Project 3, part I – M67 Cluster Photometry

Due December 4, 2012

ASTR310 Fall 2012

1 Introduction

You should keep in mind that there are two separate aspects to this project as far as an astronomer is concerned. First is the knowledge of techniques applicable to CCDs and second is the knowledge of more general techniques related to obtaining color-magnitude (C-M) diagrams, whether the data are obtained with a CCD, with photographic plates, or with a classical photometer. There are, of course, many interesting things related to the interpretation of the C-M diagram that are the motivation for such observations, but these are beyond the scope of this course.

2 The Data for M67

To get the files for this lab, make a new directory M67 with the command **mkdir M67**. Then change your working directory to be M67 with **cd M67** and copy the data files for this part of the Project by typing **cp harris/a310/M67/* .** (this dot is UNIX/LINUX shorthand for the directory where you are, so don't forget it!).

Use the list command `ls` to see what's in your directory. There should be a total of eight FITS files. There are four images of the open cluster: a pair of images of two overlapping fields taken through a V filter (m67vc.fits and m67ve.fits) and a pair of images of the same two fields taken through an R filter (m67rc.fits and m67re.fits). There are also flat-field images for each filter and dark frames for the sky exposures (dark_M67.fits) and the flatfield (dark_flat.fits). Finally there are two data files, m67.txt and ngc7243.txt, which contain the Color-Magnitude (C-M) data for M67 and NGC 7243 (the latter is for Part II) that were published in the astronomical literature. The formats are not the same since the two sets of data come from different papers. In m67.txt, column 1 is the B-V, and column 2 is the V magnitude, while in ngc7243.txt, column 1 is the V magnitude, and column 2 is the B-V.

You can view the images by starting MATLAB (see the separate write-up to review the basics of working with MATLAB) and reading in and displaying the .fits files with commands like this:

```
vc=rfits('m67vc.fits')
imagesc(vc.data);
axis image
```

where the third command makes the pixels square in the display. Now the variable *vc.data* is an array with 572 columns and 380 rows, containing the image of the cluster center through the *V* filter. The image is naturally displayed in the correct astronomical orientation by **imagesc** in this case, with North up and East to the left. Because there is a wide range of intensity between the brightest and the faintest stars, you should try displaying the logarithm of the intensity: **imagesc(log10(vc.data))**.

Since you will always want to start by reading in your images, it will save a lot of time and typing errors if you set up a file with commands to do that job for you. Within MATLAB, you can use the **edit** command to start an editor session. In a new file, enter as text a series of commands to read the files. Thus:

```
vc=rfits('m67vc.fits');
ve=readfits('m67ve.fits');
...
rf=rfits('rflat.fits');
darkM67=rfits('dark_M67.fits');
darkF=rfits('dark_flat.fits');
```

Save this file with a name like 'read_images'. Then when you start MATLAB, you need only type

```
read_images
```

and your images will all be read into the names you have given them.

3 Image Processing

The first part of the lab consists of data reduction and a short analysis of the results. The data reduction consists of three procedures:

- Removing various instrumental artifacts of the CCD, i.e., dark current removal and flat-fielding.
- Measuring the magnitudes of all of the stars on all of the images.
- Combining the overlapping images (i.e., combining the m67vc.fits image of the *center* of the cluster with the m67ve.fits image of the *eastern* part of the cluster) by identifying the stars seen on both images and averaging their magnitudes.

To get the special functions we provide for this project, type **addpath harris/a310** in your MATLAB session or add it to your script.

3.1 Dark Current Removal

First of all, subtract the dark current from each of the other frames (images). You can do this with simple statements of the form

```
vcd = vc.data - darkM67.data;
vfd = vf.data - darkF.data;
```

where the variable name *vcd* now contains the dark current-corrected image. Note that *vcd* and *vfd* are matrices, not structures like **vc** – this makes manipulations simpler. (*vc.data* is a matrix field in the structure *vc*). Now the zero of intensity is really at zero as far as instrumental terms go, although the blank portions of your images will not contain zeros because the sky at our observatory is bright. Don't forget to subtract the dark current from the flat-field images as well as from the images of the cluster. Maybe it occurs to you that you could just update the data in "read_images" file so you won't have to repeat these steps every time you work on the reduction. Good idea. Do it.

