Proyecto Fin de Grado
Ingeniería Aeroespacial

DESARROLLO DE UNA HERRAMIENTA PARA EL ANÁLISIS DE IMÁGENES ASTRONÓMICAS.

Autor: Beatriz Isora Arbona Camilleri

Tutor: Prof.Dr. Emilio Gómez González

Dpto. de Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Proyecto Fin de Grado
Ingeniería Aeroespacial

DESARROLLO DE UNA HERRAMIENTA PARA EL ANÁLISIS DE IMÁGENES ASTRONÓMICAS.

Autor:
Beatriz Isora Arbona Camilleri

Tutor:
Prof.Dr. Emilio Gómez González
Catedrático

Dpto. de Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Grado: DESARROLLO DE UNA HERRAMIENTA PARA EL ANÁLISIS DE IMÁGENES ASTRONÓMICAS.

Autor: Beatriz Isora Arbona Camilleri

Tutor: Prof.Dr. Emilio Gómez González

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis amigos.

Resumen

La ambición del ser humano por conocer aquello que no está a su alcance, ha hecho que el estudio del Universo sea uno de los principales temas de interés. En el siglo XX tiene comienzo el inicio de la era espacial, con el lanzamiento del Sputnik I (URSS), lanzado el 4 de octubre de 1957en Baikonur.

En la actualidad, disponemos de medios muy avanzados con los que poder explorar el espacio, todo lo que se encuentra más allá de la atmósfera terrestre. Así, se han diseñado satélites, cohetes, telescopios y diferentes instrumentos con los que poder ampliar nuestro conocimiento.

En el presente trabajo se van a procesar las imágenes obtenidas por el telescopio de 1.23 metros del centro astronómico Calar Alto. Se ha creado una interfaz mediante códigos implementados con el programa Matlab® que permite procesar dichas imágenes de una manera interactiva y sencilla.

Abstract

The ambition of the human trying to know what is not within his reach has made the study of the Universe one of the main topics of interest. In the twentieth century, the beginning of the space age begins with the launch of Sputnik I (USSR), launched on October 4, 1957 in Baikonur.

Nowadays, we have very advanced technology to explore space, everything that is beyond the Earth's atmosphere. Thus, satellites, rockets, telescopes and different instruments have been designed to expand our knowledge.

In this document, the images obtained by the 1.23 meter telescope of the Calar Alto astronomical center will be processed. An interface has been created through codes implemented with the Matlab® program that allows these images to be processed in an interactive and simple way.

Índice

ÍNDICE DE DIAGRAMAS DE BLOQUES

ÍNDICE DE ILUSTRACIONES

Notación

ROI	Región de interés (*region of interest*)
IRAF	Image Reduction and Analysis Facility
IDL	Interactive Data Language
FITS	Flexible Image Transport System
λ	Longitud de onda
f	Frecuencia
c	Velocidad de la luz
k	Número de onda

1 INTRODUCCIÓN

El cielo lleva observándose desde que el ser humano se volvió sedentario. En la antigüedad ya se medía el tiempo según la posición del Sol. Así se definió el concepto de Sol Medio. El Sol Medio es una construcción matemática abstracta definido para que, en el caso de que el plano del Ecuador y el de la eclíptica coincidieran, coincidiera con el Sol real en el perihelio y el afelio de la Tierra. El estudio del movimiento del Sol y las estrellas sirvieron a nuestros antepasados para crear distintos calendarios que fueron perfeccionándose a lo largo de los años, hasta que en 1582 se creó el conocido calendario gregoriano, el cual seguimos usando en la actualidad.

En el siglo XX tuvo comienzo el inicio de la era espacial, con el lanzamiento del satélite Sputnik I (URSS), lanzado el 4 de octubre de 1957en Baikonur. Desde entonces, numerosos satélites, sondas y vehículos espaciales han sido lanzados para poco a poco ir descubriendo el Sistema Solar. El primer foco de interés fue la Luna, debido a su cercanía. Algunas de las misiones que tenían como objetivo llegar a nuestro satélite natural son Luna (URSS), Pioner o Apollo, donde cabe destacar la primera misión que consiguió alunizar, Apollo 11, tripulada por Neil Armstrong, Michael Collins y Edwin "Buzz" Aldrin. A continuación, se van a nombrar las misiones espaciales que han tenido lugar en nuestro sistema. Messenger, Venera, Curiosity, Galileo, Cassini, Voyager 2 y New Horizons son los nombres de las sondas que se lanzaron a Mercurio, Venus, Marte, Júpiter, Saturno, Urano, Neptuno y Plutón, respectivamente. Además, la NASA (National Aeronautics and Space Administration) y la ESA (European Space Agency) lanzaron el Solar and Heliospheric Observatory (SOHO) para estudiar el Sol en 1995, el cual sigue operativo.

Para mejorar la investigación científica, se ponen en órbita distintos telescopios y satélites fuera de la atmósfera terrestre, para así evitar las interferencias debidas a la radiación de las distintas capas de ésta. Desde el espacio se puede observar tanto la Tierra como el espacio exterior. La Astronomía está ligada a todas las ciencias y artes, por ello es fundamental seguir avanzando en esta materia. Todos los científicos del mundo comparten el interés por conocer el Universo, lo que les lleva a cooperar en el desarrollo científico y tecnológico para poder lograr su propósito.

Las imágenes que se van a analizar se realizaron en el centro astronómico hispano-alemán de Calar Alto (CAHA) por una estudiante de Óptica de la Universidad de París, Laure Oudda. Los centros astronómicos son lugares equipados para poder observar con precisión los cuerpos celestes, por lo que permite el estudio del Universo. Se suelen situar en terrenos elevados, lejos de la ciudad y en regiones con condiciones climatológicas apropiadas para la observación astronómica. En España, contamos con varios centros astronómicos, por ejemplo, el Centro Astronómico de Yebes (Guadalajara), el Observatorio de Sierra Nevada, Calar Alto, el Centro Espacial de Canarias o el Observatorio del Teide, conocido mundialmente.

El observatorio de Calar Alto se encuentra en Almería, en la sierra de Los Filabres. El centro posee tres telescopios con aperturas de 3.5m, 2.2m y 1.23m cada uno. El telescopio de apertura de 3.5 metros, Zeiss, es el telescopio más grande de Europa Occidental continental. Además de los citados telescopios, CAHA cuenta con multitud de instrumentos, tales como: CARMENES, AstraLux, CCD Camera, LAICA, BUSCA, MOSCA, CAFE, OMEGA-2000, CAFOS, PMAS, PANIC y TWIN. Las imágenes que se recogen en este documento de la Luna y la Nebulosa Dumbbell (M27) fueron realizadas con el telescopio de 1.23 m.

El telescopio astronómico fue inventado en el siglo XVII por Galileo Galilei. Este fue mejorado con el paso de los años y hoy contamos con telescopios de gran potencia y eficacia. Son instrumentos diseñados para observar objetos lejanos. Los telescopios pueden ser refractores, reflectores o catadióptricos (mixtos), el primer telescopio de Galileo Galilei era de tipo refractor. Los refractores usan lentes convergentes o convexas, mientras que los reflectores utilizan espejos.

1.1 Objetivo:

El objetivo del presente trabajo será el procesado de imágenes obtenidas con un telescopio mediante el desarrollo de una herramienta con el software Matlab®.

Para ello, se creará una interfaz gráfica en la que se subirá la imagen a analizar y con la que se podrán realizar tareas como la extracción del fondo de la imagen, la identificación de objetos que aparezcan en ella o el cálculo del área de dichos objetos. Puesto que se va a trabajar con imágenes astronómicas, dichos objetos corresponderán regularmente con estrellas.

En dicha interfaz aparecerán distintos botones que permitirán la ejecución del procesado y la obtención del resultado final.

2 PROGRAMAS DE PROCESAMIENTO DE IMÁGENES ASTRONÓMICAS

Como resultado de la toma de imágenes del espacio mediante instrumentos astronómicos que absorben la luz de los diferentes cuerpos celestes, se diseñaron múltiples programas para realizar el procesamiento de dichas imágenes.

Algunos de los programas más utilizados por los astrónomos son IRAF o IDL. En este trabajo se ha optado por el uso del software Matlab®. A continuación, se analiza cada programa y el tipo de imagen con la que se trabaja en astronomía.

2.1 IRAF (Image Reduction and Analysis Facility)

Es el más usado en los observatorios profesionales del mundo. Se trata de un software libre muy potente. Desarrollado por el Observatorio estadounidense NOAO (National Optical Astronomy Observatories) en los años 80.

Funciona con LINUX pero mediante una máquina virtual puede ser instalado también en Windows.

Para visualizar imágenes necesita una herramienta adicional, DS9. Dentro de IRAF nos encontramos con las tareas, que son los numerosos comandos que sirven para realizar diversas acciones. Dichas tareas se agrupan según su funcionalidad en paquetes.

Los programas de reducción y análisis de datos de un telescopio se basan en este sistema, lo cual le concede la importancia que tiene.

2.2 IDL (Interactive Data Language)

El software propietario IDL pertenece a la empresa privada Exelis Visual Information Solutions y es un lenguaje de programación muy popular en la astronomía, usado para el análisis y visualización de datos. La NASA lo usó para misiones como Mariner y la International Ultraviolet Explorer.

Puede usarse con cualquier plataforma, Windows, Linux o Mac. Es un lenguaje intuitivo y fácil de aprender. Compatible con imágenes tipo FITS. La combinación de pantalla gráfica, interactividad y una operación orientada a la matriz, lo convierte en un programa fundamental en el análisis de imágenes, aunque está más orientado al procesamiento de señales.

2.3 Matlab®

Matlab® es la herramienta elegida para realizar las operaciones de procesamiento de este trabajo, en concreto su versión R2018a. Inicialmente, este software fue creado por una corporación privada estadounidense para trabajar con matrices, de ahí su nombre MATrix LABoratory (laboratorio de matrices). Se trata de un entorno sencillo de programación y visualización usado en ingeniería que dispone de un lenguaje de programación propio, lenguaje M, mucho más rápido que los lenguajes tradicionales y muy similar a ellos. Es compatible con Windows, Unix y Mac OS. Mathworks cuenta con numerosas toolboxes creadas por profesionales y encargadas de realizar diferentes tareas dentro del programa, las cuales pueden ser modificadas por el usuario.

Este programa trabaja con matrices por lo que define las imágenes como matrices mxn, donde m es el número de píxeles de largo y n el de ancho. Cada elemento de la matriz tiene un valor. Este valor va de negro a blanco, de 0 a 255.

Además, como se verá en el siguiente apartado, Matlab® ofrece la posibilidad de construir aplicaciones. En este trabajo la aplicación se realizará con el entorno GUIDE.

2.4 Imagen FITS

El formato FITS es el más usado en astronomía, avalado por la NASA y la Unión Astronómica Internacional (IAU). La abreviatura FITS significa Flexible Image Transport System.

Son imágenes digitales que se depositan en un archivo de cinta. Pueden poseer varios componentes, todos con un encabezado de 8 bits que emplea el código ASCII seguido de datos binarios. Soporta tres tipos de información, entero binario de 8 bits sin signo, entero binario con signo en complemento a dos de 16 bits y entero binario con signo en complemento a dos de 32 bits.

3 DESCRIPCIÓN DE LA INTERFAZ

La interfaz gráfica con la que se van a procesar las imágenes astronómicas en cuestión puede ser usada por cualquier usuario que no tenga conocimientos de programación básica ya que no será necesario que este acceda al código de programación de ésta.

En esta sección se definirán las instrucciones de uso de la aplicación.

