Lab 3: Photometry

Daniel Chui: daniel.chui@mail.utoronto.ca
Amanat Brar, Dusan Izcsik, Ruyi Xu

Dec 7, 2023

ABSTRACT

We used the methods of aperture and differential photometry in python to plot a B-band light curve for the KSP-OT-201509b supernova and applied power-law fitting to estimate its date of first light as September 21st ± 1 day. The date of first detection with a signal-to-noise ratio above three was found to be September 22nd ± 1 day and the date of peak brightness was October 8th ± 1 day. By comparing the difference between DOFL and DOPB we get a rise time of 17.5 days ± 1 day.

1. Introduction

Photometry is an important tool in the field of astronomy because it allows astronomers to measure the apparent brightness of objects in space. The method of photometry involves taking the intensity and flux of a well understood reference object in an image and using its properties to make conclusions of the intensity and flux of a different target object. The underlying motivation of this paper is to break down this method into its parts with particular emphasis on CCD calibration, and what are called 'aperture' and 'differential' photometry.

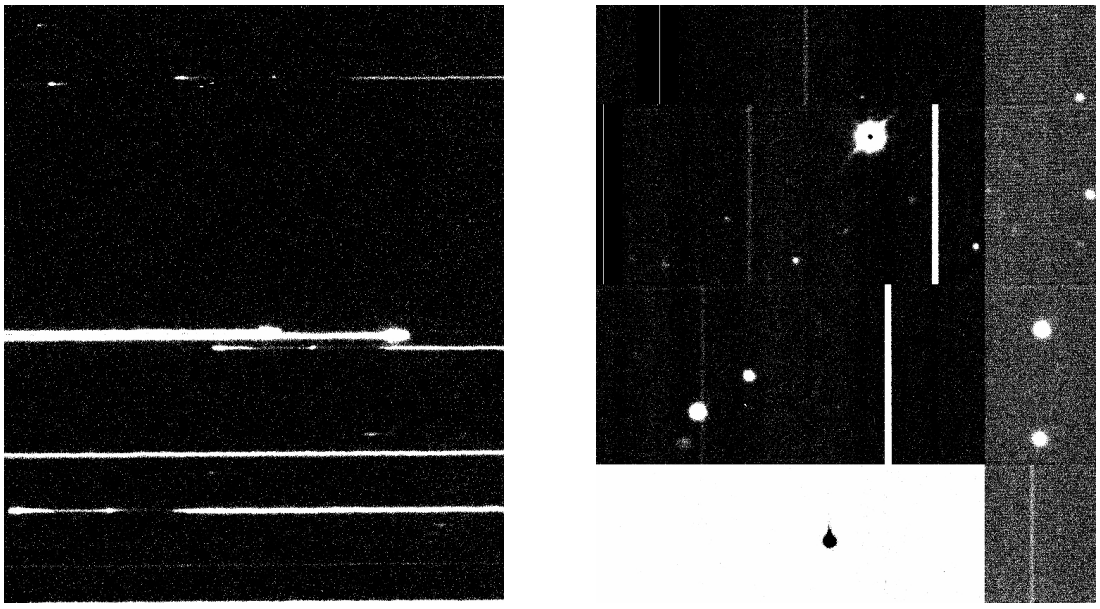
Like spectroscopy, photometry is done by calibrating photon rates to some known reference in order to make meaningful conclusions. Since the intensities measured by a CCD provide only relative information, we need to translate the measurements to concrete quantitative values through our knowledge of reference stars. Reference stars have a known brightness so we compare their intensity values to their magnitude and apply this relationship to the target object. Unlike spectroscopy, we are taking multiple images and so calibration is done repeatedly. The benefit of doing repeated calibration is that some slight inconsistencies from image to image are negligible. For example, imaging doesn't require consistent weather conditions from night to night because all objects in a particular image are affected equally.

Aperture photometry is the process of measuring the flux of objects in a CCD image. The nuance in taking this measurement involves creating an aperture and annulus that contains all the object's information while accounting for noise and biases. An aperture is a radius placed around an object in an image which the intensities used in calculations are constrained to. An annulus is a ring with thickness placed outside, and separate, from the aperture. The annulus contains the background of the object and the pixels and intensities it contains are used to calculate noise.

By using the differences of flux between the target object and reference stars we can apply the logarithmic magnitude scale alongside the reference magnitudes to find the target object's magnitude. This routine is called differential photometry and in this paper it is used to plot a supernova light curve and estimate its time of first light. Specifically, the photometry involved in plotting the light curve requires reducing data over 433 CCD images, using aperture photometry to find the date of first detection and fluxes, and differential photometry to measure the SN brightness over days. Lastly, a fit is done to the normalised flux over time in order to estimate the date of first light: the time the supernova explodes in our reference frame on earth.