3.2 Flat-Fielding

There is variation in the sensitivity of the CCD from one pixel to another. This can be due to vignetting of the optics or to variations in their electrical characteristics. This variation often depends on the wavelength of the incident light. "Flat-field" images are taken of the twilight sky or of an illuminated screen in the telescope dome through each filter to determine this response. You have flat-field images for V and R in your directory. To see the flat-field variations, let's first look at the mean value in the flat-field image:

```
mean(vfd(:))
```

assuming **vf** to be the name you have given the V band flat-field image. You will see that the image is well-exposed (but still below the maximum value of 16000 where the CCD saturates). You can then make a *normalized* flat field as follows:

```
vfdn = vfd/mean(vfd(:));
```

This image will have an average value of 1.0, but will be less where the CCD is less sensitive, and greater than one where the CCD is most sensitive. Make such an image, and look at it (you can see the range of values being displayed using **colorbar** after **imagesc**).

Now divide the normalized flat-field frames for each filter into all of the others taken through the same filter. For example:

```
vcdf = vcd./vfdn;
```

With this division, you see that this will increase the values in the image where the CCD is less sensitive, and decrease them where it's more sensitive. This is just what we want in order to put the whole frame on the same intensity scale. As a result, you should now have four images, two in V and two in R, which are bias corrected and flat fielded. These are the images we will measure. **Once again, the flat-fielding commands can be put into your "read_images" file, so you don't have to repeat them again.**

3.3 Measuring the Stellar Magnitudes

Two MATLAB procedures are used to get magnitudes. **FINDSTARS** finds the stars in an image and lists them. **APER** measures the fluxes of a star at a given position.

To see how FINDSTARS works, you can invoke it with **findstars('findstars.out',im,500,10);**, where **im** is the name of one of your images (corrected for the bias and flat-fielded, of course).

The second parameter is the "minimum value above background". If this is set too high, you will lose the faint stars, but if set too low, you will get all sorts of fluctuations that are not real stars. The background we are referring to is the sky brightness in the image - using $curval, im$ will show you that this is in the range of 200 – 400, depending on the image. So for "minimum value above background", try 500 (this is actually much too high, and will miss faint stars, but it's a good place to start to get a feel for how it all works).

The third parameter is the size of the patch that it uses to remove stars from the image. If you set this value too high bad things will happen (fainter stars will disappear as they are blotched by a nearby star). If you set it too small, not enough of the original star will be removed and the search algorithm will be confused (you will get a lot of chaff). Experiment with different values, but 10 pixels is a good value to start with.

As FINDSTARS executes, you will see that the stars that the algorithm finds are removed from the image, replaced with a patch of the local background value. As the removal proceeds the image color stretch is increased more and more, revealing fainter sources (and a lot of noise). To experiment with a fainter threshold, restart it with **findstars('findstars.out',im,50,10);**.

When it completes, it will show you the stars found overlaid as white circles over the original image. It will also return a matrix with the results of the fits to the stars. This matrix contains the following information (one star per row): X and Y coordinates (in CCD pixels), the peak brightness value, the size of the major and minor axes found by the fit (as a full-width at half-maximum), and the orientation of the ellipse (position angle). The same information is written to a file in your directory called "findstars.out" (this file is overwritten each time FINDSTARS is executed; if you want to keep previous versions, rename them before running FINDSTARS again). To see the contents of this file from within MATLAB, use the command **type** as in **type findstars.out**.

You will see that some stars are actually artifacts in the CCD (cosmic ray strikes or hot pixels), and some stars are a blend of more than one source. Note that frequently it is easy to spot these sources as their size is very different from those of real stars in the image. If you'd like, use an editor to remove the lines corresponding to the fake sources from the file. In any case, we will have another opportunity at getting rid of erroneous sources in the next steps.

Having found the stars in an image, we next use APER to measure the fluxes. This routine measures a star's intensity through a circular aperture. You give APER the name of the image and the coordinates from FINDSTARS (or using the data cursor on the image). It will also require the following information:

- The radius of the aperture for the flux measurement. You want to pick the size of the aperture to include as much of the star's light as possible, without including too much sky background. This is a compromise choice because there is a lot of light quite far from the central peak, as you can see from the bright stars, but if you include all of it then you will be including a lot of sky noise when you measure the dim stars. This is where having a well-focused telescope helps retain the dim stars because you can get away with using a small aperture. The effect of a big aperture, necessary if the telescopic focus is poor, is to lose the dim stars in the noise, and to lose any stars that have overlapping apertures. Using too small an aperture is better than using too large an aperture, but your result then becomes very sensitive to exactly how the star is centered on the pixel pattern. Using a radius about the same as the FWHM (from FINDSTARS) is probably a good place to start, but you should experiment.

