Csillagászati Laboratórium II.
3. óra: Ismerkedés az *IRAF*-fal

1. Az *IRAF* alapjai

Az *IRAF* programcsomag fejlesztése az amerikai National Optical Astronomy Observatory (NOAO) intézetében kezdődött az 1980-as évek elején. Azóta nagyon sok új csomaggal bővült. Érdemes az *IRAF* honlapján barangolni (http://iraf.noao.edu) és az *IRAF* manueleket olvasgatni.

A programcsomag nagyon összetett, rengeteg képműveletet, fotometriát, spektroszkópiát ... stb. lehet segítségével végezni. Az *IRAF*-nak saját terminálja (az úgynevezett `xgterm`) van. Meg lehet adni más terminháltípust is a telepítés során, de ez a legáltalánosabb. (Az *IRAF*-nak a konfigurációjával illetve telepítésével nem ismerkedünk meg, mivel jelentősen túlmutatna a labor célján.) A programcsomag elindításához így először egy `xgterm` terminálra van szükségünk. Ezt a

```
$ xgterm &
```

paranccsal indíthatjuk el. Egy `bash` shell-hez nagyon hasonló terminált kapunk. Érdemes külső jegyekkel megkülönböztetni a terminált, hogy véletlenül se tévesszük össze egy `bash` shellel. Például adjunk neki fekete hátteret illetve zöld betűket:

```
$ xgterm -bg black -fg green -fn 10x20 &
```

Az `-fn` kapcsoló („switch") illetve a `10x20` paraméterrel a betűk méretét vettük nagyobbra, a jobb láthatóság kedvéért (illetve, hogy 10 évnyi csillagászkodás után ne kényszerüljünk szemüveg viselésére). Az *IRAF*-fot abból a könyvtárból kell indítanunk, ahonnan saját magunknak létrehoztuk az *IRAF* `login` fájlunkat (az `mkiraf` paranccsal). Ez a `login.cl` nevű fájl. Általában a könyvtár a `home`-unk. A program indításához az `xgterm` terminálban adjuk ki a

```
$ cl
```

parancsot (`&` jel nélkül, ugyanis ezt nem a háttérben akarjuk futtatni). A program induláskor a következő informatív szöveget írja ki a standard kimenetre:

```
NOAO PC-IRAF Revision 2.12.1-EXPORT Fri Jul 12 15:54:09 MST 2002
This is the EXPORT version of PC-IRAF V2.12 supporting most PC systems.

Welcome to IRAF.  To list the available commands, type ? or ??.  To get
detailed information about a command, type 'help command'.  To run a
command or load a package, type its name.  Type 'bye' to exit a
package, or 'logout' to get out of the CL.  Type 'news' to find out
what is new in the version of the system you are using.  The following
commands or packages are currently defined:

  dataio.     images.     lists.      obsolete.   proto.      system.
  dbms.       language.   noao.       plot.       softools.   utilities.

cl>
```

Az *IRAF*-nak a promptja így néz ki, mint látjuk, hogy „`cl>`". Innentől kezdve az *IRAF*-os parancsokat így jelezzük, hogy látható legyen, hogy *IRAF* parancsról van szó, nem pedig `bash` parancsról. Mint láthatjuk, a `?` illetve a `??` parancsra az aktuális csomagokat illetve futtatható programokat (`TASK`-okat) listázza ki. A futtatható *TASK*-okról nagyon bő leírásokat kaphatunk a

```
cl> help tasknév
```

parancs segítségével. A `tasknév` helyére természetesen az aktuális *TASK* nevét kell beírni. Ha netán elírtunk valamit, akkor nem a `Backspace` billentyűvel, hanem a `Del`-lel tudunk törölni. Parancsot visszahívni pedig egy `e + enter` lenyomása után tudunk visszahívni a szokásos módon, a ↑-lal.

Az *IRAF* csomagokra és a csomagokon belül található *TASK*-okra bomlik. Az egyes csomagokba a csomag nevének a begépelésével léphetünk. Belépéskor az *IRAF* kilistázza a csomagban található további csomagokat illetve ha van, akkor *TASK*-okat. A csomagokat a név végén található ponttal jelzi nekünk az *IRAF*. Ha valamelyik csomagba belépünk (mondjuk pl. a `noao.` csomagba), akkor a prompt megváltozik, méghozzá mindíg az adott csomagra jellemző két betűre.

```
cl> noao
      artdata.      digiphot.     nobsolete.    onedspec.
      astcat.       focas.        nproto.       rv.
      astrometry.   imred.        observatory   surfphot.
      astutil.      mtlocal.      obsutil.      twodspec.
no>
```

A `noao.` csomagba belépve például csak további csomagokkal találkozunk. Az *IRAF*-nak jó tulajdonsága, hogy nem kell kiírni teljesen minden parancsot, csak addíg kell, míg nem egyértelmű, hogy melyik parancsot szeretnénk beírni. Így például a `noao.` csomagba a

```
cl> noa
```

parancs kiadásával is beléphetünk. Visszalépni a csomagokból a

```
no> bye
```

paranccsal tudunk. Ekkor megint kilistázza az ott található *TASK*-okat. Az *IRAF*-ból a

```
cl> logout
```

paranccsal tudunk kilépni (vagy az egyszerűbb `log` paranccsal).

