Observing the HAT-P-18 b Transit and Calculating Planetary mass through Light Graphs

Sarthak Sahai
Student No. 218743401

Transit Observed in collaboration with:
Lynn Al Agilly
Kaia Turchio
Hishaam Hassan

Professor: Sarah Rugheimer
Associate Professor, Research Stream, York University

York university
Lassonde School of Engineering
PHYS 3070: Planets and Planetary Systems

Sahai, November 2024

Abstract

The Planetary Transit Observing project aims to collect and analyze data from exoplanetary transits, with the goal of determining the radius of an exoplanet. The project utilizes a series of Bias, flats and transit observation images obtained through the Allan I. Carswell Observatory. These observations help improve the accuracy of future models in planetary sciences and astrophysics. Several images were captured using the CCD camera used by the Observatory within the .FITS format which allows for scientific analysis since the data ranges from a value of 0 to 65535 while storing data. This vast range allows us to visualize/record minor variations of photons received which may not be visible to the naked eye. By looking at these tiny variations, we can learn more about the exoplanets' size and composition as well. For this project, the exoplanet that was observed is known as HAT-P-18 b which revolves around the star HAT-P-18.

Keywords: HAT-P-18-B, Bias, Flat

Contents

List of Figures

List of Tables

Acronyms

AICO Allan I. Carswell Observatory. (*p. v, 2, 3*)

CCD Charge-Coupled Device. (*p. 2*)

NaN Not a Number. (*p. 7*)

RAM Random Access Memory. (*p. 7, 13*)

1

Introduction

Exoplanets, planets that exist outside of our solar system, vary widely in size and composition. They may range from small Earth-sized planets to massive Jupter-sized Gas giants. The methods for discovering these exoplanets depend on several factors such as the size of the planet, its distance from Earth, and the presence of stellar dust within our line of sight which can interfere with observations. Detecting smaller exoplanets at large distances is extremely difficult due to the limitations of our current scientific instruments. On the other hand, larger exoplanets that are relatively closer to Earth are much easier to determine and examine. However, advancements in technology and development of much more sophisticated (and often more expensive) equipment has improved our ability to discover and learn about these exoplanets. One such exoplanet is the HAT-P-18 b which is an exoplanet that lies approximately 530 Light years away.

For the completion of the Observing project, We have chosen the exoplanet HAT-P-18 b and have observed using the Allan I. Carswell Observatory (AICO) located at York University Keele Campus. This Observatory uses a Charge-Coupled Device (CCD) Camera that allows us to take high quality images of objects outside of the Earth's atmosphere in great detail. For successful completion of the project, the team had obtained 50 bias images of the CCD Camera both before and after the observing to account for pixel bias. Additionally, the team had also obtained 50 flat images as well as 706 Transit images in collaboration with the AICO team on the night of 2nd October 2024.

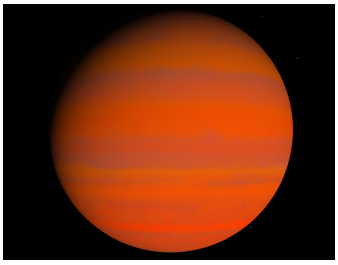


Figure 1.1: *Hypothetical Image of HAT-P-18 b as per the NASA Exoplanet Catalog* [D, 2024]

2

Star Properties

2.1 Choosing a star

Choosing a star is an important part of observing transits. we need to account for several factors such as the limitations of the observatory, environmental factors and duration of the transit. To choose a star that would provide us with the best results, our team started by finding a day where the weather is ideal for observations. Factors such as Cloud cover, Transparency and Darkness have a significant impact on the quality of our transit. To find the best possible day to observe our transit we have used the resources provided by: https://www.cleardarksky.com/c/Torontokey.html.

Once a day is found with the ideal environmental parameters, we need to account for the limitations of the observatory itself. Allan I. Carswell Observatory (AICO) has two telescopes out of which the 1-meter telescope was used for our observations. To observe exoplanet transits using this telescope we need to make sure that the exoplanet itself is above 20 degrees of altitude throughout the transit.

It is important to note that the location of the observatory impacts the visibility of exoplanet transits as well. The exact location of AICO is as follows;

Feature	Value
Longitude	79° 30′ 7.83″ W
	-79.5006061 W
	280.499394 E
Latitude	43° 46′ 13.81″ N
Altitude	209 Meters

Table 2.1: *Geographical Properties of AICO*

These values could be inserted at https://var.astro.cz/en to get a list of the exoplanets you could observe at a certain time. The transiting process may be improved significantly by choosing an exoplanet where the;

- Duration of transit is shorter
- Depth is higher
- V (MAG) is lower

By referring to the aforementioned websites and recommendations outlined by the project description, we have chosen the exoplanet HAT-P-18 b and observed it's transit on 2nd October 2024.

2.2 Properties of HAT-P-18

As previously mentioned, for this project we have chosen to observe the star HAT-P-18 b. This planet orbits a star named HAT-P-18 that lies approximately 530 Light years away [D, 2024]. This is a K2-type Star as per the Harvard Spectral Classification for Main sequence stars. The star has the radius of approximately 0.75 solar radii and

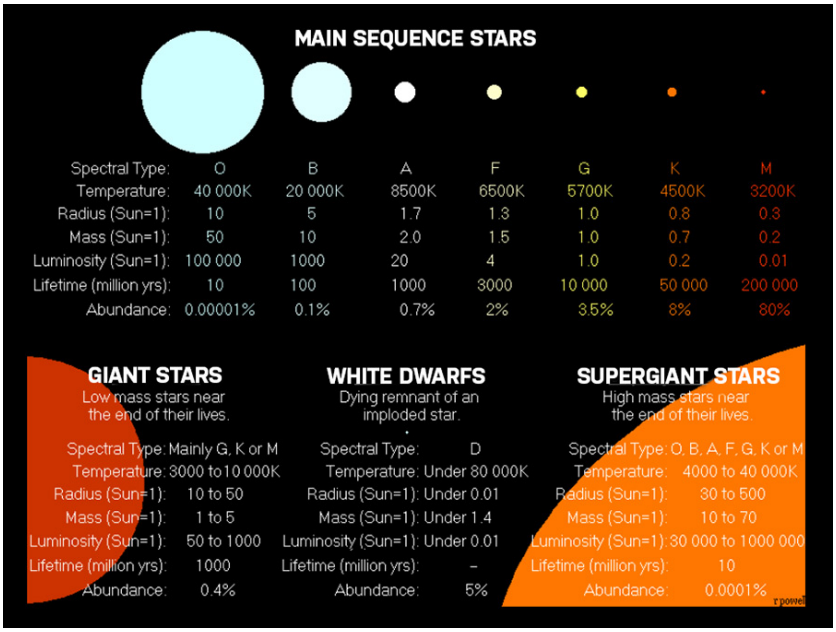


Figure 2.1: *Harvard Star Classification for Main Sequence Stars* [*Jones, n.d.*]

