IMAGE ENGINEERING

(A scientific and image processing application usage manual for NIH Image)
(NIH Image 1.55)

General Information

About this Document

This is a "non peer reviewed" guide which attempts to cover the more typical and technical questions that users of NIH-Image often need to address. The questions and answers in this guide differ from those in the NIH Image FAQ in that these are more geared at science and image processing. Some of the text was written from experience on my part and the rest was copied verbatim from the NIH-Image listserv mail group. If you have answers to questions, feel free to send these to me via email. You should know that this guide was NOT collected by Wayne Rasband, who is the author of the NIH Image program. Although I hope you find this guide useful, if you find an error don't blame Wayne for it. This manual was assembled by Mark Vivino of NIH's Division of Computer Research and Technology. You can reach me via email at mvivino@helix.nih.gov

Image Engineering

Fundamentals of densitometry

It is possible to use a scaling system for pixels which has a one to one correspondence to the concentration of what you are studying. Sample concentrations can be determined using optical, electronic, and most importantly for our purposes, a computer based imaging technique. Densitometric science was described originally by Bouguer and Lambert who described loss of radiation (or light) in passing through a medium. Later, Beer found that the radiation loss in a media was a function of the substance's molarity or concentration. According to Beer's law, concentration is proportional to optical density (OD). The

logarithmic optical density scale, and net integral of density values for an object in an image is the proper measure for use in quantitation. By Beer's law, the density of a point is the log ratio of incident light upon it and transmitted light through it.

$$OD = Log_{10}(\frac{I_o}{T})$$

There are several standard methods used to find the density of an object or a point on an image. Scanning densitometers have controlled or known illumination levels, then measure transmitted light through an object such as a photographic negative. Since both the incident and transmitted light are known quantities, the device can then compute this ratio directly. This is also the case of those who use a flat field imaging technique and capture two seperate images. The first image is of an empty light box and the second is of the specimen to be evaluated. These two can then be used in computing a log ratio.

In the case of a video camera/frame grabber combination, using a non flat field technique, several things are of note. With a camera, you do not measure OD values directly. The camera and frame grabber pixel values are linear with respect to Transmission (T), which is the anti-log of the negative of OD:
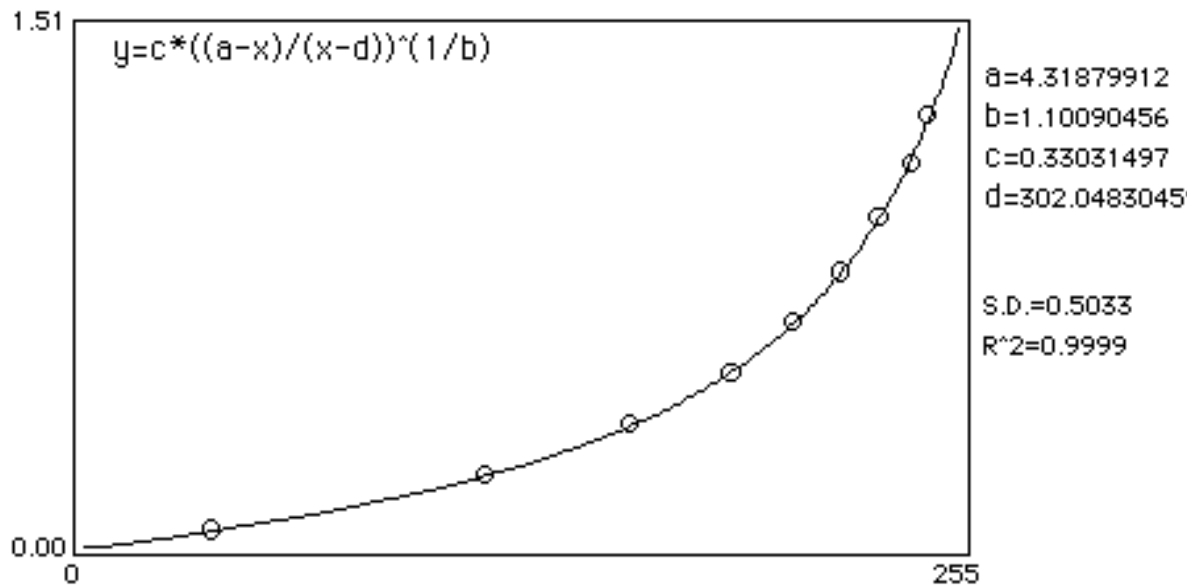
$$T = 10^{-O.D.}$$

