

AstroIMP Documentación

Eduardo Bojórquez Martínez¹

¹Departamento de Ingeniería Industrial, Universidad de Sonora, Hermosillo, Sonora.

8 de junio de 2024

Resumen

AstroIMP es una herramienta para el preprocesamiento automatizado de imágenes astronómicas, ya que la calidad de las imágenes suele ser baja en comparación con otras imágenes tradicionales. Algunas funciones son mejorar el contraste, recortar imágenes, reducir el ruido, reducir los errores *CCD*, eliminar el fondo y quedarse solo con los objetos de interés para detectar objetos automáticamente con algún algoritmo de segmentación de imágenes. Este documento tiene la intención de explicar cómo instalarlo y utilizarlo.

1. Introducción

Desde la aparición de la Inteligencia Artificial, comúnmente abreviada como *AI*, tiene como objetivo resolver problemas complejos de forma inteligente. Existen diversas definiciones de *IA*, Ben Coppin la define como: “El uso de métodos basados en el comportamiento inteligente de los humanos y otros animales para resolver problemas complejos”.

Una de las áreas de estudio de la *AI* es el reconocimiento de patrones, que podemos definir como la disciplina de la *AI* que tiene como objetivo clasificar un patrón en varias categorías o clases; Es decir, cómo las computadoras pueden observar el entorno y aprender a distinguir entre diferentes patrones, para tomar decisiones en función de sus categorías.

El reconocimiento de objetos es la tarea de identificar un objeto que se encuentra en una imagen o video. Uno de los pasos más importantes en el procesamiento de imágenes digitales es el preprocesamiento.

Para el paso de preprocesamiento se elimina todo lo que pueda generar ruido, se mejora la imagen para facilitar el análisis de la imagen y existen diferentes técnicas de mejora de la imagen. En las imágenes astronómicas, las estructuras extendidas normalmente están rodeadas de objetos puntuales (estrellas), estos últimos deben eliminarse o recortarse en los valores máximos, para evitar que causen ruido en la identificación de nebulosas de emisión y permitir una detección más fácil. Sin embargo, cuando las imágenes tienen mucho ruido, los bordes de los objetos circundantes también pueden aparecer borrosos. En algunas ocasiones, tras realizar el proceso de reducción de ruido, estos pueden verse distorsionados, provocando que los resultados sean inexactos o erróneos.

2. Arquitectura del Software

La arquitectura se divide en dos módulos: el primero es la manipulación de imágenes y el segundo es el preprocesamiento de imágenes.

El módulo de manipulación de imágenes tiene dos funciones, mejora del contraste de la imagen y recorte de imágenes.

La primera funcionalidad de contraste de imágenes tiene como objetivo manipular los datos aumentando el rango dinámico de intensidades en imágenes de bajo contraste.

La segunda funcionalidad, recortar imágenes, tiene como objetivo crear múltiples subimágenes a partir de la imagen original. Estas funcionalidades se describen en la subsección funcionalidades del software.

El preprocesamiento del módulo tiene tres funciones: reducción de ruido, reducción de errores *CCD* y eliminación de fondo.

Para la primera funcionalidad de reducción de ruido, hicimos una revisión de la literatura y encontramos que una de las formas más sencillas de reducir el ruido es a través del filtro gaussiano y mediano, en varios artículos utilizaron la combinación de ambos filtros para reducir el ruido gaussiano y eliminarlo. de los picos provocados por el ruido de Poisson. En varios artículos aplicaron operaciones morfológicas para reducir el ruido, mejorar las características de los objetos brillantes y reducir los halos. Finalmente, se utilizó el filtro de difusión anisotrópica para reforzar los bordes de los objetos y

eliminar estrellas distantes en imágenes astronómicas. Dado que el filtro de difusión anisotrópica tiene excelentes resultados, lo aplicamos para mejorar los bordes de las fuentes puntuales y los objetos extendidos.

La segunda funcionalidad es la reducción de errores del *CCD*, este proceso solo se aplica a las imágenes que se obtuvieron del *CCD* número 3 del *IPHAS* ya que es la que presenta daño al *CCD* columnas. Los errores *CCD* son sólo columnas de píxeles que no funcionan. Hay varias razones por las que una columna puede dañarse (desde electrónicas hasta mecánicas); algunas columnas están muertas (oscuras), algunas son brillantes y algunas tienen una constante agregada a lo que deberían mostrar. Un buen *CCD* normalmente no tiene más de un par de ellos. Son fáciles de arreglar cuando están aisladas (las columnas vecinas se interpolan), pero no se puede hacer nada cuando hay varias columnas dañadas juntas. Aplicamos el operador morfológico matemático *close* para reducir los errores *CCD*. Una vez que se completa este proceso, la imagen se invierte y se devuelve al rango dinámico original; el proceso detallado se describe en la siguiente subsección.

Para la tercera funcionalidad de eliminación del fondo, para eliminarlo, el primer paso es la adquisición de imágenes, el segundo paso es la normalización de la imagen con min-max, el tercero es aplicar el filtro de difusión anisotrópica introducido por primera vez por Perona-Malik, el cuarto es la normalización con min. -max para el algoritmo difuso *PFCM* propuesto por primera vez por Bezdek en 2005 y es una combinación de *PCM* introducido por Krishnapuram y Keller en 1993 y *FCM* propuesto por Bezdek en 1981. En esta versión, La función objetivo se puede obtener mediante un proceso iterativo en el que la membresía de la función, los valores de tipicidad y los centros de grupo se actualizan simultáneamente para cada grupo. Esta es una hibridación de algoritmos *PCM* y *FCM* que resuelven el problema de coincidencia de grupos en *PCM* y la sensibilidad al ruido de *FCM*.

3. Requisitos para instalar

3.1. Cómo instalar anaconda y astropy

Anaconda es una distribución de código abierto que incluye paquetes para procesar grandes volúmenes de información, realizar análisis predictivos y computación científica, es una de las formas fáciles de realizar ciencia de datos y aprendizaje automático en una sola máquina.