1.1. Képmegjelenítés

Természetesen a célunk az észlelések során készített képek feldolgozása, illetve megjelenítése. Az *IRAF* a képek megjelenítéséhez a `ds9` nevű külső programot használja. Hogy külső programot indítunk azt jeleznünk kell az *IRAF*-nak, méghozzá egy `!`-jel megadásával a parancs elé.

```
cl> !ds9 &
```

Mivel vissza szeretnénk kapni az *IRAF* promptot, ezért ide is kell a `&` jel. Kis várakozás után megjelenik a jól ismert `ds9` ablak. Képeket a

```
cl> display kép.fits #
```

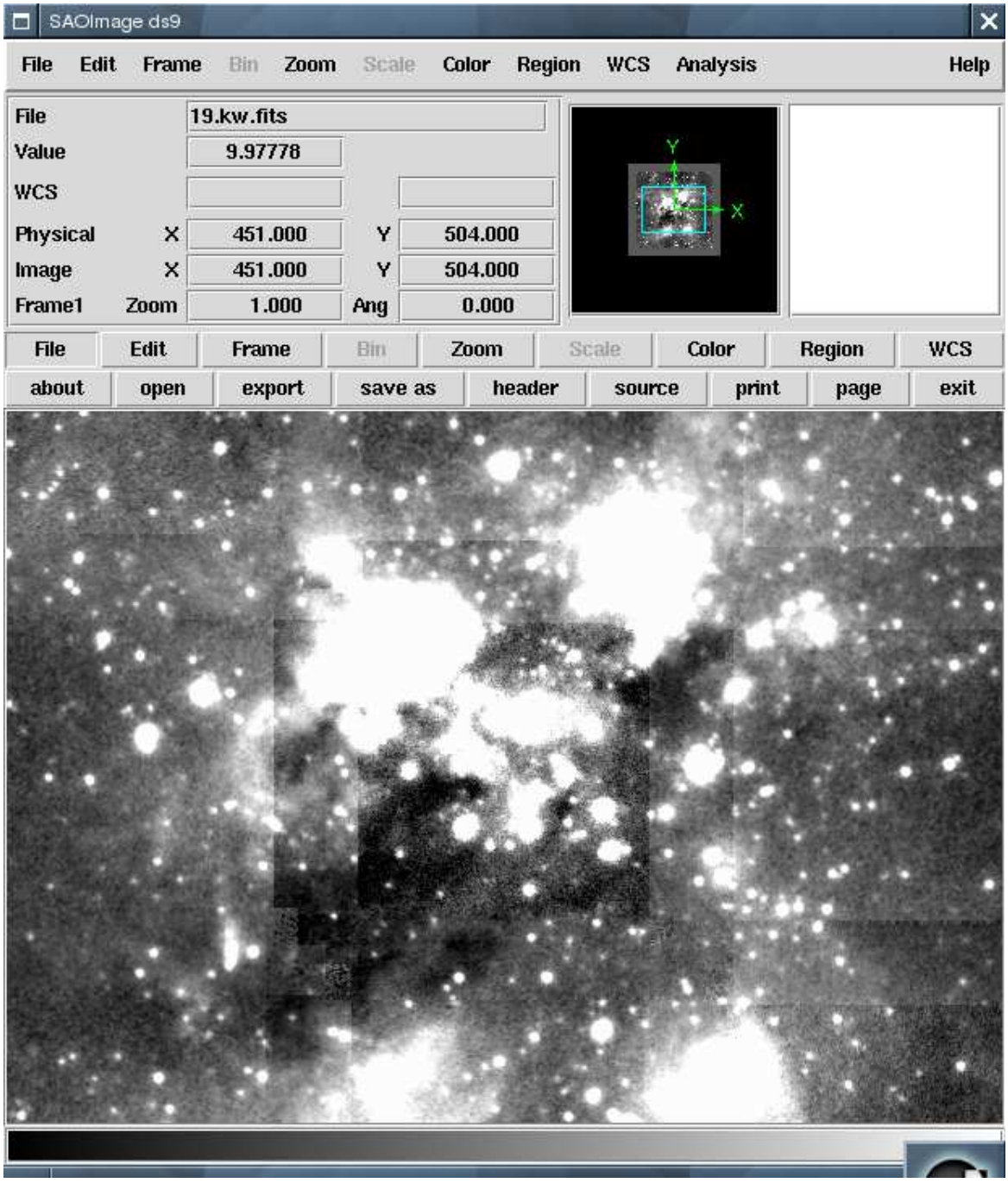
parancs kiadásával lehet ábrázoltatni. A `kép.fits` a kép neve, a `#` pedig egy sorszám, mely megadja, hogy a `ds9` hanyadik „keretébe” tegye a képet az *IRAF*. Mivel nagy valószínűség szerint (és ajánlottan) a képeink nem a `home` mappánkban van, ezért a kép elérési útvonalát meg kell adni, vagy abba a könyvtárba érdemes menni, ahol a képek vannak. A `cd` parancs értelmezett belső parancs, így például:

```
cl> cd Munka/ngc2126
cl>
```

