Methodology for image production to assist with planet searching within the Dark Energy Survey to find Planet Nine

Christopher Cooper
Candidate Number: 78641
Supervisor: **Dr. Kathy Romer**
Department of Maths and Physics
University of Sussex

This project report is submitted for the degree of
BSc Physics (with a Foundation Year)

August 2016

I would like to dedicate this dissertation to my incredibly supportive friends and family, which includes my devoted parents along with the supportive staff at the University of Sussex who have helped me overcome an incredibly difficult period of my life. Also a specific dedication to Dr. Kathy Romer who has not only acted as a supervisor to me during the course of my Final Year project but has also acted as a mentor throughout the duration of my degree and has helped to support and encourage me even during my worst times at the University of Sussex.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified explicitly within the main body of the dissertation or the Preface section.

Christopher Cooper
Candidate Number: 78641
August 2016

Preface

Throughout my Final year Project work I was required to use the Dark Energy Survey Data Management system (DESDM), whilst learning to use the system and in order to produce code that worked with it I required some technical help from Matias Carrasco-Kind (Research Scientist, University of Illinois) and Robert A Gruendl (Research Associate Professor of Astronomy, University of Illinois).
Further help with accessing and using the data made available on the DESDM system was also provided by; Philip Rooney, Carlos Fernando Vergara Cervantes, and Alberto Bermeo. I'd also like to give extra thanks Philip Rooney for introducing me to essential tools such as TOPCAT (Tool for OPerations on Catalogues And Tables) which allowed for me to be much more efficient when handling large astronomical datasets.

Also I am grateful for the help and information provided by David Gerdes (Arthur F. Thurnau Professor, Physics Department, University of Michigan) that allowed for me to use data to test my code during the proof of concept stage.
Again, I'd also like to acknowledge the support provided by Dr. Kathy Romer, in particular teaching me how to break down problems that I had faced during my project so that I could approach them in a more logicial and coherent manner.

Chapter 1 and Chapter 2 used information obtained from various sources, as cited. Chapters 3 and 4 are my original work unless stated otherwise, for example where I discuss packages, libraries, and modules that are not of my own creation. Chapter 5 pertains to future work and conclusions as such it is my original work.
This research made use of the .fits manipulation package Montage. It is funded by the National Science Foundation under Grant Number ACI-1440620, and was previously funded by the National Aeronautics and Space Administration's Earth Science Technology Office, Computation Technologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology.

Abstract

A portion of this report highlights the efforts and goals of the Dark Energy Survey (DES), this also includes a brief discussion of the apparatus used to make observations, DECam. The Dark Energy Camera (DECam) is used in combination with the Victor M. Blanco Telescope located at the Cerro Tololo Inter-American Observatory in Chile.

Whilst a key goal of the DES is to constrain the *w* dark energy parameter, teams such as those led by Professor David Gerdes (Thurnau Professor, Physics Department,University of Michigan) are using the DES data for other purpose and in this case it is to potentially find Planet Nine.

The main purpose of this report is to discuss the code I have created in conjunction with modules and packages created by others parties such as the `Montage` .fits file manipulation package. The intention of my explanation of my code is to provide a detailed resource for future users. I then present the images and webpages produced by my code. These images can then be used to further study the objects of interest featured or in some cases not featured in them.

Lastly I discuss my numerous aims for future work to make my code more versatile and useful in analysing DES data.

Table of contents

List of figures

Chapter 1

Introduction

Beyond the Kuiper belt several trans-Neptunian objects (TNOs) exist and these named so because they are minor planets that orbit the Sun with a greater semi-major axis than Neptune. However these TNOs exist in an arrangement that is not supposed to exist within the confines of our current knowledge of astronomy.
Two significant inferences have been made by C.A.Trujillo and S.S.Sheppard [14] as well as K.Batygin and M.E.Brown [2] that suggest a massive planet, Planet Nine, with a mass that is approximately greater than that of ten units of Earth mass with an orbital period of $15,000$ years is responsible for the anomalous orbital arrangement of the smaller TNOs.

1.1 Overview of the Project

During the course of my degree work I have taken part in several sessions of eyeballing data from both the XMM Cluster Survey (XCS) and the Dark Energy Survey relating to the study of galaxy clusters by Dr. Kathy Romer's doctoral students. The eyeballing of data involved looking at single CCDs and mosaics of exposures from both surveys. A thumbnail (also called a "postage stamp") image making process existed for this task already.
However, the existing process did not satisfy the requirements outlined by Professor David Gerdes (see Appendix D: Email from Professor David Gerdes) during his correspondence with my supervisor Dr. Kathy Romer. I was given the task of providing the foundation for two pieces of code; one that could take a specific set of co-ordinates, epoch, and photometric filter to produce images from the DES database to compare to other surveys, and the second was to be developed to allow for co-ordinates to be given, but to take data from any epoch and any type of photometric filter.

This resulting set of code can produce images from the DES database (DESDM) which can then be used to study known TNOs, TNO candidates, and importantly potential Planet Nine candidates. Furthermore, I had the privilege of using DES data which had previously not been used for this purpose to test my code and look at unnamed TNOs. The importance of cross-examining data between multiple surveys will be discussed during this project report. First of all I will discuss the Dark energy Survey briefly along with some important astronomy and photometry fundamentals relating to the study of the TNOs and the data used.

1.2 The Dark Energy Survey

The Dark Energy Survey (DES) is an international collaboration aiming to achieve a 5000 square degree optical/near infra-red imaging survey, in five filters (g,r,i,z,Y), of the sky in the Southern Hemisphere (Figure 1.1 on page 2), this is known as the Wide Survey. A second survey known as the DES Supernova Program (DES-SN) also being carried out, and it is this survey that is of interest currently to those studying TNOs for reasons that will be discussed in Chapter 2. However, the Wide Survey once its five year survey is completed will also be on great importance in the study of TNOs and other transients in the solar system.

In the Wide survey 4000 square degrees of the survey overlap with other surveys such as the South Pole Telescope survey (SPT) and VST ATLAS (Figure 1.2 on page 3).

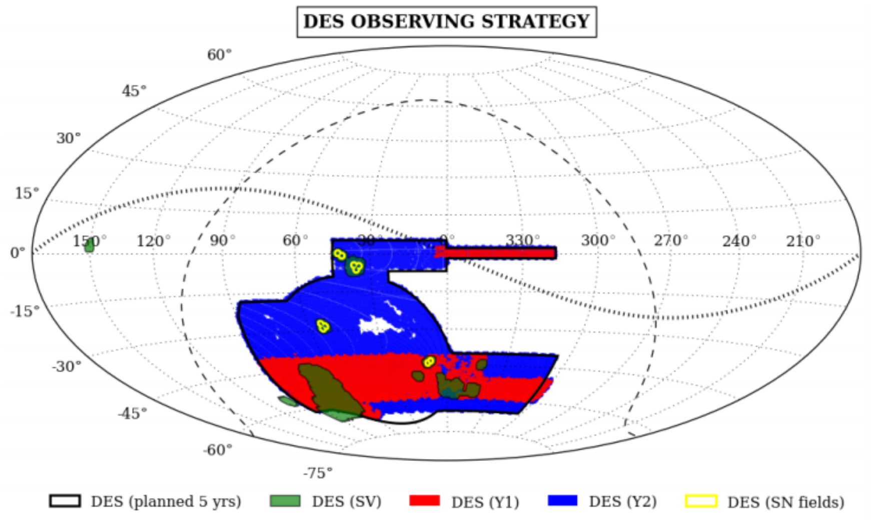


Fig. 1.1 Dark Energy Survey footprint [6, page 3]

The key goal of the DES is to constrain the dark energy equation of state, *w*, along with other goals such as testing alternate models of gravity. As with many other scientific

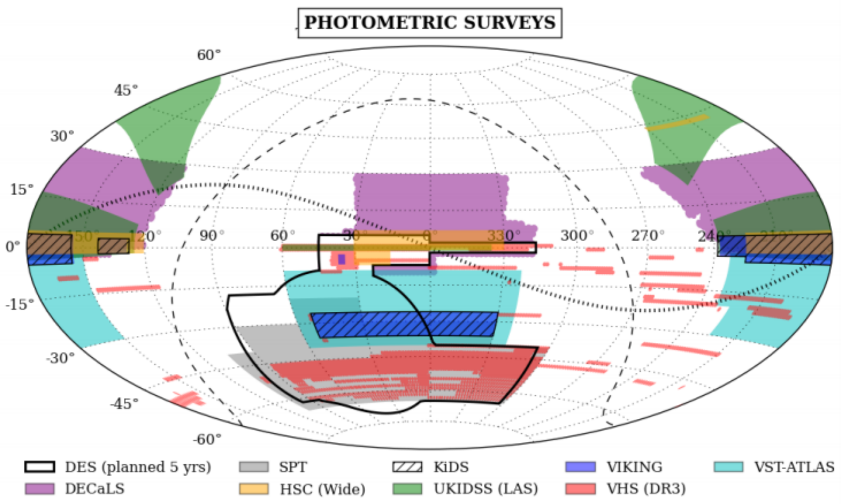


Fig. 1.2 Comparison of DES footprint to other surveys [6, page 3]

projects and surveys the data obtained is being used for a wide range of other experiments and investigations. During observation periods a small amount of time is given to teams that are studying the likes of TNOs and supernovae. Further information on the scope of the survey and other aspects of DES can be found in the modified and publicly available version of the white paper submitted to the Dark Energy Task Force [13].

1.3 The Dark Energy Camera

The Dark Energy Camera (DECam) is a wide-field 570 megapixel camera that is mounted onto the four metre aperture Blanco Telescope in Chile, that has been previously used to take measurements relating to dark energy[5] . It began operation in September 2012, beginning the scientific verification (SV) period from November 2012 until Februrary 2013, this was then followed up by beginning of full scale operations with year one (Y1) in August 2013. Observation periods will continue annually until at least 2018, meaning there will be at a minimum five observing seasons worth of data produced.
As required by the survey the camera is designed to be sensitive to the five photometric bands; g,r,i,z,and Y. The camera utilises 62 charged-coupled devices (CCDs) for its sensor array and a further 12 smaller CCDs are used for calibration and guiding the apparatus [13].

Chapter 2

Planet searching with DES

2.1 Studying the solar system wth data from the DES

The main goal of image making for DES is to produce co-added images to study galaxy clusters and the lensing that is caused by these and other massive objects in the survey footprint. In order to produce co-added images, which are multi-epoch images (composed of many single epoch exposures) usually spanning the five photometric filters (g,r,i,z,Y). However, in relation to this project I am only interestied in creating single epoch images from observation seasons as TNOs are transient objects, meaning that multi-epoch images will be of little use due to the need to observe movement of the objects.

Currently the complete observation seasons are ; scientific verification (SV), year one (Y1), year two (Y2), and year three (Y3). In the querying of DESDM within my code for creating multiple single epoch images for one set of co-ordinates I have used a table within the database that gives me access to all of the previously mentioned observation seasons (See Appendix B.1, lines 58-59).
Due to the scope of DES it allows for the study of approximately an eighth of the sky [6], providing a large field to study and observe objects within our solar system.
As of December 2015 a total of 34 new TNOs have been discovered across the DES observation periods, 19 new Jupiter trojans (a minor planet or moon that share the orbit of a larger planet or moon), and 300,000 main-belt asteroids in the Y1 and Y2 seasons alone. From the full five years of observations it is expected that more than 50 new TNOs will be discovered using DES data [6, page 4, Table 1].

2.2 Studying TNOs with the DES

Notable TNO discoveries made using DES data are the two trojans of Neptune located at its fourth Lagrange point (L4), these are QO_{441} and 2014 QP_{441} discovered by D.Gerdes et al [7] during the Y1 and Y2 observation periods.
To put the significance of these two discoveries into perspective, only eight confirmed Neptunian trojans had been discovered prior to the work of D.Gerdes et al. This is due to, in part, the quality and scope of data provided by the DES.
The DES Wide Survey of 5000 square degrees in the g,r,i,z,Y bands is primarily being used for studying areas of cosmological interest such as galaxy clusters and gravitational lensing. As such the DES Wide Survey has a lower cadence due to the shear size of the field and the amount of images it has to take, ten images are to be taken in each filter per tile over the course of the survey, meaning that a total of 50 images per tile are to be taken over a five season period. Whereas the Supernova Program (DES-SN) is of more use in the search for transients due to its relatively higher cadence of weekly intervals, with images being taken in the four bands of g,r,i,z. Of the DES-SN fields, the Stripe-82 fields and the XMM-LSS fields have ecliptic latitudes that coincide with the current position of Neptune [7] (from -20 to -15 degrees) and leading Neptune by 60 degrees of longitude.

2.3 Using DES data to detect TNOs

The capability of transient detection is determined by the depth of the single epoch exposures, rate of motion of the transient and cadence and how regular or irregular the time period between observations is. The equatorial stripe of the DES footprint covers around 500 square degrees within 20 degrees of the ecliptic plane, this makes the DES data incredibly useful to transient study as most of the objects in the solar system are within $\pm 20°$ of the ecliptic plane [6, page 4]. When observing transient objects within the DES several steps are taken to identify known and new candidates.

2.3.1 The supernova difference-imaging pipeline

To begin with the single epoch exposures taken by DECam are run through the supernova difference-imaging pipeline (DiffImg), created by R. Kessler et al. [10]. The pipeline aligns each single epoch image to a deep reference image from the DES Supernova Program (DES-SN), then it performs a pixel by pixel subtraction to remove any unnecessary data. Finally the pipeline examines the subtracted images for positive detection of point source objects, which includes the desired transients which in this case are TNOs [6, Abstract].

2.3.2 Autoscan machine-learning

The second key step is to run the subtracted images through the Autoscan machine-learning algorithm to remove any leftover artefacts and non-point source objects, this was achieved by Goldstein et al. [8]. The algorithm is based on the Random Forest learning technique that falls under a group of methods called ensemble techniques. It works by generating several models and then uses those models to make predictions. Predictions of the models are then compiled into a single prediction which should then operate far better than any of the constituent models. The models are called regression trees (this is why a collection of them is called a forest), these are composed of a set of decisions that then classify specific observations within a given dataset [15].

2.3.3 Reflex motion and triplets

The TNOs apparent motion is largely determined by the Earth's reflex motion [6], this effect is also seen with other astronomical objects such as stars when objects much more massive than TNOs orbit them. For example a sufficiently massive planet orbits a star, this causes the star to "wobble", this is the reflex motion. This can be detected by comparisons of the object being orbited against and its background. It is this reflex motion, when applied to smaller bodies and the Earth affected by a TNO, that provides sets of points with which a solution for the transient objects orbit can be formed. In the search for TNOs a triplet of points is used with the fit_radec algorithm produced by Bernstein and Khushalani [3]. This orbit solution can then be used across all DES observation seasons and even other surveys allowing for verification of the object and it's orbit.

Chapter 3

Method of making images with DES data

In this section I will discuss how I produced a set of code in order to create thumbnail images of a given set of co-ordinates. *N.B. I will be writing this section with a very meticulous approach as my goal is for it to be used as a resource for any students working with my code in the future.*
Two main tasks were set out, initially data was supplied by Professor David Gerdes to cross reference known positions of various TNOs between DES data and other surveys from which single epoch images have already been produced with the exposures containing the TNOs already being known. The second task was to create single epoch images from the same set of data but in different epochs, so that evidence of TNO orbit can be shown.

As previously mentioned a technique for co-added image creation already exists, created by Dr.E.Sheldon (Brookhaven National Laboratory, New York) the `get_raw_images` method, I have used this method myself to produce images of galaxy clusters detected by the SPT survey, Bleem et al. [4], an example can be seen in Figure 3.1 on page 8.

3.1 Explanation of toolkit choices

Before I begin to discuss the method of data retrieval, manipulation and processing it is important to briefly discuss my reasoning behind my choice of programming language, packages and libraries, software, and operating system ('OS') used.

