Title

Lucas, Miles[1] and Brandon, John[1]

Iowa State University

(Dated: 30 August 2017)

Abstract will go here.

I. INTRODUCTION

II. DATA ACQUISITION AND SETUP

Observations were made on at the Zaffarano Hall observation deck in Ames, Iowa. The night was very clear and the ambient temperature was around 32 °C. Observations were made using a Meade 8” reflector telescope with an SBIG ST-402ME CCD camera with internal V, B, and I filters.

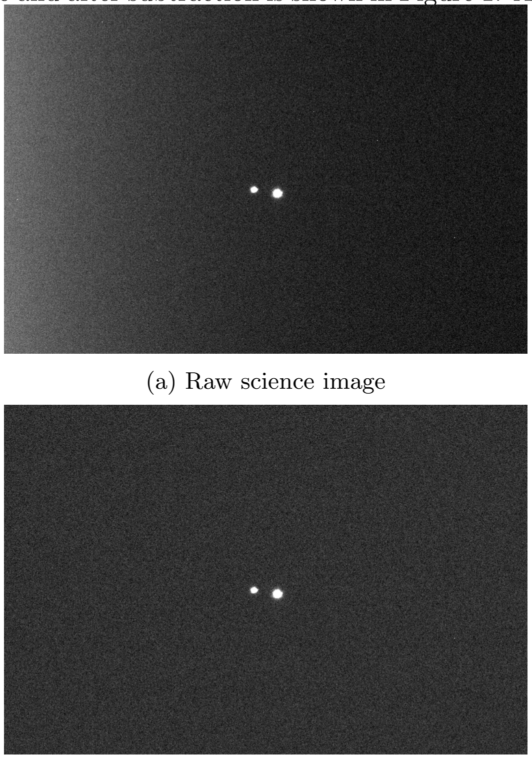
To setup the telescope we used the Meade's GPS functionality to automatically calibrate its position. This calibration involved using two point sources as guides and orienting the telescope centered on the point source. For these measurements, the calibration sources were Arcturus and Altair. We used a 32 mm eyepiece for our calibration alignment. Using the Meade selector, we navigated to the β Cygni system We then switched to a 9 mm eyepiece for fine-alignment on our target. Once we were well centered on our target we removed the eyepiece and inserted our CCD camera.

Using CCDops we selected the V filter and began focusing the CCD camera with a manual knob on the telescope. The exposure time for these focusing grabs was 0.05 s. We based our focus on maximizing the peak amplitude read out in CCDops. We took multiple images at multiple exposures with different filters, all recorded in Table **??**. For each image grab, we set up an autograb using CCDops and took the frames. At each exposure level, we also took dark frames using autograb in order to reduce systematic error.

III. DATA ANALYSIS

Our analysis included some simple image processing and then used machine learning to determine the pixel scale and field of view.

Our processing pipeline involved finding median dark frames for each exposure length and then subtracting the dark frames from each science image. We use median dark frames rather than mean because we wanted to avoid outliers in the form of cosmic rays or dark pulses. To create the median images, we used a python script that utilizes numpy's array operations for getting the median of the data arrays from each FITS file. Once we saved the three median dark frames (0.1 s, 0.5 s, 1.0 s) we used AstroImageJ to automate the subtraction for the 26 science images. A comparison of the science images before and after subtraction is shown in Figure 1. As you can see, the thermal excitation caused by the camera's electronics is removed.


(a) Raw science image

(b) Processed science image

FIG. 1: Comparison of the raw and processed science images of Albireo in photometric V at 0.1 s exposure.

To calculate the pixel scale of our image, we need to find the center of each star.

IV. RESULTS

V. CONCLUSIONS

ACKNOWLEDGEMENTS

Thank you to Dr. Charles Kerton and Brandon Marshall for their guidance and assistance in this work.

Appendix A: Observation Log

TABLE I: Observed 30 August 2017 by Miles Lucas and John Brandon

Time	File	N Frames	Object	Filter	Exposure	Camera Temp.	Notes
20:52	Albireo_V_tenth_	10	Albireo (β Cyg)	V	0.1 s	5 °C	
20:56	Albireo_V_tenth_dark_	5	Albireo (β Cyg)	V	0.1 s	5 °C	Dark frames
20:59	Albireo_V_half_	3	Albireo (β Cyg)	V	0.5 s	5 °C	
21:01	Albireo_V_half_dark	5	Albireo (β Cyg)	V	0.5 s	5 °C	Dark frames
21:03	Albireo_V_full_	3	Albireo (β Cyg)	V	1.0 s	5 °C	
21:04	Albireo_V_full_dark_	5	Albireo (β Cyg)	V	1.0 s	5 °C	Dark frames
21:05	Albireo_B_tenth_	5	Albireo (β Cyg)	B	0.1 s	5 °C	
21:06	Albireo_I_tenth_	5	Albireo (β Cyg)	I	0.1 s	5 °C	

Appendix B: Analysis Scripts

File: lab2/src/measurements.py

```python
from glob import glob

import numpy as np
from astropy.io import fits
from astropy.stats import sigma_clipped_stats
from photutils import DAOStarFinder


def get_sep(img):
    mean, median, std = sigma_clipped_stats(img, iters=5)
    daofind = DAOStarFinder(fwhm=3.0, threshold=5. * std)
    sources = daofind(img)
    positions = np.array([sources['xcentroid'], sources['ycentroid']]).T
    return np.linalg.norm(positions[0] - positions[1])

def get_pscale(seps, dalpha):
    pscales = dalpha / np.array(seps)
    pscale = np.mean(scales)
    pscale_std = np.std(scales)
    return pscale, std

def get_fov(pscale, std, dalpha, leng):
    fov = leng * pscale
    std = leng * pscale**2 / dalpha * std
    return fov, std

if __name__ == '__main__':
    filenames = glob('../data/science/processed/*V*tenth*')
    imgs = [fits.getdata(file) for file in filenames]
    seps = [get_sep(img) for img in imgs]
    dalpha = 35.3
    pscale = get_pscale(seps, dalpha)
    x_len, y_len = imgs[0].T.shape
    fovx = get_fov(*pscale, dalpha, x_len)
    fovy = get_fov(*pscale, dalpha, y_len)

    print(pscale)
    print(fovx)
    print(fovy)
```