Mint látjuk, a mappát nem jelzi az *IRAF*, így azt fejben kell tartani. Ha valami miatt mégis elbizonytalad-nánk, akkor a

```
cl> !pwd
```

segítségével megtudhatjuk, hogy hol is vagyunk. A `ds9`-nek sok „kerete” (frame) van, de nem érdemes túl sokat használni, ugyanis telítődik a memóriája. Magáról a képről nagyon sok információt megtudhatunk a `ds9` haszálatával. Az 1. ábrán látható a `ds9` ablaka. A felső sorban fő opciók találhatóak (`File, Edit, Frame ...`), az alsó sorban pedig ezeknek az opciói (pl. `about, open, export ...`), melyek változnak a felső opció függvényében. A leggyakrabban a `Frame`, `Zoom` és `Scale` funkciókat használjuk. Mivel ezek eléggé logikusak, ezeket nem magyarázzuk tovább. A `ds9` rengeteg dologra használható, scriptekkel programozható, külön órát lehetne szentelni ismertetésére. Az idő rövidsége miatt ezt nem tesszük meg.



1. ábra. A ds9 ablaka

2. Képek letöltése, kitömörítése

A félév során kiértékelendő képeket a labor honlapjáról töltsétek le (http://petra.hos.u-szeged.hu/~gaspi/). Ha esetleg a képeket nem a `titan`-ra töltöttétek le, akkor azokat másoljátok fel a szerverre az `scp` program használatával. (Külön oktatói dícséretet kap, aki a `wget` terminálos parancs segítségével tölti le a képeket.)

```
$ scp ngc2126.tgz user@titan.physx.u-szeged.hu:
passwd:
```

Most külön kiemelendő az `scp` program használata, ugynis nagyon fontos program. A megszokott másolási szintaxist követi a program, azaz a „honnan-hová" szintaxist. Mint láthatjuk, az aktuális mappában található `ngc2126.tgz`-t másoljuk fel a bizonyos felhasználónevű user `home` konyvtárába, a `titanra`.

Lépjünk be a `titan`-ra. Ezt az `ssh`-val tesszük meg. Mivel szeretnénk grafikus kezelői felületet is áthozni, ezért ezt jelezni kell az `ssh`-nak a `-X`-szel.

```
$ ssh -X user@titan.physx.u-szeged.hu
passwd:
```

Ha először jelentkezünk be az adott gépről egy másikra, akkor megkérdezi, hogy elraktározza-e az ssh kulcsát a kommunkációnak a két gép között. Ezt egy „yes" begépelésével fogadjuk el.

A korábbi másolás miatt az `ngc2126.tgz` ott van a `home` könyvtárunk gyökerében. Mivel a félév során nagyon sok adat illetve kép fog felgyülemleni, ezért hozzunk létre egy új könyvtárat az egész féléves munkánknak, és helyezzük át abba a könyvtárba a `tgz` fájlt, majd lépjünk bele abba a könyvtárba.

```
$ mkdir labor2;mv ngc2126.tgz labor2/;cd labor2
```

Az összetömörített fájlt most tömörítsük ki. Láthatóan ez egy többszörös tömörítés, egyszerre van `tar`-olva illetve `gzip`-elve a fájl. A `tar`-nak a manueljét olvasgatva rájöhetünk, hogy milyen paranccsal lehet egyszerre a két kitömörítést végrehatjani.

```
$ man tar
```

A manuelből „q" lenyomásával léphetünk ki. Sok okfejtés után a helyes kitömörítési szintaxisnak a

```
$ tar xzvf ngc2126.tgz
```

adódik. Ennek határára kitömörítődik az összes kép. Nézegessük kicsit a fájlokat és értelmezzük őket. Összesen 164 darab fájlt láthatunk a kitomörített mappákban. Ezek további `*.gz`-ek. Az egyedi tömörítések értelme az, hogy az egyik sérülése esetén a többi kép sértetlen maradhat. Az összes képre természetesen nem fogunk egyesével kitömöríteni, hanem a

```
$ gunzip *.gz */*.gz
```

parancs segístégével egyszerre az összeset kitömorítjük. Jelenleg azon kívül, hogy ezek `fit` fájlok, nem túl sok mindent tudunk róluk. A `titan`-ra telepítve van a WCSTools nevű programcsomag, melynek segítségével `konsole`-on belülről is megtudhatunk információt a képekről. Próbáljuk ki a következő parancsot:

```
$ imhead t329.fit
```

