# Looking for Brown Dwarf Binaries with a PSF fitting Python Script

Emily Welch

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Denise Stephens, Advisor

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Looking for Brown Dwarf Binaries with a PSF fitting Python Script

Emily Welch
Department of Physics and Astronomy, BYU
Bachelor of Science

Brown dwarfs form like stars but are not massive enough to fuse hydrogen fusion in the core. They form between 0.013 and 0.072 solar masses and are difficult to detect because of their low luminosities. Brown dwarfs act as point sources that when imaged spread out their light on the CCD in predictable patterns. These patterns are known as point spread functions (PSF). It is because of this ”spreading” that brown dwarf binaries are unresolved when there is small angular separation. Kyle Matt has produced a python script that uses PSF models to determine binarity for unresolved brown dwarfs. I have tested this code and have found some problems in comparing the single and binary fits, but have used the metric of flux percentage from the script output to suggest probable binaries.

Keywords: [Brown Dwarfs, BD-BD binaries, Binary fraction, IMF, Python Code, PSF fitting, NICMOS, TinyTim]

ACKNOWLEDGMENTS

This project primarily builds on the work of Kyle Matt, Keith Noll and Denise Stephens. I personally thankful to Dr. Stephens for her guidance and support throughout this research project and my undergraduate coursework. I would like to thank the BYU Physics and Astronomy Department for funding this project. And Alexandra Gerday, Jan Gerday, Ethan Welch and Heidi Lawrence, who also assisted me in editing this document.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Introduction to Brown Dwarfs

Stars are formed as cold molecular clouds of hydrogen and helium collapse due to gravity. As protostars contract, those with sufficient mass will eventually meet the temperature and pressure conditions for hydrogen fusion. Radiation pressure from hydrogen fusion then balances out the gravitational collapse, and stars will spend most of their lives in this phase as main-sequence stars.

Brown dwarfs (BD) are the less massive siblings of stars and have masses below 0.080 solar masses. They also form through gravitational collapse but are not massive enough for hydrogen fusion to occur in the core. Before they can achieve the conditions for hydrogen fusion, electron degeneracy will halt any further collapse.

Shiv S. Kumar first proposed the existence of brown dwarfs, which he then called ”black dwarfs” in 1962. His calculations showed that a stellar object below 0.08 solar masses would not have the pressure and temperature necessary to fuse hydrogen. (Kumar 1962) The first brown dwarf, Teide 1, was discovered by Dr. Rafael Rebolo in 1995 in the Pleiades Cluster. These objects were notoriously difficult to detect, as their relatively small size and cool temperatures means they are

not very luminous in the visible range. However, they are bright in the near-infrared and since the era of near-infrared all-sky surveys such as 2MASS and DENIS many more brown dwarfs have been found.

## 1.2 Importance of Binary Determination

Binary stars offer a wealth of information about the component stars, most notable the mass. With main sequence stars it is possible to also determine the mass of a star based on a luminosity, but with BDs there is no such steady mass-luminosity function. So astronomers rely on the photometric, astrometric and spectroscopic observations of BD binaries in order to form a better picture of brown dwarf characteristics. Konopacky's 2013 article entitled "The Fundamental Importance of Brown Dwarf Binaries" surveys known brown dwarf binaries and details how to determine the age, metallically, atmospheric composition, as well as other orbital and rotational parameters of various characteristics of brown dwarfs binaries.

## 1.3 Introduction to the BD IMF

While BD binary systems reveal much about BDs, BD binary determination is important to our understanding of stellar evolution. Because they are difficult to observe we are not sure how many brown dwarfs there are or what ratio of stars are BDs. The distribution of different stellar masses is called the stellar initial mass function (IMF). In 1955, Edwin E. Salpeter published the first estimate of the IMF based on stars in our solar neighborhood that were between 0.4 and 10 solar masses. He published a simple power law function, included as Eq. 1.1, with $\alpha = 2.35$ called the Salpeter function which indicates that there should be fewer stars in each mass group as you increase mass.

$$\Phi(m)dm \propto m^{-\alpha}dm; \tag{1.1}$$

This implied a diverging mass density as $m \to 0$. (Salpeter & E. 1955) In Fig. 1.1 this is evident in the plot as you look at the blue function at the low mass end of the plot.

Following Salpeter's seminal paper, there were many other researchers that published their own determinations of the IMF. As observational techniques improved, it became possible to observe low-mass stars and brown dwarfs in surveys which led to a flattening in the IMF model described by Glenn E. Miller and John M. Scalo in 1979. They determined that $\alpha = 1$ for $m \leq 1$ solar mass and and that for $m > 1$ solar mass the alpha value agreed with Salpeter's model. This is illustrated by the green function in see Fig. 1.1 which goes noticeably flat at about $m = 1$ solar mass, indicating that for stars mess than a solar mass there are about the same number of stars at each mass. They also noted that compared to theoretical calculations many clusters appeared to be deficient in lower mass stars, which supports their model. (Miller & Scalo 1979)

In 2002, Gilles Chabrier presented a BD IMF that has since become the standard. He gave the alpha value of $\alpha = 1.55$ for stars with with $m \leq 1$ solar mass and $\alpha = 2.7$ for $m > 1$ solar mass. (Chabrier 2002) Which he later further refined to differentiate between stellar binaries and individual stellar objects. (Chabrier 2003) These are presented in Fig. 1.1 below for better visualization of the IMFs. In the plot the cyan function represents Chabrier's model for individual BDs while the purple function is for stellar-binaries.

Dr. Laird Close from University of Arizona has discovered a number of binary brown dwarfs and believes that BD-BD binaries are very common. He estimates that 32-45% of brown dwarfs are in fact binaries, and that our current model of the BD IMF may not be representative of the actual population of stars.(Close et al. 2003) Konopacky believes that the overall multiplicity fraction is somewhere between 10-30%. (Konopacky 2013)

The accuracy of these IMFs is limited by uncertainty in the frequency of BDs. The substellar objects are difficult to observe and the frequency of BDs in multiple systems is unknown. Current estimates of the BD IMF, such as that of Chabrier in 2002, use a correction for unresolved binaries

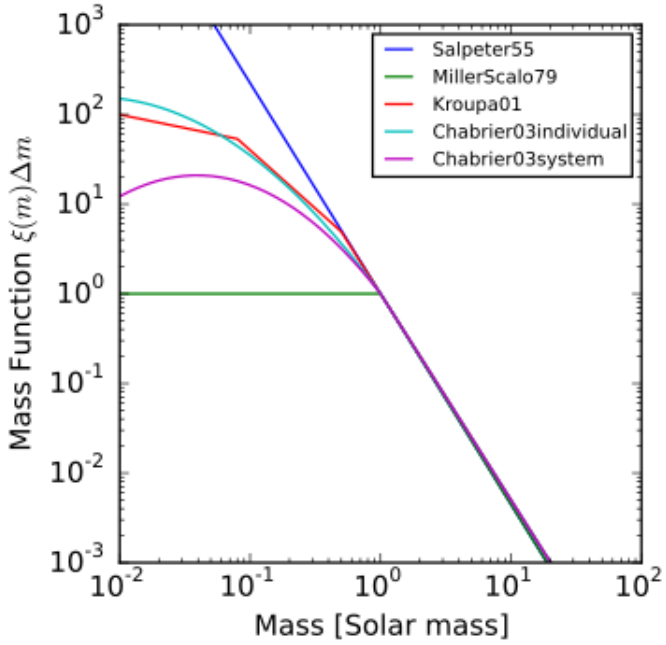


**Figure 1.1** A plot of 5 different IMFs published by Johannes Buchner in 2015 in the the Wikipedia article about IMFs.

based on the assumption that BD binaries are uncommon and that unresolved binaries will simply look like a brighter star.