3.1 GUIDE

Como se ha comentado con anterioridad, el programa que se va a utilizar para la elaboración de la aplicación de procesado de imágenes es Matlab®, concretamente la versión R2018a.

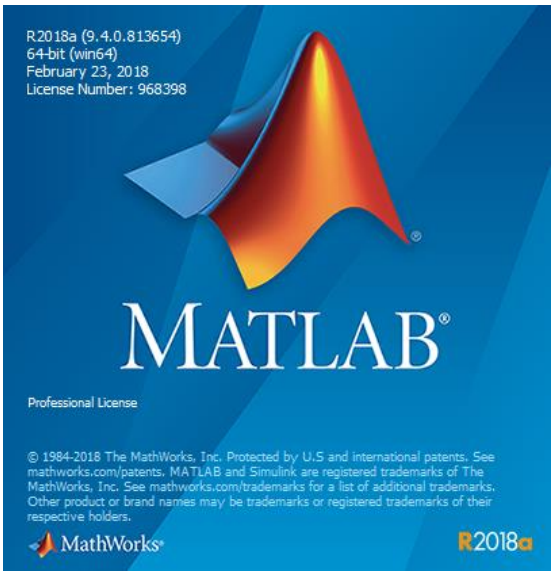

Ilustración 1 Matlab R2018a.

El entorno de programación empleado en este trabajo es GUIDE, el cual es muy intuitivo. Se basa en el diseño de interfaces de usuario (UI) añadiendo objetos de una manera muy simple, arrastrando y soltando elementos. A su vez, crea automáticamente un código dentro del programa Matlab®, archivo.m, dónde se ejecutan las instrucciones de cada elemento.

Al abrir Matlab® aparece la pantalla principal que se muestra en la siguiente imagen. Para acceder al entorno GUIDE se debe escribir ‘guide’ tal como aparece a continuación.

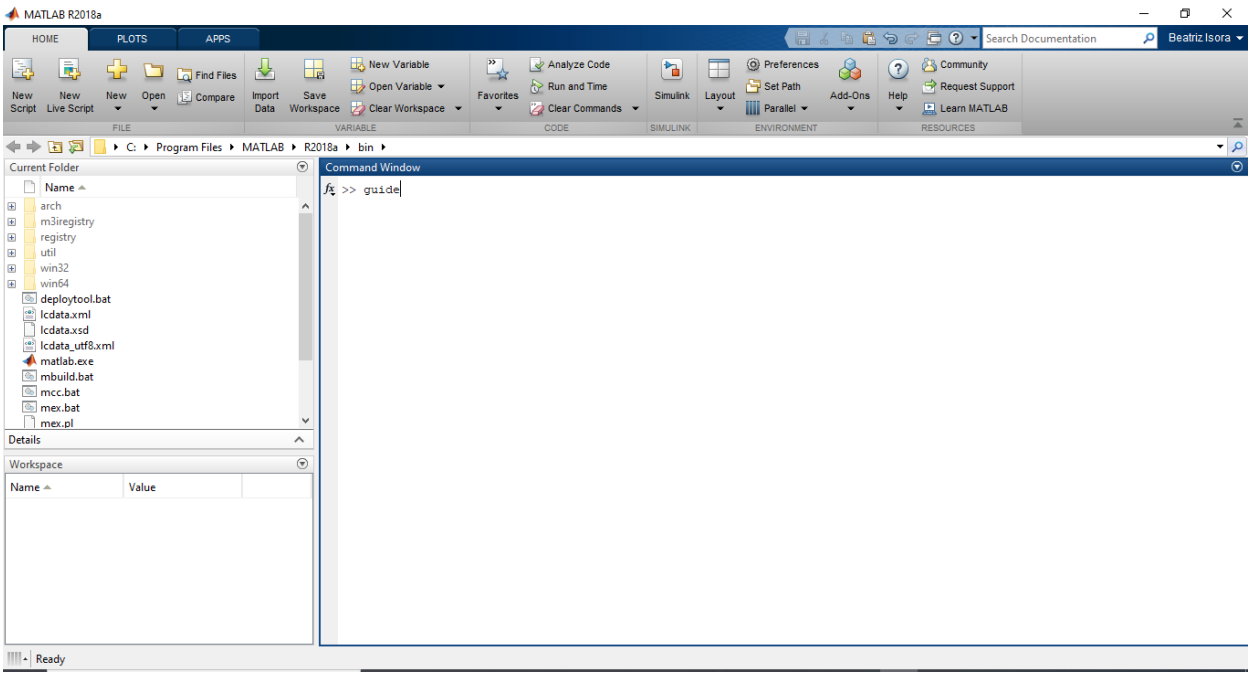

Ilustración 2 Pantalla principal Matlab.

Al pulsar el botón de ‘Enter’ aparecerá la siguiente ventana

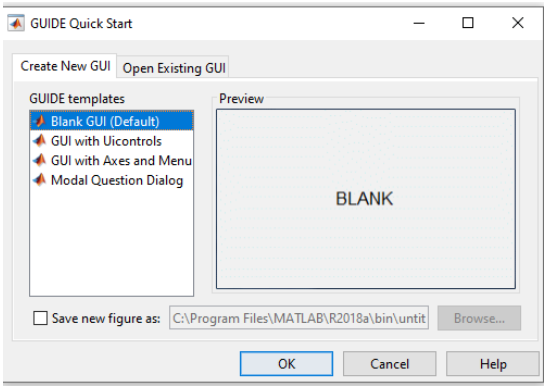

Ilustración 3 ventana de inicio GUIDE.

Se debe seleccionar la opción ‘Blank GUI (Deafault)’, ya que presenta un formulario en blanco sobre el que se añadirán los elementos deseados.

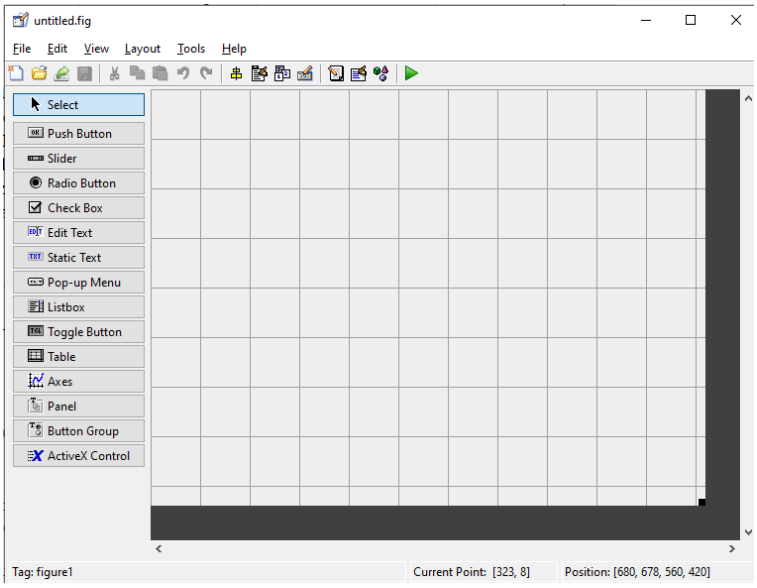

Ilustración 4 Entorno de diseño GUIDE.

En la ilustración anterior se muestra el entorno sobre el que se va a diseñar la interfaz gráfica. A la izquierda aparecen los distintos componentes de la aplicación.

Una aplicación GUIDE consta de dos archivos, un archivo.m con el código que se va a ejecutar y otro archivo.fig donde se almacenan los gráficos. El archivo.m tiene una estructura determinada y a medida que se añaden componentes en la interfaz aparece un código automático en él.

3.2 Elementos usados en la interfaz

La aplicación descrita en el apartado anterior consta de diferentes herramientas con la que poder realizar distintas tareas. A continuación se van a describir los que se han utilizado para el desarrollo de este trabajo.

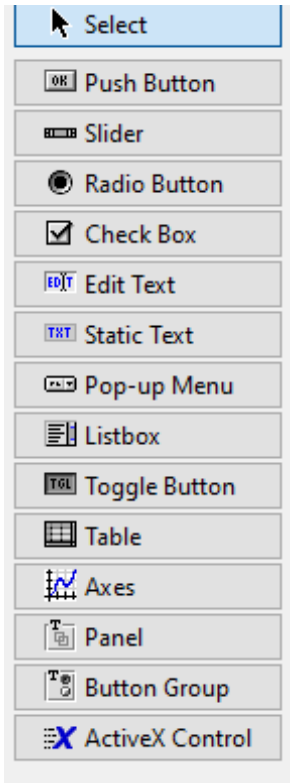

Ilustración 5 Elementos GUIDE

Tanto los botones de ‘Comenzar’, ‘Cargar imagen’, ‘Aceptar’, ‘Guardar’, ‘Procesar’, ’OK’, ‘Identificar objetos’, ‘Examinar objeto’,’Calcular área’, ‘Calcular centroide’, ‘Crear histograma de áreas’ y ‘Salir’ han sido creados con el element Push Button. Éste ejecuta una acción en el momento que se pulsa sobre él.

La pestaña Property inspector abrirá una ventana donde se define el Push Button, por ejemplo cambiando el nombre, tipo de letra o color.

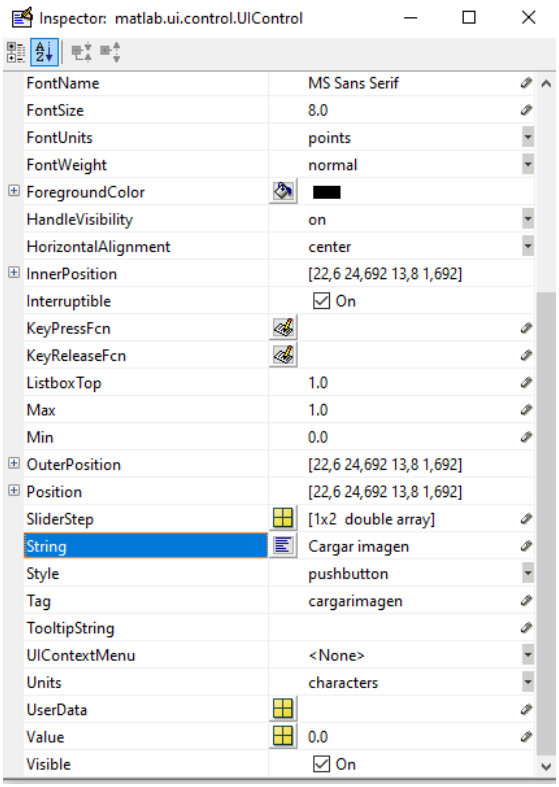

Ilustración 6 Property inspector.

