Практикум №6: обработка изображений

1 Работа с изображениями в Octave

1.1 FITS-файлы

Наиболее популярной утилитой просмотра FITS-файлов является `ds9`.

Для работы с обычными изображениями в Octave используются функции `imread`, `iminfo`, `imshow`. Основные функции обработки изображений находятся в пакете `image`. FITS-файлы — пакет `fits`. Узнать входящие в состав пакета функции можно так:

```
pkg describe -verbose fits
---
Package name:
fits
Version:
1.0.7
Short description:
The Octave-FITS package provides functions for  reading, and writing FITS
(Flexible Image Transport System) files.  This package uses the libcfitsio library.
Status:
Loaded
---
Provides:
Reading and writing FITS files
read_fits_image
save_fits_image
save_fits_image_multi_ext
```

Для чтения используем `read_fits_image`:

```
[I H] = read_fits_image('stars.fit');
imshow(I,[1000 4000])
```

Необходимо указывать пороги интенсивностей для `imshow`. Более точно можно определить диапазоны из статистики по изображению:

```
Imin = min(I(:))
 Imin =  1321
Imax(I(:))
 Imax =  52826
Imean = mean2(I)
 Imean =  2054.6
Istd = std2(I)
```

```
 Istd =  1275.5
Imed = median(I(:))
 Imed =  2009
Low = Imin; High = Imed + Istd;
imshow(I, [Low High]);
% or: imshow(I/High)
```

Как видите, изображение повернуто вправо на 90°. Если мы захотим отображать изображения как в оригинале, нужно заранее повернуть их:

```
I = rot90(I);
imshow(I, [Low High])
```

Следует помнить, что в Octave координаты изображения начинаются с верхнего левого угла (и ось Y направлена вниз), а в стандарте FITS — с левого нижнего (и ось Y направлена вверх).

Убедимся в этом:

```
I2=I;
I2([1:100],[1:100])=65000;
imshow(I2, [Low High]);
```

Сравним, как отображаются координаты пикселя в ds9.

1.2 Работа с гистограммой

Для улучшения отображения попробуем эквализовать гистограмму:

```
HE = histeq(I/High, 65536);
imshow(HE);
% compare to HE = histeq(I/High, 256);
```

Посмотрим, как меняется гистограмма изображения в зависимости от распределения шума. Сгенерируем сначала пробное изображение:

```
I = uint8(zeros(300,300));
I([1:50 250:end], :) = 200;
I([51:249], [1:30 270:end]) = 200;
I([101:199], [100:200]) = 100;
imshow(I)
imhist(I)
Iuniform = I + uint8(50*rand(size(I)));
imshow(Iuniform)
imhist(Iuniform)
Inorm = I + uint8(25*randn(size(I)));
imshow(Inorm)
Inorm = uint8(I + 25*randn(size(I)));
imshow(Inorm)
imhist(Inorm)
Ipoisson = imnoise(I, "poisson");
imshow(Ipoisson)
imhist(Ipoisson)
```

```
Isaltpepper = imnoise(I, "salt & pepper", 0.2);
imshow(Isaltpepper)
imhist(Isaltpepper)
Ispeckle = imnoise(I, "speckle", 0.03);
imshow(Ispeckle)
imhist(Ispeckle)
```

Сравним с эквализацией гистограммы: `imshow(histeq(Inorm))`.

Попробуем применить к изображению гамма-коррекцию:

```
imshow(imadjust(Iuniform, [], [], 0.5))
imshow(imadjust(Iuniform, [], [], 5))
```

