def medir_cuentas_preciso(datos_fits,pixeles_1,pixeles_2,pixeles_3,R1,R2,Imagen):
    cuentas_fondo=medir_cuentas_fondo(pixeles_3,datos_fits,metodo='gaussiana')[0]
    # Si no se ha medido bien el fondo, hay que descartar la estrella
    if cuentas_fondo==False or cuentas_fondo<=0: return False,False
    inc_cuentas_fondo=sqrt(cuentas_fondo/Imagen.ganancia+(Imagen.ruido_lectura/Imagen.ganancia)**2 +\
        (Imagen.ruido_termico*Imagen.texp/Imagen.ganancia)**2)/sqrt(len(pixeles_3))
    cuentas_total,num_pixeles,R1=ajustar_apertura(Estrella.X,Estrella.Y,
        cuentas_fondo,datos_fits,radio_min=int(R1),radio_max=R2,Imagen=Imagen)
    # Si se alcanzo el radio maximo al ajustar apertura, hay que descartar la estrella.
    if cuentas_total==False or cuentas_total<0: return False,False
    inc_cuentas_total=sqrt(cuentas_total/Imagen.ganancia+num_pixeles*(Imagen.ruido_lectura/Imagen.ganancia)**2 +\
        num_pixeles*(Imagen.ruido_termico*Imagen.texp/Imagen.ganancia)**2)
    #Sustituimos valores de estrella medida por fondo
    if cuentas_total - num_pixeles*cuentas_fondo <= 0:
        return False,False
    cuentas_estrella = cuentas_total - num_pixeles*cuentas_fondo
    inc_cuentas_estrella = sqrt(inc_cuentas_total**2 + num_pixeles*(inc_cuentas_fondo)**2)
    # Queremos que sea por segundo de exposicion
    return cuentas_estrella/Imagen.texp,inc_cuentas_estrella/Imagen.texp
```

Código A.18: Medida de parámetros fotométricos para ajustar la recta de Bouguer

```
def parametros_bouguer(Imagen,Estrella):
    # 2.5logF
    Estrella._25logF=2.5*log10(Estrella.cuentas_estrella)
    Estrella.INC_25logF=(2.5/log(10))*Estrella.inc_cuentas_estrella/Estrella.cuentas_estrella
    # M+2.5logF
    Estrella.M_25logF=Estrella.magnitud+Estrella._25logF
    Estrella.INC_M_25logF=Estrella.INC_25logF
    if Imagen.filtro=='Johnson_B' or Imagen.filtro=='Johnson_R':
        Estrella.M_25logF,Estrella.INC_M_25logF=corregir_termino_color(Imagen.filtro,
            Estrella.indice_color,Estrella.M_25logF,Estrella.INC_M_25logF,Imagen)

    return Estrella
```

Código A.19: Análisis de la imagen y búsqueda de estrellas [parte6]

```
        # Pintar circulos de medida en el mapa y los letreros
        if fichero_mapa!=0:
            imagenmapa.add_patch(Circle((Estrella.X,Estrella.Y),R1,facecolor='none',edgecolor=(0,0,0.8),
                        linewidth=1, fill=False, alpha=0.5,label='_nolegend_'))
            imagenmapa.add_patch(Circle((Estrella.X,Estrella.Y),R2,facecolor='none',edgecolor=(0,0.8,0),
                        linewidth=1, fill=False, alpha=0.5,label='_nolegend_'))
            imagenmapa.add_patch(Circle((Estrella.X,Estrella.Y),R3,facecolor='none',edgecolor=(0.8,0,0),
                        linewidth=1, fill=False, alpha=0.5,label='_nolegend_'))
            imagenmapa.annotate(Estrella.numcatalogo,xy=(Estrella.X,Estrella.Y), xycoords='data',
                    xytext=(0, 3), textcoords='offset points',fontsize=6)
            imagenmapa.annotate(Estrella.magnitud,xy=(Estrella.X,Estrella.Y), xycoords='data',
                    xytext=(0,-10), textcoords='offset points',fontsize=6)

print "Estrellas en catalogo: "+str(num_estrellas)+". Identificadas y medidas: "+str(num_encontradas)

print "Hay "+str(len(grid_borrar))+" pixeles cuyo valor sustituire. Posiciones de estrellas"
for num_pixel in range(len(grid_borrar)):
    sustituir_valor_pixel([grid_borrar[num_pixel]],datos_fits,valor_fondo[num_pixel])

# Represento las estrellas del catalogo Ducati junto a las estrellas detectadas por el programa
if fichero_mapa!=0:
    imagenmapa.set_title('Estrellas del catalogo y estrellas identificadas',size="xx-large")
    imagenmapa.imshow(datos_fits_original,norm=LogNorm(),cmap=cm.gray)
    imagenmapa.scatter(x_[1:],y_[1:],marker='.',c='g',alpha=0.2,label='ptos_catalogo')
    imagenmapa.scatter(x_ok[1:],y_ok[1:],marker='x',c='r',alpha=0.2,label='ptos_detectados')
    imagenmapa.axis([0,Imagen.resolucion_xy[0],0,Imagen.resolucion_xy[1]])
    informacion_imagen=str(UCM.date)+" UTC\n"+str(UCM.lat)+5*" "+str(UCM.lon)+"\n"+Imagen.filtro
    imagenmapa.text(0.005,0.005,informacion_imagen,fontsize='x-small',color='white',
        transform = imagenmapa.transAxes,backgroundcolor=(0,0,0,0.75))
    imagenmapa = representar_ejespolares(imagenmapa,Imagen)
    imagenmapa.legend(('En catalogo','Detectadas'),'upper right')
    if fichero_mapa=="por_pantalla":
        show(figuramapa)
    else:
        fecha=str(str(UCM.date).split(" ")[0]).split("/")
        savefig(fichero_output_filtro(fichero_mapa,fecha=Imagen.fecha,filtro=Imagen.filtro))
    close(figuramapa);
return tabla_fotometria,datos_fits
```

A.1.3. Calibración fotométrica

Código A.20: Ajuste a la recta de Bouguer [parte1]

```
def valores_a_ajustar_estrellas(tabla_fotometria):
    y_valores=[]; x_valores=[]; Dy_valores=[];
    for estrella in range(len(tabla_fotometria)):
        y_valores.append(tabla_fotometria[estrella][6])
        x_valores.append(tabla_fotometria[estrella][4])
        Dy_valores.append(tabla_fotometria[estrella][7])
    return x_valores,y_valores,Dy_valores
```