## 1.4 Introduction to PSF Fitting

To resolve the problem of the unknown multiplicity fraction of brown dwarfs, we look at identifying binaries through the method of point spread function (PSF) fitting. The brown dwarfs we are studying have an angular size that is far smaller than the maximum pixel resolution of a CCD. We term objects like this, point sources. When taking an image of a point source, it appears to "spread out" in the image. This is caused by atmospheric conditions, that can change throughout the observation window, telescope optics, other diffraction of light. Because this project worked with Hubble data, we avoided the effects of atmosphere but we will need to consider diffraction due

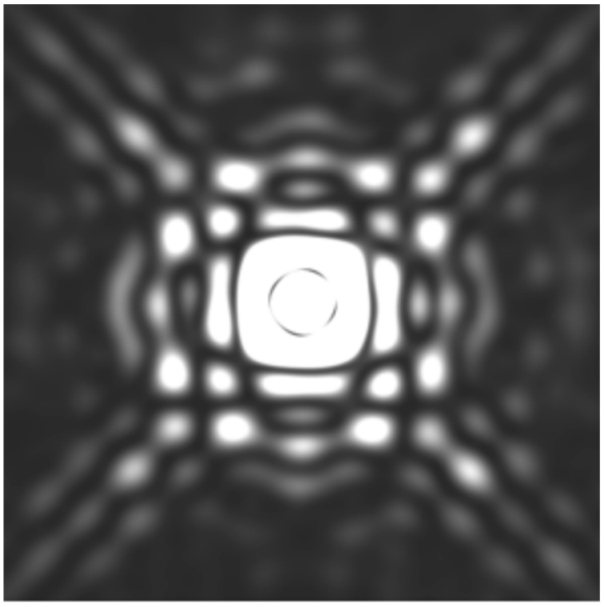

**Figure 1.2** A model PSF that was created using Tiny Tim, the pattern of the light "spread" out is caused by the diffraction of light as it enters the telescope and interacts with the physical structure and optics.

to the optical distortion of the detector, the thickness of the detector and the discrete sampling of the CCD. (Trujillo et al. 2001) Other variations in the PSF will depend on the object being imaged and the position of the image on the CCD. Included in Fig. 1.2 is an example of a model PSF which shows how the point source's light is spread. The model PSFs we used for fitting the data were all created with Tiny Tim. (Krist & Hook 2004)

# Chapter 2

# Code for PSF Fitting

This thesis focuses on testing the PSF Fitting code created by Kyle Matt. Kyle MattâĂŹs code consists of a python script based on FORTRAN code already in use by Stephens and Noll (2006). The FORTAN code effectively determined binaries but required a lot of manually editing various parameter files. Kyle Matt wrote this new script because he and Dr. Denise Stephens thought that a python code would be more user-friendly. (Matt 2017)

## 2.1 Operating Principle

The PSF fitting code written by Kyle Matt, PSFfit.py, works iteratively. To determine the best single BD fit, the PSF model created by Tiny Tim for the BD is tested at 0.1 pixel increments in the x and y directions of the user-given central pixel. Each of these 100 iterations is then compared to the data and the position with the smallest $\chi^2$ value is kept as the best single fit.

To find the best binary fit, the script uses two different PSF models. The program will test different placements of the two PSF with the primary in the central user given pixels and the secondary being in one of the pixels contained in the 3 by 3 grid around that it. And calculates the error of each fit. It is then possible to compare the best binary and best single fit.

## 2.2 Error Calculation

To determine the best fit single, Matt's code tests each of the 100 single subsampled PSF models against the data to determine the smallest $\chi^2$ value. In equation 2.1 below the observed corresponds to the calibrated data and the expected refers to the models we are using to fit the data.

$$\chi^2 = \frac{(observed - expected)^2}{expected}; \tag{2.1}$$

Essentially, the model is subtracted pixel by pixel from the data and the $\chi^2$ is calculated for each of those models. Once the function f.minchi chooses one of those 100 models f.fitpsf1 uses that model and finds the flux that gives the best fit. The fit information is printed out by PSFfit.py and includes the flux and $\chi^2$ value as well as the iteration number which tells us how long it took for the code to converge on a solution and the PSF model number which gives the central position of the PSF inside the pixel. The $\chi^2$ value is only considers the residuals of the central pixel and the eight surrounding.

For the binary fit it is necessary to find two PSF models that together create the best binary fit. This is done iteratively by the function f.twospf and f.tworef. This is also done by the technique of minimizing the $\chi^2$ value to determine which fit is better. However, in the printed output of the code returns the sum of the residual errors rather than the $\chi^2$ value. This also only considers the central pixel and the eight surrounding it.

$$Residual = |observed - expected|; \tag{2.2}$$

To illustrate the difference in these errors I would like to point out that if the expected value in a pixel is 0.8 and the observed value is 1 the residual is 0.2 while the $\chi^2$ value will be 0.05.

The table 2.1 lists various functions used in PSFfit.py to determine the single and binary fits.

| Single Fit Functions | |
|---|---|
| f.minchi | choses the PSF whose subsampled position minimizes the $\chi^2$ value |
| f.fitpsf1 | determines which flux best minimizes $\chi^2$ |
| Binary Fit Functions | |
| f.twopsf | Chooses the primary's PSF working iteratively with f.tworef |
| f.tworef | Determine the best secondary position within a pixel and determines the relative flux distribution |
| f.fitpsf2 | Determines the total flux of the system |
| f.scpsf2 | Works iteratively with fitpsf2 to determine the flux of the primary and secondary |
| f.rfpsf | Uses the flux to determine the $\chi^2$ value value. |

**Table 2.1** The list of functions written Kyle Matt which determine the best fits and the error.

# Chapter 3

# Method and Procedure

In Kyle Matt's paper "Using Model Point Spread Functions to Identify Possible Binary Brown Dwarf Systems," he outlines the use of the python code, PSFfit.py. This code is designed to help identify BD binaries and uses the technique developed by Stephens and Noll in 2006 to detect binaries among trans-neptunian objects.

## 3.1 Raw Data

PSFfit.py was specifically written for use with data collected with the Near Infrared Camera and Multi-Object Spectrometer (NICMOS) which is on the Hubble Space Telescope (HST).

The NICMOS 1 camera is 256 x 256 square pixels and is made up of four 128 x 128 square detectors, also called quadrants, which each work independently. Each pixel in the detector array is 0.043" and the camera has a field of view that is 11" by 11".

Because the four detectors in the array work independently they each have different levels of background dark current, read noise and overall responsiveness. The hot and cold pixels in each are completely independent of each other, as are the pieces of "grot" or debris on the detector. The electronics for each quadrant also create different levels of amplifier glow and it is necessary when

looking at the data to produce a uniform background level. (Viana 2009)

The data from HST includes for each image a raw data file and one that has already been partially calibrated. This file will have the form **filename_cal.fits.** In section 3.3.2 of the NICMOS Data Handbook it is explained that this cal.fit image has already been corrected using the zeros, darks and flats which are standard to any data processing in astronomy. This file also has bad pixels and cosmic rays flagged as well as a correction for the non-linear response of the detectors.

## 3.2 Calibrating the Image

To further reduce the NICMOS data we are using PyRaf, which is the python version of IRAF (Image Reduction and Analysis Facility). In PyRaf you will need to load the Space Telescope Science Data Analysis System (stsdas), the Hubble calibration package (hst_calib) and the nicmos package which contain tasks specific to calibrating the NICMOS cameras. In summary, while in PyRaf you will want to type in these commands to access the specific data reduction tasks:

**stsdas**

**hst_calib**

**nicmos**

The command pedsky in the nicmos package removes an estimate of the background sky and quadrant-dependent (detector- dependant) residual bias (collectively these are known as the pedestal). This is done by ignoring pixels that likely contain signals, and by determining the background levels of the sky and for each image quadrant. This leaves you with an image with the minimum rms deviation in pixel values.

The pyraf routine rnlincor corrects for non-linear responsivity of the NICMOS detectors. The program assumes that the non-linear count rate behaves like a power-law and is wavelength dependent, this allows for linearization of our counts and for the computation of an accurate

magnitude. (Thatte et al. 2009) To run these commands for the calibration image n9u608knq_cal.fits you type:

**pedsky n9u608knq_cal.fits n9u608knq_ped.fits**

**rnlcor n9u608knq_ped.fits n9u608knq_rnl.fits**

This image is now ready for PSF fitting.

## 3.3 Creating the PSF

To create the model PSF we are using the program created by John Krist and Richard Hook. Tiny Tim is written specifically to make PSF models for use with HST instruments. Tiny Tim consists of three scripts - tiny1, tiny2, and tiny3.

The first asks a series of basic questions and generates a parameter file that is read by tiny2, which then generates the PSF. (Krist & Hook 2004)

The first input needed for tiny1 is the camera name. Kyle Matt wrote PSFfit.py for use with NICMOS cameras 1 and 2, which are options 19 and 20 in tiny1. Then the user gives the the position of the point source on the CCD. Instead of giving an exact position we have divided the CCD into various regions as illustrated in figure 4. This allows us to reuse the model PSFs for various images that have the same spectral type, filter, and region of the CCD. (Krist & Hook 2004) The CCD is square and 256 pixels across. I have divided it radially into 12 parts and concentrically into 3 rings. In Fig. 3.1 the blue dots are in the regions m11, m1, i2, and o2 moving from left to right.