Теперь попробуем преобразовать яркость кусочно-линейной функцией

$$\begin{cases} I_{out} = 0.2I_{in}, & I_{in} < 70, \\ I_{out} = 14 + 1.5(I_{in} - 70), & 70 \leqslant I_{in} < 130, \\ I_{out} = 209 + 0.368 * (I_{in} - 130), & 130 \leqslant I_{in} < 256. \end{cases}$$

```
Ilinhyst = zeros(size(Inorm));
idx = find(Inorm < 70);
Ilinhyst(idx) = 0.2*Inorm(idx);
idx = find(Inorm >= 70 & Inorm < 130);
Ilinhyst(idx) = 14 + 1.5*(Inorm(idx)-70);
idx = find(Inorm >= 130);
Ilinhyst(idx) = 209 + 0.368*(Inorm(idx)-130);
Ilinhyst = uint8(Ilinhyst);
imshow(Ilinhyst);
imhist(Ilinhyst)
```

Попробуем сделать ступенчатое преобразование яркости: на изображении `Inorm` заменим все интенсивности в диапазоне $[50, 150]$ на 100:

```
Ilinhyst = Inorm;
Ilinhyst(find(Inorm >= 50 & Inorm <= 150)) = 100;
imshow(Ilinhyst)
imhist(Ilinhyst)
```

1.3 Фильтрация изображений

Попробуем простую медианную фильтрацию на примере пары изображений.

```
subplot(2,2,1)
subimage(Isaltpepper)
subplot(2,2,2)
subimage(medfilt2(Isaltpepper, [5 5]))
subplot(2,2,3)
subimage(Inorm)
subplot(2,2,4)
subimage(medfilt2(Inorm, [5 5]))
```

Видим, что медианный фильтр вполне неплохо справился с шумом «соль-перец» и значительно хуже — с гауссовым шумом.

Рассмотрим применение различных фильтров. Для начала создадим матрицы фильтров:

```
fgauss = fspecial("gaussian", 5, 1);
flog = fspecial("log", 5, 1);
fsobelv = fspecial("sobel");
fsobelh = rot90(fsobelv);
```

И применим их к оригинальному изображению, `Isaltpepper` и `Inorm`:

```
imshow(imfilter(I, fgauss, "symmetric"));
imshow(imfilter(Isaltpepper, fgauss, "symmetric"));
imshow(imfilter(Inorm, fgauss, "symmetric"));
imshow(histeq(imfilter(I, flog, "symmetric")));
imshow(histeq(imfilter(Isaltpepper, flog, "symmetric")));
imshow(histeq(imfilter(Inorm, flog, "symmetric")));
```

Как видите, у дифференциального фильтра отсекается отрицательная компонента — т.к. изображения имеют тип `uint8_t`, однако, можно сделать так:

```
imshow(histeq(imfilter(double(Isaltpepper), flog, "symmetric")));
imshow(histeq(imfilter(double(Inorm), flog, "symmetric")));
imshow(histeq(imfilter(double(I), flog, "symmetric")));
```

Octave неправильно рассчитывает значение фильтров, например:

```
sum(flog(:))
ans = -0.024092
sum(fgauss(:))
ans =  1.00000
sum(fsobelv(:))
ans = 0
```

У дифференциальных фильтров сумма элементов должна быть равна нулю (чтобы константные уровни превращались в нуль), а у усредняющих — единице (чтобы сохранять среднюю интенсивность по изображению). Расширим размер матрицы фильтра лапласиана гауссианы, чтобы улучшить приближение к нулю:

```
flog = fspecial("log", 17, 1);
imshow(histeq(imfilter(double(I), flog, "symmetric")));
imshow(histeq(imfilter(double(Isaltpepper), flog, "symmetric")));
imshow(histeq(imfilter(double(Inorm), flog, "symmetric")));
```

Теперь на примере первого изображения мы видим, что фильтр работает правильно.

Горизонтальная и вертикальная компоненты градиента:

```
ih = imfilter(double(Inorm), fsobelh, "symmetric");
iv = imfilter(double(Inorm), fsobelv, "symmetric");
imshow(histeq(iv)));
imshow(histeq(ih));
grad = sqrt(ih.*ih + iv.*iv);
imshow(histeq(grad));
```

Если мы попытаемся проделать то же самое для `I`, увидим, что `histeq` в данном случае не сработает, и отображать придется командой

```
imshow(grad,[min(grad(:)) max(grad(:))]);
```

Здесь `histeq` не сработал из-за слишком «кривой» гистограммы: подавляющее большинство пикселей приходится на нулевой уровень.