```
def recta_bouguer(tabla_fotometria,Imagen,UCM,fichero_tabla=0,fichero_bouguer=0,mostrar_etiquetas=0,
            fixed_y=False,fixed_err_y=False):
    # Tenemos la tabla con los datos fotometricos.
    # Esta funcion genera la recta de Bouguer con los parametros

    # mlambda0+2.5log10F=Clambda-Klambda*secz-termino_color
    # y = a + b*x
    num_puntos=50

    # Creamos la clase donde se almacenan resultados y parametros de la regresion
    class Regresion:
        pass

    Regresion.x,Regresion.y,Regresion.Dy = valores_a_ajustar_estrellas(tabla_fotometria)
    Regresion.fixed_y=fixed_y; Regresion.fixed_err_y=fixed_err_y

    Regresion,tabla_fotometria = theil_sen(Regresion,tabla_fotometria)

    # Constante instrumental
    Regresion.cte_instrumental = [Regresion.y_int, Regresion.delta_y_int]
    Regresion.coef_extincion = [-Regresion.slope, Regresion.delta_slope]

    # Si no se han encontrado estrellas, terminamos
    if Regresion.Nrel == 0:
        return Regresion,tabla_fotometria

    # Volvemos a leer los valores que quedan una vez aplicado el sigma clipping
    x_valores,y_valores,Dy_valores = valores_a_ajustar_estrellas(tabla_fotometria)

    # Para pintar los valores de la recta ajustada
    x_fit=linspace(min(x_valores),max(x_valores),num_puntos)
    y_fit=polyval([Regresion.slope,Regresion.y_int],x_fit)

    # Porcentaje de medidas en uso
    print 'Se utilizara un: '+str("%.3f" % (100*Regresion.Nrel))+'% de las medidas.'
    print "Constante instrumental: %.3f +- %.3f" % (Regresion.cte_instrumental[0],Regresion.cte_instrumental[1])
    print "Coeficiente de extincion: %.3f +- %.3f" % (Regresion.coef_extincion[0],Regresion.coef_extincion[1])
    print "Coef. correlacion Tau: %.3f" % Regresion.Kendall_tau
```

Código A.21: Cálculo de brillo de fondo de cielo [parte1]

```
def brillo_fondo(UCM,datos_fits,Regresion,Imagen,fichero_fondo=0,fichero_fondo_tabla=0,horaengrande=0):
    if Imagen.darkframe == False:
        # Restamos offset de la imagen, se toma como referencia las esquinas (zonas no iluminadas)
        pixeles_grid_bias=grid_bias(Imagen)
        bias_medido=[medir_cuentas_fondo(pixeles_grid_bias,datos_fits,metodo='gaussiana')[0],0]
        # Hay que tener en cuenta la propia senyal de bias y la ganancia para la incert.
        bias_medido[1]=sqrt(bias_medido[0]/Imagen.ganancia + (Imagen.ruido_lectura/Imagen.ganancia)**2 +\
            (Imagen.ruido_termico*Imagen.texp/Imagen.ganancia)**2)/sqrt(len(pixeles_grid_bias))
        print "Bias medido: "+str("%.2f" % bias_medido[0])+"+-"+str("%.2f" % bias_medido[1])+" en "+\
            str(len(pixeles_grid_bias))+" pixeles."
    else:
        print 'No mido senyal de bias, ya se resto un DarkFrame'
        bias_medido = [1e-8,0]

    # Brillo en la imagen. Para ver posible vinyeteo.
    if str(fichero_fondo)[0:2]=="-1":
        plot_flujos(datos_fits,Imagen,bias_medido[0])
        return 0,0

    # Brillo en el cenit. Se mide siempre.
    R = int(min(Imagen.resolucion_xy)/50+5)
    x_cenit,y_cenit = xycenit(Imagen)
    pixeles_cenit = disco_medida(x_cenit,y_cenit,R,R,R,Imagen)[0]
    pixeles_cenit = [pixeles_cenit[px_] for px_ in range(len(pixeles_cenit)) if px_%(int(R/10)+1)==0]
    media_cenit = medir_cuentas_fondo(pixeles_cenit,datos_fits,metodo='mediana')[0]
    # Corregir de respuesta
    media_cenit = corregir_respuesta(media_cenit,pixeles_cenit,Imagen)
    # Restamos el bias y calculamos incertidumbre
    cuentas_cenit = media_cenit - bias_medido[0]
    err_media_cenit = sqrt(cuentas_cenit/Imagen.ganancia + (Imagen.ruido_lectura/Imagen.ganancia)**2 + \
        + (Imagen.ruido_termico*Imagen.texp/Imagen.ganancia)**2)/sqrt(len(pixeles_cenit))
    err_media_cenit = sqrt((corregir_respuesta(err_media_cenit,pixeles_cenit,Imagen))**2 + bias_medido[1]**2)

    try: brillo_cenit = Regresion.cte_instrumental[0] - 2.5*log10(cuentas_cenit/(Imagen.texp*Imagen.area_pixel))
    except:
        brillo_cenit=10; err_brillo_cenit=10;
    else:
        err_brillo_cenit = sqrt(Regresion.cte_instrumental[1]**2 + (2.5*err_media_cenit/(log(10)*cuentas_cenit))**2)

    print "Brillo de fondo de cielo en el cenit: "+str("%.2f" % float(brillo_cenit))+\
        "+-"+str("%.2f" % float(err_brillo_cenit))+"mag/arcsec2"
```

A.1.4. Mapas de brillo de fondo de cielo

Código A.22: Cálculo de brillo de fondo de cielo [parte2]

```
if fichero_fondo!=0 or fichero_fondo_tabla!=0:
    print 'Midiendo brillo de fondo de cielo en la imagen'

    # Genero el grid de fondo
    pixeles_grid_fondo = grid_fondo(Imagen)

    # Medimos en los pixeles elegidos
    R = int(min(Imagen.resolucion_xy)/50+5) # Radio de integracion

    azimut_punto = []
    altura_punto = []
    radio_punto = []
    flujo_punto = []
    magnitud_punto = []

    for pixel in pixeles_grid_fondo:
        # pixel -> [pixelX,pixelY,azimut,altura]
        # Obtenemos todos los pixeles donde se promediara
        pixeles_fondo=disco_medida(pixel[0],pixel[1],R,R,R,Imagen)[0]
        pixeles_fondo = [pixeles_fondo[px_] for px_ in range(len(pixeles_fondo)) if px_%(int(R/10)+1)==0]
        # Medimos el flujo
        media_fondo = medir_cuentas_fondo(pixeles_fondo,datos_fits,metodo='mediana')[0]
        media_fondo = corregir_respuesta(media_fondo,pixeles_fondo,Imagen)
        # Restamos el bias y calculamos errores
        flujo_fondo = media_fondo - bias_medido[0]
        #err_flujo_fondo = sqrt(media_fondo + bias_medido**2 + err_media_fondo)
        if flujo_fondo<0:
            print "Aviso, bias mas calculado o pixel con valor anormalmente bajo " + str(flujo_fondo)
        else:
            brillofondo = Regresion.cte_instrumental[0]-2.5*log10(flujo_fondo/(Imagen.texp*Imagen.area_pixel))
            altura_punto.append(pixel[3]*pi/180)
            azimut_punto.append(pixel[2]*pi/180)
            radio_punto.append(pi/2-pixel[3]*pi/180)
            flujo_punto.append(flujo_fondo)
            magnitud_punto.append(brillofondo)