PSFs are wavelength and color-dependent so the filter used and spectral type of the BD-BD candidate are key to making the right model PSF. Patterns of spokes and rings seen in the model PSFs depend on the wavelength, the width of the filters, and the structure of the BD. For a narrow filter the filter name is sufficient, but for a medium or wide band filter you need to give the spectra

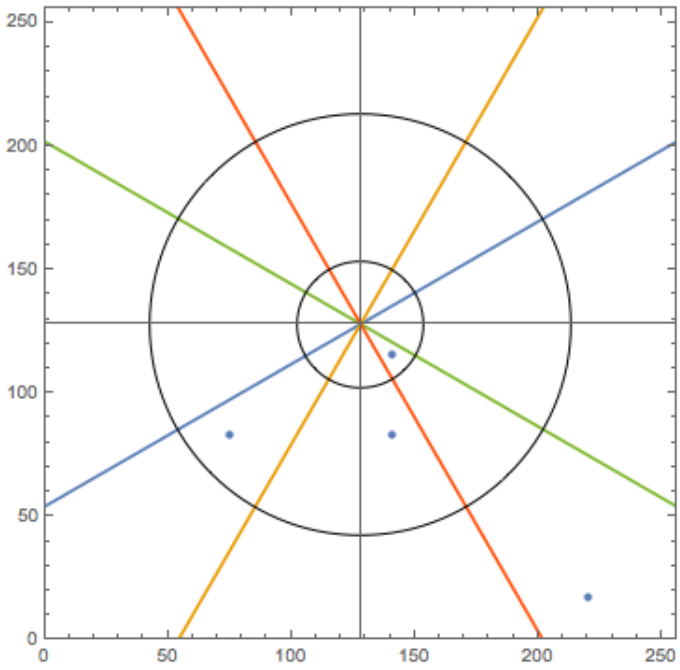


**Figure 3.1** This plot represents a 256 pixel square divided into 12 radial slices and 3 concentric circles. Each position on the CCD can be classified by the region it falls into.

of the object. (With narrow band filters the bandpass is too narrow to show color effects). We are reading in spectra from ASCII table based on the spectral type of the brown dwarf. We first determine the spectral type of each candidate BD-BD and then we take the spectra of a real BD with the same spectral type to be read into TinyTim. It is necessary to use a real BD spectra to get an accurate model PSF because of the significant structure of BDs.

As recommended for use with Kyle's code, I have chosen to generate a PSF with a diameter of 3 arc seconds and to use a subsampling factor of 10. (Matt 2017)

The naming convention I have used names the spectral types, filter, and then the region of the CCD. So for a L3 brown dwarf whose image was taken in the filter F110W, and is in the region m1 of the CCD I would create a model PSF that would be called L3F110Wm100.fits. This model PSF is included previously as Fig. 1.2.

## 3.4 PSF fitting

Once you have calibrated the image and created the model PSF it is time to do the PSF fitting. In the directory which contains the rnl.fits image you will need to add the model PSF and the files PSFfit.py, Functions.py and Cameras.json.

Then in the terminal you will open the directory and type the command. Take, for example, n9u608kng_rnl.fits which is an image of an L3 brown dwarf taken in the F090M filter with the brown dwarf in the o1 region of the CCD you would type:

**python3 PSFfit.py L3F09Mo100.fits n9u608knq_rnl.fits**

And then when prompted you will enter the coordinates of the BD.It is important to enter the exact coordinates of the brightest pixel as the center of the primary PSF because Matt's code will only search that pixel and the 8 neighboring pixels for a secondary PSF. If you are a pixel off the program may not be able to determine the location of the secondary PSF because it can not search any other pixels.

The program will then produce an output file with the results of the best fits for each of those nine pixels as the location of a secondary PSF.

# Chapter 4

# Data

After running each rnl.fits image through PSFfit.py with the PSF model, an output file is produced. The set of output files for all the images of each star constitute the data set with which binarity is determined. We consider here only BDs with at least four images.

## 4.1 Interpretation of the output

In Fig. 4.1 we see a complete output file. Which includes information about the best single fit and one best secondary PSF fit for each of the nine pixels investigated. These include the central user-given pixel and the eight surrounding it.

Under the header ”Single Fit,” there are several pieces of information included. First the number of iterations, then ID of the model PSF which fit best, the flux determined by f.fitpsf1 and the $\chi^2$ error. This fit assumes that the BD is in the coordinate that supplied by the user while running PSFfit.py. The information is displayed as shown in Fig. 4.2. The iteration number tells us how long it took the code to converge to a solution. And the ”psf number” is really telling us the central position of the PSF inside that pixel, as determined by function f.minchi described in Table 2.1. The observed flux is included and also a $\chi^2$ value which tells us the certainty of the fit. In section

```
n9u608knq - F090M
UT 2006-10-10 11:06:44
Single Fit:
iter: 2  psf:63  flux: 392.8500  minchi: 181.1846
       ---
Binary Fit:
1   error:0.62322
 Primary: (130.83, 27.45)  Secondary: (130.73, 26.84)
 Primary PSF:   60 f:  225.4371 (54.98%)
 Secondary PSF: 76 f:  184.6206 (45.02%)
 Position Angle:  83.88 Separation: 0.0265
 Primary Magnitude: 15.273 Secondary Magnitude: 15.489

2   error:0.68438
 Primary: (130.83, 27.26)  Secondary: (130.44, 26.15)
 Primary PSF:   62 f:  368.3451 (89.34%)
 Secondary PSF:  3 f:   43.9640 (10.66%)
 Position Angle:  73.72 Separation: 0.0502
 Primary Magnitude: 14.740 Secondary Magnitude: 17.047

3   error:0.71396
 Primary: (130.73, 26.95)  Secondary: (130.83, 27.55)
 Primary PSF:   75 f:  247.6318 (60.52%)
 Secondary PSF: 69 f:  161.5638 (39.48%)
 Position Angle:  83.76 Separation: 0.0261
 Primary Magnitude: 15.171 Secondary Magnitude: 15.634

4   error:0.77530
 Primary: (130.83, 27.26)  Secondary: (130.54, 26.26)
 Primary PSF:   62 f:  364.0230 (88.79%)
 Secondary PSF: 92 f:   45.9702 (11.21%)
 Position Angle:  76.80 Separation: 0.0448
 Primary Magnitude: 14.752 Secondary Magnitude: 16.999

5   error:0.87274
 Primary: (130.83, 27.26)  Secondary: (130.44, 26.55)
 Primary PSF:   62 f:  352.7193 (86.91%)
 Secondary PSF:  9 f:   53.1189 (13.09%)
 Position Angle:  64.36 Separation: 0.0348
 Primary Magnitude: 14.787 Secondary Magnitude: 16.843

6   error:0.89762
 Primary: (130.83, 27.15)  Secondary: (130.44, 27.84)
 Primary PSF:   63 f:  369.5523 (92.07%)
 Secondary PSF:  6 f:   31.8466 ( 7.93%)
 Position Angle: 122.92 Separation: 0.0341
 Primary Magnitude: 14.736 Secondary Magnitude: 17.397

7   error:0.90274
 Primary: (130.73, 27.15)  Secondary: (131.54, 28.26)
 Primary PSF:   73 f:  381.1771 (93.31%)
 Secondary PSF: 92 f:   27.3107 ( 6.69%)
 Position Angle:  57.19 Separation: 0.0587
 Primary Magnitude: 14.702 Secondary Magnitude: 17.565

8   error:1.06824
 Primary: (130.73, 27.15)  Secondary: (131.54, 26.45)
 Primary PSF:   73 f:  388.6877 (97.97%)
 Secondary PSF: 90 f:    8.0666 ( 2.03%)
 Position Angle: 142.21 Separation: 0.0460
 Primary Magnitude: 14.681 Secondary Magnitude: 18.889

9   error:1.19919
 Primary: (130.73, 27.15)  Secondary: (131.54, 27.45)
 Primary PSF:   73 f:  373.2755 (93.12%)
 Secondary PSF: 90 f:   27.5956 ( 6.88%)
 Position Angle:  23.64 Separation: 0.0370
 Primary Magnitude: 14.725 Secondary Magnitude: 17.554
```

**Figure 4.1** Output file for the image n9u608knq_cal.fits taken by the NICMOS 1 camera on HST the 10th of October 2006. 2MASSWJ0036159+182110.

```
Single Fit:
iter: 2  psf:63  flux: 392.8500  minchi: 181.1846
```

**Figure 4.2** Output file for the image n9u608knq_cal.fits taken by the NICMOS 1 camera on HST the 10th of October 2006. 2MASSWJ0036159+182110.

```
Binary Fit:
1   error:0.62322
 Primary: (130.83, 27.45)  Secondary: (130.73, 26.84)
 Primary PSF:   60 f:  225.4371 (54.98%)
 Secondary PSF: 76 f:  184.6206 (45.02%)
 Position Angle:  83.88 Separation: 0.0265
 Primary Magnitude: 15.273 Secondary Magnitude: 15.489
```

**Figure 4.3** Output file for the image n9u608knq_cal.fits taken by the NICMOS 1 camera on HST the 10th of October 2006. 2MASSWJ0036159+182110.

2.2 of this paper a discussion about the computation of the error was included, and the equation for $\chi^2$ is included at equation 2.1. A value of 181 is very high and suggests that the single fit is not very good.

The binary fit information in Fig. 4.1 is broken into 9 sections for the 9 different pixels in which PSFfit.py looks for the secondary BD PSF. Each section of the binary fit output has the same form as Fig. 4.3.

The nine outputs include the best fit for each pixel about the central pixel with the primary being in the central pixel and the secondary being in that particular pixel. That means that if the secondary is in the middle of a pixel there should be one good fit,if the secondary is at the border at a pixel there may be two good fits, and at the corner there may be four accurate fits.

