N2 - Photometrie eines offenen Sternhaufens (GDL)

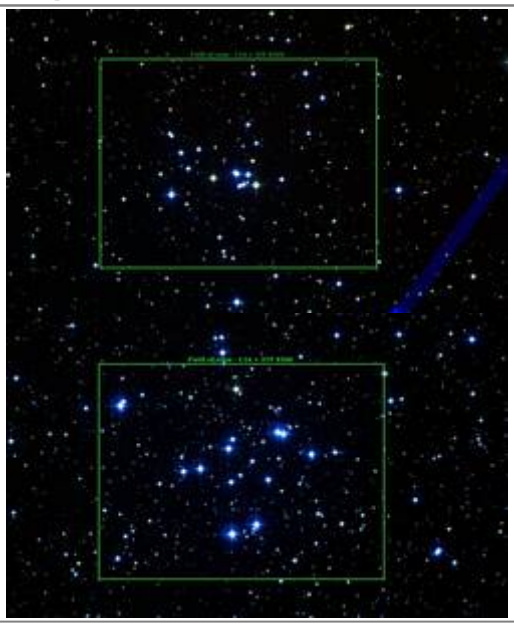
Hinweis: Dieser Artikel schildert die moderne Auswertung des N2-Versuchs mithilfe einer halbautomatischen *GDL*-Rotuine für die Reduktion von Flatfields und Darkframes sowie die Addition der Bilder pro Filter. Die klassische Version, bei der die Verschiebungen von Hand ausgemessen und manuell in eine Routine eingegeben werden, wird im Artikel Photometrie eines offenen Sternhaufens (klassisch) beschrieben.

Aufgabe

Photometrieren sie zwei offene Sternhaufen und fertigen sie Farben-Helligkeits-Diagramme (englisch: color magnitude diagram) dieser Haufen an. Ziel der Arbeit ist die Altersbestimmung (Abschätzung) der beobachteten Sternhaufen. Zur Suche nach geeigneten Sternhaufen kann u.a. Simbad genutzt werden - eine Hilfe zur Parametersuche von Simbad gibt es hier.

Kriterien, die die Sternhaufen erfüllen müssen, sind:

- Eine hohe Zahl an Sternen im Sichtfeld, die Sterndichte darf aber auch nicht zu hoch sein.
- Ein großer Teil des Sternhaufens sollte im Sichtfeld der Kamera sein.
- Der Haufen sollte nicht zu klein sein.
- Keine Starken Helligkeitsunterschiede zwischen den einzelnen Sternen im Haufen.

Gutes Beispiel	**Schlechte Beispiele**
	

Beobachtung

Der Versuch erfordert eine Nachtbeobachtung am 70cm-Teleskop des AIP oder dem OST der Uni Potsdam. Zur Vorbereitung sollte die Checkliste: Nachtbeobachtung beachtet werden. Am AIP werden in der Dämmerung (oder gegen eine weiße gleichmäßig beleuchtete Wand) Flatfields mit den

entsprechenden Filtern, sowie bei geschlossener Abdeckung (also null Belichtungszeit) Bias-Aufnahmen gemacht. Am OST können die Flatfields nach der Beobachtung mit einer Flatfielfolie aufgenommen werden. Bias-Aufnahmen sind nicht nötig, da die Kamera diesen Fehler automatisch korrigiert. Auf jeden Fall sollten außerdem Darkframes aufgenommen werden. Diese werden später zur Datenreduktion benötigt. Bei vollständiger Dunkelheit werden dann Aufnahmen der Sternhaufen durchgeführt. Um einerseits eine ausreichende Zahl an Counts zu erhalten und andererseits auch leuchtschwächere Sterne aufzunehmen, sollten die Aufnahmen mehrere Minuten dauern. Bei unzureichender Genauigkeit der Nachführung kann man auch mehrere Aufnahmen (beispielsweise von nur einer Minute Dauer) machen und später aufaddieren. Außerdem kann mit der STF8300 ein 2×2-Binning eingestellt werden um noch mehr Counts zu erhalten.

Datenreduktion mit GDL

Der Versuch kann mit *IDL* bzw. *GDL* ausgewertet werden. Eine Installationsanleitung für die freie Variante *GDL* ist hier zu finden. Eine kurze *GDL*-Einführung kann hier gefunden werden.

Ansehen der Bilder

Einloggen im Praktikumspool. Kopieren der Beobachtungsdaten (FITS-Files), inklusive Flatfield- und Bias/Darkframe-Aufnahmen aus dem Verzeichnis `~/data/<datum>` ins eigene Verzeichnis `~/data_reduction/`. Es gibt verschiedene Tools, um die im FITS-Format abgelegten zweidimensionalen CCD-Bilder (Images) zu betrachten. Beispielsweise *ds9* (neueste Version des früheren saoimage).

```
ds9 filename.fit
```

öffnet das Bild mit *ds9*. Durch Bewegen des Cursors bei gedrückter rechter Maustaste kann man Helligkeit und Kontrast variieren. Weitere Optionen (Zoom, Falschfarbendarstellungen, Drehen, Spiegeln usw.) sind per Buttons zu erreichen. Die Koordinaten der aktuellen Cursor-Position werden links oben angezeigt. Man kann auch mehrere Bilder gleichzeitig öffnen (Buttons `Frame` → `new frame`, dann weiteres File mit `File` → `open` öffnen). Mit der Blink-Option (Buttons: `Frame` → `blink`) kann man mehrere Bilder besonders gut vergleichen.