    if fichero_fondo_tabla!=0:
        # Exportar los datos a una tabla
        fondo_tabla(fichero=fichero_fondo_tabla,Imagen=Imagen,\
            x=altura_punto,y=azimut_punto,z=magnitud_punto,xname='alt',yname='az')

    if fichero_fondo!=0:
        # Grid equiespaciado donde se interpolara.
        # ri es la colatitud. Mejor para parametrizar el radio en la representacion
        ri = np.linspace(0*pi/180,75*pi/180,76)
        azimuti = np.linspace(0,360*pi/180,361)

        # Interpolacion lineal
        flujoi = griddata((azimut_punto,radio_punto),flujo_punto,(azimuti[None,:],ri[:,None]),
            method='linear',fill_value=min(flujo_punto))

        # Filtro mediante transformadas de fourier
        #flujoi = filtro_fourier(matriz_datos=flujoi,ruido_max=0.25)
        magnitudi = Regresion.cte_instrumental[0]-2.5*np.log10(flujoi/(Imagen.texp*Imagen.area_pixel))
```

Código A.23: Cálculo de brillo de fondo de cielo [parte3]

```
        # Valores max y min para la grafica.
        barra_marcas,lista_contornos,mostrar_marcas = marcas_color_contornos(matriz_valores=magnitudi,Imagen=Imagen)

        print "Generando mapa de brillo de fondo de cielo ..."
        # Grafico de contornos
        figurafondocielo = figure(figsize=(8,8),dpi=100)
        brillofondocielo = figurafondocielo.add_subplot(111,projection='polar')
        titulofondocielo = brillofondocielo.text(0, pi/2, unicode(Imagen.titulo_mapafondo,'utf-8'),
            horizontalalignment='center',size='xx-large')
        contornosfondocielo = brillofondocielo.contourf(azimuti,ri,magnitudi,cmap=cm.YlGnBu,levels=lista_contornos)
        #puntosfondocielo = brillofondocielo.scatter(azimut_punto,radio_punto,marker='.',alpha=0.5,vmin=15)

        # Radios y circunferencias de altura. Grid
        radial_locator = [ (num+1)*pi/18 for num in range(7) ]
        radial_label = ["$80$","$70$","$60$","$50$","$40$","$30$","$20$","$10$","$0$"]
        theta_locator = [ 45*num for num in range(8)]
        theta_label = ["$N$","$NE$","$E$","$SE$","$S$","$SW$","$W$","$NW$"]
        brillofondocielo.set_rgrids(radial_locator,radial_label,size="large")
        brillofondocielo.set_thetagrids(theta_locator,theta_label,size="large")
        brillofondocielo.set_theta_direction(-1)
        brillofondocielo.set_theta_offset(pi/2)
        brillofondocielo.grid(linewidth=0.8)
        # Espaciamos la barra de color un poco
        subplots_adjust(right=1)
        barracolor = colorbar(contornosfondocielo,orientation='vertical',shrink=0.85)
        subplots_adjust(right=0.80)
        # Terminamos de ajustar lo que queda de la grafica
        barracolor.set_ticks(barra_marcas,update_ticks=mostrar_marcas)
        barracolor.set_label("mag/arcsec2",rotation="vertical",size="large")
        informacion_imagen=str(UCM.date)+" UTC\n"+str(UCM.lat)+5*" "+str(UCM.lon)+"\n"+Imagen.filtro+4*" "+\
            "K="+str("%.3f" % float(Regresion.coef_extincion[0]))+"+-"+str("%.3f" % float(Regresion.coef_extincion[1]))+\
            "\n"+str("%.2f" % float(brillo_cenit))+"+-"+str("%.2f" % float(err_brillo_cenit))+"mag/arcsec2 (cénit)"

        brillofondocielo.text(5*pi/4,125*pi/180,unicode(informacion_imagen,'utf-8'),fontsize='x-small')

        if horaengrande==1:
            brillofondocielo.text(7*pi/4,108*pi/180,unicode(str(UCM.date)+" UTC\n",'utf-8'),fontsize='large')
            brillofondocielo.text(0.5*pi/4,85.3*pi/180,Imagen.filtro,fontsize='large')

        if fichero_fondo == "por_pantalla":
            show(figurafondocielo)
        else:
            savefig(fichero_output_filtro(fichero_fondo,fecha=Imagen.fecha,filtro=Imagen.filtro))
            close(figurafondocielo);

return brillo_cenit,err_brillo_cenit
```

A.1.5. Parámetros de entrada

Código A.24: Lectura del fichero de configuración [parte1]

```
def lectura_configuracion(fichero):
    # Abrimos el fichero en python
    contenido_fichero_config=open(fichero, 'r').readlines()
    opciones = []
    for linea in range(len(contenido_fichero_config)):
        # Quito espacios y saltos de linea
        contenido_fichero_config[linea]=contenido_fichero_config[linea].replace("\n","")
        #contenido_fichero_config[linea]=contenido_fichero_config[linea].replace(" ","")
        if contenido_fichero_config[linea]!="":
            if contenido_fichero_config[linea][0]!="#":
                opcion=contenido_fichero_config[linea].split('=')
                if len(opcion)==2:
                    opcion[0] = opcion[0].replace(" ","")
                    if '"' not in opcion[1] and "'" not in opcion[1]:
                        opcion[1] = opcion[1].replace(" ","")
                    opcion[1] = opcion[1].replace('"','')
                    opcion[1] = opcion[1].replace("'","")
                    opcion[1] = opcion[1].replace("\r","")
                    opcion[1] = opcion[1].replace("\n","")
                    opciones.append(opcion)
    return opciones
```

Código A.25: Lectura del fichero de configuración [parte2]