The Fig. 4.4 shows the nine different coordinates from the binary fit section of the output form PSFfit.py for the image n9u608knq_cal.fits. In the figure the grid lines show the separation of the

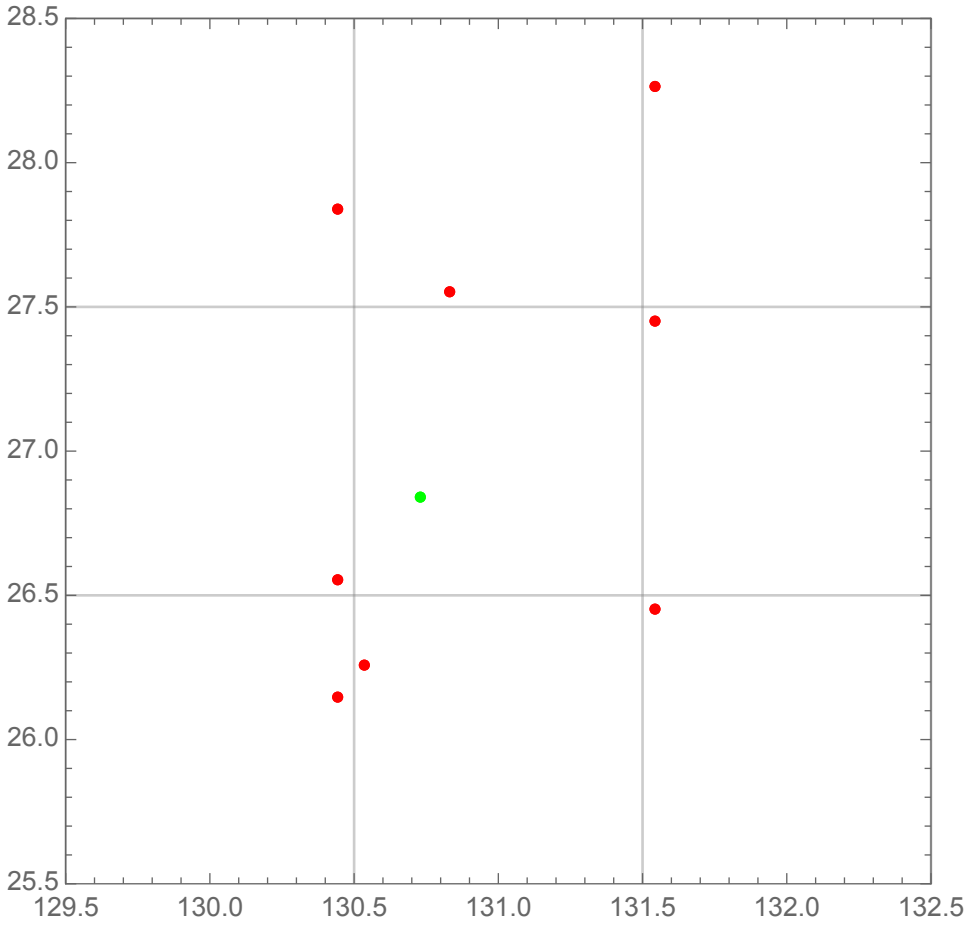


**Figure 4.4** Shows the 9 different binary fits that are included in the output of Fig. 4.1. In this figure the point in the central pixel represents location of the secondary PSF, and is indicated in green.The points in red are not real solutions.

nine pixels centered on the user given central pixel that contains the primary BD. Notice how each of the proposed secondary PSF locations and can be associated with one of the sections of the binary fit output shown Fig. 4.1.

In the output the coordinates for both the primary and the secondary PSFs are given as an x and y position on the CCD, as well positions of the PSF within their pixel and the flux associated with each BD. In the second to last line of the output second also give the position angle and separation between the BDs. The separation is given in units of arcseconds. While each pixel is 0.043”. This

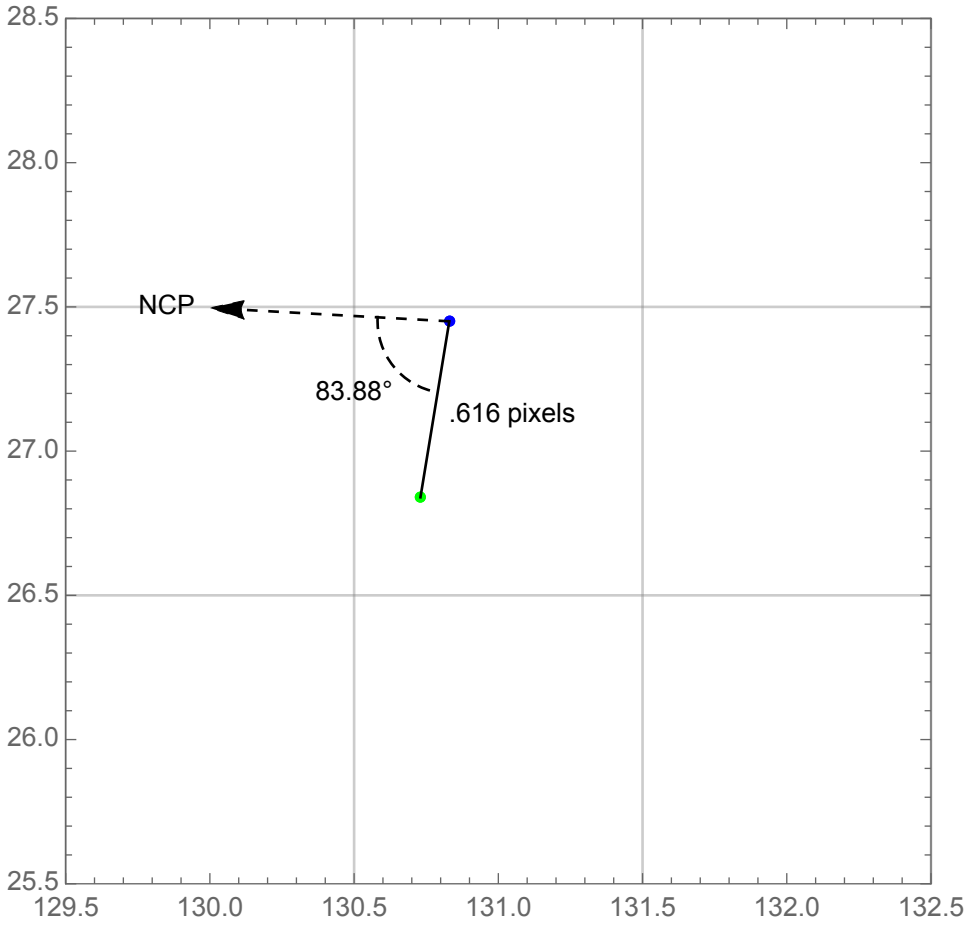


**Figure 4.5** The primary PSF's center is in blue and the secondary's PSF is in green. Also indicated are the separation and the position angle for the best binary fit from the output of PSFfit.py for the image n9u608knq_cal.fits.

means that for the output in Fig. 4.3 the centers of the two PSFs are separated by about .616 pixels on the CCD. The position angle is also included, in the case of binaries the position angle is the angular offset of the secondary star from the primary relative to the north celestial pole in degrees measured in the direction of increasing right ascension (RA). In the case of the output in Fig. 4.3 the position angle is 83.88.he center of both of these PSFs are shown in Fig. 4.5. The primary is in blue, and the secondary in green with separation and position angle indicated.

When confirming the binary fit for the various images, it is important to note the fluxes, the

separation and the position angle so that you can show that that you consistently find the same binary fit for the BD candidate.

The error included in Fig. 4.3's output for each fit is calculated as a sum of residual errors as explained in section 2.2 of this paper. The unit of flux used is a Jansky, and so the sum of the residual is .62 Jy for the nine pixels.

Here we find it important to note again that the error included in the outputs for the single and binary fits are calculated differently. Because the errors are not the same, we can not directly compare the results of the fits to make a determination about binarity.

Due to the limitations of the code which make it difficult to compare errors. We will rely solely on the metric of the relatively even division of the flux in the system to determine binarity in this paper, using the separation to ensure that the flux division is not due to the position of a PSF at a detector boundary. In this paper we are using as necessary criteria that the flux is more evenly divided that 40-60 and that there is a separation of at least 0.5 pixels. This technique is biased towards finding equal mass binaries, and a summary of these probably binaries is included in Table 4.1and they are compared with the known binaries from the same data set from the spectroscopic determinations of Leanne Farnbach.

## 4.2 Probable Binaries

As explained in section 4.1 we will determine binarity of probable binary systems based on the distribution of flux. Our data set includes 21 objects that fit the criteria of having at least four images from the NICMOS 1 or 2 on HST that were considered to be "good " data (the BD binary candidate was in the image, not near an edge, quadrant boundary or area of bad pixels).

While running this data through PSF.py, I kept a spreadsheet for each object, Table A.1 is included an abbreviated version for the BD-BD candidate 2MASSWJ0036159+182110, which has

been previously discussed, using images taken with NICMOS 1. Included in the table are the flux division and the separation in pixels for binary determination. The separation is originally given in arc seconds and because there are 0.043 ” per pixel we can make a unit conversion.