3.1.1 Operating system

Whilst many undergraduate students, whom I need to consider for future work, will be more comfortable with a Windows based OS, there appears stronger presence of Unix based and

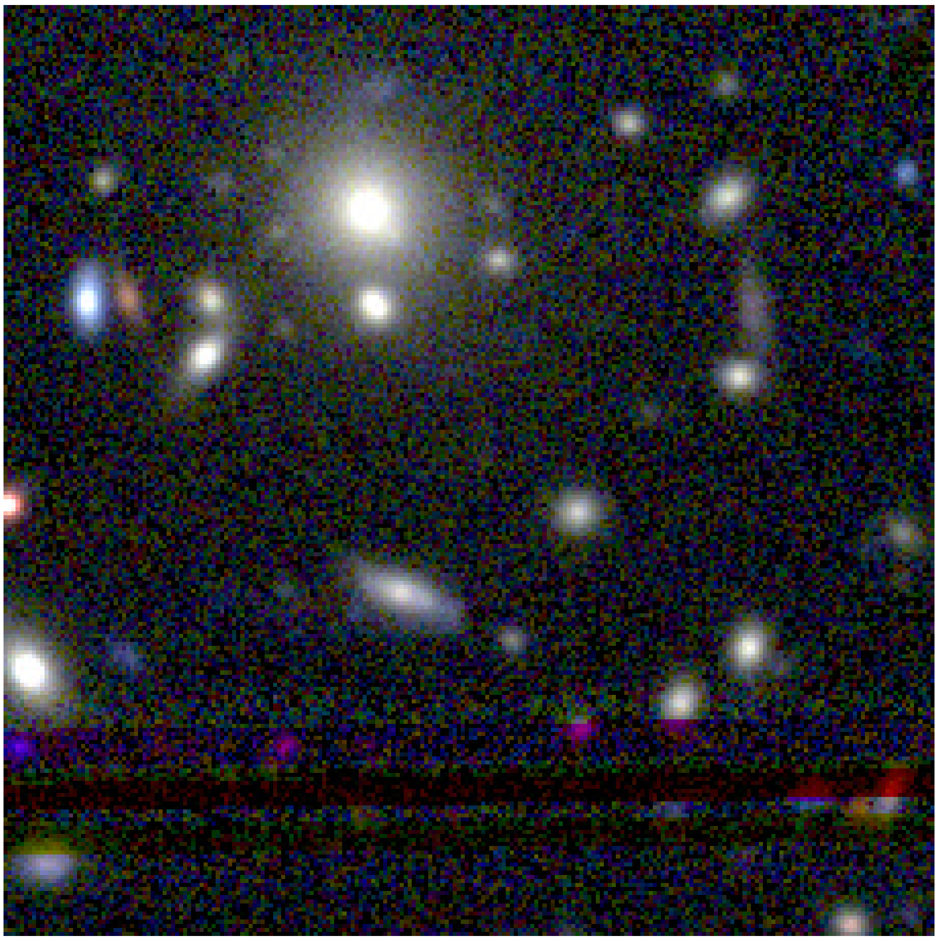


Fig. 3.1 SPT-CL J0405-4916 [4], visualised using *get_raw_images* created by Dr.E.Sheldon, using DES data

Unix-like OSs within the scientific community along with most, if not all, of these OSs being open source. Furthermore, the Apollo high powered computing cluster at Sussex University is based on Scientific Linux. It was important, therefore, to choose a program compatible to a Unix-like environment.

3.1.2 Programming language

Python was a clear choice of programming language to prototype and develop the necessary scripts for this project. This is due to it being an incredibly accessible language for both beginners and advanced programmers, coupled with the fact it is a popular choice amongst scientists. Furthermore a large number of libraries such as `astropy` (https://pypi.python.org/pypi/astropy) are incredibly useful when taking future work with my code into consideration. More importantly the python command line interpreter used to interface with DESDM, called EasyAccess, can be called upon as module by Python and has an excellent API to work with. Specifically this project uses various versions of Python 2.7, Python 3 was avoided due to several packages not being available or having stability issues with it.

3.1.3 Programs

The DES database stores the .fits files in compressed versions, giving it a .fits.fz extension, as storing the vast amount of data would even more storage than the sizeable amount that is dedicated to the DES. In order to unpack these compressed .fits.fz files a program called `funpack` is used (https://heasarc.gsfc.nasa.gov/fitsio/fpack/) and was created by R.Seaman (NOAO) et al. [12] .

3.1.4 Packages and libraries

Due to the nature of this project a fair amount of flexible image transport system (FITS) files had to be manipulated in order to produce useful images. To do this I chose to use Montage (http://montage.ipac.caltech.edu/) as it came with incredibly useful modules such as `mSubCubeCut()` that allowed me to cut raw .fits files down to a reasonable size so that the relevant co-ordinates could be viewed easily.

3.1.5 Visualisation

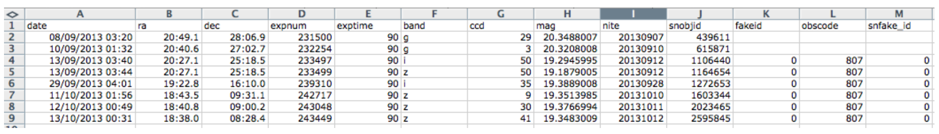
In order to view the .fits files representing the individual ccds it was necessary to use SAOImage DS9(http://ds9.si.edu/site/Home.html). This application can be called upon in the command line and therefore can also be incorporated into my code, allowing for me or any other user to preview data as it is being processed if they so wish. However for the time being this feature has been removed in order to speed up the processing of several different datasets. SAOImage DS9 was used also to test the best possible region sizes, scaling, and colourmaps. These aspects will be discussed later on in this chapter.

3.1.6 Image file format

My reasoning for choosing the `JPEG` file format is due to its widespread popularity amongst programs and the quality can be scaled down in the event that file size becomes an issue. However, the ideal format for these images considering they use a grey colour map is the Portable Network Graphics (PNG) format as it is better suited to the scientific environment due to its lossless compression. This was my initial choice of format, yet several technical issues were encountered when using the PNG format and with repeated attempts at fixing these issues I decided that it would have to be an issue to be resolved in future work if I was to progress with the project at an acceptable rate.

3.2 Retrieving data from the DES Data Management system (DESDM)

The information I had to work with were .csv files relating to either named or un-named TNOs, this included the number of ccd that they had been detected in, in a given exposure and filter. An example of the data given can be seen in figure 3.2 on page 10. It was my task to create a process that could quickly retrieve data from DESDM based on these values and then create thumbnail images highlighting the object of interest. This would form the first major part of my project, making images of the single epoch data given to me. By being



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	date	ra	dec	expnum	exptime	band	ccd	mag	nite	snobjid	fakeid	obscode	snfake_id
2	08/09/2013 03:20	20:49.1	28:06.9	231500	90	g	29	20.3488007	20130907	439611			
3	10/09/2013 01:32	20:40.6	27:02.7	232254	90	g	3	20.3208008	20130910	615871			
4	13/09/2013 03:40	20:27.1	25:18.5	233497	90	i	50	19.2945995	20130912	1106440	0	807	0
5	13/09/2013 03:44	20:27.1	25:18.5	233499	90	z	50	19.1879005	20130912	1164654	0	807	0
6	29/09/2013 04:01	19:22.8	16:10.0	239310	90	i	35	19.3889008	20130928	1272653	0	807	0
7	11/10/2013 01:56	18:43.5	09:31.1	242717	90	z	9	19.3513985	20131010	1603344	0	807	0
8	12/10/2013 00:49	18:40.8	09:00.2	243048	90	z	30	19.3766994	20131011	2023465	0	807	0
9	13/10/2013 00:31	18:38.0	08:28.4	243449	90	z	41	19.3483009	20131012	2595845	0	807	0

Fig. 3.2 Raw observation data provided by D.Gerdes for the TNO RN43

given the key information mentioned and displayed in figure 3.2 on page 10 ,the task became a case of working out how to use the given data to download the relevant .fits files through the use of urls related to DESDM and making sure that the data was correctly formatted. Formatting the data involved converting the format of the tables from .xlsx to .csv using Microsoft Excel, this is because I personally find it much easier to manipulate comma separated variable (.csv) files when using them in conjunction with scripts. Furthermore the .xlsx format had caused issues with the data by automatically formatting it, for example with the column for ccd number ,'ccd', which is column G in figure 3.2 there is no preceding zero before the value '3'. The significance of this is that the url is structured like so:

https://desar2.cosmology.illinois.edu/DESFiles/new_archive/Archive/OPS/red/ ["RUN"]/red/DECam_00["expnum"]/DECam_00["expnum"]_["ccd"].fits.fz

In the above case this would produce a url of:

https://desar2.cosmology.illinois.edu/DESFiles/new_archive/Archive/OPS/red/ 20130911085123_20130909/red/DECam_00232254/DECam_00232254_3.fits.fz

However, this would cause a 404 error as this page would not exist due to the naming convention of the urls where the `_3.fits.fz` should in fact be `_03.fits.fz`. This url structure is used in both the single and multi-epoch image creating scripts.

The DES run value (RUN in the my code and .csv files) was also not provided in the raw data, for this I had to query DESDM through EasyAccess which involved using SQL queries, specifically in the Oracle friendly syntax as DESDM is an Oracle database primarily. To retrieve the run information I used the query seen in D.1 on page 56.

3.3 The process for making for single epoch images

After the data had been formatted correctly and the additional run information had been obtained the tables could then be run through my `get_single_epoch.py` script, which can be viewed in full in A on page 39.

It is recommended that the rest of this section is read in tandem with the script as I will be discussing my code in the order that it is written and, unless stated otherwise, line references are referring to the lines within this script.

3.3.1 Front matter: modules and packages

To begin with we import the key modules and parts of modules, as seen on lines 2-4 of A on page 39. The module `os` allows for us to run command line prompts from the script itself and the `sys` module is used to pass user determined arguments through the command line to the script. When needing to manipulate .csv files the module `csv` is also an obvious choice as it provides the means for the reading and writing of data of that format. The library `requests` is an incredibly powerful yet simple HTTP library, and is used in my code to connect to the DES database to retrieve .fits files. Lastly on line 2 of A we call on the `shutil` module, this gives the script the capability to copy and move files and directories.

On line 3 easyaccess is imported, as `ea`, this is the Python interpreter used to connect to the DESDM database. However it is only use currently by one function, to be discussed, which is in place to be used for future work to streamline the process of obtaining and appending the run information for exposures where the exposure number is already known.

Lastly, on line 4, `defaultdict` is imported and this module allows for dictionaries to be created in which is used in the generic function `ReadFromCSV()`.

3.3.2 Explanation of the function: `ReadFromCSV()`

This function is a generic method for reading .csv tables and therefore allowing other functions to manipulate the appropriate files, it can be found across lines 37-46 of the code in A. In this case `ReadFromCSV()` takes a single called `fname`, this comes from the user defined filename for the .csv, this is achieved through the use of two separate arguments, `baseDir` (argument 1) and `csvFile` (argument 2), found on lines 140 and 142 respectively. These are then used by the main program on line 144 in conjunction with the function `ReadFromCSV()` to give the variable `CandInfo` that is then used extensively with the other defined functions .

3.3.3 Explanation of the function: `MakeFolders()`

This function, located on lines 49-56, simply creates folders in the directory specified by the `baseDir` and `Candcsv` variables that are set by arguments 1 and 2 respectively when the script is run. Within the function exists a `for in range` loop, lines 38-43 of A, this loop makes use of `ReadFromCSV()` through the `Candcsv` variable to read each exposure number, "expnum". The loop then defines the `newpath` variable by combining `baseDir` and the exposure number being currently read. An *`if not`* statement is nested within the *`for`* loop, this calls on the `os.path.exists()` to see if `newPath` doesn't exist and if true it calls on `os.makedirs()` to create a directory specified by `newPath`. The following nested `else` statement in the `for` loop will notify the user if the folder already exists.

In summary, the purpose of the `for` loop is to prevent new directories from being made and overwriting any data that already exists, this will then mean the next function `DownloadFits()` would re-download data that already exists as well as any additional data.

3.3.4 Explanation of the function: `DownloadFits()`

Located on lines 58-86, this function performs the vitally important step of downloading the necessary .fits files to then manipulate and finally produce images.

Again it only has two parameters, `baseDir` and `CandInfo` , that have been previously discussed (*N.B.* `Candcsv` *and* `CandInfo` *are essentially the same parameter, the difference is purely a remanent of development that will be removed for future work*).

The function begins by defining `session` variable that takes the value of the `requests.Session()` function. This is followed by the variables required to access the DES database, `DESusername` and `DESpassword`, and these must be defined within the script currently by the user. The last variable `base` defines the 'base' part of the url used to download data from, this is to shorten the url written within the `for` loop and to allow for easy modification in any future work should the url structure need to be changed.

Within the `for` loop, which uses the same `in range` premise as that found in the `MakeFolders()` function along with many other functions in the script. The `for` loop changes the directory created by `MakeFolders()` using `os.chdir`. The url from which the .fits.fz file is to be downloaded from is defined. This is achieved by using the `ReadfromCSV()` function under the guise of `CandInfo` to call on the dictionary of values associated to this particular exposure, as can be seen on lines 68-70, and this is where the ccd value being correctly formatted is important as discussed in the section **??** on page **??**.

Now the variable `request` is defined and this contains the `session` variable that then makes use of the `get` function from the `requests` module. This will be used to start a session with the DES database with the given `url` parameter and the `auth` parameter makes use of the previously stored `DESusername` and `DESpassword` variables. Whilst not particularly necessary for these .fits.fz files the `stream` parameter is set to `True` as it stops the file from being held on memory until it is saved as a file. By having the files memory allocation reduced whilst downloading it will mean that with larger files that the program is more efficient with memory.

I have then nested `if` and `else` statements that alert the user as to whether or not the download procedure is successfully proceeding,lines 75-79. As a part of the `else` statement that begins on line 77, after displaying the message to show that the file has begun to download the function then writes the .fits.fz file to the correct directory using the `shutil.copyfileobj()` function. Then two command line instructions are carried out using `os.system()`, the first on line 84 uses `funpack` to unpack the .fits.fz file to the working directory. This is followed up by the second command, line 85-86, that removes the .fits.fz file as to reduce a now redundant file as it has been decompressed by `funpack`.

This function now provides the means to download all the relevant data, so that the data processing block of functions can begin in the main program.

3.3.5 Explanation of the function: `subCubeCut()`

The `subcubeCut()` function, lines 88-98, performs the cutting of the .fits files about the necessary co-ordinates using the Montage packages `mSubCube` command line function. First of all the function changes the working directory to that of the necessary exposure with `os.chdir`. The variables `x_y_param` and `cmd` are defined, `x_y_param` defines the square dimension to be cut and in this case I have opted for 0.06667 degrees which is approximation of 4 arcminutes to produce a 4x4 arcmin square cutout.

We then have the `cmd` variable that forms the string to be passed as a command. First of all it calls on the `mSubCube` command, followed by the name of the .fits file to be cut. It then adds second and third arguments that are the right ascension (RA) and declination (DEC) of the object in degrees for the command to cut around. As read from the relevant table of data. These co-ordinates must be in degrees, so it is recommended that a package such as `astropy` is used to convert co-ordinates of other systems (e.g. sexagesimal). The fourth and last argument passed is the `x_y_param` variable. Finally on line 98 the function executes the command via the `os.systemt()` function.

3.3.6 Explanation of the function: `WriteRegionFile()`

This function creates a region file, .reg, to be used within the `Circles()` function that in turn uses ds9 to circle the object of interest. The parameters of the circle; position, radius, and colour are determined by this function, `WriteRegionFile()`.

`WriteRegionFile()` exists between lines 102-118, after a change to the correct working directory the first variable `regname` is defined and this uses `open()` to create the .reg file for the relevant exposure.

`x_circle` and `y_circle` variables are then defined, these determine the x and y positions of the centre of the circle, RA and DEC, respectively and these positions are determined by the co-ordinates of the object . Again these are in degrees, as is the `radius` variable that sets the radius of circle region to be created to 0.5 degrees, this was worked out to be the best value to use for the given data as it created a circle that was big enough to see but not big enough to cover other objects that are not of interest. The `contents` variable brings together all the previously mentioned variables as a string to be written to the .reg file using `regname.write(contents)` on line 122. Lastly the file is closed using `regname.close()` as to avoid memory errors and to finish the process of file creation so that the function can then iterate to the next .reg file to be written.