Para crear la tarea habrá que generar un código, mediante el comando 'View Callback>>Callbacks' y aparecerá automáticamente elsiguiente código en el archive .m

```
% --- Executes on button press in cargarimagen.
function cargarimagen_Callback(hObject, eventdata, handles)
% hObject    handle to cargarimagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

El botón de ajuste de contraste se realize con un Slider. Este botón representa un rango de valores que va variando a medida que se desplaza la barra de un extremo a otro. En este caso se ha ejecutado de forma que modifique el contraste de la imagen cargada.

También se ha usado la opción de Button group para crear el panel de opciones tanto de la Roi como de la elección de fondo o primer plano.

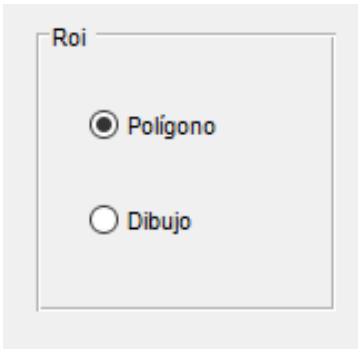

Ilustración 8 Group button roi.

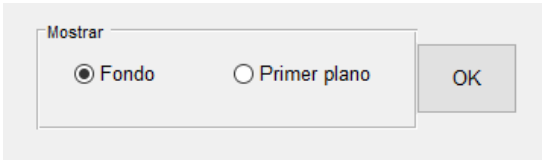

Ilustración 7 Group button mostrar.

Este elemento permite elegir una de las opciones que aparece para procesar la imagen. Se ejecutará la tarea asociada a dicho botón.

Por último, se agregan los marcos donde se muestran las imágenes. Estos se crean mediante el elemento Axes. En la interfaz se han desactivados los ejes ya que no son necesarios en la aplicación que se va a crear.

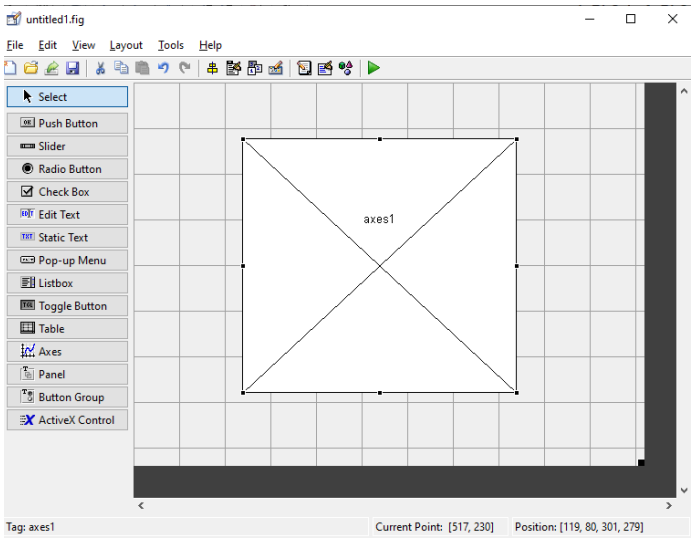

Ilustración 9 Axes GUIDE.

3.3 Descripción de la aplicación

La descripción de la aplicación se va a realizar exponiendo una serie de diagramas de bloques.

- Diagrama de bloques general:

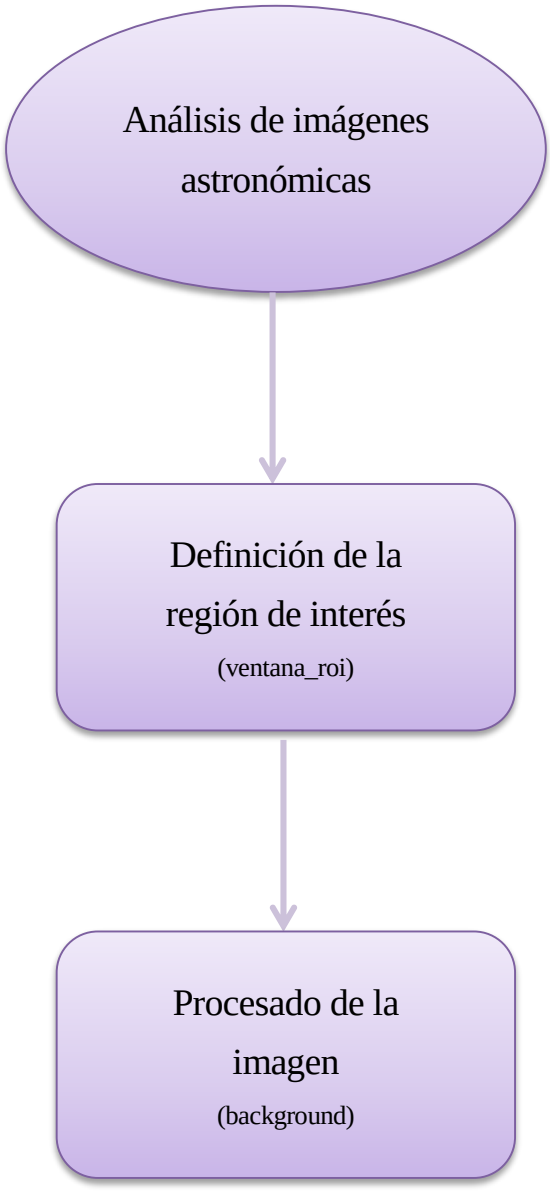

Diagrama 1 General.

- Definición de la región de interés:

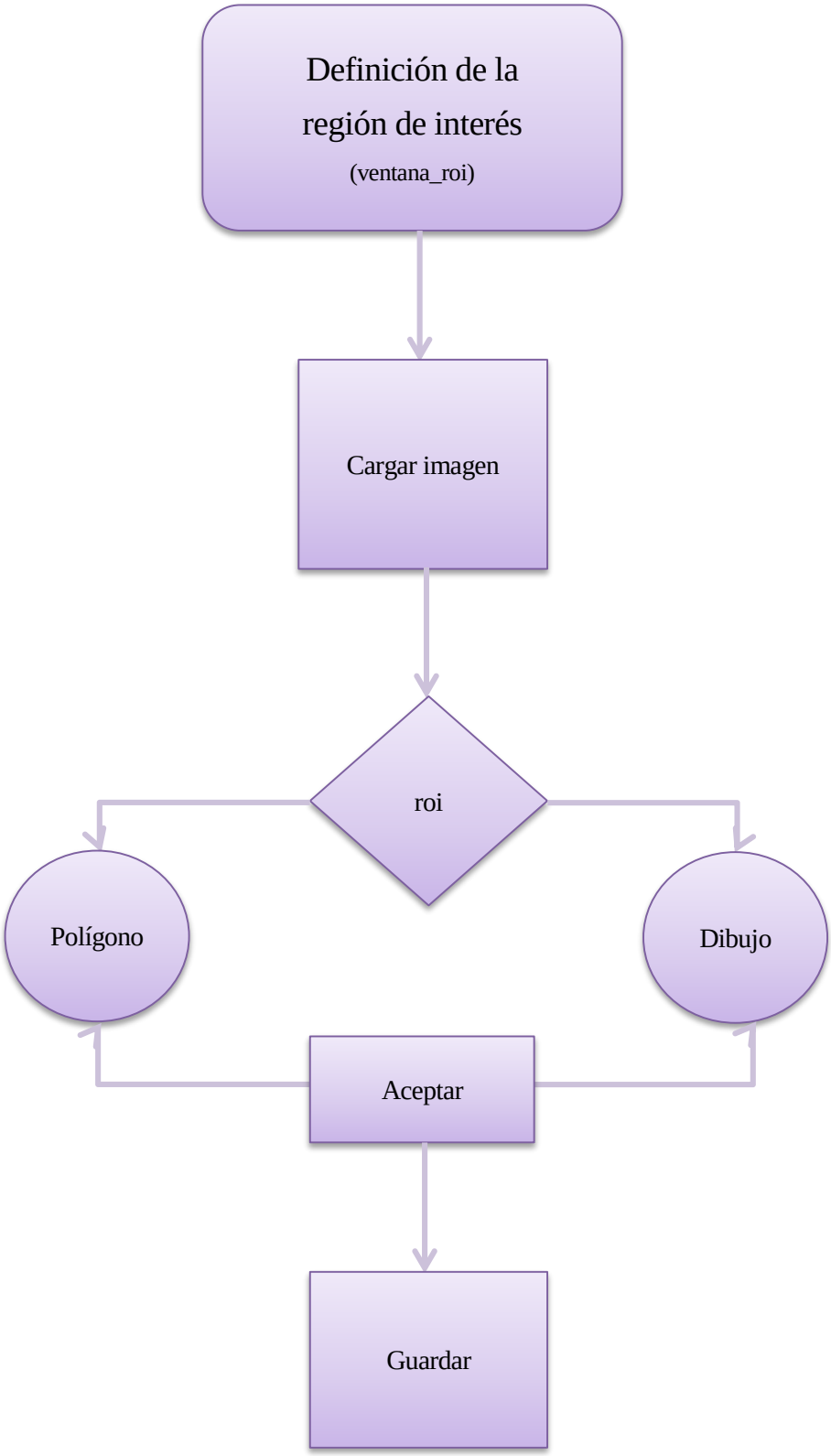

Diagrama 2 Definición roi.

- Procesar imagen:
  - Dividir imagen:

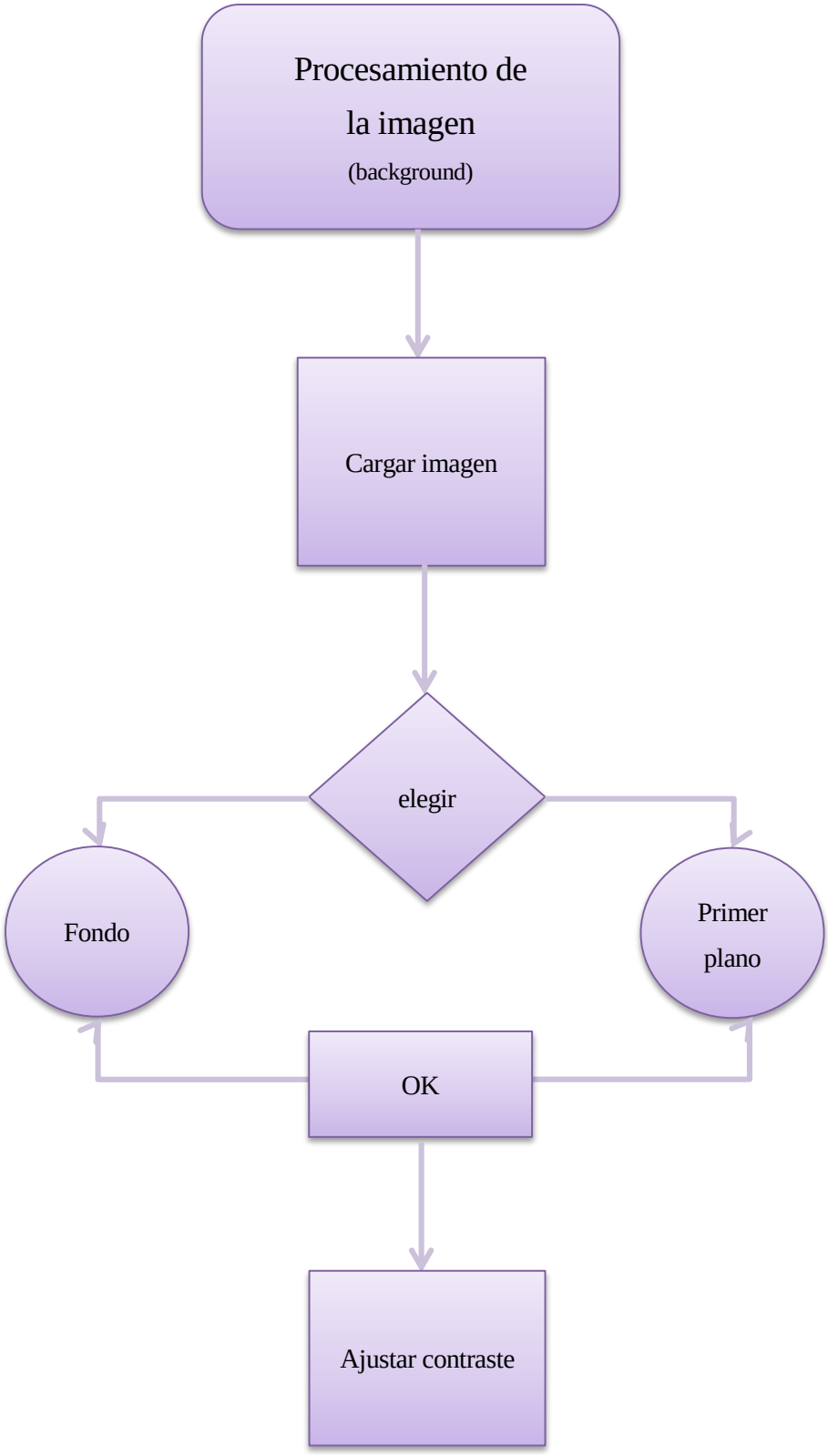

Diagrama 3 Dividir imagen.

- Identificar Objetos:

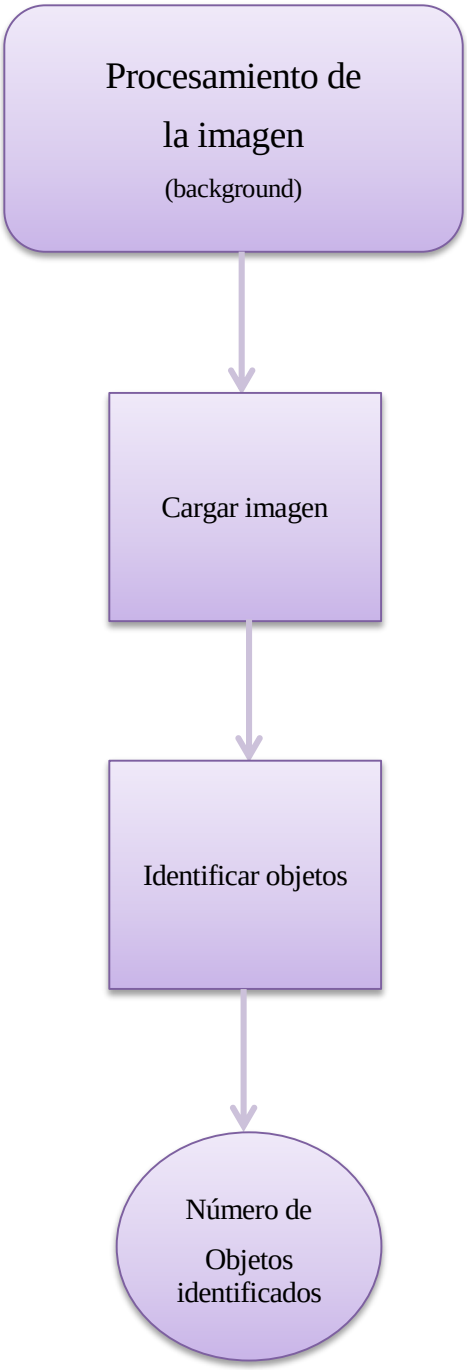

Diagrama 4 Identificar objetos.

- Examinar objetos:

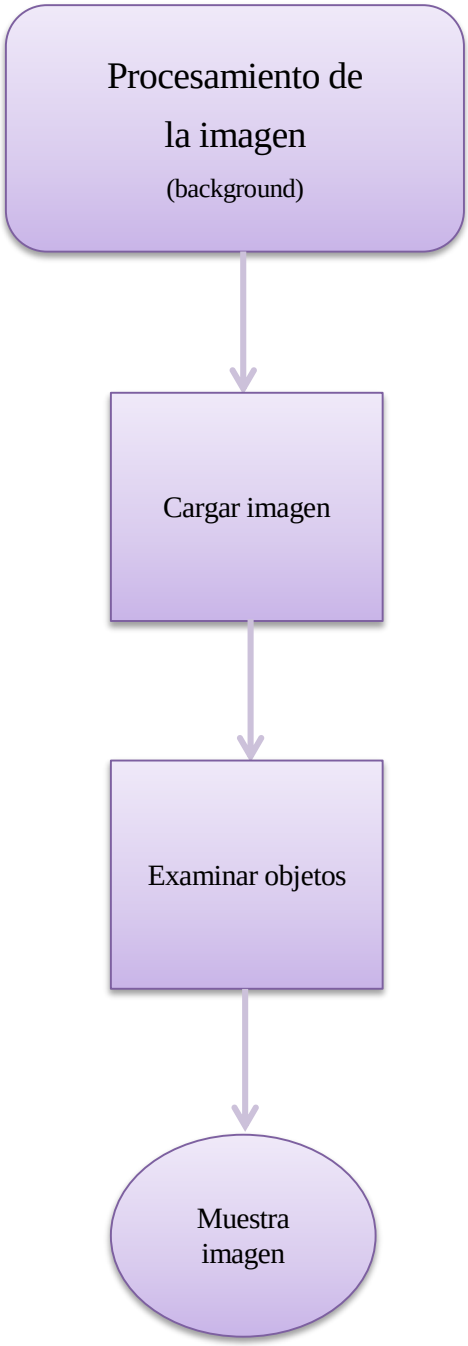

Diagrama 5 Examinar objetos.

- Cálculo de áreas:

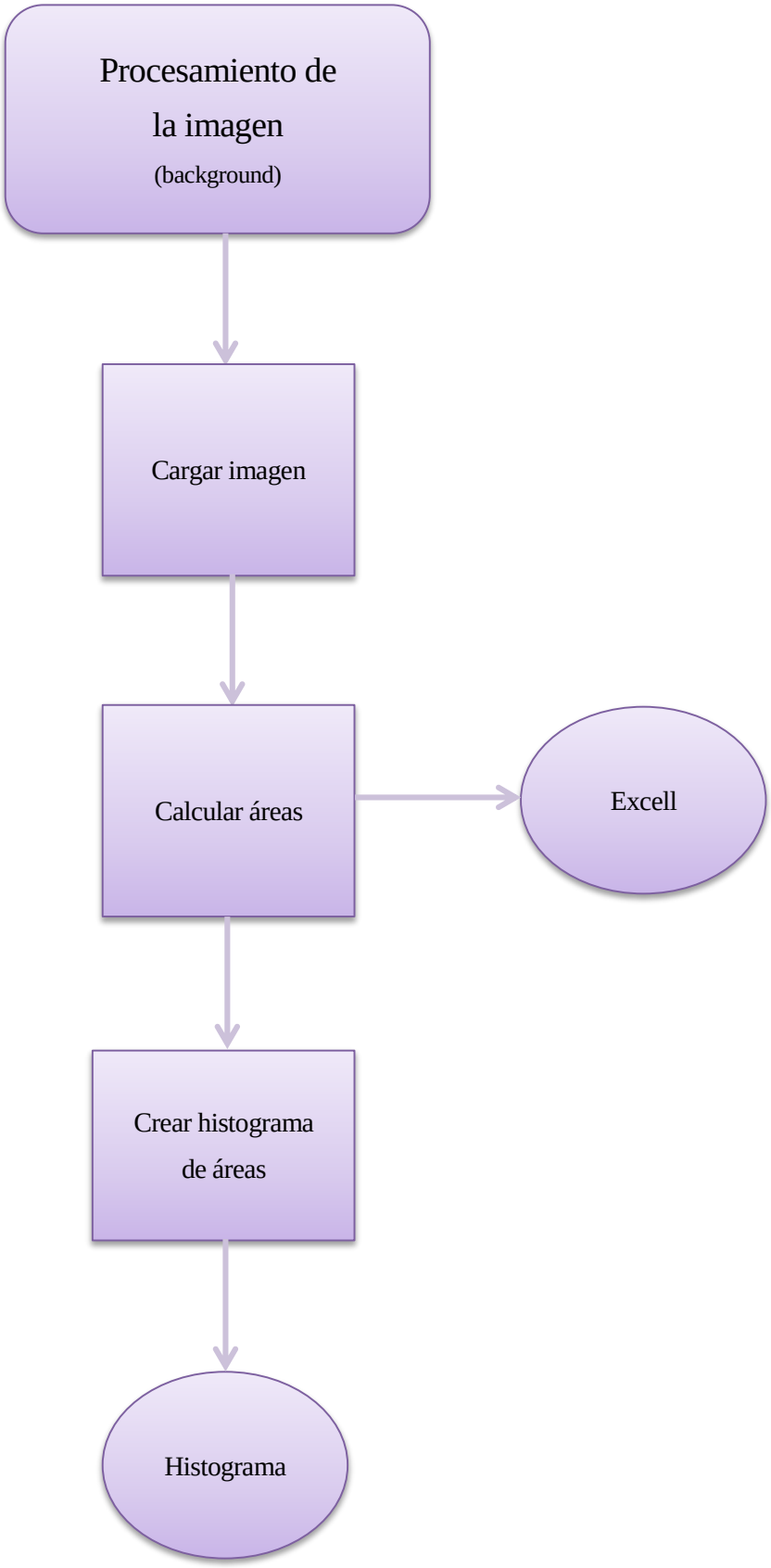

Diagrama 6 Cálculo de áreas.

- Cálculo de centroides:

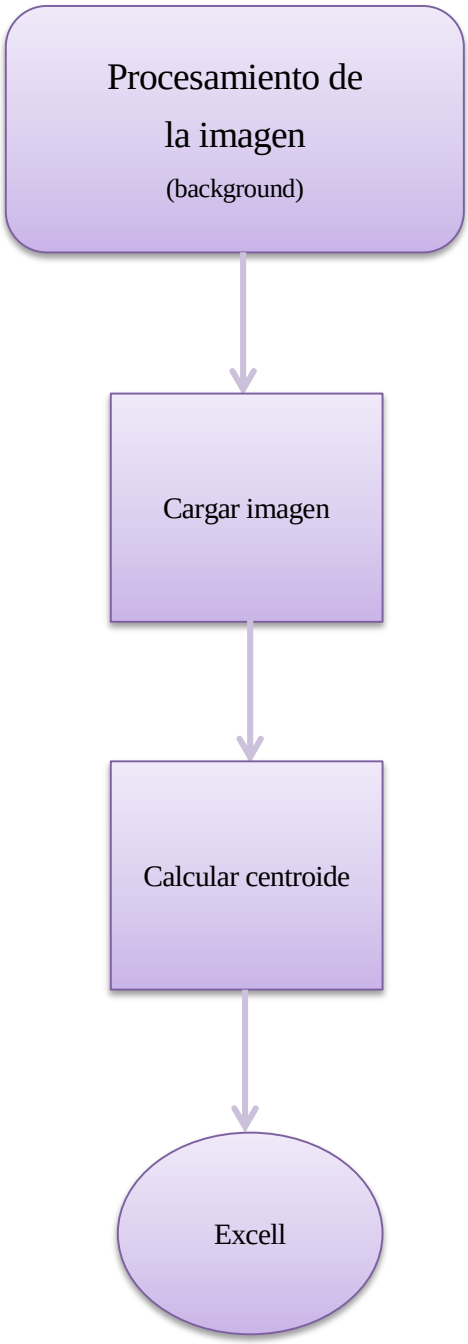

Diagrama 7 Cálculo de centroides.

4 GUÍA DE USUARIO

