UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Reducción de datos astronómicos basada en procesamiento de imágenes para la robotización de telescopios

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Kevin Allekotte Hallberg

Director: Dr. Pablo De Cristóforis

Codirector: Dr. Mario Melita

Buenos Aires, Abril 2013

REDUCCIÓN DE DATOS ASTRONÓMICOS BASADA EN PROCESAMIENTO DE IMÁGENES PARA LA ROBOTIZACIÓN DE TELESCOPIOS

Uno de los objetivos de la Astronomía en el presente es estudiar y catalogar lo más exhaustivamente posible los cuerpos celestes que nos rodean. La adquisición de imágenes del cielo y su análisis es el trabajo diario de astrónomos que buscan descubrir fenómenos astronómicos. En el pasado se realizaban observaciones de forma manual y se analizaban artesanalmente por un experto. Hoy en día, los telescopios cuentan con monturas robotizadas y controlables electrónicamente, y la adquisición de imágenes es digital gracias a la tecnología CCD. Esto posibilita la automatización de los procesos de observación de fenómenos astronómicos.

En este trabajo presentamos un sistema novedoso para automatizar el proceso de descubrimiento de asteroides, basura espacial y otros objetos móviles. Este sistema está basado en el procesamiento de las imágenes adquiridas, extrayendo los datos relevantes (reducción astronómica) para encontrar los objetos móviles en el cielo nocturno y controlar el telescopio para el seguimiento automático de los mismos. Las ventajas de reemplazar las tareas rutinarias de un astrónomo por un sistema que no requiere de la intervención humana son múltiples, ya que posibilitan la exploración continua (aun en ambientes hostiles), logrando un mayor volumen de observación, y por lo tanto, más probabilidades de descubrir fenómenos astronómicos.

Palabras claves: Procesamiento de imágenes, reducción de datos astronómicos, robotización de telescopios, descubrimiento de objetos móviles.

IMAGE PROCESSING BASED ASTRONOMICAL DATA REDUCTION FOR ROBOTIC TELESCOPE AUTOMATION

One of the goals in modern Astronomy is to achieve an exhaustive mapping of celesital bodies of the sky. The acquisition of images of the sky and their analysis is the daily work of astronomers who search for discovery of astronomical phenomena. In the past, observations were carried out manually and analyzed by experts. Nowadays it is possible to electronically control robotic telescopes and image data acquisition is digitalized with CCD technology, enabling automated observations of astronomical phenomena.

In this work we present a novel system for autonomous discovery of asteroids, space trash and other moving objects. This system performs astronomical image data reduction based on image processing techniques to find moving objects in the night sky and control the telescope for an automated tracking. The replacement of rutinary tasks performed by astronomers using a system which does not require human intervention has multiple advantages and enables a continuous exploration (even in hostile enviornments) resulting in higher probabilities of achieving astronomical discoveries.

Keywords: Image processing, astronomical data reduction, telescope robotization, moving object discovery.

Índice general

1. INTRODUCCIÓN

Uno de los objetivos de la Astronomía en el presente es estudiar y catalogar lo más exhaustivamente posible los cuerpos celestes que nos rodean y en particular los cuerpos móviles del sistema solar con el fin de producir conocimiento básico. También es útil en tecnologías relacionadas como ser la explotación minera de asteroides y la prevención de catástrofes causadas por impactos.

La adquisición de imágenes y su análisis es el trabajo diario de astrónomos que buscan descubrir fenómenos astronómicos. En el pasado se realizaban observaciones de forma manual y se analizaban artesanalmente por un experto. Hoy en día, los telescopios cuentan con monturas robotizadas y controlables electrónicamente, y la adquisición de imágenes es digital gracias a la tecnología CCD. Esto permite en principio que se puedan controlar remotamente y posibilita un análisis digital de las observaciones, pero además hace permite el desarrollo de software que automatice el control y facilite la extracción de datos astronómicos de las imágenes. El proceso mediante el cual se extrae y describe la información que contienen estas imágenes se llama "reducción".

El objetivo de este trabajo es desarrollar un sistema autónomo que integre una cámara CCD y una montura robótica de manera de ser capaz de tomar decisiones de posicionamiento y parametrización de imágenes en función del análisis de los datos obtenidos de éstas. En particular, para hacer descubrimientos de fenómenos astronómicos de forma automática. Como resultado de este trabajo se presenta un sistema que realiza un procesamiento de imágenes para detectar objetos móviles (como ser asteroides y basura espacial) sin intervención humana. En la práctica, este sistema podría utilizarse para automatizar el proceso de observación que hace un astrónomo y reportar descubrimientos automáticamente.

Las ventajas de reemplazar las tareas rutinarias de un astrónomo son más que obvias, y además permite la exploración incesante y en lugares hostiles para la presencia humana como la Antártida. De esta forma se logra mayor volumen de observación, y por lo tanto, más probabilidades de lograr descubrimientos.

La solución planteada consiste por un lado en el control y robotización del telescopio, y por otro, el procesamiento de imágenes y detección de objetos móviles. Esta última parte involucra las siguentes tareas: reducción astrométrica y fotométrica, alineación, detección de móviles. El workflow planteado para una observación es en orden: Control, detección y seguimiento (control con los parámetros del objeto detectado).

A continuación se detalla el desarrollo organizado de la siguiente forma:

Estado del arte

Se describen desarrollos previos y se mencionan trabajos relacionados

Descripción del método propuesto

Scope
Alcance del trabajo y aplicaciones.

Framework
Entorno y pre-requisitos de ejecución.

Adquisición
Acerca del control de la cámara y la adquisición de imágenes.

Reducción
Extracción de estrellas (fuentes) y características de las imágenes.

Fotometría
Análisis de las magnitudes y flujo luminoso de las fuentes.

Alineación
Acerca del método de alineación de secuencias de imágenes.

Detección de objetos móviles
Búsqueda de fuentes que describen trayectorias a lo largo de la secuencia.

Astrometría
Conversión entre coordenadas en píxeles y coordenadas celestes.

Control de telescopio
Manejo de la montura robótica.

Detalles implementativos

Decisiones de diseño
Detalles del diseño del software.

Arquitectura del software
Estructura general del software.

Diseño de módulos/clases
Organización de los módulos y clases del software.

Resultados

Análisis de los resultados obtenidos y la efectividad del método.

2. ESTADO DEL ARTE

La gestión de la observación de los fenómenos astronómicos de interés, desde el planeamiento de las observaciones (cálculo de efemérides, de tiempos de integración, diseño de secuencias de imágenes, etc), hasta la operación de cada componente del observatorio (por ejemplo de la montura del telescopio, de la cámara CCD, del foco, de la rueda de filtros, etc), demanda una gran cantidad de tiempo material y atención para un operador que realice todas las tareas involucradas de forma manual. Por otro, lado las decisiones a tomar por el sistema están siempre bien determinadas por el estado de algún periférico. Por ejemplo, se decide al apertura de la cúpula en función de los datos arrojados por la central meteorológica, se decide corregir el foco en función del perfil de brillo de la estrella obtenido de una imagen de la cámara CCD, etc.

No todos los grandes observatorios cuentan con una soporte informático de manera que sea posible realizar observaciones de acuerdo a "scripts" programados. En algunos casos, para mencionar uno de nuestro país, como el CASLEO (2.15m) [1] la operación es manual, sólo se pueden programar secuencias de imágenes, pero el movimiento del telescopio no puede ser manejado por un script. En el caso de los instrumentos del Observatorio Astronómico Austral se cuenta con un software común (P2PP [2]) que sincroniza la operación del telescopio e instrumentos de observación. Pero la operación no es autónoma en tanto que está manejada por un operador. Un ejemplo de telescopio robótico es el telescopio Liverpool (2m) ubicado en al isla de La Palma, Canarias, España; o los telescopios Faulkes (2.5m) ubicados en Hawaii (EEUU) y en el desierto de Atacama, Chile. Pero en estos casos el sistema de gestión está adaptado a los instrumentos en particular y no es público. También funcionan autónomamente el telescopio ESO (2.2m), dentro del programa CRON, encargados del seguimiento de GRB's detectados por diversas misiones espaciales, pero tampoco en este caso el sistema es público.

Los sistemas públicos de administración de telescopios como ser ACP, The Sky-ORCHESTRATE, etc. no son fácilmente modificables como para realizar estrategias de detección de objetos móviles o, en general, de toma de decisiones a partir de la lectura de las imágenes.

El Northern Optical Astronomical Observatory (NOAO) desarrolló IRAF [3] (Image Reduction and Analysis Facility), un sistema que utiliza las subrutinas de la librería CFitsIO [4] que implementa casi todos los métodos de reducción de datos digitales, incluyendo los métodos del paquete DAOPHOT [5]. El principal inconveniente de este sistema es que es está pensado para el análisis manual de los datos, por lo que usarlo como una librería con una interfaz programable resulta complicado. Existen los wrappers de python, PyRAF [6], pero éstos no mejoran los inconvenientes fundamentales de la operación programada de IRAF. Además, IRAF cuenta con muchas funcionalidades adicionales que no son de utilidad pa-

ra los efectos de este trabajo y hacen la instalación del sistema innecesariamente compleja y, más importante, no cuenta con rutinas de detección de objetos móviles.

Por eso en este trabajo se decidió reimplementar las funcionalidades requeridas (parte del paquete DAOPHOT) de una forma eficiente y práctica para la automatización del proceso de observación.

3. DESCRIPCIÓN DEL MÉTODO PROPUESTO

3.1. Scope

El proceso completo del descubrimiento de un objeto móvil en el cielo nocturno abarca desde la instalación del telescopio hasta el reporte del descubrimiento cuando existe la certeza de que es un objeto natural desconocido, y comprende el análisis fotométrico y astrométrico correspondiente.

El alcance de este trabajo incluye casi por completo este proceso, desde el control del telescopio, la adquisición de imágenes y su alineación, la detección de objetos móviles y su análisis astronómico. El sistema en funcionamiento en una plataforma adecuada (con buenas condiciones atmosféricas y lumínicas) puede descubrir y monitorear objetos móviles previamente desconocidos.

La implementación es configurable para descubrir tanto asteroides como basura espacial (dependiendo del campo de observación y los tiempos de integración), e ignora otros fenómenos de características diferentes. Con mínimas modificaciones puede adaptarse al monitoreo o a la búsqueda de planetas extrasolares.

El desarrollo propuesto es aplicable a prácticamente cualquier telescopio con montura robótica, sensor CCD controlable por software, y una unidad de procesamiento (que puede ser una computadora, notebook o procesador embebido) con la capacidad de correr el framework requerido bajo el sistema operativo Linux.

En esta primera versión que se presenta como resultado de esta tesis, el sistema se limita a la detección de objetos móviles con trayectorias lineales en el plano del cielo, y perfil de brillo del objeto similar al de las estrellas.

3.2. Framework

El hardware requerido para la utilización de este sistema es: un telescopio con una montura robótica y sensor CCD con interfaces programables (drivers), y una computadora como unidad de procesamiento. Los drivers soportados en esta versión son el INDI para la montura Temma, y el driver de Apogee para el sensor CCD, pero el diseño del sistema permite agregar drivers para otros dispositivos muy fácilmente.

El software está escrito en lenguaje Python, por lo que es portable a cualquier sistema operativo con intérprete python (Unix, OSX, Windows), la versión presentada está diseñada para Ubuntu 12.10+. Además, el sistema depende de las siguientes librerías:

Python-OpenCV:
Open Computer Vision Library [7] con soporte para Python. Usada en el

procesamiento de imágenes, especialmente las funciones de transformación, filtrado y convolución de matrices, y FLANN (Fast Library for Approximate Nearest Neighbors [8]).

NumPy [9]:
Librería numérica de Python. Se usan las estructuras de datos y operaciones básicas con matrices y ajuste con cuadrados mínimos, entre otros.

SExtractor [10]:
Source Extractor extrae posiciones y magnitudes de las fuentes (estrellas y astros) de las imágenes capturadas. Usa técnicas de redes neuronales. En este trabajo también presentamos la implementación de otro método de extracción de fuentes, DAOPHOT [5]. El software puede trabajar con cualquiera de los 2 métodos y los 2 resultan comparativamente efectivos.

PyFITS [11]:
Librería de lectura-escritura de imágenes FITS y la meta-información. FITS (Flexible Image Transport System) es el formato estándar de imágenes astronómicas. Almacena muchísimo más rango lumínico que otros formatos de imagen y puede almacenar metadatos de fotometría y calibración espacial.

Astrometry.net [12]:
Astrometric calibration API. Es un servicio de calibración astrométrica de datos que ubica espacialmente imágenes astronómicas en el cielo. Si bien tenemos los parámetros astrométricos del telescopio al momento de la adquisición de imágenes, éstos suelen ser imprecisos o hasta incorrectos. Por eso se usa este servicio para calibrar la posición y tener una conversión precisa entre píxeles y coordenadas astronómicas.

El servicio tiene una interfaz programable vía HTTP, por lo que es necesaria la conexión a internet desde la unidad de procesamiento, pero es posible bajar los catálogos y instalar el servicio de forma local si se quiere implementar el software en un ambiente sin conexión.

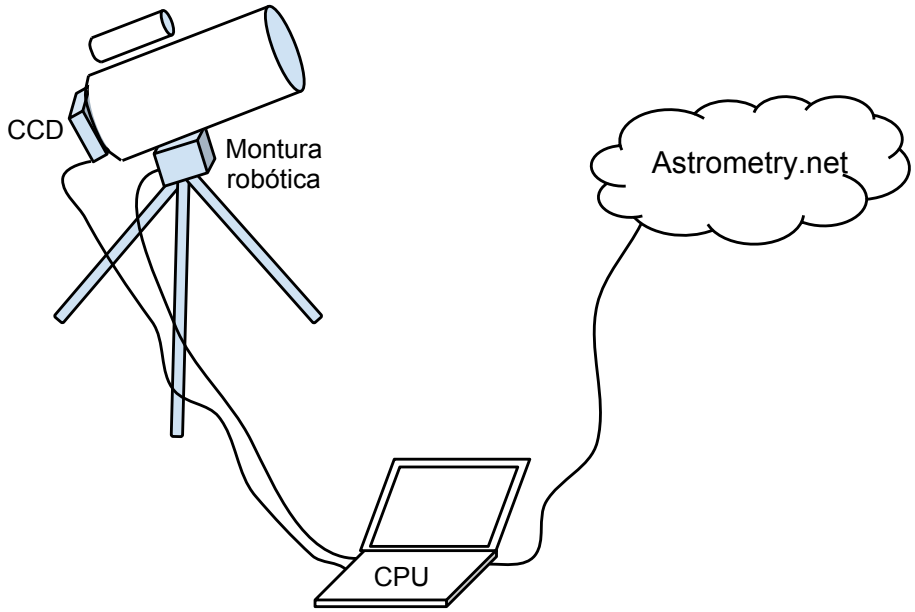


Fig. 3.1: Esquema del sistema

3.3. Adquisición

El proceso de adquisición de imágenes consiste básicamente en el control adecuado del sensor CCD para obtener las imágenes en formato FITS, y escribir los metadatos correspondientes en el header. Para obtener datos de buena calidad es necesario que la observación sea en un lugar alejado de fuentes luminosas (como ciudades), con un cielo despejado y preferentemente sin luna, con buenas condiciones atmosféricas.

Para la exposición se abre el diafragma por un tiempo determinado (tiempo de integración) y el CCD produce una matriz de datos (píxeles) que representa el flujo de energía proveniente de una región del cielo detectado alrededor del eje óptico del telescopio en su plano focal. Mediante la reducción apropiada de esta matriz pueden determinarse posiciones en la esfera celeste (astrometría) y medirse los flujos de energía de las fuentes (fotometría).

Es conveniente que el sensor esté frío ($\sim$-25ºC) durante este proceso para no introducir ruido; por eso los CCD incluyen un sistema de enfriado. La aplicación es responsable de manejar parámetros como temperatura, velocidad de los ventiladores, tiempo de exposición, etc.

El resultado de la adquisición es una matriz de 16MPixel (4096x4096 pixels en

el caso del CCD Apogee Alta F16) con valores de luminosidad para cada pixel, que se guarda en formato FITS sin pérdida de información. Se pueden usar distintos filtros para el telescopio, pero el resultado de la captura es monocromático, i.e., un escalar por cada pixel.

En la figura 3.2 se observa una imagen típica adquirida con el sensor CCD.

Fig. 3.2: Una imagen obtenida de una exposición (y convertida a valores adecuados para la visualización)

Una vez obtenidos los datos se agregan metadatos e información adicional en el header del archivo. Entre los datos más importantes que se deberían almacenar podemos mencionar:

JUL-DATE:
El dia-hora (timestamp) de la exposición en formato de fecha Juliana, es decir, el tiempo en días transcurrido desde el mediodía del 1 de enero de 4713 BC (JD). También se usa el MJD = JD - 2400000.5 (días desde el 17 Nov. 1858), o J2000.0 = JD - 2451545.0 (días desde 1 Ene. 2000). Se puede almacenar el timestamp del comienzo, mitad o final de la exposición.

RA, DEC (Ascensión Recta y Declinación):
Las coordenadas ecuatoriales del centro de la imagen. RA, o α, es el ángulo del objeto proyectado sobre el ecuador celeste desde el punto Aries (equivalente a la longitud geográfica) y DEC, δ, es el ángulo entre el ecuador celeste y el objeto (equivalente a la latitud geográfica). Estos dos valores forman un espacio de coordenadas del cielo en el que las estrellas y objetos lejanos están fijos a lo largo del tiempo, por lo que resulta útil para describir posiciones de objetos en el cielo.

Esta información proviene en principio de la montura del telescopio, es decir, de las coordenadas a las que apuntamos el telescopio para la observación. Por esta razón no cuenta con mucha precisión ya que depende de la calibración/alineación del telescopio y contiene errores mecánicos.

En la figura 3.3 se puede observar el sistema de coordenadas ecuatoriales.

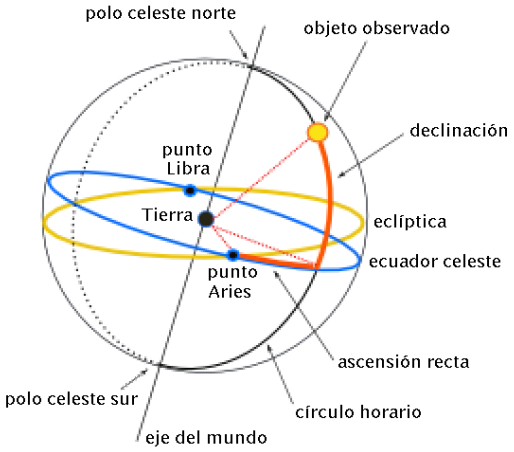


Fig. 3.3: Coordenadas ecuatoriales

CD:

Coordinate Description matrix. La matriz que define la rotación y escala de la imagen respecto a las coordenadas ecuatoriales.

$$\begin{pmatrix} \text{CD1\_1} & \text{CD1\_2} \\ \text{CD2\_1} & \text{CD2\_2} \end{pmatrix} = scale \cdot \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

La escala (scale) depende de la distancia focal del telescopio, f (es fija para un telescopio determinado) y el ángulo θ es idealmente 0, pero depende de la correcta alineación del telescopio. La relación es:

$$scale = tan^{-1}(\frac{px}{f}) \qquad \text{, donde } px \text{ es el tamaño del pixel en arcsec.}$$

Con esta matriz y RA, DEC se puede construir una transformación entre coordenadas en pixels de la imagen y coordenadas ecuatoriales (ver 3.8.1)

AIRMASS:

La masa de aire/atmósfera que existe entre el telescopio y el objeto observado. Depende del ángulo del telescopio respecto al horizonte y la altura sobre el nivel del mar, entre otros.

Además se almacenan los datos del telescopio, del CCD, tiempo de exposición y otros datos.

Las imágenes generalmente contienen un par de cientos de estrellas (más o menos dependiendo de la región del cielo y del campo angular de los instrumentos). El perfil en la imagen de cada estrella se puede aproximar con una curva Gaussiana 2D (si la cámara está correctamente enfocada). El perfil unidimensional de esta gaussiana se caracteriza en astronomía por medio de su ancho a la mitad del valor máximo, que es el FWHM (Full Width at Half Maximum), que depende del proceso dispersivo en la atmósfera y por lo tanto es aproximadamente constante para todas las fuentes puntuales de la imagen.

Además la imagen contiene ruido de fondo que se caracteriza con dos componentes: una aditiva y una multiplicativa. La componente aditiva se sustrae utilizando imágenes de tiempo de integración 0 (bias) y imágenes de obturador cerrado (darks). La componente multiplicativa se calibra utilizando imágenes de fuentes que iluminan homogéneamente el detector (flat).

También se producen detecciones espúreas causadas por rayos cósmicos que aleatoriamente llegan al sensor y que provocan picos de intensidad en la imagen, o píxeles defectuosos del sensor.

En la figura 3.4 pueden observarse los valores de una fila de pixels de una imagen correspondiente a una detección espúrea (causada por un rayo cósmico) y una estrella.