Alternativ können alle Bilder gleichzeitig geöffnet werden. Dazu wird

```
ds9 *.fit
```

eingeben. In *ds9* können die Frames wie oben betrachtet werden oder jeder Frame einzeln hintereinander (Button `Frame` → `Singe Frame`). Dabei kann zwischen den Frames durch Drücken der Tabulatortaste umgeschaltet werden.

Es sind die **brauchbaren** Beobachtungen für die weitere Bearbeitung auszuwählen, in denen die Sterne als runde Scheibchen zu sehen sind. Bilder mit ovalförmigen Sternen sind nicht zu verwenden.

Reduktions-Pipeline: Darkframes, Flatfields und Bildaddition

Um auch mit einer größeren Menge von Daten zurecht zu kommen, gibt es eine *GDL*-Routine, die pro Filter die Korrekturen für Darkframe und Flatfield durchführt und anschließend die Bilder pro Filter aufaddiert. Die Routine nimmt keine Qualitätskontrolle der Bilder vor, unbrauchbare Beobachtungen müssen also auf jeden Fall vorher aussortiert werden.

Vorbereitung

Man kopiere sich zunächst die Programmroutine `n2_add_images.pro` aus dem Verzeichnis `~/scripts/n2/` in sein lokales Arbeitsverzeichnis. Danach sollte man diese mit einem beliebigen Texteditor öffnen, um die Pfadangaben für die Bilder entsprechend anzupassen. Um eine größere Menge von Bildern bequem einlesen zu können, erwartet das Programm eine Trennung der Daten in verschiedene Unterverzeichnisse. Es muss jeweils ein Verzeichnis für die Aufnahmen pro Filter existieren sowie jeweils ein weiteres für die Flatfields und Darkframes pro Filter. Die Darkframes sind zwar filterunabhängig, unterscheiden sich aber ggf. in den Belichtungszeiten. (Mindestens zwischen Flatfields und Bildern, ggf. sogar zwischen den jeweils zu den B- und V-Flatfields gehörenden Darkframes.) Dasselbe Verzeichnis kann also nur dann für zwei Fälle angegeben werden, wenn sich die Belichtungszeiten nicht unterscheiden. Eine mögliche Verzeichnisstruktur wäre:

```
/flats/v/
/flats/b/
/images/v/
/images/b/
/darks/images/
/darks/flat-v/
/darks/flat-b/
```

Hinweis: Um Fehlermeldungen beim Einlesen zu vermeiden, sollten die Unterverzeichnisse in denen Bilder liegen selbst keine weiteren Unterverzeichnisse haben. (Dies ist beispielsweise der Fall, wenn man obiger Verzeichnisstruktur folgen, aber die Darkframes der Aufnamen direkt in /darks/ statt in /darks/images/ speichern, um sich vermeintlich ein Unterverzeichnis zu sparen.) Wichtig ist außerdem, dass niemals Bilder aus verschiedenen Filtern im gleichen Verzeichnis liegen, da das Programm sonst diese mischen und vorzeitig addieren würde. Je nachdem wie die Daten im Rahmen der Beobachtung gespeichert wurde ist also zunächst eine gewisse Umsortierung nötig. Die entsprechenden Pfade müssen dann im oberen Teil von `n2_add_images.pro` eingetragen werden und zwar relativ zu dem Verzeichnis, in das man sich `n2_add_images.pro` kopiert hat.

Ausführung der Pipeline

Die Ausführung der Routine erfolgt innerhalb von *GDL*. Dazu einfach ins Verzeichnis wechseln, in das n2_add_images.pro kopiert wurde und mittels `gdl` die *GDL*-Konsole starten. Dort muss die Routine zunächst kompiliert und dann aufgerufen werden. Das Kompilieren geschieht in *GDL* mittels

```
.compile n2_add_images.pro
```

In *IDL* kann dasselbe erreicht werden mittels

```
.rnew n2_add_images.pro
```

Die so übersetzte Programme können in beiden Fällen dann durch Eingabe des Programmnamens ausgeführt werden:

```
add_images
```

Die Ergebnisse liegen in einem neuen Unterverzeichnis namens `output`.

Datenauswertung mit GDL

Es empfiehlt sich, alle folgende Schritte nicht auf der Kommandozeile von *GDL* auszuführen, sondern ein kleines Programm name.pro zu schreiben. Dazu einen Texteditor mit der gewünschten Programmdatei öffnen, z.B.:

```
kate name.pro &
```

In diese Datei müssen nun mindestens die folgenden zwei Zeilen geschrieben werden, damit *GDL* es als gültiges Programm identifiziert:

```
pro programmname
end
```

Wie oben beschrieben wird *GDL* mittels

```
gdl
```

aufgerufen und mittels

```
.compile name.pro
```

compiliert. Das so übersetzte Programme kann dann durch Eingabe des Programmnamens

```
programmname
```

ausgeführt werden.