or:

$$O.D. = -log_{10}(T) = log_{10}(\frac{1}{T})$$

Since this is often a source of confusion among those designing systems for densitometry you should again note that the camera does not measures T, nor does it measure OD. Camera systems, CCD's, and any frame grabber conversion values (pixel values) have been designed so that they are linear with respect to T. It isn't meaningful to take the minus log of the pixel value since these are not T values.

Nevertheless, you want to do densitometry and need a scale (not pixel values) which correlates to concentration or OD. Further, it may not be convienient to measure the incident light and do a log ratio. Fortunately, you can use an external standard, such as an OD step tablet or a set of protein standards on a gel. NIH Image has the built in "Calibrate..." command to allow you to transform pixel values directly from a scale which is linear with respect to T and into a scale which correlates to OD or concentration. The calibrate command, used with standards, is best done with an exponential, rodbard or other fit since the relationship of OD to T is not a simple linear ($y=mx+b$) relationship (see equation above) and because the camera may not be perfectly linear with respect to T over the range of density values you use as standards. In other words you have both created a LUT of OD values for each linear to T pixel value and you help compensate for slight nonlinearities of the camera.

A sample calibration curve fit:

1.51

y=c*((a−x)/(x−d))^(1/b)

a=4.31879912
b=1.10090456
C=0.33031497
d=302.0483045

S.D.=0.5033
R^2=0.9999

0.00

0

255

There are several other points of note which you should adhere to in performing your densitometric analysis. Your standards should always exceend the range of data which you want to image and perform density measurements on. You should not use curve fit data (or the pixel values) which extend beyond the last, or before the first calibration point. Additionally, there is a point at which the camera can no longer produce meaningful output when additional light is input (saturation). You could also have a low light level condition where the camera or CCD can not produce a measureable output (under-exposure). You will notice that these data points do not fit well into an end of the curve, or could basically ruin the fit of the data. You should remove these points from your calibration data and not use the density values for these pixels in your measurements.

References:

Kodak Corporation, KODAK Neutral Density Attenuators, Kodak publication no P-114, Photographic Products Group, 1982

Kodak Corporation, Scientific Imaging with KODAK Films and Plates, Eastman Kodak publication no. P-315, 1987

Webster JG, Medical Instrumentation, Application and Design, Boston, Houghton Mifflin Company, 1978:95,518-9.

Textbooks of Quantitative Chemistry

Camera signal to noise (SNR) and significant bits

Camera signal to noise ratio is defined as the peak to peak signal output current to root mean square noise in the output current. Although this sounds confusing, the value describes whether the camera is one which can give you 256 significant gray levels or not. A formula to convert camera SNR into number of significant digits is very useful. The formula to do this is defined as :

$$SNR \ (dB) = 6.02n + 1.8$$

Where n will be the number of significant bits. You will need 50 or higher dB for the potential to convert a signal to 8 bits. In practice you will need a better camera than this to get 8 bits since it is likely most conditions for imaging are not ideal.

In order to get a true 8 bits, you will also need to capture the video using a frame grabber that does not introduce error. A

frame grabber which has an analog to digital converter (ADC) with less than one half bit of differential non-linearity, <u>specified at the video bandwidth</u>, is needed to capture 8 significant bits.