In the output from each star we are verifying the flux distribution, the separation and how many good fits there are. probable binarity is based on the two criteria for each image:

**1.The flux is split more evenly that 40-60**

**2. The separation is greater than of equal to .5 pixels**

This is a very rudimentary way of declaring a probably binary because I determined the acceptable flux split, and the separation arbitrarily.

If more than 50% of the images suggested that the candidate was a binary, I determined that the BD was a probable binary as recorded in Table 4.1, and also included the average separation and flux of the primary for all of the fits that did suggest binarity.

| BD binary candidate | Average Separation of PSFs | Average Flux of Primary | Binary? | Summary Table |
|---|---|---|---|---|
| 2MASSWJ0036159+182110 | 0.0240 (0.558) | 53.95% | Yes | A.1 |
| 2MASSIJ0103320+193536 | 0.0472 (1.01) | 53.05% | Yes | A.2 |
| 2MASSWJ031059+1648 | 0.0363 (0.844) | 57.17% | Yes | A.3 |
| 2MASS J03480772-6022270 | | | No | A.4 |
| 2MASS 1439+19 | | | No | A.5 |
| 2MASS 0445-3048 | | | No | A.6 |
| 2MASS J05591914-1404488 | | | No | A.7 |
| 2MASS J06244595-4521548 | | | No | A.8 |
| 2MASSI J0825196 + 211552 | | | No | A.9 |
| 2MASS J08354256-0819237 | | | No | A.10 |
| 2MASS J10224821+5825453 | | | No | A.11 |
| 2MASS 1108+68 | | | No | A.12 |
| 2MASS J12171110-0311131 | | | No | A.13 |
| 2MASS J12464678+4027150 | | | No | A.14 |
| 2MASS 1300+1912 | | | No | A.15 |
| 2MASS J1507476 -162738 | | | No | A.16 |
| 2MASS J16241436+0029158 | | | No | A.17 |
| 2MASS 1658+70 | | | No | A.18 |
| 2MASSI J1721039 + 334415 | | | No | A.19 |
| 2MASS J17503293+1759042 | | | No | A.20 |
| 2MASS J17534518-6559559 | | | No | A.21 |

**Table 4.1** A list of the 21 BD-BD studied and if they are probable binaries based on the even sharing of flux.

# Chapter 5

# Conclusions

## 5.1 Limitations

The PSF fitting script is still in the developmental stage, and there are improvements that can still be made. The primary limitations of the script include the aforementioned error comparison, the speed of the script and the method of pixel selection.

While different computers have different specs and different computing power, on the university computer I used to run the images, it takes about nine minutes per image. It is also only possible to run about eight images at the same time or using the command nice to distribute CPU time.

It might also be possible to speed to program by allowing it to read in the user input from a spreadsheet rather than prompt the user. This would allow the program to also run continuously which makes the time needed to run an image less prohibitive.

An additional and more detrimental limitation is that the program is only designed to work at very small separations. The program also assumes that the primary is in the given central pixel and only searches of the secondary in the three by three grid of pixels about the central pixel. For this program to be more broadly applicable, it needs a larger search radius and smarter pixel selection

such that the primary does not have to be in the central pixel. Spectroscopicly, for example, Leanne Farnbach indicated that 2MASS0559 was a binary but our investigations suggest that it isn't based on our criteria, if we allowed for a less-even division of fluxes the candidate would have appeared to be a BD - BD binary.

Along the same vein of thought it is also problematic to assume that a BD much be in a particular pixel because the program has difficulty determining position is an object is located near of at the boundary between pixels, and this has appeared as an interesting result in the fits of objects such as 2MASS0103.

The importance of our separation criteria is underscored by objects such as 2MASS 1439 which have images that show a good flux distribution but this can be explained by the BD being at a pixel boundary.

The most easily resolved of all of these problems should be that the output gives the error differently for single and binary fits. This makes it is difficult to compare and determine which is the better fit. For a single fit, the output gives a minchi error, and for the secondary fit, it gives the sum of all of the residuals, as discussed in section 2.2. MattâĂŹs documentation fails to explain how to compare the single and binary fits using these errors. The only comment about error comparison made by Matt in his 2017 thesis is included below.

"To determine whether these results indicate the detection of a binary or not, we want to see that the binary fit is significantly better than the single fit. If the best binary fit is more than 3ÏČ better than the best single fit, as measured by âĹĘÏĞ2, we consider the system to be a probable binary." (Matt 2017)

However, the program output fails to include the ÏČ value or a residual error for the single fit.

## 5.2 Conclusions

The original objective of using PSFfit.py to study these images was to test the code to see if it is accurate in determining which brown dwarf candidates are actually binaries. With this goal, I went through each image finding the coordinates of the candidate and then calibrating the images and creating the model PSFs. I discovered that the errors of the single and binary fits were different types of error and therefore incompatible with each other. While there should be a simple fix, I leave the solution to a better programmer than I. Instead, I devised a simple metric based on the output to say if a star is a probable binary and leave further modification and testing of the code as a continuation of this project. Even though the program cannot decisively identify BD binaries, there are five probable candidates that I recommend to further study.

The program also was limited in which pixels it could search in for the best PSF fit in both the single and binary fits

This program will be useful only after its errors are fixed. It has shone the capacity to resolve some BD-BD binaries that would otherwise be too close to differentiate. Once the code has better fit comparisons, (and other improvements), this program will be ready to resolve BD binaries in mass with the goal of elucidating the multiplicity fraction of brown dwarfs.

# Appendix A

# Summary Tables for BD-BD Candidates

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n9u608k4q_cal.fits | No | | | | | | | | | | | |
| n9u608keq_cal.fits | No | | | | | | | | | | | |
| n9u608knq_cal.fits | Yes | 131 | 27 | L3F090Mo1 | 54.98% | 45.02% | 131 | 27 | 1 | 0.0265 (0.616) | Yes | |
| n9u608kwq_cal.fits | Yes | 61 | 27 | L3F090Mo11 | 50.02% | 49.98% | 61 | 27 | 2 | 0.0254 (0.591) | Yes | |
| n9u608k7q_cal.fits | No | | | | | | | | | | | |
| n9u608khq_cal.fits | No | | | | | | | | | | | |
| n9u608kqq_cal.fits | Yes | 131 | 27 | L3F110Mo1 | 52.08% | 47.92% | 131 | 26 | 2 | 0.0219 (0.509) | Yes | |
| n9u608kzq_cal.fits | Yes | 61 | 27 | L3F110Mo11 | 43.63% | 56.37% | 61 | 27 | 3 | 0.0290 (0.674) | Yes | |
| n8yj06deq_cal.fits | Yes | 222 | 47 | L3F110Wo2 | 57.95% | 42.05% | 222 | 47 | 1 | 0.0061 (0.148) | No | Not enough separation |
| n8yj06dgq_cal.fits | No | | | | | | | | | | | |
| n9u608kaq_cal.fits | No | | | | | | | | | | | |
| n9u608kjq_cal.fits | No | | | | | | | | | | | |
| n9u608ksq_cal.fits | Yes | 131 | 27 | L3F110Wo1 | 53.69% | 46.31% | 131 | 28 | 1 | 0.0219 (0.509) | Yes | |
| n9u6081lq_cal.fits | Yes | 61 | 27 | L3F110Wo11 | 51.91% | 48.09% | 62 | 27 | 4 | 0.0183 (0.426) | No | Not enough separation |
| n9u608k6q_cal.fits | No | | | | | | | | | | | |
| n9u608kfq_cal.fits | No | | | | | | | | | | | |
| n9u608kpq_cal.fits | Yes | 131 | 27 | L3F145Wo1 | 50.09% | 49.91% | 131 | 28 | 2 | 0.0231 (0.537) | Yes | |
| n9u608kyq_cal.fits | Yes | 61 | 27 | L3F145Wo11 | 41.91% | 58.09% | 62 | 27 | 3 | 0.0215 (0.500) | Yes | |
| n9u608k9q_cal.fits | No | | | | | | | | | | | |
| n9u608kiq_cal.fits | No | | | | | | | | | | | |
| n9u608krq_cal.fits | Yes | 131 | 27 | L3F160Wo1 | | | | | | | No | Flux distribution not even |
| n9u6081Oq_cal.fits | Yes | 61 | 27 | L3F160Wo11 | | | | | | | No | Flux distribution not even |
| n9u608k3q_cal.fits | No | | | | | | | | | | | |
| n9u608kdq_cal.fits | No | | | | | | | | | | | |
| n9u608kmq_cal.fits | Yes | 131 | 27 | L3F164No1 | 55.68% | 44.32% | 131 | 28 | 1 | 0.0251 (.584) | Yes | |
| n9u608kvq_cal.fits | Yes | 61 | 27 | L3F164No11 | | | 62 | 27 | | | No | Flux distribution not even |
| n8yj06ddq_cal.fits | Yes | 222 | 47 | L3F170Mo2 | 55.90% | 44.10% | 222 | 46 | 1 | 0.0192 (.447) | No | Not enough separation |
| n8yj06dfq_cal.fits | No | | | | | | | | | | | |
| n9u608k2q_cal.fits | No | | | | | | | | | | | |
| n9u608kbq_cal.fits | No | | | | | | | | | | | |
| n9u608kkq_cal.fits | Yes | 131 | 27 | L3F190No1 | | | | | | | No | |
| n9u608ktq_cal.fits | Yes | 61 | 27 | L3F190No11 | 54.58% | 54.42% | 62 | 27 | 5 | 0.0215 (0.500) | Yes | Flux distribution not even |