La distribución Anaconda está disponible en el sitio oficial para Windows, Mac y Linux:

<https://www.anaconda.com/download/success>

Descargue el instalador para su sistema operativo y luego siga los pasos:

Para Windows, no lo instale como administrador a menos que se requieran privilegios de administrador:

1. Una vez completada la descarga, debe ir a la carpeta Descargas y hacer doble clic en el instalador para iniciarlo.
2. Luego haga clic en Siguiente.
3. Lea los términos de la licencia y haga clic en Acepto.
4. Después de eso, debe seleccionar quién se instalará; se recomienda instalar Just Me, que instalará Anaconda Distribution solo en la cuenta de usuario actual. Seleccione una instalación para Todos los usuarios solo si necesita realizar la instalación para todas las cuentas de usuarios en la computadora (lo que requiere privilegios de administrador de Windows).
5. Haga clic en Siguiente.
6. Seleccione una carpeta de destino para instalar Anaconda y haga clic en Siguiente. Instale Anaconda en una ruta de directorio que no contenga espacios ni caracteres Unicode.
7. Elija si desea agregar Anaconda a su variable de entorno PATH o registrar Anaconda como su Python predeterminado. No recomendamos agregar Anaconda a su variable de entorno PATH, ya que esto puede interferir con otro software. A menos que planea instalar y ejecutar varias versiones de Anaconda o varias versiones de Python, acepte el valor predeterminado y deje esta casilla marcada. En su lugar, utilice el software Anaconda abriendo Anaconda Navigator o Anaconda Prompt desde el menú Inicio.
8. Haga clic en Instalar. Si desea ver los paquetes que Anaconda está instalando, haga clic en Mostrar detalles.
9. Haga clic en Siguiente.

10. Opcional: para obtener más información sobre el servicio de portátiles en la nube de Anaconda, visite <https://www.anaconda.com/code-in-the-cloud>. O haga clic en Continuar para continuar.
11. Después de una instalación exitosa, verá el cuadro de diálogo "Gracias por instalar Anaconda".

Para Mac:

1. Descargue el instalador gráfico (.pkg) de macOS para su sistema.
2. Haga doble clic en el archivo descargado y haga clic en Continuar para iniciar la instalación.
3. Responda las indicaciones en las pantallas Introducción, Léame y Licencia.
4. Anaconda recomienda que elija Instalar para todos los usuarios de esta computadora. A partir de la versión 2024.02-1, la ubicación predeterminada del instalador es /opt/anaconda3. Para instalar en otro lugar, seleccione Instalar en un disco específico.
5. Haga clic en Instalar.
6. Una vez que se complete la instalación, haga clic en Continuar.
7. Opcional: para obtener más información sobre el servicio de portátiles en la nube de Anaconda, visite <https://www.anaconda.com/code-in-the-cloud>. O haga clic en Continuar para continuar.
8. Una instalación exitosa muestra el agradecimiento por instalar Anaconda Distribution.

Astropy es una colección de paquetes de software escritos en el lenguaje de programación Python y diseñados para su uso en astronomía. El software es un paquete central único y gratuito para servicios astronómicos debido al uso cada vez más generalizado de Python por parte de los astrónomos y para fomentar la interoperabilidad entre varios paquetes de astronomía de Python existentes.

Para instalar Astropy, puede utilizar el siguiente comando en el indicador de Anaconda:

```
conda install -c conda-forge astropy
```

3.2. Cómo instalar los requisitos previos AstroIMP

3.2.1. OpenCV

La Biblioteca de Visión por Computador de Código Abierto, abreviada como OpenCV, es una biblioteca de funciones de programación principalmente para visión por computadora en tiempo real. En el campo de la computación visual y el procesamiento de imágenes, OpenCV se ha consolidado como una herramienta fundamental. Esta biblioteca de código abierto ofrece una amplia gama de funcionalidades que permiten desde la manipulación básica de imágenes hasta el desarrollo de complejos sistemas de reconocimiento facial y seguimiento de objetos. Fue desarrollada originalmente por Intel, la biblioteca es multiplataforma y tiene licencia de software gratuito y de código abierto bajo la licencia Apache 2. A partir de 2011, OpenCV presenta aceleración de GPU para operaciones en tiempo real.

Para instalar la biblioteca OpenCV para Python, puedes hacerlo con el siguiente comando:

```
pip install opencv-python
```

3.2.2. Imutils

Imutils es una biblioteca para Python con una serie de funciones convenientes para realizar funciones básicas de procesamiento de imágenes como traducción, rotación, cambio de tamaño, esqueletización, visualización de imágenes Matplotlib, clasificación de contornos, detección de bordes y mucho más fácil con OpenCV.

Para instalar la biblioteca imutils, puedes hacerlo con el siguiente comando:

```
pip install imutils
```

4. Cómo utilizar AstroIMP

4.1. Especificar el directorio de entrada y salida.

Dado que se requiere en todas las funciones para proporcionar un directorio con un conjunto de imágenes, primero es necesario especificar el directorio de entrada; es decir, las imágenes que se desean analizar y el directorio de salida para guardar los resultados obtenidos dependiendo de la función elegida.

Para especificar el directorio de entrada de las imágenes:

```
python principal.py -d «directorio de entrada»
```

Para especificar el directorio de salida de las imágenes:

```
python principal.py -r «directorio de salida»
```

4.2. Especificar imágenes de recorte

Dado un directorio con un conjunto de imágenes y el tamaño deseado especificado para crear múltiples subimágenes a partir de la imagen original. La herramienta puede generar subimágenes con el tamaño especificado para el ancho de cada subimagen, la altura será del mismo tamaño que el ancho especificado. Las imágenes de prueba que utilizamos tienen un tamaño de 2048 de ancho y 4096 de alto en píxeles, y el tamaño deseado fue de 512 píxeles, generó 32 subimágenes con un tamaño de 512 x 512 píxeles.