Для смягчения общего контраста можно использовать прием, называемый «нерезким маскированием» (применяемый еще в фотографии на фотоэмульсию). Для этого воспользуемся формулой

$$I_{out} = N \cdot I_{in} - F(I_{in}),$$

где F – некоторый усредняющий фильтр (в оригинале — расфокусированное позитивное изображение).

```
imshow(2*Inorm-imfilter(Inorm, fgauss, "symmetric"));
```

Посмотрим, как изменится размытое изображение при нерезком маскировании:

```
ig = imfilter(I, fgauss, "symmetric");
subplot(1,2,1)
imshow(ig)
subplot(1,2,2)
imshow(histeq(2*ig-imfilter(ig, fgauss, "symmetric")))
```

Очистим все и вернемся к FITS-файлам. Загрузим изображение и построим дифференциальные фильтры с более крупным ядром:

```
clear
[I H] = read_fits_image('stars.fit');
imshow(histeq(I,65536));
fgauss = fspecial("gaussian", 30, 5);
flog = fspecial("log", 37, 5);
```

Попробуем теперь применить к изображению нерезкое маскирование и различные фильтры:

```
imshow(histeq(I-imfilter(I, fgauss, "symmetric"), 65536));
imshow(histeq(imfilter(I, fgauss, "symmetric"), 65536));
imshow(histeq(imfilter(I, flog, "symmetric"), 65536));
imshow(histeq(2*I-imfilter(I, fgauss, "symmetric"), 65536));
imshow(histeq(medfilt2(I), 65536));
imshow(histeq(I-medfilt2(I, [15 15]), 65536))
```

Последний вариант является первым приближением к вычитанию фона.

Посмотрим, как можно найти приближенную маску для поиска звезд:

```
Imed = medfilt2(I, [3 3]);
 % [3 3] is default, we can omit it here
ImedLG = imfilter(Imed, flog, "symmetric");
imshow(ImedLG, [-0.1, 0.1]);
mask = logical(zeros(size(I)));
```

Теперь нужно подобрать подходящий пороговый уровень:

```
mask = logical(zeros(size(I))); mask(find(ImedLG < 0)) = 1;
imshow(mask)
```

```
mask = logical(zeros(size(I))); mask(find(ImedLG < -0.005)) = 1;
imshow(mask)
```

Уже такое грубое приближение позволило обнаружить некоторое количество звезд на изображении. Если использовать более точный («робастный») метод выделения фона (а здесь фон достаточно перекошен), получатся результаты куда лучше.

Отобразим оригинал, заменив нулями все точки, где маска истинна, а затем — наоборот:

```
J=I; J(mask)=0; % hide stars
imshow(histeq(J, 65536));
J=I; J(~mask)=0; % hide background
imshow(J);
```

Наша грубая маска определила все звезды, а также некоторое количество шумов, но при помощи морфологических операций мы можем удалить мелкие объекты (правда, не факт, что не потеряем таким образом слабые звезды; кроме того, мы видим на первом изображении, что особо яркие звезды маскированы не полностью, наш способ достаточно плох и годится лишь для очень грубой астрометрии).

1.4 Морфологические операции, поиск связных областей

Попробуем подчистить немного нашу маску.

```
imshow(imopen(mask, strel('disk', 1, 0)));
imshow(imopen(mask, strel('disk', 2, 0)));
imshow(imopen(mask, strel('disk', 3, 0))); % use this
mask = imopen(mask, strel('disk', 3, 0));
```

Теперь можем аналогично просмотреть маскированные изображения:

```
J=I; J(mask)=0; % hide stars
imshow(histeq(J, 65536));
J=I; J(~mask)=0; % hide background
imshow(J);
```

Но нам интересней выделить все звезды из изображения. Используем поиск связных областей:

```
[L NUM]= bwlabel(mask, 8);
NUM
NUM =  200
imshow(L, [1 NUM])
```