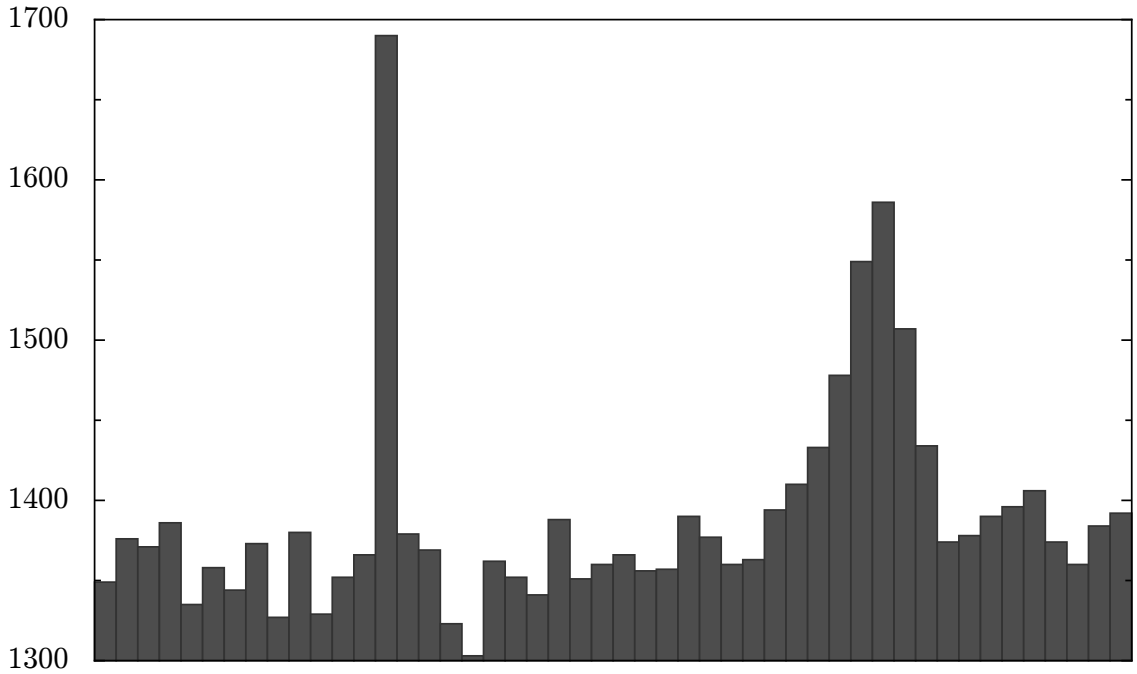


Fig. 3.4: Una fila de pixels en la que se observa un típico rayo cósmico y una estrella.

En las figuras 3.5, 3.6, 3.7, 3.8 se observan imágenes de tres estrellas de diferente brillo y un pixel afectado por un rayo cósmico.

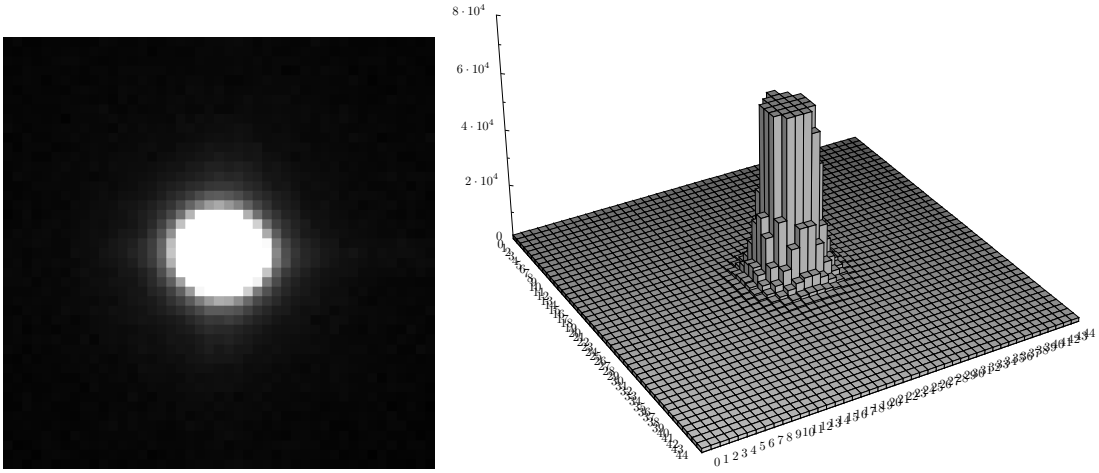


Fig. 3.5: Una estrella brillante

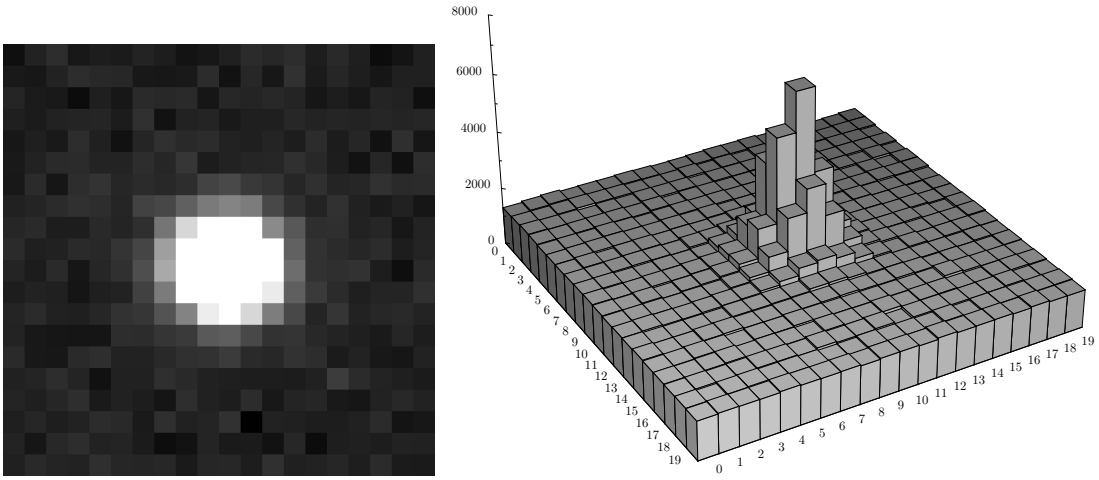


Fig. 3.6: Una estrella de brillo medio

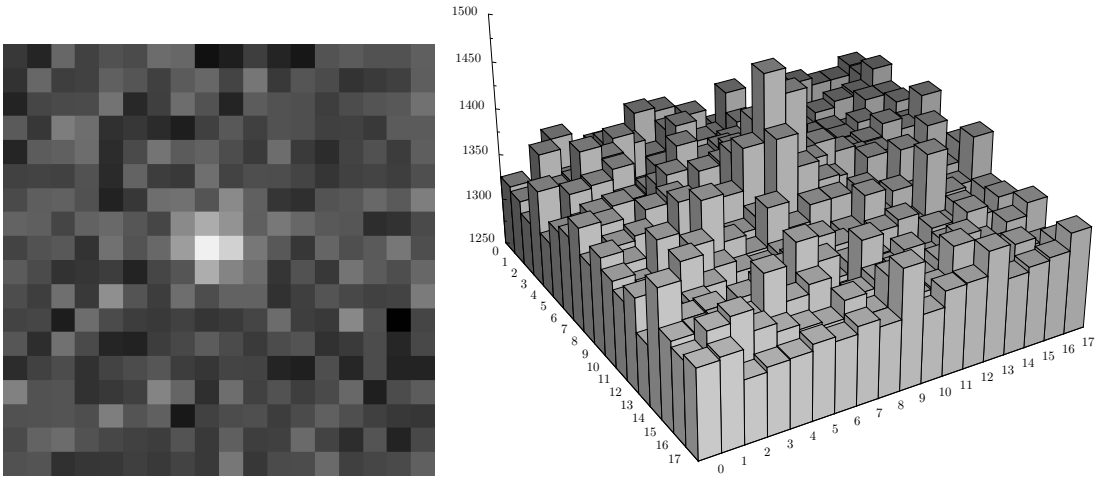


Fig. 3.7: Una estrella que casi se pierde en el ruido de fondo

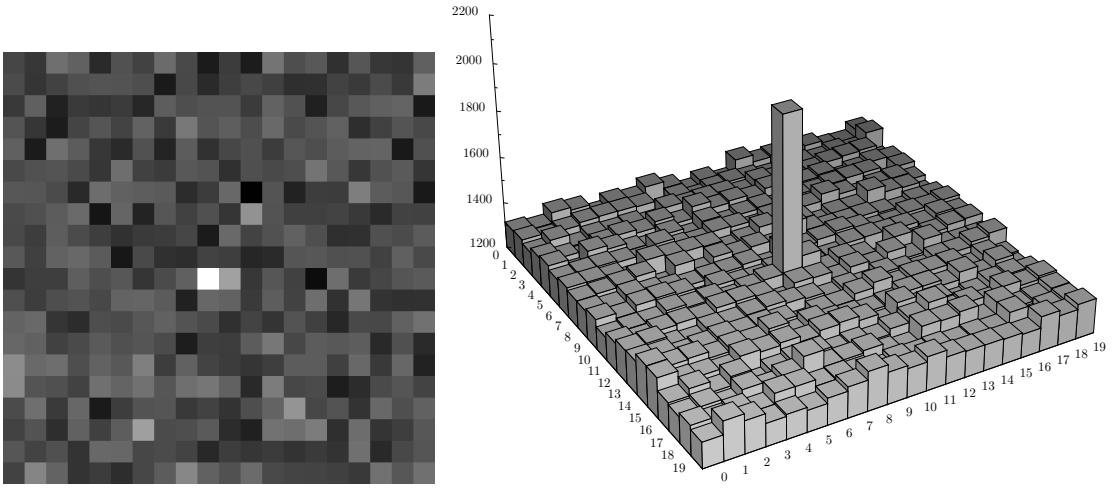


Fig. 3.8: Un pixel afectado por un rayo cósmico

Los objetos móviles que nos interesa estudiar aparecen en la imagen con el mismo perfil que las estrellas, siempre y cuando se cumpla la siguiente condición sobre la velocidad v del objeto en cuestión:

$$v \ll \frac{\text{FWHM}}{\Delta t_{int}} \qquad \text{, donde } \Delta t_{int} \text{ es el tiempo de integración o de exposición.}$$

Si el objeto emite (o refleja) luz y el tiempo de exposición es el adecuado, tiene el mismo perfil que una estrella en la imagen capturada. Si el tiempo de exposición es mayor aparecería elongado o como un trazo en la imagen, pero detectar ese tipo de objeto queda para trabajo futuro.

La forma de detectar los objetos móviles es capturando una secuencia de imágenes y analizando si hay un movimiento de uno (o más) de los astros respecto a los demás. La captura de secuencias no es más que la captura de imágenes sucesivas a intervalos establecidos. Estos intervalos deberían ser suficientemente largos como para que el objeto se mueva perceptiblemente, es decir:

$$\Delta t \gg \frac{FWHM}{v} \qquad \text{, donde } \Delta t \text{ es el intervalo entre sucesivas imágenes,}$$

pero suficientemente corto como para que el objeto aparezca al menos 4 veces antes de desaparecer del campo:

$$\frac{D}{N\Delta t} > v \qquad \text{, siendo } D \text{ el tamaño del campo y con } N = 4$$

Los intervalos de tiempos entre imágenes Δt se eligen para optimizar la detección de objetos con velocidad característica de los objetos de interés. En la tabla 3.1 se muestran las velocidades y los tiempos de intervalos entre imágenes Δt de distintos objetos detectables.

En la tabla 3.1 se muestran las velocidades (en arcsec/min) típicas en el plano del cielo en la época de la oposición de diferentes tipos de objetos: NEO (Near Earth Object; objeto cercano a la Tierra), MBA (Main Belt Asteroid; Asteroide del cinturón principal), MBAE (Main Belt Asteroid Exterior; del cinturón exterior), Troyano (asteroide en la órbita de Júpiter), Centauro (asteroide del sistema solar exterior) y TNO (Trans-Neptunian Object; objeto trans-neptuniano). También se muestran aproximaciones de tiempos (en horas) típicos de recorrido del objeto en un campo de 30 arcmin.

Tipo	Velocidad ($\frac{\text{arcsec}}{\text{min}}$)	Tiempo (h)
NEO	1	30
MBA	0.5	60
MBAE	0.4	75
Troyano	0.3	100
Centauro	0.1	300
TNO	0.06	500

Tab. 3.1: Velocidades típicas de distintos tipos de objetos móviles

3.4. Reducción

La reducción de datos es el proceso de simplificar la información obtenida de un experimento u observación (adquisición) y extraer datos significativos y relevantes al problema. En este caso nos interesa obtener información de las imágenes que nos permita analizarlas para encontrar objetos móviles.

Inicialmente tenemos una secuencia de imágenes con las características antes mencionadas; lo que resulta más obvio de extraer es un índice de las estrellas y astros que contiene cada imagen. Esto parece simple en un principio, pero hay algunos factores a tener en cuenta. Dada la naturaleza de las imágenes, éstas a veces contienen mucho ruido, ya que la luz que llega de las estrellas a la Tierra es poca, y hay muchos factores como la atmósfera y fuentes luminosas (por ejemplo la luna) que afectan la calidad de la imagen. Por esto el método tiene que ser robusto y confiable y debería obtener buenos resultados independientemente de las condiciones.

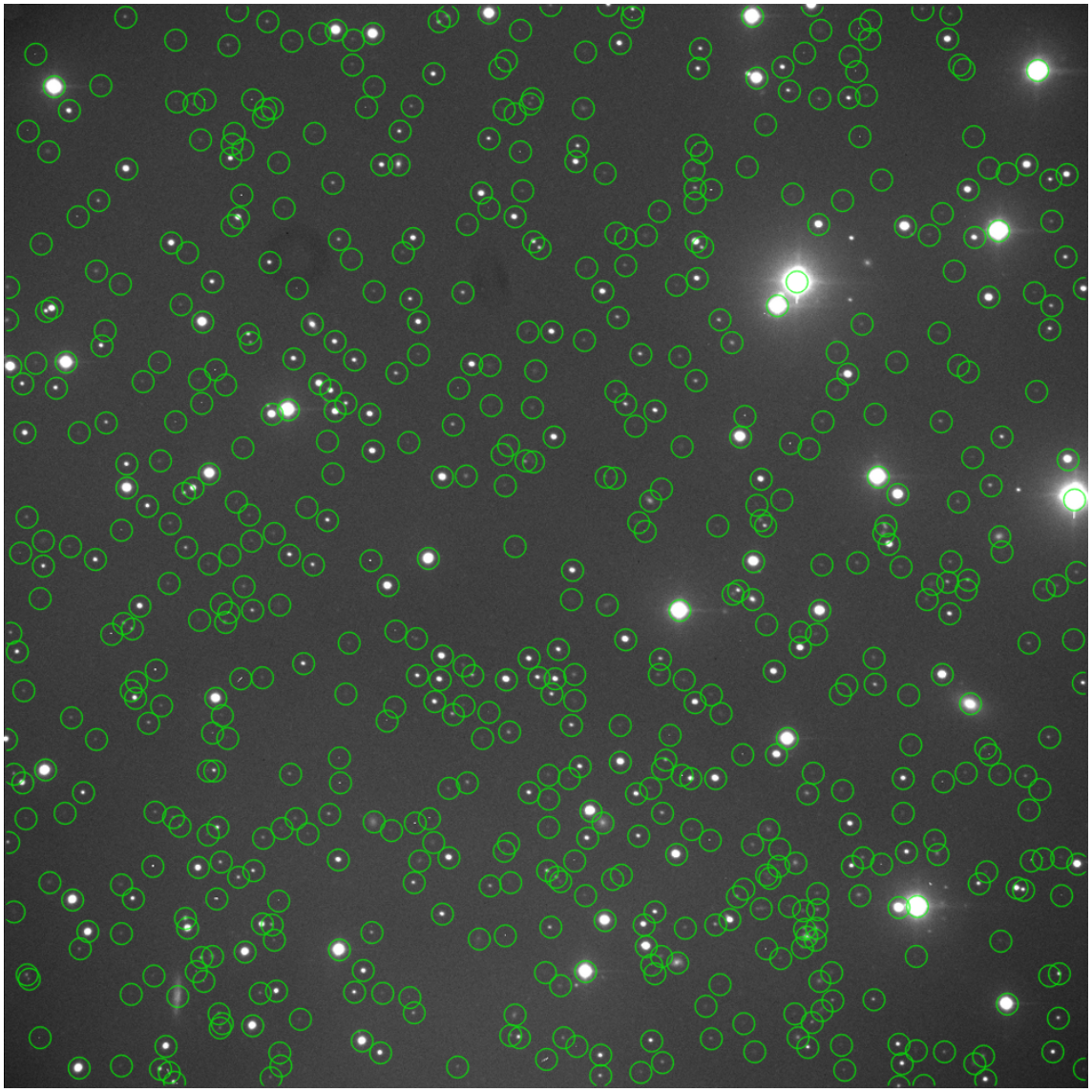


Fig. 3.9: Fuentes detectadas en una imagen

A continuación se detallan dos métodos usados (indistintamente) en la aplicación: DAOPHOT [5], que implementamos en Python como parte del trabajo a partir del paper original, y SExtractor [10], que lo usamos como una librería externa.

3.4.1. DAOPHOT

Una forma muy naïve de extraer las posiciones de las estrellas de la imagen es buscar los máximos locales. Esto obviamente resulta poco efectivo ya que el ruido del sensado introduce muchos picos que son máximos locales y que no son estrellas, y algunas estrellas no tienen un máximo bien definido (pueden estar truncadas por saturación y tener forma de meseta en vez de pico).

El método propuesto DAOPHOT [5] es una versión mejorada de la extracción de máximos que elimina estos errores. La forma en la que opera es aplicando primero un filtro de convolución a la imagen original, y luego analiza los máximos locales para eliminar falsos positivos. Además, calcula el centroide de cada feature encontrado para obtener precisión de sub-pixel en la posición.

Filtrado

El objetivo del filtrado es responder para cada pixel de la imagen le pregunta ¿Si una estrella estuviera centrada en este pixel, cuán brillante sería? La respuesta es un valor numérico que se calcula ajustando una curva gaussiana a una vecindad del pixel. Si corresponde al centro de una estrella, el ajuste va a ser bueno y por lo tanto va a tener un valor proporcional al brillo de la estrella. Si por el contrario el pixel es parte del fondo o del borde de una estrella, este valor va a ser bajo o incluso negativo. Por lo tanto, los píxeles de la imagen filtrada donde el valor es positivo y grande, probablemente correspondan con posiciones cercanas al centro de una estrella o fuente puntual. En detalle, según el método propuesto en [5], tenemos que:

Sea D la matriz que representa la imagen original, y $D_{i,j}$ el valor del pixel (i, j) de la imagen (proporcional a la cantidad de luz recibida en ese pixel en la exposición). Sea G una gaussiana bidimensional, circular de altura 1:

$$G(\Delta i, \Delta j; \sigma) = e^{-\frac{(\Delta i^2 + \Delta j^2)}{2\sigma^2}}$$

Sea H_{i_0,j_0} el brillo central del perfil que mejor ajusta a una estrella centrada en (i_0, j_0) en la imagen y b la estimación del nivel basal local (el valor promedio de los pixels si no estuviera la estrella). Entonces se cumple la relación:

$$D_{i,j} \doteq H_{i_0,j_0} G(i - i_0, j - j_0; \sigma) + b$$

con (i, j) cerca de (i_0, j_0), y $\doteq$ representa el ajuste por cuadrados mínimos de los datos para los pixels (i, j) en una región alrededor de (i_0, j_0).

Luego el valor de H_{i_0,j_0} está dado por cuadrados mínimos:

$$H_{i_0,j_0} = \frac{\sum(GD) - (\sum G)(\sum D)/n}{\sum(G^2) - (\sum G)^2/n}$$

donde n es la cantidad de píxeles del área.

Luego, se calcula la matriz H (de las mismas dimensiones que D) que es el valor de $H_{i,j}$ para cada pixel de la imagen. Para esto se usa un filtro de convolución que es equivalente a la ecuación anterior:

$$H_{i_0,j_0} = \sum_{i,j} (W_{(i-i_0,j-j_0)} D_{i,j})$$

donde

$$W_{\Delta i,\Delta j} = \frac{G(\Delta i, \Delta j; \sigma) - (\sum G)/n}{\sum (G^2) - (\sum G)^2/n}$$

En la figura 3.10 se observa una fila de pixels antes y después de la aplicación del filtro W que contiene una estrella, un par de estrellas fusionadas, una galaxia, una detección espúrea de un rayo cósmico, y un pixel defectuoso.