Zwischen den beiden bisher vorhandenen Befehlen in der Datei folgen nun alle Anweisungen zum Einlesen und Bearbeiten der Daten. Es können (und sollten) `help`-Kommandos eingestreut werden, um den Erfolg einzelner Arbeitsschritte zu überprüfen.

Einlesen der Bilder

Die *GDL*-Funktion `readfits` liest ein FITS-File ein und speichert das Bild in einem Array. Diesen Arrays sollten für die weitere Ansprache in *GDL* handliche Namen gegeben werden (z.B. `blue` für das Bild im blauen Filter):

```
blue = float(readfits('N188B1.fit'))
```

wobei offensichtlich der Name des aktuell einzulesenden Bildes (z. B. N188B1.fit) zu verwenden ist. Diese Prozedur wiederholt man mit allen brauchbaren Aufnahmen. Anmerkung: Der Datentyp der Bilder sollte zur Sicherheit in `float` geändert werden, da bei der Addition besonders heller Bilder die Größe des Datentyps `int` nicht ausreicht. Falls die Bilder nicht im selben Verzeichnis wie die pro-Datei liegt, muss der relative Pfad angegeben werden. Liegt das im Beispiel verwendete Bild z.B. im Unterverzeichnis `data/blue`, so lautet der korrekte Aufruf von readfits nun:

```
  blue = float(readfits('data/blue/N188B1.fit'))
```

Liegen die Bilder nicht in einem Unterverzeichnis des aktuellen Verzeichnisses, kann mittels `../` der Pfad auch auf jeweils auf die nächsthöhere Ebene verweisen. Im Normalfall gibt es nach erfolgreichem Einlesen der FITS-Dateien eine Ausgabe in der Konsole. Während dies am Anfang hilfreich sein kann, wirkt es mit zunehmender Dateizahl oft störend und kann daher durch den Parameter `/SILENT` unterdrückt werden:

```
  blue = float(readfits('data/blue/N188B1.fit',/SILENT))
```

Abspeichern von FITS-Dateien

Schon bearbeitete Bilder können wieder als FITS-Dateien abgespeichert werden mit dem Kommando

```
writefits, 'bildname.fit', <Bildname in GDL>

z.B.

writefits, 'blue.fit', blue
```

Übereinanderlegen der Bilder

Die automatische Pipeline aus dem ersten Schritt (Datenreduktion) liefert ein aufaddiertes Bild pro Filter, doch auch die B- und V-Bilder sind im Regelfall gegeneinander verschoben und haben durch den Zuschnitt auf den Überlappunsbereich nicht mehr zwingend die gleichen Abmessungen. Dies muss nun noch durch einen passenden Zuschnitt der Bilder korrigiert werden.

Ausmessen der Verschiebung

Zunächst ist (z.B. mit *ds9*) die relative Verschiebung der zwei aufardierten Bilder untereinander zu ermitteln. Dazu sucht man auf den beiden Bildern ein und denselben, markanten Stern (nicht zu hell, und etwa in der Mitte des Gesichtsfeldes) heraus und notiert für jedes Bild seine exakte Pixel-Position (*ds9* zeigt oben im Fenster die Position des Maus-Cursors an, nachdem man die Maus auf dem Stern platziert hat, kann man mittels der Richtungstasten auf der Tastatur pixelweise über den Stern gehen, um die Position genau festzulegen). Zur Kontrolle wiederholt man die Prozedur für einen zweiten Stern. Des Weiteren ist es für die spätere Arbeit nötig die Halbwertsbreite bzw. FWHM eines typischen Sterns des Sternhaufens zu kennen. Dies kann ebenfalls mit *ds9* ausgemessen werden (vielleicht den Mittelwert aus einigen FWHM bilden).

Schneiden der Bilder

Die linke Seite (x = 0) der am Ende neu zugeschnittenen Bilder wird durch dasjenige Bild definiert, in dem der vermessene Stern die kleinste x-Koordinate aufweist (analog für y). Angenommen, dies ist x_b (d.h. die blaue Aufnahme). Dann hat das andere Bild v einen x-Versatz von $\Delta x_v = x_{v} - x_{b}$. Das Bild habe vorher $N \times M$ Pixel. Klar, dass das Bild dann in x-Richtung auf $N - \Delta x_{v}$ Pixel verkürzt werden muss. Man beachte, dass in *GDL* die Indizierung mit 0 beginnt.

Ausführen der Verschiebung bei jedem Bild mit den entsprechenden Δx_i, Δy_i, z. B.:

```
visualcut = visual[dx_v:N − dx_v, dy_v:M − dy_v]
```

also beispielsweise:

```
visualcut = visual[13:322, 0:324]
```

(Achtung, es kann auch der Fall sein, dass bei der y-Koordinate das andere Bild die kleinere Koordinate aufweist!) Man sollte den Erfolg des Schneidens mit einem help überprüfen. In der Ausgabe der Variablen sollten die geschnittenen Bilder nun Arrays gleicher Größe sein.