2. Data and Observation

The supernova data was provided my Professor Dae-Sik Moon[6] and is a collection of 433 fits files of night sky images. The images are 710 x 850 pixel resolution and are taken over a range of dates from September 20th, 2015 to Octover 30th, 2015. Each CCD pixel has an 0.4 x 0.4 arcsec angular size for a resulting 15 x 15 arcmin angular size over the entire image. Each image is centred, dark-reduced, and flat-fielded but over the entire collection, a handful of poor images were filtered by eye and corrected for intensity overflow.



(a) Bad Image Ex 1.

(b) Bad image Ex 2.

Fig. 1.—: Two examples of bad images that were filtered by eye. In Figure(a) the stars are streaked across the width of the image and in Figure(b) the image is segmented into rectangular chunks of varying intensity. There is also an overflow pixel at the centre of the star towards the upper middle of the image.

The reference objects used for the differential photometry analysis are three stars whose information was given to us by Professor Dae-Sik Moon in *Table 1.* from the Lab 3 manual.

Reference Stars	(RA,DEC)	B-band Magnitudes
REF 1	(00:56:49.70, -37:01:38.31)	16.32±0.07
REF 2	(00:56:46.43, -37:02:29.50)	17.16±0.08
REF 3	(00:56:58.27, -36:58:16.60)	15.61±0.02

Table 1:: Reference Star Information

3. Data Reduction

After initially screening poor images by eye, a collection of quantitative processes were done to further reduce the image count to only useful data. The first was correcting for odd pixels that received an overflow of intensity. Overflow happens when a pixel receives an intensity value that exceeds the critical value held within a single bit. This leads the preceding bit to flip which causes the excess intensity value to read negative. To correct overflow, a line was added when reading in the file that adds the positive peak back when it's read as negative [6.1]. The noise in the images also tended to read off as small negative pixels so we performed CCD intensity calibration by adding an intensity value of 500 to all pixels in the routine. This calibration solved some negative log10 errors and did not affect the results of the differential method which deals with relative flux values. Next, some of the images had an excess of noise and were filtered out by SNR. Signal-to-noise filtering was done after the differential photometry step by checking the ratio between the magnitudes and magnitude errors for the SN. If the star/SN did not produce a strong enough signal and was 'muffled' by the noise such that SNR was less than three, it was discarded[6.7]. The final step in reducing the data was done after plotting a light curve. Significant outliers in the plots were traced back to their images and discarded. These images often contained hot pixels or passing objects within the aperture of the stars and SN that did not represent meaningful SN evolution.

4. Data Analysis

Aperture photometry was done through the use of binary masks. In this paper, the reference stars and the SN were masked through the addition of an aperture and annulus mask. The masks allowed us to perform direct numerical operations with the images without scaling the data up or down. In depth, the method of aperture photometry involved the following steps: taking the point spread function of the CCD and its standard deviation, overlaying the binary mask, and recording the pixel intensities through the aperture and annulus to find flux, background values, and uncertainties.

The point spread function describes the behaviour of how the CCD processes an object's light. It's intrinsic and need only be calculated for the analysis. It shows how the light is spread out over the detector. The goal is to capture this light within an aperture and by taking the aperture radius as three sigma we get 95% of an object's light. By convention the annulus radii are taken as 6 sigma and 12 sigma. In this paper reference star 1 was used to calculate a PSF sigma of 2.8 [6.2]:

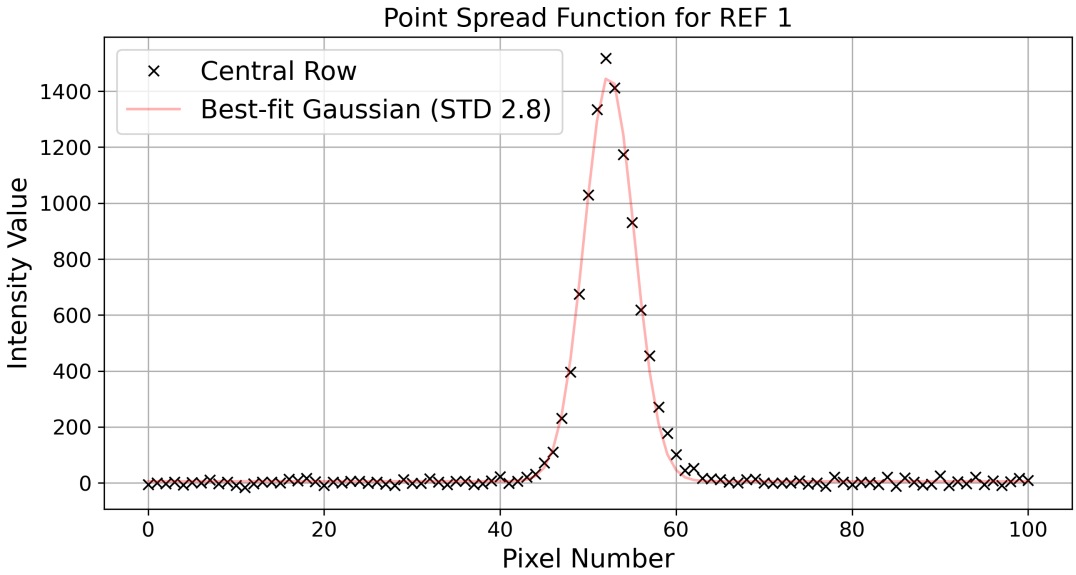


Fig. 2.—: Plot of the PSF for the CCD. The data is a horizontal slice of intensities across the central row of a reference star 1 image. For all objects in sky the PSF should be approximately the same and so the sigma value here, 2.8, is used throughout the analysis.