holds a mass of 0.77 solar masses. Although the star has an Apparent Magnitude of only 12.76 making it impossible to see the star without the use of a telescope. For the project, a telescope with the primary mirror of 1 meter was used.

Additionally, this star was found in the year 2010 and has only 1 exoplanet orbiting it. More properties of the star may be found in Table: 2.2

Name	Value	Units	Description
Name	HAT-P-18	N/A	Name of the star
Spectral Type	K2	N/A	Classification of the star based on its spectrum
Apparent Magnitude	12.76	N/A	Brightness of the star as seen from Earth
Distance	530	Light-Years	Distance from Earth to the star
Mass	0.77	Solar Masses	Mass of the star relative to the Sun
Radius	0.75	Solar Radii	Radius of the star relative to the sun

Table 2.2: *HAT-P-18 Star Properties* [*Tokyo, n.d.*]

3

Observing the Transit

When observing a transit it is crucial to take a series of pictures of the star before and after the transit to be able to form a smooth transit curve where we can measure the transit depth much more effectively. It is also essential to take bias and flat images to account for the camera error and remove any unwanted outliers within our data.

For our project we have taken 50 bias images before observing the transit. taking multiple bbias images allows us to calculate the mean bias at each pixel allowing us to have a much more accurate master bias. Since bias values may change during the operation of a camera, we have also taken 50 bias images after observing the transit. These 100 bias images were then stacked to calculate the master bias that is used for Image Data Reduction under Section 4.2.

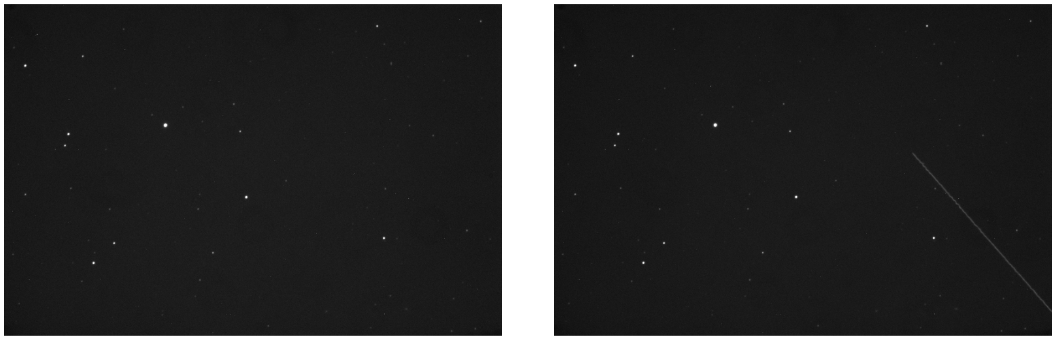
The camera bias values significantly change under light. to account for these we also take a series of images called flats. Similar to the bias images, we have taken 50 flat images to account for the bias under light as well. These images were also stacked to calculate the master Flat as outlined under Section 4.3.

3.1 Star Transit Images and Outliers

About 706 images were taking during the transit of HAT-P-18 b. out of these images, 6 of them were excluded from the data set due to outliers being present within the image. Presence of outliers lead to unwanted photons being captured by our camera that may distort our Scientific Data. By removing these images, we ensure no sudden spikes or dips within our data is present leading to a much smoother transit curve. A visual representation of an acceptable transit image compared to a non-acceptable image is outlined by Figure 3.1

3.2 Process of generating a transit curve

Once we observe a transit we have an abundance of Data. Unfortunately, most of the data we do record, is corrupted by factors such as the environment, bias, outliers



(**a**) *Image without outliers* (**b**) *Image with outliers*

Figure 3.1: *Differences between Image with and without Outliers*

present within the science images and others. To prepare this data to useful scientific information we would be performing the following steps:

- Calculate the Master Bias
- Calculate the Master Flat
- Remove the Master Flat and Bias from the science images
- Cosmic Ray and air glow removal
- Star alignment

Once we have have cleaned the images, we are now able to perform scientific analysis. For our project, we would be using this data to form a transit curve and calculate several parameters such as the transit depth, planet radius, Ingress and Egress duration etc. Note that observing the same transit multiple times also allows us to understand the composition of the planet itself.

once the data is cleaned we are now able to perform analysis without worrying about outliers within our data. To begin with the generation of our transit curve, we need to achieve a couple tasks first. Namely;

- Finding the Star in our science images
- Aperature testing for accurate photon level detection
- Annulus subtraction and differential photometry for lower errors

Once we have finished with cleaning and reducing the amount of distortions within our data, we could finally generate a transit curve.

$$transit\ depth(\delta) = (\frac{R_{planet}}{R_{stellar}})^2$$

The formula for transit depth stated above was used to calculate the radius of the planet in section 6.0 of this report. Note that the transit curve depth itself was calculated by normalizing the transit curve and visually inspecting the difference between the amount of photons (normalized) received before and during the transit.

4

Image Data Reduction

When an image is taken, it initially contains unwanted data that may skew our results. This may be removed from a series of procedures that are described in later portions of this section. Data processing for this project was done using the programming language python and in the Google Colab Work Environment. Please note that this data uses large amounts of Random Access Memory (RAM) and in order to replicate the results, either a RAM size of 70GBs or access to Google Colabs TPU server is recommended.