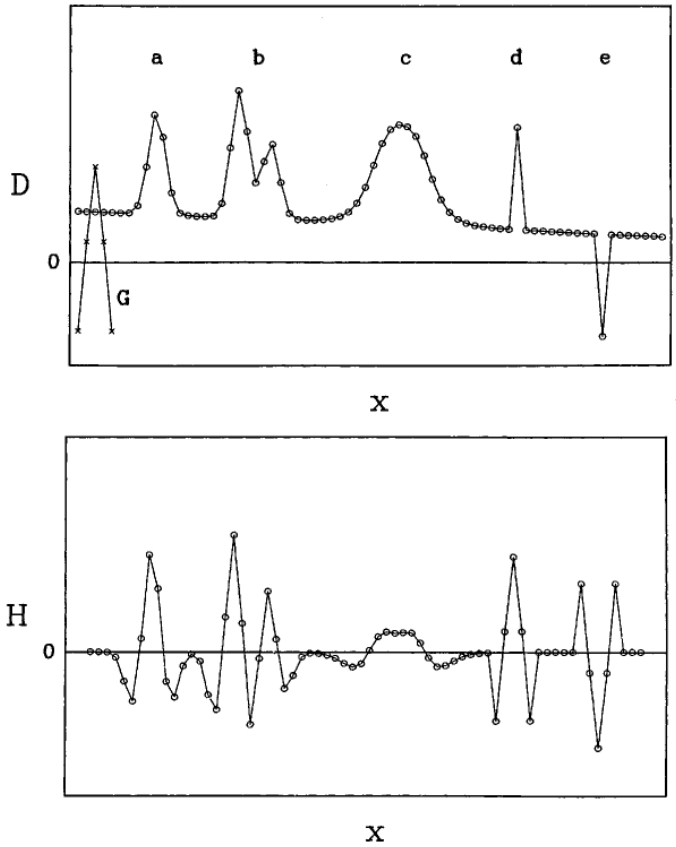
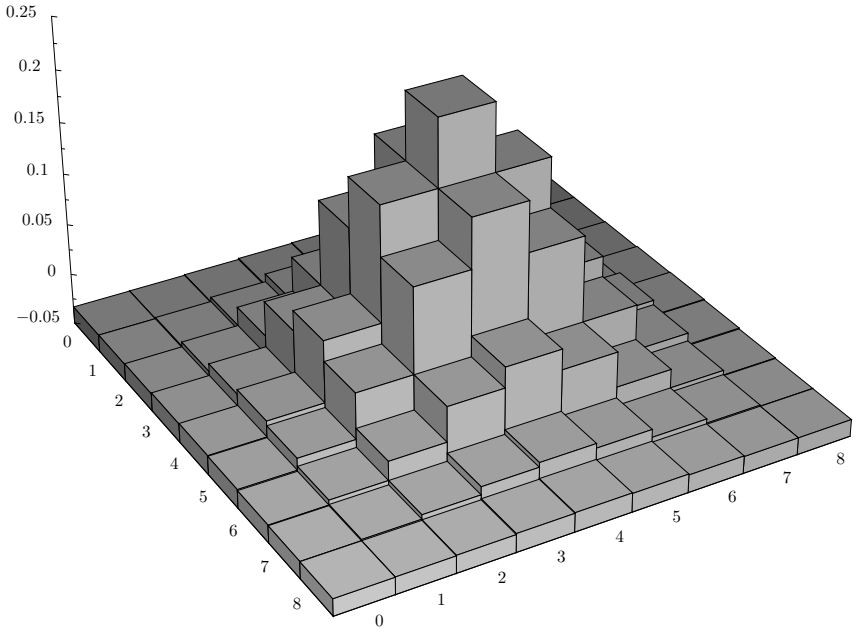


Fig. 3.10: Representación en 1D de la imagen original y luego de la aplicación del filtro W. (a) una estrella, (b) estrellas fusionadas, (c) una galaxia, (d) un rayo cósmico, (e) un pixel defectuoso. Imagen extraída de [5].

Por lo tanto, aplicando el filtro de convolución W a la matriz D (la imagen original), obtenemos la matriz H en donde los máximos representan puntos para los cuales hay un buen ajuste de una curva gaussiana, y por lo tanto, una buena probabilidad de que sea el centro de una estrella o fuente luminosa.



$$\begin{bmatrix} -0{,}033 & -0{,}033 & -0{,}033 & -0{,}032 & -0{,}031 & -0{,}032 & -0{,}033 & -0{,}033 & -0{,}033 \\ -0{,}033 & -0{,}032 & -0{,}028 & -0{,}020 & -0{,}016 & -0{,}020 & -0{,}028 & -0{,}032 & -0{,}033 \\ -0{,}033 & -0{,}028 & -0{,}009 & +0{,}025 & +0{,}045 & +0{,}025 & -0{,}009 & -0{,}028 & -0{,}033 \\ -0{,}032 & -0{,}020 & +0{,}025 & +0{,}108 & +0{,}156 & +0{,}108 & +0{,}025 & -0{,}020 & -0{,}032 \\ -0{,}031 & -0{,}016 & +0{,}045 & +0{,}156 & +0{,}222 & +0{,}156 & +0{,}045 & -0{,}016 & -0{,}031 \\ -0{,}032 & -0{,}020 & +0{,}025 & +0{,}108 & +0{,}156 & +0{,}108 & +0{,}025 & -0{,}020 & -0{,}032 \\ -0{,}033 & -0{,}028 & -0{,}009 & +0{,}025 & +0{,}045 & +0{,}025 & -0{,}009 & -0{,}028 & -0{,}033 \\ -0{,}033 & -0{,}032 & -0{,}028 & -0{,}020 & -0{,}016 & -0{,}020 & -0{,}028 & -0{,}032 & -0{,}033 \\ -0{,}033 & -0{,}033 & -0{,}033 & -0{,}032 & -0{,}031 & -0{,}032 & -0{,}033 & -0{,}033 & -0{,}033 \end{bmatrix}$$

Fig. 3.11: La matriz de convolución W

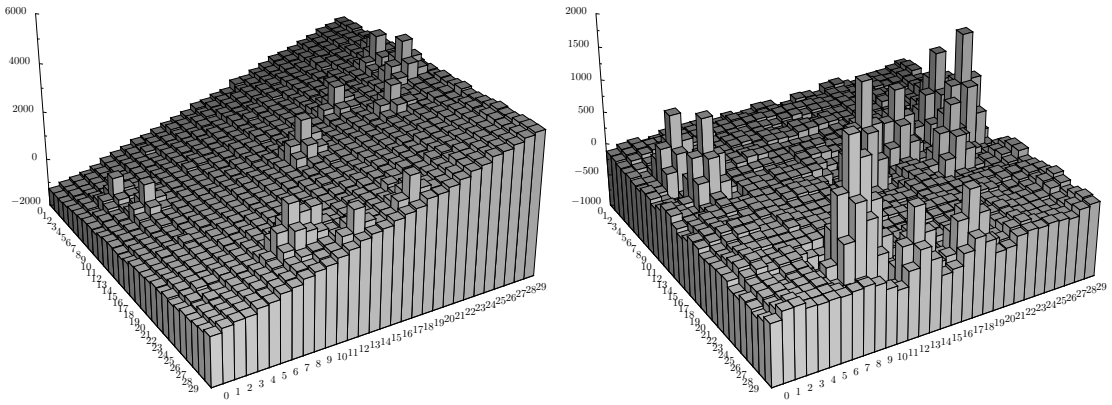


Fig. 3.12: Imagen sintética y el resultado del filtrado. Observar que el fondo queda en 0

De la imagen filtrada, H, se obtienen los máximos locales, es decir, los píxeles que tienen un valor mayor al de sus vecinos. Además, este valor debe estar por encima de un umbral para separar los features del ruido, como se observa en la figura 3.13. De esta forma obtenemos un representante por cada estrella, que es aproximadamente su centro.

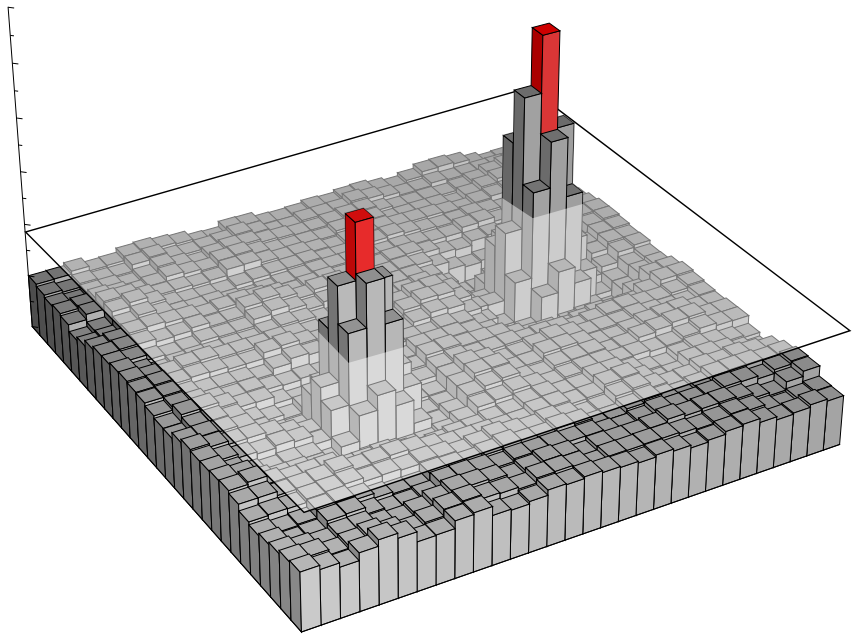


Fig. 3.13: Extracción de máximos que están por encima de un umbral `Hmin`

Selección

Algunos de los puntos hallados no corresponden a estrellas o objetos de nuestro interés. Estos falsos positivos pueden ser rayos cósmicos, manchas o simplemente artefactos del ruido o imperfecciones. Para filtrar estos features, los autores de [5] proponen analizar su "acritud" (sharpness) y redondez (roundness).

El parámetro de sharpness del feature en la posición (i_0, j_0) se define como:

$$\text{sharp}_{i_0,j_0} = \frac{D_{i_0,j_0} - \langle D_{i,j} \rangle}{H_{i_0,j_0}}$$

siendo $\langle D_{i,j} \rangle$ el promedio de los $D_{i,j}$ con (i, j) alrededor de (i_0, j_0), pero excluyéndolo.

El trabajo original propone que los valores de sharpness para estrellas reales se encuentran entre 0.2 y 1.0, pero haciendo nuestro propio análisis de esta propiedad para puntos reconocidos como válidos y falsos positivos, encontramos experimentalmente que el rango de valores para features válidos puede estar entre 0.2 y 0.7.

La redondez de un feature es un parámetro de la "elongación" de una fuente, es decir, compara el ancho y el alto del ajuste obtenido. Se define el parámetro roundness como:

$$\text{round} = 2\frac{h_y - h_x}{h_y + h_x}$$

donde h_y y h_x son convoluciones como la W mostrada anteriormente, pero en una sola dimensión, i.e., de una sola fila a lo alto y largo respectivamente. Es decir, corresponden a las gaussianas

$$g_x(\Delta i;\sigma) = e^{-\frac{\Delta i^2}{\sigma^2}} \quad \text{y} \quad g_y(\Delta j;\sigma) = e^{-\frac{\Delta j^2}{\sigma^2}} \text{ respectivamente}$$

Este parámetro descarta fácilmente filas y columnas de píxeles como las que aparecen por saturación, aunque no filtra las fuentes elongadas en "diagonal". El trabajo original proponía rangos entre -1.0 y 1.0, pero analizando nuestros datos encontramos que podemos ajustar este rango aún más, a -0.5 hasta 0.5, como se observa en la figura 3.14.

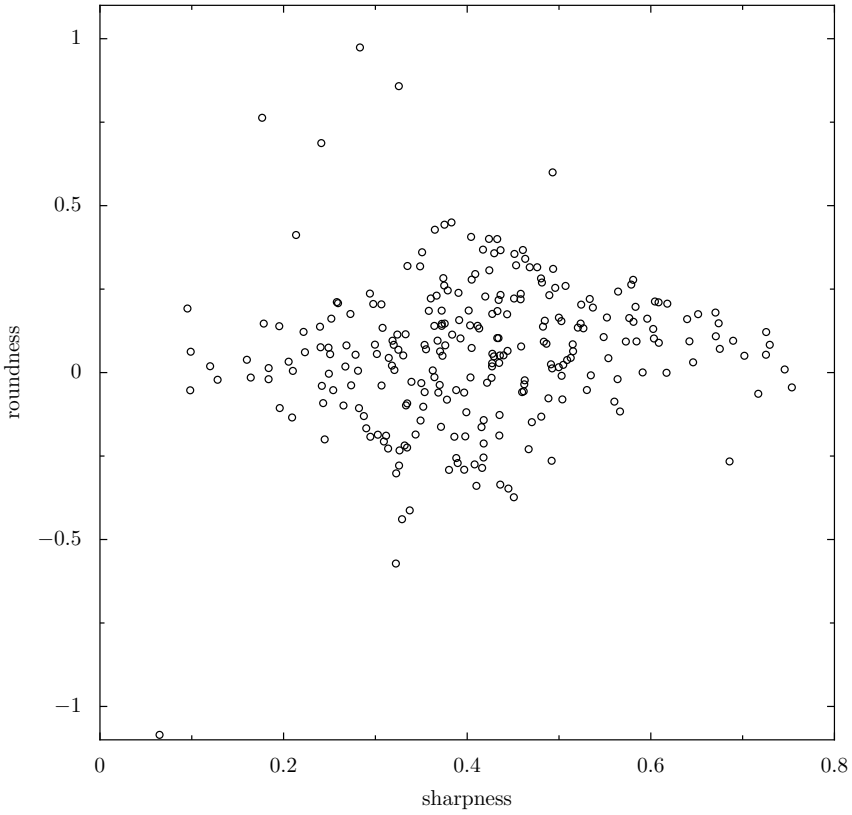


Fig. 3.14: Gráfico de los valores de sharpness y roundness de los features.

Entonces, los las posiciones de las estrellas y astros que nos interesan son los puntos hallados con este método que tienen parámetros de sharpness y roundness en los rangos indicados.

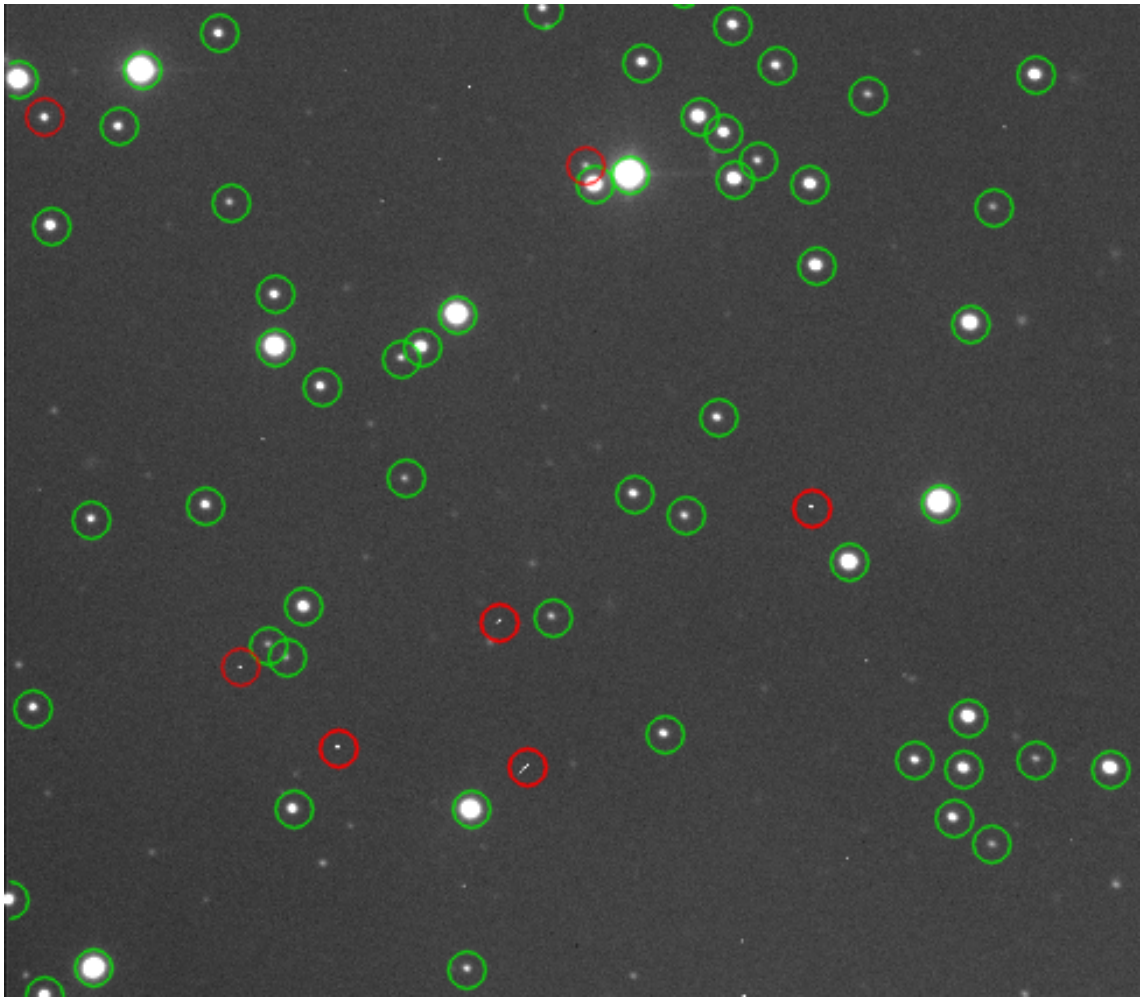


Fig. 3.15: En la imagen se muestran con un círculo los máximos locales encontrados y en rojo los features filtrados por sharpness y/o roundness (fuera de rango).

Refinamiento sub-pixel de la posición

El método extrae los máximos de la imagen filtrada H que son representantes de las fuentes de la imagen. Estas posiciones son enteras, por lo que tienen la precisión del tamaño del pixel. Es posible refinar aún más esta posición calculando el centroide de cada feature.

El centroide (o centro de masa) de un feature en la matriz H se puede calcular como:

$$C_{i_0,j_0} = \frac{\sum_{i,j} H_{i,j} \cdot (i,j)}{\sum_{i,j} H_{i,j}}$$

con (i, j) en una vecindad de (i_0, j_0).

El resultado es una posición con precisión de sub-pixel como se observa en la figura 3.16.

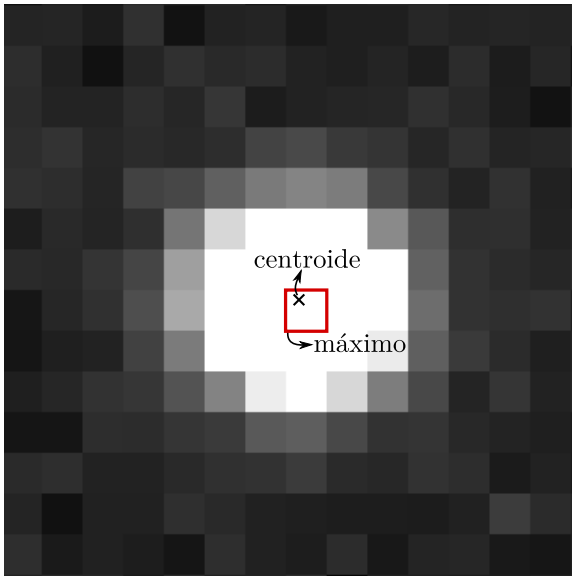


Fig. 3.16: Pixel con el valor máximo de la fuente y centroide del objeto con precisión sub-pixel

3.4.2. SExtractor

SExtractor (Source Extractor) [10] es un software de extracción de fuentes luminosas de imágenes basado en redes neuronales. El método comienza por estimar el "background" de la imagen para homogeneizarla y eliminar el nivel de luz basal. Luego aplica una segmentación a la imagen para caracterizar áreas de distinta luminosidad. Con esto separa las posibles fuentes luminosas de las áreas vacías. A la imagen segmentada le aplica un filtro de convolución. Acá es donde intervienen las redes neuronales, ya que las convoluciones que utiliza provienen de entrenamientos con refuerzos de otras imágenes analizadas. Luego hace "deblending", que es el proceso de separar las fuentes que pueden estar superpuestas. Este paso es importante para imágenes de áreas muy densas de fuentes, pero es trivial para imágenes con relativamente pocas estrellas que están suficientemente alejadas entre sí. Finalmente calcula la fotometría de las fuentes encontradas sumando el flujo de luz en áreas alrededor del centro de la fuente.

SExtractor tiene la capacidad de calcular muchos parámetros de cada fuente. Esto está orientado a la detección de galaxias y otras fuentes, pero a fines de nuestro trabajo sólo nos interesa el flujo lumínico de las estrellas y posibles objetos móviles. Por ello sólo extraemos los parámetros de posición (X, Y) y la luminosidad (FLUX) de cada fuente.

Con cualquiera de los dos métodos antes mencionados (DAOPHOT y SExtractor) obtenemos las posiciones X, Y de las fuentes (estrellas y otros astros) de cada imagen, que nos va a servir para los siguientes pasos del método propuesto.

3.5. Fotometría

La fotometría es una técnica desarrollada a partir de la necesidad de medir el brillo de las estrellas.

En primera instancia se obtiene una magnitud instrumental, que depende de las características de los instrumentos incvolucrados en la medición, que luego se convierte a un valor estándar, refiriendo la medida a un patrón de referencia, típicamente el brillo de estrellas previamente catalogadas. Para la reducción fotométrica es necesario calibrar el efecto de la extinción atmosférica, es decir estimar el coeficiente de extinción realizando mediciones diferenciales con fuentes calibradas.

Para determinar el brillo con mayor exactitud debe contarse con mediciones en un rango de longitudes de onda o en varios filtros, pero esa calibración más precisa está fuera de los objetivos de este trabajo, dado que no se cuenta todavía con una rueda de filtros.

La determinación de la magnitud instrumental se realiza usualmente a través de lo que se ha dado en llamar fotometría de apertura. Esta medición consiste simplemente en determianr la cantidad de cuentas significativas dentro de un área que contiene a la fuente de interés. El número de cuentas significativas es la obtenida después de sustraer el nivel de ruido y de haber compensado otros factores que degradan la imagen.

3.5.1. FLUX

Como parte del método de DAOPHOT en este trabajo incluimos una medida del flujo de las fuentes encontradas, que se aleja un poco del método propuesto en el trabajo original por simplicidad y eficiencia. El objetivo es obtener para cada fuente un descriptor que distinga los features entre sí y que sea constante a lo largo de los frames de modo de poder identificarlos.

Lo que obtenemos para cada feature es el flujo de luz recibido de la fuente, que representa la cuenta de los fotones recibidos en algún área cerca de la posición del feature y restando el "background", que es la cuenta de esa área si no existiera la fuente.