Как видим, нумерация объектов идет слева-направо, сверху-вниз. Чтобы выделить конкретный объект, мы можем применить маску такого типа:

```
J = zeros(size(I)); idx = find(L == 100); J(idx) = I(idx);
imshow(J);
```

Мы выделили одну звезду. Можем выделить тонкую полоску фона вокруг нее:

```
mask100 = logical(zeros(size(I)));
mask100(idx) = 1;
mask100 = xor(imdilate(mask100, strel('disk',2,0)), mask100);
imshow(mask100);
```

И посчитать по ней статистику:

```
bg100 = I(mask100);
mean(bg100)
ans =  2063.1
std(bg100)
ans =  61.711
```

Но для вычисления положений центроидов удобней использовать другую функцию поиска связных областей: `bwconncomp`.

```
K = bwconncomp(mask, 8);
p = regionprops(K);
```

По умолчанию в `p` (это — массив структур) содержатся поля `Area` (площадь объекта), `BoundingBox` (координаты бокса, в котором содержится объект: левый верхний угол и размер) и `Centroid` (геометрический центр объекта). По первым двум параметрам мы можем отфильтровать слишком мелкие и слишком крупные объекты, а также объекты, у которых отношение ширины к высоте сильно отличается от единицы (т.е. там внутри — явно не круг). Третий параметр позволяет судить о примерном расположении центра объекта (более точно, конечно, мы можем вычислить его лишь взяв данные с оригинального изображения, вычтя фон и посчитав точный центроид).

Можно нарисовать несколько боксов:

```
rectangle('Position', p(1).BoundingBox, 'EdgeColor','g','LineWidth',2)
rectangle('Position', p(3).BoundingBox, 'EdgeColor','g','LineWidth',2)
rectangle('Position', p(100).BoundingBox, 'EdgeColor','g','LineWidth',2)

centroids = cat(1, p.Centroid);
plot(centroids(:,1), centroids(:,2), 'ko');
```

Теперь сравним координаты центроида изображения звезды с координатами центроида маски. Обратите внимание, что `BoundingBox` имеет координаты «с половинками»:

```
p(100).BoundingBox
502.500   710.500    14.000    14.000
imshow(mask100);
rectangle('Position', p(100).BoundingBox, 'EdgeColor','g','LineWidth',2)
```

Это связано с тем, что при отображении изображений целым координатам соответствует центр пикселя. Контур, описывающий нашу область интереса, не должен пересекать пиксели, из-за этого координаты его левого угла на 0.5 пикселя меньше, а ширина — на 1 пиксель больше. Следовательно, для того, чтобы получить часть изображения, содержащую только пиксели зоны интереса, нам нужно будет сделать преобразование:

```
Bstart = p(100).BoundingBox(1:2) + 0.5;
Bsize = p(100).BoundingBox(3:4) - 1;
rectangle('Position', [Bstart Bsize], 'EdgeColor','r','LineWidth',2);
```

Теперь наш бокс включает все пиксели изображения звезды. Вычислим суммарный поток и центроид:

```
bg = mean(bg100); % mean background
[y x] = ind2sub(size(I), idx);
% coordinates of pixels in mask of 100th star
```

```
M = sum(I(idx)) - numel(idx)*bg
% energy in circle
M =  55148.54545
xc=0; yc=0; for n=1:numel(x); i=I(idx(n))-bg; xc+=i*x(n); yc+=i*y(n);
printf("%d(%d,%d)\n",i,x(n),y(n)); endfor; xc/=M; yc/=M;
xc
 % X centroid position
xc =  509.36
yc  % Y centroid position
yc =  718.41
p(100).Centroid
 % centroid by mask
ans =
509.63   717.57
imshow(histeq(I, 65536)); % take a look at real image
% real centroid:
rectangle('Position', [xc-1 yc-1 2 2], 'EdgeColor','g','LineWidth',2,
    "Curvature", 1)
% centroid of mask:
rectangle('Position', [p(100).Centroid-1 2 2], 'EdgeColor','r','LineWidth',2,
    "Curvature", 1)
```

Мы видим, что настоящий центр тяжести изображения звезды достаточно сильно отличается от геометрического центра маски. Особенно сильно это проявляется в случае деформаций изображения (вследствие сдвига изображения, комы или прочих аберраций).