Nun kann das finale v- und b-Bild z. B. mit

```
writefits, 'visual.fit', visual
```

im FITS-Format gespeichert werden. Eines davon wird zur Identifikation der Eichsterne benötigt. Die Bilder sollten ausserhalb *GDL* mit *ds9* geöffnet werden und scharfe, runde Sterne zeigen. Beim `Blinken` der Bilder sollten die Sterne nicht hin- und herhüpfen.

Identifikation der Sterne

Finden der Sterne

Die Identifikation der Sterne z.B. im Bild `badd` und das Ausmessen ihrer Helligkeit erfolgt in *GDL* durch:

```
find, badd, xb, yb, fluxb
```

Dabei sind `xb`, `yb` und `fluxb` Namen indizierter Variablen, die hier definiert werden und unter denen später die Resultate angesprochen werden können. Es folgen verschiedene Anfragen, mit denen man die Kriterien, nach denen die Sterne identifiziert werden sollen, spezifizieren muss. Zum einen ist dies die Halbwertsbreite, die man sich bereits zu Beginn notiert hat (z.B. 3 Pixel). Den Wert für die Hintergrundintensität kann man jeweils im zuvor aufaddierten und gespeicherten Bild mit *ds9* bestimmen. Für die restlichen Parameter können die default-Werte verwendet werden (einfach mit Enter bestätigen).

Diese Prozedur liefert eine Tabelle, in der unter anderem die x- und y-Position der Sterne (`xb`, `yb`), sowie deren Fluss (`fluxb`) stehen. Es sollten möglichst viele Sterne, jedoch kein “Dreck” identifiziert

werden. Sind zu wenig Sterne gefunden worden, so muss man die `find`-Prozedur wiederholen und die Identifikationskriterien verändern. Hat man z.B. einen zu großen Intensitätswert für den Hintergrund angegeben, so sind schwächere Sterne nicht identifiziert worden. Bei einer zu großen Angabe für die FWHM können mehrere Sterne als ein einzelner identifiziert worden sein; bei einer zu kleinen Standardabweichung kann es dazu kommen, dass auch “Schmutzeffekte” als Sterne identifiziert werden. Man muss also die Parameter etwas verändern und gegebenenfalls die Prozedur wiederholen. – Das Verfahren wird auf das finale blaue und visuelle Bild angewandt. Der Name des Bildes und die Indizes müssen dann natürlich für das jeweilige Bild angepasst werden.

Hat man gute Werte für die find-Parameter gefunden, ist es oftmals lästig diese bei jedem Programmdurchlauf neu eingeben zu müssen. Es bietet sich daher dann an diese mit in den Befehlsaufruf zu schreiben und den Befehl zusätzlich mit der silent-Option zu versehen. Die vollständige Syntax des `find`-Befehls lautet:

```
find, image, x, y, flux, sharpness, roundness, hmin, fwhm, roundlim,
sharplim, PRINT=print, SILENT=silent
```

Dabei sind `sharpness` und `roundness` Arrays, die (genau wie die Sternpositionen selbst) vom `find`-Befehl erst gefüllt werden, erst danach folgen die sonst über die Kommandozeile eingegebenen Parameter. Ein typisches Nutzungsbeispiel lautet:

```
find, badd, xb, yb, fluxb, sharpb, roundb, 300, 4.0, [-2.0,2.0], [0.2,1.0],
/silent
```

In diesem Fall wäre 300 der Schwellwert für die Hintergrundintensität und 4.0 der FHWM-Wert. Die weiteren Werte können als Intervall eingegeben werden, so daß hier einfach die internen Standardkriterien weiter verwendet wurden.

Hinweis: Es hat sich als günstig erwiesen, vor der `find`-Prozedur am aufaddierten Bild eine Background-Korrektur zu machen. Dabei wird zunächst das Minimum ermittelt:

```
bmin = min(badd)
```

Dieser Wert sollte zunächst mit einer kleinen Verschiebung nach unten abgezogen werden. Hat man so zum Beispiel für das Minimum einen Wert von 360 bestimmt, dann könnte die Korrektur wie folgt aussehen:

```
badd = badd - 300.
```

In die `find`-Routine kann nun ein kleinerer Wert als Hintergrundintensität (z.B. 200.) angegeben werden. Da es hierbei aber immer Schwankungen geben wird, sollte dieser nicht auf 0 gesetzt werden.

Umrechnung in Magnituden

Da im folgenden in Magnituden gearbeitet wird, müssen die Flüsse entsprechend umgerechnet werden. Man beachte, dass die Magnituden nur bis auf eine additive Konstante bestimmt sind, solange noch keine Eichung durchgeführt wurde.

```
magb = flux2mag(fluxb)
magv = flux2mag(fluxv)
```

Kreuzkorrelation der Ergebnisse

Nun müssen diejenigen Sterne herausgesucht werden, die in beiden Spektralbereichen identifiziert worden sind. Dies geschieht mit `srcor`, wobei die vorher erstellten Tabellen nach gleichen x- und y-Koordinaten durchsucht werden. Als Parameter muss die “Ungenauigkeit” der Koordinatenbestimmung angegeben werden. Im unteren Beispiel wurde eine Ungenauigkeit von 2 Pixeln gewählt, d. h. wenn die x- bzw. y-Koordinaten eines Sterns im blauen und im visuellen Bild um maximal 2 voneinander abweichen, wird der Stern noch als dasselbe Objekt identifiziert.

```
srcor, xb, yb, xv, yv, 2, ib, iv
```