4.1 Loading the Data

To start processing the data we have to first load the data that we already have. Additionally, we also need to import libraries that may aid us to process our data. In our case, the data to be processed was saved in a Google Drive folder and retrieved. Please note that although the python library Numpy offers similar functions, it also assumes Not a Number (NaN) values to be infinite which leads to data corruption. To prevent this, we have used the python library CCDPROC instead. The Code that was used as well as their outputs are as follows:

```
        # Here's a cool shark :)


#               ,
#             .';
#          .-'` .'
#        ,`.-'-.`\
#       ; /     '-'
#       | \       ,-,
#       \  '-.__   )_`'._
#        '.     ```      ``'--._
#       .-' ,                   `'-.
#        '-'`-._           ((   o   )
#               `'--....(`- ,__..--'
#                        '-'`
```

Listing 4.1: *A cool shark*

```
#installing libraries
pip install astropy ccdproc
pip install photutils
pip install astroalign
```

Listing 4.2: *Installing the Libraries*

```
        # This code block is dedicated to import libraries to perform calculations
        ↪ throughout this document
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimgx
import ccdproc as ccdp
import astropy as ap
from astropy.io import fits
import time
import os
from photutils.background import Background2D, SExtractorBackground
from photutils.aperture import CircularAperture, aperture_photometry
from IPython.display import clear_output
import astroalign as aa

# Importing and downloading documents from Google Drive
from google.colab import drive, files
drive.mount('/content/drive')
gDrivePath = '/content/drive/My Drive/2024-25/PHYS 3070/Observing Project/Images/'
```

Listing 4.3: *Importing more libraries and Connecting to the Google Drive Folder*

The code in Listing 4.3 Provided us with the following output:

```
Mounted at /content/drive
```

Now when the data is loaded we now scan the data for outliers as mentioned in Section 2.2. For this we have gone through every picture individually and found out that the images on the following indices contain outliers; 108, 160, 201, 233, 234, 242, 375, 691. Thus, the following code was used to remove the aforementioned indices within our transit image data set

```
        #removing the outliers in the transit data
indexToBeRemoved = [108,160,201,233,234,242,375,691] # already subtracted 1 for
↪ indexing
for i in indexToBeRemoved[::-1]: # Sort indexToBeRemoved in descending order
  transitImages.pop(i)  # Remove images with indexToBeRemoved
  print('Removed image ' + str(i+1) + ' of ' + str(len(transitImages)) + ' from Transit
  ↪ Images')
  clear_output(wait=True)
```

Listing 4.4: *Removing the outliers from the dataset*

The code in 4.4 Provided us with the following output:

```
Removed image 109 of 698 from Transit Images
```

The output implies that after correcting the data set i.e. removing the outliers, we remain with 698 transit images. Now that we have our data set ready to be used, we can now calculate our Bias' , Flats and other major factors of the project much more accurately.

4.2 Data Analysis Part 1A - Master Bias generation

As of this stage in the code, we have stored all 100 (50 before and 50 after) of our bias images in a variable called bias Images. We then stacked all of these images and calculated the median pixel value across all pixels to obtain our Master Bias. We then corrected our Flat and transit data by subtracting our master Bias from them. This ensures that our scientific data isn't affected by the Camera bias. This allows us to examine the scientific data only and prevents further outliers caused by the camera.

This is represented by the code in 4.5

```
1         # calculating the master bias using the median method
2 totalBias = ccdp.combine(biasImages, method='median')
3 print('totalBias Image calculated')
4
5 # displaying the total bias image
6 plt.imshow(totalBias,cmap='gray')
7 plt.title('Total Bias Image')
8 plt.xlabel('X-axis (pixels)')
9 plt.ylabel('Y-axis (pixels)')
10
11 # subtracting the master bias from the flats and the transit images
12 flatsCorrected = [ccdp.subtract_bias(flatsImage, totalBias) for flatsImage in
   ↪  flatsImages]
13 transitCorrected = [ccdp.subtract_bias(transitImage, totalBias) for transitImage in
   ↪  transitImages]
14 print('Flats and Transit Images corrected')
```

Listing 4.5: *Calculating the Master Bias and data correction*

By running the code provided in listing 4.5 we have received the following output:

```
totalBias Image calculated Flats and Transit Images corrected
```

4.3 Data Analysis Part 1B - Master Flat Generation

Since we have corrected our flat and transit images by subtracting the bias images, we have now accounted for the irregularities between the pixels by subtracting the

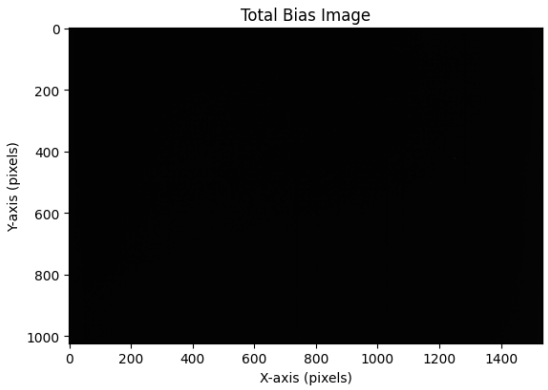


Figure 4.1: *Master Bias*

median pixel brightness value from them. This allows us to calculate our Master Flat and dividing that with our transit image data gives us the true image of the sky without any irregularities that may be caused by the camera.

To do this we have used the following code:

```
        #stacking the flats
stackedFlats = ccdp.combine(flatsCorrected, method='median')
print('Stacked Flats Image calculated')
clear_output(wait=True)

#dividing the mean of the flats
totalFlats = stackedFlats / np.mean(flatsCorrected)

# displaying the calculated master flat image
plt.imshow(totalFlats,cmap='gray')
plt.title('Total Flats Image')
plt.xlabel('X-axis (pixels)')
plt.ylabel('Y-axis (pixels)');
```

Listing 4.6: *Calculating of the Master Flat*

The code mentioned in listing 4.6 has provided us with the following output:

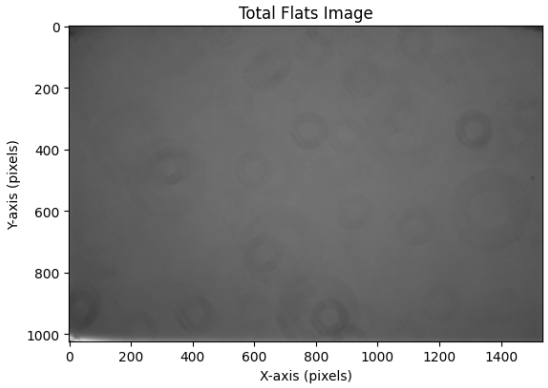


Figure 4.2: *Master Flat*

After calculating the master flat 4.2 we then divide this by our transit image data using the following code:

```
        # dividing each transit image by the master flat
transitCorrectedflatDivided = [transitCorrectedImage / totalFlats for
↪  transitCorrectedImage in transitCorrected]
print('Transit Images corrected by dividing flats')
```

Listing 4.7: *Correcting transit data by dividing flats*

The code provided in listing 4.7 provided us with the following output:

```
Transit Images corrected by dividing flats
```

4.4 Data Analysis Part 1C and 1D - Cosmic Ray and air glow removal

Fortunately our data set wasn't corrupted by Cosmic Rays, thus we didn't have to perform cosmic ray removal. Although if necessary, it could've been done using the python library AstroScrappy. Despite the absence of cosmic rays, our data was slightly corrupted by the air glow of the atmosphere. The air glow could be calculated and removed using pre-existing python libraries such as photutils. This is implemented in the code below:

```
        # Note: We did not have to perform cosmic ray removal since our data was didn't
        ↪  have any cosmic rays

# defining the background estimator method
bkgEstimate = SExtractorBackground()

# defining an empty variable to account for the code running multiple times
transitImagesNoBackground = []
for i in range(len(transitCorrectedflatDivided)):
  bkg = Background2D(transitCorrectedflatDivided[i], (50,50),filter_size=(3,3),
  ↪  bkg_estimator=bkgEstimate)
  transitImagesNoBackground.append(transitCorrectedflatDivided[i] - bkg.background)
  print('Corrected Image: ' + str(i+1) + ' of ' +
  ↪  str(len(transitCorrectedflatDivided)))
  clear_output(wait=True)
```

Listing 4.8: *Removal of air glow from the tranist data*

4.5 Data Analysis Part 1E - Star Alignment

At this stage of the data correction process, we are nearly ready to begin data analysis. After completing Bias Correction, Flat Correction, and Cosmic Ray/Air Glow removal,

the final step in preparing the data is to align the stars so that all frames have the stars positioned in approximately the same location. This significantly improves the quality of the transit data, as it eliminates the need for active tracking and adjustments in each image.

To perform this, we have used the following code:

```
        # aligning all the pictures
alignedTransitImages = []
referenceTransit = transitImagesNoBackground[0]
alignedTransitImages.append(referenceTransit)
for image in transitImagesNoBackground[1:]: # starting from the second image since the
↪  first image is the reference image
  alignedTransitImage , footprint = aa.register(image, referenceTransit)
  alignedTransitImages.append(alignedTransitImage)
  print('Aligned Image: ' + str(len(alignedTransitImages)) + ' of ' +
  ↪  str(len(transitImagesNoBackground)))
  clear_output(wait=True)
```

Listing 4.9: *Aligning the transit Images*

At this point we have also displayed a sample aligned transit image to verify that our order of operations aren't having any detrimental affects to our data.

```
        plt.imshow(alignedTransitImages[280],cmap='gray')
plt.title('Sample Aligned Image')
plt.xlabel('X-axis (pixels)')
plt.ylabel('Y-axis (pixels)');
```

Listing 4.10: *Displaying a randomly chosen image*

Running the code mentioned in listing 4.10 provides us with the following output:

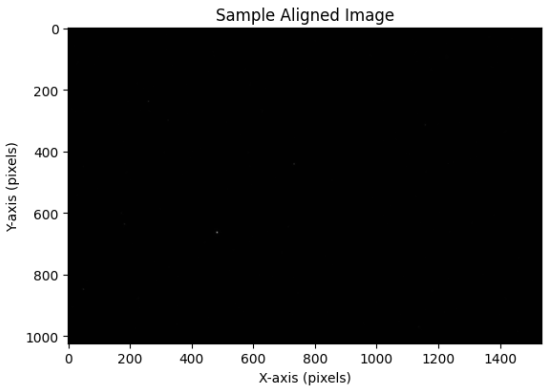


Figure 4.3: *Sample Aligned Image*

Note: The sample Aligned image may be darker than the images provided in Section 2.2 since the Sample Aligned Image was exported in PNG rather than .FITS

This marks the end of Image Data Reduction and our transit data is now ready for analysis. To summarize, we have;

1. Subtracted Bias Frame from Transit and Flat Images
2. Divided Transit Images by Flat Frames
3. performed dynamic background subtraction (Air Glow removal)
4. Aligned the stars within our transit data set

At this point it is advisable to store the transit image data somewhere to prevent running all the aforementioned code multiple times while working on the code. Please note that this part has no effect on the data, but helps save RAM and other resources in the long run. To perform this, the following code was used:

```
        # Define the path to save the images
gDrivePath = '/content/drive/My Drive/2024-25/PHYS 3070/Observing
↪  Project/AlignedTransitImages/'

# Save each aligned image as a FITS file
for i, image in enumerate(alignedTransitImages):
  # Create the file name
  fileName = 'Aligned Image ' + str(i + 1) + '.fit'

  # Save the image data to the file
  hdu = fits.PrimaryHDU(image.data)  # Create a FITS HDU
  hdu.writeto(gDrivePath + fileName, overwrite=True)  # Write to file

  print('Saved image ' + str(i + 1) + ' of ' + str(len(alignedTransitImages)))
  clear_output(wait=True)

print('All aligned images saved to Google Drive.')
```

Listing 4.11: *saving data to google drive*

Once the data was saved, the code block was removed to prevent overwriting of data or possible data corruption.