3.3.7 Explanation of the function: `Circles()`

This function, on lines 120-137,uses the region file created by `WriteRegionFile()` to place the region file 'over' the relevant .fits file by executing a SAOImage DS9, `ds9` command through `os.system()`. Using this command allows us to set variables to pass arguments to `ds9`, so that the function:

- Bins the pixels by a factor of three, see line 126, with the `bin_value` variable. This means that for every three pixels a single composite pixel replaces them and it's values determined by the sum of its constituents . This is to reduce the amount of unwanted noise that is not reduced by adjustments to the scaling.

- Changes the colour map and scale. `map_scale` defines the colour map to be used to grey, with the argument of `-cmap grey`. The scale is then set to a logarithmic scale, `log`, using the zmax algorithm by including `-zmax`. After manually adjusting the scale parameters and using various type of scaling such as `linear` and `square root` with different scaling algorithms I found that across multiple exposures that using a logarithmic scale in combination with the zmax algorithm is the most consistent in producing useful images. The zmax algorithm was created using the Image Analysis and Reduction Facility (IRAF) algorithm `zscale` and the maximum pixel value, which by using the grey colour map is just the brightness of the pixel. There is very little in the way of documentation of the zscale algorithm, however the Space Telescope Science Institute (STScI) has published source code for the algorithm publicly as compact webpage [9].

  The logarithmic scale is achieved with the equation,

  $$y = \frac{log(ax+1)}{log(a)} \tag{3.1}$$

  In this equation x is takes a value ranging from 0 to 1 based on the initial pixel value and a is the parameter that determines the distribution of colour on the scale. y is the resulting pixel value of the algorithm[11].

- Corrects the zoom on the .fits file, with the variable `zoom` and this due to the scale on the x-axis altering the image due to aesthetic reasons.

- Exports the .fits file as a .jpeg as defined in `export_img` variable, note that this is done at an image quality of 100, as currently the file size is not an issue and this value can be changed at any time on a scale of 0-100.

3.3.8 Explanation of the function: `mViewerImage()`

`mViewerImage()` is currently not in use, it is commented out on line 195, and acts as an alternative, provided by the Montage package, to the previously mentioned `Circles()`

method. I've included it as an example of how the command can be used if future users wish to use Montage for setting the correct scaling and colour mapping parameters instead.

3.3.9 Explanation of the function: `CopyImgs()`

This simple function copies the images produced by `Circles()` to the directory `Images` that is created in the main program block. This directory specific to the object of interest that contains all relevant images from all exposures, this will then be used as the main directory from which the images are sourced by the necessary HTML page. The function can be found on lines 149-156.

3.3.10 The main program block

Now that the functions have been discussed at length, I will now briefly talk about the program itself. The main program block begins on line 159 and terminates at line 201. By setting the main program block as an `if __name__ == "__main__":` statement it allows for other programs and scripts to call on it in future work with ease by changing a few lines of code where necessary. Furthermore it allows for a closed environment in which to lay out the logical steps that the program will take when executed. To make this process easier to follow I will now numerically list the steps taken by the block:

1. Lines 162-166 are arguments that are provided by the users input on the command line. The first argument `baseDir` specifies the top-level directory to work in, then follows the `csvFile` argument that defines the name of the .csv file to be analysed for the propoerties of each exposure allowing for other functions to iterate through all of the exposures. The last argument is `needToPreProcess` which will only function if `true` or `false`.

2. `CandInfo` on line 167 has previously been discussed and combines arguments `baseDir` and `csvFile` to give the function `ReadFromCSV()` a .csv file with its path to read.

3. If the `needToPreProcess` argument has the value `true` then the program will create the necessary directories and download the raw .fits.fz files and unpack them with `MakeFolders()` and `DownloadFits()` respectively. f `os.system()` is used to create a directory for the images. After this the .fits files are manipulated and images are created, with the functions already being discussed previously. Feedback is provided to let the user know if this has been successful or if an error has occurred.

However, if `needToPreProcess` is `false` then the program will begin to process the data. The previously discussed functions; `WriteRegionFile()`, `subCubeCut()`, `Circles()`, and `CopyImgs()` are used between lines 190 and 193 to achieve this.

4. Finally the program provides the user with more feedback on lines 197-201, notifying the user that no new data was downloaded and the `false` argument was correctly passed. Alternatively the program will inform the user that an error has occurred.

An example of the command entered into the command line to execute this program with pre-processing would be:

```
python single_epoch.py the/directory/you're/working/from/
csv_containing_object_information.csv true
```

3.4 The code used for multi-epoch image making)

In order to produce images with the same co-ordinates across multiple epochs, the previously discussed single epoch method had to be altered somewhat. For me to tackle this problem I have broken down the code into two separate scripts, with one for retrieving the required values to create a new table and therefore .csv file. The other is an alteration of the code used to retrieve single epoch .fits.fz files and process the images A on page 39.

3.4.1 Retrieving information from DESDM for .csv file creation

The first script to discuss is that used to retrieve the necessary information for other exposures that cover the same co-ordinates we are interested in. DESDM is accessed via the previously discussed easyaccess client and it is imported as `ea` on line 2 of B.1 on page 44. For the rest of this subsection and any sub-subsections, unless stated explicitly, it is assumed we are referencing the code found in Appendix B.1 on 44.

3.4.2 Front matter: modules, packages, and easyaccess

This script uses the same packages and modules as the previously discussed code for creating the single epoch images,3.3.1 on page 11 . The only exception is easyaccess the DESDM Python Command Line Interpreter, this provides Python with an API and therefore functions to connect and access DESDM with and has been previously mentioned. These functions will be discussed individually in due course.

3.4.3 Explanation of functions; `ReadFromCSV()`, `CreateOtherEpochDirectory()`, and `CreateOtherEpochSubDirectory()`

The `ReadFromCSV()` is identical to the function previously discussed in 3.3.2 on 12.

The first new function encountered is `CreateOtherEpochDirectory()` on lines 17-22, creates a new directory called `Other_Epochs` within the directory specified by the argument `baseDir` for the exposures from other epochs relating to the co-ordinates of interest.

Then sub-directories are created within the `Other_Epochs` directory by `CreateOtherEpochSubDirectory()`. Both of these directory creating functions use nested `if not` statements to check if the path to the directory already exists and if not creates the directory and both will provide the user with feedback to notify them the appropriate feedback if the path already exists, which may need to be removed.

3.4.4 Explanation of the function: `QueryEasyAccess()`

This function that spans lines 35-76 is fairly large and as such I will use a list to aid in its explaination in order to make it clearer and easier for the reader to follow.

- The function takes four input parameters; `baseDir`,`csvFile`, `CandInfo`, and `CandName`. The parameters; `baseDir`, `csvFile`, `CandInfo` are identical to their previously discussed counterparts in **??**] on 39. A new argument is `CandName` that is the third argument for this code.

- The variables `RA_TOLERANCE` and `DEC_TOLERANCE` are used to calculate the range of right ascension and declination the query should search for, I have used approximations for this value by finding the dimensions of the ccds already used in the single epoch image creation. To avoid any edges of ccds and false positives the tolerances fall well within the boundaries of every ccd. This obviously poses the problem of missing parts of exposures, which needs to be overcome with future work currently.

- `RA_MIN`,`RA_MAX`, `DEC_MIN`, and `DEC_MAX` use the previously mentioned tolerance variables to calculate the range based up the co-ordinates provide for the search and are found across lines 38-54. These values will all be calculated in degrees as this is the standard format for RA and DEC in the DESDM tables. As they will be used as a part of an Oracle SQL query they must be in a string format to match the rest of the query and at the same time we need the resulting numerical value to be a floating point number, therefore the structure of `str(float())` is used to ensure this.

- I have then defined two variables that make use of the `easyaccess` API, the first is `connection` and this invokes the `ea.connect()` function (N.B. easyaccess is imported as `ea`) to connect with the DESDM Oracle database. No password or username is required as these values will already be held by the easyaccess interpreter. The variable `cursor` is then created to shorten the function within `ea.connect()` called `cursor()`. A cursor ,in the context of databases, allows for a query result to be manipulated row by row, which is necessary here as we need to create a .csv from the results.

- The key part to this function is the `query` variable located on lines 58-61, this is what will be passed by easyaccess to DESDM and be used to retrieve the necessary information. The table `felipe.ME_IMAGES_Y3A1_FINALCUT` contains all exposure information from the Scientific Verification (SV) period until and including the Y3A1 observations. This is the most complete table that was available to me at the time of processing the images. By changing this part of the variable any other table of interest can be accessed, however the user must be sure to check that the column values such as `RA_CENT` and `DEC_CENT` exist in the other tables. Otherwise alterations to the code may need to be made where appropriate. `RA_CENT` and `DEC_CENT` are the co-ordinates of the centre of a given ccd, by searching for these values within the ranges using the `between` statement means that a field around the co-ordinates of interest is produced. The ranges are slightly smaller than a ccd, as previously discussed, so ideally future work would involve increasing the coverage of this area. This search is also currently restricted to the filter (in this case it is called `BAND`) of the initial exposure in which the co-ordinates were first found in. This is to make it more streamlined when trying to compare between datasets. However, I am confident this limitation can be removed without causing an major issues.

- The query is then printed on line 63 for the user to inspect to make sure the correct information is being passed to DESDM.

- The `connection` variable is then used to call on the `easyaccess` function `connection.query_and_save()` to save the query result as table in the .csv format to the relvant directory. The functions on line 71 `load_table()` and line 75 `connection.mytables()` uploads the table to your private DESDM space, I've included this as a means to backup results. However, for any large amounts of data it is suggested you comment out lines 71 and 75 to prevent running out of storage space.

- `connection.close()` on line 76 terminates the connection to DESDM.

3.4.5 The main program block

This program block simply defines the three arguments; `baseDir`, `csvFile`, and `CandName` used by the previously mentioned functions between lines 97-104. On line 95 the`CandInfo` is also defined again in the same manner that was carried out in the single epoch image making method.

3.5 Code for creating images from the multi-epoch data

Fortunately this code, B.2 on page 46, mostly mirrors that created for single epoch image creation. However I have included it for completeness, as part of my planned future work on the code is to merge the querying and image creation together into a single package. I have pre-emptively imported `easyaccess` to account for this merging.

However a key difference can be found on lines 74-76, by querying DESDM I have been able to obtain the exact path within the .fits database for each exposure and therefore with this modification to the `url` variable within the DownloadFits() function it is possible to download any .fits.fz file from any observation season. This is achieved by introducing data obtained from `CandInfo["PATH"]` within the .csv.

Lastly on line 171 the `Expnum` variable definied by argument 2 replaces `CandName` as this program is run based on the exposure, this is because I have run the code on an exposure by exposure basis (of the single epoch exposures). This is purely a remnant of me testing the process and it can be run for the entire single epoch database of the candidate. For example; candidate 'TEST46' is associated to `test46.csv`, within it is an exposure '123456' and '654321', instead of passing these exposures separately as arguments simply passing `test46` as `Expnum` will suffice. This confusion will be removed with future work.

3.6 Creation of HTML pages with Python

The final piece of code I have created is used to streamline the process of creating HTML pages for each candidate. This code can be found in Appendix C on page 52 and it will be assumed I am referring to this code for the rest of this section unless stated otherwise.

3.6.1 Front matter and `ReadFromCSV()`

The modules and packages imported on lines 1 and 2 have been previously discussed in 3.3.1 on page 11. The function `ReadFromCSV()` has also been discussed previously in 3.3.2 on page 12.

3.6.2 Explanation of the function:`MoveImg()`

This function begins on line 15 and terminates on line 20.Two parameters we are already familiar with from previous discussion, `baseDir` and `CandInfo`. The third parameter is `ImageDir` that specifies the path to the directory that contains the images created by the image making processes. This parameter is created simply by combining `baseDir` and the string `Images`.

Two variables are then defined, `ImgIn` contains the images path that needs to be moved and `ImgOut` specifices the new path for the image, which will be the folder called 'Images' specific to that exposure and the other corresponding exposures for a particular table or candidate. Moving the images is simply achieved by utilising the `shutil` function `copyfile()`, in conjunction with the `ImgIn` and `ImgOut` variables as the functions parameters.

3.6.3 Explanation of the function:`makeHomepage()`

Found on lines 22-29 this function is a remanent of development and is not actually invoked in the main program block. The main use of this function was to prototype the homepage for my webspace (http://astronomy.sussex.ac.uk/ cc359/html/homepage.html) and to keep a reference for a basic homepage header format. As such it can be ignored for the remainder of this discussion.

3.6.4 Explanation of the function:`makeHTML()`

This is the main function of the program as it writes the necessary contents for the .html file, and it exists on lines 32-60. The function takes five parameters, all of which are variables determined by arguments in the main program block; `baseDir`,`CandName`,`htmlName` are all determined directly by arguments. `CandInfo` is the variable containing the data from the respective csv file as achieved by `ReadFromCSV()` , this leaves `ImageDir` that is created by the addition of `'Images'` to the `baseDir` variable and it is also a string.

The first variable encountered within the function is on line 33, `html` contains the information to create the blank and writeable .html file using the `baseDir` and `htmlName` variables. This is followed by the variable `Header` that spans lines 34-43, this variable contains all of the information relating to the header of the HTML file. The header in these HTML files contains the information corresponding to the; meta information, style of the page, heading, and how the table environment within which the data is presented.

The function then writes the header file to the appropriate HTML file with `html.write(Header)` on line 44.

This is followed by a `for` loop that creates several variables across lines 46-54 that are then combined to form the `Line` variable which contains all the necessary information for each row in the table of data. The `html.write()`function is used to write this line to the relevant .html file on line 56.
As an exception these variables have been manually altered for the `RA` and `DEC` for the M51a proof of concept page due to a difference in co-ordinate system, the proof of concept will be discussed in due course.

Lastly the footer for the HTML page is created that closes off the body and the table of the page using the `Footer` variable on line 58, it is written to the file with `html.write()` on line 59 and to end the function the .html file is closed with `html.close()` on line 61.

3.6.5 Discussion of the main program block

For the HTML page creating code the main program block is relatively short compared to the previously discussed programs. It spans lines 67-86, the four variables corresponding to the user provided arguments are placed first on lines 70-76. On lines 79 and 80 the `CandInfo` and `ImageDir` variables are defined and then the function `os.system()` is used to create the image directory in the current working directory on line 81 using the `ImageDir` variable as the argument to the `mkdir` command.

Lines 84 and 86 the functions `MoveImg()` and `makeHTML()` are invoked concluding the main program block.

3.7 Summary of Chapter 3: Method of making images with DES data

In this chapter I have discussed all the relevant programs that I have created in order to produce and present both single epoch and multi-epoch images.I have also discussed the packages and modules that I have used that were created by other parties. Additionally, justifications have also been made for aspects such as programming language and operating system choices.

Now, I will present and discuss the results of using my code in conjunction with the DES data in Chapter 4.

Chapter 4

Results of image processing

In this chapter I will discuss and display the results of using my code to retrieve data and process it into viewable images. These images have then been used to create HTML webpages, this in order to make my data accessible for others to view.

In order to test my code for the multi-epoch problem I used a proof of concept candidate, as was advised by my supervisor Dr. K.Romer. I will be using Messier 51a [1], also known as the Whirlpool galaxy, for my proof of concept candidate. This will be discussed after displaying my results from the single epoch data processing.

4.1 Single epoch data processing results

By being given most of the necessary information such as ccd number and exposure number the task of gaining single epoch data would only require me to retrieve the .fits file data, as discussed in 3.2 beginning on page 10. newline For this section I will be presenting the images associated to the named TNO called RN43. However the first figure **??** on page 25 is an example of how a ccd appears when opened as a raw.fits file in a default linear scaling. After the correct scaling and colour map is applied objects begin to become visible as can be seen in figure **??** on 25. For the purposes of this dissertation figures **??** and **??** have been created manually and as such the scaling may not be exact to the method used to create the proof of concept images.