En este apartado se pretende describir la interfaz realizada con GUIDE en Matlab®. Se ha organizado de forma sencilla para que su uso sea intuitivo y no presente ninguna dificultad a la hora de manejarlo. Se mostrarán imágenes de la aplicación para que el usuario pueda hacerse una idea de cómo funciona.

4.1 Guía

A continuación se enumerarán los pasos a seguir para usar aplicación. A medida que se va seleccionando una tarea aparecerá una descripción del proceso en la ventana inicial de Matlab®.

1. En primer lugar se procederá a abrir el programa Matlab® y se seleccionará la carpeta 'INTERFAZ ANÁLISIS IMÁGENES'.

2. Se ejecuta el archivo 'portada.m'. Acto seguido se abrirá la interfaz, donde aparecerá la portada de la aplicación.

Ilustración 10 Portada.

3. El usuario deberá pulsar el botón 'comenzar' para que aparezca la siguiente ventana.

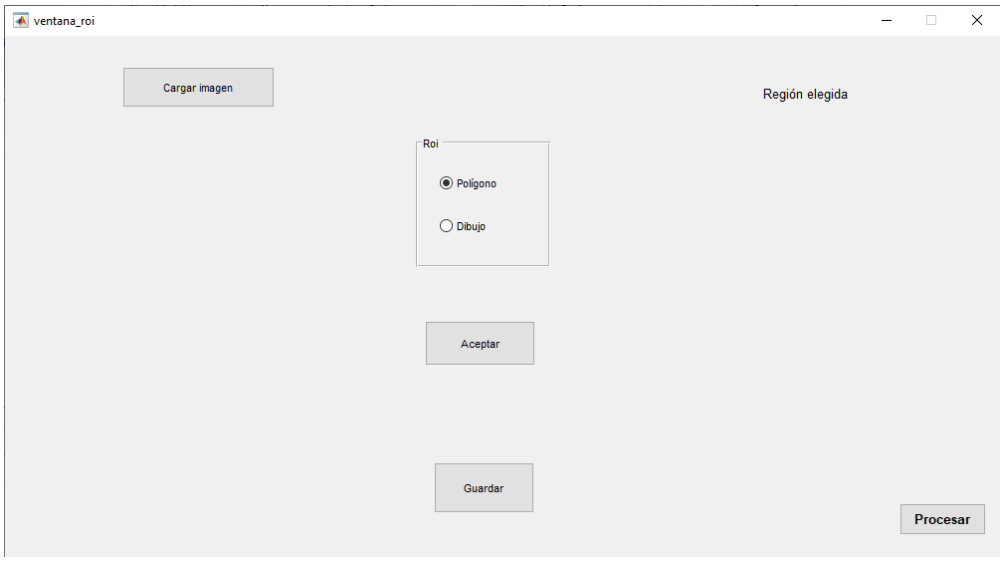

Ilustración 12 Ventana de realización roi.

4. Al pinchar sobre el botón 'Cargar imagen' se abrirá la carpeta desde la cual se podrá elegir la imagen deseada. Se ha programado para que sólo aparezcan las imágenes con formato .jpg o .tif

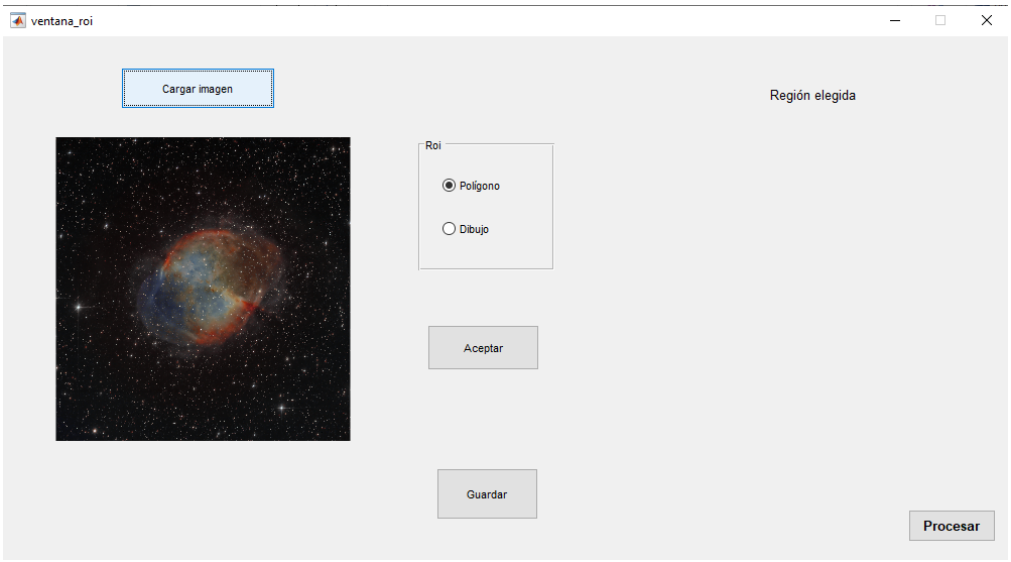

Ilustración 11 Imagen cargada en la interfaz.

5. El siguiente paso será la selección de la región de interés (roi).

Para el análisis de una imagen es aconsejable seleccionar la sección que interesa para el estudio. Por ello se ha creado el siguiente Group button con el que se podrá recortar la imagen cargada.

Se distinguen dos opciones, realizar una roi con forma poligonal o mediante un dibujo a mano alzada.

En caso de querer definir la roi mediante un polígono se deberá pulsar el botón ‘polígono’. Para ello el usuario tendrá que seleccionar una serie de puntos sobre la imagen de la izquierda haciendo click donde se desee. Para esta opción es necesario primero seleccionar la opción ‘dibujo’ y después volver a seleccionar ‘polígono’.

En el caso de querer realizar la región a mano alzada se seleccionará el botón ‘dibujo’ y con el ratón, se dibujará el contorno de la roi sobre la imagen de la izquierda.

En ambos casos será necesario pulsar el botón ‘Aceptar’ para poder seleccionar los puntos de la imagen.