References:

IEEE Standard Dictionary of Electrical and Electronic Terms

Analog-digital Conversion Handbook, Staff of Analog Devices, Edited by Daniel Sheingold, Prentice Hall, 1986

Designing your own optimal threshold

      If you do not like the automatic threshold which NIH Image picks for your data, you can implement alternative thresholding with a macro. If you want a consistently picked threshold point based on specific image parameters you might try the following technique. Find an appropriate threshold by trying a multiple of the number of standard of deviations past the mean. Running the macro below will help you do this. Once you are satisfied with the right multiple (can be decimal), code the number into your macro and execute it on all the images you want to analyze. If you have a border or other artifact which is not in all images, be sure to exclude this by using a ROI.

```
Macro 'Std Dev Threshold [T]';
VAR
  Count, Threshold:integer;
  StdDev,TheMean,MultFactor:real
BEGIN
 ResetGrayMap;
{Pick a multiplication that works for your dataset}
 MultFactor := GetNumber('StdDevs past the mean to threshold?',1.0);
{
Hardcode the number in the macro when you find a good value
that seems to work well with your dataset
}
{MultFactor := 1.0;}
 ResetCounter;
 Measure;
 StdDev := rStdDev[rCount];
 TheMean := rMean[rCount];
{Apply the multiplication and threshold}
 Threshold := TheMean+ round(MultFactor*StdDev);
 SetThreshold(Threshold);
 Showmessage('Threshold level =',Threshold,'\\Using ',MultFactor:4:2, ' standard of deviations','\from the mean');
END;
```

Cell Colony Counting macros

The technique described below relates to the cell colony counting macros available in the user_macro directory on zippy.

Digitize cell colony.

If needed perform flat field and dark current correction to image.

Digitize ruler

Spatially calibrate image

Enter into software a minimum and maximum area for a single object to count. (i.e. anything less than min is dust or junk, anything greater than max is an overlap).

Perform background subtraction on image- use "Rolling ball" technique. See article: Sternberg, Stanley, Biomedical Image Processing, IEEE Computer, 1/83 22-34.

Convolve image with a Gaussian (5x5) kernal.

Threshold image-have software find optimal threshold via iterative histogram technique. See article:Ridler, T.W. and Calvard, S., Picture Thresholding Using an

Iterative Selection Method, IEEE Transactions on Systems, Man and Cybernetics, SMC-8, No
8, 1978, 630-632
Count all particles >min and <max (singular particles), each particle is found using an
automatic outlining technique (reproducible).
Find the average area of each particle >min and <max (singular particle average area)
Find the total area of particles >max- (overlapping particles).
Divide this area by the average area of singular particles.
Add the number of singular particles to the calculated value from the previous step step
yielding the total cell colony population.

Convolution

From salzman@Athena.MIT.EDU posting on nih-
image@soils.umn.edu

>Does anyone know what a good reference for determining how
a given
>convolve filter will effect an image? Also the other way
around- how to
>determine the proper array to achive a specific effect. Here I
am talking
>about effects more suited artistic type stuff than image
analysis.

What you see on the screen is called "2D direct space," while
the
representation of an image in terms of its frequency components
is called
"2D Fourier space."
Convolution of an image by a filter (in direct space) is identical
to
multiplication of the Fourier transform ("FT") of the image by
the FT of the
filter, followed by retransformation back into direct space.

All you need to do to understand the influence of a filter is to

understand
what its Fourier transform looks like. For instance, a piece of dust is a
tophat function-- 1 at the center, falling suddenly to zero at some radial
distance-- and its FT looks like a sombrero (actually, an Airy function,
sometimes called the Sinc function)-- near 1 at the center, but vibrating
around zero as the distance from the center increases. That is why you
sometimes see concentric rings in high-resolution images: interference due
to the wave-nature of photons.

There are some subleties due to the difference between complex amplitude and
real intensity, but you can ignore them for the purposes of this discussion.

Useful Fourier duals, read either left-to-right or right-to-left, include

  Tophat $(Sin(X))/X$ [Sinc]
  $Exp(-AxX^2)$ $Exp((-X^2)/A)$ [Gaussian]
  Delta function   constant
  $Cos(X)$ $DeltaFn(X/2)+DeltaFn(-X/2)$
  $Sin(x)$ $DeltaFn(X/2)-DeltaFn(-X/2)$
  Sawtooth $((Sin(X))^2)/(X^2)$

To perform the convolution on an image in direct space, you multiply a
discrete representation of the filter, called a kernel matrix, by a stencil

around each element of the image, sum, and assign the result to the
coordinates in the image. Any real matrix can serve as a convolution
kernel, but to conserve information, the sum of all the elements of the
convolution kernel should equal 1. For instance, if the convolution is
represented as the 3x3 kernel

A11 A12 A13
A21 A22 A23
A31 A32 A33

and the image is designated by the 10x10 array of pixels

B00 .. B90

.. .. ..