As mentioned, the aperture and annulus are done with binary masks, making operations simple and noninvasive. To make them we created a separate array with dimensions matching the image of the target object and filled it with one and zero values. One- if its in the aperture or annulus and zero if not. With the PSF computed, the radii for the masks was straight forward. To apply the masks we did a matrix-matrix multiplication of the two arrays. This multiplies the parts of the object image by one if it falls within the mask and zero if its outside. For example, we can see by calling (aperture * sub_image), we have the array of intensities in the aperture. With information about the intensities passing through the mask we calculated the flux, background, and uncertainties for the stars and SN [6.3]. The first observation of the SN to surpass SNR three was named the date of first detection. It was found as Sep 22nd ± 1 day[1], and the file before the first detection is the last non-detection on Sep 21st ± 1 day (see Figure 4). The validity of the aperture routine is demonstrated by comparing the fluxes of the reference stars in Figure 3.

[1]The uncertainties for all dates in this lab are given a general ± 1 day to account for each night observation since Moon explained that error propagation for dates was not advisable.

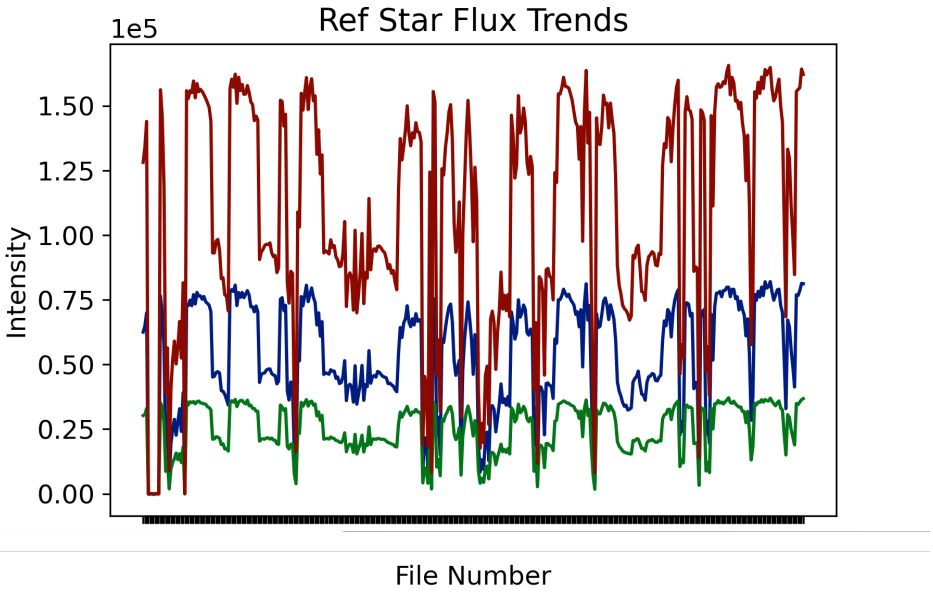
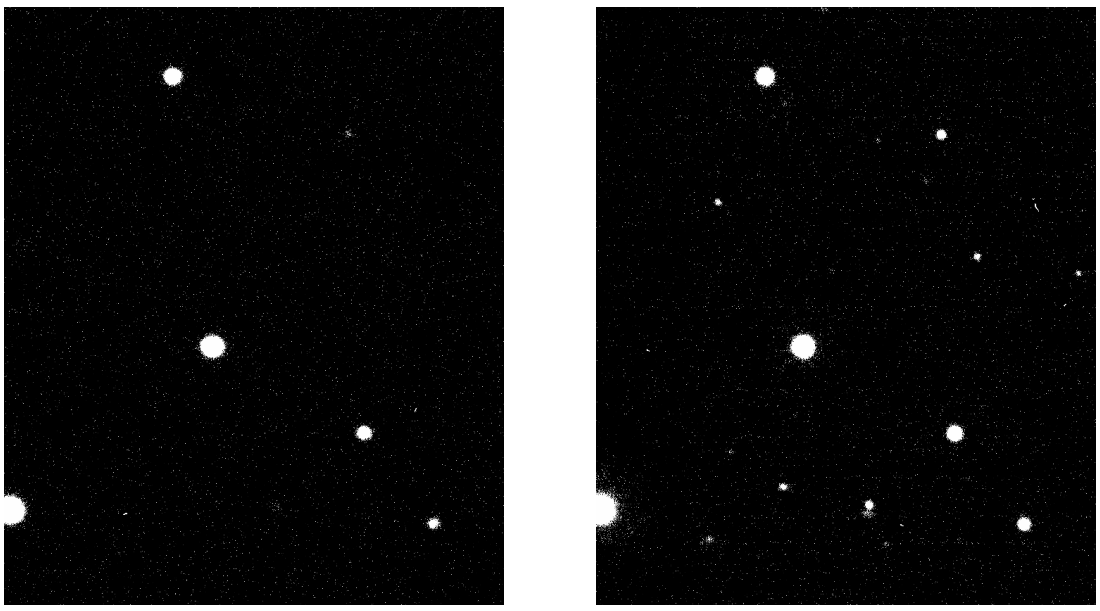