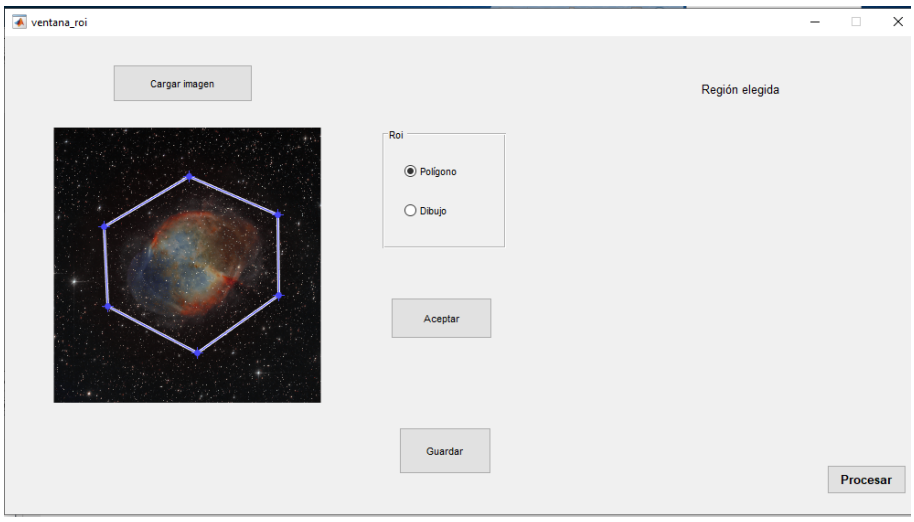

Ilustración 13 Selección de la roi.

6. Una vez se haya realizado el paso anterior, hacer doble click sobre la región descrita para que aparezca en el cuadro de la derecha.

7. Pulsar el botón ‘Guardar’ para salvar la imagen de la región de interés.

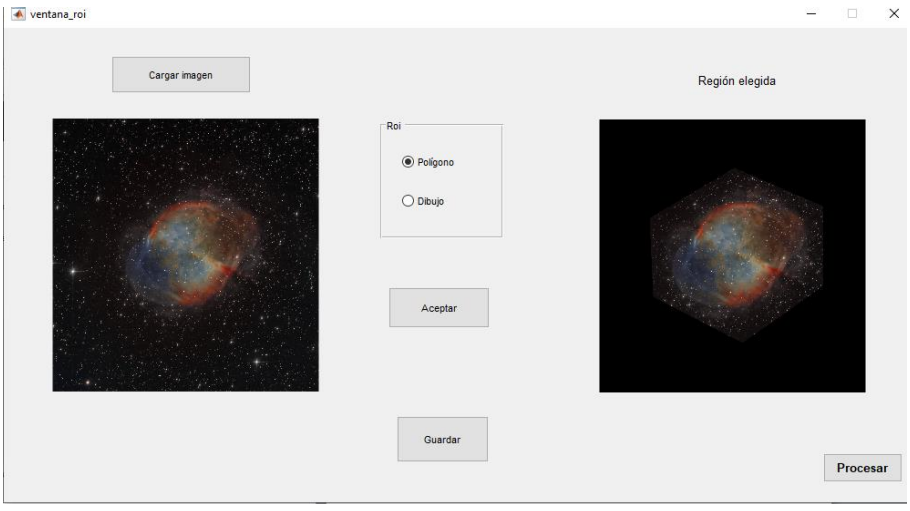

Ilustración 14 Región elegida.

8. Para pasar al procesado habrá que seleccionar el botón que aparece en la esquina inferior derecha, 'Procesar'.

9. Aparecerá una ventana nueva donde habrá que cargar la imagen deseada de la misma manera que en el cuarto paso, pulsando el botón 'Cargar imagen'. Aparezcan sólo imágenes con formato .jpg o .tif ya que son los más usados en astronomía.

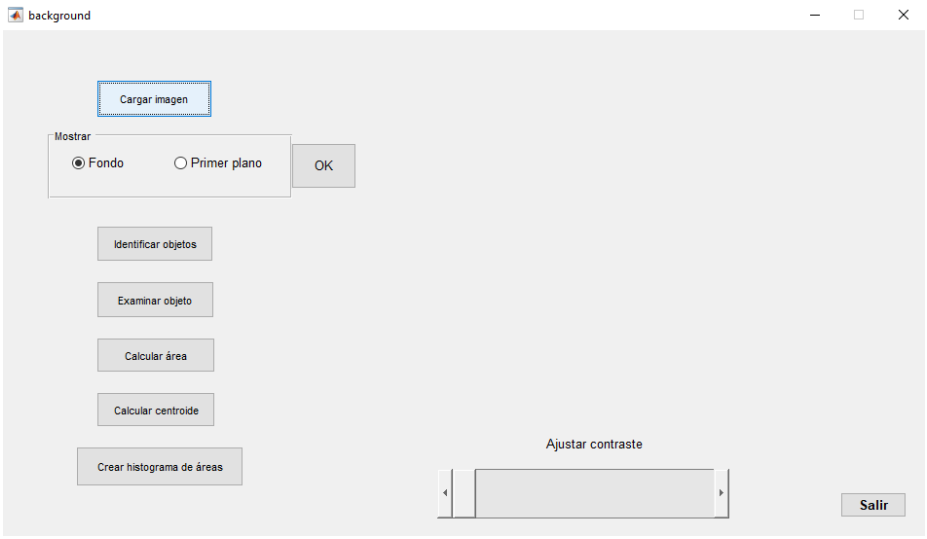

Ilustración 15 Ventana background.

10. Ahora se presentan dos opciones. Si se quiere analizar el fondo de la imagen clickar dos veces sobre 'fondo' y en caso contrario, seleccionar el botón 'primer plano'. Una vez elegido, se pulsa el botón 'OK' para validar la acción.

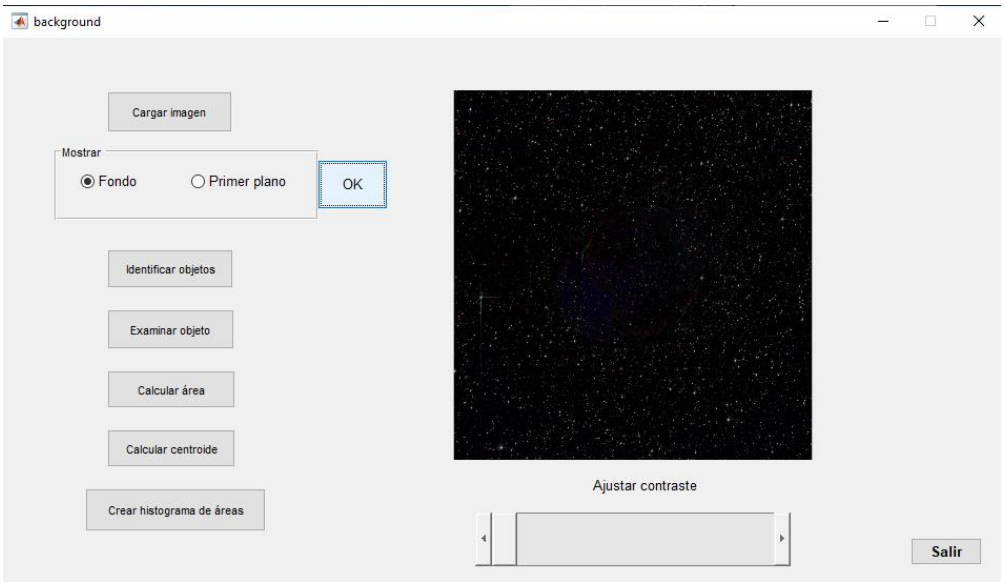

Ilustración 16 Fondo de la imagen.

La extracción del fondo de la imagen consiste en jugar con la iluminación de las distintas zonas de ésta. El procedimiento que usa Matlab® para realizar esta tarea es la apertura morfológica, que consiste en una erosión seguida de una dilatación. Después, se restará a la imagen original la imagen de fondo calculada. Podría realizarse en un solo paso mediante el comando imtophat.

Además hemos añadido la operación contraria, con la que se obtendrá el plano principal de la imagen a estudiar.

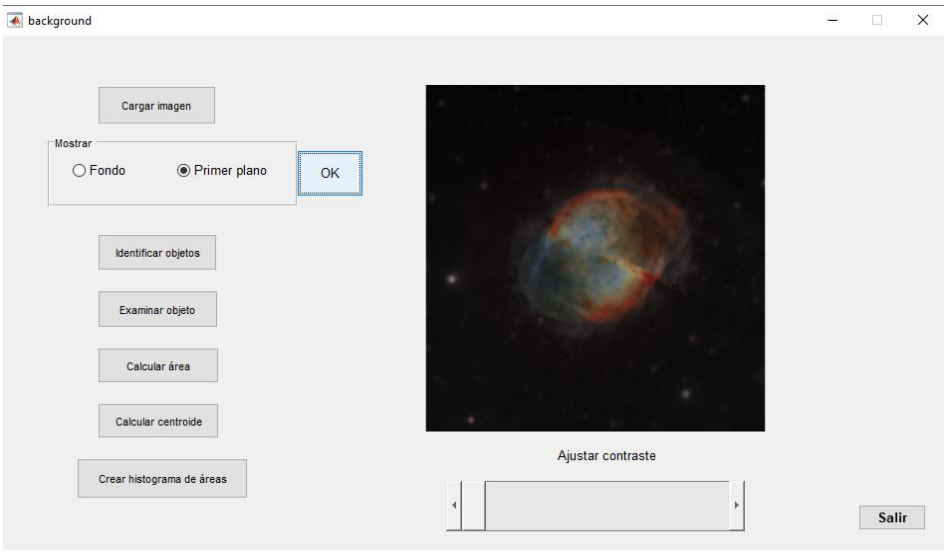

Ilustración 17 Primer plano de la imagen.

11. Debajo de la imagen aparece un slider con el que se puede variar el contraste de la misma desplazando la tecla hacia izquierda o derecha. En la siguiente ilustración se puede observar el cambio de contraste cuando el slider se posiciona en el centro.

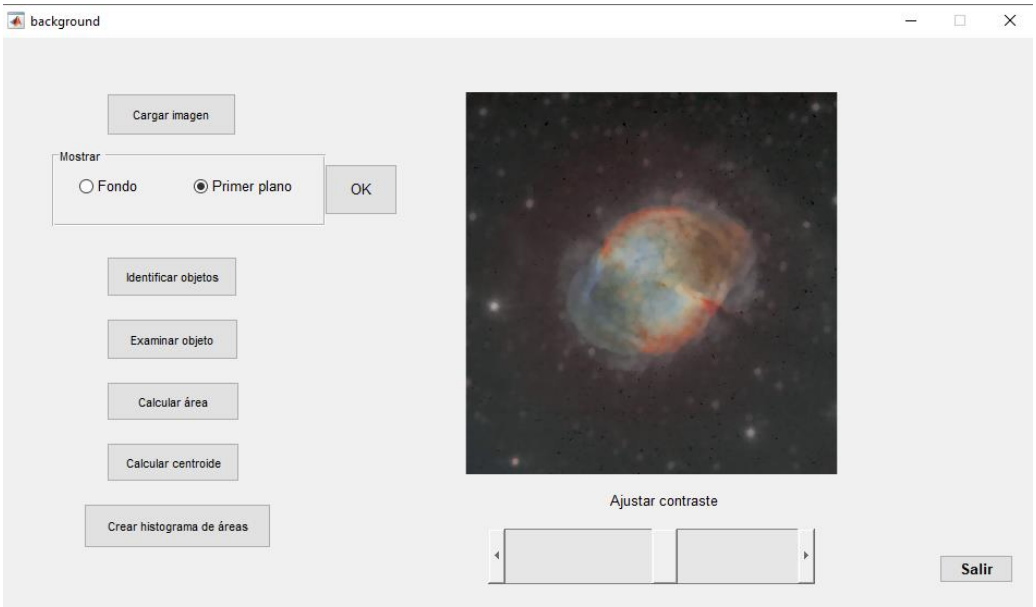

Ilustración 18 Cambio de contraste con slider.

12. Al seleccionar ‘Identificar objetos’ aparecerán datos sobre la imagen y sus objetos en la ventana principal de Matlab.