Вычислять большое количество характеристик удобней иначе. Например, так:

```
regioninfo = regionprops(L,I,'MajorAxisLength','MinorAxisLength','PixelIdxList',
    'PixelValues');
```

мы получим значения большой и малой полуосей эллипса с идентичными нашей фигурой моментами (что позволит судить о «круглости» каждого объекта), список индексов пикселей и список интенсивностей пикселей на изображении. Если бы фон у нашего объекта был идеально гладким, можно было бы добавить еще свойство `'WeightedCentroid'`, чтобы сразу получить координаты центра каждой звезды, но, увы, для каждой звезды придется считать фон аналогично вышесказанному — через кольца. Для тесных полей потребуется убедиться, что зона вокруг звезды свободна от другой звезды (при помощи все тех же морфологических операций).

Аргументом `regionprops` может выступать и структура, полученная при помощи `bwconncomp`.

Теперь найдем все объекты более-менее круглой формы:

```
roundness = [regioninfo.MajorAxisLength] ./ [regioninfo.MinorAxisLength];
staridx = find(roundness < 1.1 & roundness > 0.9);
 % indexes of "stars"
imshow(ismember(L, staridx)); % show filtered image
```

Теперь, если пройтись итеративно по каждому объекту с номером `staridx`, можно посчитать для него индивидуальную маску фона (как было показано выше — через дилатацию), определить таким образом уровень фона и использовать его для вычисления прочих параметров: полной интенсивности внутри маски и координат центроида.

Если мы добавим в `regionprops` свойство `'Area'`, у нас появится возможность до получения номеров объектов, возможно являющихся звездами, отсортировать их по убыванию площади (скорей всего — и по убыванию яркости):

```
[~, i] = sort([regioninfo(staridx).Area], 'descend'); % get sorted indexes
staridxsorted = staridx(i);
```

И можем отобразить окрестности первых 10 по площади маски:

```
imshow(histeq(I, 65536));
for i=1:10;
c = hsv2rgb([i/10. 1 1]);
rectangle('Position', p(staridxsorted(i)).BoundingBox, 'EdgeColor', c,
    'LineWidth',2);
endfor
for i = 1:10; ltxt{i} = sprintf("star #%d", i); endfor
legend(ltxt)
```

Видим, что у ярких звезд маска потеряла приличную часть крыльев, а также одна достаточно яркая звезда была пропущена. Более надежным будет отсортировать по значению интенсивности.

2 Первичная обработка снимков

Наиболее распространенными светоприемниками в астрофизике оптического диапазона являются ПЗС-матрицы. Среди прочих светоприемников этого диапазона они имеют наилучшие характеристики. Однако, есть у них и недостатки: в тонких подложках возможна интерференция (особенно ближе к ИК-диапазону), приводящая к возникновения «фрингов»; а наличие хотя бы одного очень яркого источника приводит к «растеканию заряда», усугубляющемуся при считывании изображения. КМОП-светоприемники в значительной мере избавлены от проблем растекания заряда, но в них есть другая проблема: каждый пиксель по сути снабжен собственным усилителем, в результате точная фотометрия с КМОП превращается в сплошной ад!

Процесс получения изображений на ПЗС можно в упрощенном виде свести к формуле:

$$ADU_{x,y} = \gamma F_{x,y}(e_{x,y} + D_{x,y})T + B_{x,y} + R,$$

(здесь опущен процесс дискретизации, растекания заряда и т.п.), где ADU – отсчеты, считанные с пикселя с координатами (x, y); γ – «gain», коэффициент преобразования фотоэлектронов в ADU (зависит от длины волны излучения); F – поглощение излучения вследствие неидеальности оптической системы (компенсируется делением на «плоские поля»); e – количество фотоэлектронов, возникших при попадании кванта в данную ячейку (зависит от длины волны), оно зашумляется множеством факторов: фотонным шумом, турбулентностью атмосферы и т.д., и т.п.; D – «темновой ток», определяющийся электронами, возникающими в результате тепловых процессов в теле полупроводника; B – «bias», начальное смещение (даже при нулевой экспозиции — собственно, так кадры «bias» и получают) для устранения нелинейности; T – время накопления сигнала (экспозиция); R – шум считывания.