Fig. 3.—: Rough plot of the flux/intensity values of the reference stars over time. Besides the occasional outliers, the fluxes follow the same trends which tells us that that the aperture method is consistent. The trends in flux are accounted for in differential photometry so long as this consistency persists.



(a) First Detection

(b) Last Non-Detection

Fig. 4.—: Figure(a) presents the image of the first detection while Figure(b) presents the last non-detection. Adjusted in DS9 to a similar contrast, we can see that its is apparent in neither image that the SN has exploded (to the right off bottom left star).

The uncertainty for intensity values involved the gain, background uncertainty, and pixel counts for the aperture and annulus. The gain tells you how many electrons it takes to induce one intensity value. In the analysis of this paper gain was always set to four. The uncertainty equation is given as:

$$\Delta F = \sqrt{\frac{F}{g} + N_{aperture}\left(1 + \frac{\pi}{2}\frac{N_{aperture}}{N_{annulus}}\right)\sigma_{bg}^2}$$

The differential photometry step involved taking the flux values and given B-band magnitudes of each reference star and applying the logarithmic magnitude scale equation to obtain an estimate of the SN magnitude [1]. We then averaged the magnitudes of the SN from each reference star in an image and took that to be the estimate for one picture [6.5]. Like flux, error propagation from flux to magnitude was an involved process and was addressed using equations [2][3]

$$M_{SN} = m_{ref_i} - 2.5\log(\frac{I_{SN}}{I_{ref_i}}) \tag{1}$$

$$\sigma_{mref} = 1.087(\frac{\Delta F_{ref}}{F_{ref}}) \tag{2}$$

$$\sigma_{mSN} = \sqrt{m_{ref}^2 + \sigma_{mref}^2 + \sigma_{mSN}^2} \tag{3}$$

Since we averaged the SN B-band magnitudes over the reference stars we have to propagate error similarly:

$$\sigma_{MSN} = \sqrt{\frac{\sigma_{ma}^2 + \sigma_{mb}^2 + \sigma_{mc}^2}{9}} \tag{4}$$

With aperture and differential photometry complete, we plotted the light curve. It shows the change in B-band magnitude of the SN over time. The curve was fitted with a quartic polynomial in order to estimate the date of peak magnitude/brightness which came out to be October 8th, ± 1 day.