Para esto necesitamos una estimación del background de toda la imagen. Definimos la matriz BG a partir de D (la imagen original) como un filtro de erosión (filtro de convolución de mínimos) de D:

$$BG_{i_0,j_0} = \min(D_{i,j})$$

con (i, j) dentro de un radio r alrededor de (i_0, j_0).

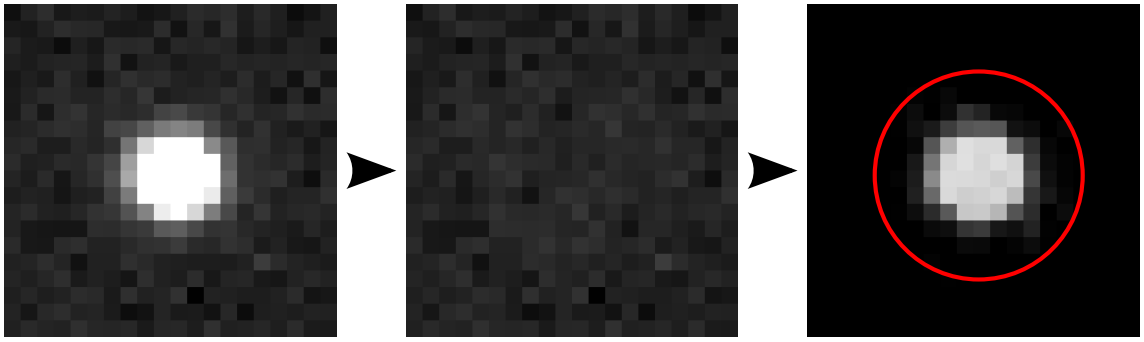


Fig. 3.17: (a) Imagen original (b) estimación del fondo (c) background sustraído

Resulta efectivo aplicar un filtro de mediana a D antes del filtro de erosión para deshacerse de minimos defectuosos que afectan la estimación del fondo.

El flujo de una fuente ubicada en (i_0, j_0) se calcula entonces como

$$FLUX_{i_0,j_0} = \sum_{(i,j)} (D_{i,j} - BG_{i,j})$$

con (i, j) en un círculo alrededor de (i_0, j_0) (ver figura 3.17).

En campos poco poblados se puede calcular el flujo de una fuente sin estimar el background tomando como referencia del nivel basal un anillo alrededor de la fuente. En este caso se puede definir como:

$$FLUX_{i_0,j_0} = \sum_{\substack{(i,j)\in \\ \text{circ}(i_0,j_0)}} D_{i,j} - \sum_{\substack{(i,j)\in \\ \text{ring}(i_0,j_0)}} D_{i,j} \times \frac{\#\text{circ}(i_0, j_0)}{\#\text{ring}(i_0, j_0)}$$

donde $\text{circ}(i_0, j_0)$ y $\text{ring}(i_0, j_0)$ son regiones circulares y de anillo alrededor de (i_0, j_0) respectivamente (ver figura 3.18).

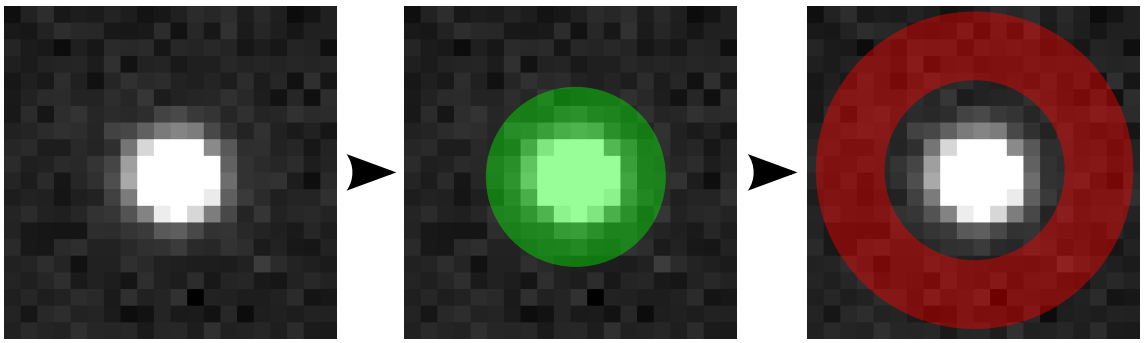


Fig. 3.18: (a) Imagen original (b) $\text{circ}(i_0, j_0)$ (c) $\text{ring}(i_0, j_0)$

Este valor obtenido es proporcional a la cantidad de luz recibida de la fuente y es estable a lo largo de los frames para un mismo cuerpo celeste.

3.6. Alineación

Si bien la secuencia de imágenes adquiridas fueron obtenidas en las mismas coordenadas del cielo, es común que se encuentren desalineadas entre sí debido a errores mecánicos de la montura y el telescopio. Además, las estrellas se mueven respecto al telescopio a causa de la rotación de la Tierra, y aunque la montura haga el trabajo de corregir este efecto a medida que pasa el tiempo, éste no es perfecto. El resultado es que las sucesivas imágenes se vean desplazadas unos píxeles en X y en Y o incluso un poco rotadas una respecto de la otra. La escala de las imágenes no varía porque la distancia focal del telescopio es fija.

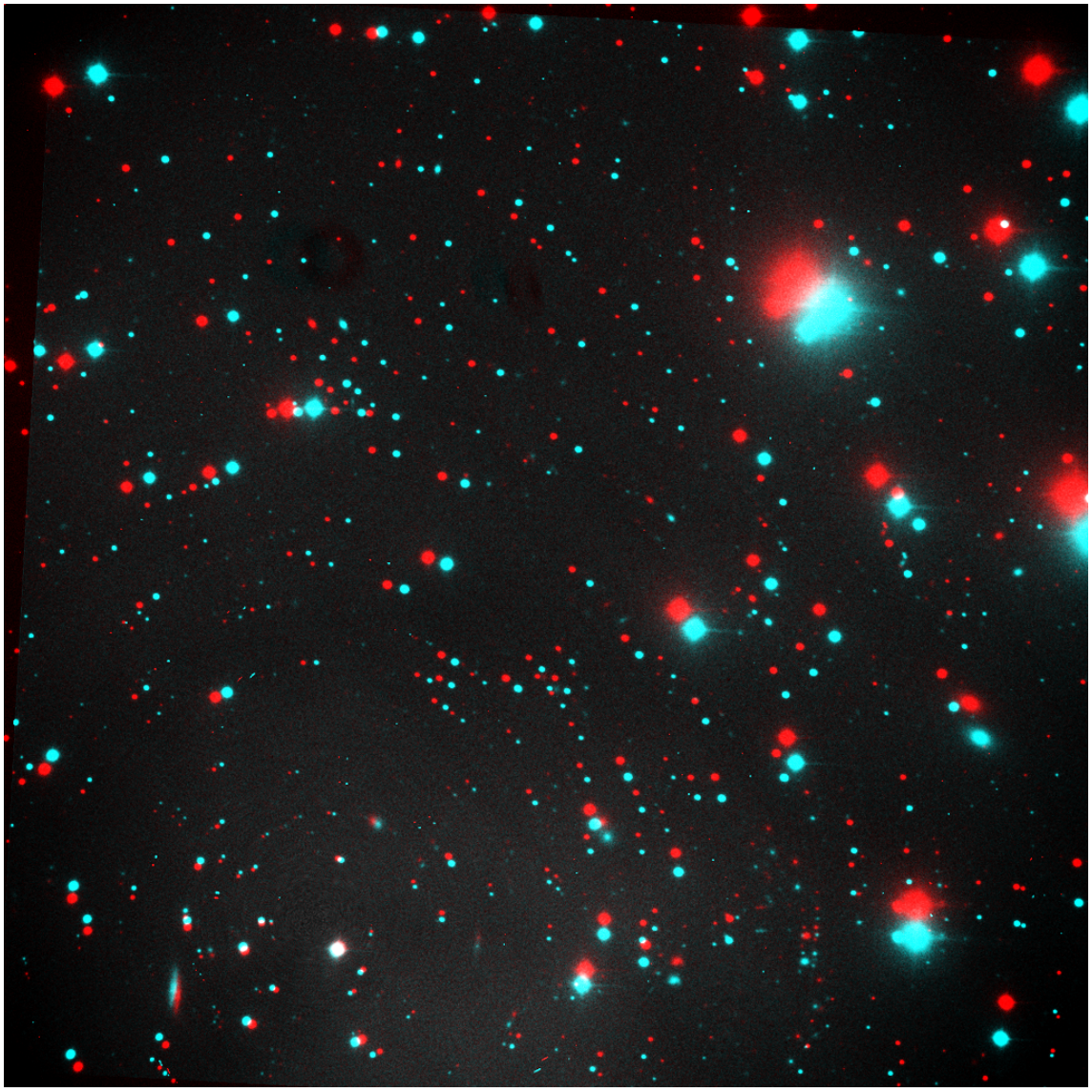


Fig. 3.19: Superposición de 2 imágenes consecutivas de una secuencia que se encuentran desalineadas.

En la figura 3.19 se muestran dos imágenes consecutivas de la secuencia su-

perpuestas y se observa la desalineación de las mismas.

Dado que la desalineación de las imágenes se da en 3 grados de libertad (desplazamiento en X, en Y y rotación), existe una transformación rígida que las relaciona. Una transformación rígida está dada por una matriz de rotación R ($R \in \mathbb{R}^{2\times 2}$ ortogonal con $det(R) = 1$) y una traslación $t \in \mathbb{R}^2$. Si tomamos la primera imagen como referencia y conocemos la transformación rígida de cada imagen sucesiva respecto de ésta, sabemos que el pixel en (x, y) de la imagen corresponde con el pixel $T(x, y) = R(x, y) + t$ de la primera; o, que la posición de una estrella situada en (x, y) corresponde a una la posición $T(x, y)$ en la primera imagen.

En coordenadas homogéneas representamos el punto (x, y) como $(x, y, 1)$, y podemos representar la transformación como una matriz:

$$T = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

donde θ es el ángulo de rotación y t_x, t_y la traslación. La transformación de un punto (x, y) es ahora

$$T \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

El problema de la alineación consiste entonces en encontrar las transformaciones de cada imagen para llevarlas al mismo sistema de referencia. Para lograr esto vamos a usar las fuentes extraídas anteriormente y buscar correspondencias entre features de las imágenes.

3.6.1. ICP

El método ICP (Iterative Closest Point) [13] supone que los features no se encuentran muy lejos de su posición original, cosa que es correcto asumir en nuestro caso porque queremos corregir errores de alineación debidos a perturbaciones en el movimiento del telescopio.

Supongamos que queremos alinear la imagen A con la B. Entonces buscamos la transformación que lleve a cada punto en B a su correspondiente en A. Dado que las coordenadas de las estrellas están fijas a lo largo del tiempo, vamos a buscar una transformación que haga coincidir la mayor cantidad de pares de features. Es decir, que alinie la mayor cantidad de estrellas entre las dos imágenes.

La forma de operar es suponer que cada feature de la imagen B corresponde con el feature más cercano a éste en la imagen A (como se observa en la figura 3.20), y buscar la transformación que mejor ajuste a todas las correspondencias; y repetir este proceso hasta que se estabilice.

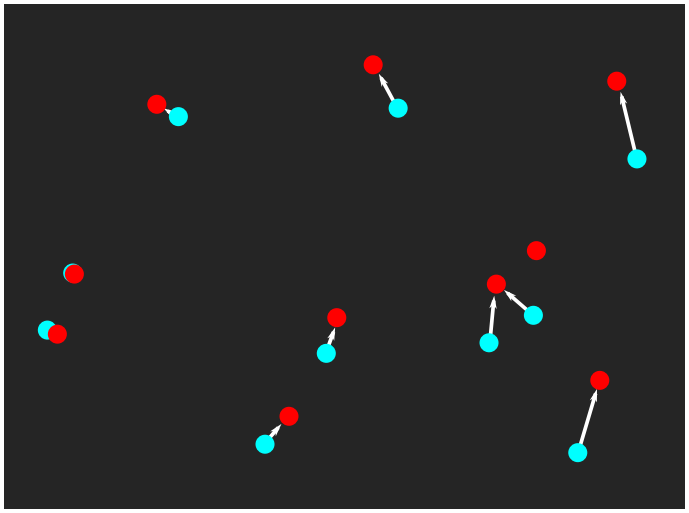


Fig. 3.20: Correspondencias usadas en el algoritmo ICP

En cada iteración se determina una correspondencia entre features de las 2 imágenes y se usa el método RANSAC (Random Sample and Consensus [14]) para encontrar la transformación.

3.6.2. RANSAC

En RANSAC [14], para cada punto (x, y) de la fuente origen tenemos un punto (x', y') en la segunda lista. La transformación debería ser la que mejor ajusta a todos los pares, es decir, tal que para la mayor cantidad de puntos, $T(x, y) \simeq (x', y')$. Al ser una transformación de 3 grados de libertad, este sistema está siempre sobredeterminado.

El algoritmo funciona de la siguiente manera:

Tomamos 2 puntos (a y b) al azar y sus correspondencias (a' y b'). Buscamos que estos dos puntos no estén demasiado cerca ya que el algoritmo es inestable dado un pequeño error en los las posiciones de éstos. Luego con estos dos pares podemos calcular el ángulo θ y la traslación t que los relaciona (ver figura 3.21).

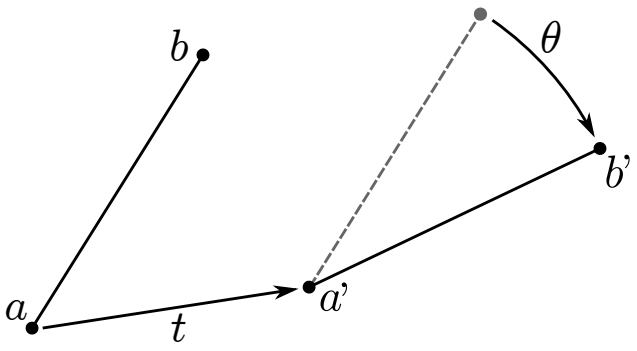


Fig. 3.21: Vector de traslación t y ángulo θ que relaciona a (a, b) con (a', b').

Con esto podemos construir la matriz de transformación

$$T = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

y la aplicamos al resto de los puntos.

Luego contamos cuántas correspondencias alinea esta transformación y el error obtenido. Es decir, para cuántos puntos p se cumple que $T(p) \simeq p'$. El error de alineación lo tomamos como la mediana de las distancias entre los $T(p)$ y p'.

Iteramos desde la elección de 2 puntos al azar y generamos nuevas transformaciones para luego quedarnos con la que minimiza el error, es decir, con la que alinea los features con más exactitud.

Para alinear la secuencia completa alineamos las primeras dos imágenes, luego la 2da con la 3era y así sucesivamente. Es más conveniente alinear cada imagen con la anterior y no cada una respecto de la primera porque de esta forma sólo corregimos por el error en un intervalo acotado de tiempo y no el error acumulado. Luego calculamos la transformación absoluta de la imagen I_N respecto de la primera acumulando las transformaciones desde la primera hasta la I_N, es decir:

$$T_N = T_{N,1} = T_{2,1} \cdot T_{3,2} \cdot \cdots \cdot T_{N,N-1},$$

donde $T_{i,j}$ es la matriz de transformación que alinea la imagen I_i con la I_j.

Si aplicamos la transformación a los features de cada imagen obtenemos las posiciones de las fuentes luminosas de cada una en el mismo sistema de referencia. Es decir que una estrella que aparece siempre en las mismas coordenadas RA y DEC a lo largo del tiempo, lo vamos a reconocer como un feature en coordenadas X e Y fijas a lo largo de la secuencia.

3.7. Detección de objetos móviles

El objetivo de este trabajo es la detección de objetos móviles. En los capítulos anteriores mostramos formas de extraer fuentes luminosas y alinear las secuencias de imágenes, que son pasos esenciales para la detección. Ahora veremos cómo encontrar fuentes que se mueven a lo largo de la secuencia.

Como mostramos en los capítulos de Adquisición y Reducción, los objetos móviles que nos interesa encontrar, tienen un perfil parecido al de las estrellas, y por lo tanto deberían ser reconocidos como features en cada imagen (o en algunas imágenes). Los objetos que buscamos se desplazan a velocidad constante, y dado que el campo de visión del telescopio es chico (mucha distancia focal), podemos considerar que la trayectoria es una línea recta.

Es decir, buscamos una sucesión de features que sean colineales (todos sobre la misma recta) y que estén distribuidos en intervalos proporcionales a los intervalos de tiempos de los frames. Al moverse a velocidad constante, la distancia

que recorre entre dos frames es proporcional a la diferencia de tiempos de esos frames.

Si pensamos a los features como puntos en un espacio tridimensional con coordenadas (x, y, tiempo), lo que buscamos es una sucesión de features colineales en este espacio (ver figura 3.22).

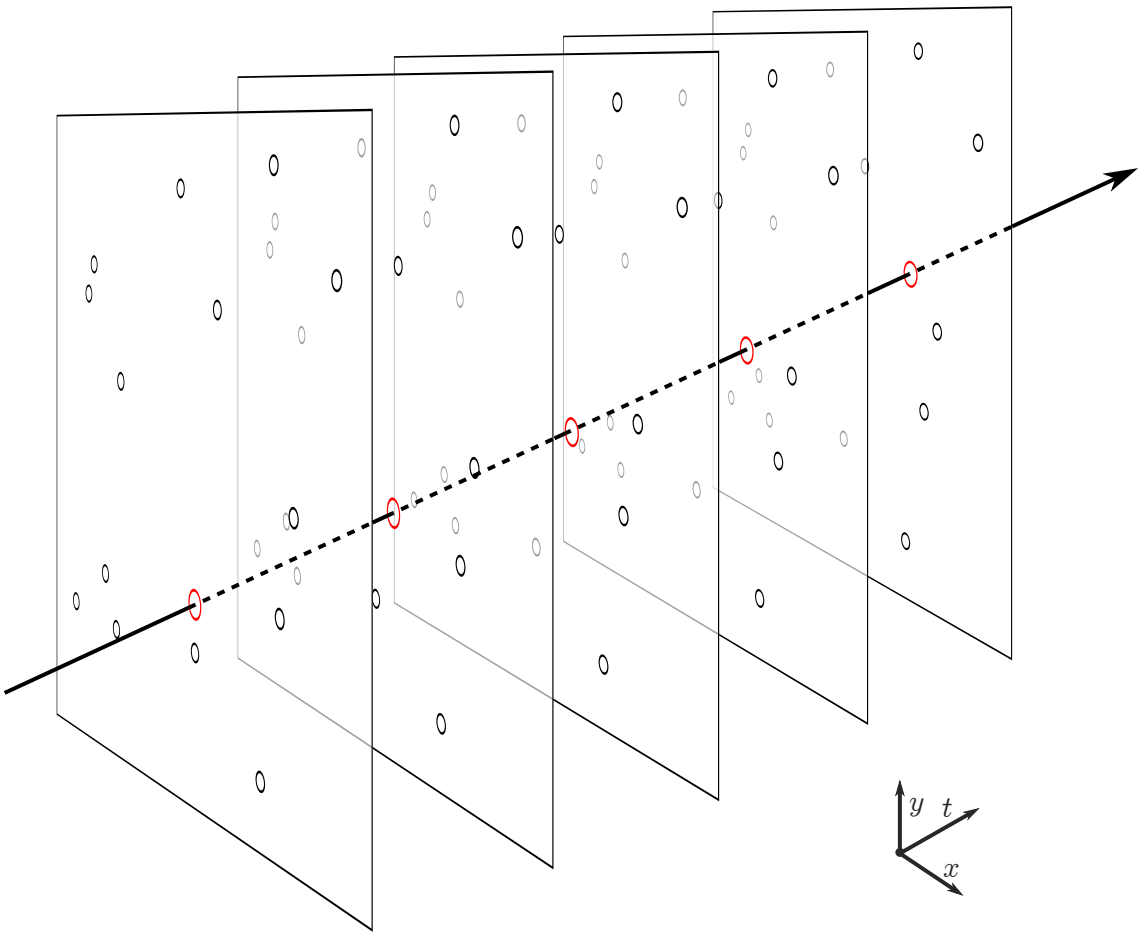


Fig. 3.22: Conjunto de features colineales en tiempo y espacio

Lo primero que hacemos es descartar los features colineales que no se mueven ni en X ni en Y, que son las estrellas y astros fijos en el cielo. Para esto ejecutamos el siguiente algoritmo:

luego, los features que tienen `features[f]` > 1 son objetos que aparecen en el mismo lugar más de una vez a lo largo del tiempo, y por lo tanto lo consideramos como un objeto fijo. Los demás features (que tiene `features[f]` $= 1$) son los que aparecen en ese lugar una sola vez en la secuencia, y por lo tanto son posibles candidatos de ser objetos que aparecen en distintos lugares a lo largo del tiempo y formar una trayectoria.

Además de un posible objeto móvil, entre los features de la secuencia suele haber ruido (features que aparecen en un frame, por ejemplo rayos cósmicos detectados), por lo que el objetivo ahora se reduce a encontrar una sucesión de

Algorithm 1 Features correspondientes a astros fijos.

```
features ← ∅
for all frame do
    for all feature f in frame do
        if ∃f′ ∈ features : f′_x ≃ f_x and f′_y ≃ f_y then
            features[f′]++
        else
            features[f] = 1
        end if
    end for
end for
```

más de 3 features entre los restantes que formen una trayectoria. También puede haber múltiples objetos móviles en la misma secuencia y queremos detectarlos a todos.