```
def opciones_config(fichero_configuracion,Imagen):
    filtros=["U","B","V","R","I"]
    opciones=lectura_configuracion(fichero_configuracion)
    # Algunos valores por defecto.
    Imagen.ctes_inst_almacenadas={"Johnson_"+filtro:[False,False] for filtro in filtros}
    Imagen.termino_color = {"Johnson_"+filtro:[False,False] for filtro in filtros}
    Imagen.nivel_fondo = {"Johnson_"+filtro:[False,False] for filtro in filtros}
    Imagen.flatfield = {"Johnson_"+filtro:False for filtro in filtros}
    Imagen.darkframe = False
    # Recorremos todas las opciones del fichero.
    for opcion in opciones:
        if opcion[0]=="obs_lat": Imagen.lat=float(opcion[1])
        elif opcion[0]=="obs_lon": Imagen.lon=float(opcion[1])
        elif opcion[0]=="despl_x_ptos": Imagen.despl_x_ptos=float(opcion[1])
        elif opcion[0]=="despl_y_ptos": Imagen.despl_y_ptos=float(opcion[1])
        elif opcion[0]=="cte_R": Imagen.cte_R=float(opcion[1])
        elif opcion[0]=="cte_AZ": Imagen.cte_AZ=float(opcion[1])
        elif opcion[0]=="altura_min": Imagen.altura_min=float(opcion[1])
        elif opcion[0]=="radio_base": Imagen.radio_base=float(opcion[1])
        elif opcion[0]=="lim_deteccion_base": Imagen.lim_deteccion_base=float(opcion[1])
        elif opcion[0]=="lim_Kendall_tau": Imagen.lim_Kendall_tau=float(opcion[1])
        elif opcion[0]=="ganancia": Imagen.ganancia=float(opcion[1])
        elif opcion[0]=="ruido_lectura": Imagen.ruido_lectura=float(opcion[1])
        elif opcion[0]=="ruido_termico": Imagen.ruido_termico=float(opcion[1])
        elif opcion[0]=="bits_ccd": Imagen.bits_ccd=float(opcion[1])
        elif opcion[0]=="magnitud_max": Imagen.magnitud_max = float(opcion[1])
        elif opcion[0]=="numero_max_estrellas": Imagen.numero_max_estrellas = int(opcion[1])
        elif opcion[0]=="area_pixel": Imagen.area_pixel = float(opcion[1])
        elif opcion[0]=="titulo_mapafondo": Imagen.titulo_mapafondo = str(opcion[1])
        elif opcion[0]=="angulo_interp": Imagen.angulo_interp=opcion[1].split(",")
        elif opcion[0]=="caidas_interp": Imagen.caidas_interp=opcion[1].split(",")
        elif opcion[0]=="darkframe": Imagen.darkframe=opcion[1]
        else:
            # Comprobamos si es uno de los filtros definidos
            for filtro in range(len(filtros)):
                nombrefiltro = "Johnson_"+filtros[filtro]
                if opcion[0]=="cte_inst_"+filtros[filtro]:
                    Imagen.ctes_inst_almacenadas[nombrefiltro] = [float(opcion[1].split(",")[0]),\
                        float(opcion[1].split(",")[1])]
                elif opcion[0]=="termino_color_"+filtros[filtro]:
                    Imagen.termino_color[nombrefiltro] = [float(opcion[1].split(",")[0]),\
                        float(opcion[1].split(",")[1])]
                elif opcion[0]=="nivel_fondo_"+filtros[filtro]:
                    Imagen.nivel_fondo[nombrefiltro] = [float(opcion[1].split(",")[0]),\
                        float(opcion[1].split(",")[1])]
                elif opcion[0]=="flatfield_"+filtros[filtro]:
                    Imagen.flatfield[nombrefiltro] = opcion[1]

    return Imagen
```

Código A.26: Fichero de configuración

```
# Configuracion para AstMon-UCM

# Observatorio
obs_lat = 40.450941
obs_lon = -3.726065

# Offset ptos (Astrometria)
despl_x_ptos=-23.0
despl_y_ptos=-21.0
cte_AZ =89
cte_R=14.2

# Deteccion de estrellas, valores base
numero_max_estrellas = 80
altura_min = 15
radio_base = 1.8
lim_deteccion_base = 1.1
# Ganancia estimada por Jaime Zamorano (22 Mayo), imag. del amanecer e/ADU
ganancia = 0.5
# RON e- y termico e-/s. http://www.astrosurf.com/buil/qsi/comparison.htm
ruido_lectura = 8.7
ruido_termico = 0.02
bits_ccd = 16
magnitud_max = 2.5

# Fotometria
lim_Kendall_tau=0.7
# Correccion de color mag + 2.5logF = C-Ksecz -correccion_color(ind.color)
termino_color_B = 0.0215419,0.00729124
termino_color_R = -0.34425197,0.005027
# Valores medios enero-abril 2012, medidos 26 Mayo 2012.
#cte_inst_B=9.962,0.018
#cte_inst_R=10.298,0.065
#cte_inst_V=10.366,0.048

# Fondo de cielo # [206.265*tamanyo_pixel(micras)/focal(mm)]^2
area_pixel=61265.160324
titulo_mapafondo="Observatorio UCM [AstMon-UCM]"

# Niveles filtros para mapa de brillo de fondo
nivel_fondo_R=15.0,17.5
nivel_fondo_V=15.5,18.5
nivel_fondo_B=16.0,19.5

# Medido del flatfield, no sabemos si esta aplicandose
angulo_interp = 0,7.275,21.397,29.989,37.949,53.863,57.608,70.661,77.816,90
caidas_interp = 1,0.9485,0.8722,0.8349,0.7975,0.7347,0.7296,0.6941,0.6761,0.64

# Imagenes de calibracion (no se corrige con la respuesta anterior. Si se corrige bias)
#flatfield_U = ""
#flatfield_B = ""
#flatfield_V = ""
#flatfield_R = ""
#darkframe = ""



# ---------------------------------------------
# Valores almacenados en ASTMON-UCM

#cte_inst_B=9.678644, 0.057116
#cte_inst_V=10.094794, 0.022143
#cte_inst_R=9.906132, 0.014973
```

A.1.6. Script lanzador

Código A.27: Ayuda del programa