**Table A.1** Summary of 2MASSWJ0036159+182110 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8xw03trq_cal.fits | Yes | 140 | 39 | L6F108No1 | 53.18% | 46.82% | 141 | 39 | 1 | 0.0560 (1.302) | Yes | Secondary on border of (141,38) & (141,39) |
| n8xw03u0q_cal.fits | Yes | 140 | 155 | L6F108Nm6 | 55.29% | 44.71% | 141 | 155 | 2 | 0.0408 (0.949) | Yes | |
| n8xw03u9q_cal.fits | Yes | 140 | 39 | L6F108No1 | 50.32% | 49.68% | 140 | 39 | 4 | 0.0387 (0.900) | Yes | |
| n8xw03ujq_cal.fits | Yes | 140 | 155 | L6F108Nm6 | 56.09% | 43.91% | 140 | 155 | 1 | 0.0505 (1.174) | Yes | Multiple "good" fits, not on edges |
| n8xw03tsq_cal.fits | Yes | 140 | 39 | L6F113No1 | 57.45% | 42.55% | 140 | 39 | 1 | 0.0482 (1.121) | Yes | |
| n8xw03u1q_cal.fits | Yes | 140 | 155 | L6F113Nm6 | 47.13% | 52.87% | 140 | 156 | 2 | 0.0398 (0.926) | Yes | |
| n8xw03uaq_cal.fits | Yes | 140 | 39 | L6F113No1 | 50.96% | 49.04% | 139 | 39 | 1 | 0.0534 (1.242) | Yes | |
| n8xw03ukq_cal.fits | Yes | 140 | 155 | L6F113Nm6 | 53.99% | 46.01% | 139 | 155 | 1 | 0.0504 (1.172) | Yes | |

**Table A.2** Summary of 2MASSIJ0103320+193536 with PSFfit.py output.

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8xw04upq_cal.fits | Yes | 166 | 77 | L9F108Nm2 | 56.77% | 43.23% | 166 | 78 | 1 | 0.0384 (0.893) | Yes | |
| n8xw04v9q_cal.fits | Yes | 167 | 194 | L9F108Nm5 | 58.23% | 41.77% | 167 | 194 | 4 | 0.0085 (0.198) | No | Not enough separation |
| n8xw04vbq_cal.fits | Yes | 166 | 77 | L9F108Nm2 | | | | | | | No | Flux distribution not even |
| n8xw04vlq_cal.fits | Yes | 167 | 194 | L9F108Nm5 | 44.96% | 55.04% | 166 | 194 | 1 | 0.0230 (0.535) | Yes | |
| n8xw04uqq_cal.fits | Yes | 166 | 77 | L9F113Nm2 | | | | | | | No | Flux distribution not even |
| n8xw04vaq_cal.fits | Yes | 166 | 194 | L9F113Nm5 | 40.80% | 59.20% | 167 | 194 | 2 | 0.0273 (0.635) | Yes | |
| n8xw04vcq_cal.fits | Yes | 166 | 77 | L9F113Nm2 | 41.60% | 58.40% | 166 | 78 | 1 | 0.0573 (1.333) | Yes | |
| n8xw04vmq_cal.fits | Yes | 166 | 189 | L9F113Nm5 | 55.36% | 44.64% | 166 | 190 | 2 | 0.0355 (0.826) | Yes | |

**Table A.3** Summary of 2MASSWJ031059+1648 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8nh02mzq_cal.fits | Yes | 125 | 84 | T8F090Mm12 | | | | | | | No | Flux distribution not even |
| n8nh02kvq_cal.fits | Yes | 120 | 66 | T8F110Wm12 | | | | | | | No | Flux distribution not even |
| n8nh02kwq_cal.fits | Yes | 120 | 66 | T8F110Wm12 | | | | | | | No | Flux distribution not even |
| n8nh02kyq_cal.fits | Yes | 150 | 66 | T8F110Wm1 | | | | | | | No | Flux distribution not even |
| n8nh02l0q_cal.fits | Yes | 150 | 66 | T8F110Wm1 | | | | | | | No | Flux distribution not even |
| n8nh02mfq_cal.fits | Yes | 126 | 69 | T8F170Mm12 | | | | | | | No | Flux distribution not even |
| n8nh02mhq_cal.fits | Yes | 156 | 69 | T8F170Mm1 | | | | | | | No | Flux distribution not even |
| n8nh02mmq_cal.fits | Yes | 156 | 99 | T8F170Mm2 | | | | | | | No | Flux distribution not even |
| n8nh02mnq_cal.fits | Yes | 125 | 99 | T8F170Mm12 | | | | | | | No | Flux distribution not even |
| n8nh02moq_cal.fits | Yes | 95 | 99 | T8F170Mm11 | | | | | | | No | Flux distribution not even |
| n8nh02mxq_cal.fits | Yes | 95 | 68 | T8F170Mm12 | | | | | | | No | Flux distribution not even |

**Table A.4** Summary of 2MASS J03480772-6022270 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj12b3q_cal.fits | Yes | 151 | 94 | L6F110Wm2 | | | | | | | No | Flux distribution not even |
| n8yj12b6q_cal.fits | Yes | 197 | 94 | L6F110Wm3 | | | | | | | No | Flux distribution not even |
| n8yj12b2q_cal.fits | Yes | 151 | 94 | L6F170Mm2 | 150 | 94 | 48.05% | 51.95% | 1 | 0.0155 (0.360) | No | Not enough separation |
| n8yj12b5q_cal.fits | Yes | 197 | 94 | L6F170Mm3 | | | | | | | No | Flux distribution not even |

**Table A.5** Summary of U10312 or 2MASS 1439+19 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj47ueq_cal.fits | Yes | 208 | 139 | L3F110Wo5 | | | | | | | No | Flux distribution not even |
| n8yj47ugq_cal.fits | Yes | 254 | 139 | L3F110Wo4 | | | | | | | No | Flux distribution not even |
| n8yj47udq_cal.fits | Yes | 208 | 139 | L3F170Mo5 | | | | | | | No | Flux distribution not even |
| n8yj47ufq_cal.fits | Yes | 254 | 139 | L3F170Mo4 | | | | | | | No | Flux distribution not even |

**Table A.6** Summary of U10329 or 2M0445-3048 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8qt04a9q_cal.fits | Yes | 134 | 77 | T5F090Mm1 | 52.61% | 47.39% | 134 | 78 | 1 | 0.0343 (.780) | Yes | |
| n8qt04abq_cal.fits | Yes | 134 | 77 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04aeq_cal.fits | Yes | 145 | 77 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04afq_cal.fits | Yes | 145 | 77 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04ajq_cal.fits | Yes | 145 | 89 | T5F090Mm1 | 46.31% | 53.69% | 145 | 89 | 1 | 0.0350 (0.814) | Yes | Secondary on border of (145,88) & (141,89) |
| n8qt04akq_cal.fits | Yes | 145 | 89 | T5F090Mm1 | 54.09% | 45.91% | 145 | 89 | 3 | 0.0347 (0.807) | Yes | |
| n8qt04auq_cal.fits | Yes | 134 | 78 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04awq_cal.fits | Yes | 134 | 78 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04azq_cal.fits | Yes | 145 | 78 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04beq_cal.fits | Yes | 134 | 78 | T5F090Mm1 | 52.65% | 47.35% | 134 | 77 | 2 | 0.0389 (0.905) | Yes | |
| n8qt04bfq_cal.fits | Yes | 134 | 78 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt04bjq_cal.fits | Yes | 145 | 78 | T5F090Mm1 | 49.69% | 50.31% | 145 | 77 | 1 | 0.0388 (0.902) | Yes | Secondary on border of (145,77) & (145,78) |
| n8qt04bkq_cal.fits | Yes | 145 | 78 | T5F090Mm1 | 53.68% | 46.32% | 145 | 77 | 2 | 0.0345 (0.802) | Yes | Secondary on border of (145,77) & (145,78) |
| n8qt04bnq_cal.fits | Yes | 145 | 90 | T5F090Mm1 | 48.69% | 51.31% | 145 | 89 | 1 | 0.0343 (0.798) | Yes | |
| n8qt04zvq_cal.fits | Yes | 145 | 77 | T5F090Mm1 | 59.37% | 40.63% | 145 | 78 | 2 | 0.0354 (0.823) | Yes | |
| n8qt03lrq_cal.fits | Yes | 130 | 98 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03lvq_cal.fits | Yes | 141 | 98 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03m1q_cal.fits | Yes | 141 | 109 | T5F090Mi2 | | | | | | | No | Flux distribution not even |
| n8qt03mmq_cal.fits | Yes | 141 | 97 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03n5q_cal.fits | Yes | 130 | 98 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03n9q_cal.fits | Yes | 141 | 98 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03neq_cal.fits | Yes | 141 | 109 | T5F090Mi2 | | | | | | | No | Flux distribution not even |
| n8qt03nqq_cal.fits | Yes | 130 | 97 | T5F090Mm1 | 48.96% | 51.04% | 129 | 97 | 2 | 0.0271 (0.630) | Yes | |
| n8qt03obq_cal.fits | Yes | 141 | 97 | T5F090Mm1 | | | | | | | No | Flux distribution not even |
| n8qt03ogq_cal.fits | Yes | 141 | 109 | T5F090Mi2 | | | | | | | No | Flux distribution not even |
| n8qt03orq_cal.fits | Yes | 130 | 97 | 5F090Mm1 | 40.58% | 59.42% | 129 | 97 | 1 | 0.0305 (0.709) | Yes | |
| n8qt03owq_cal.fits | Yes | 141 | 97 | T5F090Mm1 | 47.35% | 52.65% | 141 | 97 | 1 | 0.0218 (0.507) | Yes | |