Además tenemos que tener en cuenta que posiblemente detectemos el objeto sólo en un subconjunto de los frames, por lo que tenemos que buscar colinealidades en diferentes subconjuntos de frames, aunque no hayamos detectado el feature entremedio.

Para lograr esto elegimos de a 3 frames al azar (f1, f2, f3) y buscamos tríos de features colineales en (x, y, tiempo). Una posibilidad es probar todas las combinaciones, lo que tendría una complejidad de $n_{\text{f1}} \times n_{\text{f2}} \times n_{\text{f3}}$. Sin embargo podemos hacerlo más eficiente, con complejidad $n_{\text{f1}} \times n_{\text{f2}} + n_{\text{f3}} \times log(n_{\text{f1}} \times n_{\text{f2}})$ de la siguiente manera:

Para cada par de features (p1, p2) del frame f1 y f2 ($n_{\text{f1}} \times n_{\text{f2}}$ combinaciones) calculamos el vector de desplazamiento p1 $\rightarrow$ p2 normalizado en el tiempo $t_{\text{f2}} - t_{\text{f1}}$ y estimamos dónde debería estar el feature del trío en el frame f3 para tiempo t_{f3}. Luego comparamos los features de f3 con las $n_{\text{f1}} \times n_{\text{f2}}$ posiciones calculadas y evaluamos cuáles completan un trío colineal. Si existe un feature p3 en f3 que es muy cercano a la posición estimada de p1 $\rightarrow$ p2 en el tiempo t_{f3}, entonces p1, p2 y p3 son colineales en (x, y, tiempo).

Una vez que tenemos estas posibles trazas de 3 puntos calculamos la recta que las une y evaluamos los frames restantes para buscar si tienen features sobre esta recta. Idealmente, si el objeto móvil fue detectado en todos los frames, vamos a encontrar un feature en cada frame que satisface la ecuación de la recta (con algún error) y habremos encontrado las sucesivas posiciones del objeto en cada frame. Como es posible que no hayamos detectado el objeto en todos los frames, nos alcanza con verificar que en total hay 4 puntos detectados en la recta. 3 puntos tienen una considerable probabilidad de ser ruido alineados por casualidad, y si pedimos 5 puntos alineados podemos perder casos positivos (por ejemplo si sólo tenemos 5 imágenes en la secuencia y en una fallo la detección).

Repetimos proceso con distintas ternas de frames al azar en cada iteración

para ir encontrando nuevas trazas. Cada vez que hallamos una verificamos si es una nueva o si tiene superposición con otra ya encontrada. Es posible que cuando hallamos una traza no encontremos todos los features que la componen, pero luego en otra iteración encontramos otra traza que comparte features con la primera, y por lo tanto las unimos (ver pseudocódigo 2).

Algorithm 2 Pseudocódigo para encontrar trazas de features en las imágenes.

```
trazas ← ∅
for N = #Frames do                                   ▷ Repetimos N veces
    f1, f2, f3 ← random.sample(Frames, 3)
    factor ← (t_f3 − t_f2) / (t_f2 − tf1)
    triplets ← get_linear_triplets(features_f1, features_f2, features_f3, factor)
    for all triplet ∈ triplets do
        recta ← AjustarRecta(triplet)
        traza ← {triplet_1, triplet_2, triplet_3}   ▷ Conjunto de features sobre la recta.
        ▷ Se agregan a la traza los features que cumplen la ecuación de la recta.
        for all frame ∈ Frames\ {f1, f2, f3} do
            for all feature ∈ frame do
                if feature ∝ recta then
                    traza ← traza ∪ {feature}
                end if
            end for
        end for
        if ∃traza′ ∈ trazas : traza ≅ traza′ then
            merge(traza, traza′)
        else
            trazas ← trazas ∪ {traza}
        end if
    end for
end for
```

Al finalizar obtenemos conjuntos de features que están alineados en espacio y tiempo, os ea que corresponden a objetos que se mueven en línea recta y a velocidad constante. Lo próximo que hacemos es calcular la recta que mejor ajusta a esta traza de features, la cual nos permite saber la posición y velocidad estimadas del objeto. Para esto usamos la técnica de cuadrados mínimos, ya que es un sistema sobredeterminado, como muestra el siguiente pseudocódigo:

Con esto obtenemos la posición del objeto en (X, Y) para algún tiempo t_0 y su velocidad de desplazamiento en $(\Delta X, \Delta Y)$ por unidad de tiempo. Notar que con esta información también podemos rellenar la posición estimada del objeto para los frames en los que no lo detectamos.

Algorithm 3 Ajuste por cuadrados mínimos de una traza

▷ Buscamos la solución a $[x|1][t] = 0$
▷ (el plano paralelo al eje Y que mejor ajusta los datos).
$m_{xt}, c_{xt} \leftarrow \text{LeastSquares}([x|1], t)$
▷ Buscamos la solución a $[y|1][t] = 0$
▷ (el plano paralelo al eje X que mejor ajusta los datos).
$m_{yt}, c_{yt} \leftarrow \text{LeastSquares}([y|1], t)$
▷ Calculamos la posición inicial p_0 (para t_0) y el vector velocidad $v = [\frac{\Delta X}{\Delta t}, \frac{\Delta Y}{\Delta t}]$
$p_0 \leftarrow [\frac{t_0 - c_{xt}}{m_{xt}}, \frac{t_0 - c_{yt}}{m_{yt}}]$
$v \leftarrow [\frac{1}{m_{xt}}, \frac{1}{m_{yt}}]$

3.8. Astrometría

En las secciones anteriores mostramos cómo detectar objetos móviles en la secuencia de imágenes, y obtuvimos las posiciones y velocidades de los objetos en píxeles. El objetivo de esta sección es mostrar cómo usamos la información de las imágenes para convertir estas coordenadas en píxeles a coordenadas del cielo (RA, DEC).

En la parte de adquisición vimos que tenemos información aproximada de las coordenadas de las imágenes, pero es poco precisa porque está dada por la montura del telescopio y tiene errores mecánicos. Para tener una buena estimación de la posición del objeto en el cielo necesitamos una matriz de transformación (CD) de píxeles a coordenadas RA, DEC de más precisión.

Una forma de hacer esto es usar catálogos existentes de las estrellas del cielo y alinearlas con las estrellas detectadas en las imágenes. Se puede usar una interfaz web (o repositorio local) de un catálogo conocido como el USNO-B1 y buscar la correspondencia entre estrellas y features detectados, y luego encontrar la transformación de forma parecida a la descripta en la sección 3.6, pero esta vez también teniendo en cuenta la escala. En este caso la transformación sería una similaridad, y también alcanza con dos pares de puntos correspondientes para calcularla, por lo que podríamos usar el mismo método de RANSAC [14]. El mayor desafío en este caso es encontrar las correspondencias entre puntos. Esto se debe por un lado a que la estimación de las coordenadas iniciales puede ser muy imprecisa (los pares de puntos de las correspondencias podrían estar muy alejados entre sí, y además hay que buscar en un área mayor del catálogo), y por otro lado los parámetros de luminosidad pueden no corresponderse, por las distintas formas que existen de medirlos y porque dependen de variables del momento de la observación que pueden no estar disponibles (condiciones atmosféricas, masa de aire, etc).

Un método más robusto, sin embargo, que no necesita estimación inicial de las coordenadas y que depende sólo de las posiciones en píxeles de algunos features para la alineación con el catálogo es el propuesto por Astrometry.net [12].

3.8.1. Astrometry.net

Astrometry.net [12] un servicio que aporta datos astrométricos de imágenes. Dada una imagen o una lista de features (posiciones X, Y), los compara con los astros en catálogos previamente indexados y encuentra la ubicación exacta del input en el cielo. No requiere una estimación previa de las coordenadas porque tiene un método muy eficiente de buscar patrones en todo el catálogo.

El método se basa en detectar patrones que son asterismos de 4 cuerpos y calcular un descriptor que es invariante a la posición, rotación y escala que luego es comparado con los descriptores de asterismos del catálogo. Lo que hace al método tan eficiente es que los descriptores del catálogo se encuentran previamente indexados y entonces la búsqueda es una búsqueda por cercanía en el espacio de las dimensiones del descriptor.

El descriptor funciona de la siguiente manera:

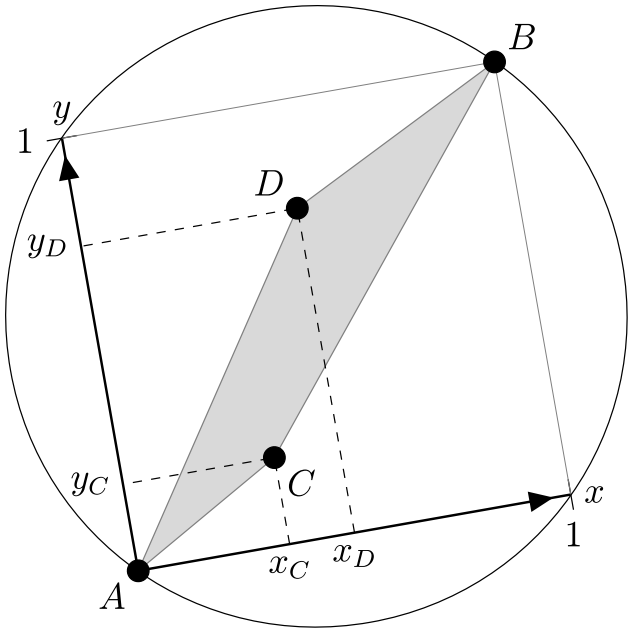


Fig. 3.23: Asterismo con el sistema de coordenadas usado por el descriptor

Los dos features que se encuentran más alejados entre sí los llamamos A y B, y los otros dos son C y D (ver figura 3.23). Definimos un eje de coordenadas que tiene como origen al punto A y el $(1, 1)$ en el punto B. En este sistema de coordenadas la posición de C es (x_C, y_C) y la de D es (x_D, y_D) y se encuentran dentro del círculo de la figura. Luego el descriptor de este asterismo es el vector (x_C, y_C, x_D, y_D).

El par A, B y el par C, D son en principio intercambiables entre sí (A↔B, C↔D), pero se rompe esta simetría requiriendo que $x_C \leq x_D$ y que $x_C + x_D \leq 1$.

El descriptor propuesto es invariante a la posición, rotación y escala del asterismo (depende sólo de la posición relativa entre ellos) y por lo tanto es igual para las coordenadas en píxeles como en RA, DEC; haciendo comparables los patrones

de las imágenes con los del catálogo. Además tiene otras propiedades favorables, por ejemplo que pequeñas perturbaciones en las posiciones relativas sólo producen pequeñas perturbaciones en el descriptor, haciéndolo robusto a errores de medición, y también, si las estrellas se encuentran uniformemente distribuidas, los descriptores se encuentran uniformemente distribuidos en (y haciendo buen uso de) su espacio 4-dimensional.

Lo que le da la eficacia al método es la correcta indexación del catálogo para una búsqueda eficiente. Computar los descriptores de todos los posibles asterismos de 4 estrellas sería imposible dada la gran cantidad de combinaciones existentes (n^4). Tampoco es una buena idea computar los descriptores de todos los asterismos de la imagen, ya que podría haber miles de features.

Para alinear imágenes en cualquier parte del cielo, es deseable que el índice contenga asterismos uniformemente distribuidos en todo el catálogo. Además las imágenes pueden ser de diferentes distancias focales, por lo que se necesitarían descriptores de diferentes escalas. También hay que tener en cuenta que las imágenes tienen más probabilidades de contener los astros más brillantes, por eso éstos deberían ser elegidos con mayor probabilidad.

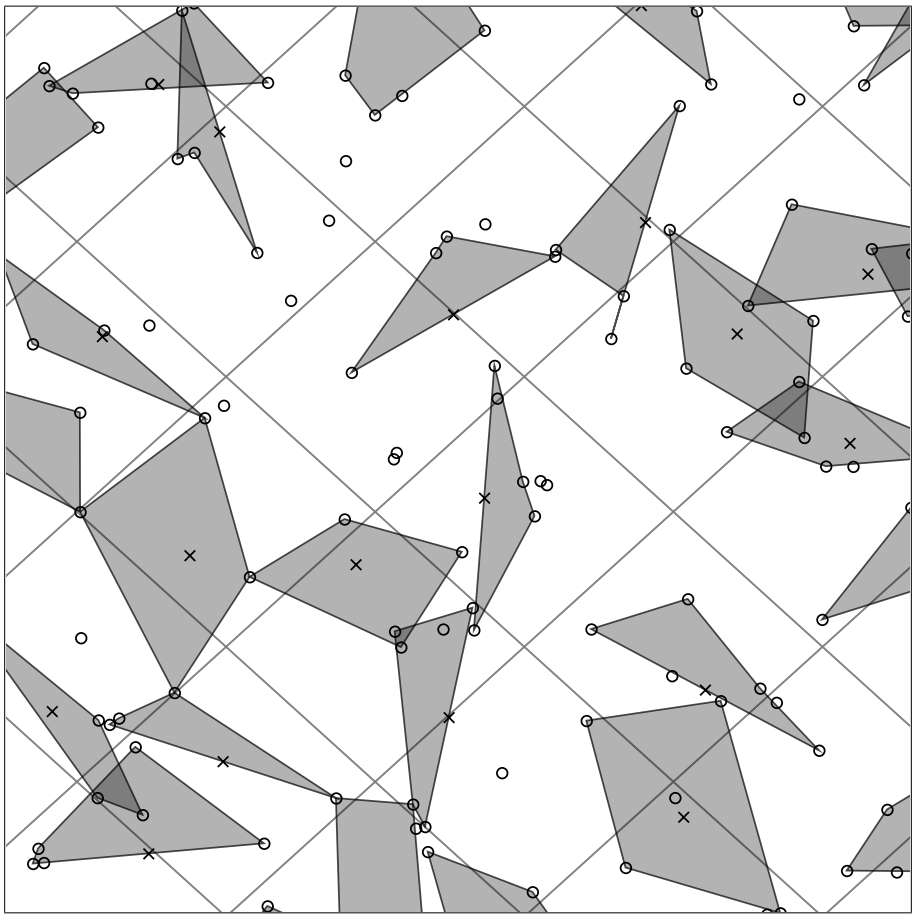


Fig. 3.24: Asterismos uniformemente distribuidos según una grilla.

Para lograr todas estas propiedades se categorizan las estrellas en celdas según una grilla y se buscan asterismos en cada celda de tamaño parejo (ver figura 3.24). De esta forma se tienen descriptores uniformemente distribuidos. Así mismo se construyen sub-índices con grillas de distinto tamaño para contemplar todas las escalas. De forma similar se buscan asterismos en la imagen dada para comparar con los índices.

Existen estructuras de datos y algoritmos eficientes para buscar descriptores cercanos a los encontrados en los índices. Cuando se establece una correspondencia (match) entre los descriptores se computa la transformación que relaciona los asterismos (que mapee los A y los B entre sí) y se comprueba si los demás features corresponden suficientemente bien con otros elementos del catálogo. Si es un buen match significa que la imagen se alinea muy bien con esa porción del cielo en el catálogo y se reporta la transformación.

Para usar el servicio Astrometry.net se puede usar la API pública a través de HTTP, o se puede instalar como una aplicación local (también requiere la instalación de los índices del catálogo). En la implementación presentada en este trabajo se optó por usar el servicio de la API via HTTP, o sea que el sistema requiere una conexión a internet, pero se podría fácilmente descargar la versión local de Astrometry.net o incluso instalarla a nivel de red local para ser usada por muchos clientes en el mismo entorno.

El servicio devuelve los datos de calibración en forma de metadatos que se pueden incluir directamente en el header de la imagen FITS analizada. Estos metadatos incluyen:

CRPIX1, CRPIX2:
Los valores de X y Y que se toman como referencia para la alineación, en general corresponden al centro de la imagen dada.

CRVAL1, CRVAL2:
Las coordenadas RA y DEC respectivamente que corresponden a la posición de referencia X, Y. (Las coordenadas del cielo del centro de la imagen).

CD:
La matriz de rotación y escala de la transformación.

$$\begin{pmatrix} \mathrm{CD1\_1} & \mathrm{CD1\_2} \\ \mathrm{CD2\_1} & \mathrm{CD2\_2} \end{pmatrix} = scale \cdot \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

$\mathrm{CD1\_1}$ es el incremento en RA por cada pixel en X,
$\mathrm{CD1\_2}$ es el incremento en DEC por cada pixel en X,
$\mathrm{CD2\_1}$ es el incremento en RA de por cada en Y,
$\mathrm{CD2\_2}$ es el incremento en DEC por cada pixel en Y.

Es decir, las coordenadas espaciales del pixel (x, y) serían:

$$\mathrm{RA} = (x - \mathrm{CRPIX1}) \times \mathrm{CD1\_1} + (y - \mathrm{CRPIX2}) \times \mathrm{CD2\_1} + \mathrm{CRVAL1}$$

$$\mathrm{DEC} = (y - \mathrm{CRPIX2}) \times \mathrm{CD2\_2} + (x - \mathrm{CRPIX1}) \times \mathrm{CD1\_2} + \mathrm{CRVAL2}$$

A partir de estos datos podemos construir la matriz de transformación para coordenadas homogéneas de la siguiente forma:

$$T = \begin{bmatrix} \mathrm{CD1\_1} & \mathrm{CD1\_2} & \mathrm{CRVAL1} \\ \mathrm{CD2\_1} & \mathrm{CD2\_2} & \mathrm{CRVAL2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\mathrm{CRPIX1} \\ 0 & 1 & -\mathrm{CRPIX2} \\ 0 & 0 & 1 \end{bmatrix}$$

De esta forma las coordenadas (RA, DEC) de un feature en las coordenadas (x, y) de la imagen se calculan como: $(\mathrm{RA}, \mathrm{DEC}) = T(x, y, 1)$

Con esto podemos convertir las coordenadas en píxeles del objeto móvil encontrado a coordenadas absolutas en el cielo: La posición inicial (para un t_0):

$$(\mathrm{RA}_0, \mathrm{DEC}_0) = T(x_0, y_0, 1)$$

La velocidad:

$$(\Delta\mathrm{RA}, \Delta\mathrm{DEC}) = T(\Delta X, \Delta Y, 0)$$

Estos datos sirven para reportar el objeto encontrado y compararlo con catálogos y verificar si es un descubrimiento nuevo (y en ese caso reportarlo), y también para seguirlo con el telescopio y encontrarlo en algún tiempo futuro.

3.9. Control de telescopio

Si bien necesitamos controlar el telescopio para adquirir las imágenes de la secuencia inicial, presentamos la solución a este problema en esta etapa del desarrollo porque es un problema más relevante cuando tenemos un objetivo móvil a seguir. Exploramos ahora cómo es el control de la montura para mover el telescopio y adquirir imágenes.

La montura se ubica entre el telescopio y el trípode y tiene 2 motores que permiten 2 grados de libertad para el movimiento correspondientes a las coordenadas RA y DEC. La que usamos en este trabajo es una Temma Takahashi que controlamos a través del driver INDI, pero la arquitectura propuesta permite extender el software para nuevas monturas con diferentes drivers fácilmente.

Al encender la montura hay que calibrarla informando las coordenadas en las que se encuentra inicialmente. Para esto hay que alinear el telescopio apuntándolo a un lugar en el cielo con coordenadas conocidas y luego setear estar coordenadas en el driver de la montura. A partir de ese momento la montura va ajustando la coordenada RA a medida que pasa el tiempo para compensar por la rotación de

la Tierra. Luego podemos enviar coordenadas (RA, DEC) al driver y la montura se moverá al lugar deseado.

Teniendo la posición inicial (para algún tiempo t_0) del objeto móvil y su vector de velocidad $(\Delta\mathrm{RA}, \Delta\mathrm{DEC})$ podemos calcular su posición para algún tiempo futuro y usar las coordenadas para seguir al objeto.

Las coordenadas del objeto para un tiempo t son:

$$(\mathrm{RA}, \mathrm{DEC})_t = (\mathrm{RA}_0, \mathrm{DEC}_0) + (\Delta\mathrm{RA}, \Delta\mathrm{DEC}) \cdot (t - t_0)$$

Obviamente hay que comprobar que el objeto se encuentre sobre el horizonte en ese tiempo antes de apuntar con el telescopio.

Así, se puede adquirir una nueva secuencia de imágenes y repetir todo el proceso descrito en este desarrollo y mejorar la estimación de desplazamiento del objeto.

4. DETALLES IMPLEMENTATIVOS

En este capítulo mostraremos en detalle particularidades de la implementación del software. En general fue diseñado de forma que sea fácil extenderlo y modificarlo en el futuro o para usos un poco distintos. Para esto está separado en módulos con funciones específicas con partes intercambiables.

4.1. Hardware

El hardware en el que fue realizada la experimentación y con el que se obtuvieron los resultados de este trabajo son:

4.1.1. Montura robótica

La montura que controla el telescopio es una Temma EM-200.

Fig. 4.1: Montura robótica Temma que se ubica entre el telescopio y el trípode.

4.1.2. CCD

La cámara utilizada es una Apogee Alta F16 que tiene una resolución de 4096x4096 píxeles en un sensor de 36.8 $\times$ 36.8mm (tamaño de pixel de 9 micrones).