```
def mostrar_ayuda():
    print '\n~~ PyAstMon ~~'
    print 'Requisitos: Python 2.7, Matplotlib, Pyephem, Pyfits, Scipy, Numpy'
    print '\nOpciones disponibles:'
    print '-h: muestra esta ayuda.'
    print '-i fichero_entrada: fichero raw fits a analizar.'
    print '-f [anyo,mes,dia]: fecha a analizar. Los corchetes son importantes.\n'+\
            '  puede indicarse "todo" si se desea analizar todo el PC. mes y dia opcionales'
    print '--cada-imagen: analiza una a una las imagenes. \n'+\
            '  Util para ver la evolucion a lo largo de la noche'
    print '-om fichero_salida: fichero de salida para el mapa de estrellas detectadas.\n'+\
            '  si se omite el fichero se muestra por pantalla, si se omite la opcion.\n'+\
            '  no se muestra'
    print '-ot fichero_salida: fichero de salida para la tabla de fotometria. Lo mismo que\n'+\
            '  con el mapa, se puede omitir fichero o la opcion entera'
    print '-or fichero_salida: fichero de salida para los resultados de fotometria. Lo mismo que\n'+\
            '  con el mapa, se puede omitir fichero o la opcion entera'
    # Opciones en uso
    print '-ob fichero_salida: fichero de salida para la grafica de la recta de Bouguer. Idem'
    print '-of fichero_salida: fichero de salida para el mapa de brillo de fondo de cielo. Idem'
    print '-oft fichero_salida: fichero de salida para la tabla de brillo de fondo de cielo. Idem'
    print '\nfichero_salida solo indica el nombre base del fichero, el formato y la ruta\n'+\
            '  por ejemplo, /ruta/al/mapa.png. Automaticamente se completara el nombre\n'+\
            '  "mapa" con la fecha y el filtro utilizado.\n'
    exit(0)
```

Código A.28: Errotes en parámetros de entrada

```
def parametro_incorrecto():
    print '\nERROR: Parametro incorrecto '+str(sys.argv[1])
    mostrar_ayuda()

def fecha_o_fichero_no_especificado():
    print '\nERROR: Fecha o fichero no especificad@, o especificad@ incorrectamente'
    mostrar_ayuda()

def sin_parametros():
    print '\nERROR: Necesito al menos un parametro'
    mostrar_ayuda()
```

Código A.29: Análisis de fechas

```
def fecha_entrada():
    # Vemos como ha sido introducida la fecha y decidimos que tipo de analisis se hace
    try: sys.argv[2]
    except: fecha_o_fichero_no_especificado()
    else:
        try: fecha=sys.argv[2][1:-1].split(",")
        except:
            fecha_o_fichero_no_especificado()
        else:
            if sys.argv[2]=='todo' or sys.argv[2]=="todo":
                anyos=range(2010,2012+1,1)
                meses=range(1,12+1,1)
            else:
                if len(fecha)==1:
                    anyos=[int(fecha[0])]
                    meses=range(1,12+1,1)
                elif len(fecha)==2:
                    anyos=[int(fecha[0])]
                    meses=[int(fecha[1])]
                elif len(fecha)==3:
                    anyos=[int(fecha[0])]
                    meses=[int(fecha[1])]
                    dias=[int(fecha[2])]
                else:
                    fecha_o_fichero_no_especificado()
        dias_mes={1:31, 2:28, 3:31, 4:30, 5:31, 6:30, 7:31, 8:31, 9:30, 10:31, 11:30, 12:31}
        fechas=[]
        hoy_anyomesdia=str(datetime.now()).split(' ')[0].split('-')
        for anyo in anyos:
            # Posibilidad de que el anyo se de en formato corto.
            anyo_ = anyo%2000 + 2000
            for mes in meses:
                try: dias
                except:
                    if mes==2 and (anyo_%4==0 and ((anyo_ %100!=0) or (anyo_%400==0))):
                        dias=range(1,29+1,1)
                    else:
                        dias=range(1,dias_mes[mes]+1,1)

                for dia in dias:
                    utilizar_fecha=True
                    if anyo_>int(hoy_anyomesdia[0]):
                        utilizar_fecha=False
                    elif anyo_==int(hoy_anyomesdia[0]):
                        if mes>int(hoy_anyomesdia[1]):
                            utilizar_fecha=False
                        elif mes==int(hoy_anyomesdia[1]):
                            if dia>=int(hoy_anyomesdia[2]):
                                utilizar_fecha=False
                    if utilizar_fecha==True:
                        fechas.append([anyo_,mes,dia])
        sys.argv.remove(sys.argv[2]); sys.argv.remove(sys.argv[1])
        return fechas
```

Código A.30: Fichero de entrada

```
def fichero_entrada():
    try: sys.argv[2]
    except: fichero_entrada_no_especificado()
    else:
        if opciones_entrada.get(sys.argv[2], lambda : None)():
            fichero_entrada_no_especificado()
        else:
            fichero=sys.argv[2]
            # me cargo el parametro para seguir leyendo las opciones de entrada
            sys.argv.remove(sys.argv[2]); sys.argv.remove(sys.argv[1])
            return fichero
```

Código A.31: Fichero de salida

```
def fichero_output():
    # Tres posibilidades: o no se muestra, o se muestra por pantalla, o se envia a un fichero.
    # En esta funcion consideramos las 2 ultimas.
    try: sys.argv[2]
    except:
        sys.argv.remove(sys.argv[1])
        return 'por_pantalla'
    else:
        if opciones_entrada.get(sys.argv[2], lambda : None)():
            sys.argv.remove(sys.argv[1])
            fichero_out='por_pantalla'
        else:
            fichero_out=sys.argv[2]
            sys.argv.remove(sys.argv[2]); sys.argv.remove(sys.argv[1])
        return fichero_out
```

A.1.7. Lectura del SQM-LE

Código A.32: Lectura de las medidas tomadas por el SQM_LE [parte1]

```
import os,sys,time,signal
from datetime import datetime
from time import sleep

def buscar_sqm():
    try: sys.argv[1]=='-n'
    except:
        print 'Buscando SQM ...'
        def signal_handler(signal, frame):
            print '\nCtrl-C received from keyboard - script exiting'
            sys.exit(0)
        signal.signal(signal.SIGINT, signal_handler)
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.setblocking(False)
        if hasattr(socket,'SO_BROADCAST'):
            s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
        #s.setsockopt(socket.SOL_SOCKET, IN.SO_BINDTODEVICE, "eth1"+'\0')
        s.sendto("000000f6".decode("hex"), ("255.255.255.255", 30718))
        buf=''
        starttime = time.time()
        print "Looking for replies; press Ctrl-C to stop."
        while True:
            try:
                (buf, addr) = s.recvfrom(30)
                if buf[3].encode("hex")=="f7":
                    print "Received from %s: MAC: %s" % (addr, buf[24:30].encode("hex"))
            except:
                #Timeout in seconds. Allow all devices time to respond
                if time.time()-starttime > 3:
                    break
                pass
        return addr
    else:
        print 'No busco el SQM. Se usa ip por defecto'
        return 0

def formato_texto(contenido):
    texto=""
    for linea in contenido:
        for columna in linea:
            texto=texto+str(columna)+";"
        texto=texto[:-1]+"\n"
    return texto
```

Código A.33: Lectura de las medidas tomadas por el SQM_LE [parte2]