As you can see across Figures 4.3-4.10 on pages 26-29 my code has successfully been able to download the raw .fits files from the DES database, cut the images down to an approximate 4x4 arcmin area and then apply a grey colour map with the zmax scaling algorithm, the pixel brightness scale can be seen underneath. I decided to include this scale in order to display the variations between exposures to justify the use of the zmax algorithm instead of

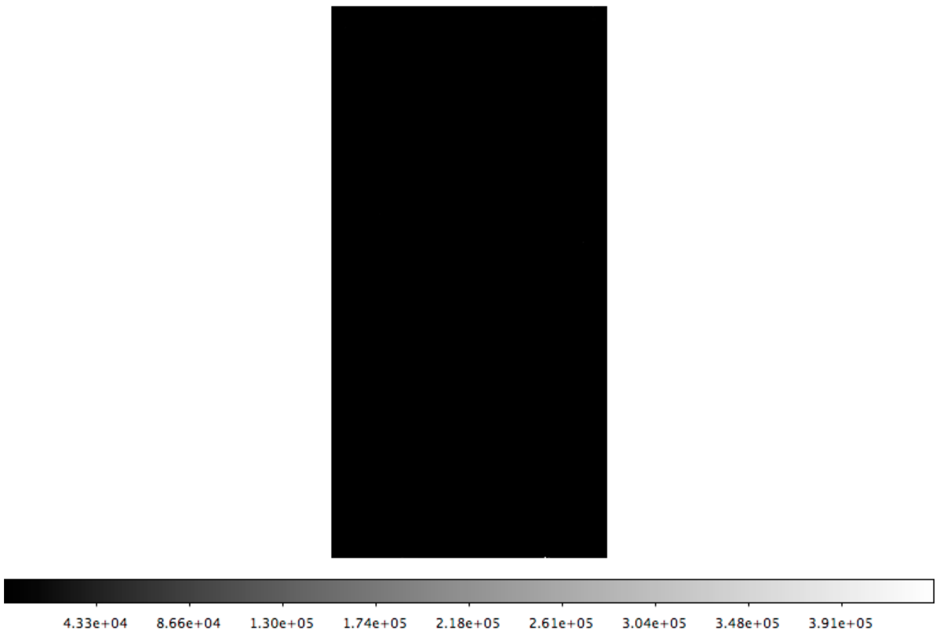


Fig. 4.1 The raw .fits file as seen in SAOImage DS9 of exposure 361655 relating to M51a

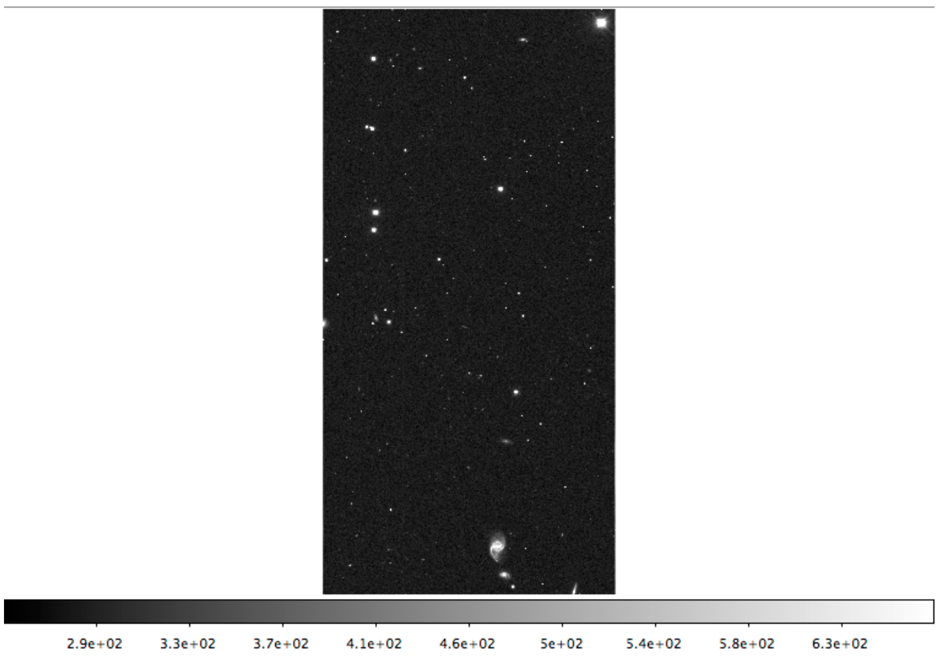


Fig. 4.2 The same exposure from figure **??** with scaling and colour map applied, M51a is visible at the bottom centre-right of the ccd