Para recortar imágenes puedes usar el argumento:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -c
```

4.3. Especificar preproceso automático

4.3.1. Mejorar contraste

La mejora de imágenes es un área del procesamiento de imágenes que consta de una colección de métodos dependientes de la tarea que acentúan características importantes para un objetivo específico, haciendo que el análisis humano o automatizado sea más efectivo. Las imágenes astronómicas contienen frecuentemente muchos objetos puntuales y, al mismo tiempo, estructuras extensas y difusas con objetos puntuales incrustados; Generalmente, todas las imágenes se ven afectadas por el ruido debido al proceso de adquisición de los dispositivos *CCD*, por lo que la reducción de ruido es uno de los pasos necesarios para la detección de diferentes objetos.

Para mejorar el contraste y eliminar el fondo, primero es necesario especificar el argumento -p.

Para especificar si va a preprocesar las imágenes:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -p
```

El algoritmo *ZScale* se introdujo por primera vez en el software *IRAF* y se actualizó para Python con el nombre *PyRAF*. Está diseñado para evitar la laboriosa tarea de calcular un histograma de imagen completo mostrando valores de imagen cercanos al valor medio de la imagen. Las imágenes

astronómicas suelen tener un histograma extremadamente puntiagudo que coincide con el continuo en un espectro bidimensional o con el cielo de fondo en imágenes directas, lo que puede beneficiarse enormemente de esto.

Especifica si se aplica mejora de contraste, utilizando el algoritmo `zscale`:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -p  
-zs
```

El algoritmo rango percentil calcula el rango de valores de píxeles que se utilizarán al ajustar el contraste de las imágenes FIT mediante un corte percentil simple. Se utiliza una submuestra de los datos de la imagen de entrada por razones de eficiencia.

Especifica si se aplica mejora de contraste, utilizando el rango percentil del algoritmo:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -p  
-pr
```

El algoritmo `arcsinh` percentil ubica los límites inferior y superior que contienen todos los objetos visibles utilizando un `arcsinh` de mapeo que muestra los valores de la imagen.

Especifica si se aplica mejora de contraste, utilizando el algoritmo `arcsinh` percentil:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -p  
-ap
```

4.3.2. Reducción de errores CCD y reducción de picos de ruido elevados

En la herramienta implementamos la reducción de errores *CCD*, los pasos que aplicamos se basan en el trabajo de Candéas en 1999 utilizan operadores morfológicos para realzar los picos que se encuentran contenidos en la imagen, para luego eliminarlos. Los pasos detallados que se siguieron son:

- Invierte la imagen con su rango dinámico original. La revisión de la literatura menciona que es una buena práctica en astronomía porque se mejora el contraste de la imagen y es más fácil de visualizar para el ojo humano.
- Restaura la imagen original con la invertida. Candéas, llevó a cabo este procedimiento para mejorar la imagen invertida y reducir los efectos ópticos de los halos alrededor de estrellas brillantes.

- Aplicar el operador morfológico matemático close. Al utilizar este operador, se elimina todo el ruido interno presente en la región del objeto y el fondo no se ve afectado. Esta técnica también tiene la propiedad de resaltar los picos contenidos en la imagen. Nos interesan aquellos con píxeles saturados, y efecto floración, entre otros. Para lograr esto, se utilizó una estructura tipo disco con un radio $r = 8$ para mejorar la imagen invertida y resaltar los picos altos contenidos en la imagen.
- Una vez completada la aplicación del operador close, la imagen se invierte nuevamente y el resultado se resta de la imagen original.
- Finalmente, se devuelve a su rango dinámico original.

Para especificar si desea reducir los errores de CCD:

```
python principal.py -d ¡directorio de entrada! -r ¡directorio de salida! -p
-ccde
```

Debido al rango dinámico de los objetos puntuales y extendidos, los tipos de ruido descritos en las secciones anteriores y los errores del *CCD*, es necesario eliminar el fondo de la imagen para mejorar la detección de objetos extendidos con baja emisión. . Para lograr la eliminación del fondo, se determina un umbral con la ayuda del algoritmo *PFCM*.

4.3.3. Eliminación de fondo

Para la eliminación de fondo utiliza el algoritmo *PFCM*, traducido a python por Djemai en 2018, con este algoritmo generamos dos grupos, el objetivo es dividir el fondo de la imagen y los objetos extendidos o estrellas que se encuentran en la imagen, con el resultado obtenido se genera una máscara en la que al fondo se le asigna el valor 0, y a los objetos se le asigna el valor 1. Luego se genera una nueva imagen multiplicando los valores de la máscara generada con la imagen original, en de esta forma mantenemos los valores originales de la imagen, y el resto queda con un valor de 0 (negro).

Para especificar la eliminación del fondo utilizando el algoritmo *PFCM*:

```
python principal.py -d «directorio de entrada» -r «directorio de salida» -p
-pf
```

5. Ejemplos

Los siguientes ejemplos fueron ejecutados con el IDE Spyder incluido con Anaconda; sin embargo, en la consola o símbolo de sistema, se puede ejecutar con el comando `python` el nombre del archivo `principal.py` y los argumentos especificados con anterioridad.

5.1. Mejora de contraste

5.1.1. Ejemplo de uso de algoritmo ZScale

Para el ejemplo, se han creado dos carpetas como se muestra en la figura 1, en la que se creó la carpeta `imagenes_fits` dentro del mismo directorio donde se encuentra el código fuente, y también la carpeta `resultados`.































	<code>.idea</code>		06/06/2024 02:30 p. m.	Carpeta de archivos	
	<code>__pycache__</code>		21/05/2024 01:30 p. m.	Carpeta de archivos	
	<code>imagenes_fits</code>		06/06/2024 02:23 p. m.	Carpeta de archivos	
	<code>resultados</code>		06/06/2024 06:42 p. m.	Carpeta de archivos	
	<code>__init__.py</code>		26/02/2020 04:18 p. m.	Archivo PY	1 KB
	<code>dice_coef_pfcmm.py</code>		22/03/2024 03:34 p. m.	Archivo PY	1 KB
	<code>gui.py</code>		06/06/2024 11:35 a. m.	Archivo PY	2 KB
	<code>histogram.py</code>		28/03/2024 10:21 a. m.	Archivo PY	1 KB
	<code>image.py</code>		20/05/2024 05:08 p. m.	Archivo PY	15 KB
	<code>line.py</code>		18/04/2024 01:48 p. m.	Archivo PY	6 KB
	<code>PFCM.py</code>		07/04/2024 10:43 a. m.	Archivo PY	6 KB
	<code>pointarray.py</code>		18/04/2024 01:50 p. m.	Archivo PY	14 KB
	<code>preprocess.py</code>		20/05/2024 06:24 p. m.	Archivo PY	22 KB
	<code>principal.py</code>		06/06/2024 02:44 p. m.	Archivo PY	4 KB
	<code>rename_files.py</code>		20/05/2024 05:53 p. m.	Archivo PY	1 KB

Figura 1: Listado de archivos y carpetas del directorio del código fuente.

Dentro de la carpeta `imagenes_fits` se han colocado 5 imágenes en formato `fits`, como se muestra en la figura 2.











	PN_NGC_7048_Ha_1.fits		29/08/2020 08:29 p. m.	Archivo FITS	16,403 KB
	PN_NGC_7354_1_Ha.fits		29/08/2020 08:29 p. m.	Archivo FITS	16,403 KB
	PN_SH_1-89_Ha_4.fits		29/08/2020 08:29 p. m.	Archivo FITS	16,400 KB
	PN_SH_2-71_Ha_2.fits		29/08/2020 08:29 p. m.	Archivo FITS	16,400 KB
	PN_SH_2-188_Ha_1.fits		29/08/2020 08:29 p. m.	Archivo FITS	16,403 KB

Figura 2: Listado de archivos del directorio imagenes_fits.

En la consola o simbolo del sistema en windows, ejecutamos la siguiente instrucción para hacer uso del algoritmo ZScale.

```
C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_fits -r resultados -p -zs
Preprocess...
PN_NGC_7048_Ha_1.fits
Contrast enhancement with zscale...
PN_NGC_7354_1_Ha.fits
Contrast enhancement with zscale...
PN_SH_1-89_Ha_4.fits
Contrast enhancement with zscale...
PN_SH_2-188_Ha_1.fits
Contrast enhancement with zscale...
PN_SH_2-71_Ha_2.fits
Contrast enhancement with zscale...
```

Figura 3: Ejecución de la herramienta en el simbolo de sistema de windows, en el ejemplo se ejecuta la instrucción -zs para hacer uso del algoritmo ZScale.

En la figura 3, se muestra el ejemplo de cómo ejecutar la herramienta en el simbolo del sistema de windows. Primeramente, se accedió a la carpeta donde está el código fuente, y se ejecuta principal.py, le proporcionamos el directorio de entrada y salida, en este caso, se pusieron las carpetas imagenes_fits y resultados, y debido a que queremos ejecutar ZScale, requerimos el argumento -p seguido de -zs; por lo tanto, la instrucción quedaría de la siguiente manera:

python principal.py -d imagenes_fits -r resultados -p -zs

Después de ejecutar dicho comando, los resultados se guardan en la carpeta resultados. En la figura 4, se muestran los archivos del directorio resultados, la herramienta generará una imagen tif y otra con el resultado obtenido después de aplicar el ajuste de contraste con ZScale.

5.1.2. Ejemplo de uso del algoritmo percentile range

En la mismas carpetas del ejemplo anterior se ejecutó el comando para generar las imágenes usando ajuste de contraste percentile range. En la figura

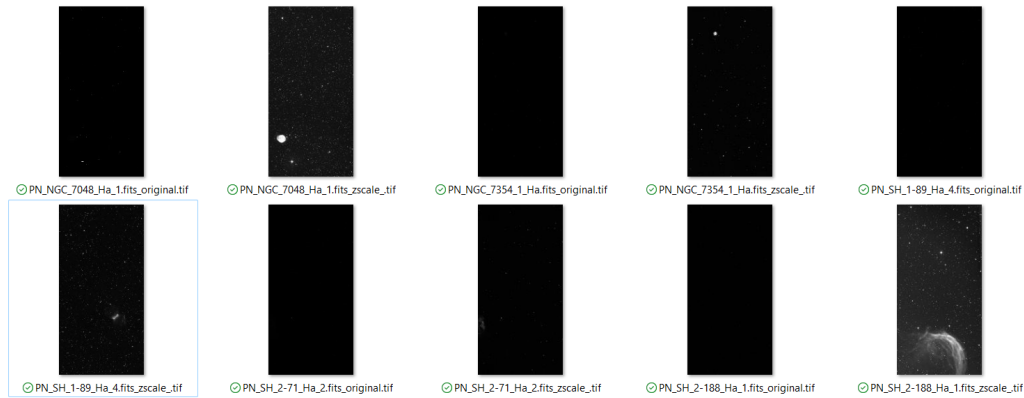


Figura 4: Listado de archivos en la carpeta destinada para almacenar los resultados del algoritmo ZScale.

5 se muestra el ejemplo de ejecución del algoritmo. Al igual que en el ejemplo anterior, se accedió a la carpeta donde está el código fuente, y se ejecuta `principal.py`, le proporcionamos el directorio de entrada y salida, en este caso, se pusieron las carpetas `imagenes_fits` y `resultados`, y debido a que queremos ejecutar percentile range, requerimos el argumento `-p` seguido de `-pr`; por lo tanto, la instrucción quedaría de la siguiente manera:

```
python principal.py -d imagenes_fits -r resultados -p -pr
```

```
C:\Users\eduar\OneDrive\Documents\tesis\AstroIMP>python principal.py -d imagenes_fits -r resultados -p -pr
Preprocess...
PN_NGC_7048_Ha_1.fits
Contrast enhancement with percentile range...
PN_NGC_7354_1_Ha.fits
Contrast enhancement with percentile range...
PN_SH_1-89_Ha_4.fits
Contrast enhancement with percentile range...
PN_SH_2-188_Ha_1.fits
Contrast enhancement with percentile range...
PN_SH_2-71_Ha_2.fits
Contrast enhancement with percentile range...
```

Figura 5: Ejecución de la herramienta en el símbolo de sistema de windows, en el ejemplo se ejecuta la instrucción `-pr` para hacer uso de percentile range.

Después de ejecutar dicho comando, los resultados se guardan en la carpeta `resultados`. En la figura 6, se muestran los archivos del directorio `resultados`, la herramienta generará una imagen `tif` y otra con el resultado obtenido después de aplicar el ajuste de contraste con percentile range.

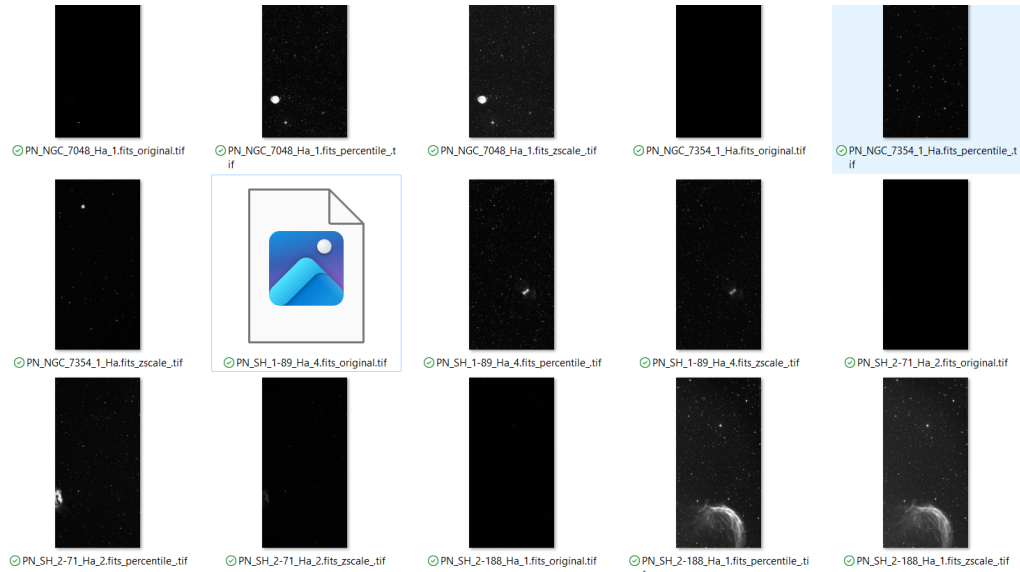


Figura 6: Listado de archivos en la carpeta destinada para almacenar los resultados de percentile range.

5.1.3. Ejemplo de uso del algoritmo arcsinh percentile

En la mismas carpetas del ejemplo anterior se ejecutó el comando para generar las imágenes usando ajuste de contraste arcsinh percentile. En la figura 7 se muestra el ejemplo de ejecución del algoritmo. Al igual que en el ejemplo anterior, se accedió a la carpeta donde está el código fuente, y se ejecuta principal.py, le proporcionamos el directorio de entrada y salida, en este caso, se pusieron las carpetas imagenes_tiff y resultados, y debido a que queremos ejecutar arcsinh percentile, requerimos el argumento -p seguido de -ap; por lo tanto, la instrucción quedaría de la siguiente manera:

```
python principal.py -d imagenes_tiff -r resultados -p -ap
```

Después de ejecutar dicho comando, los resultados se guardan en la carpeta resultados. En la figura 8, se muestran los archivos del directorio resultados, la herramienta generará una imagen tif y otra con el resultado obtenido después de aplicar el ajuste de contraste con arcsinh percentile.

```

C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_tiff -r resultados -p
-ap
Preprocess...
PN_NGC_7048_Ha_1.fits_original.tif
Contrast enhancement with arcsin percentile...
PN_NGC_7354_1_Ha.fits_original.tif
Contrast enhancement with arcsin percentile...
PN_SH_1-89_Ha_4.fits_original.tif
Contrast enhancement with arcsin percentile...
PN_SH_2-188_Ha_1.fits_original.tif
Contrast enhancement with arcsin percentile...
PN_SH_2-71_Ha_2.fits_original.tif
Contrast enhancement with arcsin percentile...

```

Figura 7: Ejecución de la herramienta en el símbolo de sistema de windows, en el ejemplo se ejecuta la instrucción -ap para hacer uso de arcsinh percentile.

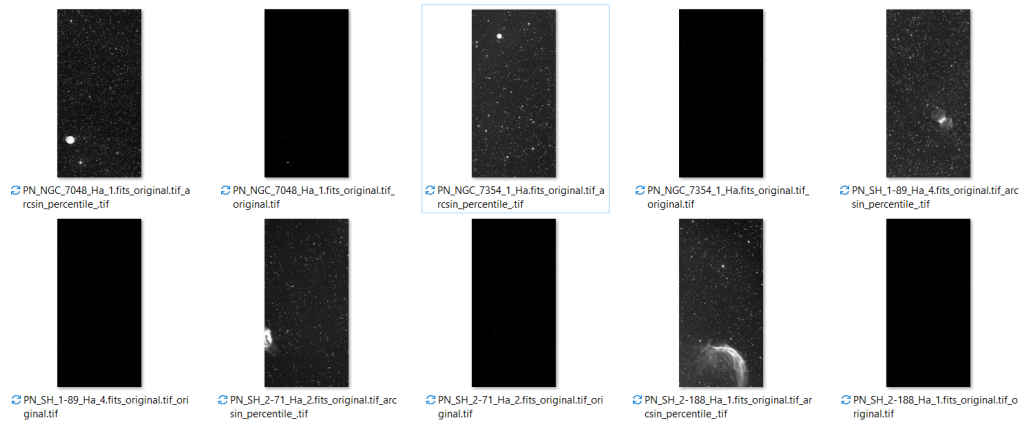


Figura 8: Listado de archivos en la carpeta destinada para almacenar los resultados del algoritmo arcsinh percentile.

5.1.4. Ejemplo de uso del algoritmo arcsinh percentile range

En la mismas carpetas del ejemplo anterior se ejecutó el comando para generar las imágenes usando ajuste de contraste arcsin percentile range. En la figura 9 se muestra el ejemplo de ejecución del algoritmo. Al igual que en el ejemplo anterior, se accedió a la carpeta donde está el código fuente, y se ejecuta principal.py, le proporcionamos el directorio de entrada y salida, en este caso, se pusieron las carpetas imagenes.tiff y resultados, y debido a que queremos ejecutar arcsinh percentile range, requerimos el argumento -p seguido de -apr; por lo tanto, la instrucción quedaría de la siguiente manera:

```
python principal.py -d imagenes.fits -r resultados -p -apr
```

Después de ejecutar dicho comando, los resultados se guardan en la carpe-

```

C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_tiff -r resultados -p
-apr
Preprocess...
PN_NGC_7048_Ha_1.fits_original.tif
Contrast enhancement with arcsin percentile range...
PN_NGC_7354_1_Ha.fits_original.tif
Contrast enhancement with arcsin percentile range...
PN_SH_1-89_Ha_4.fits_original.tif
Contrast enhancement with arcsin percentile range...
PN_SH_2-188_Ha_1.fits_original.tif
Contrast enhancement with arcsin percentile range...
PN_SH_2-71_Ha_2.fits_original.tif
Contrast enhancement with arcsin percentile range...

```

Figura 9: Ejecución de la herramienta en el símbolo de sistema de windows, en el ejemplo se ejecuta la instrucción -apr para hacer uso de arcsinh percentile range.

ta resultados. En la figura 10, se muestran los archivos del directorio resultados, la herramienta generará una imagen tif y otra con el resultado obtenido después de aplicar el ajuste de contraste con arcsinh percentile range.

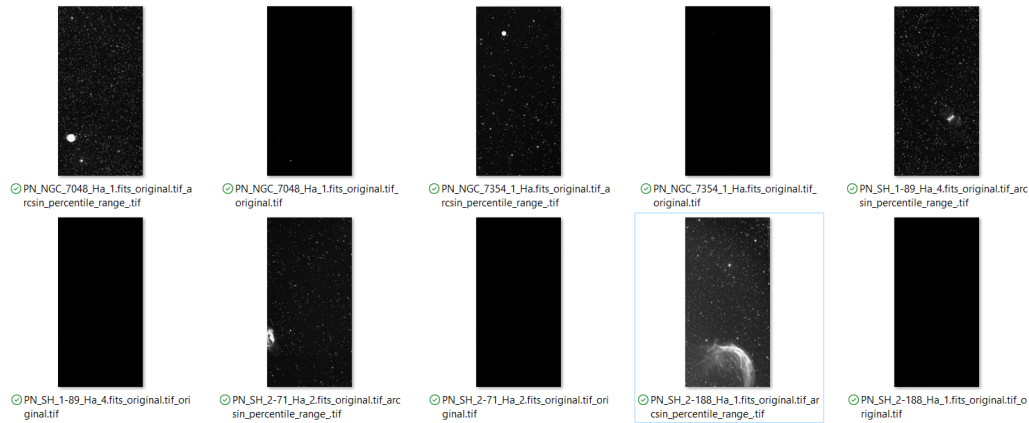


Figura 10: Listado de archivos en la carpeta destinada para almacenar los resultados del algoritmo arcsinh percentile range.

5.2. Recortar imágenes

En este ejemplo, se tomó una imagen de tamaño 2049 x 4096 pixeles, y se crearon dos carpetas imagenes_tiff y resultados_recortar.

En la figura 11 se muestra el ejemplo de ejecución de la funcionalidad recortar imágenes. Se accedió a la carpeta donde está el código fuente, y

```
C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_tiff -r resultados_recortar -c
Crop images...
```

Figura 11: Ejecución de la herramienta en el símbolo de sistema de windows, en el ejemplo se ejecuta el comando -c para recortar las imágenes.

se ejecutó la siguiente instrucción: `python principal.py`, le proporcionamos el directorio de entrada con `-d` y salida con `-r`, en este caso, se especificaron las carpetas `imagenes_tiff` y `resultados_recortar` respectivamente, y debido a que queremos ejecutar la funcionalidad para recortar la imagen, requerimos el argumento `-c`; por lo tanto, la instrucción quedaría de la siguiente manera:

`python principal.py -d imagenes_tiff -r resultados_recortar -c`

Después de ejecutar dicho comando, los resultados se guardan en la carpeta `resultados`. En la figura 12, se muestran los archivos del directorio `resultados`, la herramienta generará varias imagenes tif con tamaño de 512 x 512 pixeles.

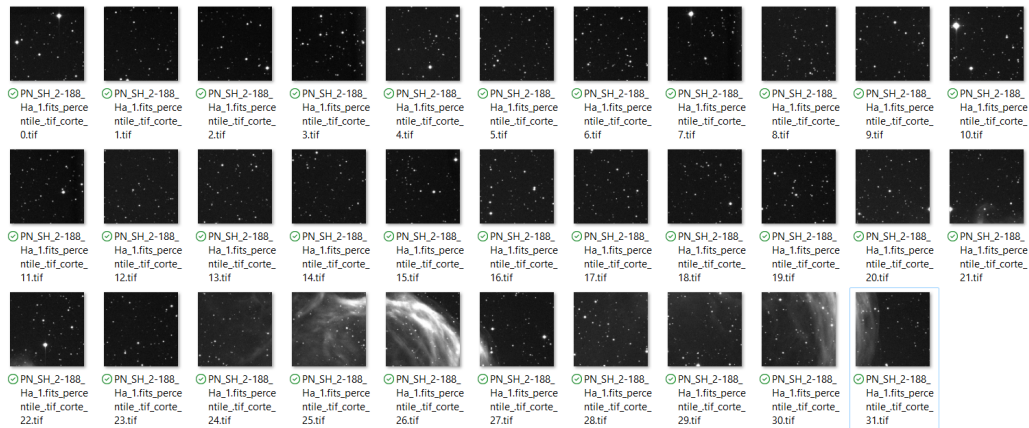


Figura 12: Listado de archivos en la carpeta destinada para almacenar los resultados del recorte realizado a la imagen, en la que genera varias imágenes de tamaño 512 x 512 pixeles.

5.3. Eliminación de fondo con PFCM

Para este ejemplo, es requerido que primero se tenga la imagen con ajuste de contraste, en este caso, se ejecutó primero el algoritmo de percentile range,

puesto que es el que proporciona mejor resultado, se creó otra carpeta que se llama `imagenes_tiff`, para almacenar las imágenes a eliminar el fondo.