- The inner and outer radii of a ring through which the program will measure the sky brightness. This value is then normalized to the area of the star aperture and subtracted from the star's flux. The inner radius should be greater or equal to the star aperture. The outer radius is again a compromise: a large aperture will give a better mean sky value, but at the same time may approach another star, which will contaminate the sky measurement.

- Finally, APER needs the number of electrons/ADU for the CCD (so it can do Poisson statistics). For the photometrics CCD on the 20-inch, which took these data, it is probably best to use the manufacturer's value of $K = 16$. This is of course (!) the value **K** that you determined in the first lab. Note that for the camera presently used in the 20-inch (the Apogee camera that you will use to get your own images for the second part of this lab) the gain is $K = 2.3$.

Using this information, you can now try to run APER using, for example, the coordinates of the first star returned by FINDSTARS: **[flx,err]=aper(im,108.2,272.1,8,12,15,16)**. When executing, APER will show you a cutout of your image centered on the position you gave it, with a white circle indicating the region over which the flux of the star will be calculated, and two red circles showing you the annulus you selected for the sky. APER returns two values, the star flux and its error.

To make things a bit simpler, use the function called **all_aper('phot.out','findstars.out',im)**. This routine takes the name of the file written by FINDSTARS (you can edit it to remove erroneous sources if you want) and the image, and it executes APER for each source in the list. Please make a copy of this in your directory **cp harris/a310/all_apr.m .** and edit it to change the parameters to others you may want to try. After executing APER for each source, it will ask you whether the result is good or bad (1 or 0). Enter the corresponding value (**note that you cannot run all_aper within a script, because it needs your interactive input!** If you are tired of pressing keys, change they way all_aper works!). Use this to mark as "bad" the photometry for sources that are blended, caused by hot pixels, or too close to the edges of the CCD. If you make a mistake do not worry, you can correct it later. This information will then be written into a file called "phot.out" together with the result of APER. This is your chance to flag some stars as bad, if you have not already removed them by editing the "findstars.out" file. Be sure to mark as bad any spurious source (like a hot pixel) and any star where the aperture overlaps with another star. Remember that you can get a hard copy of the photometry by using the UNIX command to print the file: **lpr phot.out**. The file can be loaded into a MATLAB matrix **p** using **p=load('phot.out')**. The columns correspond to: (1,2) the x and y coordinates, (3) the flux of the source, (4) the error in the flux, and (5) the value you entered indicating whether the photometry is valid or not. If you made a mistake when you entered the flag, that can be easily corrected by editing the file.

Those stars which are in the part of the sky where the central ("c") and eastern ("e") images overlap will have been measured twice in each filter. By comparing the results, we can estimate the measurement error. In addition, the best value for the magnitude in a given color will be the average of the two measured values. You will use a procedure called **pmerge** to merge the two photometry files from the "e" and "c" plates. You first must identify a reference star seen on both images, and write down its coordinates on both the "e" and "c" images. For example, suppose the two images through the visual filter are called **vc** and **ve**. Then display the first image with **figure(1); imagesc(vc)** and the other with **figure(2); imagesc(ve)**. You can then use the data cursor to obtain the coordinates of the same star in one image and another.

Suppose a star in the **vc** image has coordinates x=100 and y=200, and the same star in the **ve** image has coordinates x=250 and y=190. Then you would run **merge** as follows:

pmerge('vmerge.pht','veast.out',[340,277],'vcen.out',[108,272])

where we assume that **all_aper** has been run on the eastern image to produce the file "veast.out". (You want to put the east image first so the new coordinates will be positive.) The result will be found in a file called "vmerge.out". This file has a format similar to that produced by **all_aper**, with seven columns: (1,2) x and y coordinates (those in the second image will be shifted to coincide with those in the first), (3) the flux, (4) the error in the flux, (5) the flag indicating validity (should be all ones, invalid sources are expunged in the merge process), (6,7) the location of the star in the original files (0 if not in one of them). Be careful that the two files you wish to merge do not contain any blank lines, as this will result in an error when you run **pmerge**.