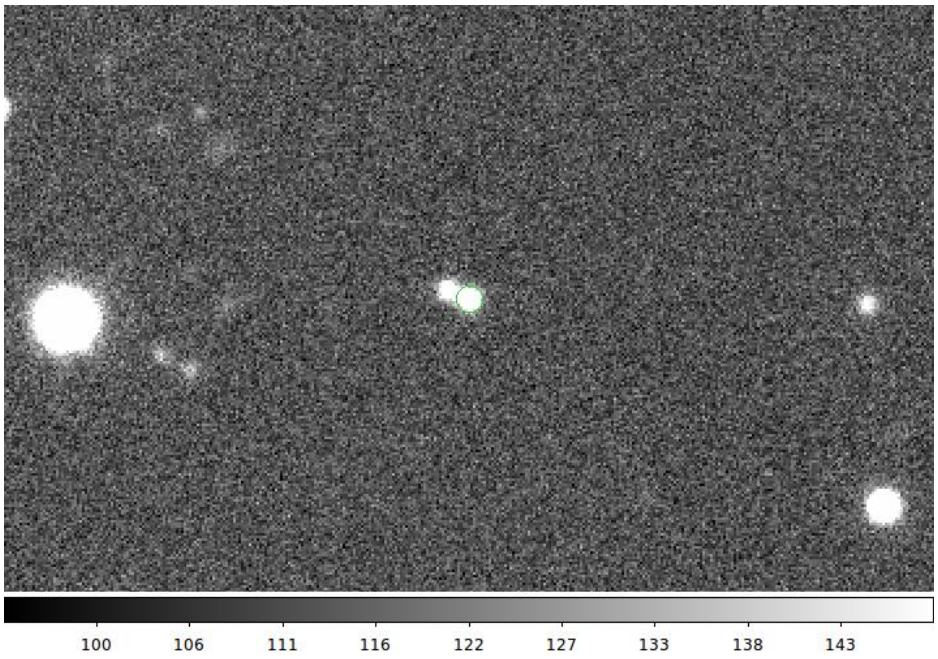


Fig. 4.3 RN43 as seen in exposure 231500 on ccd 29 of DECam

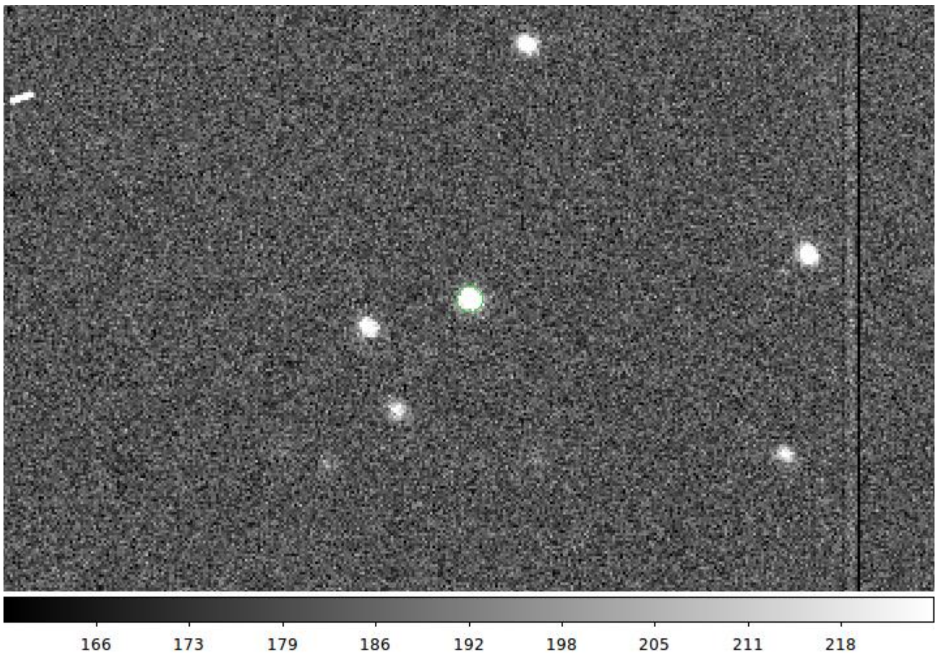


Fig. 4.4 RN43 as seen in exposure 232254 on ccd 3 of DECam

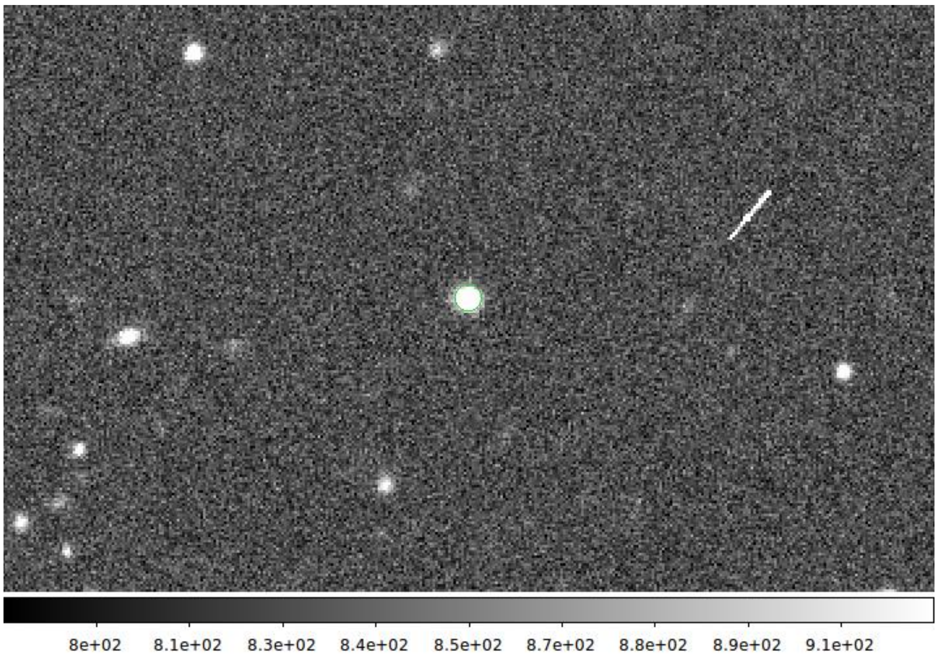


Fig. 4.5 RN43 as seen in exposure 233497 on ccd 50 of DECam

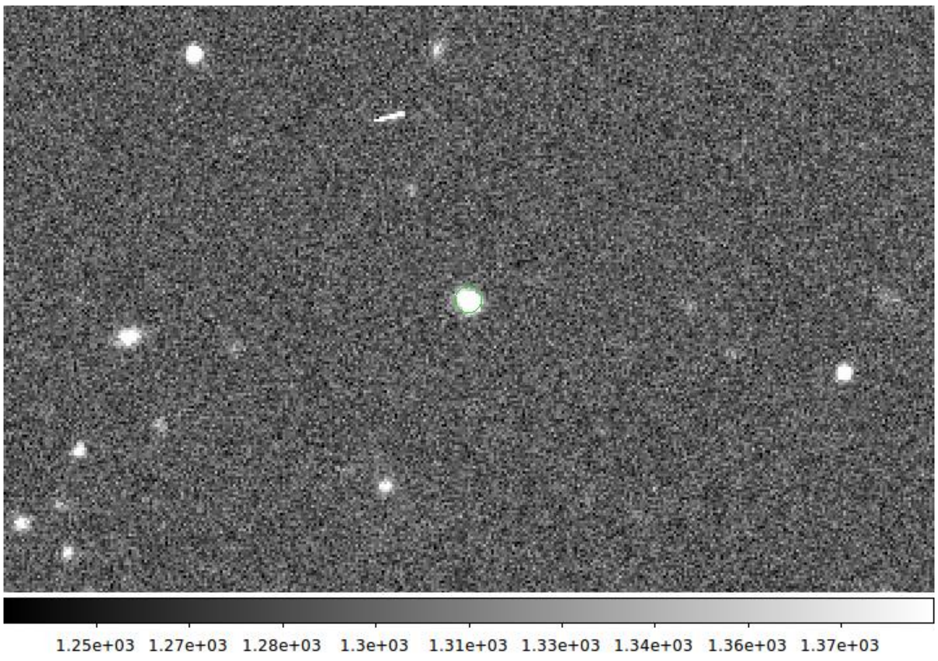


Fig. 4.6 RN43 as seen in exposure 233499 on ccd 50 of DECam

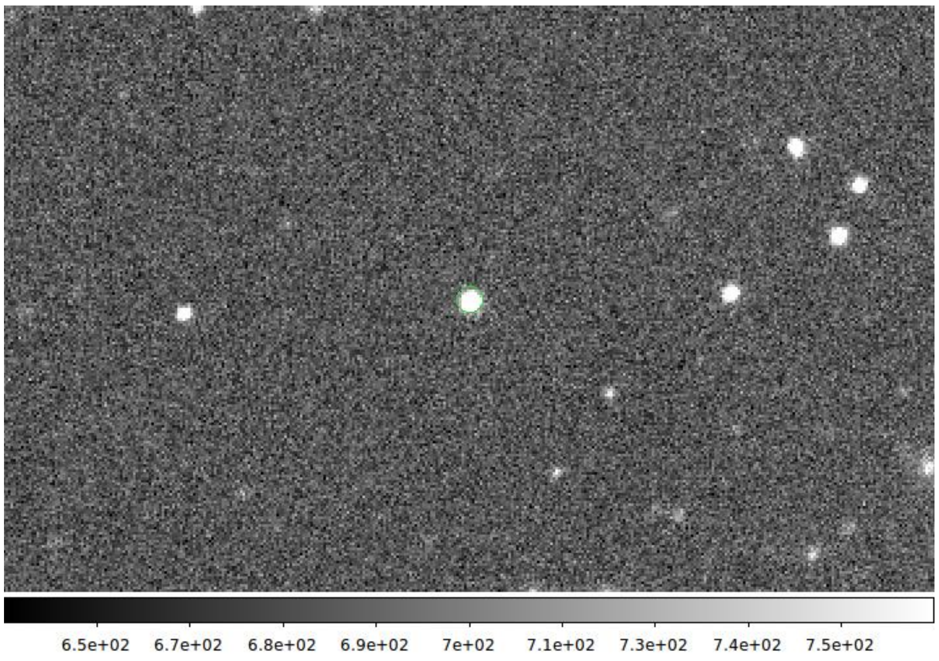


Fig. 4.7 RN43 as seen in exposure 239310 on ccd 35 of DECam

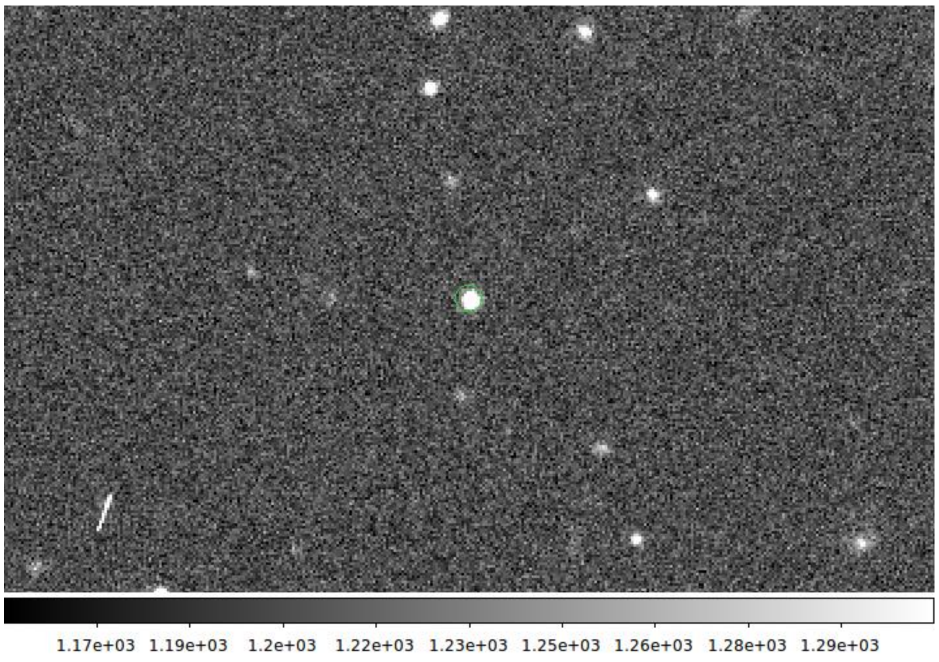


Fig. 4.8 RN43 as seen in exposure 242717 on ccd 9 of DECam

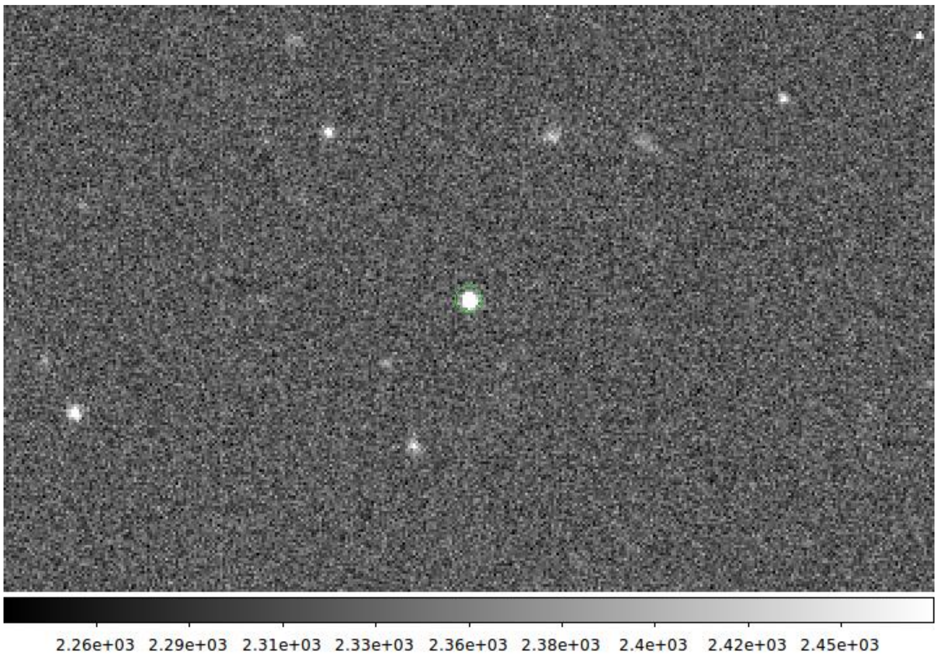


Fig. 4.9 RN43 as seen in exposure 243048 on ccd 30 of DECam

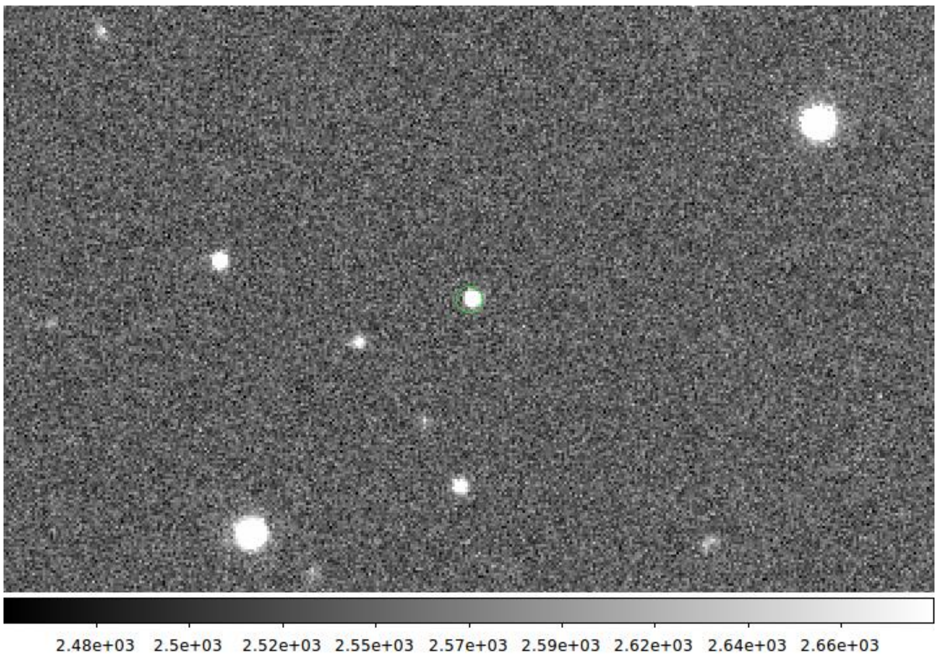


Fig. 4.10 RN43 as seen in exposure 243449 on ccd 41 of DECam

using pre-defined limits of pixel brightness for all exposures.

4.1.1 Pixel brightness: A brief discussion

It is evident there is a wide range of pixel brightnesses just in this small selection of exposures and this is due to the different filters being used. Figures 4.3 and 4.4 on page 26 are exposures taken in the g-filter which corresponds to green wavelength light of approximately 510 nm. As expected their pixel brightness scale is of a factor 10^2 and even then the values sit around the low 100's and 200's.

Whereas figures 4.5 and 4.7 on pages 27 and 28 respectively are in the i-filter that is sensitive to the much higher wavelength 806nm of the infrared section of the spectrum. The remaining images are all exposures that were taken in the z-filter of DECam and is sensitive to the slightly higher 900nm wavelength section of infrared.

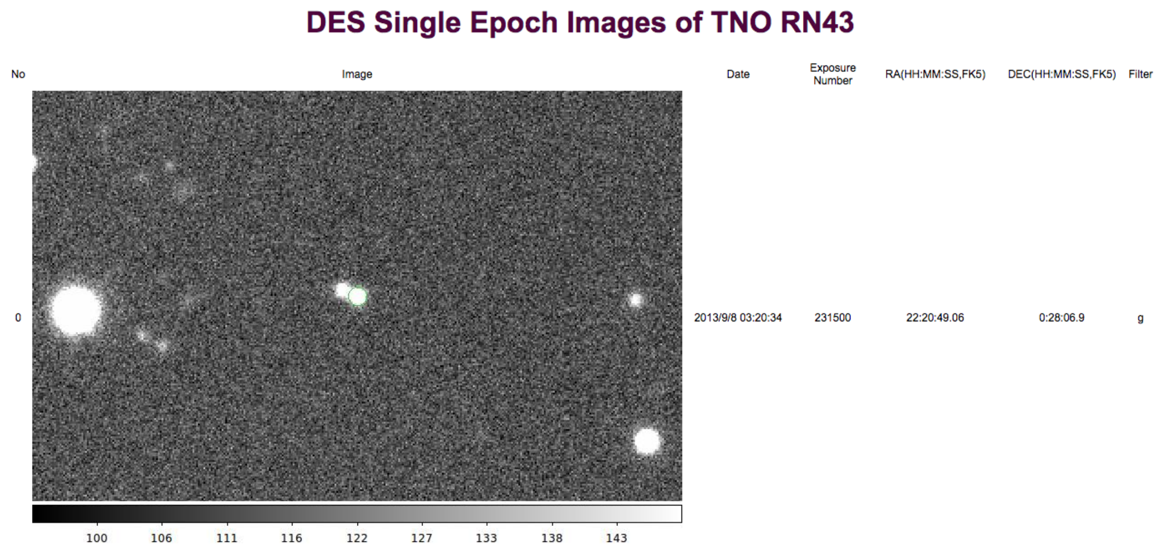
These last two filters, due to their sensitivity, capture much lower frequency light than the g-filter and therefore radiation emitting objects further away are more easily detected. However due to the relative close proximity of TNOs different filters have not appeared to hinder the observation of these objects.

4.1.2 Results of the HTML generating program

By using the code discussed in section 3.6 which is also displayed in the appendix C.1 I have also successfully produced webpages to display my results. Figure 4.11 is an example of the exposure shown in figure 4.3 on 26. The rest of this page can be found at: `http://astronomy.sussex.ac.uk/~cc359/html/Named/RN43/RN43.html`.

4.2 Proof of concept candidate: Messier 51a (M51a)

The reason for choosing M51a as my proof of concept candidate is due to it being a very bright galaxy, indicating that it would be easy to detect and therefore if my code was interpreting the given co-ordinates correctly and then retrieving the correct exposure and ccd without any other input than co-ordinates. This would mean my code used to create single epoch images would now be viable for creating images with the correct co-ordinates across multiple epochs when used in conjunction with my exposure information retrieval code, as discussed in section 3.4.1 beginning on page 17. Due to the difference brightness of M51a compared to TNOs, the `linear` scaling was used instead of `log`.
The following fig 4.12 is the final image created for the proof of concept candidate M51a.



DES Single Epoch Images of TNO RN43

No	Image	Date	Exposure Number	RA(HH:MM:SS,FK5)	DEC(HH:MM:SS,FK5)	Filter
0		2013/9/8 03:20:34	231500	22:20:49.06	0:28:06.9	g

Fig. 4.11 RN43 in exposure 231500 as displayed as a part of a HTML web page created by my .html file generating program

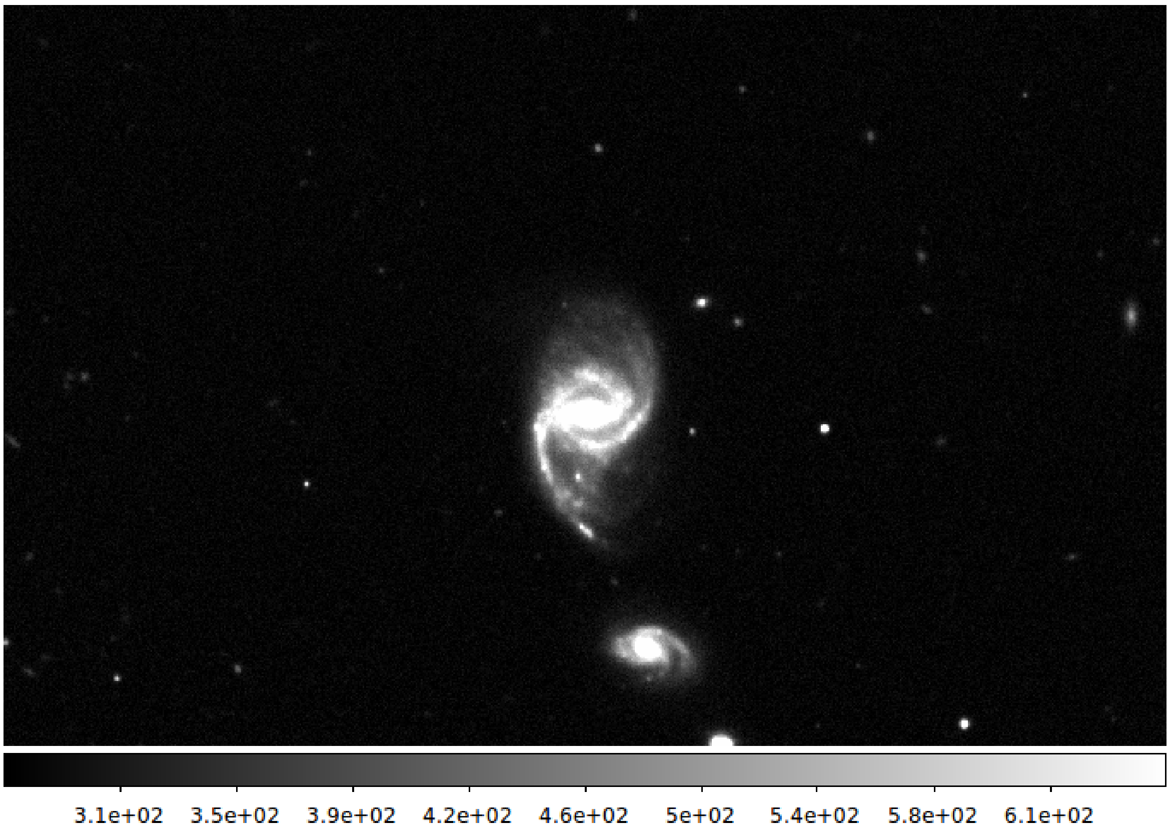
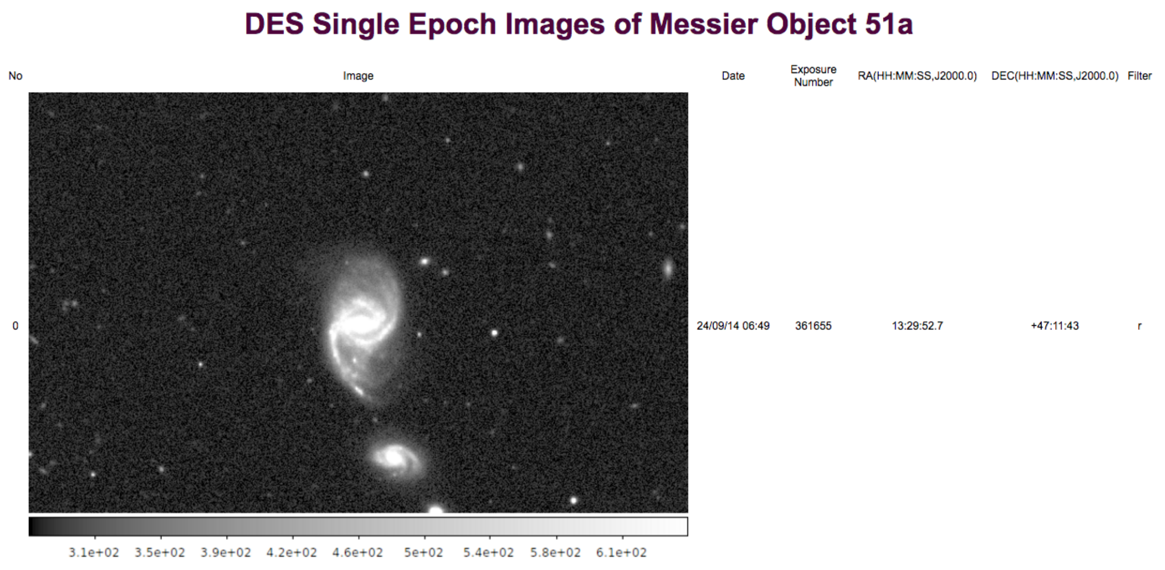


Fig. 4.12 Fully processed image of M51a from ccd 46 of exposure 361655 in the DES


DES Single Epoch Images of Messier Object 51a

No	Image	Date	Exposure Number	RA(HH:MM:SS,J2000.0)	DEC(HH:MM:SS,J2000.0)	Filter
0		24/09/14 06:49	361655	13:29:52.7	+47:11:43	r

Fig. 4.13 Figure 4.12 as seen when uploaded to my webspace.

I have then used the HTML file generating code to produce a page (http://astronomy.sussex.ac.uk/~cc359/html/POC/X3/X3.html) on my webspace for this image. As seen in figure 4.13.

4.3 Multi-epoch images of a given exposure

After creating the single epoch and the proof of concept images I then used my modified code to create images of several different epochs for each exposure previously processed in the single epoch candidate set. In the this section I am going to present the images created for the co-ordinates used to image RN43 in exposure 231500.

As expected nothing that would represent a TNO is present in the circle region that has been applied based on the co-ordinates provided, however notice that the object to the left of the circle is present in all three figures; 4.3,4.14, and 4.15. Furthermore there other objects that are present across all three images, suggesting that the object circled in the initial image in figure 4.3 is a transient object. However further analysis and many more images are required for this to properly shown, which is where using data from other surveys becomes important.