```
def cabecera_estandar_sqm():
    # Lectura de la cabecera del fichero, al final de este script.
    contenido_cabecera=""
    contenido_script = open(sys.argv[0], 'r').readlines()
    texto_valido=False;
    empieza_texto="'''EMPIEZACABECERA"
    termina_texto="TERMINACABECERA'''"

    for linea in contenido_script:
        # Arreglamos un poco la linea, quitando saltos, \r y demas
        linea = linea.replace("\r","")
        linea = linea.replace("\r","")

        if len(linea)>=len(termina_texto):
            if linea[0:18] == termina_texto:
                texto_valido=False
                #print 'Fin de lectura de la cabecera'

        if texto_valido==True: contenido_cabecera +=linea

        if len(linea)>=len(empieza_texto):
            if linea[0:18] == empieza_texto:
                texto_valido=True
                #print 'Comienzo de lectura de la cabecera'

```

Código A.34: Lectura de las medidas tomadas por el SQM_LE [parte3]

```
def leer_valores():
    global ftp_truncar
    global de_dia
    # Fecha actual
    fecha_y_hora = str(datetime.now()).split(" ")
    fecha_ = fecha_y_hora[0]
    hora_ = str(fecha_y_hora[1])[:-3] # Quitamos decimales que sobran
    while len(hora_.split(":"))<3: hora_=hora_+":00"

    # El fichero de datos acumulativo (por meses)
    if os.path.exists(nombre_fichero):
        #print 'Abriendo fichero de datos preexistente: '+str(nombre_fichero)
        fichero_ = open(nombre_fichero, "a+");
    else:
        print 'Creando nuevo fichero de datos: ' + str(nombre_fichero)
        fichero_ = open(nombre_fichero, 'w');
        fichero_.write(cabecera_estandar_sqm())
    # El fichero del FTP, datos del dia actual.
    if ftp_truncar == True or not (os.path.exists(nombre_fichero_ftp)):
        fichero_ftp = open(nombre_fichero_ftp, 'w');
        fichero_ftp.write(cabecera_estandar_sqm())
        ftp_truncar = False
    # Nuevo modo append, 'w' trunca el fichero al intentar leerlo.
    fichero_ftp = open(nombre_fichero_ftp, 'a+');

    if int(hora_[0:2])<9 or int(hora_[0:2])>16: # 9 y 17 respectivamente
    # Evitamos que tome datos inutiles durante el dia
        de_dia=False
        s.send('rx')
        msg = ''
        while len(msg) < 57:
            chunk = s.recv(57-len(msg))
            if chunk == '':
                print 'Error de conexion'
                raise RuntimeError, "socket connection broken"
            msg = msg + chunk

        # En el manual de SQM se lee por caracteres
        # r, 00.00m,0000594121Hz,0000000000c,0000000.000s, 057.6C
        brillo_fondo = float(msg[2:8])-offset_mag
        freq_sensor = float(msg[10:20])
        flujo_fondo = float(msg[23:33])
        temp_sensor = float(msg[49:54])

        # Escribiendo medidas
        contenido=[[fecha_+"T"+hora_,fecha_+"T"+hora_,temp_sensor,flujo_fondo,freq_sensor,brillo_fondo]]
        if int(brillo_fondo)<=0 or int(brillo_fondo)>30:
            print 'Aviso: Medida invalida o saturada ... '
        fichero_.write(formato_texto(contenido))
        fichero_ftp.write(formato_texto(contenido))
    else:
        de_dia=True
        if int(hora_[0:2])>=10 and ftp_truncar==False:
            if str(fichero_ftp.read())!=str(cabecera_estandar_sqm()):
                ftp_truncar = True
                print 'Se truncara el fichero del FTP'
        print hora_[0:5]+' UTC. Es de dia, no mido'
        #brillo_fondo='-'; temp_sensor='-'
    fichero_.close()
    fichero_ftp.close()
    return fecha_,hora_,brillo_fondo,temp_sensor
```

Código A.35: Lectura de las medidas tomadas por el SQM_LE [parte4]

```
# Intentamos buscar el SQM

try:
    ip_puerto = buscar_sqm()
    host_=ip_puerto[0]
    puerto_=10001#ip_puerto[1]
except:
    #host_='169.254.58.133';
    host_='147.96.21.177'
    puerto_=10001

# Calibracion del dispositivo (positivo)
offset_mag = 0.11


# Donde guardar el fichero con los datos
fecha_actual=str(datetime.now()).split(" ")[0][0:7]
nombre_fichero="C:/SQM/SQM_UCM_"+fecha_actual+".dat"
nombre_fichero_ftp="C:/Astmon/public_ftp/SQM_UCM.dat" # Datos de la noche actual

num=0; pausa=60;

ftp_truncar = False
de_dia = True

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host_,int(puerto_)))
except:
    print 'Conexion perdida. Espera hasta siguiente reinicio\n'+\
    'por la tarde o reinicia manualmente este PC'
    exit(1)

while 1<2:
    try:
        fecha,hora,brillo_fondo,temp_sensor=leer_valores()
    except:
        if de_dia==False:
            try:
                print 'Conexion perdida ...'
                #os.system('reinicio_programado.bat')
            except:
                pass
    else:
        num+=1
        print 'lectura '+str(num)+' finalizada.'
        print 'Fecha y Hora: '+str(fecha) + "  "+ str(hora)
        print 'Brillo de fondo: ' + str(brillo_fondo)
        print 'Temperatura sensor: '+str(temp_sensor)
    sleep(pausa)
s.close
```

Código A.36: Lectura de las medidas tomadas por el SQM_LE [parte5]