Избавиться от шума считывания невозможно, однако, как мы помним, если усреднить N кадров, то R уменьшится в $\sqrt{N}$ раз. Кроме того, чем больше экспозиция, тем большим

будет вклад прочих шумов. Однако, при малых экспозициях может проявляться «эффект затвора», вызванный тем, что время открывания/закрывания затвора конечно.

Чтобы скорректировать интенсивность на уровень смещения, мы можем сделать как можно больше кадров с нулевой экспозицией («bias»), получить медиану стопки изображений и вычесть из изображения. При этом шум bias'а также будет уменьшен в $\sqrt{N}$ раз.

Для коррекции на тепловой шум нам необходимо накопить как можно больше «дарков» (в идеале их экспозиция должна быть такой же, как экспозиция кадра данных, а условия должны быть абсолютно такими же: температура чипа, окружающей среды и т.д., и т.п.). При закрытом затворе мы получим:

$$ADU_{x,y} = \gamma D_{x,y} T + B_{x,y} + R,$$

т.е. «автоматически» сюда же включен уровень смещения. Следовательно, если наши «дарки» имеют ту же самую экспозицию, что и основной научный кадр, то для удаления тепловых шумов достаточно вычесть медианное усреднение стопки «дарков» из нашего кадра. Однако, не всегда это возможно: скажем, при получении спектра высокого разрешения экспозиция может длиться несколько часов! Естественно, мы не можем получить даже десятка темновых кадров с такой экспозицией (т.к. и условия поменяться успеют, и начнется новая ночь). В подобных случаях прибегают к хитрости: темновой ток пропорционален времени экспозиции. Это означает, что можно сделать несколько серий коротких экспозиций (скажем, одна, пять, десять и двадцать минут) в течение дня, а затем экстраполировать их. Вот в этом случае для получения такого «искусственного дарка» необходимо будет, помимо всех остальных кадров, снимать еще и bias'ы. Для каждой экспозиции T мы вычисляем темновой ток как $D_T = M(D_{Ti}) - M(B_i)$, где M – медиана. При помощи линейной экстраполяции мы получаем значение D для заданной экспозиции. Но т.к. это значение уже избавлено от смещения, для коррекции нам необходимо будет из изображения вычесть не только D, но и B! Это — единственный случай, когда нам необходимо будет получить и стопку кадров смещения (как говорилось выше, если «темновые» сняты с той же экспозицией, что и основной кадр, то смещение автоматически корректируется при вычитании «темновых» из «объекта»).

Значительно более сложной является коррекция на неоднородности оптической системы (вплоть до пылинок на фильтрах и прочих оптических поверхностях), F. Термин «плоскость» в данном случае носит двоякий смысл: для прямых снимков это — неоднородность освещенности по кадру. Для спектральных — еще и неоднородность кривой спектральной чувствительности. Отвлечемся пока от спектроскопии. Основными факторами, искажающими интенсивность объектов в кадре, являются виньетирование (экранирование боковых пучков апертурой оптического тракта) и прочие неоднородные свойства оптических систем. В идеальной ситуации для компенсирования F нам необходимо получить снимок абсолютно однородно засвеченного протяженного объекта с угловыми размерами, значительно превосходящими поле зрения нашей оптической системы. В случае телескопов с малым полем зрения для данных целей хватает вечерних или утренних снимков неба при максимально рассеянном солнечном свете во время гражданских сумерек. При большом поле зрения необходимо использовать специальные источники света с как можно более равномерным освещением. После получения «плоских» кадров следует вычислить медиану из их стопки, вычесть «темновые», затем нормировать на максимум интенсивности, а далее — разделить подготовленный (с вычетом «темновых») научный кадр на полученное изображение. При этом мы получаем:

$$I_{x,y} = \frac{\gamma F_{x,y} e_{x,y} T + R}{\gamma F_{x,y} T + R} \approx e_{x,y} T.$$