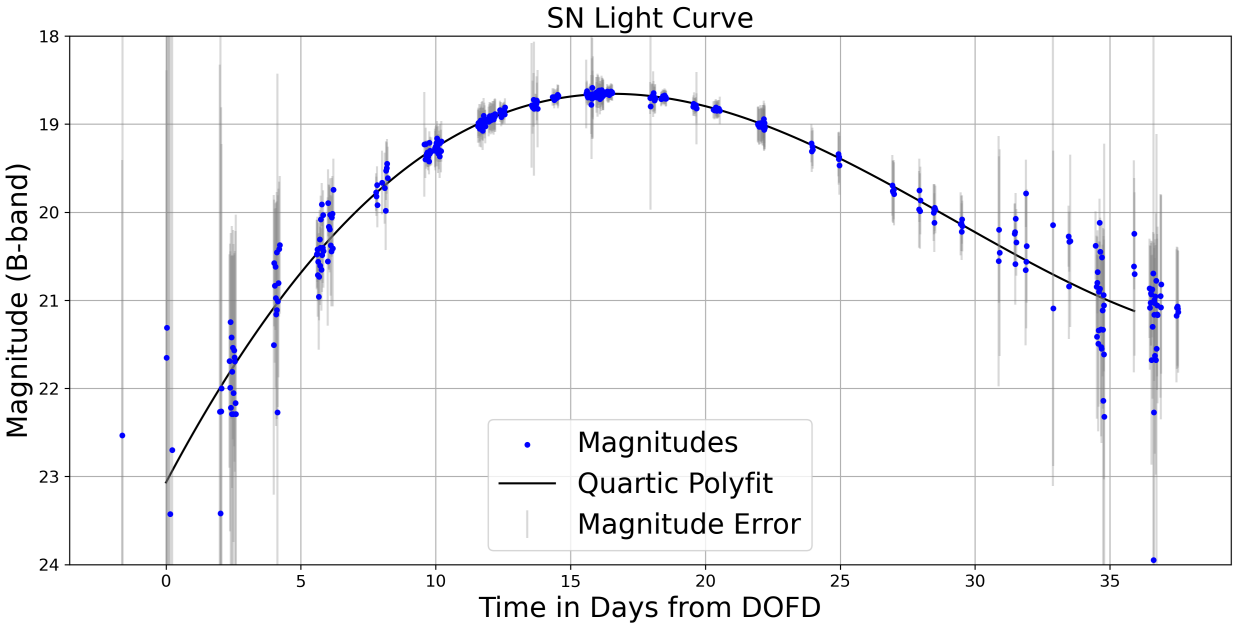


Fig. 5.—: The lightcurve for KSP-OT-201509b fit with a quartic polynomial. The polynomial estimates a magnitude of 18.7 .

To find the date of first light, we took the early brightness curve where brightness was under 40% of its peak and fit a quadratic equation [5]; the minimum of which would fall on the date of first light. The brightness curve is a normalised flux representation of the light curve. To compute it we converted the magnitudes back to fluxes and divided by the peak flux [6.8]. The quadratic fit estimated the date of first light on September 21st, ± 1 day and the rise time which is the amount of days between the date of first light and peak brightness was estimated as 17.5 days ± 1 day. [6].

The conversion back to flux was as follows:

$$F_{Norm} = \frac{10^{-0.4*M}}{F_{Max}}$$

$$\Delta F_{Norm} = 0.4F \log(10)\sigma_M$$

The fitted equation:

$$y = C(t - t_1)^2 \tag{5}$$

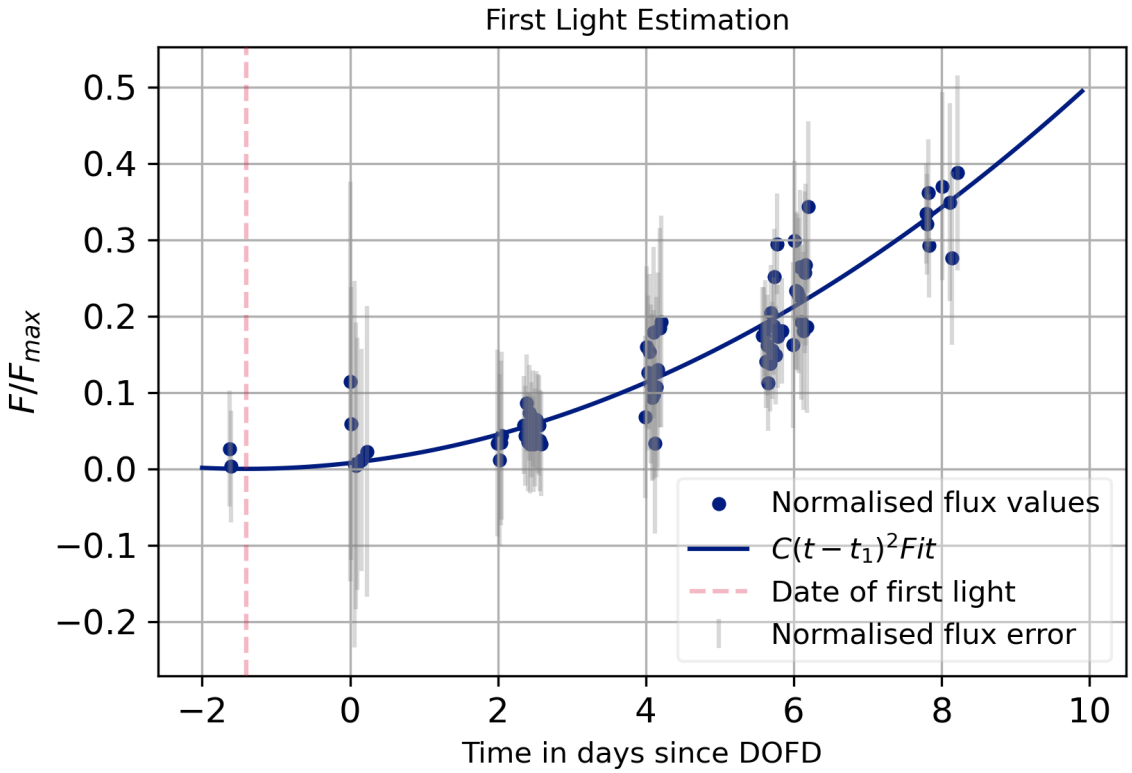


Fig. 6.—: The early epoch of the SN light curve in normalised flux values. The red verticle line estimates the date of first light. Which is also the minimum of the quadratic fit.