5

Flux Data Reduction

To perform Flux Data Reduction we have no re-imported the data back from google drive to save computational resources. The following code block allows us to effectively import the data from a certain folder within google drive:

```
        # Loading all Aligned transit images from the google drive
alignedTransitImages = []
for i in range(1,699): # number of transit images
  image = fits.open(gDrivePath + 'AlignedTransitImages/' + 'Aligned Image ' + str(i) +
  ↪  '.fit')
  imageData = image[0].data
  alignedTransitImages.append(ccdp.CCDData(imageData, unit='adu'))
  print('Loaded images ' + str(i) + ' of ' + str(698))
  clear_output(wait=True)
```

The data is now imported and ready to use for Flux Data Reduction

5.1 Data Analysis Part 2A - Star Detection

Once the data is loaded, we now need to find the star within our image. To do this we have used a finder chart from the exoplanet database provided by VarAstro. HAT-P-18-b is located at approximately the center of the Figure 5.1

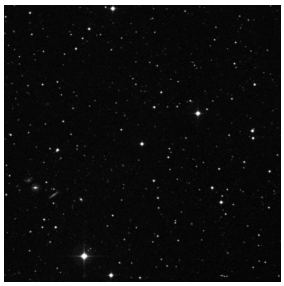


Figure 5.1: *Star Finder Chart for HAT-P-18 b* [*VarAstro, n.d.*]

By comparing the star finder to our calculated transit images we could get the approximate location of the star in pixels which could be plotted for ease of viewing. The following code represents the star by a red dot for ease of viewing.

```
        # The Star was manually identified by comparing the transit images to the
        ↪  images provided by the exoplanet database provided in the course PHYS 3070
# upon examining the Aligned transit images, we notice that the Exoplanet is located at
↪  the coordinates (732,582)

# Locating the star
HatP13BStarLocation = (732,442)
plt.imshow(alignedTransitImages[280],cmap='gray')
plt.plot(HatP13BStarLocation[0],HatP13BStarLocation[1],'ro', label='Hat-P-18-b')
plt.legend()
plt.title('Star Location')
plt.xlabel('X-axis (pixels)')
plt.ylabel('Y-axis (pixels)');
print('Star Location: ' + str(HatP13BStarLocation) + ' pixels')

```

Listing 5.1: *locating the star HAT-P-18*

Upon running the code mentioned in listing 5.1 we receive the following output:

```
Star Location: (732, 442) pixels
```

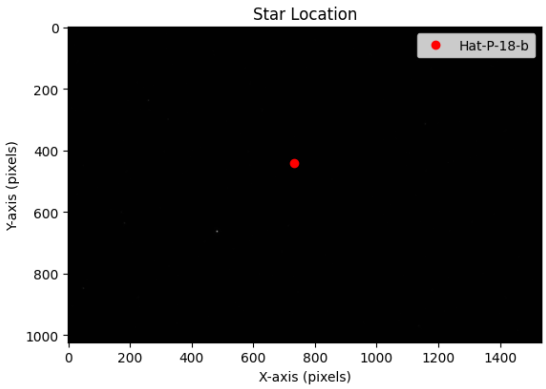


Figure 5.2: *Location of the Star*

5.2 Data Analysis Part 2B - Aperture Test

Once we have located our star we need to draw a circular aperture around the star from radius varying between 1-20 pixels to find the aperture diameter that measures the total flux of the star but minimal to none background distortions. This test was performed using the following code:

```
        # Creating a graph for circular Aperture vs recorded flux from 1 to 20 pixel
        ↪ size

# Defining aperture size
radiusAperture = np.arange(1,21,0.2)

# Defining calculated flux
calculatedFlux = []

# Calculating the flux in respect to the aperture size
for radius in radiusAperture:
  aperture = CircularAperture(HatP13BStarLocation, r=radius) # defining the aperture
  phot_table = aperture_photometry(alignedTransitImages[280], aperture) # calculating
  ↪ the flux value
  calculatedFlux.append(phot_table['aperture_sum'][0].value)

# Plotting the graph
plt.plot(radiusAperture,calculatedFlux,'g',label='Calculated Flux')

# Adding a line at aperture size, 5,6,7
plt.axvline(x=5, color='b', linestyle='--',label='aperture = 5')
plt.axvline(x=6, color='purple', linestyle='--',label='aperture = 6')
plt.axvline(x=7, color='r', linestyle='--',label='aperture = 7')

# Describing the plot
plt.legend()
plt.xlabel('Radius of Aperture (pixels)')
plt.ylabel('Flux (electrons)')
plt.title('Aperture vs Flux for transit Star');

print('Upon interpolating the data from the acquired graph\nwe notice that the aperture
↪ size of about 6 pixels capture the whole transit')
```

Listing 5.2: *Aperture radius test*

Upon running the code mentioned within listing 5.2 we receive the following output:

```
Upon interpolating the data from the acquired graph we notice that the aperture
size of about 6 pixels capture the whole transit
```

Upon analysing Figure 4.3, understand that the aperture size of 6 pixels is ideal for our calculations. This value will be used for subsequent calculations.

5.3 Data Analysis Part 2C and 2D - Annulus Subtraction and Differential photometry

Since we have performed dynamic background subtraction (Air Glow removal) already, we do not need to perform the annulus background subtraction since our data

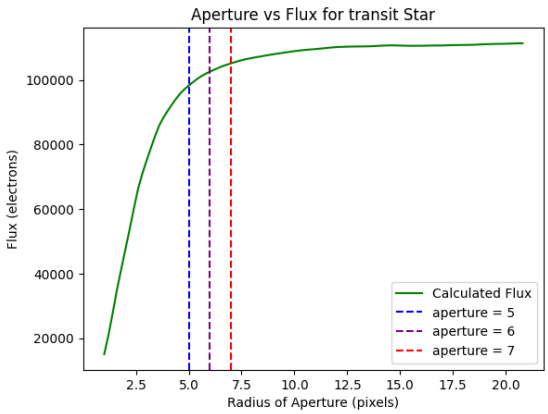


Figure 5.3: *Aperture Vs Flux for Transit Star*

isn't corrupted by air glow or other stars near our transiting star.

To perform differential photometry we need to locate reference stars. The location of the reference stars were calculated similarly to how the location of the Transit star was found. Changing the light parameters of the image may help in finding appropriate reference stars for the project.