```
'''EMPIEZACABECERA
# Light Pollution Monitoring Data Format 0.1.5
# URL: http://www.darksky.org/LANstandards/clan_format_0.1.5.pdf
# Number of header lines: 35
# This data is released under the following license: ODbL 1.0 http://opendatacommons.org/licenses/odbl/summary/
# Device type: SQM-LE
# Instrument ID: SQM_LE-UCM
# Data supplier: Jaime Zamorano / Universidad Complutense de Madrid
# Location name: Tenerife - Observatorio del Teide
# Position: 40.450941, -3.726065, 650
# Local timezone: UTC
# Time Synchronization: NTP
# Moving / Stationary position: STATIONARY
# Moving / Fixed look direction: FIXED
# Number of channels: 1
# Filters per channel: HOYA CM-500
# Measurement direction per channel: 0., 0.
# Field of view: 20
# Number of fields per line: 6
# SQM serial number: 1952
# SQM firmware version: 6.7.0.1
# SQM cover offset value: -0.11
# SQM readout test ix: i,00000004,00000003,00000021,00001687
# SQM readout test rx: r, 18.73m,0000000004Hz,0000130978c,0000000.284s, 031.2C
# SQM readout test cx: c,00000019.69m, 0000300.000s, 023.2C,00000008.71m, 029.3C
# Comment: SQM installed on the roof of the Physics building
# Comment: (Universidad Complutense de Madrid)
# Comment: last cleaned: May 2012
# Comment:
# Comment: Capture program: PySQMLE
# blank line 30
# blank line 31
# blank line 32
# UTC Date & Time, Local Date & Time, Temperature, Counts, Frequency, MSAS
# YYYY-MM-DDTHH:mm:ss.fff;YYYY-MM-DDTHH:mm:ss.fff;Celsius;number;Hz;mag/arcsec^2
# END OF HEADER
TERMINACABECERA'''
```

A.1.8. Gráficas del SQM-LE

Código A.37: Gráficas con las medidas tomadas por el SQM_LE [parte1]

```
OBS = ephem.Observer()
OBS.lat = 40.450941*ephem.pi/180
OBS.lon = -3.726065*ephem.pi/180
OBS.elev = 650

# Leemos el fichero SB (fichero de texto que contiene datos para una noche)

ficheros=[];
datos=[]
filtros=['SQM']

directorio_diario_base = 'C:/SQM/graficos/'
if not os.path.exists(directorio_diario_base):
    os.makedirs(directorio_diario_base)

ficheros_entrada = ['C:/Astmon/public_ftp/SQM_'+OBS_name+'.dat']
ficheros_figura = ['C:/Astmon/public_ftp/SQM_'+OBS_name+'.png',\
    'C:/SQM/graficos/SQM_'+OBS_name+'_'+str(datetime.now()-timedelta(days=1)).split(" ")[0]+'.png']

for fichero in ficheros_entrada:
    ficheros.append(fichero)
    datos.append(open(fichero, 'r').readlines())

figura = plt.figure(figsize=(8,8))
grafica_altsol = figura.add_subplot(211)
grafica_hora = figura.add_subplot(212, sharey=grafica_altsol)

# Espaciado entre subplots
plt.subplots_adjust(hspace=0.3)

# Limitamos en eje Y el rango de magnitudes mostradas
plt.ylim([15.5,18.5])

# format the ticks (frente a alt sol)

tick_valores = range(-90,-0+5,5)
tick_marcas = np.multiply([grados for grados in tick_valores],ephem.pi/180.0)
tick_etiquetas = [str(grados) for grados in tick_valores]

grafica_altsol.set_xticks(tick_marcas)
grafica_altsol.set_xticklabels(tick_etiquetas)
grafica_altsol.grid(True,which='major',ls='-')
grafica_altsol.grid(True,which='minor')
```

Código A.38: Gráficas con las medidas tomadas por el SQM_LE [parte2]

```
# format the ticks (frente a hora)

dialocator = mdates.DayLocator()
horalocator   = mdates.HourLocator()
diaFmt     = mdates.DateFormatter('\n\n%d %b %Y')
horaFmt = mdates.DateFormatter('%H')

grafica_hora.xaxis.set_major_locator(dialocator)
grafica_hora.xaxis.set_major_formatter(diaFmt)
grafica_hora.xaxis.set_minor_locator(horalocator)
grafica_hora.xaxis.set_minor_formatter(horaFmt)

grafica_hora.format_xdata = mdates.DateFormatter('%Y-%m-%d_%H:%M:%S')
grafica_hora.grid(True,which='major',ls='-')
grafica_hora.grid(True,which='minor')

# Fin de ticks

#figura.autofmt_xdate()

colores = {"B":"b","V":"y","R":"r","SQM":"g"}

anoch_labl_dias=[]
madr_labl_dias=[]
```

Código A.39: Gráficas con las medidas tomadas por el SQM_LE [parte3]

```
for cada_fichero in range(len(datos)):
    datos[cada_fichero]=[linea[:-1] for linea in datos[cada_fichero] if linea[0]!="#"]
    # Extraemos datos a una matriz
    matriz_medidas = [datos[cada_fichero][caso].replace("\n","").split(';') \
        for caso in range(len(datos[cada_fichero]))]
    for cada_filtro in filtros:
        class anochecer:
            fechas=[]
            alt_sol=[]
            brillos=[]
            filtros=[]
            dias=[]

        class madrugada:
            fechas=[]
            alt_sol=[]
            brillos=[]
            filtros=[]
            dias=[]

        for fila in range(0,len(matriz_medidas),1):
            if len(matriz_medidas[fila])<4:
                continue
            # Filtro
            filtro = ['SQM']
            # Fecha y hora
            fecha_str,hora_str=matriz_medidas[fila][0].split("T")
            anyo = int(fecha_str.split("-")[0])
            mes = int(fecha_str.split("-")[1])
            dia = int(fecha_str.split("-")[2])
            hora = int(hora_str.split(":")[0])
            if len(hora_str.split(":"))==3:
                minuto = int(hora_str.split(":")[1])
                segundo = int(float(hora_str.split(":")[2]))
            elif len(hora_str.split(":"))==2:
                minuto = int(hora_str.split(":")[1])
                segundo = 0
            elif len(hora_str.split(":"))==1:
                minuto=0; segundo=0;

            fecha_hora = datetime(anyo,mes,dia,hora,minuto,segundo)
            OBS.date = ephem.date(fecha_hora)

            # Descartamos valores diurnos para la grafica
            margen = 5 # Num. datos 0.0 de margen
            if hora > 12 :
                if fila<margen and float(matriz_medidas[fila][5])<=0.0:
                    print 'Descartando dato, tarde'; continue
                elif float(matriz_medidas[fila][5])<=0.0 and \
                            float(matriz_medidas[fila+margen][5])<=0.0:
                    print 'Descartando dato, tarde'; continue
            if hora <=12 and fila>(margen-1):
                if fila>=len(matriz_medidas)-margen and \
                            float(matriz_medidas[fila][5])<=0.0:
                    print 'Descartando dato, manyana'; continue
                elif float(matriz_medidas[fila][5])<=0.0 and \
                            float(matriz_medidas[fila-margen][5])<=0.0:
                    print 'Descartando dato, manyana'; continue

            # Calculamos la altura del Sol
            Sol = ephem.Sun(OBS)
            altura_sol = Sol.alt
            # Fecha en formato para label
            fecha_lbl = int(fecha_str.split("-")[0]+fecha_str.split("-")[1]+\
                    fecha_str.split("-")[2])
            # Brillo fondo de cielo
            brillo_mag = float(matriz_medidas[fila][5])
```

Código A.40: Gráficas con las medidas tomadas por el SQM_LE [parte4]