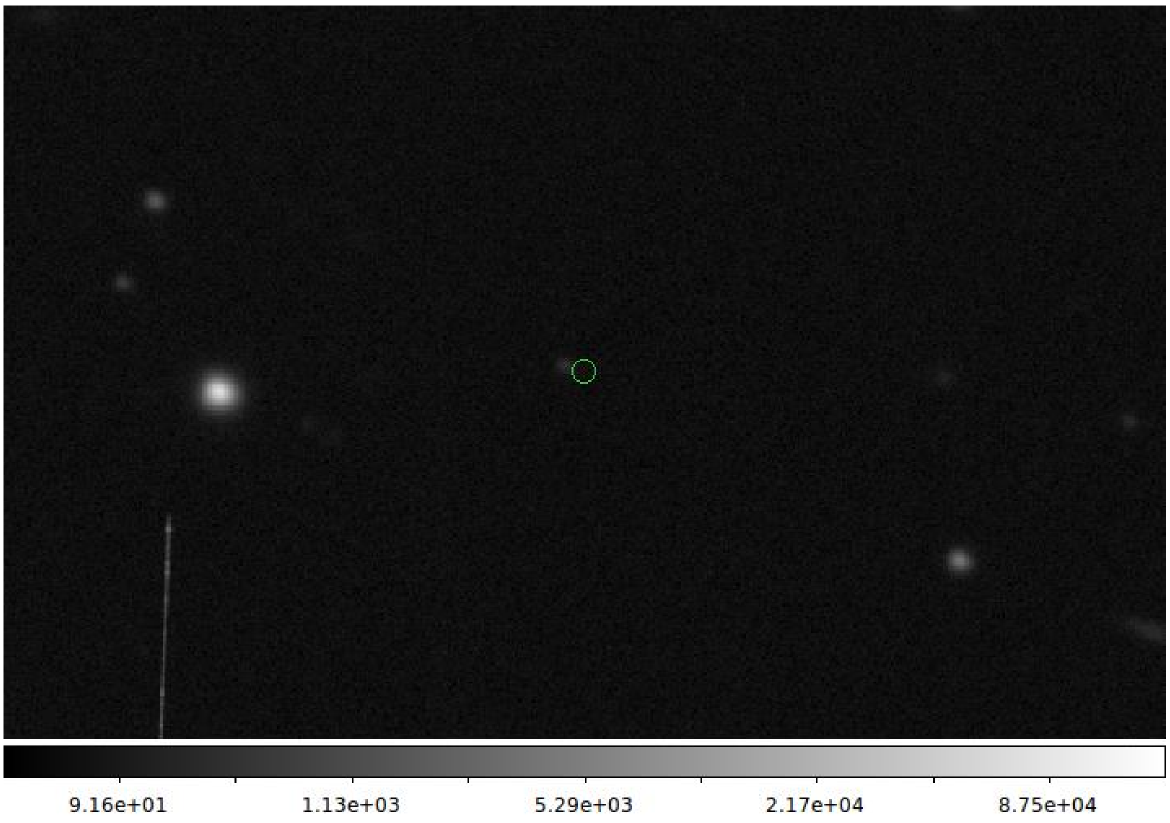


Fig. 4.14 Image of ccd 3 in exposure 232254, cut and combined with the region file for the co-ordinates of RN43 in figure 4.3 on page 26.

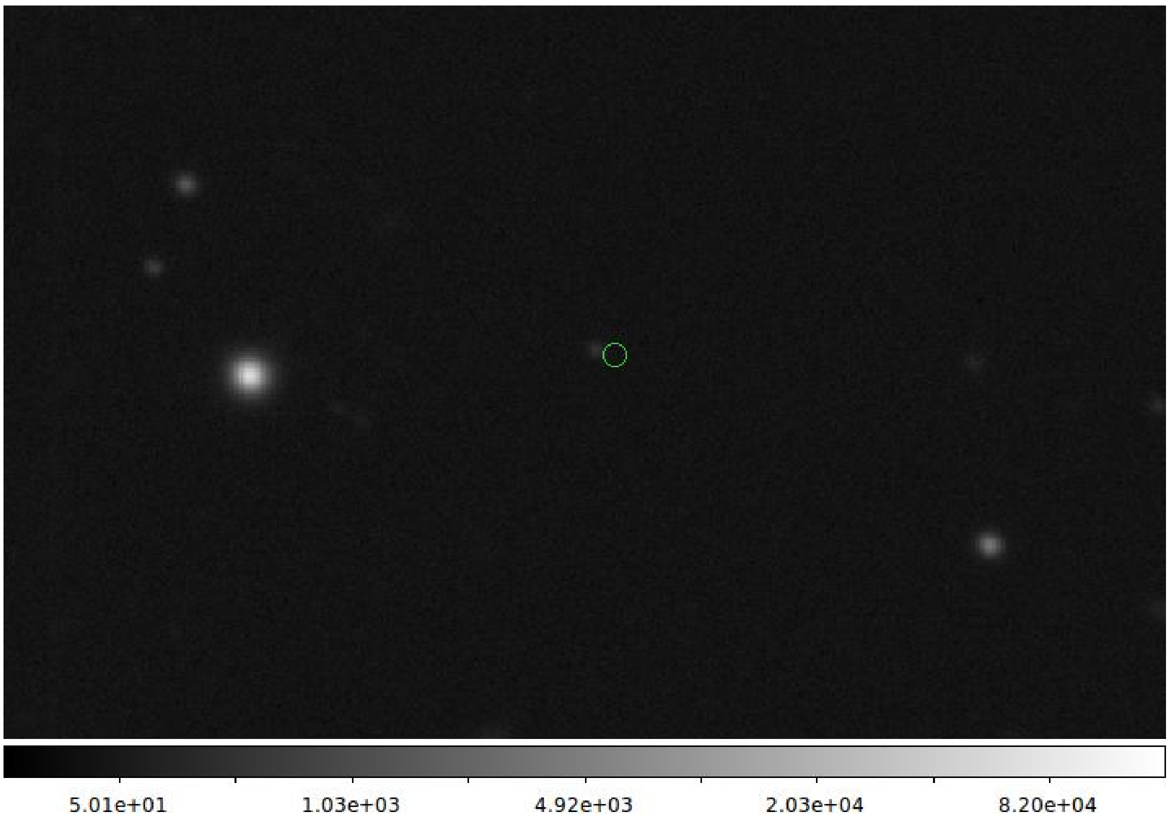


Fig. 4.15 Image of ccd 51 of exposure 232790 , again cut and combined with the region file for the co-ordinates of RN43 in figure 4.3 on page 26.

4.4 Summary

In this chapter I have presented examples of the images created using the data processing method that I have created in conjunction with modules and packages such as `Montage` and `ds9`. I have also presented examples of the webpages that have been created using my HTML file generating code.
These images can now be used to study the TNOs in question further and prove that my method for image processing works.

Chapter 5

Future work and Conclusions

In this last Chapter I will discuss the future work I have planned in terms of data processing and follow up work on my code. Then I will discuss each chapter of this dissertation briefly in the conclusion section.

5.1 Future work

Due to the nature of this project there is a fair amount of work I would like to carry out in order to improve my code and its capabilities.

First of all I will be creating a new set of web pages relating to the images of the TNO co-ordinates in other epochs (multi-epoch images) that are hyperlinked to the respective exposure. This will be achieved by modifying the HTML generating code to include hyperlinks. This will also provide me an opportunity chance to add navigational functions to my webpages by modifying the same code. By doing this I will likely include an argument to the program that specifies whether a single epoch page is being created or a page for the "multi-epoch" images. This is due to the slight differences such as hyperlinks and navigation between the two page types.

I'd then like to add more functionality to my image processing programs by allowing the user to provide arguments to specify the colour map, scaling type and algorithm they want to use when creating images using the `ds9` package. This would likely require a help menu to be added to provide the user with more information on these variables, a useful addition would be to provide them with a url linking them straight to the `ds9` reference manual for further information if they require it.

As previously mentioned I would also like to reduce the amount of the survey that is potentially being disregarded in my multi-epoch image creation process, this is simply due to the tolerance of the automated SQL query in the program for data retrieval, as discussed in 3.4.4 of Chapter 3.

Another overhaul I'd like to make to the entire set of code is to change the format of all images produced to the PNG format due to its lossless compression, however this will mean overcoming several errors that I was having with this format when displaying it on my webspace. This will help to reduce any introduction of artefacts to my final images.

Lastly I'd like to introduce multi-threading capabilities to my code so that I can begin to process larger amounts of data using the University of Sussex high-powered computing cluster in order to provide those working on the search for Planet Nine with larger data sets to work with where images are concerned.

5.2 Conclusions

Chapter 1 contained a brief overview of the Dark Energy Survey (DES) and the apparatus, DECam, used to make observations for the DES, with these exposures being used to produce images with my code.

Then, in Chapter 2 I discussed the DES and how the data it is producing is being used beyond the scope of constraining the w parameter for Dark Energy, and how it is being used to study TNOs and other transient objects in the distant edges of the solar system.

The bulk of this report was contained within Chapter 3 as it discussed, at length, every piece of code I composed to produce images and webpages. This also included brief discussions of packages and modules used that were obtained from other parties, and the specific functions from these that I used. My reasoning for making this chapter somewhat verbose is due to it being a resource for anyone using my code in the future. They will likely be wanting to make modifications to it and as such will want to see the operations each function performed and I also accounted for readers potentially having a very limited knowledge of programming in general.

A sample of the finished images and webpages were included in Chapter 4, to include all of the images and webpages produced in the single document would have dramatically increased it’s length.

Due to the simplicity of my code and it’s ability to successfully produce images, a student is already working with it to produce images and to make sure that it works on other systems that run on Unix or Unix-like operating systems. Hopefully this will mean the future work I have planned for the code can be achieved a faster rate than I would ever achieve on my own.

Appendix A

Code for single epoch image making

N.B. The code in these appendices has been formatted to fit the page correctly, as such some lines of code have been wrapped and do not appear line for line compared to the actual .py and .html files.

```python
#!/usr/bin/ python
import csv, os, sys, requests,shutil
import easyaccess as ea
from collections import defaultdict


#Pre-requisites for use:
# requests : a python library for HTTP requests (http://docs.python-requests.org/en
#/master/)
# Montage: an incredibly powerful package for .fits manipulation
#(http://montage.ipac.caltech.edu/docs/download.html),
#read the documentation for installing carefully and learn how to use SubCube
#and mViewer.
#Access to the DES database , the password and username is those of DESSCI.
#Please change these values in the script atapproximately lines 44 and 45.

#Note if .csv contains ccd's in range of 0-9 the number will need a leading zero
#currently!!!
#The only variables that need changing are DESusername and DESpassword currently,
#I will add them in as arguments to the main process later as it is more secure
#that storing
#This command script works as follows in command line or terminal (this was
#programmed on a Linux distribution so may need changing):
#python single_epoch_no_pass.py the/directory/your/working/from/
#the_csv_you_want_to_analyse.csv argument3
```

```
#Argument3 = 'true' , if you want to make folders and download the fits files ,
#argument3= 'false' if you simply want to cut and make .jpeg images of already existi
#.fits files



###Note if .csv contains ccd's in range of 0-9 the number will need a leading
#zero currently!!!


# Generic CSV file dictionary generator
def ReadFromCSV(fname):
    with open(fname, 'rb') as csvfile:
        reader = csv.DictReader(csvfile)
        data = defaultdict(list)
        for row in reader:
            for (key, value) in row.items():
                data[key].append(value)

        return data
    return None


def MakeFolders(baseDir, Candcsv):    #i.e. Candcsv = CandInfo["expnum"]
    for iterator in range(len(Candcsv["expnum"])):
        newPath = baseDir + Candcsv["expnum"][iterator]
        if not os.path.exists(newPath):
            os.makedirs(newPath)
        else:
            print "Folder␣for␣" + Candcsv["expnum"][iterator] + "␣already␣exists"
    return None

def DownloadFits(baseDir, CandInfo):
    session = requests.Session()
    DESusername = 'cc359' #Username and Password to be stored separately as either
                          #variables or possibly user inputs on session start up
    DESpassword = '*****'
    base = 'https://desar2.cosmology.illinois.edu/DESFiles/desardata/OPS/red/'
    #Example url: https://desar2.cosmology.illinois.edu/DESFiles/desardata/OPS/
    #coadd/20130523000011_DES0414-5748/coadd/DES0414-5748_g.fits.fz
    for iterator in range(len(CandInfo["expnum"])):
        os.chdir(baseDir+'/'+CandInfo["expnum"][iterator])
        url = base+ CandInfo["RUN"][iterator] +'/raw/DECam_00'+
        CandInfo["expnum"][iterator] +'/'+'DECam_00'+
        CandInfo["expnum"][iterator]+'_'+CandInfo["ccd"][iterator]+'.fits.fz'
        #print url
        request = session.get(url, auth=(DESusername,DESpassword),stream=True)

        # Place file into correct folder
        if request.status_code != 200:
                        print 'Did␣not␣connect␣for␣url␣'+ url
        else:
            print "Downloading␣fits.fz␣file␣"+CandInfo["expnum"][iterator]+
            "_"+CandInfo["ccd"][iterator] +".fits.fz"
            with open('DECam_00'+CandInfo["expnum"][iterator]+'_'+
                      CandInfo["ccd"][iterator]+'.fits.fz','wb') as f:
                #writes the fits file in the folder with appropriate name
                                shutil.copyfileobj(request.raw,f)
            os.system('for␣i␣in␣$(ls);␣do␣funpack␣$i;␣done')
            os.system('rm␣DECam_00'+CandInfo["expnum"][iterator]+'_'+
                      CandInfo["ccd"][iterator]+'.fits.fz')

def subCubeCut(baseDir,CandInfo):
    for iterator in range(len(CandInfo["expnum"])):
        os.chdir(baseDir+'/'+CandInfo["expnum"][iterator])

        x_y_param= '0.06667' #x and y of cut length in degrees,
                             #this is approximately 4x4 arcmins
        cmd='mSubCube␣'+'DECam_00'+CandInfo["expnum"][iterator]+'_'+
        CandInfo["ccd"][iterator]+'.fits␣'+CandInfo["expnum"][iterator]+
        '_cut'+'.fits␣'+CandInfo["ra␣(deg)"][iterator]+'␣'+
        CandInfo["dec␣(deg)"][iterator]+'␣'+x_y_param
        os.system(cmd)



def WriteRegionFile(baseDir, CandInfo):
    for iterator in range(len(CandInfo["expnum"])):
        os.chdir(baseDir+'/'+CandInfo["expnum"][iterator])
        regname= open(CandInfo["expnum"][iterator]+'.reg',"w")

        x_circle=CandInfo["ra␣(deg)"][iterator]  #x position ,
                                                #d specifies it is in degrees , RA
        y_circle= CandInfo["dec␣(deg)"][iterator] #y position, also in degrees, DEC
        radius= """0.5'""" #radius of circle to draw, physical units w/o d,
                           #again d specifies in degrees

        contents= 'fk5;circle␣'+x_circle+'␣'+y_circle+'␣'+radius+'␣#␣color=green'
        #circle(x,y,radius) , the '#' seperates the circle command
        #and colors the circle in this case

        regname.write(contents)
        regname.close()

def Circles(baseDir,CandInfo):   #Applies the region file created by
                                 #WriteRegionFile() and creates an image from the
            #fits file in thiscase using a logarithmic scale colourmapped grey
    for iterator in range(len(CandInfo["expnum"])):
        os.chdir(baseDir+'/'+CandInfo["expnum"][iterator])
        fits_name=CandInfo["expnum"][iterator]+'_cut'+'.fits'
        bin_value= "␣-bin␣factor␣3"
        exit= '␣-exit'
        #save_fits= ' -save '+CandInfo["expnum"][iterator]+'_cut'+'.fits'
        map_scale= '␣-log␣-cmap␣grey␣-zmax'
        zoom = '␣-zoom␣'
        export_img= '␣-saveimage␣jpeg␣'+CandInfo["expnum"][iterator]+'_cut'+'.jpeg'+'


        cmd= 'ds9␣-tile␣'+fits_name+'␣-region␣file␣'+
        CandInfo["expnum"][iterator]+'.reg␣'+map_scale+zoom+export_img+exit

        os.system(cmd)



def mViewerImage(baseDir,CandInfo):
    for iterator in range(len(CandInfo["expnum"])):
        os.chdir(baseDir+'/'+CandInfo["expnum"][iterator])
        cutfit= CandInfo["expnum"][iterator]+'_cut'+'.fits'
        imgout= CandInfo["expnum"][iterator]+'_cut'+'.jpeg'
        cmd= 'mViewer␣-gray␣'+cutfit+'␣0s␣99.5%␣'+'linear␣-out␣'+imgout
        os.system(cmd)

def CopyImgs(baseDir,CandInfo):
    for iterator in range(len(CandInfo["expnum"])):
        filename= CandInfo["expnum"][iterator]+'_cut'+'.jpeg'
        filepath_input= baseDir+CandInfo["expnum"][iterator]+"/"+filename
        filepath_output= baseDir+"Images/"+filename
        cmd="cp␣"+filepath_input+"␣"+filepath_output

        os.system(cmd)