`ib` und `iv` geben die Indexwerte an, die die Sterne in der mit find erstellten Tabelle für das blaue bzw. visuelle Bild hatten. Anhand dieser Indizes müssen die Positionen und Magnituden aus den ursprünglichen Tabellen ausgelesen werden. Sind z.B. 226 gemeinsame Sterne gefunden worden, so muss ein Feld (array) vom Typ `float` mit 226 Einträgen angelegt werden, in welches die sortierten Magnituden geschrieben werden können.

```
bmag=fltarr(226)
vmag=fltarr(226)
for i=0, 225 do bmag[i]=magb[ib[i]]
for i=0, 225 do vmag[i]=magv[iv[i]]
```

Diese Dimensionierung lässt sich auch mittels Variablen automatisieren. Um die Länge eines Arrays herauszufinden, gibt es den *GDL*-Befehl `size`, welcher allerdings wiederum ein Array mit verschiedenen Informationen über das übergebene Array liefert. Der vollständige Befehl, um die Anzahl der gemeinsam gefundenen Sterne zu ermitteln und in einer Variable `anz` zu speichern, lautet daher

```
anz = (size(ib,/dimensions))(0)
```

Ob dabei wie hier `ib` oder `iv` genutzt wird, ist natürlich egal, da beide Arrays gleich dimensioniert sein sollten. Statt der Zahl 226 oben kann nun stets `anz` benutzt werden, bzw. `anz-1` für den in den Schleifen benötigten Wert 225.

Vorläufiges Farben-Helligkeits-Diagramm

Nun kann das FHD (englisch: color magnitude diagram, CMD) geplottet werden, also die visuelle Magnitude über der Differenz aus blauer und visueller Magnitude. Das FHD an sich wird mit einem *Python*-Skript geplottet, für das man die Ergebisse von *GDL* in eine Datei `cmd.dat` ausgeben lässt:

```
openw, 1, 'cmd.dat'
for i=0,225 do printf,1,i,(bmag[i]-vmag[i]),vmag[i]
close, 1
```

Die 1 steht für eine frei wählbare Kanalnummer, die dabei eindeutig der zu öffnenden Datei (hier `cmd.dat`) zugeordnet ist. Sollte diese Nummer für Probleme beim aufruf des *GDL*-Programms sorgen, ist sie möglicherweise schon anderweitig vergeben und es sollte eine andere Nummer gewählt werden. 225 ist hier die Zahl der gefundenen gemeinsamen Sterne minus 1 (*GDL*-Zählung!). Um nun das vorläufige FHD zu plotten kopiert man sich aus dem Ordner `~/scripts/n2/` die Datei `plot_cmd.py` in sein aktuelles Arbeitsverzeichnis. Dieses sollte man dann mit dem Editor *kate* öffnen und zum Beispiel den Plot-Bereich für die x- und y-Koordinate (`xRangeMin`,`xRangeMax`,`yRangeMin`,`yRangeMax`) sowie Name des zu erstellenden Plots (`filename`) und den Namen des Sternhaufens (`nameOfStarcluster`) anpassen.

```
##########################################################################
######
############################ Script Parameters
###################################
##########################################################################
######

# name of cmd data file
CMDFileName = "cmd.dat"
# names of output file
filename    = "?"
# filetype of output, supported filetypes: ps, png, gif, svg, tex
filetype    = "png"
# name of the star cluster
nameOfStarcluster = "?"
## x and y range to plot (change according to your data)
# range of the x coordinate to plot [xRangeMin:xRangeMax]
xRangeMin = "?"
xRangeMax = "?"
# same for y
yRangeMin = "?"
yRangeMax = "?"
# number of the V column and the B-V column in the 'cmd.dat' file (default:
columnB_V = "2" and columnV = "4")
columnB_V = "2"
columnV   = "4"
```

Nach dem Speichern dieser Datei, kann sie von der Konsole aus mit dem Befehl

```
python plot_cmd.py
```

ausführen. Dabei sollte man sich auf der Konsole im gleichen Ordner befinden wie die Datei. Bei Erfolg sollte das vorläufige FHD dann in diesem Ordner als Grafik zu finden sein. Über die Variable `filetype` kann auch das zu erstellenden Dateiformat gewählt werden.

Alternative: Man kann sich den Plot auch mit *GDL* erstellen:

```
plot, bmag-vmag, vmag, psym=1, yrange=[-17,-25]
```

Mit `yrange` bzw. `xrange` kann ein Plotausschnitt gewählt werden. Dies ist nötig, um die y-Achse umzudrehen (größere Magnitude bedeutet kleinere Helligkeit!). Mit `psym` kann das Symbol der

Datenpunkte bestimmt werden.