```
>> portada
    Connectivity        ImageSize          NumObjects    PixelIdxList
    ____________    ____________________    __________    _____________

        26          2578    2500        3       2649      [1x2649 cell]

>>
```

Ilustración 19 Objetos de la imagen.

13. Si se selecciona el botón ‘Examinar objetos’ se mostrarán los objetos localizados en la imagen a color, sobre un fondo blanco. Para ello se ha eliminado el ruido, eliminando objetos de menos de 50 píxeles.

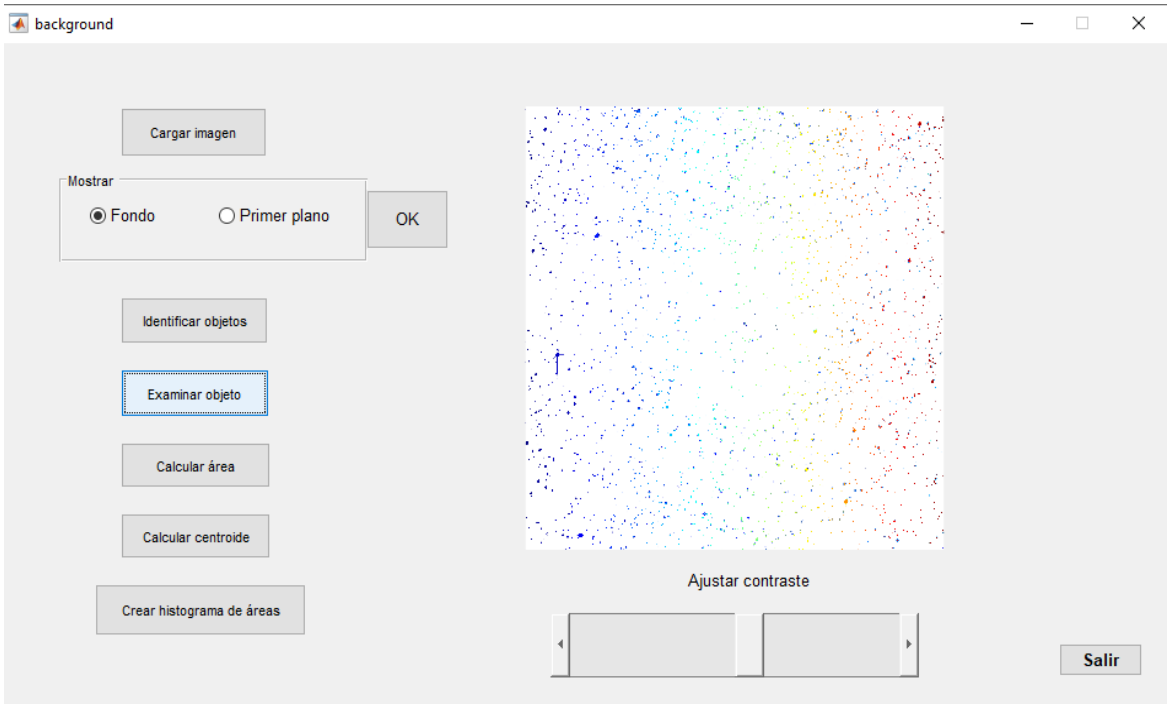

Ilustración 20 Objetos examinados.

14. Para Calcular el área de los objetos debemos saltarnos el paso anterior.

Justo después de identificar los objetos (paso 12) clickar sobre ‘Calcular Área’ y se creará un archivo .xls donde se recogerán los valores de las áreas. Además en la ventana principal de Matlab se mostrará el valor del área mínimo.

15. Para Calcular el centroide de los objetos debemos saltarnos el paso 13 (al igual que en el paso anterior).

Justo después de identificar los objetos (paso 12) clickar sobre ‘Calcular Centroide’ y se creará un archivo .xls en el que aparecerán los valores de los centroides. Igual que con las áreas, se mostrará en la ventana principal de Matlab el valor del centroide mínimo.

16. Por último, si se quiere obtener el histograma de las áreas, sólo habrá que pulsar sobre el botón 'Crear histograma de áreas' y éste aparecerá en una ventana nueva. Un histograma es una representación gráfica en forma de barras, en este caso muestra el valor de las áreas.

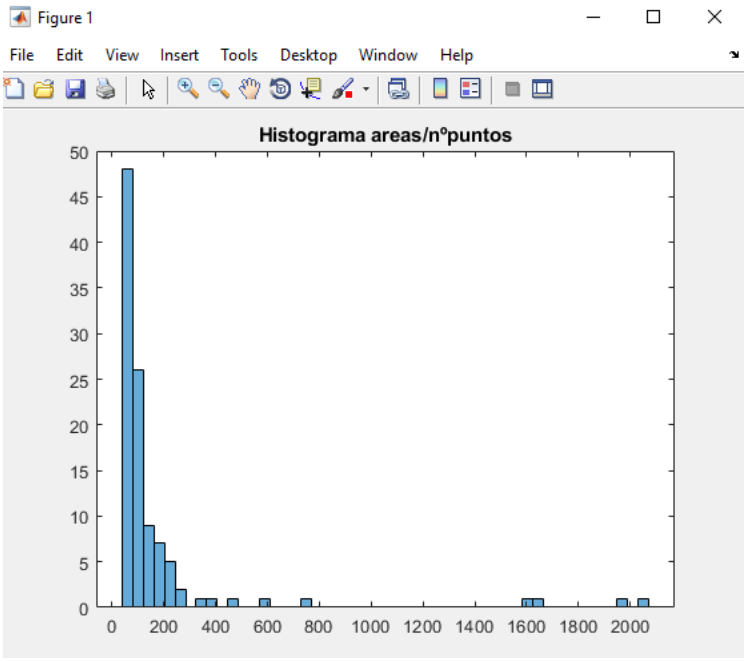

Ilustración 21 Histograma de áreas.

5 CONVERSOR

Se ha realizado también un código en Matlab® que permite convertir los datos obtenidos de las imágenes del telescopio en otras unidades. Se ingresará el valor de la longitud d onda, frecuencia, energía y número de onda en las unidades en las que se haya conseguido y automáticamente aparecerá la conversión en pantalla.

La longitud de onda es la distancia entre dos puntos consecutivos que se encuentran en la misma fase, es decir, que están en el mismo estado de vibración. Se representa con la letra griega λ y es la inversa de la frecuencia (f).Se mide en m. La longitud de onda determina los colores, por ejemplo, el rojo se corresponde a unos valores de λ que van de 625 a 740 nm y el violeta entre 380 y 435 nm. Estos corresponden con los límites del espectro de luz visible. En el código se hará la conversión con tres unidades, nm, μm y Å (angstrom)

La frecuencia se mide en Hz o min^{-1}. Mide las oscilaciones por unidad de tiempo.

La siguiente fórmula relaciona la frecuencia y la longitud de onda. Se puede observar como son inversamente proporcionales por lo que si la frecuencia aumenta, λ disminuye.

$$f = \frac{c}{\lambda}$$

Donde c es la velocidad de la luz, $c \approx 3x10^8\, {}^{m}/_{s}$.

La energía aumenta con la frecuencia y disminuye con la longitud de onda. Se medirá la energía en J o eV.

Por último, el número de onda (k) son las veces que vibra la onda por unidad de distancia. Se relaciona con la longitud de onda de la siguiente manera: $k=2\pi/\lambda$ por lo que es una magnitud de frecuencia.

A continuación, se muestra una imagen donde se pueden ver los resultados del código.

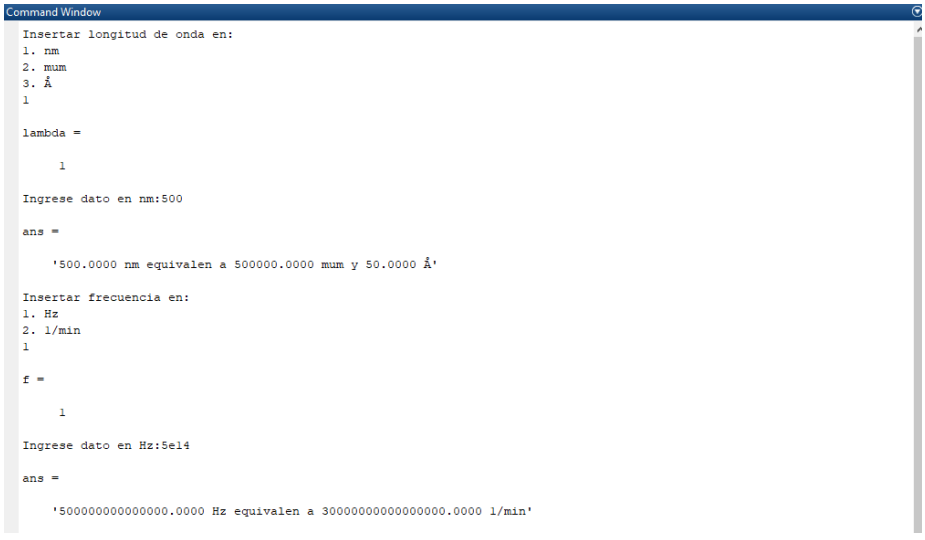

Ilustración 22 Conversor longitud de onda y frecuencia.

Ilustración 23 Conversor energía y número de onda.

CONCLUSIONES. FUTURAS LÍNEAS DE TRABAJO

En el presente trabajo se ha desarrollado una interfaz gráfica de procesado de imágenes astronómicas. Esta aplicación servirá al centro astronómico de Calar Alto para seleccionar la región de interés de las imágenes obtenidas con sus telescopios así como identificar objetos en ella, calcular áreas y centroides o separar el fondo y el primer plano de la misma.

Lo que se pretende es poder realizar las mismas tareas que se harían con alguno de los programas que usan a diario los astrónomos, ya descritos en el segundo apartado de este documento, pero utilizando la herramienta Matlab®, ya que es un programa más accesible y con un lenguaje sencillo y cómodo.

Puesto que el objetivo final es usar dicho software para la obtención de imágenes astronómicas, en un futuro sería interesante añadir a la aplicación las distintas funciones integradas en IRAF o IDL para conseguir imágenes comparables a las que se obtienen usando dichos métodos.

BIBLIOGRAFÍA

[1] http://www.aero.us.es/move

[2] http://www.astronum.net/astronum/temas/darks/darks.htm

[3] https://es.mathworks.com/help/images/ref/adaptthresh.html

[4] https://es.mathworks.com/help/images/

[5] C:\Users\beita\OneDrive\Documents\TFG\Documentos\Vision Artificial y Procesamiento Digital de Imagenes Usando Matlab.html

[6] http://jc-info.blogspot.com/2011/02/suavizamiento-filtro-mediana-codigo.html

[7] https://carretedigital.com/blog/fotografia-astronomica/

[8] http://www6.uniovi.es/vision/intro/node38.html

[9] https://es.mathworks.com/help/images/ref/medfilt2.html?s_tid=srchtitle

[10] http://www.astrosurf.com/astronosur/pixinsight/preprocessing-1.htm

[11] http://www.astrosurf.com/astronosur/deepskystacker_tutorial.html

[12] https://antonioheras.com/historia_de_astronomia/principios-astronomia.htm

[13] https://www.astromia.com/historia/

[14] http://feinstein.com.ar/LaAstronomiadetodoslosdias.html

[15] https://www.geoenciclopedia.com/observatorios-astronomicos/

[16] https://www.vix.com/es/btg/curiosidades/5646/5-observatorios-que-todo-amante-de-la-astronomia-deberia-conocer

[17] http://www.redalyc.org/pdf/1794/179414895002.pdf

[18] http://www.telescopios.org/

[19] http://www.caha.es/

[20] http://www.astronomiadecampo.com/procesamiento-de-imagenes-astronomicas/

[21] http://misistemasolar.com/como-funciona-un-telescopio/

[22]http://www.aaquarks.com/web/Asociacion_Astronomica_Quarks/Portada/Entradas/2010/3/1_Imagenes_en_la_astronomia_profesional.html

[23] http://imagenes-pablo.blogspot.com/2010/06/ejercicio-filtro-de-la-mediana_10.html

[24] http://www.viguri.org/astro/docs/tutorial_procesamiento/procesamiento.htm

[25] https://es.mathworks.com/help/images/roi-based-processing.html

[26] https://es.mathworks.com/help/images/building-guis-with-modular-interactive-tools.html

[27] http://mathforum.org/kb/message.jspa?messageID=6167831

[28] http://www.utm.mx/~vero0304/HCPM/GUI_Matlab.pdf

[29] https://es.mathworks.com/help/images/ref/regionprops.html

[30] https://fits.gsfc.nasa.gov/fits_viewer.html

[31] http://ds9.si.edu/doc/ref/file.html

[32] https://graffica.info/fits-formato-de-archivo/

[33] http://ds9.si.edu/doc/ref/mfile.html

[34] https://es.mathworks.com/help/matlab/ref/fitsread.html

[35] https://www.spacetelescope.org/projects/fits_liberator/datasets_archives/

[36] http://hosting.astro.cornell.edu/academics/courses/astro3310/Matlab_images.html

[37] https://www.spacetelescope.org/projects/fits_liberator/download_v301/

[38] http://chandra.harvard.edu/photo/openFITS/

[39] https://www.spacetelescope.org/projects/fits_liberator/stepbystep/

[40] https://www.youtube.com/watch?v=WW78DbvK-Sc

[41] https://www.lawebdelprogramador.com/foros/Matlab/1313910-superposicion-de-2-imagen-en-matlab.html

[42] https://es.mathworks.com/help/matlab/ref/rgb2gray.html

[43] https://es.mathworks.com/matlabcentral/answers/342072-how-to-create-an-rgb-image

[44] https://es.mathworks.com/help/matlab/ref/colorspec.html

[45] https://es.calameo.com/read/0001888059ae636fd6c5f

[46] https://es.mathworks.com/help/images/ref/imadjust.html

[47] https://www.youtube.com/watch?v=zcGPmZK1QOw

ANEXO. CÓDIGO DEL PROGRAMA

Para finalizar se adjuntan los códigos descritos en Matlab® para la elaboración de la aplicación.

Se ha dividido en tres archivos que corresponden con las tres interfaces gráficas: portada.m, ventana_roi.m y background.m

- Portada:

```
function varargout = portada(varargin)
% PORTADA MATLAB code for portada.fig
%      PORTADA, by itself, creates a new PORTADA or raises the existing
%      singleton*.
%
%      H = PORTADA returns the handle to a new PORTADA or the handle to
%      the existing singleton*.
%
%      PORTADA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PORTADA.M with the given input arguments.
%
%      PORTADA('Property','Value',...) creates a new PORTADA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before portada_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to portada_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help portada

% Last Modified by GUIDE v2.5 17-Aug-2019 12:23:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @portada_OpeningFcn, ...
                   'gui_OutputFcn',  @portada_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before portada is made visible.
function portada_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to portada (see VARARGIN)

i=imread('portada.jpg');
image(i)
axis off

% Choose default command line output for portada
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes portada wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = portada_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes during object creation, after setting all properties.


% --- Executes on button press in guardar.

% --- Executes on button press in comenzar.
function comenzar_Callback(hObject, eventdata, handles)
% hObject    handle to comenzar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(portada);
ventana_roi


% --- Executes during object creation, after setting all properties.
function etsius_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etsius (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: place code in OpeningFcn to populate etsius
i=imread('logo_etsi_us.png');
image(i)
guidata(hObject,handles);
axis off


% --- Executes during object creation, after setting all properties.
function fisica_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fisica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate fisica
i=imread('logo_fisica_etsi.jpg');
image(i)
guidata(hObject,handles);
axis off


% --- Executes during object creation, after setting all properties.
function caha_CreateFcn(hObject, eventdata, handles)
% hObject    handle to caha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate caha
i=imread('logo_calar_alto.jpg');
image(i)
guidata(hObject,handles);
axis off
```

- Ventana roi:

```
function varargout = ventana_roi(varargin)
% VENTANA_ROI MATLAB code for ventana_roi.fig
%      VENTANA_ROI, by itself, creates a new VENTANA_ROI or raises the
existing
%      singleton*.
%
%      H = VENTANA_ROI returns the handle to a new VENTANA_ROI or the handle
to
%      the existing singleton*.
%
%      VENTANA_ROI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VENTANA_ROI.M with the given input
arguments.
%
%      VENTANA_ROI('Property','Value',...) creates a new VENTANA_ROI or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ventana_roi_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ventana_roi_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ventana_roi

% Last Modified by GUIDE v2.5 28-Aug-2019 18:18:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ventana_roi_OpeningFcn, ...
                   'gui_OutputFcn',  @ventana_roi_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ventana_roi is made visible.
function ventana_roi_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ventana_roi (see VARARGIN)
```

```
axes(handles.axes1);
axis off
axes(handles.axes2);
axis off

% Choose default command line output for ventana_roi
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ventana_roi wait for user response (see UIRESUME)
% uiwait(handles.figure1);

function setGlobalModo(val)
global modo;
modo = val;

function r =getGlobalModo
global modo;
r = modo;



% --- Outputs from this function are returned to the command line.
function varargout = ventana_roi_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in cargar.
function cargar_Callback(hObject, eventdata, handles)
% hObject    handle to cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[im,direc]= uigetfile({'*.jpg; *.tif'});
cargar= strcat(direc, im);
im=imread(cargar);
axes(handles.axes1);
imshow(im);
cla(handles.axes2); %borra imagen derecha
set(handles.cargar, 'USerData', im); %guardar imagen


% --- Executes on button press in Poligono.
function Poligono_Callback(hObject, eventdata, handles)
% hObject    handle to Poligono (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of Poligono

% --- Executes on button press in Dibujo.
function Dibujo_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Dibujo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of Dibujo


% --- Executes on button press in Rectangulo.

function roi_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in roi
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%global modo; %"modo" sirve para todo el programa

if hObject==handles.poligono
    setGlobalModo(1);
end
if hObject==handles.dibujo
    setGlobalModo(2);
end

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes during object creation, after setting all properties.