```

```

if __name__ == "__main__":


    baseDir = sys.argv[1] # Pass in the location as a command line argument,
                          # means its configurable if another script wishes to
                          # use this script
    csvFile = sys.argv[2]
    needToPreProcess= sys.argv[3]
    CandInfo = ReadFromCSV(baseDir + csvFile)

    if needToPreProcess == 'true':
        MakeFolders(baseDir, CandInfo)

        DownloadFits(baseDir, CandInfo)

        os.system("mkdir "+baseDir+"Images")
        WriteRegionFile(baseDir, CandInfo)
        subCubeCut(baseDir, CandInfo)
        Circles(baseDir,CandInfo)
        CopyImgs(baseDir,CandInfo)

        print "Finished creating directories, downloading image data"+
        " and making images"
    else:
        print "Directories could not be created and/or data could not be"+
        "retrieved. \n Please check the arguments provided on the command line "
        +"and the credentials in the source code."

    if needToPreProcess == 'false' :
            # Run Montage subCube cut process
        os.system("mkdir "+baseDir+"Images")
        WriteRegionFile(baseDir, CandInfo)
        subCubeCut(baseDir, CandInfo)
        Circles(baseDir,CandInfo)
        CopyImgs(baseDir,CandInfo)
            # Run Montage mViewer image from fits
        #mViewerImage(baseDir,CandInfo)

        print "Nothing new downloaded, cutting and image making done!"
    else:
        print "Something went wrong, check that your exposure directories are"+
        " occupied with the relevant .fits file. \n Try re-downloading the files "+
        "by typing 'true' as the third argument"
```

Appendix B

Code for multi-epoch image making

B.1 Code used to retrieve mutli-epoch data from DESDM

```
import csv, os, sys, requests, shutil
import easyaccess as ea
from collections import defaultdict

# Generic CSV file dictionary generator
def ReadFromCSV(fname):
    with open(fname, 'rb') as csvfile:
        reader = csv.DictReader(csvfile)
        data = defaultdict(list)
        for row in reader:
            for (key, value) in row.items():
                data[key].append(value)

        return data
    return None

def CreateOtherEpochDirectory(baseDir,CandName):
    newPath= baseDir+'/Other_Epochs'
    if not os.path.exists(newPath):
        os.makedirs(newPath)
    else:
        print "Folder␣for␣other␣Epochs␣for␣"+CandName+"␣already␣exists"

def CreateOtherEpochSubDirectory(baseDir,Candname):
    for iterator in range(len(CandInfo["expnum"])):
        new_Sub_Path= baseDir+'/Other_Epochs/'+CandInfo["expnum"][iterator]
        if not os.path.exists(new_Sub_Path):
            os.makedirs(new_Sub_Path)
        else:
            print "Folder␣for␣"+CandInfo["expnum"][iterator]+"␣already␣exists␣in"+
            "Other_Epochs"



def QueryEasyAccess(baseDir,csvFile,CandInfo,CandName):

    for iterator in range(len(CandInfo["expnum"])):
        RA_TOLERANCE=0.2  #These values are the tolerance/deviation from the given
                          #fixed co-ordinates, these values are currently based
                          #on values just
        DEC_TOLERANCE=0.05 #under the dimensions of a ccd approximately, before
                           #I was using RA_TOLERANCE= 0.247 and DEC_TOLERANCE= 0.0742
                            # However, these values were found to include too many
                            #exposures that led to the location being outside
                            #of the ccd.



        RA_MIN=str(float(CandInfo["ra␣(deg)"][iterator])-RA_TOLERANCE)  # Minimum RA
                                                                     #value used in SQL query belo
        RA_MAX=str(float(CandInfo["ra␣(deg)"][iterator])+RA_TOLERANCE)  # Maximum RA

        DEC_MIN=str(float(CandInfo["dec␣(deg)"][iterator])-DEC_TOLERANCE) # Minimum D
        DEC_MAX=str(float(CandInfo["dec␣(deg)"][iterator])+DEC_TOLERANCE) # Maximum D
        connection=ea.connect()
        cursor = connection.cursor()

        query="select␣RA_CENT,DEC_CENT,EXPNUM,CCDNUM,BAND,PATH,FILENAME␣from␣felipe."
        "ME_IMAGES_Y3A1_FINALCUT␣where␣(RA_CENT␣between␣"+RA_MIN+"␣and␣"+
        RA_MAX+')␣and␣(DEC_CENT␣between␣'+DEC_MIN+'␣and␣'+DEC_MAX+')'+
        "AND␣BAND='"+CandInfo["band"][iterator]+"'"

        print query

        #Save to a file

        connection.query_and_save(query,baseDir+'Other_Epochs/'+
                                  CandInfo["expnum"][iterator]+'/'+
                                  str(CandInfo["expnum"][iterator])+'.csv')
        #Upload table
        connection.load_table(str(CandInfo["expnum"][iterator])+'.csv',
                                        name='TABLE_'+str(CandName)+'_'+
                                        str(CandInfo["expnum"][iterator])+'_FNAME')

        connection.mytables()
        connection.close()

#Find other observations:
##Use easyaccess docmentation and automate SQL queries
##Save table and then import to local

##Then retrieve the .fits of every ccd within observation >>> to be done in
        #separate python script (for now), also unnecessary to do every CCD
        #using the SQl query from above
if __name__ == "__main__":


    baseDir = sys.argv[1] #Directory containing the .csv file

    csvFile = sys.argv[2]

    CandName = sys.argv[3]


    CandInfo = ReadFromCSV(baseDir + csvFile)

    CreateOtherEpochDirectory(baseDir,CandName)#Creates the "Other_Epochs" directory
                                                        #within main directory that expnum

    CreateOtherEpochSubDirectory(baseDir,CandName) #Creates a sub director within
    #"Other_Epochs" for each expnum,this is to place the output of the easyaccess
    #query into to avoid any confusion between single and multi-epoch data

    QueryEasyAccess(baseDir,csvFile,CandInfo,CandName)#Queries DESDM via easyaccess
    #and outputs a .csv for each expnum in the "main" file into a subdirectory of
    #the "Other_Epochs" directory
```

B.2 Code for creating images from multi-epoch data

```
#!/usr/bin/ python
import csv, os, sys, requests,shutil
import easyaccess as ea
from collections import defaultdict

```

```
6
7  #This is an alternate version of get_single_epoch.py for finding multiple
8  #images for a given RA and DEC.
9
10 #Pre-requisites for use:
11 # requests : a python library for HTTP requests
12 #(http://docs.python-requests.org/en/master/)
13
14 # Montage: an incredibly powerful package for .fits manipulation
15 #(http://montage.ipac.caltech.edu/docs/download.html),
16 #read the documentation for installing carefully
17 #and learn how to use SubCube and mViewer.
18
19 #Access to the DES database , the password and username is those of DESSCI.
20 #Please change these values in the script at lines 44 and 45.
21
22
23 #Note if .csv contains CCDNUM's in range of 0-9 the number will need a leading
24 #zero currently!!!
25 #The only variables that need changing are DESusername and DESpassword
26 #currently , I will add them in as arguments to the main process later
27 #as it is more secure that storing
28
29 #This command script works as follows in command line or terminal
30 #(this was programmed on a Linux distribution so may need changing):
31 #  python single_epoch_no_pass.py the/directory/your/working/from/
32 #                                 the_csv_you_want_to_analyse.csv argument3
33 #Argument3 = 'true' , if you want to make folders and download
34 #the fits files , argument3= 'false' if you simply want to cut and
35 #make .jpeg images of already existing .fits files
36
37
38
39 #Note if .csv contains CCDNUM's in range of 0-9 the number will need a
40 #leading zero currently!!!
41
42
43 # Generic CSV file dictioanry generator
44 def ReadFromCSV(fname):
45     with open(fname, 'rb') as csvfile:
46         reader = csv.DictReader(csvfile)
47         data = defaultdict(list)
48         for row in reader:
49             for (key, value) in row.items():
```

```
                data[key].append(value)

        return data
    return None

def MakeFolders(baseDir, Candcsv):    #i.e. Candcsv = CandInfo["EXPNUM"]
    for iterator in range(len(Candcsv["EXPNUM"])):
        newPath = baseDir + Candcsv["EXPNUM"][iterator]
        if not os.path.exists(newPath):
            os.makedirs(newPath)
        else:
            print "Folder␣for␣" + Candcsv["EXPNUM"][iterator] +
                  "␣already␣exists"
    return None


def DownloadFits(baseDir, CandInfo):
    session = requests.Session()
    DESusername = 'cc359' #Username and Password to be stored separately
    DESpassword = '*****'
    base = 'https://desar2.cosmology.illinois.edu'
        for iterator in range(len(CandInfo["EXPNUM"])):
        os.chdir(baseDir+'/'+CandInfo["EXPNUM"][iterator])

        url = 'https://desar2.cosmology.illinois.edu/DESFiles/desarchive/'+
              str(CandInfo["PATH"][iterator])+'/'+
              str(CandInfo["FILENAME"][iterator])+'.fz'
        request = session.get(url, auth=(DESusername,DESpassword),stream=True)
        print [iterator]

        if request.status_code != 200:
                        print 'Did␣not␣connect␣for␣url␣'+ url
        else:
            print "Downloading␣fits.fz␣file␣"+CandInfo["EXPNUM"][iterator]+"_"+
                   CandInfo["CCDNUM"][iterator] +".fits.fz"
            with open('DECam_00'+CandInfo["EXPNUM"][iterator]+'_'+
                      CandInfo["CCDNUM"][iterator]+'.fits.fz','wb') as f:
                #writes the fits file in the folder with appropriate name
                                shutil.copyfileobj(request.raw,f)
            os.system('for␣i␣in␣$(ls);␣do␣funpack␣$i;␣done')
            os.system('rm␣DECam_00'+CandInfo["EXPNUM"][iterator]+'_'+
                      CandInfo["CCDNUM"][iterator]+'.fits.fz')


```

```
def subCubeCut(baseDir,CandInfo):
    for iterator in range(len(CandInfo["EXPNUM"])):
        os.chdir(baseDir)

        x_y_param= '0.06667' #x and y of cut length in degrees,
                             #this is approximately 4x4 arcmins
        cmd='mSubCube␣'+'DECam_00'+CandInfo["EXPNUM"][iterator]+'_'+
            CandInfo["CCDNUM"][iterator]+'.fits␣'+CandInfo["EXPNUM"][iterator]+
            '_cut'+'.fits␣'+CandInfo["ra␣(deg)"][iterator]+'␣'+
            CandInfo["dec␣(deg)"][iterator]+'␣'+x_y_param
        os.system(cmd)



 def WriteRegionFile(baseDir, CandInfo):
     for iterator in range(len(CandInfo["EXPNUM"])):
         os.chdir(baseDir+'/'+CandInfo["EXPNUM"][iterator])
         regname= open(CandInfo["EXPNUM"][iterator]+'.reg',"w")

          x_circle=CandInfo["ra␣(deg)"][iterator]  #x position , RA
        #is in degrees , RA
          y_circle= CandInfo["dec␣(deg)"][iterator] #y position, DEC
          radius= """0.5'""" #radius of circle to draw

          contents= 'fk5;circle␣'+x_circle+'␣'+y_circle+'␣'+radius+
          '␣#␣color=green' #circle(x,y,radius) , the '#'
                           #seperates the circle command and colors the circle

         regname.write(contents)
         regname.close()

def Circles(baseDir,CandInfo):  #Applies the region file created by
                                #WriteRegionFile() and creates an image
                                #from the fits file in this case using a
                                #logarithmic scale colourmaps grey
    for iterator in range(len(CandInfo["EXPNUM"])):
        os.chdir(baseDir)
        fits_name=CandInfo["EXPNUM"][iterator]+'_cut'+'.fits'
        bin_value= "␣-bin␣factor␣3"
        exit= '␣-exit'
        #save_fits= ' -save '+CandInfo["EXPNUM"][iterator]+'_cut'+'.fits'
        export_img= '␣-saveimage␣jpeg␣'+CandInfo["EXPNUM"][iterator]+'_cut'+
                    '.jpeg'+'␣100'
        map_scale= '␣␣-log␣-cmap␣grey␣-zmax'
        zoom = ' -zoom '

        cmd= 'ds9 -tile '+fits_name+' -region file '+ Expnum+'.reg '+
             map_scale+zoom+export_img+exit

        os.system(cmd)



def mViewerImage(baseDir,CandInfo):
    for iterator in range(len(CandInfo["EXPNUM"])):
        os.chdir(baseDir)
        cutfit= CandInfo["EXPNUM"][iterator]+'_cut'+'.fits'
        imgout= CandInfo["EXPNUM"][iterator]+'_cut'+'.jpeg'
        cmd= 'mViewer -gray '+cutfit+' 0s 99.5% '+'linear -out '+imgout
        os.system(cmd)

def CopyImgs(baseDir,CandInfo):
    for iterator in range(len(CandInfo["EXPNUM"])):
        filename= CandInfo["EXPNUM"][iterator]+'_cut'+'.jpeg'
        filepath_input= baseDir+CandInfo["EXPNUM"][iterator]+"/"+filename
        filepath_output= baseDir+"Images/"+filename
        cmd="cp "+filepath_input+" "+filepath_output

        os.system(cmd)


if __name__ == "__main__":


    baseDir = sys.argv[1] # Pass in the location as a command line argument,
                          # means its configurable if another script wishes
                          #to use this script
    Expnum = sys.argv[2]
    needToPreProcess= sys.argv[3]

    CandInfo = ReadFromCSV(baseDir + Expnum+'.csv')

    if needToPreProcess == 'true':
        MakeFolders(baseDir, CandInfo)

        DownloadFits(baseDir, CandInfo)
               os.system("mkdir "+baseDir+"Images")
        WriteRegionFile(baseDir, CandInfo)
        subCubeCut(baseDir, CandInfo) # Run Montage subCube cut process
        Circles(baseDir,CandInfo)
        CopyImgs(baseDir,CandInfo)

        print "Finished␣batch␣downloading,␣cutting,␣and␣making␣images"
    else:
        print "Something␣went␣wrong,␣check␣that␣your␣exposure␣directories"+
        "are␣occupied␣with␣the␣relevant␣.fits␣file.␣\n"+
        "Try␣re-downloading␣the␣files␣by␣typing␣'true'␣as␣the␣third␣argument"

    if needToPreProcess == 'false' :

        os.system("mkdir␣"+baseDir+"Images")
        WriteRegionFile(baseDir, CandInfo)
        subCubeCut(baseDir, CandInfo) # Run Montage subCube cut process
        Circles(baseDir,CandInfo)
        CopyImgs(baseDir,CandInfo)


        print "Nothing␣new␣downloaded,␣cutting␣and␣image␣making␣done!"
    else:
        print "Something␣went␣wrong,␣check␣that␣your␣exposure␣directories"+
        "are␣occupied␣with␣the␣relevant␣.fits␣file.␣\n"+
        "Try␣re-downloading␣the␣files␣by␣typing␣'true'␣as␣the␣third␣argument"
```