A `konsole`-ra a parancs kiírja a kép minden információját, mely a fejlécben megtalálható. Számunk lényeges információk az `OBJECT, RA, DEC, FILTER`. Az `imhead` az *IRAF*-on belül is működik, csak kicsit más paraméterezéssel, ott ugyanezt a teljes fejlécet a

```
cl> imhead t329.fit l+
```

parancsra kapjuk meg. A lényeges információkat listaszerűen a `gethead` paranccsal kaphatunk meg.

```
$ gethead RA DEC OBJECT FILTER *.fit
```

Láthatjuk, hogy a képek között található flat-kép, az M67-ről illetve a NGC 2126-ról készült felvétel különböző szűrővel.

3. Redukálás, a csillagászati képkorrekciók - elmélet

Mivel ezen észlelés során csak flat-képeket készítettünk, ezért a felvételekre csak a flat-korrekciót hajtjuk végre. Ennek ellenére itt részletezem a bias, dark illetve sky korrekció lényegét is, illetve, hogy hogyan kell végrehajtani őket. Az eljárásukat végrehajtásuk logikus sorrendjében részletezem.

3.1. A bias(zero)-korrekció: az alapszint beállítása

A bias korrekció a kamerának a saját nullaszintű alapzaját hivatott korrigálni. Ha a dark képeink ugyanannyi ideig készültek, mint a korrigálandó csillagászati felvételek illetve flat-képek, akkor erre a korrekcióra nincs szükség, ugyanis a dark kép már maga is tartalmazza az alapszintű zajt. Viszont ha eltérő idejűek, és netán a dark képeket így kétszerezni kéne, akkor az új korrekciós dark képeknél már kétszeres alapszint lenne, és így hamis eredményekhez jutnánk.

Azaz egy képnél a pixelenként mérhető intenzitáshoz hozzájárul a kamera zaja. Ez az alapszintből, illetve egy, az idővel lineárisan növekvő tagból (dark) áll.

$$\mathrm{I}_{\mathrm{N}(i,j)}(t) = \mathrm{I}_{\mathrm{B}(i,j)} + \mathrm{I}(t)_{\mathrm{D}(i,j)} \cdot t \tag{1}$$

/N - Noise (zaj); B - Bias (alapszint); D - Dark (sötétáram); t - idő; i,j - pixelkoordináták/

A bias korrekciós kép elkészítéséhez több bias képet készítünk minden észlelési nap, és ezeket átlagoljuk naponként. Egy nap alatt az alapszint értéke nem szokott jelentősen megváltozni. Érdemes az észlelés elején és végén is készíteni korrekciós képeket. Az átlagolt képet a dark- illetve flat- és objektum-képekről is levonjuk. Azaz a többi képnél mérhető maradékintenzitás értéke egy i,j pixelkoordinátánál n darab bias-kép esetén (D - dark, O - objektum, F - flat):

$$\mathrm{I}^{bias}_{\mathrm{D,F,O}(i,j)} = \mathrm{I}(0)_{D,F,O(i,j)} - \frac{\sum_{k=1}^{n} \mathrm{I}_{B(i,j),k}}{n} \tag{2}$$

3.2. A dark-korrekció: a sötétáram levonása

A dark-korrekció nagyon hasonlóan zajlik a bias-korrekcióhoz. Egy nagy különbséggel bír a dark-korrekció a bias-korrekcióhoz képest. Mint a 1. képletben látható, a dark-kép értéke időfüggő. A dark-képet gyakorlatilag ugyanúgy csinálják, mint bármelyik másik képet, csak egy nagy különbséggel. A kamerának a rekesze („shutter") ekkor zárva van, azaz semmi fény nem jut bele a kamerába, így az ekkor kapott kép megadja a kamerának az érzékenységét a termikus zajokra. Logikusan, minél hosszabb ideig exponálunk, annál több „termikus elektron" esik bele a CCD kamera pixel-csapdáiba. Mivel a CCD kamerák nagy dinamikai tartományon lineárisan viselkednek, ezért a dark-képeket a szükséges expozíciós időhoz lehet skálázni, természetesen az alapszintű korrekciók elvégzése után. A korrigáláshoz használt átlagolt dark-kép levonása utáni maradékintenzitást a flat- illetve objektum-képek esetén a 2. képlethez hasonlóan lehet felírni ...

$$\mathrm{I}_{\mathrm{F,O}(i,j)} = \mathrm{I}^{bias}_{\mathrm{F,O}(i,j)} - \frac{\sum_{l=1}^{m} \mathrm{I}^{bias}_{D(i,j),l}}{m} \tag{3}$$