Kalibration

Die Helligkeiten (Magnituden) sind bislang nur bis auf eine Verschiebung (additive Konstante, den sogenannten Zeropoint) bestimmt. Die Eichung (Kalibration) stellt ein erhebliches Problem dar. Es bieten sich im Prinzip zwei Möglichkeiten an:

1. Alternative

Man findet auf der Aufnahme Sterne, dessen B- und V-Magnituden bekannt sind. Dann muss man diese Sterne in der Tabelle identifizieren und die Werte der gesamten Tabelle so verschieben, dass die identifizierten Sterne die Sollwerte erhalten.

Sternfeld finden

Zur Eichsternidentifikation gibt es mehrere Möglichkeiten:

Am besten (und daher empfohlene Variante) geht das mit der Simbad-Datenbank. Hierfür gibt man unter `basic search` den Objekt-Namen (also den des Sternhaufens) ein, sucht nach dem Sternhaufen und klickt in der sich öffnenden Übersicht rechts auf `AladinLite`. Dann muss man das aufgenommene Feld wiedererkennen. Zum Vergleich betrachtet man dazu eine unserer Aufnahmen mit *ds9*

```
ds9 filename.fit
```

Unter Umständen muss unsere Aufnahme gedreht und / oder gespiegelt werden.

Mögliche Eichsterne identifizieren

Hat man das Feld identifiziert, suche man sich möglichst 5-6 Vergleichssterne in der Mitte des Feldes, für die B- und V-Helligkeit bekannt sind und notiere deren Helligkeiten für später. Die Magnituden lassen sich auf unterschiedlichen Wegen ermittel:

1. Am einfachsten geht es in dem man in der *Aladin-Lite*-Ansicht rechts unter `Catalogues` die Option `SIMBAD` auswählt. Hierdurch wird ein Overlay aktiviert, welches alle in *Simbad* aufgeführten Objekte zeigt. Klickt man auf eins der Objekte werden links in einer Tabelle alle *Simbad* bekannten Parameter aufgeführt. Scrollt ran runter findet man dort auch die Magnituden. Nicht für alle Objekte sind auch Magnituden verfügbar.
2. Hat man *JAVA* installiert kann man auch das `Aladin Java Applet` verwenden, welches ähnliche Funktionen bereit hält wie *Aladin-Lite*. In dem Applet können die hervorgehobenen Objekte angeklickt werden und so zusätzliche Information über diese Objekte, unter anderem bekannte Magnituden, abgerufen werden.
3. Alternativ kann man z.B. in der Simbad-Übersicht zu dem Sternhaufen durch einen Klick auf die Option `query around` (with radius...) eine Übersicht aller Objekte im Sternfeld bekommen.

Wählt man weiterhin die Option `plot this list of objects`, erscheint eine schematische Karte in der die dort markierten Sterne anklickbar sind. Wenn bekannt, bekommt man dann deren Namen und B- und V -Werte angegeben.

4. Man kann aber auch die am besten belichtete visuelle Aufnahme mit dem Digitized Sky Survey der betreffenden Gegend vergleichen und den Bildausschnitt identifizieren. Sodann lässt man sich als Overlay Katalogsterne der wichtigsten Kataloge (Bonner Durchmusterung und Henry Draper (HD) Catalogue) anzeigen (man beachte die Scrollbalken in den entsprechenden Menufenstern!). Hat man glücklicherweise Katalogsterne in dem aufgenommenen Feld, kann man mit Hilfe der Simbad-Datenbank vielleicht deren B- und V-Magnituden in Erfahrung bringen, die dann zur Eichung verwendet werden können.

Nun gilt es die soeben identifizierten Vergleichssterne unseren eigenen Ergebnissen zuzuordnen. Zum darstellen der Sterne, die mit dem unserem *GDL*-Programm identifizierte wurden, steht ein *Python*-Skript (`plot_stars.py`) im Verzeichnis `~/scripts/n2/` zur Verfügung. Dieses sollte für die Nutzung in den Ordner kopiert werden, in dem auch das *GDL*-Programm zur Auswertung des jeweiligen Sternhaufens liegt. Danach kann man es zum Beispiel mit dem Text-Editor *kate* öffnen und im oberen Bereich

```
 # enter full path to your reduced fits-file
 visualPath = 'vadd.fit'

 # enter full path to the star position file (format: index, x position, y
position, b magnitude, v magnitude)
 starPath = 'stars.dat'

 # output file:
 outfile = 'starmap.png'
```

anpassen. Hierbei müssen eigentlich nur die Pfade zu einem der addierten und gespeicherten Bilder im FITS-Format (Variabel: `visualPath`) und der Pfad zu einer Datei angegeben werden, in der die Sternpositionen und die jeweiligen Helligkeiten im V- und B-Band angegeben sind (Variable: `starPath`). Dies erreicht man im *GDL*-Skript, indem man wie im vorigen Abschnitt, diese Größen in eine Datei schreibt. Diese könnte zum Beispiel `stars.dat` heißen und könnte z.B. mittels folgender Zeilen erstellt werden:

```
openw, 2, 'stars.dat'
for i=0,anz-1 do printf,2,i,xv[iv[i]],yv[iv[i]],bmag[i],vmag[i]
close, 2
```

Nach dem Speichern des *Python*-Skriptes, kann man dieses mit folgendem Befehlt von der Konsole aus ausführen:

```
python plot_stars.py
```