Appendix C

Code to automate production of .html webpages

C.1 HTML file generating code

```
import csv, os, sys, shutil
from collections import defaultdict

def ReadFromCSV(fname):
    with open(fname, 'rb') as csvfile:
        reader = csv.DictReader(csvfile)
        data = defaultdict(list)
        for row in reader:
            for (key, value) in row.items():
                data[key].append(value)

        return data
    return None

def MoveImg(baseDir,CandInfo,ImageDir):
    for iterator in range(len(CandInfo["expnum"])):
        ImgIn=baseDir+CandInfo["expnum"][iterator]+'/'+CandInfo["expnum"][iterator]
        +'_cut'+'.jpeg'
        ImgOut=ImageDir+'/'+CandInfo["expnum"][iterator]+'_cut'+'.jpeg'
        shutil.copyfile(ImgIn,ImgOut)

def makeHomepage(baseDir,CandInfo):
    html= open('homepage.html',"w")
    Header="""<style>body {background-color:#ffffff;}h1"""+
    """{color: #660000; text-align: center; font-family: "Arial"; """+
    """text-decoration: underline}h2{font-family:"Arial" }"""+
    """ul{font-family: "Arial"}</style>
"""
    html.write(Header)


def makeHTML(baseDir,CandInfo,htmlName,CandName,ImageDir):
    html= open(baseDir+htmlName+'.html',"w")
    Header="""<html><head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"><style>"""+
    """body {background-color:#ffffff;}h1 {color: #4f053c; text-align: center; """+
    """font-family: "Arial"}table {font-family: "Arial"; font-size: 12px; """+
    """text-align: center;}</style></head><body>"""+
    """<h1>DES Single Epoch Images of TNO """+
    CandName+"""</h1><table><colgroup><col width="50"><col width="200">"""+
    """<col width="300"><col width="200"><col width="200"><col width="200">"""+
    """</colgroup><tbody><tr><td>No</td><td>Image</td><td>Date</td><td>Exposure'"""+
    """ Number</td>"""+
    """<td>RA(HH:MM:SS,FK5)</td><td>DEC(HH:MM:SS,FK5)</td><td>Filter</td></tr>\n"""
    html.write(Header)
    for iterator in range(len(CandInfo["expnum"])):
        Imagelink="Images"+"/"+str(CandInfo["expnum"][iterator])+'_cut.jpeg'
        itera='<tr><td>'+str(iterator)+'</td><td>'
        date=str(CandInfo["date"][iterator])+'</td><td>'
        expnum=str(CandInfo["expnum"][iterator])+'</td><td>'
        ra=str(CandInfo["ra"][iterator])+'</td><td>'
        dec=str(CandInfo["dec"][iterator])+'</td><td>'
        img ='<img␣src='+Imagelink+'␣height="512"␣width="512"␣border="0"></td><td>'
        filt=str(CandInfo["band"][iterator])+'</td></tr>\n'
        Line=itera+img+date+expnum+ra+dec+filt

        html.write(Line)

    Footer='</tbody></table></body></html>'
    html.write(Footer)
    html.close()





if __name__ == "__main__":


    baseDir = sys.argv[1] # Pass in the location as a command line argument,
                          # means its configurable if another script wishes to
                          #use this script
    csvFile = sys.argv[2]

    htmlName = sys.argv[3]
    CandName= sys.argv[4]


    CandInfo = ReadFromCSV(baseDir + csvFile)
    ImageDir=baseDir+"Images"
    os.system('mkdir␣'+ ImageDir)


    MoveImg(baseDir ,CandInfo ,ImageDir)

    makeHTML(baseDir ,CandInfo ,htmlName ,CandName ,ImageDir)
```

C.2 Example of HTML file produced from code

```
<html><head>
    <meta http-equiv="content-type" content="text/html;␣charset=UTF-8"><style>body
{background-color:#ffffff;}h1 {color: #4f053c; text-align: center; font-family:
 "Arial"}table {font-family: "Arial"; font-size: 12px; text-align: center;}</
style></head><body><h1>DES Single Epoch Images of TNO RN43</
h1><table><colgroup><col width="50"><col width="200"><col width="300"><col
width="200"><col width="200"><col width="200"></colgroup><tbody><tr><td>No</
td><td>Image</td><td>Date</td><td>Exposure Number</td><td>RA(HH:MM:SS,FK5)</
td><td>DEC(HH:MM:SS,FK5)</td><td>Filter</tr>


<tr><td>0</td><td><img src=Images/231500_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/9/8 03:20:34</td><td>231500</td><td>22:20:49.06</
td><td>0:28:06.9</td><td>g</td></tr>

<tr><td>1</td><td><img src=Images/232254_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/9/10 01:32:50</td><td>232254</td><td>22:20:40.56</
td><td>0:27:02.7</td><td>g</td></tr>

<tr><td>2</td><td><img src=Images/233497_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/9/13 03:40:05</td><td>233497</td><td>22:20:27.10</
td><td>0:25:18.5</td><td>i</td></tr>

```

```
<tr><td>3</td><td><img src=Images/233499_cut.jpeg height="498" width="714"
border="0"></td><td>2013/9/13 03:44:07</td><td>233499</td><td>22:20:27.09</
td><td>0:25:18.5</td><td>z</td></tr>

<tr><td>4</td><td><img src=Images/239310_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/9/29 04:01:45</td><td>239310</td><td>22:19:22.82</
td><td>0:16:10.0</td><td>i</td></tr>

<tr><td>5</td><td><img src=Images/242717_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/10/11 01:56:51</td><td>242717</td><td>22:18:43.51</
td><td>0:09:31.1</td><td>z</td></tr>

<tr><td>6</td><td><img src=Images/243048_cut.jpeg height="498" width="714"
 border="0"></td><td>2013/10/12 00:49:51</td><td>243048</td><td>22:18:40.77</
td><td>0:09:00.2</td><td>z</td></tr>

<tr><td>7</td><td><img src=Images/243449_cut.jpeg height="498" width="714"
border="0"></td><td>2013/10/13 00:31:20</td><td>243449</td><td>22:18:37.99</
td><td>0:08:28.4</td><td>z</td></tr>

</tbody></table></body></html>
```

Appendix D

SQL Queries

D.1 Example query for retrieval of run values for given exposures

```
SELECT EXPNUM,RUN FROM Y1A1_FIRSTCUT_EVAL WHERE EXPNUM IN
(231500,232254,233497,233499,239310,242717,243048,243449);
```

Appendix E

Email from Professor David Gerdes

On 05/02/2016 17:02, David Gerdes wrote:
Hi Kathy,

So I would like something like this:

A search produces an object of interest at a certain (ra, dec) in a given exposure number / ccd. It would be great to press a button and get:

- Thumbnail of the region of interest, with the object circled and some key properties displayed (e.g. flux, flux_err, spread_model, spread_model_err, etc.)
- Thumbnails of the same region in other epochs, prioritized by quality (e.g. t_eff, fwhm_asec) and band (prefer the same band as the object, but others are also of interest).

Also it would be nice if the thing could interact with some kind of candidate database so things could be marked and commented after inspection and we know where they sit in the workflow. (I have spent too much time re-discovering or re-rejecting the same candidates...)

I talked with Luiz about this awhile ago. The answer was "sounds useful, it's in our plans", but I haven't heard anything lately. I know he is interested in TNOs (he just established an EC arrangement for a Brazilian group interested in TNO occultations), so it's probably worth asking again. Personally I have found the portal to be nearly unusable but I know others have had a more positive experience.

Cheers,

Dave

Bibliography

[1] H. Arp. Atlas of Peculiar Galaxies. *ApJS*, 14:1, November 1966.

[2] K. Batygin and M. E. Brown. Evidence for a Distant Giant Planet in the Solar System. *AJ*, 151:22, February 2016.

[3] Gary Bernstein and Bharat Khushalani. Orbit fitting and uncertainties for kuiper belt objects. *The Astronomical Journal*, 120(6):3323, 2000.

[4] L. E. Bleem, B. Stalder, T. de Haan, K. A. Aird, S. W. Allen, D. E. Applegate, M. L. N. Ashby, M. Bautz, M. Bayliss, B. A. Benson, S. Bocquet, M. Brodwin, J. E. Carlstrom, C. L. Chang, I. Chiu, H. M. Cho, A. Clocchiatti, T. M. Crawford, A. T. Crites, S. Desai, J. P. Dietrich, M. A. Dobbs, R. J. Foley, W. R. Forman, E. M. George, M. D. Gladders, A. H. Gonzalez, N. W. Halverson, C. Hennig, H. Hoekstra, G. P. Holder, W. L. Holzapfel, J. D. Hrubes, C. Jones, R. Keisler, L. Knox, A. T. Lee, E. M. Leitch, J. Liu, M. Lueker, D. Luong-Van, A. Mantz, D. P. Marrone, M. McDonald, J. J. McMahon, S. S. Meyer, L. Mocanu, J. J. Mohr, S. S. Murray, S. Padin, C. Pryke, C. L. Reichardt, A. Rest, J. Ruel, J. E. Ruhl, B. R. Saliwanchik, A. Saro, J. T. Sayre, K. K. Schaffer, T. Schrabback, E. Shirokoff, J. Song, H. G. Spieler, S. A. Stanford, Z. Staniszewski, A. A. Stark, K. T. Story, C. W. Stubbs, K. Vanderlinde, J. D. Vieira, A. Vikhlinin, R. Williamson, O. Zahn, and A. Zenteno. Galaxy Clusters Discovered via the Sunyaev-Zel'dovich Effect in the 2500-Square-Degree SPT-SZ Survey. *ApJS*, 216:27, February 2015.

[5] Cerro Tololo Inter-American Observatory Consortium. Victor blanco 4-m telescope, http://www.ctio.noao.edu/noao/node/9, August 2016.

[6] Dark Energy Survey Collaboration, T. Abbott, F. B. Abdalla, J. Aleksić, S. Allam, A. Amara, D. Bacon, E. Balbinot, M. Banerji, K. Bechtol, A. Benoit-Lévy, G. M. Bernstein, E. Bertin, J. Blazek, C. Bonnett, S. Bridle, D. Brooks, R. J. Brunner, E. Buckley-Geer, D. L. Burke, G. B. Caminha, D. Capozzi, J. Carlsen, A. Carnero-Rosell, M. Carollo, M. Carrasco-Kind, J. Carretero, F. J. Castander, L. Clerkin, T. Collett, C. Conselice, M. Crocce, C. E. Cunha, C. B. D'Andrea, L. N. da Costa, T. M. Davis, S. Desai, H. T. Diehl, J. P. Dietrich, S. Dodelson, P. Doel, A. Drlica-Wagner, J. Estrada, J. Etherington, A. E. Evrard, J. Fabbri, D. A. Finley, B. Flaugher, R. J. Foley, P. Fosalba, J. Frieman, J. García-Bellido, E. Gaztanaga, D. W. Gerdes, T. Giannantonio, D. A. Goldstein, D. Gruen, R. A. Gruendl, P. Guarnieri, G. Gutierrez, W. Hartley, K. Honscheid, B. Jain, D. J. James, T. Jeltema, S. Jouvel, R. Kessler, A. King, D. Kirk, R. Kron, K. Kuehn, N. Kuropatkin, O. Lahav, T. S. Li, M. Lima, H. Lin, M. A. G. Maia, M. Makler, M. Manera, C. Maraston, J. L. Marshall, P. Martini, R. G. McMahon, P. Melchior, A. Merson, C. J. Miller, R. Miquel, J. J. Mohr, X. Morice-Atkinson, K. Naidoo, E. Neilsen, R. C. Nichol, B. Nord, R. Ogando, F. Ostrovski, A. Palmese, A. Papadopoulos, H. V. Peiris,

J. Peoples, W. J. Percival, A. A. Plazas, S. L. Reed, A. Refregier, A. K. Romer, A. Roodman, A. Ross, E. Rozo, E. S. Rykoff, I. Sadeh, M. Sako, C. Sánchez, E. Sanchez, B. Santiago, V. Scarpine, M. Schubnell, I. Sevilla-Noarbe, E. Sheldon, M. Smith, R. C. Smith, M. Soares-Santos, F. Sobreira, M. Soumagnac, E. Suchyta, M. Sullivan, M. Swanson, G. Tarle, J. Thaler, D. Thomas, R. C. Thomas, D. Tucker, J. D. Vieira, V. Vikram, A. R. Walker, R. H. Wechsler, J. Weller, W. Wester, L. Whiteway, H. Wilcox, B. Yanny, Y. Zhang, and J. Zuntz. The Dark Energy Survey: more than dark energy - an overview. *MNRAS*, 460:1270–1299, August 2016.

[7] D. W. Gerdes, R. J. Jennings, G. M. Bernstein, M. Sako, F. Adams, D. Goldstein, R. Kessler, S. Hamilton, T. Abbott, F. B. Abdalla, S. Allam, A. Benoit-Lévy, E. Bertin, D. Brooks, E. Buckley-Geer, D. L. Burke, D. Capozzi, A. Carnero Rosell, M. Carrasco Kind, J. Carretero, C. E. Cunha, C. B. D'Andrea, L. N. da Costa, D. L. DePoy, S. Desai, J. P. Dietrich, P. Doel, T. F. Eifler, A. Fausti Neto, B. Flaugher, J. Frieman, E. Gaztanaga, D. Gruen, R. A. Gruendl, G. Gutierrez, K. Honscheid, D. J. James, K. Kuehn, N. Kuropatkin, O. Lahav, T. S. Li, M. A. G. Maia, M. March, P. Martini, C. J. Miller, R. Miquel, R. C. Nichol, B. Nord, R. Ogando, A. A. Plazas, A. K. Romer, A. Roodman, E. Sanchez, B. Santiago, M. Schubnell, I. Sevilla-Noarbe, R. C. Smith, M. Soares-Santos, F. Sobreira, E. Suchyta, M. E. C. Swanson, G. Tarlé, J. Thaler, A. R. Walker, W. Wester, Y. Zhang, and DES Collaboration. Observation of Two New L4 Neptune Trojans in the Dark Energy Survey Supernova Fields. *AJ*, 151:39, February 2016.

[8] D. A. Goldstein, C. B. D'Andrea, J. A. Fischer, R. J. Foley, R. R. Gupta, R. Kessler, A. G. Kim, R. C. Nichol, P. E. Nugent, A. Papadopoulos, M. Sako, M. Smith, M. Sullivan, R. C. Thomas, W. Wester, R. C. Wolf, F. B. Abdalla, M. Banerji, A. Benoit-Lévy, E. Bertin, D. Brooks, A. Carnero Rosell, F. J. Castander, L. N. da Costa, R. Covarrubias, D. L. DePoy, S. Desai, H. T. Diehl, P. Doel, T. F. Eifler, A. Fausti Neto, D. A. Finley, B. Flaugher, P. Fosalba, J. Frieman, D. Gerdes, D. Gruen, R. A. Gruendl, D. James, K. Kuehn, N. Kuropatkin, O. Lahav, T. S. Li, M. A. G. Maia, M. Makler, M. March, J. L. Marshall, P. Martini, K. W. Merritt, R. Miquel, B. Nord, R. Ogando, A. A. Plazas, A. K. Romer, A. Roodman, E. Sanchez, V. Scarpine, M. Schubnell, I. Sevilla-Noarbe, R. C. Smith, M. Soares-Santos, F. Sobreira, E. Suchyta, M. E. C. Swanson, G. Tarle, J. Thaler, and A. R. Walker. Automated Transient Identification in the Dark Energy Survey. *AJ*, 150:82, September 2015.

[9] Space Telescope Science Institute. Module zscale,http://stsdas.stsci.edu/stsci_python_epydoc/numdisplay/numdisplay.zscale-module.html, January 2011.

[10] R. Kessler, J. Marriner, M. Childress, R. Covarrubias, C. B. D'Andrea, D. A. Finley, J. Fischer, R. J. Foley, D. Goldstein, R. R. Gupta, K. Kuehn, M. Marcha, R. C. Nichol, A. Papadopoulos, M. Sako, D. Scolnic, M. Smith, M. Sullivan, W. Wester, F. Yuan, T. Abbott, F. B. Abdalla, S. Allam, A. Benoit-Lévy, G. M. Bernstein, E. Bertin, D. Brooks, A. Carnero Rosell, M. Carrasco Kind, F. J. Castander, M. Crocce, L. N. da Costa, S. Desai, H. T. Diehl, T. F. Eifler, A. Fausti Neto, B. Flaugher, J. Frieman, D. W. Gerdes, D. Gruen, R. A. Gruendl, K. Honscheid, D. J. James, N. Kuropatkin, T. S. Li, M. A. G. Maia, J. L. Marshall, P. Martini, C. J. Miller, R. Miquel, B. Nord, R. Ogando, A. A. Plazas, K. Reil, A. K. Romer, A. Roodman, E. Sanchez, I. Sevilla-Noarbe, R. C. Smith, M. Soares-Santos, F. Sobreira, G. Tarle, J. Thaler, R. C. Thomas,

D. Tucker, A. R. Walker, and DES Collaboration. The Difference Imaging Pipeline for the Transient Search in the Dark Energy Survey. *AJ*, 150:172, December 2015.

[11] Smithsonian Astrophysical Observatory. How it works, scale.

[12] W. Pence, R. Seaman, and R. White. Fpack and Funpack User's Guide: FITS Image Compression Utilities. *ArXiv e-prints*, December 2011.

[13] The Dark Energy Survey Collaboration. The Dark Energy Survey. *ArXiv Astrophysics e-prints*, October 2005.

[14] C. A. Trujillo and S. S. Sheppard. A Sedna-like body with a perihelion of 80 astronomical units. *Nature*, 507:471–474, March 2014.

[15] ŷhat blog. Random forests in python, http://blog.yhat.com/posts/random-forests-in-python.html, June 2013.