3.3. A flat-korrekció: a kamera hibáinak korrigálása

Ha megnézünk egy nyers CCD képet, akkor azon akár jól kivehető struktúrák találhatóak. Ezek több dolognak tudhatóak be. A CCD chipet takaró vékony üveglemez, illetve a szűrő idő alatt szennyeződik. Ezek a szennyeződések zavarnak a képalkotásban, illetve intenzitásbeli különbséget okoznak a CCD különböző területein. Emellett a kamerának a különbözö területei alapból is különböznek az intenzitásbeli érzékenységükben (különböző szintű a félvezető szennyezettsége). Ezeket a hibákat hivatott korrigálni a flat-kép. A flat-képet tipikusan két módszerrel szokták előállítani. A csillagászok egyik fele az egyik módszerre, a másik fele pedig a másikra esküszik. A célja mindkettőnek azonos: olyan képek készítése, melyek homogénen megvilágított területről készülnek. Az egyik módszer esetén a szürkületi (esti vagy hajnali) égboltot fotozzák. Ezt hívják *„sky-flat"*-nek. A másik módszer kicsikét földhöz ragadottabb. Ennél a kupola belsejében található (egyenletesen megvilágított?) fehér lapot fotózzák a megfigyelők a távcsővel. Ezt hívják *„dome-flat"*-nek. Természetesen az első módszer nagyon nagy távcsöveknél nem működik, annyira érzékenyek.

A flat-korrekcióhoz használt kép készítése kicsikét bonyolultabb, mint a korábbiak. Magát a korrekcióhoz használt képeket először medián (vagy szimpla) átlagolják. Az így kapott képet viszont skálázni kell, ugyanis a végső képpel osztani fogunk, így nem szeretnénk nagy számmal osztani, hanem inkább egy egy körüli számmal. Bizonyos szórási kritériumokat megadva kiszámoljuk a kapott átlagkép átlagintezitását. Ez egy egész szám lesz. Ezzel a számmal elosztjuk az átlagolt flat-képünket. Így egy egy körüli átlagintenzitással rendelkező flat-képünk lett. Ezzel az átlagolt és normált flat-képpel aztán leosztjuk az objektum képeket. Természetesen nullával nem szeretünk osztani, így a nullákat (ha egyáltalán előfordulnak) egy nulla körüli kicsi számmal helyettesítjük.

$$\mathrm{I}_{\mathrm{O}(i,j)} = \mathrm{I}^{bias,dark}_{\mathrm{O}(i,j)} - \frac{\frac{\sum_{a=1}^{b} \mathrm{I}^{bias,dark}_{F(i,j),a}}{b}}{\sum_{i=1,j=1}^{M,L} \left(\frac{\frac{\sum_{a=1}^{b} \mathrm{I}^{bias,dark}_{F(i,j),a}}{b}}{M \cdot N} \right)} \tag{4}$$

Így megkaptuk a végső korrigált objektum képeinket. Ezek segítségével megcsinálhatjuk az égi háttér korrekcióját.

3.4. A sky-korrekció: az égi háttér változásának korrigálása

A sky-korrekcióhoz használt képeket az objektumképekből lehet előállítani. Rövid idő alatt (∼15 perc) az égi háttér értéke ugyanazon pontban nem változik jelentősen. Mivel a képek nem teljesen ugyanazok irányba készülnek, ezért a csillagok nem ugyanazokra a pixelekre esnek. Ezért ha egy idoponthoz generálni akarunk egy sky-képet, akkor csak szimplán medián átlagolni kell az időpont előtt illetve után körülbelül 15 perccel készült képeket (természetesen az adott időpontban készült kép nem lehet benne az átlagoláshoz használt képek között). Az átlagoláshoz nem árt valami szórási kritériumot is megadni, főleg ha sűrű csillagmezőről készült a kép. A végső sky-képet aztán le kell vonni az adott időponthoz tartozó képből. Látható, hogy a sky-korrekciós képek egyenként, képekhez készülnek. Általában azonban meg szokták állapítani az égi háttér értékét és nem csinálnak sky-korrekciót.

4. Redukálás, a csillagászati képkorrekciók - gyakorlat

4.1. A flatképek kombinálása

A korábbi pontban megnéztük, hogy milyen tipusú képeink vannak. Láthatóan vannak flat, M67 illetve NGC 2126-os képeink B, V, R és I szűrőkben. A gyakorlat során csak a flat-korrekciókat fogjuk elvégezni. A bias- és dark-korrekció hasonlóan zajlik az *IRAF*-fal, mint a flat-korrekció. Mivel a Schmidt távcsövön található Photometrics kamerának kicsi a dark illetve bias zaja, ezért azzal nem foglalkoztunk.