% --- Executes on button press in guardar.
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

guardar=getimage(handles.axes2);
if isempty(guardar)
    disp("No hay imagen a guardar");
    return
end
formatos={'*.jpg','JPEG(*.jpg)';'*.tiff','TIFF(*.tif)'};
[nomb,ruta]=uiputfile(formatos,'Guardar imagen');
if nomb==0
    return
end
fName=fullfile(ruta,nomb);
imwrite(guardar,fName);
disp("Imagen guardada");

% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
im= get (handles.cargar, 'UserData');
```

```
if getGlobalModo==1;
    disp("Empezar a pintar ROI de polígono");
    h=roipoly(im);
    %reemplazo parte blanca de la máscara con la imagen original
    [F C]=size(h);
    for i=1:F;
        for j=1:C;
            if h(i,j)==1 %la seccion del ROI tiene pixeles de valor 1
(blanco)
            roi(i,j)=im(i,j);
            else
            roi(i,j)=0;
            end
        end
    end
    maskedRgbImage = bsxfun(@times, im, cast(h, 'like', im));
    axes(handles.axes2);
    imshow(maskedRgbImage);
    guidata(hObject,handles);
end
if getGlobalModo==2
    disp("Empezar a pintar ROI de mano alzada");
    k=imfreehand(handles.axes1);
    wait(k);
    mask=createMask(k);
    maskedRgbImage = bsxfun(@times, im, cast(mask, 'like', im));
    axes(handles.axes2);
    imshow(maskedRgbImage);
    guidata(hObject,handles);
end


% --- Executes during object creation, after setting all properties.
function cargar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in procesar.
function procesar_Callback(hObject, eventdata, handles)
% hObject    handle to procesar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(ventana_roi);
background
```

- Background:

```
function varargout = background(varargin)
% BACKGROUND MATLAB code for background.fig
%      BACKGROUND, by itself, creates a new BACKGROUND or raises the existing
%      singleton*.
%
%      H = BACKGROUND returns the handle to a new BACKGROUND or the handle to
%      the existing singleton*.
%
%      BACKGROUND('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BACKGROUND.M with the given input
arguments.
%
%      BACKGROUND('Property','Value',...) creates a new BACKGROUND or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before background_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to background_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help background

% Last Modified by GUIDE v2.5 18-Aug-2019 16:52:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @background_OpeningFcn, ...
                   'gui_OutputFcn',  @background_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before background is made visible.
function background_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to background (see VARARGIN)

axes(handles.axes1);
```

```
axis off

% Choose default command line output for background
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes background wait for user response (see UIRESUME)
% uiwait(handles.contraste);


% --- Outputs from this function are returned to the command line.
function varargout = background_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in fondo.
function fondo_Callback(hObject, eventdata, handles)
% hObject    handle to fondo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in objetos.
function objetos_Callback(hObject, eventdata, handles)
% hObject    handle to objetos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% %guardar imagen en axes1
% imagen=getimage(handles.axes1);
% if isempty(imagen)
%     return
% end
% formatos={'*.jpg','JPEG(*.jpg)';'*.tiff','TIFF(*.tif)'};
% [nomb,ruta]=uiputfile(formatos,'Guardar imagen');
% if nomb==0
%     return
% end
% fName=fullfile(ruta,nomb);
% imwrite(imagen,fName);

im3=getimage(handles.axes1);
im4=double(im3);
bw=imbinarize(im3); %umbral de la imagen
bw2=bwareaopen(bw,50); %eliminar ruido de fondo, elimina objetos de menos de
50 pixeles
cc=bwconncomp(bw2);
aTable=struct2table(cc);
disp(aTable);
xlswrite(strcat('Objetos-',date,'.xlsx'),aTable.NumObjects) %crea excell con
objetos
disp(strcat("Excel para N° Objetos guardado en: ",'Objetos-',date,'.xlsx'));
guidata(hObject,handles);
```

```
% --- Executes on button press in examinar.
function examinar_Callback(hObject, eventdata, handles)
% hObject    handle to examinar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
im3=getimage(handles.axes1);
bw=imbinarize(im3); %umbral de la imagen
bw2=bwareaopen(bw,50); %eliminar ruido de fondo, elimina objetos de menos de
50 pixeles
cc=bwconncomp(bw2);
%aTable=struct2table(cc);
%star=false(size(bw2));
%star(cc.PixelIdxList{50})=true; %solo examina un componente
%ver todos los objetos
L=labelmatrix(cc);
%whos L
L2=rgb2gray(L);
rgb=label2rgb(L2);
axes(handles.axes1);
disp("Enseña objetos en la imagen");
imshow(rgb)

% --- Executes on button press in area.
function area_Callback(hObject, eventdata, handles)
% hObject    handle to area (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

im3=getimage(handles.axes1);
bw=imbinarize(im3); %umbral de la imagen
bw2=bwareaopen(bw,50); %eliminar ruido de fondo, elimina objetos de menos de
50 pixeles
cc=bwconncomp(bw2);
stardata=regionprops(cc,'basic')
%stardata(5).Area; %medida para un objeto concreto

%medida de area para cada uno
star_areas=[stardata.Area];
xlswrite(strcat('Areas-',date,'.xlsx'),star_areas) %crea excell con areas
disp(strcat("Excel para Areas guardado en: ",'Areas-',date,'.xlsx'));
%encontrar el más pequeño
[min_area,idx]=min(star_areas)


% --- Executes on button press in histograma.
function histograma_Callback(hObject, eventdata, handles)
% hObject    handle to histograma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

im3=getimage(handles.axes1);
bw=imbinarize(im3); %umbral de la imagen
bw2=bwareaopen(bw,50); %eliminar ruido de fondo, elimina objetos de menos de
50 pixeles
cc=bwconncomp(bw2);
stardata=regionprops(cc,'basic')
%stardata(5).Area; %medida para un objeto concreto

%medida de area para cada uno
```

```
star_areas=[stardata.Area];

nstar=50;
figure
histogram(star_areas, nstar)
title('Histograma areas/n°puntos')


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider

imagen=get (handles.ok, 'UserData');
val=get(hObject,'Value');
contraste=imadjust(imagen,[],[],val);
axes(handles.axes1);
imshow(contraste);

guidata(hObject,handles)
set(handles.slider1, 'USerData', contraste);

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes on button press in cargar.
function cargar_Callback(hObject, eventdata, handles)
% hObject    handle to cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[im,direc]= uigetfile({'*.jpg; *.tif'});
cargar= strcat(direc, im);
im=imread(cargar);
axes(handles.axes1);
imshow(im);
set(handles.cargar, 'USerData', im); %guardar imagen
```

```
guidata(hObject,handles);
% --- Executes on button press in primerplano.
function primerplano_Callback(hObject, eventdata, handles)
% hObject    handle to primerplano (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when selected object is changed in mostrar.
function mostrar_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in mostrar
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global modo; %"modo" sirve para todo el programa

if hObject==handles.fondo
    modo=1;
end
if hObject==handles.primerplano
    modo=2;
end


% --- Executes on button press in ok.
function ok_Callback(hObject, eventdata, handles)
% hObject    handle to ok (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global modo;
im= get (handles.cargar, 'UserData');

if modo==1;
    %abertura morfologica
    background=imopen(im,strel('disk',15));
    %sustraer imagen
    im2=im-background;
    axes(handles.axes1);
    imshow(im2)
end
if modo==2;
    %abertura morfologica
    background=imopen(im,strel('disk',15));
    %sustraer imagen
    im2=background;
    axes(handles.axes1);
    imshow(im2)
end
im2=getimage(handles.axes1);
guidata(hObject,handles);
set(handles.ok, 'USerData',im2);


% --- Executes on button press in centroide.
function centroide_Callback(hObject, eventdata, handles)
% hObject    handle to centroide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
im3=getimage(handles.axes1);
bw=imbinarize(im3); %umbral de la imagen
```

```
bw2=bwareaopen(bw,50); %eliminar ruido de fondo, elimina objetos de menos de
50 pixeles
cc=bwconncomp(bw2);
stardata=regionprops(cc,'basic')
%stardata(5).Area; %medida para un objeto concreto

%medida de area para cada uno
star_centroide=[stardata.Centroid];
xlswrite(strcat('Centroides-',date,'.xlsx'),star_centroide) %crea excell con
centroide
disp(strcat("Excel para Centroides guardado en: ",'Centroides-
',date,'.xlsx'));
%encontrar el más pequeño
[min_centroide,idx]=min(star_centroide)


% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(background);
```

- Conversor:

```
%conversor de unidades
clear all; clc;
%longitud de onda
lambda=double(input(['Insertar longitud de onda en:\n1. nm \n2. mum \n3. Å
\n']))

switch lambda
    case 1
        a=double(input('Ingrese dato en nm:'));
        b=a*1000;
        c=a/10;
        sprintf('%.4f nm equivalen a %.4f mum y %.4f Å',a,b,c)
    case 2
        d=double(input('Ingrese dato en  mum:'));
        e=d/1000;
        g=d/10000;
        sprintf('%.4f mum equivalen a %.4f nm y %.4f Å',d,e,g)
    case 3
        h=double(input('Ingrese dato en Å:'));
        i=h*10;
        j=h*10000;
        sprintf('%.4f Å equivalen a %.4f nm y %.4f mum',h,i,j)
end

%frecuencia
f=double(input(['Insertar frecuencia en:\n1. Hz \n2. 1/min \n']))
switch f
    case 1
        s=double(input('Ingrese dato en Hz:'));
        l=s*60;
        sprintf('%.4f Hz equivalen a %.4f 1/min',s,l)
    case 2
        m=double(input('Ingrese dato en 1/min:'));
        n=m/60;
        sprintf('%.4f 1/min equivalen a %.4f Hz',m,n)
end

%energia
E=double(input(['Insertar energia en:\n1. eV \n2. J \n']))
switch E
    case 1
        o=double(input('Ingrese dato en eV:'));
        p=o/1.60219e-19;
        sprintf('%.4f eV equivalen a %.4f J',o,p)
    case 2
        q=double(input('Ingrese dato en J:'));
        r=q*1.60219e-19;
        sprintf('%.4f J equivalen a %.4f eV',q,r)
end

%numero de onda
k=double(input(['Insertar numero de onda en:\n1. 1/cm \n2. 1/m \n']))
switch k
    case 1
        t=double(input('Ingrese dato en 1/cm:'));
        u=t*100;
        sprintf('%.4f 1/cm equivalen a %.4f 1/m',t,u)
    case 2
```

```
        v=double(input('Ingrese dato en 1/m:'));
        w=v/100;
        sprintf('%.4f 1/m equivalen a %.4f 1/cm',v,w)
end
```