3.4 Plotting the Color-Magnitude Diagram

Now you need to get the color of your stars. A "color" is the difference between the magnitudes measured through different filters. Since we have used the V and the R filters, we will construct the (V-R) color. The (instrumental) magnitude is simply defined as $m = -2.5\log(F)$, where F is the flux of the source. We will later convert this instrumental magnitude to one of the standard systems.

To calculate the colors use the **scolor** task on the merged photometry files corresponding to the V and the R filters. This task will take the difference between the V and R magnitudes, dropping any stars that were only measured in one filter, or that have been flagged as bad in either of them.

Suppose the two photometry files are called "vmerge.pht" and "rmerge.pht". We have to again locate the coordinates of a reference star on the m67ve and the m67re images (assuming that the "e" file was first when we ran merge). Then we use a command like this:

scolor('vrfile.col','vmerge.pht',[208,272],'rmerge.ph',[106,270])

where the numbers in [...] are the x & y coordinates, and 'vrfile' is a name of your choice. The procedure will write a file called **vrfile.col** which will have seven columns: (1,2) x and y coordinates (referred to the first image), (3) V magnitude, (4) R magnitude, (5) V-R color, and (6,7) locations of the star in the original files. The procedure will also put up a plot of V vs. (V-R). This should be a recognizable C-M diagram since the conversion to the standard system will only alter its shape a bit and also change scales on the coordinate axes somewhat.

Finally, you have to correct for atmospheric extinction, convert the data to the standard color system and convert from V-R to B-V. All three of these transformations can be combined into a single linear transformation but you have to figure out the transformation coefficients. (This simplification works only because we know the magnitudes of two of the stars in the images.) Data on the two stars are given in the table (from Eggen & Sandage ApJ 140, 130, 1964); the approximate coordinates refer to the m67re.fits image.

x	y	V	B-V
340	274	9.69	1.355
489	164	12.82	0.585

You have to calculate the four transformation coefficients a, b, c, and d in the expressions

$$\begin{aligned} B - V &= a * (v_0 - r_0) + b \qquad (1) \\ V &= v_0 + c * (v_0 - r_0) + d, \qquad (2) \end{aligned}$$

where v_0 and r_0 are the magnitudes in your instrumental system while B and V are the magnitudes in the standard system. Find the two standard stars in your .col file from their coordinates. Then you will have numerical values for four equations, and you can solve for a, b, c, and d.

Once you have the four transformation coefficients, read in your instrumental magnitude and color matrix with

p=load('vrfile.col')

and then use MATLAB's array arithmetic to get the properly standardized V and (B-V) values:

BmV = a*p(:,5) + b
V = p(:,3) + c*p(:,5) + d

You can now use **plot** to generate the H-R diagram. To get the magnitude axis to increase downwards you can flip the y-axis using **set(gca,'ydir','reverse')**. A file in your directory named m67.txt contains published data for this cluster. You can use the **hold** command (see inside MATLAB **help hold**) to plot this data on the same graph to see how they compare. When you get a satisfactory plot, you should create a hard copy and print it out to include it in your report (see MATLAB tutorial).

You should leave all of your data reduction files in your directory and a listing of your final v_0, r_0, V, and $B - V$ should be included with the write-up. Also include all of the parameters that you used in the various commands.

The write-up of this part, i.e., using M67 data, should be $\sim$ 2 pages long, *excluding* your plots and parameter lists. Please add printouts of the **.m** files that you wrote to calibrate and analyze the data. The analysis is simply to compare your result with the published data on this cluster. You should include a C-M plot of your results in the write-up. In the comparison you should compare the quality of the two data sets, i.e., the amount of scatter in the plots. You don't expect a 1:1 correspondence between stars in the two data sets since they were taken using completely different techniques but the general shape of the two plots should be similar (Are they? If not, why not?). In your plot point out the main sequence, giant branch, etc.

Include a short discussion of the quality of your data, as you deduce from comparing the photometry of the stars that appear in two frames taken with the same filter (hint: look at the last column of the output of **pmerge** to backtrack which stars have good photometry in both the center and east frames). How does this compare to the purely statistical errors? If they are different, what reasons could there be for this?

Point out any stars that look like they may not belong to the cluster using your data alone and explain why. Then do it in comparison with the published data. If you are the first one to measure a cluster, you don't have the luxury of a comparison and so you need to develop criteria from your data alone.

Finally be sure to include a justification for not having to measure any extinction stars or standard stars. The easiest way to do this is to show that the three transformations involved can be replaced by one grand linear transformation. Explain what you are doing in the derivation.