A korrekciókat az *IRAF*-on belül a `noao. - imred. - ccdred` csomagon belül található *TASK*-okkal hajthatjuk végre. Lépjünk bele a csomagokba egymás után.

```
cl> noao
      artdata.      digiphot.     nobsolete.    onedspec.
      astcat.       focas.        nproto.       rv.
      astrometry.   imred.        observatory   surfphot.
      astutil.      mtlocal.      obsutil.      twodspec.
no> imred
      argus.      crutil.     echelle.    iids.       kpnocoude.  specred.
      bias.       ctioslit.   generic.    irred.      kpnoslit.   vtel.
      ccdred.     dtoi.       hydra.      irs.        quadred.
im> ccdred
      badpiximage       ccdmask           flatcombine       mkskyflat
      ccdgroups         ccdproc           mkfringecor       setinstrument
      ccdhedit          ccdtest           mkillumcor        zerocombine
      ccdinstrument     combine           mkillumflat
      ccdlist           darkcombine       mkskycor
cc>
```

Mivel még nagy valószínűség szerint a `home` könyvtárotokban vagytok, ezért váltsatok át oda, ahol a képek vannak.

```
cc> cd labor2/20020202/
cc>
```

Most létre kell hoznunk négy listát, mely a 4 szűrőre készített flat-képek neveit tartalmazza. Ezt egy egyszerű shell scripttel megoldhatjuk:

```
$ gethead OBJECT FILTER *.fit | grep V | grep flat | awk '{print $1}' > vflat.ls
```

Illetve ugyanezt B, R és I szűrőkre is, logikusan átnevezve, illetve mást „grep"-elve. Így lesz egy `bflat.ls`, `vflat.ls`, `rflat.ls` illetve egy `iflat.ls` nevű fájlunk. Győződjünk meg arról, hogy mindegyik fájlunkban benne van a szükséges listafájl.

```
$ cat *.ls
```

Miután megvannak a szükséges listák, végezzük el a flat-képek kombinálását. Ezt a `flatcombine` *TASK* végzi. Minden *IRAF TASK*-nak van paraméter fájlja, mit lehet szerkeszteni. Így nem parancssorban kell megadni minden paramétert. Ezt a szerkesztést az `epar` paranccsal lehet előhívni. Például a `flatcombine` paraméterfájlja így néz ki:

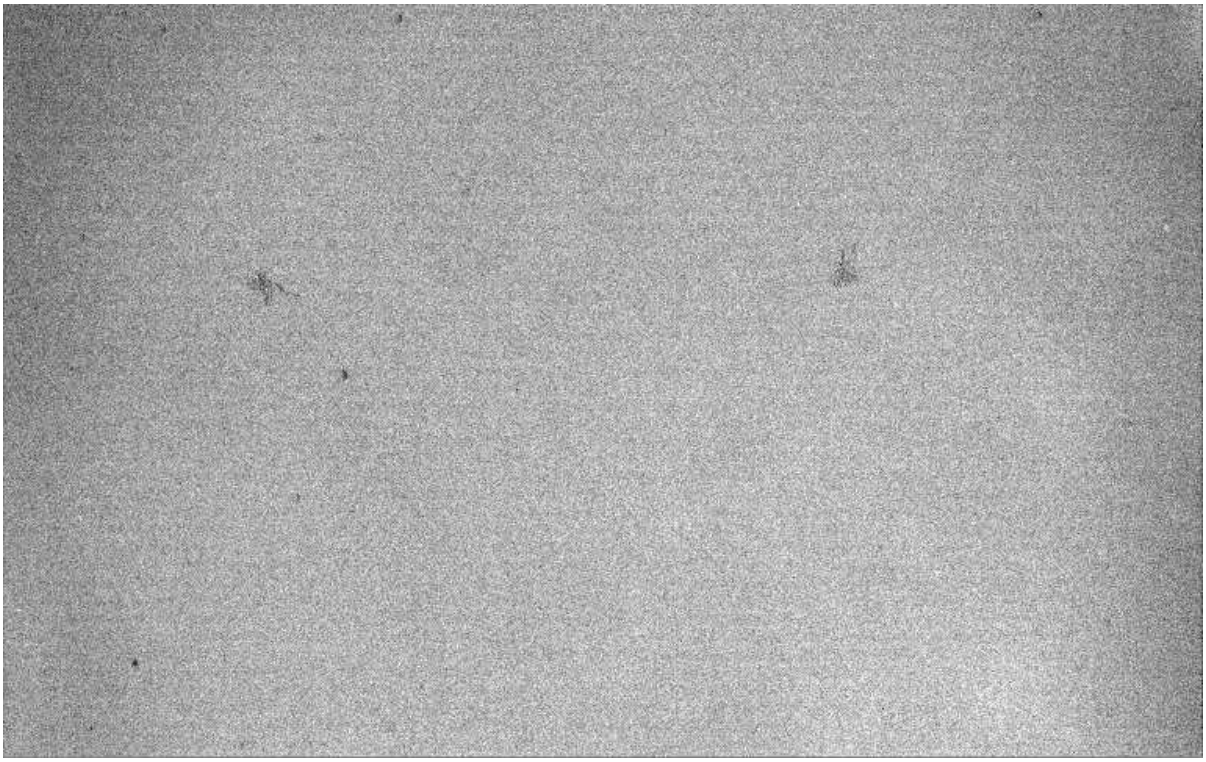
```
cl> epar flatcombine

                     Image Reduction and Analysis Facility
PACKAGE = ccdred
   TASK = flatcombine

input   =             @vlat.ls  List of flat field images to combine
(output =                vflat) Output flat field root name
(combine=               median) Type of combine operation
(reject =              sigclip) Type of rejection
(ccdtype=                 flat) CCD image type to combine
(process=                   no) Process images before combining?
(subsets=                   no) Combine images by subset parameter?
(delete =                   no) Delete input images after combining?
(clobber=                   no) Clobber existing output image?
(scale  =                 none) Image scaling
(statsec=                     ) Image section for computing statistics
(nlow   =                    1) minmax: Number of low pixels to reject
(nhigh  =                    1) minmax: Number of high pixels to reject
(nkeep  =                    1) Minimum to keep or maximum to reject
(mclip  =                   no) Use median in sigma clipping algorithms?
(lsigma =                   3.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=                 13.5) ccdclip: CCD readout noise (electrons)
(gain   =                 1.28) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                -0.5) pclip: Percentile clipping parameter
(blank  =                   1.) Value if there are no pixels
(mode   =                   ql)
```