**Table A.7** Summary of 2MASS J05591914-1404488 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj36k5q_cal.fits | Yes | 182 | 89 | L5F110Wm2 | | | | | | | No | Flux distribution not even |
| n8yj36k7q_cal.fits | Yes | 228 | 89 | L5F110Wo3 | | | | | | | No | Flux distribution not even |
| n8yj36k4q_cal.fits | Yes | 182 | 89 | L5F170Mm2 | 54.01% | 45.99% | 182 | 88 | 1 | 0.0136 (0.316) | No | Not enough separation |
| n8yj36k6q_cal.fits | Yes | 228 | 89 | L5F110Wo3 | | | | | | | No | Flux distribution not even |

**Table A.8** Summary of U20244 or 2MASS J06244595-4521548 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8xw08yfq_cal.fits | Yes | 189 | 67 | L8F108No2 | | | | | | | No | Flux distribution not even |
| n8xw08yoq_cal.fits | Yes | 189 | 183 | L8F108Nm5 | 54.40% | 45.60% | 189 | 184 | 2 | 0.0409 (0.951) | Yes | |
| n8xw08yxq_cal.fits | Yes | 189 | 67 | L8F108No2 | 59.32% | 40.68% | 188 | 67 | 1 | 0.0214 (0.497) | No | Not enough separation |
| n8xw08z7q_cal.fits | Yes | 188 | 183 | L8F108Nm5 | 44.47% | 55.53% | 189 | 184 | 1 | 0.0288 (0.670) | Yes | |
| n8yj11bgq_cal.fits | Yes | 168 | 167 | L8F110Wm5 | 49.69% | 50.31% | 167 | 167 | 2 | 0.0232 (0.540) | Yes | |
| n8yj11biq_cal.fits | Yes | 214 | 167 | L8F110Wo4 | | | | | | | No | Flux distribution not even |
| n8xw08ygq_cal.fits | Yes | 189 | 67 | L8F113No2 | | | | | | | No | Flux distribution not even |
| n8xw08ypq_cal.fits | Yes | 189 | 184 | L8F113Nm5 | | | | | | | No | Flux distribution not even |
| n8xw08yyq_cal.fits | Yes | 188 | 67 | L8F113No2 | 44.96% | 55.04% | 188 | 67 | 1 | 0.0043 (0.010) | No | Not enough separation |
| n8xw08z8q_cal.fits | Yes | 188 | 183 | L8F113Nm5 | 42.27% | 57.73% | 189 | 184 | 1 | 0.0242 (0.563) | Yes | |
| n8yj11bfq_cal.fits | Yes | 168 | 167 | L8F170Mm5 | | | | | | | No | Flux distribution not even |
| n8yj11bhq_cal.fits | Yes | 213 | 167 | L8F170Mo4 | 48.87% | 51.13% | 214 | 168 | 2 | 0.0579 (1.347) | Yes | also a good fit for (214, 166) |

**Table A.9** Summary of U10721 or 2MASSIJ082519+2115 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8xw09ffq_cal.fits | Yes | 190 | 120 | L5F108Nm3 | | | | | | | No | Flux distribution not even |
| n8xw09fhq_cal.fits | Yes | 190 | 236 | L5F108No6 | 40.15% | 59.85% | 290 | 134 | 1 | 0.0195 (0.453) | No | Not enough separation |
| n8xw09fqq_cal.fits | Yes | 190 | 120 | L5F108Nm3 | | | | | | | No | Flux distribution not even |
| n8xw09ftq_cal.fits | Yes | 190 | 236 | L5F108No6 | | | | | | | No | Flux distribution not even |
| n8yj03tgq_cal.fits | Yes | 211 | 68 | L5F110Wo2 | (52.06% | 47.94% | 211 | 68 | 2 | 0.0136 (0.316) | No | Not enough separation |
| n8yj03tiq_cal.fits | No | 256 | 68 | L5F110Wo3 | | | | | | | | Not well in image |
| n8xw09fgq_cal.fits | Yes | 190 | 120 | L5F113Nm3 | | | | | | | No | Flux distribution not even |
| n8xw09fiq_cal.fits | Yes | 189 | 236 | L5F113No6 | | | | | | | No | Flux distribution not even |
| n8xw09frq_cal.fits | Yes | 190 | 120 | L5F113Nm3 | | | | | | | No | Flux distribution not even |
| n8xw09fuq_cal.fits | Yes | 190 | 236 | L5F113No6 | | | | | | | No | Flux distribution not even |
| n8yj03tfq_cal.fits | Yes | 211 | 68 | L5F170Mo2 | | | | | | | No | Flux distribution not even |
| n8yj03thq_cal.fits | No | 256 | 68 | L5F170Mo3 | | | | | | | | Not well in image |

**Table A.10** Summary of U10742 or 2MASSIJ0835-0819 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj01rrq_cal.fits | No | 254 | 107 | L1F110Wo3 | | | | | | | | Not well in image |
| n8yj01rtq_cal.fits | No | | | | | | | | | | | Not in image |
| n9nk11loq_cal.fits | Yes | 103 | 92 | L1F110Wm11 | | | | | | | No | Flux distribution not even |
| n9nk11lqq_cal.fits | Yes | 149 | 93 | L1F110Wm1 | | | | | | | No | Flux distribution not even |
| n8yj01rqq_cal.fits | No | 254 | 107 | L1F170Mo3 | | | | | | | | Not well in image |
| n8yj01rsq_cal.fits | No | | | | | | | | | | | unclear, may be on the edge |
| n9nk11lnq_cal.fits | Yes | 103 | 92 | L1F170Mm11 | | | | | | | No | Flux distribution not even |
| n9nk11lpq_cal.fits | Yes | 149 | 92 | L1F170Mm1 | | | | | | | No | Flux distribution not even |

**Table A.11** Summary of U20373 or 2MASS J10224821+5825453 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj56jqq_cal.fits | Yes | 135 | 106 | L1F110Wi1 | 50.85% | 49.15% | 135 | 106 | 3 | 0.0096 (.223) | No | Not enough separation |
| n8yj56jsq_cal.fits | Yes | 181 | 106 | L1F110Wm3 | | | | | | | No | Flux distribution not even |
| n8yj56jpq_cal.fits | Yes | 135 | 106 | L1F170Mi1 | | | | | | | No | Flux distribution not even |
| n8yj56jrq_cal.fits | Yes | 181 | 106 | L1F170Mm3 | | | | | | | No | Flux distribution not even |

**Table A.12** Summary of U10960 or 2MASS 1108+68 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8nh07gwq_cal.fits | Yes | 158 | 90 | T8F110Wm2 | | | | | | | No | Flux distribution not even |
| n8nh07gxq_cal.fits | Yes | 188 | 90 | T8F110Wm3 | 53.62% | 46.38% | 187 | 90 | 2 | 0.0220 (0.512) | Yes | |
| n8nh07gyq_cal.fits | Yes | 188 | 120 | T8F110Wm3 | | | | | | | No | Flux distribution not even |
| n8nh07gzq_cal.fits | Yes | 172 | 81 | T8F170Mm2 | | | | | | | No | Flux distribution not even |
| n8nh07h1q_cal.fits | Yes | 202 | 81 | T8F170Mo2 | 57.54% | 42.46% | 202 | 82 | 1 | 0.0258 (0.600) | Yes | |
| n8nh07h2q_cal.fits | Yes | 202 | 112 | T8F170Mm3 | | | | | | | No | Flux distribution not even |
| n8nh07h3q_cal.fits | Yes | 172 | 112 | T8F170Mm3 | | | | | | | No | Flux distribution not even |
| n8nh07h5q_cal.fits | Yes | 142 | 112 | T8F170Mi2 | | | | | | | No | Flux distribution not even |
| n8nh07h6q_cal.fits | No | | | | | | | | | | | Not in image |