Конечно, «плоские» кадры мы можем снимать с любой подходящей экспозицией. Но для коррекции на темновые токи, если экспозиция «плоских» отличается от экспозиции «научных» кадров, нам необходимо будет сделать еще один набор «темновых».

Как пример медианного усреднения стопки изображений, сгенерируем зашумленные изображения (можно использовать и `awgn`):

```
for N=1:10; II(:,:,N) = I+100*randn(size(I)); endfor
imshow(histeq(II(:,:,2),65536))
```

Подобным образом формируются на ПЗС реальные изображения (разве что шумы мы несколько преувеличили). Именно из-за шумов для получения качественных кадров dark и flat необходимо сделать их как можно больше! После чего стопка изображений усредняется:

```
Imed = median(II, 3);
subplot(2,2,1)
imshow(histeq(II(:,:,2),65536))
subplot(2,2,2)
imshow(histeq(II(:,:,5),65536))
subplot(2,2,3)
imshow(histeq(Imed,65536))
subplot(2,2,4)
imshow(histeq(I,65536))
```

Даже для шума такого уровня мы приблизились к оригиналу достаточно неплохо всего-то на десяти изображениях. Если же их сделать хотя бы сотню, восстановление будет еще лучше. А учитывая то, что медианный фильтр хорошо удаляет выбросы, такой способ отлично очищает опорные кадры от космических частиц.

И для изображений объекта мы можем выполнять подобные манипуляции для снижения шума каждого $e_{x,y}$. Но, в этом случае процесс восстановления будет намного сложней, т.к., во-первых, телескоп имеет неидеальное сопровождение (и объекты не только немного «размазываются», но еще и смещаются от экспозиции к экспозиции); во-вторых, в процессе получения изображений могут изменяться условия (seeing, температура и пр.), в результате чего PSF изображений также будет меняться (следовательно, придется сделать некоторую деконволюцию и приведение изображения к единой координатной системе, а это чревато ошибками интерполяции).

3 Задание для самостоятельного выполнения

В данных к заданию содержатся кадры object, dark и flat. Выполните следующее.

1. Из «научных» кадров выберите один с наилучшим качеством изображений.
2. Вычислите медианные значения темновых и плоских, получите нормированное изображение.
3. Подберите подходящий фильтр, который позволит получить маску, выделяющую изображения звезд.
4. Подготовьте m-файлы для выполнения рутинных операций: вычисления маски кольца фоновых пикселей, определения уровня фона, освещенности от звезды и координат ее центроида.
5. Отсортируйте маски по уменьшению площади, для дальнейших вычислений используйте первые пятьсот.

6. Вычислите освещенности от звезд в пределах полученных масок. Отсортируйте по уменьшению освещенности. Выберите первые 20.
7. Выполните астрометрию для этих самых ярких звезд кадра, получив координаты (x, y) центроидов.
8. Проверьте полученные величины (преобразуйте ADU в звездные величины и проведите нормировку на стандартные объекты из каталогов; для координат центроидов учтите, что изображения повернуты на 90° и ось Y направлена вниз) по каталогам. Определите погрешность своих расчетов (звездных величин и координат).

Для определения каталожных величин воспользуйтесь aladin, xephem, stellarium (и т.п.) или on-line сервисами каталогов. Преобразовать координаты (x, y) на изображении в (α, δ) можно при помощи утилиты `xy2sky` из пакета `wcstools` (либо при помощи `ds9`). Не забывайте о собственных движениях звезд: координаты из каталога (в т.ч. те, которыми мы можем воспользоваться в `ds9`) привязаны к эпохе J2000, а с тех пор прошло уже два десятилетия.