Ha nem kívánjuk tovább szerkeszteni akkor a `:q` („quit") lenyomásával elmentjük a változásokat és kilépünk. Ha megfelelő a beállított paraméteregyüttes számunkra, akkor a `:g` („go") lenyomásával futtathatjuk az adott *TASK*-ot. A `flatcombine` help-jének tanulmányozásával nézzük meg, hogy melyik paraméterrel mit állítunk be. A lényeges paraméterek: `input, output, combine, reject, ccdtype, delete, rdnoise, gain`. A `rdnoise` és `gain` értéke kamerafüggő. Ellenőrizzük le, hogy mi a helyes beállítás a piszkéstetői Schmidt távcső esetén (http://www.konkoly.hu)!

Ha úgy gondoljuk, hogy minden rendben be van állítva, akkor futassuk a *TASK*-ot. A létrejött képeket ábrázoltassuk a `ds9`-cel és ellenőrizzuk vizuálisan, hogy jók-e a flat-képek. Ha minden jól ment, akkor egy a



2. ábra. Egy V szűrős flat-kép

2. ábrához hasonlót kell kapnunk. Nézzük meg jól a képet! Láthatóak a felszíni struktúrák, illetve az optikai hibából adódó befénylések. Sajnos ma már a Schmidtnek közelítően se ilyen szép a flat-képe.

4.2. A képek vizsgálata - `imexamine`

Gyakran előfordul, hogy szükség van a képeknek valamely tulajdonságára, mint paraméterre (vagy csak egyszerűen kíváncsiak vagyunk). A képek egyszerű vizsgálatát az `imexamine` *TASK*-kal végezhetjük el. Vizsgáljuk meg az egyik elkészített flat-képet!

```
cc> imexam
```

A parancs hatására az egerünk kurzora ugrik egyet és nem kapjuk vissza az *IRAF* promptot. Vigyük át az egerünk kurzorát a `ds9` ablakára. Észrevehetően megváltozik egy karikává! Ekkor a kép különböző területeiről szerezhetünk információkat billentyűk lenyomásával. Hogy melyik billentyűvel mit tudunk ábrázolni, azt a `?` lenyomásával kapjuk meg.

```
                    -- IMEXAMINE COMMANDS --

                    CURSOR KEY COMMAND SUMMARY

? Help              h Histogram         p Previous frame    x Coordinates
a Aperture Sum      i Image cursor      q Quit              y Set origin
b Box coords        j Line gauss fit    r Radial plot       z Print grid
c Column plot       k Col gauss fit     s Surface plot      , Quick phot
d Load display      l Line plot         t Output image      . Quick prof fit
e Contour plot      m Statistics        u Vector plot
f Redraw            n Next frame        v Vector plot
g Graphics cursor   o Overplot          w Toggle logfile
```

Állapítsátok meg a flat-képek háttérértékét, és az egyiknek a hisztogram ábráját a `gimp` *program segítségével mentsétek le eps-ként, majd a végső nagydolgozatba illesszétek be. A megállapított háttér illetve háttérszórások . . . stb. értékeket táblázatba foglaljátok majd össze. A B, V, R, I flat-képeket tegyétek a* `ds9` *különböző frame-jeibe, és blinkelve hasonlítsátok össze őket. Ha van valami észrevételetek, akkor azt szóban foglaljátok össze.*