Fig. 4.2: Apogee Alta F16

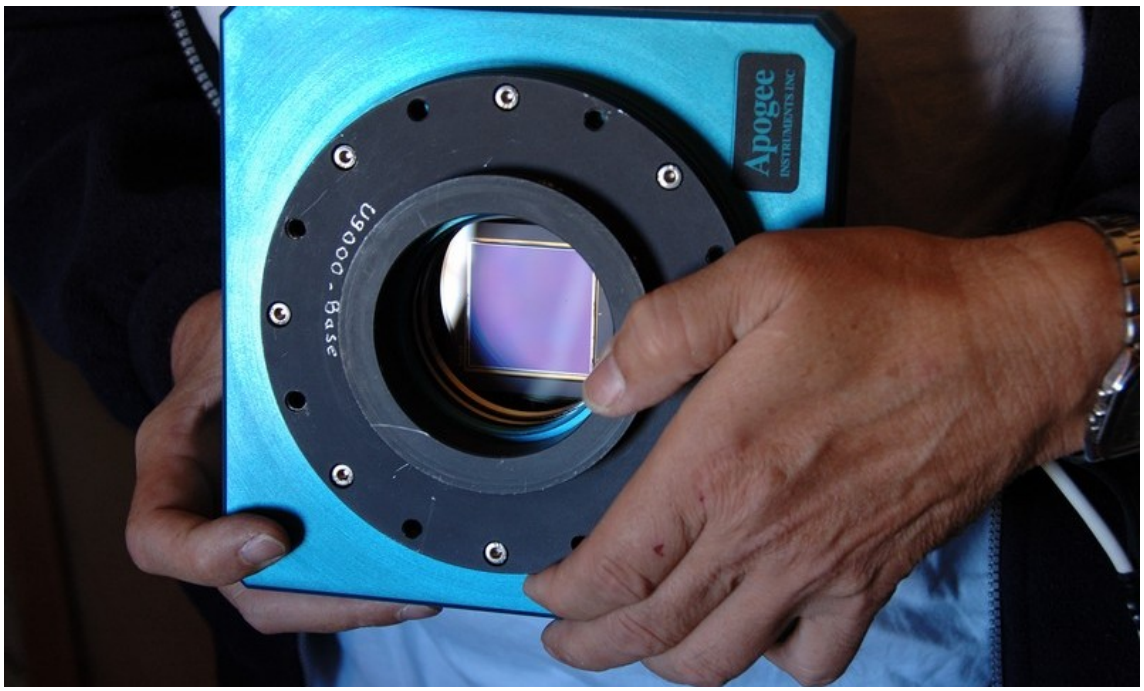


Fig. 4.3: Apogee Alta F16

4.1.3. Telescopio

El telescopio usado es el Takahashi Mewlon-210, que tiene una distancia focal efectiva de 2415mm y un cociente focal efectivo de 1:11.5.

Fig. 4.4: Telescopio Takahashi Mewlon-210 con el que se realizaron los experimentos.

4.2. Decisiones de diseño

Como puede verse en las secciones del método, el workflow del procesamiento es de tipo pipeline. Es decir, se ejecuta en etapas secuenciales y cada etapa requiere que la etapa anterior esté completa.

El pipeline propuesto del procesamiento desde la adquisición hasta el seguimiento es el siguiente:

Control del telescopio:
Apuntar el telescopio hacia las coordenadas en las que se quiere hacer la observación.

Adquisición:
Adquirir una secuencia de imágenes en un intervalo de tiempo

Reducción:
Extraer las posiciones de las fuentes de las imágenes (features).

Fotometría:
Análisis cuantitativo del flujo de luz de cada fuente y estimación de la magnitud.

Alineación:
Alineación de la secuencia de imágenes para que tengan un sistema de coordenadas homogéneo.

Detección de objetos móviles:
Búsqueda de objetos que se mueven linealmente a lo largo de la secuencia.

Control del telescopio:
Nuevamente se controla el telescopio para hacer seguimiento del objeto encontrado (si lo hay).

Observar como cada etapa procesa los datos en función de los resultados obtenidos de la etapa anterior.

Teniendo en cuenta la forma en la que se lleva a cabo el procesamiento, se diseñó el software con módulos separados que pueden ser usados independientemente entre sí, siempre y cuando esté disponible el resultado del proceso anterior.

Para esto se implementa un sistema de caché, es decir que cada vez que un módulo procesa un set de datos, guarda el resultado en archivos permanentes. De esta forma, cada vez que un proceso requiere datos de un proceso anterior se invoca un procedimiento que levanta los datos de caché si fueron computados previamente, o obtiene el resultado en el momento. De esta forma podemos ejecutar el módulo de detección de objetos móviles sobre una secuencia, y automáticamente se invocan los módulos de reducción, alineación, etc. Además se puede volver a ejecutar este proceso con diferentes variables, y funcionaría muy rápido gracias a que los resultados intermedios se encuentran guardados.

A continuación se detallan decisiones de diseño particulares de la implementación.

4.2.1. Matching

Tanto en la detección de objetos móviles como en la alineación se habló de establecer correspondencias entre features. Esto consiste en encontrar para cada feature f, el feature más cercano a éste en un conjunto DST dado, ya sea en distancia euclídea como en similitud de los descriptores. En el caso del matching euclídeo se busca el feature f' en DST que esté a menos distancia del feature f, es decir, que minimice $(\mathrm{f}_x - \mathrm{f}'_x)^2 + (\mathrm{f}_y - \mathrm{f}'_y)^2$. En el caso del matching por descriptor se busca el $\mathrm{f}' \in \mathrm{DST}$ cuyo descriptor sea el más parecido al de f. La función de distancia entre descriptores es la misma que para el espacio euclídeo pero en las dimensiones del descriptor, es decir $\sum_i (\mathrm{f}_{x_i} - \mathrm{f}'_{x_i})^2$. En cualquiera de los 2 casos buscamos una correspondencia en el conjunto DST que minimice la función de distancia.

Si queremos encontrar las correspondencias (matches) entre un conjunto de features SRC y un conjunto DST, buscamos para cada feature en SRC su match en DST. Esto corresponde a una función de SRC $\rightarrow$ DST, en la que cada SRC tiene un DST correspondiente, y puede haber dos $\mathrm{f} \in \mathrm{SRC}$ que correspondan al mismo $\mathrm{f}' \in \mathrm{DST}$ (y puede haber $\mathrm{f}' \in \mathrm{DST}$ sin correspondencias).

En la práctica sólo consideramos matches "buenos" a los que tienen una distancia menor a un umbral de error ϵ entre f y f'. Es decir que además de la función de correspondencia de feature más cercano a cada uno, necesitamos obtener también la distancia de cada match para poder filtrar luego por el umbral.

Una forma naïve de calcular el matching entre dos conjuntos es por cada feature f $\in$ SRC, recorrer los features f' $\in$ DST y guardar el índice y la distancia entre f y f'. Esto tiene una complejidad de $\#\mathrm{SRC} \times \#\mathrm{DST}$ que no es usable en la práctica.

Para hacerlo de forma eficiente usamos la implementación de OpenCV de FLANN (Fast Approximate Nearest Neighbor Search [8]). Este es un método aproximado pero muy preciso de búsqueda de vecino más cercano que se basa en estructuras de datos de árboles kd-randomizados y k-media jerárquicos. Funciona creando estas estructuras de datos con los features de DST y luego puede buscar cada $\mathrm{f} \in \mathrm{SRC}$ de manera eficiente. En la práctica resulta muy eficiente y a pesar de ser aproximado resuelve perfectamente las necesidades de este trabajo.

4.3. Arquitectura del software

Como se mencionó anteriormente, el workflow de este sistema tiene forma de pipeline, es decir que cada parte procesa los datos suministrados por la parte anterior y entrega el resultado a la parte siguiente. Cada parte es responsable de una tarea específica y el procesamiento completo de los datos consiste en ejecutar cada tarea sucesivamente. Esto se refleja en la arquitectura del sistema de forma casi directa.

Cada tarea es llevada a cabo por un módulo diferente, que además son procesos independientes. Esto permite ejecutarlos de forma aislada (suministrándole

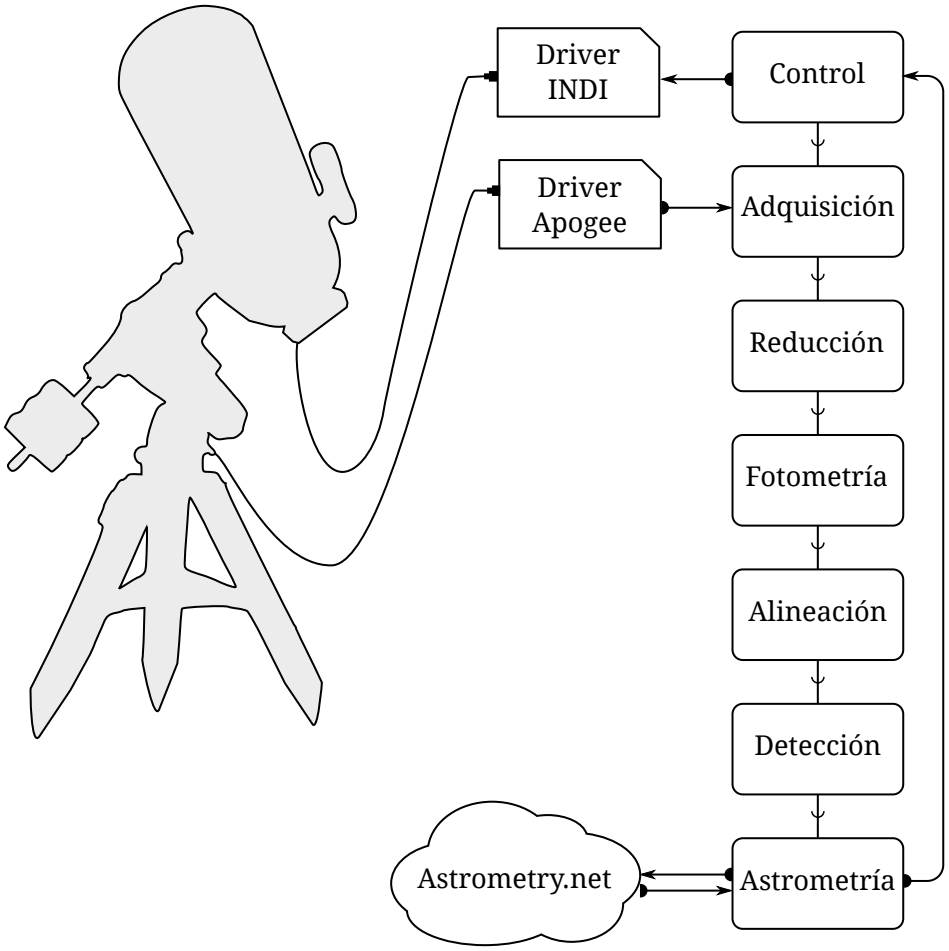


Fig. 4.5: Diagrama de la arquitectura del sistema.

los datos correctos). Además se puede ejecutar cada tarea en distintos procesadores o en paralelo (cuando termina la adquisición del primer batch se puede hacer la reducción al mismo tiempo que se hace la adquisición del segundo batch, etc). También tiene la ventaja de que se puede parar la ejecución entre cada tarea y resumir, o reprocesar sólo una parte sin tener que ejecutar todo desde el principio.

La comunicación con los drivers está diseñada en forma de "wrappers" de modo que es fácil agregar drivers para dispositivos distintos, implementando solamente la interfaz requerida.

En este trabajo usamos el driver de INDI para el control de la montura y el driver de Apogee para la cámara CCD. El procesamiento se lleva a cabo en una computadora conectada al telescopio y con conectividad a internet para obtener los datos astrométricos de Astrometry.net. También se pueden bajar los índices de los catálogos e instalar astrometry.net de forma local para prescindir de conexiones externas. (o, gracias al diseño del pipeline, obtener los datos de detección del workflow y ejecutar la tarea de astrometría en otra instancia).

4.4. Diseño de módulos/clases

Los módulos en los que esta divivido el software corresponden a cada una de las partes del pipeline de procesamiento: Control, Adquisición, Reducción (DAOPHOT y SExtractor), Fotometría (DAOPHOT y SExtractor), Alineación, Detección, Astrometría y además hay módulos auxiliares como la interfaz con el catálogo, el generador de imágenes sintéticas, y un módulo de visualización de imágenes y secuencias.

Dentro de estos módulos se distinguen las siguientes clases:

`FitsImage`**:**
: Encapsulamiento de las imágenes fits, con métodos para obtener datos y metadatos (Usa la libreria PyFITS)

`Sequence`**:**
: Representa una secuencia de imágenes y contiene básicamente una lista de `FitsImage`.

`AlignedSeq`**:**
: Extiende la `Sequence` y representa una secuencia alineada.

`Telescope`**:**
: Es la abstracción de las operaciones que se pueden realizar con un telescopio (CCD + montura).

`CCD`**:**
: Interfaz de un sensor CCD y las operaciones disponibles, siendo la más importante `expose()` que lleva a cabo la adquisición. Para agregar nuevos CCD simplemente se extiende esta clase y se implementan las operaciones requeridas.

`SynthCCD` **y** `ApogeeCCD`**:**
: Representan un sensor virtual que genera imágenes sintéticas y un sensor Apogee que es un wrapper de los drivers, respectivamente.

`Mount`**:**
: Interfaz de una montura robótica. El método importante es `moveto()` que mueve la montura a loas coordenadas dadas. Para agregar nuevas monturas se extiende esta clase.

`FakeMount` **y** `TemmaIndiMount`**:**
: Representan una montura virtual y la montura Temma respectivamente. `TemmaIndiMount` es un wrapper del driver de INDI para esa montura que implementa las operaciones de `Mount`.

`Display`**:**
: Encapsula la visualización de imágenes FITS.

`VideoWriter`:
: Generador de videos a partir de secuencias de imágenes.

`StarSynth`:
: Generador de imágenes sintéticas con parámetros de tamaño y ruido. Puede generar imágenes a partir de estrellas del catálogo o al azar.

En la figura 4.6 se muesta el diagrama de clases del software del sistema.

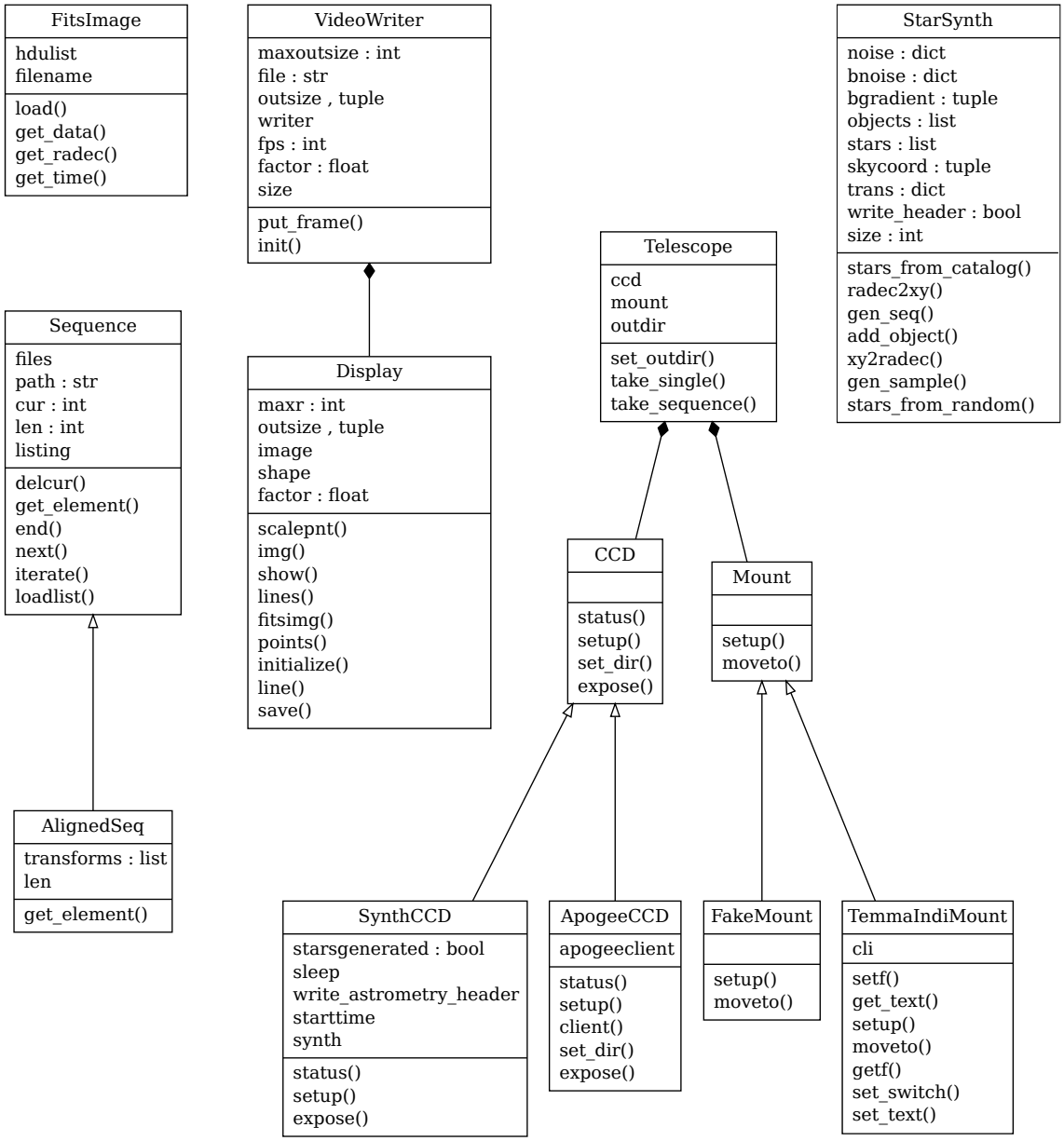


Fig. 4.6: Diagrama de las clases del software.

5. RESULTADOS

5.1. Reducción

En el proceso de reducción tenemos dos posibles métodos: DAOPHOT y SExtractor. En esta sección comparamos ambos métodos y analizamos ventajas y desventajas de cada uno.

Una métrica que nos interesa evaluar para determinar la eficacia de los métodos de reducción es el comportamiento a distintos niveles de ruido. Para esto generamos imágenes sintéticas a partir de la información de las estrellas que fueron obtenidas del catálogo USNO-B1. Por cada estrella se dibuja un perfil similar al gaussiano según la magnitud reportada en el catálogo, tal como aparecería en una imagen tomada por un CCD. A esto le incorporamos ruido de fondo con cierto nivel basal y cierta amplitud que simula el ruido de fondo típico encontrado en las imágenes adquiridas.

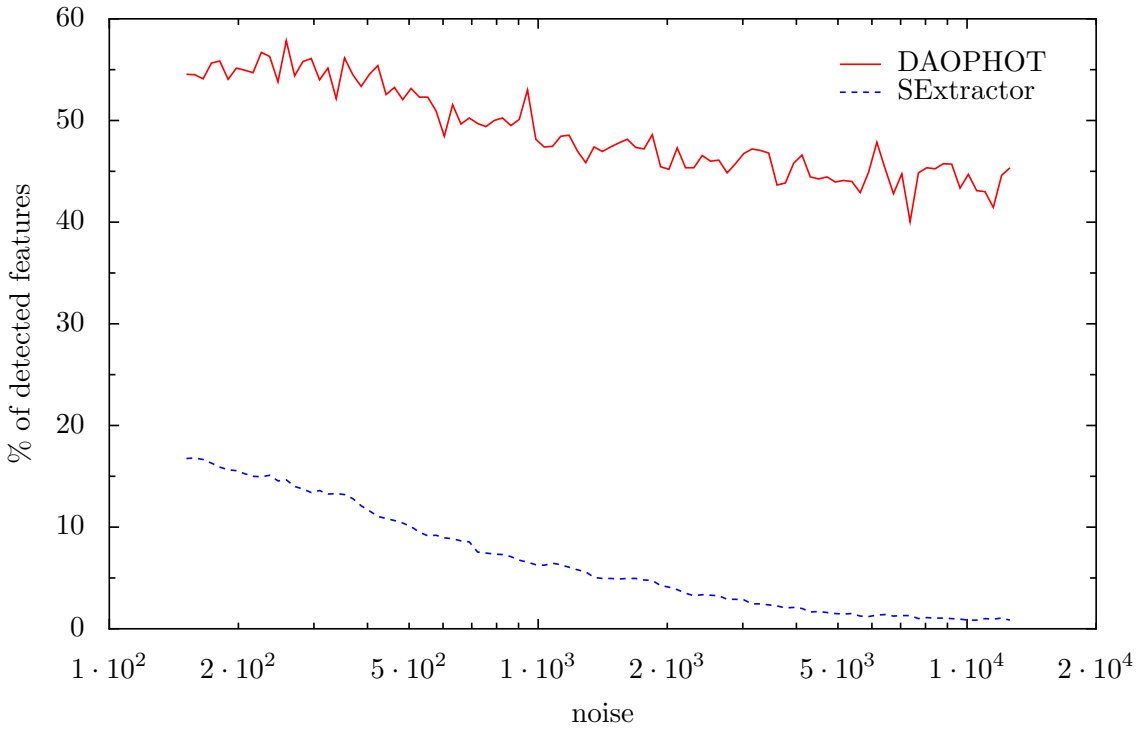


Fig. 5.1: Comparación de la cantidad de features detectados por SExtractor y DAOPHOT. En este experimento ejecutamos DAOPHOT con el parámetro `Hmin`=200.

En la figura 5.1 se muestra el porcentaje de estrellas detectadas en función del

nivel de ruido de fondo introducido. Dado que el FLUX (brillo) de las estrellas llega hasta órdenes de aproximadamente 10^5 o 10^6, a medida que aumenta el ruido hay estrellas que se pierden porque no son suficientemente brillantes.