Rise time:

$$\text{Date of Peak Magnitude - Date of First Light} = (16.15) - (-1.40) = 17.55 \text{ Days} \pm 1 \text{ day} \tag{6}$$

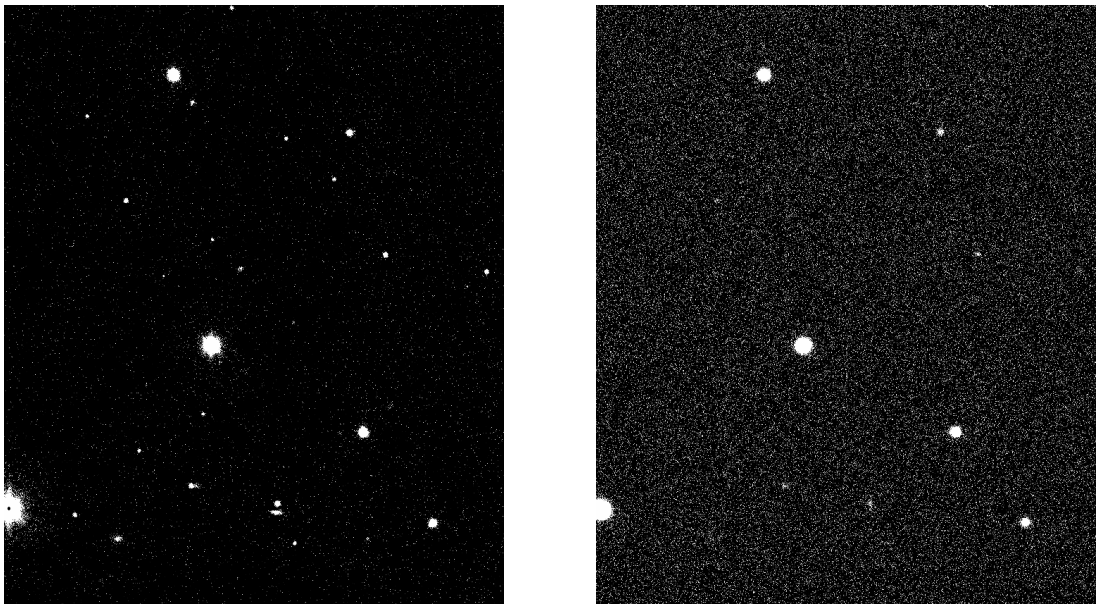
We can confirm that the quadratic is a good fit by computing the chi square value for 2 parameters. Chi squared was found to be within an acceptable limit [6.9]. It suggests a confidence level between 68 and 90 percent as suggested by the *Numerical Recipes* Chapter 15 Table provided, again, by Professor Dae-Sik Moon.

$$\chi^2 = \sum \frac{(F_{Norm} - P2)^2}{P2} = 4.15$$

5. Discussion & Conclusion

After comparing the results in this paper to that of Professor Dae-Sik Moons we find a discrepancy. While the results of this paper mostly agree with values like the peak magnitude date, rise time, and date of first light, it differs in the date of first detection. This analysis finds a date of first detection on the

21st while Moon's paper puts it almost on the 25th. In this paper SNR was found by taking the ratio of B-band magnitude and B-band uncertainty, but it was originally suggested by Moon to do it via fluxes. This point of contention is likely not relevant though and its more likely that there is an issue with the aperture photometry step. Our first idea was that the +500 calibration value may have had an affect on SNR. Changing this number did affect SNR slightly but not in any meaningful way. After skimming through the files by hand we get a better estimate of first light of the 26th. The files leading up to the 26th are noisy and hard to read which could play a role in the SNR detection error. By comparison here are the first detection as perceived by eye and the last non detection:



(a) First Detection (b) Last Non-Detection

Fig. 7.—: Figure(a) presents the image of the first detection taken by eye instead of SNR and Figure(b) presents the last non-detection. The SN is noticeably visible in the first detection but, like many images before it, the last non detection image is noisy and misses a lot of features in the night sky including the SN.

To go about resolving this issue we need only look through the files by eye and find the first detection; moreover, discard the noisy images that muffle low flux objects.

6. Bibliography

Moon, D. S., Ni, Y. Q., Drout, M. R., González–Gaitán, S., Afsariardchi, N., Park, H. S., ... & Lee, Y. (2021). Rapidly declining hostless type ia supernova ksp-ot-201509b from the kmtnet supernova program: transitional nature and constraint on 56ni distribution and progenitor type. The Astrophysical Journal, 910(2), 151.

Appendix: Code

A collection of code used in the analysis section of the paper.