Once the appropriate reference stars are located, we need to perform the aperture radius test on these stars as well. This was performed using the code below:

```
        # Plotting the Aperture vs flux for all the reference stars as well

# defining the location of reference stars
reference1Location = (325,300)
reference2Location = (184,636)
reference3Location = (1157,315)
reference4Location = (51,848)

# defining a function that automates the aperture vs flux graphs
def ApertureVsFlux(location):
  # Defining aperture size
  radiusAperture = np.arange(1,21,0.2)

  calculatedFlux = []
  for radius in radiusAperture:
    aperture = CircularAperture(location, r=radius) # defining the aperture
    phot_table = aperture_photometry(alignedTransitImages[280], aperture) # calculating
    ↪  the flux value
    calculatedFlux.append(phot_table['aperture_sum'][0].value)

  plt.plot(radiusAperture,calculatedFlux,label='Calculated Flux at ' + str(location))

ApertureVsFlux(reference1Location)
ApertureVsFlux(reference2Location)
ApertureVsFlux(reference3Location)
ApertureVsFlux(reference4Location)

```

```
# Adding a line at aperture size, 4,5,6,7
plt.axvline(x=4, color='m', linestyle='--',label='aperture = 4')
plt.axvline(x=5, color='b', linestyle='--',label='aperture = 5')
plt.axvline(x=6, color='purple', linestyle='--',label='aperture = 6')
plt.axvline(x=7, color='r', linestyle='--',label='aperture = 7')

# Describing the plot
plt.legend()
plt.xlabel('Radius of Aperture (pixels)')
plt.ylabel('Flux (electrons)')
plt.title('Aperture vs Flux for transit Star');
```

Listing 5.3: *finding the appropiate aperture for reference stars*

Upon running the code mentioned in listing 5.3 we have received the following output:

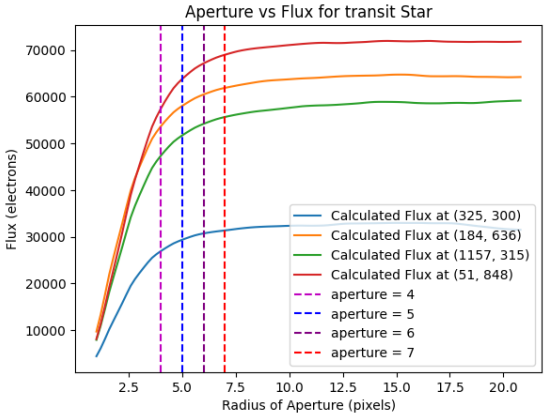


Figure 5.4: *Aperture test for reference stars*

Upon running the aperture test for all reference stars, we notice that the appropriate aperture radius for all of these reference stars are 6 pixels. The location of the reference stars used in respect to the transit star was displayed using the following code:

```
        # displaying the reference star locations
plt.imshow(alignedTransitImages[280],cmap='gray')
plt.plot(reference1Location[0],reference1Location[1],'ro', label='Reference Star 1')
plt.plot(reference2Location[0],reference2Location[1],'bo', label='Reference Star 2')
plt.plot(reference3Location[0],reference3Location[1],'go', label='Reference Star 3')
plt.plot(reference4Location[0],reference4Location[1],'mo', label='Reference Star 4')
plt.legend()
plt.title('Reference Star Locations')
plt.xlabel('X-axis (pixels)')
plt.ylabel('Y-axis (pixels)');
```

Listing 5.4: *Location of Reference Stars*

Upon running the code mentioned in listing 5.4 we have received the following output:

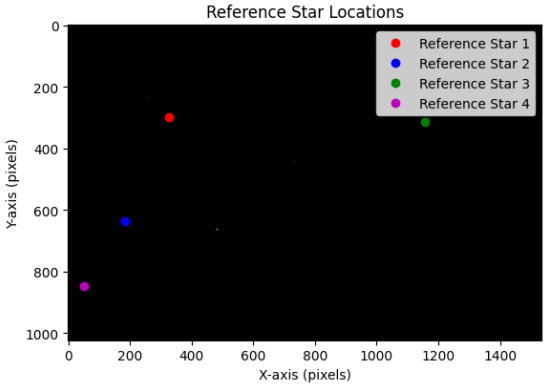


Figure 5.5: *Location of Reference Stars*

Once we have the reference stars and their apertures, we calculate the flux coming from all of our reference stars as well as the transit star. This was performed using the following code:

```
        # Annulus Background subtraction was not performed since we didn't have data descrepencies

# For differential photometry we need to calculated the flux of multiple stars (including reference stars)
# thus we'll define a function to calculate the flux for us
def calculateFlux(location, radiusAperture): # accounting for aperture radius differences across different reference stars

  # defininf the aperture
  aperture = CircularAperture(location, r=radiusAperture)

  flux = []
  for i, image in enumerate(alignedTransitImages):
    phot_table = aperture_photometry(image, aperture)
    flux.append(phot_table['aperture_sum'][0].value)
    print(f'Calculated Flux for image: '+str(i+1)+' of ' + str(len(alignedTransitImages)))
    clear_output(wait=True)
  return flux

# Since we previously calculated, the aperture radius for the transit star is 6
# Note that the aperture sizes for the reference stars have been calculated the same way as the transit star

transitFlux = calculateFlux(HatP13BStarLocation,6)
reference1Flux = calculateFlux(reference1Location,6)
reference2Flux = calculateFlux(reference2Location,6)
reference3Flux = calculateFlux(reference3Location,6)
reference4Flux = calculateFlux(reference4Location,6)
```

26

Listing 5.5: *Calculating the flux of the interstellar objects*

This flux was then visualized by plotting using the following code:

```
        # Plotting the flux data calculated
plt.plot(transitFlux,'b',label='Transit Star')
plt.plot(reference1Flux,'r',label='Reference Star 1')
plt.plot(reference2Flux,'purple',label='Reference Star 2')
plt.plot(reference3Flux,'g',label='Reference Star 3')
plt.plot(reference4Flux,'m',label='Reference Star 4')
plt.legend()
plt.title('Flux vs Number of Images')
plt.xlabel('Number of Images')
plt.ylabel('Flux (electrons)');
```

Listing 5.6: *Code to visualize the incoming flux*

Upon running the code mentioned in listing 5.6 we receive the following output:

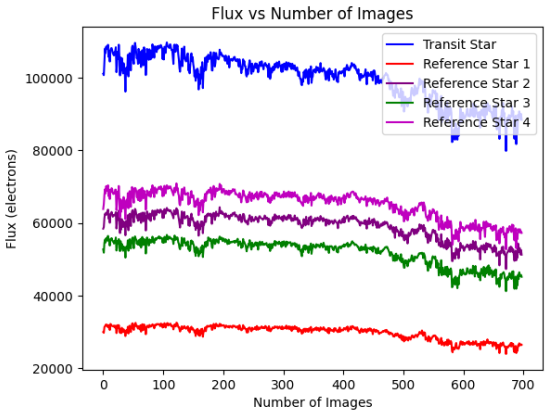


Figure 5.6: *Flux captured by interstellar objects over time*

Upon examining the flux plots of the interstellar objects, we notice that the charts have similar dips. These dips are caused by variations in the atmosphere. This could be leveraged by differential photometry where we take the weighted average of flux from the reference stars. This was done using the following Code:

```
        # Plotting the Weighted Average from the reference stars

# Calculating the weights using the inverse variance
weights = [1/(np.std(reference1Flux)**2), 1/(np.std(reference2Flux)**2),
↪  1/(np.std(reference3Flux)**2)]
weights = [weight/sum(weights) for weight in weights] #normalizing the weights

# calculating the weightage average
```

```
weightedAverage = (weights[0]*np.array(reference1Flux) +
↪  weights[1]*np.array(reference2Flux) + weights[2]*np.array(reference3Flux)) /
↪  np.sum(weights)

# Plotting the weighted average
plt.plot(weightedAverage,'r',label='Weighted Average')
plt.title('Weighted Average')
plt.xlabel('Number of Images')
plt.ylabel('Flux (electrons)');
```

Listing 5.7: *Calculating the weighted average of flux*

Upon running the code mentioned in 5.7 we receive the following output:

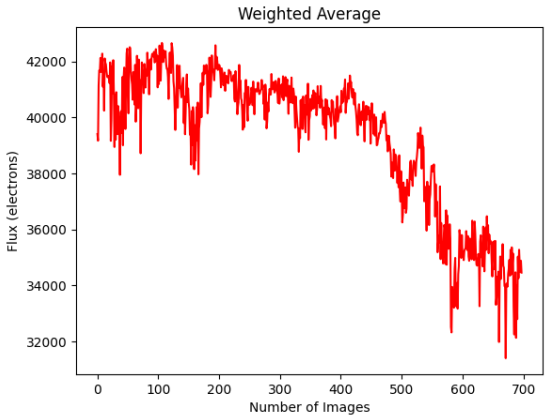


Figure 5.7: *Weighted average of the flux of reference stars*

This weighted average was then divided by the flux received by the transit star to receive our final transit light curve. This was performed using the following code:

```
        #calculating the transit curve
transitCurve = transitFlux / weightedAverage

# Normalizing this transit curve for future calculations
noTransitFlux = np.concatenate([transitCurve[:100], transitCurve[598:]])
baselineFlux = np.mean(noTransitFlux)
transitCurve = transitCurve / baselineFlux

# calculating the line of best fit
coefficient = np.polyfit(np.arange(1,len(transitCurve)+1),transitCurve,7)
lineBestFit = np.polyval(coefficient,np.arange(1,len(transitCurve)+1))

#plotting the transit curve
plt.plot(transitCurve,'b',label='Transit Curve')
plt.plot(lineBestFit,'r',label='Line of Best Fit')
plt.legend()
plt.title('Transit Curve')
plt.xlabel('Number of Images')
```

```
19 plt.ylabel('Flux (electrons)');
```

Listing 5.8: *Calculating the transit curve*

Note that we have also normalized our curve for ease of calculations. This also helps in calculating the Transit depth much more intuitively. Upon running the code mentioned in listing 5.8 we receive the following output:

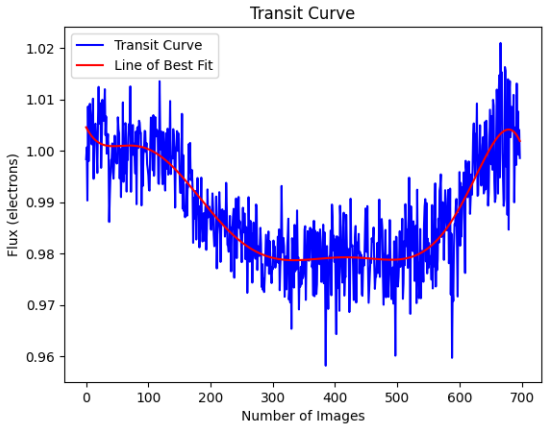


Figure 5.8: *Transit Light Curve of HAT-P-18 b*

The image displayed by this image is our final transit light curve which could be used for further calculations. This also marks the end of the Flux data Reduction. To summarize, we have done the following:

1. Identify the transit and comparison stars
2. Perform aperture tests
3. Differential Photometry

6

Planet Related Calculations

The transit data could be used to perform calculations that tell us a lot about the shape and composition of the planet itself (Charbonneau, 1999). To do this, HAT-P-18 properties mentioned in 2.2 were used. The code that does the calculations is as follows:

```
        # labeling the ingress start/end and the egress start/end

#plotting the transit curve
plt.plot(transitCurve,'b',label='Transit Curve')
plt.plot(lineBestFit,'r',label='Line of Best Fit')
plt.axvline(x=100, color='b', linestyle='--',label='Ingress Start')
plt.axvline(x=250, color='g', linestyle='--',label='Ingress End')
plt.axvline(x=550, color='y', linestyle='--',label='Egress Start')
plt.axvline(x=670, color='purple', linestyle='--',label='Egress End')
plt.legend()
plt.title('Transit Curve')
plt.xlabel('Number of Images')
plt.ylabel('Flux (electrons)');

# to calculate the minimum value, we can calculate the mean of the data between Ingree
↪  End and Egress Start
minFluxValue = transitCurve[250:670].mean()
print('Minimum Flux Value: ' + str(minFluxValue))

# calculating the transit depth
transitDepth = 1- minFluxValue
print('Transit Depth: ' + str(transitDepth))

# radius of the Star
# we know that the radius of the star HAT-P-18 is 0.68 * Our Sun
# cite: https://science.nasa.gov/exoplanet-catalog/hat-p-18-b/
radiusSun = 696340 #km
radiusStar = 0.75 * radiusSun

# calculating the radius of the planet
# we know that (r_planet / r_star) = sqrt(delta_flux / flux)
radiusPlanet = radiusStar * np.sqrt(transitDepth / transitCurve[405])
```

```
32 print('Radius of the Planet: ' + str(radiusPlanet) + ' km')
33
34 # We have calculated the radius of the planet as 69112 km which is pretty close to the
   ↪  actual value
```