Dabei sollte man sich in der Konsole im gleichen Ordner befinden, in den man das Skript kopiert hat. In diesem Ordner findet sich nun eine Grafik mit dem Namen `starmap.png`. **Diese Abbildung bitte auch an das Protokoll anhängen.** In dieser Grafik sind das Sternfeld des aufaddierten Bildes im Negativ, sowie alle identifizierten Sterne als Markierung zu erkennen. Dieses Bild kann man nun mit den anfangs erwähnten Eichsternen vergleichen. Hat man im Bild einen Stern identifiziert, zu dem man in *Simbad* die Helligkeiten im V- und B-Band feststellen konnte, so kann man die Nummer der Markierung im Bild `starmap.png` notieren und in der Datei `stars.dat` die entsprechenden

Helligkeiten des Sterns in den eigenen Aufnahmen unter dieser Nummer finden. Die Sterne in der Datei `stars.dat` sind auch nach aufsteigenden Y Werten sortiert, sodass man sich auch die Position des Sterns im Bild heraussuchen kann, welches als Hintergrund für die star map verwendet wurde. Diese Methode ist vielleicht besser geeignet, wenn man sehr viele Sterne in seinem Bild identifiziert hat und es somit sehr unübersichtlich ist.

Aus der Differenz dieser Werte eines Sterns mit den *Simbad*-Daten ergibt sich die Eichkonstante. Dieser Vorgang wird für 5-6 Sterne wiederholt und anschließend wird der Mittelwert der Eichkonstanten gebildet, wobei die Streuung 0.1 mag nicht übersteigen sollte.

2. Alternative

Man hat auf separaten Aufnahmen, jedoch in möglichst guter zeitlicher und räumlicher Nähe einen Vergleichsstern aufgenommen, dessen Magnituden bekannt sind. Dann müssen diese Aufnahmen in genau der gleichen Weise ausgewertet werden wie die Sternfeldaufnahmen. Zusätzlich ist in diesem Fall zu beachten, dass die Magnituden auf eine einheitliche Belichtungszeit bezogen werden. Hat man also die aufaddierten visuellen Bilder insgesamt 440 Sekunden belichtet, dividiert man `fluxv` durch 440 (vor der Umrechnung in Magnituden) und bezieht so den Fluss auf 1 Sekunde (`fluxb = fluxb / 440`). Das gleiche geschieht mit den Belichtungen des Vergleichssterns. Am Ende wird wieder die Tabelle der Magnituden so um additive Konstanten verändert, dass sich für den Vergleichsstern die Sollwerte für `bmag` und `vmag` ergeben.

Nachdem nun die Eichwerte für das FHD bestimmt wurden, kann man diese im *GDL*-Programm von den berechneten vorläufigen Magnituden abziehen:

```
magbkal = magb - beich
magvkal = magv - veich
```

Mit diesen Werten könnt ihr nun ein kalibriertes FHD mit euren scheinbaren Magnituden plotten. Hierfür brauchen wir noch die Farbe, daher ergänzen wir noch die Zeile:

```
magbminusv = magbkal - magvkal
```

Rötung & absolute Magnituden

Beim Vergleich eures scheinbaren FHDs mit der Literatur wird euch allerdings auffallen, dass die Hauptreihe verschoben scheint. Dieser entsteht durch die interstellare Materie, die sich auch zwischen den Sternen unserer Milchstrasse befindet. Wie alle andere Materie auch, kann sie durch Licht angeregt werden. Diese Energie gibt sie später wieder ab, allerdings nicht auf derselben Wellenlänge, sondern niederenergetischer, also auf der roten Seite des Spektrum. Man spricht daher bei diesem Effekt auch von Rötung, nicht zu verwechseln mit der Rotverschiebung:

$(B-V)_{0} = (B-V) - E_{(B-V)}$

$V_{0} = V - A_{V}$

Die Rötung wird mathematisch durch den Ausdruck $E_{(B-V)}$ beschrieben, den man auch als

Farbexzess bezeichnet. Er beschreibt exakt den Unterschied zwischen dem hier gemessenen $(B-V)$ und dem ungeröteten, ursprünglichen Wert $(B-V)_{0}$. Die Rötung betrifft allerdings auch die Ordinate, wo der Korrekturterm mit A_{V} bezeichnet wird. Allerdings sind A_{V} und $E_{(B-V)}$ nicht unabhängig voneinander, sondern lassen sich mit R_V ineinander umrechnen:

$A_{V} = R_V \cdot E_{(B-V)}$

In der Sonnenumgebung wird R_V oft auf 3.1 gesetzt (Seaton 1979). Damit muss also nur noch der Wert für $E_{(B-V)}$ bekannt sein, um die nötigen Korrekturen komplett durchzuführen, indem man die obigen Formeln in *GDL*-Code übersetzt. Entsprechende Werte findet ihr via *Simbad* über die mit einem Objekt assoziierten Paper (Veröffentlichungen) oder direkt über Datenbanksuche bei *VizieR*. Vergesst in jedem Fall nicht, den benutzten Wert und die genaue Quelle (d.h. in der Regel das direkt benutzte oder zu einem *VizieR*-Eintrag gehörende Paper) in eurem Protokoll anzugeben.