```
C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_tiff -r resultados -p
-pf
Preprocess...
PN_NGC_7048_Ha_1.fits_percentile_.tif
PN_NGC_7354_1_Ha.fits_percentile_.tif
PN_SH_1-89_Ha_4.fits_percentile_.tif
PN_SH_2-188_Ha_1.fits_percentile_.tif
PN_SH_2-71_Ha_2.fits_percentile_.tif
```

Figura 13: Ejecución de la herramienta en el simbolo de sistema de windows, en el ejemplo se ejecuta el algoritmo PFCM y eliminación del fondo.

En la figura 13 se muestra el ejemplo de ejecución del algoritmo. Se accedió a la carpeta donde está el código fuente, y se ejecutó la siguiente instrucción: `python principal.py`, le proporcionamos el directorio de entrada con `-d` y salida con `-r`, en este caso, se especificaron las carpetas `imagenes_tiff` y `resultados` respectivamente, y debido a que queremos ejecutar el algoritmo PFCM y elimine el fondo de la imagen, requerimos el argumento `-p` seguido de `-pf`; por lo tanto, la instrucción quedaría de la siguiente manera:

```
python principal.py -d imagenes_tiff -r resultados -p -pf
```

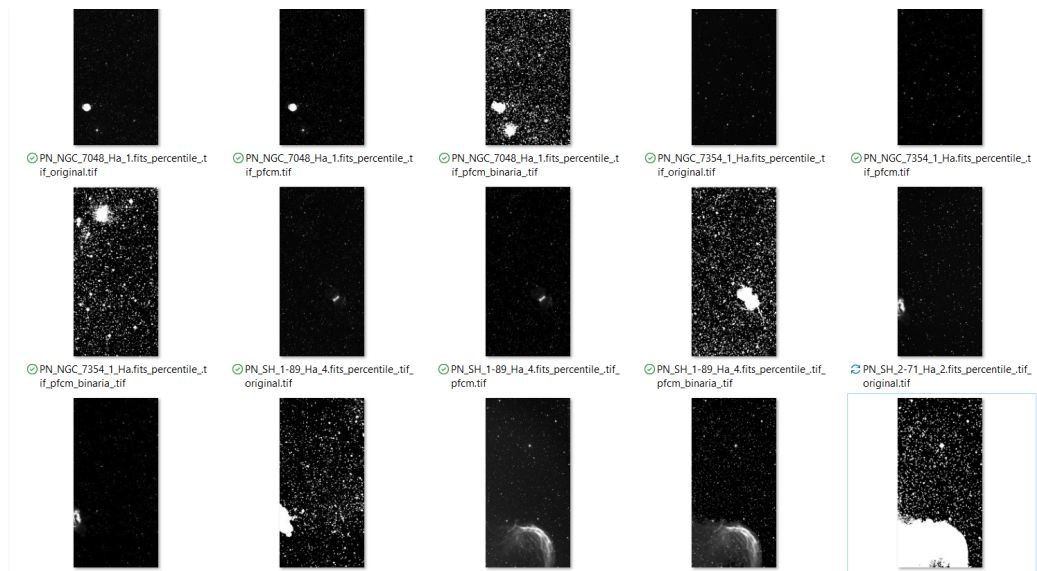


Figura 14: Listado de archivos en la carpeta destinada para almacenar los resultados del algoritmo PFCM y eliminación del fondo.

Después de ejecutar dicho comando, los resultados se guardan en la carpeta resultados. En la figura 14, se muestran los archivos del directorio resultados, la herramienta generará una imagen tif, una imagen con el resultado obtenido después de aplicar el algoritmo PFCM, en dicho resultado solo se mostrará una imagen binaria en donde solo se mostrará el fondo en negro y los objetos en blanco, y dicha imagen sirve como máscara, para generar otra imagen que elimina el fondo y se queda con los objetos con el rango dinámico original.

5.4. Reducción de errores de CCD

Para este ejemplo, se agregaron las imágenes originales en formato fits a la carpeta imagenes_fits, en este caso, se ejecutó primero la instrucción -ccde seguido de -pr para que al finalizar la reducción de errores se generara una nueva imagen con ajuste de contraste y pueda verse el resultado. Se creó otra carpeta que se llama resultados_errores_ccd, para almacenar el resultado de la ejecución de la reducción de errores de CCD.