Listing 6.1: *Planet Calculations*

Executing the code mentioned in listing 6.1 provided us with the following output:

```
Minimum Flux Value: 0.9829553087665456
Transit Depth: 0.017044691233454423
Radius of the Planet: 69112.5122071523 km
```

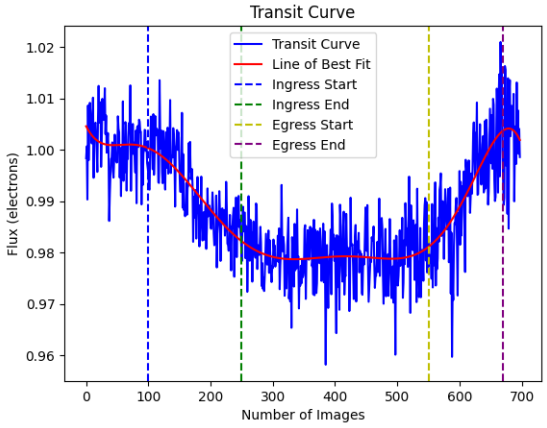


Figure 6.1: *Transit Light Curve of HAT-P-18 b Labelled*

7

Results and Discussion

By using the formula $transit\ depth(\delta) = (\frac{R_{planet}}{R_{stellar}})^2$ and analysing the data received by the output of the code mentioned in listing 3.1 we gather the following information:

Name	Value	Units	Description
Transit Depth	1.7	%	Depth of the transit caused by the planet
Planet Radius	69,112	km	Radius of the planet in kilometers
Planet Radius	10.85	$R_\oplus$	Radius of the planet relative to Earth
Planet Radius	0.989	R_J	Radius of the planet relative to Jupiter
Ingress Duration	105.18	min	Time taken for the planet to fully enter the star's disk
Egress Duration	96.03	min	Time taken for the planet to completely exit the star's disk
Total Transit Duration	91.08	min	Total time of the planet's transit across the star

Table 7.1: *HAT-P-18 b Planetary Properties* [*Tokyo, n.d.*]

To improve the observational accuracy of the star, four nearby reference stars were used to identify the host star, ensuring reliable target selection. Atmospheric variations present during the observation were removed by calculating the weighted average of reference star flux values. This approach effectively reduced environmental distortions, leading to a reliable transit light curve.

While the current method using weighted averages for differential photometry was sufficient, future observations could explore additional methods such as Gaussian process regression or atmospheric modeling to enhance accuracy. To improve future observations, the following steps are recommended:

- Prepare detailed star plots and simulate transit curves in advance to minimize uncertainties during alignment.
- Optimize the timing of observations to ensure longer baselines before and after the transit.
- Enhance equipment calibration processes, particularly for flat-fielding and bias corrections.
- Explore automated tools for star alignment to reduce manual errors.

Star alignment uncertainties were reduced during the data reduction process. However, integrating automated alignment tools and implementing redundancy checks could further minimize potential errors in future projects. To conclude, the observation successfully captured HAT-P-18 b's transit, with results closely aligning with published values. Despite challenges in coding, equipment, and observational planning, this project demonstrated the potential for improved accuracy in future work through refined techniques and enhanced preparation.

Bibliography

Charbonneau, David (1999). "Detection of Planetary Transits Across a Sun-like Star". In: *The Astrophysical Journal*. Accessed: 2024-11-10. URL: https://iopscience.iop.org/article/10.1086/312457/meta.

D, Bolles. (2024). "HAT-P-18 b". In: *NASA Exoplanet Catalog*. Accessed: 2024-11-09. URL: https://science.nasa.gov/exoplanet-catalog/hat-p-18-b/.

Jones, T. (n.d.). "Types of Stars". In: *AstroBackyard* (). Accessed: 2024-11-9. URL: https://astrobackyard.com/types-of-stars/.

Tokyo, Exoplanet (n.d.). "HAT-P-18". In: (). Accessed: 2024-11-10. URL: https://www.exoplanetkyoto.org/exohtml/HAT-P-18.html.

VarAstro (n.d.). "HAT-P-18 b". In: *Exoplanet Catalog* (*ETD*) (). Accessed: 2024-11-10. URL: https://var.astro.cz/en/Exoplanets/387.

Appendices

A

Appendix A

The programming language Python was used to develop visuals and perform calculations related to this project. The Python code can be found at here.

If the Hyperlink doesn't work, please copy and paste the following link at the URL bar:
`https://colab.research.google.com/drive/1XXxJX7_yDx-V1UrZpLohpQQtX9ga1m9G?usp=sharing`

Additionally, ChatGPT was used to understand the function of libraries for certain parts of the assignment. The ChatGPT log could be found here

`https://chatgpt.com/share/673088f9-c5e4-8006-a1da-1263ee5746f1`

If the link to the code doesn't work, please contact Sarthak Sahai at sahai@my.yorku.ca.

B

Appendix B

The Following is a paragraph dedicated to the usage of ChatGPT in this project:

ChatGPT has been an incredibly helpful tool in allowing me to complete the project both efficiently and effectively. With access to a vast knowledge base, I was able to save countless hours that would have otherwise been spent researching how specific libraries or classes . It assisted me in researching complex topics and provided valuable guidance on optimizing my code. Its ability to adapt to my questions and provide instant feedback made the project completion process much smoother, helping me stay focused. In conclusion, ChatGPT was an invaluable resource, enabling me to understand Python libraries and classes that I had no prior experience with.