Abschließend sollten noch die scheinbaren Magnituden in absolute Magnituden konvertiert werden, damit später ein Vergleich mit Isochronen möglich ist. Hierfür muss das entsprechend Distanzmodul oder die Entfernung des Sternhaufens aus Papern (Veröffentlichungen) herausgesucht werden und die entsprechende Korrektur vorgenommen werden.

Um nun das endgültige FHD mit dem *Python*-Skript zu plotten, ist es erneut nötig, die Daten in eine Datei zu schreiben. Dazu kann die bereits verwendete “cmd.dat” überschrieben werden:

```
openw, 3, 'cmd.dat'
for i=0,anz-1 do printf,3,i,magbminusv[i],magbkal[i],magvkal[i]
close, 3
```

FHD plotten

Ausgabe mit Python inklusive Isochronen

Wie oben bereits beschrieben, kann die geänderte Datei “cmd.dat” mit dem Skript `plot_cmd.py` wie folgt geplottet werden:

```
python plot_cmd.py
```

Das Skript `plot_cmd.py` bietet auch die Möglichkeit an Isochronen zu plotten. Diese müssen zunähst im Internet recherchiert werden, was jedoch kein großes Problem darstellen sollte, da viele Webseiten die sich Sternentwicklung beschäftigen diesen Service anbieten. Die Variablen des Skripts müssen entsprechend der Struktur der Isochrondateien angepasst werden. Das Skript erwartet darüberhinaus, dass die einzelnen Isochronen jeweils in eigenen Dateien in einem gemeinsamen Ordner abgelegt wurden.

```
##########################################################################
######
############################ Script Parameters
#####################################
##########################################################################
######
```

```
# name of cmd data file
CMDFileName = "cmd.dat"
# names of output file
filename    = "?"
# filetype of output, supported filetypes: ps, png, gif, svg, tex
filetype    = "png"
# name of the star cluster
nameOfStarcluster = "?"
## x and y range to plot (change according to your data)
# range of the x coordinate to plot [xRangeMin:xRangeMax]
xRangeMin = "?"
xRangeMax = "?"
# same for y
yRangeMin = "?"
yRangeMax = "?"
# number of the V column and the B-V column in the 'cmd.dat' file (default:
columnB_V = "2" and columnV = "4")
columnB_V = "2"
columnV   = "4"

# size of the plotted output figure in cm, default 8cm x 8cm
size_x = "?"
size_y = "?"
#------------------------------------------------------------------------
#####  OPTIONAL: Isochrone fit  #####
# path to the directory with the isochrone data files
isoDir = "?"
# B-V or B column in the isochrone data file?
ISOcolumntype = "B-V"
# number of the B/(B-V) column and the V column in isochrone data file
ISOcolumnB = "?"
ISOcolumnV = "?"
```

Alternative mit GDL

Wie oben beschrieben kann das FHD auch mittels *GDL* geplottet werden:

```
plot, magbkal-magvkal, magvkal, psym=1, yrange=[18,12]
```

Anpassung des y-Ranges und der Beschriftung des Plots ist durch weitere Parameter möglich, die dem obigen Kommando angefügt werden können, z.B.:

```
title="CMD of NGC 9999", xtitle="B-V in mag", ytitle="V in mag"
```

Wenn man mit dem Diagramm zufrieden ist, kann es in eine PostScript-Datei umgewandelt werden.

```
set_plot, 'ps'
device, filename='CMD.ps'
plot, magbkal-magvkal, magvkal, psym=1, yrange=[18,12]
```

```
device, /close
```

Um wieder das Plotfenster auf dem Bildschirm zu aktivieren:

```
set_plot, 'x'
```

In einem neuen Fenster kann man sich die PostScript-Datei mit z.B. mit *Ghostview*

```
gs CMD.ps
```

ansehen und über den print-Befehl ausdrucken.

Protokoll

Legen Sie dar, wie sich Kugelsternhaufen von offenen Sternhaufen unterscheiden, mit besonderer Betonung auf den Charakter des beobachteten Objektes. Gehen Sie darauf ein, wie sich insbesondere Sternhaufen von allgemeinen Sternansammlungen unterscheiden. Erklären Sie kurz den Lebensweg eines Stern und gehen Sie dabei auf die Darstellung des Hertzsprung- Russell-Diagramms und des Farben-Helligkeits-Diagrammes ein. Erläutern Sie die Altersbestimmung eines Sternhaufens mittels Analyse des Abknickpunktes und Isochronen und vergleichen Sie Ihr FHD mit bekannten Diagrammen. Ermitteln Sie das Alter des beobachteten Sternhaufens mittels Abknickpunkt und Vergleich mit Isochronen. Vergleichen sie ihre Ergebnisse mit der Literatur und diskutieren sie gegenbenfalls Abweichungen. Es ist ein praktikumsübliches Protokoll anzufertigen.

Übersicht: Praktikum

From:
https://polaris.astro.physik.uni-potsdam.de/wiki/ - **OST Wiki**

Permanent link:
https://polaris.astro.physik.uni-potsdam.de/wiki/doku.php?id=de:praktikum:photometrie

Last update: **2023/09/19 06:51**