6.1. Reading function and Setting DOFD

The following code includes the function for reading in .fits files and the method of setting the DOFD and converting .fits files headers into concrete time values (days).

```
#Imports
import numpy as np
from astropy.io import fits
from datetime import datetime

img_files = np.loadtxt('AST325-326-SN-filelist.txt', dtype=str, delimiter=',')

#Read File Function
def read_file(file_name):
    hdu = fits.open(file_name)[0]
    data = hdu.data.astype(int)
    header = hdu.header
    data[data < -1000] = ((2**15-1)-np.abs(data[data < -1000])) + (2**15-1)
    return data, header

#Date Array
hed1 = fits.open('AST325-326-SN-20150922.5444.fits')[0].header #Setting 0 as DOFD
ref = datetime.fromisoformat(hed1["DATE-OBS"])
file_times = []
for f in img_files: #Converting file headers into a time array
    header1 = fits.open('/Users/danielchui/Lab3/snfits/'+str(f))[0].header
    dt = datetime.fromisoformat(header1["DATE-OBS"])
    file_times.append((dt-ref).total_seconds()/86400)
```

6.2. Point Spread Function

The following code is used to obtain the PSF sigma value.

```
import numpy as np
from scipy.optimize import curve_fit

def gaus(x,a,x0,sigma, c):
    return a*np.exp(-(x-x0)**2/(2*sigma**2)) + c

sub_im_a = data1 #Data box around a single reference star a in one image (data1)
```

```
amp = 100 #Best guess for amplitude of PSF
x, y = np.arange(sub_im_a.shape[0]), sub_im_a[(sub_im_a.shape[0]-1)//2,:] # Splitting the image down th
popt, pcov = curve_fit(gaus, x,y, p0=[amp,sub_im_a.shape[0]//2,10, 0])# and curvefiting the gaussian fu
print("The sigma value is:", popt[2])
```

6.3. Aperture Photometry Routine per Object

The following routine is used per object to obtain its fluxes, background, and uncertainties.

```
def create_aperture_function(size, radius):
    x, y = np.meshgrid(np.arange(size), np.arange(size)) #meshgrid to overlay
    center_x, center_y = size // 2, size // 2 # distance to centre of aperture
    distance = np.sqrt((x - center_x)**2 + (y - center_y)**2)
    aperture_mask = distance <= radius #Creating binary truth value
    return aperture_mask

def create_annulus_function(size, radius_in, radius_out):
    x, y = np.meshgrid(np.arange(size), np.arange(size))
    center_x, center_y = size // 2, size // 2
    distance = np.sqrt((x - center_x)**2 + (y - center_y)**2)
    annulus_mask = (distance >= radius_in) & (distance <= radius_out)
    return annulus_mask

def get_flux(image,gain):
    N_an = annulus[annulus>0].size #Number of pixels in annulus
    N_ap = aperture[aperture>0].size
    bg = np.sum(image*annulus)/N_an #Average noise in background
    flux = np.sum((image- bg)*aperture) #Flux
    bg_var2 = np.var(image*annulus) #Variance in background
    flux_error = (flux/gain + N_ap*(1 + (np.pi/2)*(N_ap/N_an))*bg_var2)**.5 #Flux error
    return N_an, N_ap, bg, flux, flux_error, bg_var2

aperture_radius = 3*round(PSF,1)
radius_in = 6*round(PSF,1)
radius_out = 12*round(PSF,1)
aperture = create_aperture_function(image_size, aperture_radius)
annulus = create_annulus_function(image_size, radius_in, radius_out)

N_an, N_ap, bg, flux, flux_error, bg_var2 = get_flux(sub_im,4)
```

6.4. Double For Loop Structure

Aperture and Differential photometry are done in a double for loop. The first loop goes over each image file while the second goes over each reference star. Differential photometry, which follows this subsection, is done entirely within in the first loop which concerns one SN per image.

```
from astropy.wcs.utils import skycoord_to_pixel

#Double For Loop
for i, image in enumerate(img_files):

    data1, header = read_file(image)
    w = WCS(header)

    #Refernce Information
    star_pos_a = skycoord_to_pixel(stars[0], w)
    star_pos_b = skycoord_to_pixel(stars[1], w)
    star_pos_c = skycoord_to_pixel(stars[2], w)
    star_band_a = 16.32 #+- 0.07
    star_band_b = 17.16 #+- 0.08
    star_band_c = 15.61 #+- 0.02
    m_ref = [star_band_a,star_band_b,star_band_c]
    ref_stars = [star_pos_a, star_pos_b, star_pos_c]

    #Empty arrays to be filled
    star_fluxes = []
    star_flux_er = []

    for j, star in enumerate(ref_stars):
        star_x, star_y = star
        star_x_int, star_y_int = star_x.astype(int), star_y.astype(int)
        boxsize = 33

        #Creating a 'windowed' sub image around each ref star
        sub_im = data1[star_y_int-boxsize: star_y_int+boxsize+1,star_x_int-boxsize: star_x_int+boxsize+
        sub_im = sub_im+500 #CCD calibration value

        ##############################
        ##Begin Aperture Photometry##
        ##############################
```