```
            if hora > 12:
                anochecer.fechas.append(fecha_hora)
                anochecer.alt_sol.append(altura_sol)
                anochecer.brillos.append(brillo_mag)
                anochecer.filtros.append(filtro)
                if fecha_lbl not in anoch_labl_dias:
                    anoch_labl_dias.append(fecha_lbl)
            else:
                madrugada.fechas.append(fecha_hora)
                madrugada.alt_sol.append(altura_sol)
                madrugada.brillos.append(brillo_mag)
                madrugada.filtros.append(filtro)
                if fecha_lbl not in madr_labl_dias:
                    madr_labl_dias.append(fecha_lbl)

            if len(madrugada.fechas)>0:
                anochecer.fechas.append(madrugada.fechas[0])
                anochecer.alt_sol.append(madrugada.alt_sol[0])
                anochecer.brillos.append(madrugada.brillos[0])
                anochecer.filtros.append(madrugada.filtros[0])

        if len(anochecer.fechas)>0 and len(anochecer.brillos)>0:
            grafica_altsol.plot(anochecer.alt_sol,anochecer.brillos,color='g')
            grafica_hora.plot(anochecer.fechas,anochecer.brillos,color='g')

        if len(madrugada.fechas)>0 and len(madrugada.brillos)>0:
            grafica_altsol.plot(madrugada.alt_sol,madrugada.brillos,color='b')
            grafica_hora.plot(madrugada.fechas,madrugada.brillos,color='b')
```

Código A.41: Gráficas con las medidas tomadas por el SQM_LE [parte5]

```
grafica_altsol.set_title('Sky Brightness (SQM-'+OBS_name+')',fontsize='x-large')
grafica_altsol.set_xlabel('Solar altitude ',fontsize='large')
grafica_altsol.set_ylabel('Sky Brightness',fontsize='large')
grafica_altsol.text(0.03,0.09,'PM: '+str(anoch_labl_dias)[1:-1],color='g',\
    fontsize='small',transform = grafica_altsol.transAxes)
grafica_altsol.text(0.03,0.03,'AM: '+str(madr_labl_dias)[1:-1],color='b',\
    fontsize='small',transform = grafica_altsol.transAxes)

grafica_hora.set_title('Sky Brightness (SQM-'+OBS_name+')',fontsize='x-large')
grafica_hora.set_xlabel('Time (UTC)',fontsize='large')
grafica_hora.set_ylabel('Sky Brightness',fontsize='large')
grafica_hora.text(0.03,0.09,'PM: '+str(anoch_labl_dias)[1:-1],color='g',\
    fontsize='small',transform = grafica_hora.transAxes)
grafica_hora.text(0.03,0.03,'AM: '+str(madr_labl_dias)[1:-1],color='b',
    fontsize='small',transform = grafica_hora.transAxes)

for fichero_figura in ficheros_figura:
    #plt.show(figura)
    figura.savefig(fichero_figura, bbox_inches='tight')
```

A.1.9. Fotómetro SSP3

Código A.42: Comparación con el fotómetro SSP-3

```
UCM_=coordenadas_observatorio()
cabecera_fits,datos_fits,UCM,tiempo_exp,filtro=lectura_fits_calibracion(sys.argv[1],UCM_)
X,Y = horiz2xy(0,90)
for radio in radios:
    pixeles_1=disco_medida(X,Y,radio,500,600)[0]
    numero_pixeles_=0;cuentas_total=0
    for pixel_ in pixeles_1:
        cuentas_total+=10**(-0.4*cuentas_estrella_mas_fondo(pixeles_1,datos_fits)[0]
        numero_pixeles_+=1
    cuentas_media=cuentas_total/numero_pixeles
    mag_fondo = -2.5*log10(cuentas_media)
    print "Para R="+str(radio)+" tenemos "+str(mag_fondo)
```

A.2. Cómo obtener el código completo

El código completo se suministra, junto a todos los resultados y experimentos realizados en el CD que se adjunta. Dado que el desarrollo de dicho código está en una fase activa, posibles cambios y mejoras se irán introduciendo en el repositorio GIT del autor de estas líneas: `http://minaya.dyndns.org:443/repositorio/?p=astmon.git;a=summary` . Desde aquí se pueden descargar "snapshots" del código en su estado actual, además de aparecer ramas de proyecto según se publiquen versiones estables del programa.

Referencias

[Bui] Christian Buil. Complement : How to compute the photon transfer curve and the electronic gain ? http://www.astrosurf.com/buil/20d/20dvs10d.htm; [Online; accesed 19-Mayo-2012].

[cG02] M. R. calabretta and E. W. Greisen. Representations of celestial coordinates in fits. 07 2002. Apartado 5.1.8.

[Cor] Magali Coralie, Amaud. Vignetting effect of two identical fisheye lenses. http://goo.gl/a4CT5. [Online; accessed 19-Mayo-2012].

[ea11] Aceituno et al. An all sky transmission monitor: Astmon. *arXiv:1107.1682v1*, 2011.

[iAS11] iTec Astronómica S.L. Manual de instrucciones (astmon). 2011. Página 19.

[Ina] Mehlika Inanici. Applications of image based rendering in lighting simulation: development and evaluation of image based sky model. http://faculty.washington.edu/inanici/Publications/BS09_0264_271.pdf; [Online; accessed 19-Mayo-2012].

[Lab] Chuleta de laboratorio. http://www.gae.ucm.es/fisatom/docencia/labofis/chuleta/chuleta.pdf; [Online; accessed 08-Abril-2012].

[Mor11] Pablo Ramirez Moreta. Brillo de fondo de cielo con astmon. 2011. http://eprints.ucm.es/15000/.

[Pat03] F. Patat. A robust algorithm for sky background computation in ccd images. *arXiv:astro-ph/0301534v1*, 2003.

[QSI] QSImaging. Measuring ccd read noise. http://www.qsimaging.com/ccd_noise_measure.html.

[Wika] Wikipedia. Hora_del_planeta. http://es.wikipedia.org/wiki/La_hora_del_planeta; [Online; accessed 08-Abril-2012].

[Wikb] Wikipedia. Kendall tau rank correlation coefficient. http://en.wikipedia.org/wiki/Kendall_tau_rank_correlation_coefficient; [Online; accessed 08-Abril-2012].

[Wikc] Wikipedia. Theil–sen estimator. http://en.wikipedia.org/wiki/Theil-Sen_estimator; [Online; accessed 08-Abril-2012].