Como se puede ver, en general el método de DAOPHOT encuentra más fuentes que SExtractor, pero esto depende de y es controlable por el parámetro `Hmin` (el nivel de threshold de la detección de máximos en la imagen filtrada). Ambos métodos son consistentes en la cantidad de estrellas encontradas para todos los valores de ruido que pueden darse en imágenes adquiridas, y por lo tanto son igualmente efectivos para la alineación donde se consideran sólo los features más brillantes. A efectos de la detección, en cambio, es preferible detectar objetos que tengan muy poco brillo por sobre el ruido de fondo ya que aumenta las chances de encontrar un objeto móvil aunque éste no sea muy brillante o las condiciones de observación no sean óptimas.

Otra métrica que nos interesa considerar el el tiempo que tarda cada método. Al no ser una aplicación en tiempo real se puede ser más permisivo con el tiempo de ejecución, pero un algoritmo más eficiente significa también que se requiere menos poder de procesamiento, y por ende se puede ejecutar sobre un procesador más barato como ser una notebook o incluso procesador embebido.

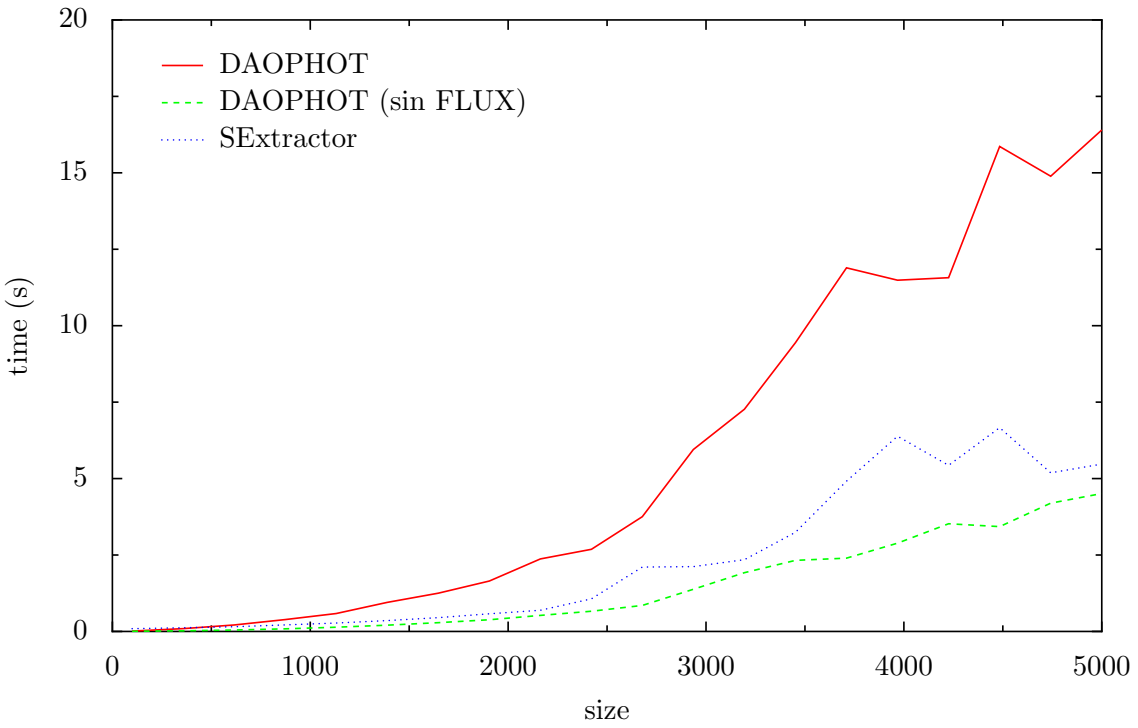


Fig. 5.2: Comparación de tiempos de la ejecución de los distintos métodos en función del tamaño de la imagen (size es el ancho y alto en pixels).

En el gráfico de la figura 5.2 se muestra el tiempo que lleva la reducción de datos con los distintos métodos para diferentes tamaños de la imagen original, con una cantidad fija de estrellas (300). En este caso comparamos 2 versiones de

DAOPHOT, que son con y sin la medición del FLUX (brillos) respectivamente. A efectos de encontrar los n features más brillantes no hace falta calcular exactamente el FLUX, ya que se puede ajustar la cantidad encontrada variando el threshold `Hmin` de la detección de máximos locales. Como es esperable, los tres métodos aumentan aproximadamente de forma cuadrática con el tamaño de la imagen (casi linealmente según la cantidad de pixels totales).

En la figura 5.3 graficamos el tiempo requerido por cada método en función de la cantidad de estrellas presentes en la imagen, con un tamaño fijo (1000x1000 pixels). Se observa claramente que mientras el tiempo de ejecución de SExtractor aumenta linealmente con la cantidad de features detectados, DAOPHOT tiene tiempos prácticamente constantes.

Esto se debe a que el procesamiento de DAOPHOT consiste solamente de filtros y operaciones directas sobre la imagen y no efectúa operaciones particulares con cada objeto. La ventaja de esto se hace visible en campos de estrellas muy densos. En este caso DAOPHOT puede operar mucho más rápido que SExtractor. También se observa que DAOPHOT con el cómputo de flujos tarda consistentemente más que el mismo método sin este cómputo, independiente de la cantidad de estrellas.

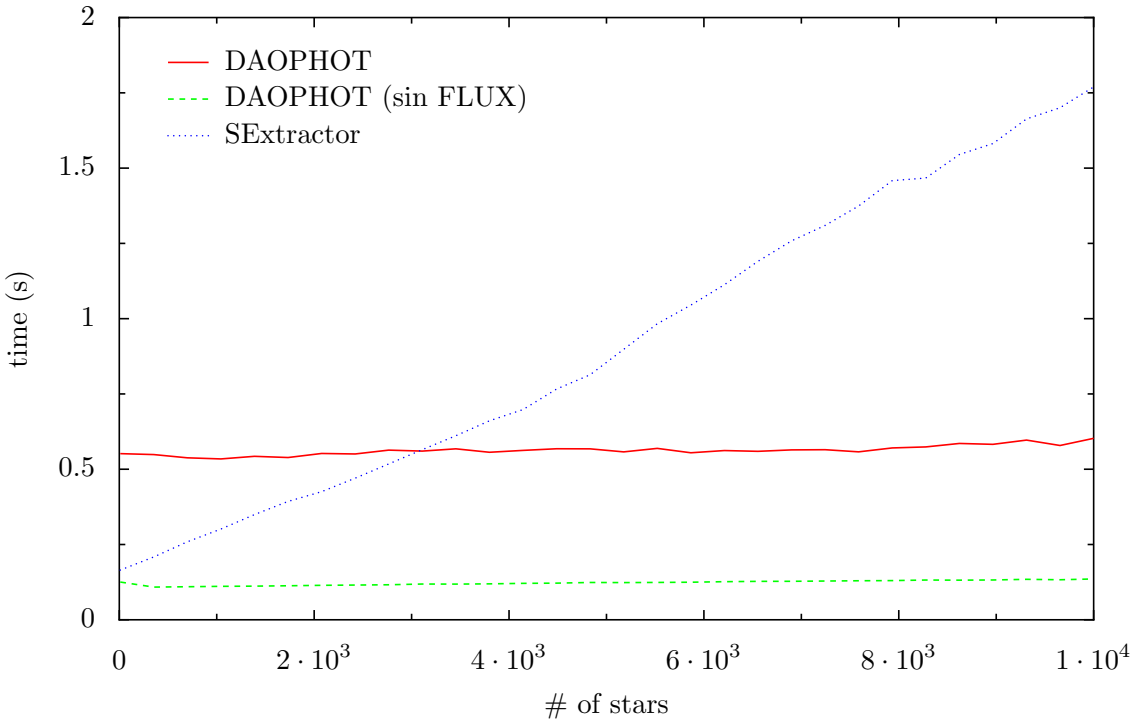


Fig. 5.3: Comparación de tiempos de la ejecución de los distintos métodos en función de la cantidad de features

Combinando los dos resultados de tiempos recién presentados, se muestra a continuación un gráfico de los tiempos de ejecución de los métodos para distintos tamaños de imágenes de entrada, pero esta vez con un una densidad de estrellas

constante. Es decir, a medida que aumenta el tamaño de la imagen aumenta la cantidad de estrellas, pero la relación de cantidad de estrellas por unidad de área en píxeles es fija (~ 20 estrellas por cada 100×100 pixels, que es aproximadamente la densidad de estrellas en imágenes adquiridas).

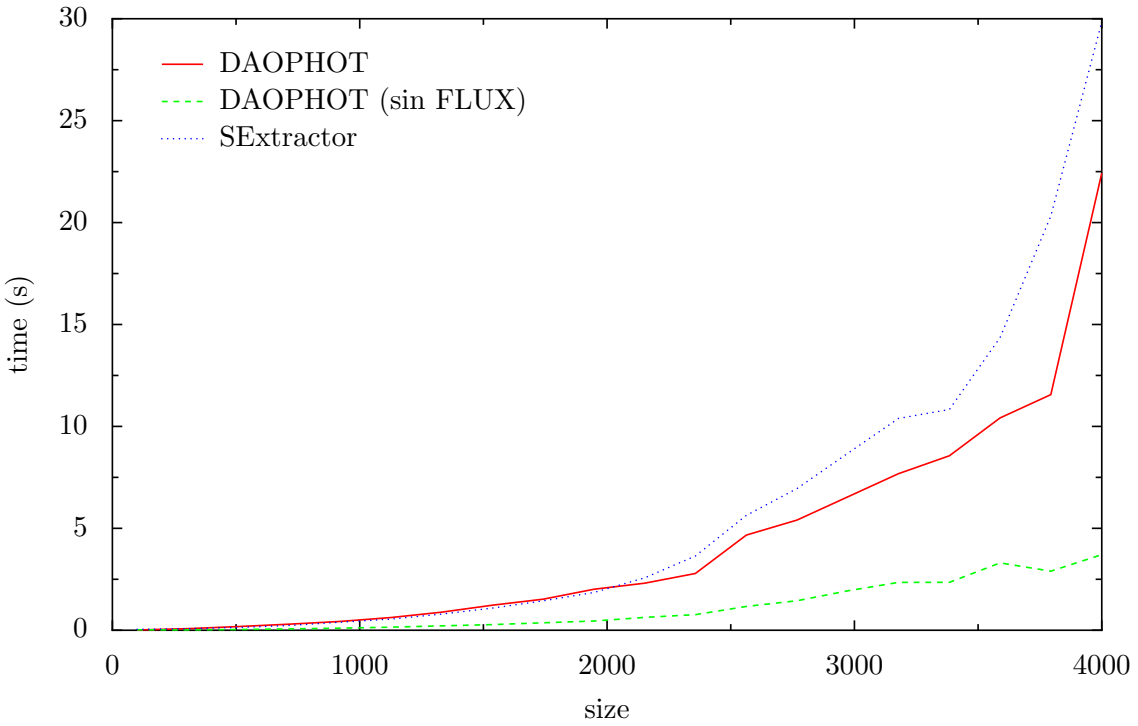


Fig. 5.4: Comparación de tiempos de la ejecución de los distintos métodos con densidad de estrellas constante

5.2. Fotometría

El objetivo de la fotometría a efectos de este trabajo es obtener un descriptor que pueda identificar fuentes y que sirva para decidir si los features detectados en sucesivas imágenes de la secuencia corresponden al mismo objeto. Para esto debería haber poca variación en el flujo calculado de una misma fuente a lo largo de los frames, y es deseable que distintas fuentes tengan flujos notablemente separados para no confundirlos entre sí.

Al igual que en la reducción, tenemos dos métodos para calcular la fotometría de los objetos, SExtractor y DAOPHOT. En este resultado vemos la curva de luz de distintas fuentes de la imagen. Cada curva representa el flujo obtenido de la fuente a lo largo de los frames. Como es de esperar, las curvas son constantes (no varía el flujo detectado a lo largo de los frames) y no se cruzan mucho entre sí (son distinguibles por su descriptor fotométrico).

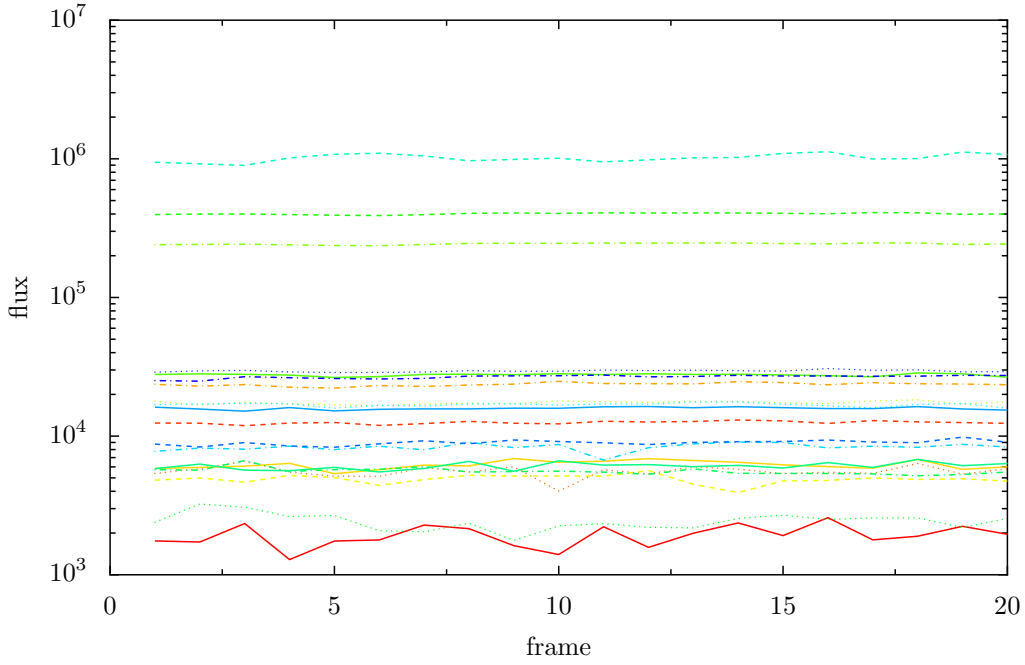


Fig. 5.5: Valor del flujo detectado por DAOPHOT de los objetos a lo largo de los frames

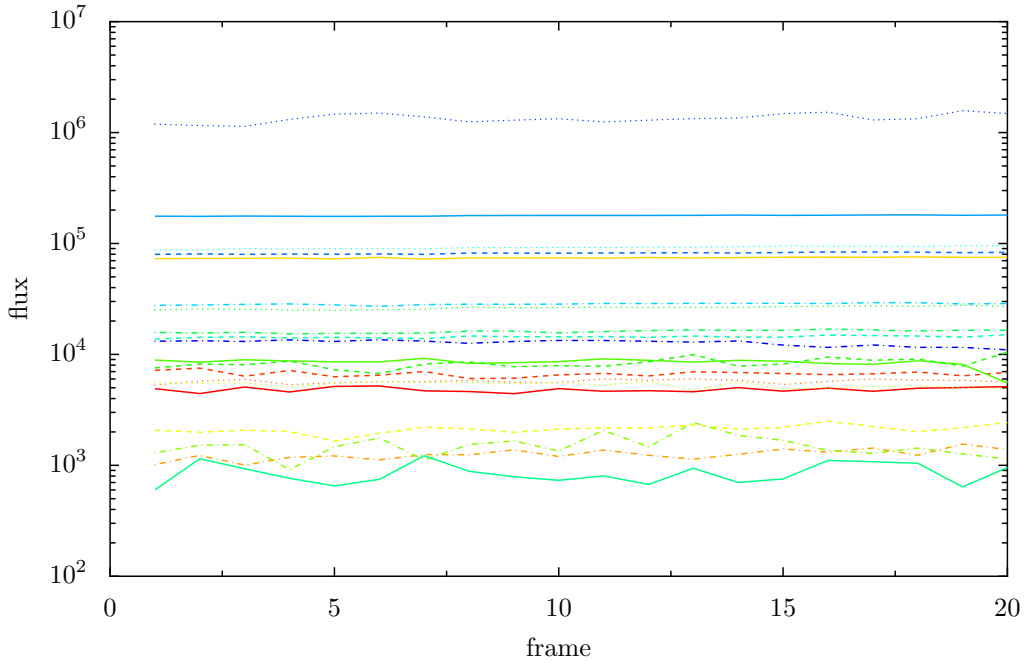


Fig. 5.6: Valor del flujo detectado por SExtractor de los objetos a lo largo de los frames

5.3. Alineación

Para probar la calidad del método de alineación generamos casos de prueba con conjuntos de features con distintas variaciones. Los parámetros que influyen en la alineación son: ángulo de rotación, vector de traslación, errores en las posiciones de los features.

En la figura 5.7 se observa el porcentaje de éxito en la alineación segun la rotación entre los conjuntos de features. Se ve que el método alinea correctamente cuando el ángulo está dentro de los 4º, y pierde efectividad cuando este ángulo aumenta. Como se explicó en el capítulo de adquisición, en general las imágenes de la secuencia sufren desalineaciones de pocos grados de rotación, por lo que el método es útil a efectos de este trabajo.

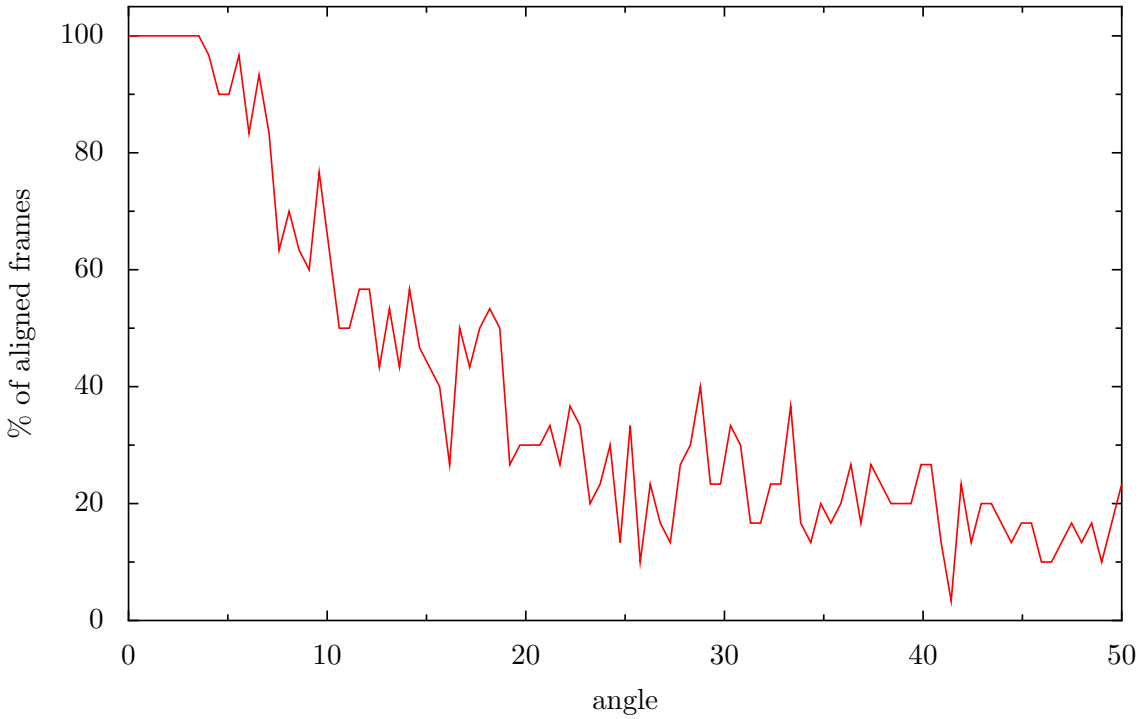


Fig. 5.7: Porcentaje de efectividad de la alineación en función del ángulo de rotación entre imágenes

En la figura 5.8 se grafica el porcentaje de éxito según la magnitud del vector de traslación. Se observa que el método alinea correctamente con translaciones de hasta 60 pixels, que supera el desplazamiento estimado entre sucesivas imágenes.

El tercer parámetro medido es el comportamiento de la alineación según el error en la estimación de los features detectados. En la figura 5.9 se observa que el método es efectivo cuando el error de cada feature se encuentra dentro de los 2 pixels. Este parámetro es ajustable en el método de alineación, variando la tolerancia del radio de error de los features. En general los detectores de features encuentran fuentes con precisión de un pixel, por lo que una tolerancia de 2 pixels es adecuada.