4.3. A flat-képekkel való korrekció

A korrekciókat a `ccdproc` *TASK*-kal lehet végrehajtani. Ezzel lehet mindegyik képkorrekciót végrehajtani (akár egyszerre is, mivel az *IRAF* tudja a logikus sorrendjüket). Mivel a `ccdproc` is a `ccdred.` csomagon belül található, ezért nem kell kilépnünk belőle. A `ccdproc` paraméterfájlja a következőképp néz ki:

```
                      Image Reduction and Analysis Facility
PACKAGE = ccdred
   TASK = ccdproc

images  =             @vobj.ls  List of CCD images to correct
(output =            @vfobj.ls) List of output CCD images
(ccdtype=                     ) CCD image type to correct
(max_cac=                    0) Maximum image caching memory (in Mbytes)
(noproc =                   no) List processing steps only?

(fixpix =                   no) Fix bad CCD lines and columns?
(oversca=                   no) Apply overscan strip correction?
(trim   =                   no) Trim the image?
(zerocor=                   no) Apply zero level correction?
(darkcor=                   no) Apply dark count correction?
(flatcor=                  yes) Apply flat field correction?
(illumco=                   no) Apply illumination correction?
(fringec=                   no) Apply fringe correction?
(readcor=                   no) Convert zero level image to readout corr.?
(scancor=                   no) Convert flat field image to scan corr.?

(readaxi=               column) Read out axis (column|line)
(fixfile=                     ) File describing the bad lines and columns
(biassec=                     ) Overscan strip image section
(trimsec=                     ) Trim data section
(zero   =                     ) Zero level calibration image
(dark   =                     ) Dark count calibration image
(flat   =           vflat.fits) Flat field images
(illum  =                     ) Illumination correction images
(fringe =                     ) Fringe correction images
(minrepl=                   1.) Minimum flat field value
(scantyp=            shortscan) Scan type (shortscan|longscan)
(nscan  =                    1) Number of short scan lines

(interac=                  yes) Fit overscan interactively?
(functio=                     ) Fitting function
(order  =                    4) Number of polynomial terms or spline pieces
(sample =                    *) Sample points to fit
(naverag=                    1) Number of sample points to combine
(niterat=                    1) Number of rejection iterations
(low_rej=                   3.) Low sigma rejection factor
(high_re=                   3.) High sigma rejection factor
(grow   =                   0.) Rejection growing radius
(mode   =                   ql)
```

Nézegessük a `ccdproc` help-fájlját. Ugye, mint tanultuk, ezt a

```
cc> help ccdproc
```

paranccsal tudjuk előhívni. Itt meg tudjuk nézni, hogy melyik beállítási paraméter mire való. Állítsuk be a paraméterfájlt úgy, hogy elvégezze a `:g` lenyomására a flat-korrekciót a képeken! Mivel a flat-képek szűrőkre jellemzőek, ezért az objektumképeket szűrőnként külön listába kell helyeznünk (ugyanúgy, ahogy a flat-kombinációknál a flat-képeket). Alakítsátok át a korábbi `shell-scriptet` ennek megfelelően, és hozzatok létre 4 listát, melyben az objektumképek vannak felsorolva, szűrőnként. *Tipp: a „>>" segítségével már létező fájlok végére tudunk írni. Szimpla „>" használatával felülírjuk a korábbi fájlokat. Vagy nézzük meg, hogy milyen más fejléc utalhat esetleg arra, hogy objektumról van szó (esetleg az IMAGETYP?).*

Az *IRAF*-ra jellemző, hogy hogyha nem adunk meg kimeneti listát (`output`) akkor felülírja a korábbi fájljainkat. Ha nagyon biztosak vagyunk a dolgunkban és nincs szükségünk a korábbi képekre, akkor ez teljesen jó is, viszont nagyon rizikós. Ajánlatos kimeneti listát készíteni. A kimeneti fájlneveket tartalmazó listafájlban ugyanannyi név kell, hogy szerepeljen, mint a bemenetiben, különben az *IRAF* hibaüzenetet ad. Egy egyszerű `shell-script`-tel könnyen generálhatunk új listákat, pl:

```
$ gethead IMAGETYP FILTER * | grep OBJECT | awk '$3=="V"{print "f"$1}' > vfobj.ls
```

Ha úgy gondoljátok, hogy helyesek a beállítások, akkor futassátok a `ccdproc`-ot (természetesen mindegyik szűrőre). *Hasonlítsátok össze blinkelve néhány képet korrigálás előtt és után. A képeket* `imexamine`*-nal is hasonlítsátok össze, szövegben fejtsétek ki a változásokat, illetve táblázatban foglaljátok össze számszerűen az eltéréseket a két kép között.*

Könnyen megeshet, hogy a flat-korrekció nem tökéletes (pl. a sky-flat készítése már nem teljesen szürkületi égbolton készült és egy csillag bekerült a képbe). Ha találtok hibát, akkor azt korrigáljátok és csináljátok újra a korrekciót.

Végül a képeket objektumok szerint válogassuk szét, külön mappákba téve őket.

```
$ mkdir M67
$ gethead OBJECT f*.fit | grep M67 | awk '{print "mv" $1 "M67"}' | bash
$ mkdir NGC2126
$ gethead OBJECT f*.fit | grep NGC2126 | awk '{print "mv" $1 "NGC2126"}' | bash
```