6.5. Differential Photometry Routine

For each SN, the log scale magnitude calculations are done and averaged over the reference stars. Uncertainty is also included.

```
    ##############################
    ###End Aperture Photometry###
    ##############################

SN_pos = skycoord_to_pixel(SN, w)
SN_x, SN_y = SN_pos
SN_x_int, SN_y_int = SN_x.astype(int), SN_y.astype(int)
boxsize = 33
image_size = 67

sub_im_SN = data1[SN_y_int-boxsize: SN_y_int+boxsize+1,SN_x_int-boxsize: SN_x_int+boxsize+1]
sub_im_SN = sub_im_SN + 500

N_SN_an, N_SN_ap, bg_SN, flux_SN, flux_error_SN, bg_var2_SN = get_flux(sub_im_SN,4)

sigma_ref_a = 1.087*(star_flux_er[0]/star_fluxes[0])
sigma_ref_b = 1.087*(star_flux_er[1]/star_fluxes[1])
sigma_ref_c = 1.087*(star_flux_er[2]/star_fluxes[2])
sigma_SN =  1.087*(flux_error_SN/flux_SN)
sigma_m_a2 = (0.07**2 + sigma_ref_a**2 + sigma_SN**2)
sigma_m_b2 = (0.08**2 + sigma_ref_b**2 + sigma_SN**2)
sigma_m_c2 = (0.02**2 + sigma_ref_c**2 + sigma_SN**2)
sigma_m = ((sigma_m_a2 + sigma_m_b2 + sigma_m_c2)**.5)/9
```

6.6. Removing Bad Files

Visibly poor images were removed by hand but a few bad images got through and were removed systematically: many times in hindsight. The code for removing "nan" resulting from logging negative flux values are given here.

```
    ###################################
    ###End of Differntial Photometry###
    ###################################

    #Only appending data points with positive flux values less than a reasonable 9000
    if 0 < flux_SN < 9000:
         fluxes.append(flux_SN)
         flux_er.append(flux_error_SN)
         m_SN_a = m_ref[0] + 2.512*np.log10(star_fluxes[0]/flux_SN)
         m_SN_b = m_ref[0] + 2.512*np.log10(star_fluxes[1]/flux_SN)
         m_SN_c = m_ref[0] + 2.512*np.log10(star_fluxes[2]/flux_SN)

         m_SN = (m_SN_a + m_SN_b + m_SN_c)/3
```

```
            mags.append(m_SN)
            mags_un.append(sigma_m)
        else: file_times.remove(file_times_copy[i])



##########################################
######Cleaning out Nans and Outliers#######
##########################################

for i in np.arange(0,len(mags),1):
    if math.isnan(mags[i]):
        bad.append(i)

for i in bad:
    mags[i] = "pop"
    mags_un[i] = "pop"
    file_times[i] = "pop"

k = "pop"
while (k in mags):
    mags.remove(k) #76
while (k in file_times):
    file_times.remove(k)
while (k in mags_un):
    mags_un.remove(k)

purified_mags = np.array(mags)
purified_img_files = np.array(file_times)
purified_mags_un = np.array(mags_un)
```

6.7. SNR

Here is the formidable SNR calculation.

```
snr = purified_mags/purified_mags_un #Computing all SNR

purified_mags = purified_mags.tolist()
purified_img_files = purified_img_files.tolist()
purified_mags_un = purified_mags_un.tolist()

#Filtering all SNR with a backwards list removal (im a genius)
num = np.arange(0,len(mags),1)
revnum = num[::-1]
for i in revnum:
```

```
    if snr[i] < 3:
        purified_mags.remove(purified_mags[i])
        purified_img_files.remove(purified_img_files[i])
        purified_mags_un.remove(purified_mags_un[i])

purified_mags = np.array(mags) #Converting everything back to useful arrays
purified_img_files = np.array(file_times)
purified_mags_un = np.array(mags_un)
```

6.8. Normalised Flux Calculations

After having computed the normalised flux values as described in the report we limit the values for the upcoming quadratic fit.

```
norm_flux = [normalised flux data]

res_norm_flux = []
for i in np.arange(0,82,1):
    if norm_flux[i] < 0.4:
        res_norm_flux.append(norm_flux[i])
res_norm_flux = np.array(res_norm_flux)

res_time_indices_worst= np.where(norm_flux<0.4)[0]
res_time_indices = res_time_indices_worst[res_time_indices_worst < 200]

res_times = []
res_norm_flux_er = []
for i in res_time_indices:
    res_times.append(purified_img_files[i])
    res_norm_flux_er.append(norm_flux_er[i])
res_times = np.array(res_times)
```

6.9. Chi Squared value and 'goodness' of fit

```
def P2(t,t1,c):
    return c*(t-t1)**2
expected_values = P2(res_times,popt2[0],popt2[1])
red_X2 = (np.sum(((res_norm_flux-expected_values)**2)/expected_values))
```