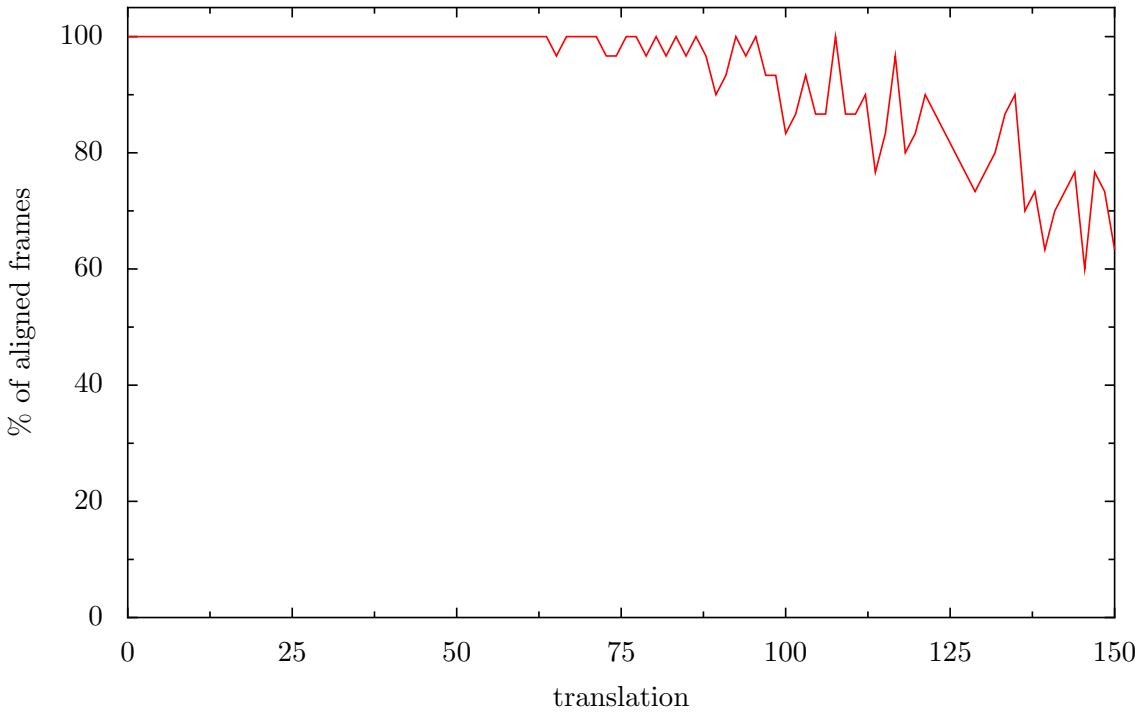


Fig. 5.8: Efectividad de la alineación en función de la traslación en pixels entre sucesivas imágenes

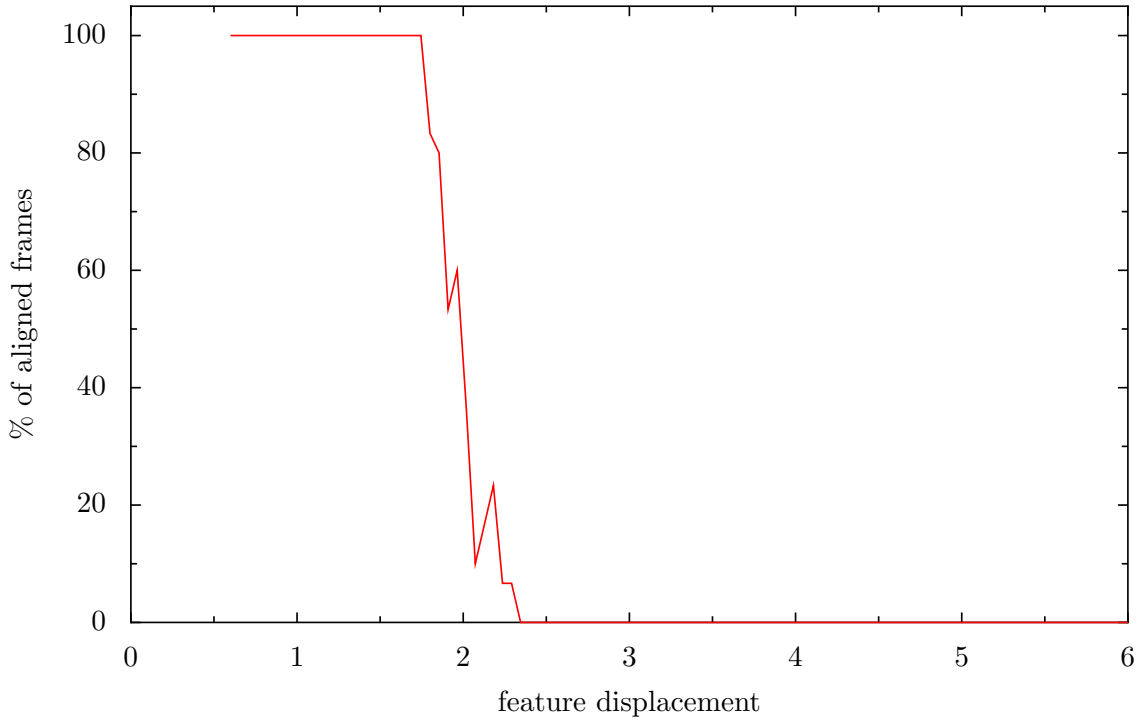


Fig. 5.9: Éxito de la alineación en función del error en las posiciones de los features

5.4. Detección de objetos móviles

Para imágenes sintéticas, medimos la efectividad de la detección de objetos para diferentes brillos del mismo. En la figura 5.10 se observa el porcentaje de éxito de la detección en función del flujo de luz recibido del objeto. Se puede ver que detecta objetos con un flujo de al menos 1900, que es muy poco brillante considerando el ruido de fondo y comparado con otras fuentes presentes en la imagen.

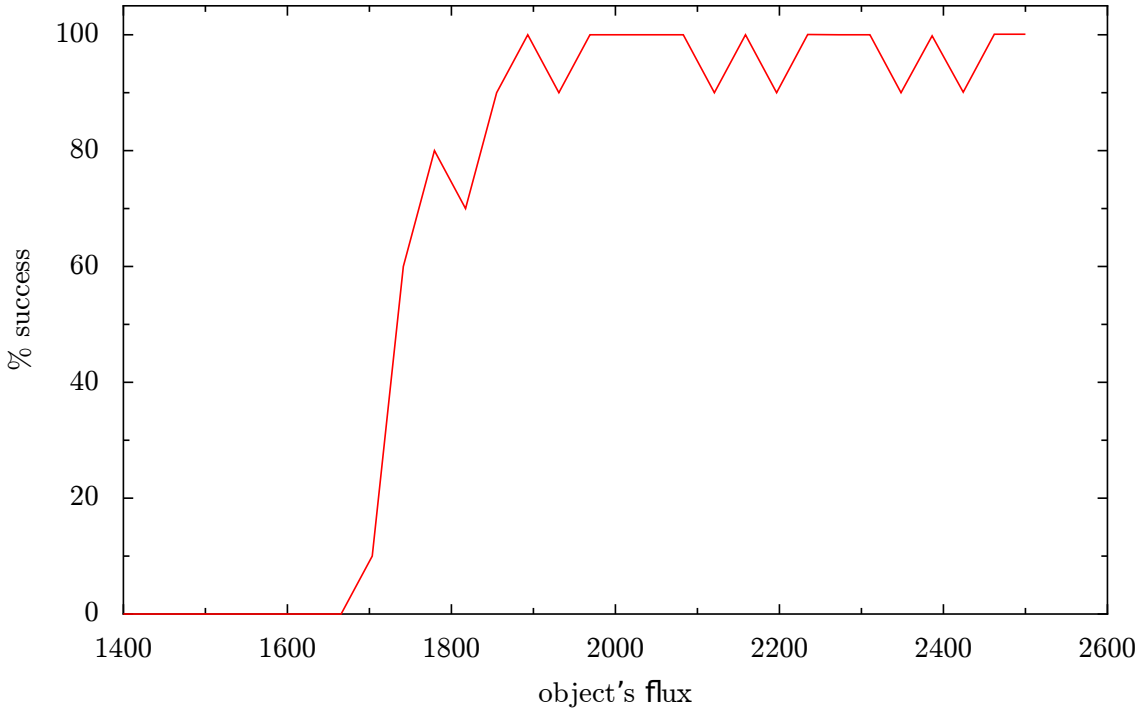


Fig. 5.10: Eficacia de la detección de objetos de distinto brillo

En el caso de imágenes reales adquiridas con un telescopio obtuvimos resultados muy satisfactorios. Las secuencias de imágenes que contienen objetos móviles provienen de un set de datos del Observatorio CASLEO ubicado en El Leoncito, San Juan, Argentina [1]. El set de datos utilizado en este experimento corresponde a una observación del asteroide 41427 [15].

En la imagen de la figura 5.11 se muestra la detección de la trayectoria del asteroide 41427 y también de otro objeto encontrado que es prácticamente imperceptible en las imágenes.

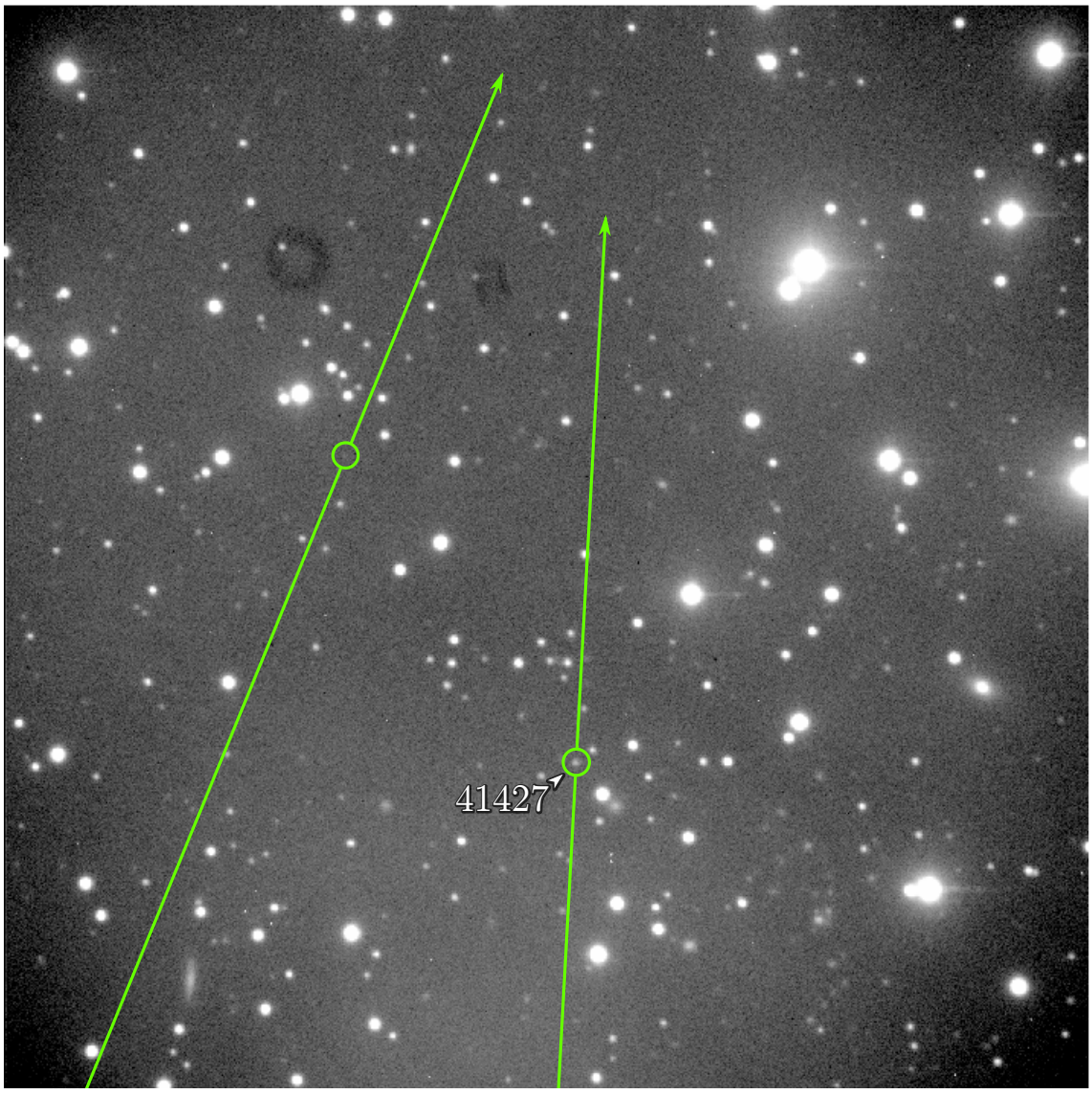


Fig. 5.11: Objetos móviles detectados en secuencias de imágenes adquiridas

5.5. Astrometría

Cuando encontramos un objeto móvil confiamos de Astrometry.net [12] para obtener las cordenadas del cielo y luego convertimos las coordenadas en píxeles a cordenadas en RA, DEC. Para comprobar la precisión de este método obtenemos la información de alineación de una imagen adquirida y comparamos las posiciones resultantes de los features con las fuentes del catálogo de la misma región del cielo.

En la figura 5.12 se grafican las posiciones de los features (con círculos) y las del catálogo (con cruces). Se ven muchas correspondencias entre fuentes que se alinean perfectamente, lo cual indica una buena alineación de la imagen.

Por lo tanto podemos concluir que las coordenadas del objeto móvil hallado

son precisas.

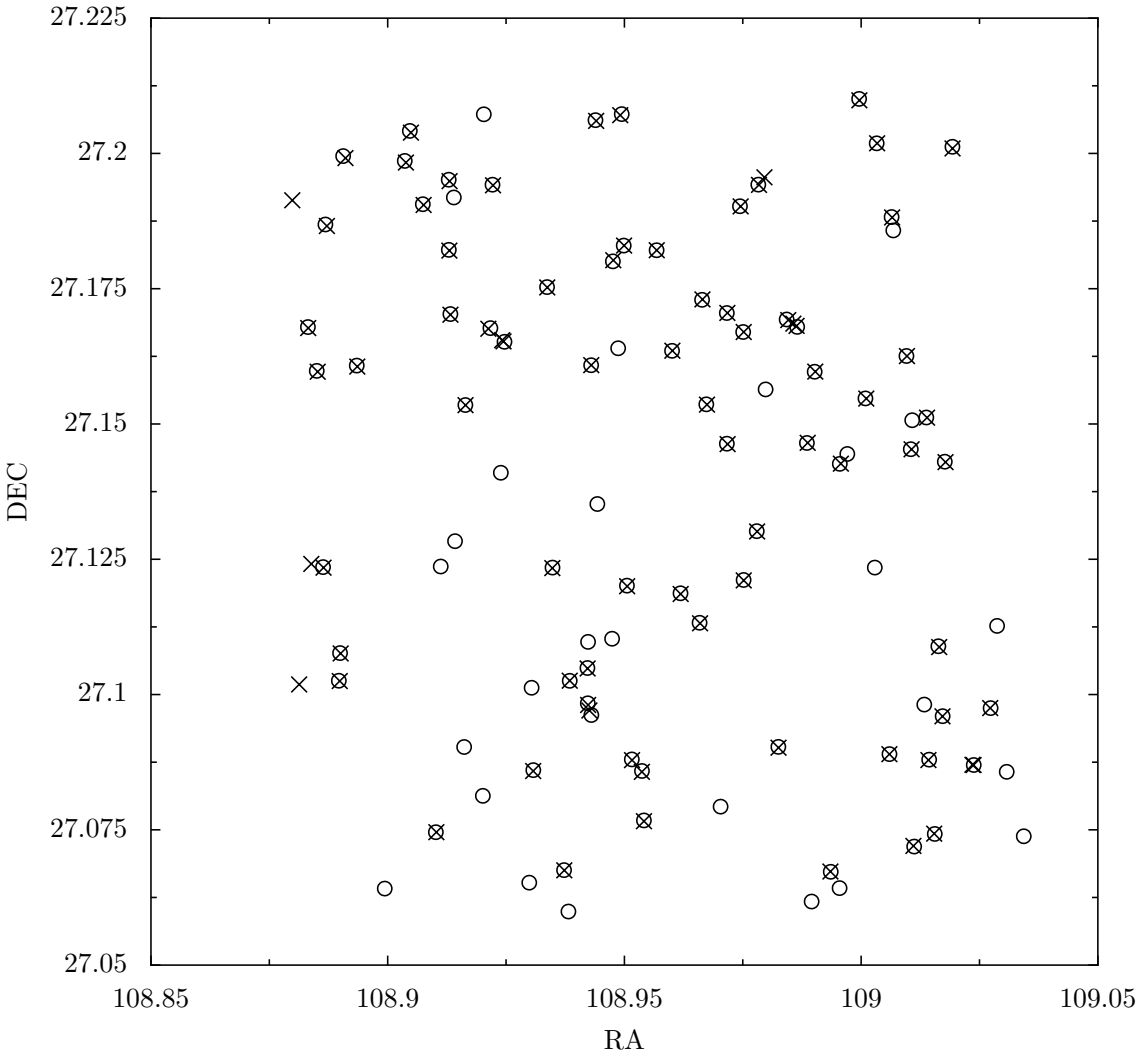


Fig. 5.12: Comparación de las posiciones de los features detectados y alineados con Astrometry.net y estrellas del catálogo USNO-B1 de la misma región del cielo

6. CONCLUSIONES

En este trabajo presentamos un sistema que automatiza el descubrimiento de asteroides y otros objetos móviles.

En la sección 3.3 detallamos el proceso de adquisición de las imágenes y describimos las propiedades de éstas. Mostramos cómo es el perfil de las fuentes capturadas, el efecto de distintos factores que introducen ruido en las imágenes (background, rayos cósmicos, etc), y mencionamos los metadatos que las acompañan.

En la sección 3.4 mostramos dos métodos de reducción de las imágenes: DAOPHOT y SExtractor. Se implementaron las rutinas de DAOPHOT para la extracción de las posiciones de fuentes puntuales (que consiste de filtrado, selección y refinamiento) y explicamos el uso de SExtractor como una librería externa. Como se ve en los resultados, los dos métodos resultan efectivos y detectan las fuentes puntuales presentes en la imagen, incluyendo eventuales objetos móviles.

También mostramos e implementamos un método de análisis fotométrico simplificado (sección 3.5) de las fuentes con el objetivo de obtener descriptores para los features para poder identificarlos. Los resultados muestran que el valor fotométrico obtenido de esta forma es estable a lo largo de las secuencias y es un descriptor útil para encontrar correspondencias de features entre imágenes.

En la sección 3.6 detallamos un método de alineación de secuencias de imágenes usando los features de la reducción, basado en ICP y RANSAC. Resulta efectivo para corregir desalineaciones que se encuentran en secuencias de adquisiciones típicas causadas por errores mecánicos.

En la sección 3.7 propusimos un método de detección de objetos móviles en las secuencias de imágenes, que opera buscando colinealidades en los conjuntos de features detectados. Experimentación con imágenes sintéticas generadas por computadora y imágenes reales muestra que el método es muy efectivo y detecta objetos de poco brillo que son muy difíciles de ver.

Luego describimos el uso de Astrometry.net (sección 3.8) para obtener datos astrométricos de las imágenes y encontrar las coordenadas celestes de los objetos móviles encontrados y reportar descubrimientos. La calidad de los datos astrométricos hallados de esta forma son de buena calidad y por lo tanto aseguran una localizacion y estimación de velocidad precisa de los objetos en el cielo.

Finalmente, en la sección 3.9, mostramos la operación de la montura robótica del telescopio para hacer seguimiento de los objetos descubiertos.

En el capítulo 4 detallamos la implementación del sistema creado como parte de esta tesis con los métodos descriptos en el capítulo 3, incluyendo el Hardware (4.1), Decisiones de diseño (4.2), la Arquitectura del sistema (4.3) y el Diseño de módulos (4.4).

El capítulo 5 presenta los resultados obtenidos de la implementación de los métodos. Estos resultados muestran que los métodos son efectivos y útiles para resolver los problemas planteados en la tesis, y por lo tanto que el sistema puede lograr descubrimientos de objetos móviles en el cielo nocturno tal como propuesto en los objetivos de esta trabajo.

6.1. Trabajo futuro

En este trabajo se enfatizó la detección de objetos móviles mediante el procesamiento de imágenes. Como trabajo futuro queda pendiente el testeo exhaustivo y puesta a punto del sistema de control automático para la robotización.

Por otro lado está proyectado el desarrollo de un sistema de gestión que permita la administración de "colas" para la ejecución secuencial de programas de observación. El fin último que se persigue es lograr un observatorio completamente autónomo que pueda operar en lugares hostiles para la presencia humana (como la Antártida) y que tome decisiones en función de los datos procesados y de catálogos existentes para el descubrimiento de fenómenos astronómicos desconocidos.

Bibliografía

[1] Complejo Astronómico el Leoncito. (2013) Casleo. [Online]. Available: http://www.casleo.gov.ar/

[2] European Southern Observatory. (2013) P2pp. [Online]. Available: https://www.eso.org/sci/observing/phase2/P2PPTool.html

[3] National Optical Astronomy Observatories. (2012) Iraf. [Online]. Available: http://iraf.noao.edu/

[4] National Aeronautics and Space Administration. (2013) Cfitsio. [Online]. Available: http://heasarc.gsfc.nasa.gov/fitsio/

[5] P. B. Stetson, "DAOPHOT - A computer program for crowded field stellar photometry," *Publ.Astron.Soc.Pac.*, vol. 99, p. 191, 1987.

[6] Space Telescope Science Institute. (2012) Pyraf. [Online]. Available: http://www.stsci.edu/institute/software_hardware/pyraf

[7] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[8] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[9] NumPy Developers. (2012) Numpy. [Online]. Available: http://www.numpy.org

[10] E. Bertin and S. Arnouts, "SExtractor: Software for source extraction." *Astronomy & Astrophysics, Supplement*, vol. 117, pp. 393–404, Jun. 1996.

[11] Space Telescope Science Institute. (2013) Pyfits. [Online]. Available: http://www.stsci.edu/institute/software_hardware/pyfits

[12] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis, "Astrometry.net: Blind astrometric calibration of arbitrary astronomical images," *The Astronomical Journal*, vol. 139, no. 5, p. 1782, 2010. [Online]. Available: http://stacks.iop.org/1538-3881/139/i=5/a=1782

[13] Z. ZHANG, "Iterative point matching for registration of free-form curves," 1992.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[15] Science Daily. (2013) Object 41427. [Online]. Available: http://comets-asteroids.sciencedaily.com/1/44439/41427-2000-DY4