**Table A.13** Summary of 2MASS-121711-031113 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8xw10daq_cal.fits | Yes | 185 | 74 | L3F108Nm2 | | | | | | | No | Flux distribution not even |
| n8xw10dcq_cal.fits[1] | Yes | 186 | 190 | L3F108Nm5 | 54.26% | 45.74% | 186 | 189 | 2 | 0.0488 (1.135) | Yes | |
| n8xw10dlq_cal.fits | Yes | 185 | 74 | L3F108Nm2 | 58.95% | 41.05% | 185 | 74 | 1 | 0.0096 (0.223) | No | Not enough separation |
| n8xw10doq_cal.fits | Yes | 186 | 190 | L3F108Nm5 | | | | | | | No | Flux distribution not even |
| n8xw10dbq_cal.fits | Yes | 185 | 74 | L3F113Nm2 | 47.93% | 52.07% | 185 | 74 | 1 | 0.0232 (0.540) | Yes | |
| n8xw10ddq_cal.fits | Yes | 185 | 190 | L3F113Nm5 | | | | | | | No | Flux distribution not even |
| n8xw10dmq_cal.fits | Yes | 185 | 74 | L3F113Nm2 | | | | | | | No | Flux distribution not even |
| n8xw10dpq_cal.fits | Yes | 186 | 190 | L3F113Nm5 | | | | | | | No | Flux distribution not even |

**Table A.14** Summary of 2MASS J12464678+4027150 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj28uvq_cal.fits | No | | | | | | | | | | | Not in image |
| n8yj28v1q_cal.fits | No | | | | | | | | | | | Not in image |
| n9nk17ugq_cal.fits | Yes | 177 | 84 | L1F110Wm2 | | | | | | | No | Flux distribution not even |
| n9nk17v3q_cal.fits | Yes | 223 | 84 | L1F110Wo3 | | | | | | | No | Flux distribution not even |
| n8yj28uuq_cal.fits | No | | | | | | | | | | | Not in image |
| n8yj28v0q_cal.fits | No | | | | | | | | | | | Not in image |
| n9nk17ufq_cal.fits | Yes | 177 | 84 | L1F170Mm2 | | | | | | | No | Flux distribution not even |
| n9nk17v2q_cal.fits | Yes | 223 | 84 | L1F170Mo3 | | | | | | | No | Flux distribution not even |

**Table A.15** Summary of U11115 or 2MASS 1300+1912 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj02wyq_cal.fits | Yes | 34 | 42 | L5F110Wo11 | | | | | | | No | Flux distribution not even |
| n8yj02x0q_cal.fits | Yes | 80 | 42 | L5F110Wo12 | | | | | | | No | Flux distribution not even |
| n8yj02wxq_cal.fits | Yes | 34 | 42 | L5F170Mo11 | | | | | | | No | Flux distribution not even |
| n8yj02wzq_cal.fits | Yes | 80 | 42 | L5F170Mo12 | | | | | | | No | Flux distribution not even |

**Table A.16** Summary of U11296 or 2MASSW J1507476 -162738 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8nh21qoq_cal.fits | No | | | | | | | | | | | Not in image |
| n8nh21qeq_cal.fits | Yes | 145 | 104 | T6F110Wm2 | | | | | | | No | Flux distribution not even |
| n8nh21qfq_cal.fits | Yes | 176 | 104 | T6F110Wm3 | | | | | | | No | Flux distribution not even |
| n8nh21qgq_cal.fits | Yes | 176 | 134 | T6F110Wm4 | | | | | | | No | Flux distribution not even |
| n8nh21qhq_cal.fits | Yes | 160 | 95 | T6F170Mm2 | 49.08 % | 50.92% | 160 | 95 | 1 | 0.0096 (0.223) | No | Not enough separation. |
| n8nh21qjq_cal.fits | Yes | 190 | 95 | T6F170Mm3 | | | | | | | No | Flux distribution not even |
| n8nh21qkq_cal.fits | Yes | 190 | 126 | T6F170Mm3 | | | | | | | No | Flux distribution not even |
| n8nh21qlq_cal.fits | Yes | 160 | 126 | T6F170Mm3 | | | | | | | No | Flux distribution not even |
| n8nh21qnq_cal.fits | Yes | 130 | 126 | T6F170Mi2 | | | | | | | No | Flux distribution not even |

**Table A.17** Summary of SDSS-162414+002916 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj62elq_cal.fits | Yes | 144 | 94 | L1F110Wm1 | | | | | | | No | Flux distribution not even |
| n8yj62enq_cal.fits | Yes | 190 | 94 | L1F110Wm3 | | | | | | | No | Flux distribution not even |
| n8yj62ekq_cal.fits | Yes | 144 | 93 | L1F170Mm1 | | | | | | | No | Flux distribution not even |
| n8yj62emq_cal.fits | Yes | 190 | 94 | L1F170Mm3 | 44.99% | 55.01% | 190 | 94 | 3 | 0.0096 (.223) | No | Not enough separation |

**Table A.18** Summary of U11668 or 2MASS 1658+70 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj34a9q_cal.fits | No | | | | | | | | | | | Not in image |
| n8yj34abq_cal.fits | No | | | | | | | | | | | Not in image |
| n9nk22lmq_cal.fits | Two in frame | | | | | | | | | | | Neither Work |
| n9nk22loq_cal.fits | Yes | 224 | 84 | L3F110Wo3 | | | | | | | No | Flux distribution not even |
| n8yj34a8q_cal.fits | No | | | | | | | | | | | Not in image |
| n8yj34aaq_cal.fits | No | | | | | | | | | | | Not in image |
| n9nk22llq_cal.fits | Two in frame | | | | | | | | | | | Neither Work |
| n9nk22lnq_cal.fits | Yes | 224 | 84 | L3F170Mo3 | | | | | | | No | Flux distribution not even |

**Table A.19** Summary of U11694 or 2MASSI J1721039 + 334415 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8nh22q9q_cal.fits | No | | | | | | | | | | | Not in image |
| n8nh22otq_cal.fits | Yes | 147 | 65 | T2F110Wm1 | 47.04% | 52.96% | 147 | 66 | 1 | 0.0218 (0.507) | Yes | |
| n8nh22ovq_cal.fits | Yes | 177 | 65 | T2F110Wm2 | | | | | | | No | Flux distribution not even |
| n8nh22poq_cal.fits | Yes | 177 | 96 | T2F110Wm2 | | | | | | | No | Flux distribution not even |
| n8nh22ppq_cal.fits | Yes | 161 | 57 | T2F170Mm1 | 59.95% | 40.05% | 161 | 57 | 1 | 0.0086 (0.200) | No | Not enough separation. |
| n8nh22prq_cal.fits | Yes | 191 | 57 | T2F170Mo2 | 40.97% | 59.03% | 191 | 57 | 1 | 0.0178 (0.414) | No | not enough separation. |
| n8nh22psq_cal.fits | Yes | 191 | 88 | T2F170Mm2 | | | | | | | No | Flux distribution not even |
| n8nh22puq_cal.fits | Yes | 161 | 88 | T2F170Mm2 | | | | | | | No | Flux distribution not even |
| n8nh22q8q_cal.fits | Yes | 131 | 87 | T2F170Mm1 | | | | | | | No | Flux distribution not even |

**Table A.20** Summary of SDSS-175032+175904 with PSFfit.py output

| Image | Object? | X-Coord. | Y-Coord | TinyTim | Primary Flux% | Secondary Flux% | Binary-X | Binary-Y | Fit Num. | Separation (in pixels) | Binary? | Extra Info |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n8yj41biq_cal.fits | Yes | 169 | 93 | L3F110Wm2 | | | | | | | No | Flux distribution not even |
| n8yj41bkq_cal.fits | Yes | 216 | 93 | L3F110Wo3 | | | | | | | No | Flux distribution not even |
| n8yj41bhq_cal.fits | Yes | 170 | 93 | L3F170Mm2 | | | | | | | No | Flux distribution not even |
| n8yj41bjq_cal.fits | Yes | 216 | 93 | L3F170Mo3 | | | | | | | No | Flux distribution not even |

**Table A.21** Summary of U20760 or 2MASS J17534518âĹŠ6559559 with PSFfit.py output

# Bibliography

Chabrier, G. 2002, The Astrophysical Journal, 567, 304

—. 2003, Publications of the Astronomical Society of the Pacific, 115, 763

Close, L. M., Siegler, N., Freed, M., & Biller, B. 2003, ApJ, 587, 407

Konopacky, Q. M. 2013, Memorie della Societa Astronomica Italiana, 84, 1005

Krist, J., & Hook, R. 2004, 6

Kumar, S. S. 1962, AJ, 67, 579

Matt, K. 2017, BYU Press

Miller, G. E., & Scalo, J. M. 1979, AJ, 41, 513

Salpeter, & E., E. 1955, ApJ, 121, 161

Thatte, D., Dahlen, T., & et al. 2009, NICMOS Data Handbook

Trujillo, I., Aguerri, J. A. L., Cepa, J., & Gutiérrez, C. M. 2001, MNRAS, 328, 977

Viana, A., W. T. e. a. 2009, NICMOS Instrument Handbook