```
C:\Users\eduar\OneDrive\Documentos\tesis\AstroIMP>python principal.py -d imagenes_tiff_errores_ccd -r resultados -p -ccde -pr
Preprocess...
HII_LBN_188.69+04.25_Ha_3.fits
HII_LBN_188.69+04.25_Ha_3.fits
HII_NGC_7000_2_1_Ha.fits
HII_RFS_546_Ha.fits
HII_SH_2-241_Ha_3.fits
HII_SH_2_131_Ha_2.fits
Contrast enhancement with percentile range...
HII_NGC_7000_2_1_Ha.fits
HII_LBN_188.69+04.25_Ha_3.fits
HII_NGC_7000_2_1_Ha.fits
HII_RFS_546_Ha.fits
HII_SH_2-241_Ha_3.fits
HII_SH_2_131_Ha_2.fits
Contrast enhancement with percentile range...
HII_RFS_546_Ha.fits
HII_LBN_188.69+04.25_Ha_3.fits
HII_NGC_7000_2_1_Ha.fits
HII_RFS_546_Ha.fits
HII_SH_2-241_Ha_3.fits
HII_SH_2_131_Ha_2.fits
Contrast enhancement with percentile range...
HII_SH_2-241_Ha_3.fits
HII_LBN_188.69+04.25_Ha_3.fits
HII_NGC_7000_2_1_Ha.fits
HII_RFS_546_Ha.fits
HII_SH_2-241_Ha_3.fits
HII_SH_2_131_Ha_2.fits
Contrast enhancement with percentile range...
```

Figura 15: Ejecución de la herramienta en el símbolo de sistema de windows, en el ejemplo se ejecuta la instrucción -ccde para reducir errores de CCD.

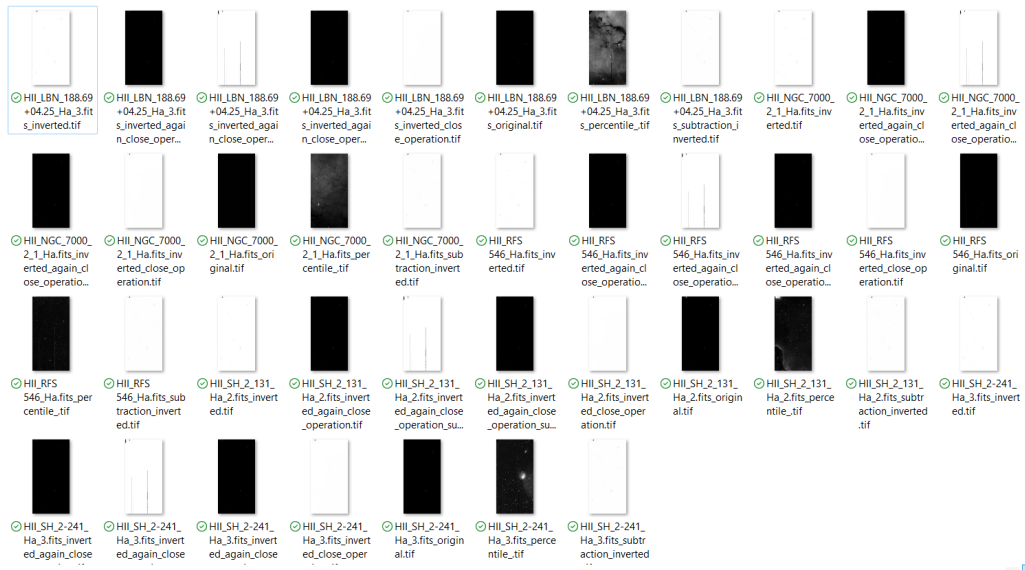


Figura 16: Listado de archivos en la carpeta destinada para almacenar los resultados de la aplicación de operadores morfológicos para reducir errores de CCD.

En la figura 15 se muestra el ejemplo de ejecución de la instrucción `-ccde` para reducir errores CDD. Se accedió a la carpeta donde está el código fuente, y se ejecutó la siguiente instrucción: `python principal.py`, le proporcionamos el directorio de entrada con `-d` y salida con `-r`, en este caso, se especificaron las carpetas `imagenes.fits` y `resultados` respectivamente, y debido a que queremos ejecutar el operador morfológico `close` para reducir los errores CCD, requerimos el argumento `-ccde` seguido de `-pr`; por lo tanto, la instrucción quedaría de la siguiente manera:

```
python principal.py -d imagenes.fits -r resultados_errores_ccd -ccde -pr
```

Después de ejecutar dicho comando, los resultados se guardan en la carpeta `resultados`. En la figura 16, se muestran los archivos del directorio `resultados`, la herramienta generará varias imágenes `tif`: La primera imagen es la original, la segunda imagen es el resultado obtenido después de invertir la imagen original, la tercera imagen es la que se resta la imagen original con la invertida, la cuarta imagen es después de aplicar el operador morfológico `close`, la quinta imagen es después de invertir el resultado anterior, la sexta imagen es después de restar el resultado con la imagen original y la última imagen es el resultado de devolver al rango dinámico original.

6. Conclusión

El procesamiento de imágenes en astronomía es complejo y requiere la realización previa de varios pasos para garantizar que el ruido, el fondo de la imagen y las estrellas no interfieran con la detección de objetos extendidos. No se puede aplicar un método tradicional de segmentación de imágenes por sí solo, ya que la calidad de las imágenes no es la misma que la de una imagen tradicional.

Dada la naturaleza difusa de las nebulosas de emisión, y el ruido producido por el *CCD*, lo mejor era aplicar el filtro Perona Malik y el algoritmo posibilista difuso (*PFCM*) para obtener un umbral y poder eliminar el fondo de la imagen, dejando solo estrellas y objetos extendidos.

Esta herramienta permite eliminar el fondo de la imagen, dejando solo los objetos de interés, dicho proceso es fundamental para poder aplicar técnicas de segmentación de imágenes, y que la imagen no tenga ruido que pueda afectar la detección automatizada de objetos extendidos.