B90 .. B99

then the convolution produces a 10x10 array of pixels C00..99, and

C62 := (A11xC51 + A12xC52 + A13xC53
     + A21xC61 + A22xC62 + A23xC63
     + A31xC71 + A32xC72 + A33xC73
)

and analogously for the other 99 pixels Cii. (When the stencil overhangs an
edge, the overhanging elements are ignored (e.g. use a 2x3, 3x2, or 2x2
stencil as appropriate), and the remaining kernel elements should be
renormalized to sum to 1.)

The FT of the FT of the function is the original function again, which means
that the FT of a sombrero is a tophat. So, a reasonable "sharpening filter"
is the convolution kernel

.25 -.5 .25
-.5   2 -.5
.25 -.5 .25

The FT of a thin bell curve (Gaussian) is a fat bell curve and vice versa.
So, a good smoothing ("unsharpening") filter is the convolution

kernel
```
 .05 .10 .05
 .10 .40 .10
 .05 .10 .05
```

An good edge-emphasizing kernel sums to zero, not to one. Try
```
 -1 0 1
 -1 0 1
 -1 0 1
```
for Western edges, and 90 degree rotations of it for the other compass
directions.

From Norm Hurst norm_hurst@maca.sarnoff.com posting on nih-image@soils.umn.edu

What convolution does is add the image to itself at many different
displacements (as many as there are numbers in the kernel). The position of
each number relative to the number in the center determines the [X,Y]
displacement of a shifted image, and its value determines its relative
contribution to the sum (that's why it's called a "weight"). The weight in the
center determines the amount of the "main" (unshifted) image that comes
through.

If a tap weight is negative, you get an inverted, shifted image added into the

total.

For weird effects, use large kernels, sparsely populated (mostly zeros). Large
displacements clearly appear as distinct images; smaller displacements just
look like blurring (or sharpening, depending on the values).

Here's a 9x9 kernel with just a few values set to non-zero:

```
3   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0  -4   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   5   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   4
0   0   0   0   0   0   0   0   0
```

This will give you a main image (the "5") and three other images, one of them a
negative.

Even with this large matrix the displacement is slight. You need bigger
kernels to get bigger displacements, but computation time goes up as the square
of the displacement you are trying to acheive. I tried a 39x39, and you have
to be patient.

Try cascading convolutions (convolve the result of a convolution).

At very large kernels you will notice that Image automatically makes a
selection of a subset of the picture; this is because when you get closer than
half the kernel width to and edge, some weights "fall off" the

edge, and it's
not clear what to do. Wayne opted to not even try to filter those
pixels.

I don't think the kernel must have and odd-by-odd size (but it
must be square
for some reason... Wayne?), but if your kernel is even-by-even,
there is no
"center", and your output image will be shifted a little (1/2
pixel).

For kernels whose taps add up to zero, Image automatically adds
a gray-level
offset to the filtered data (because half of the pixels will be
negative, and
Image only diplays positive numbers). No one knows how to display negative
light :-)

Cross correlation

From russ@mat.mte.ncsu.edu reply on nih-
image@soils.umn.edu

past questions and answers:

>> >Isn't plain old 'filtering' (ie convolution) fairly similar to
>> >cross-correlation, except for a constant multiple?
>>
>> I do not think it is the same. Spatial convolution is the same
thing as
>> multiplication in the frequency domain. Cross-correlation, on
the other
>> hand, is performed by convolution in the frequency domain.

\>\>
\>correlation is multiplication by the complex conjugate in the frequency domain

Cross-correlation can be carried out in either the frequency or spatial
domains with identical results. In the spatial domain, you would use the
brightness pattern of one image as the filter, and apply it in the usual
way to the second image. In the frequency domain, you just multiply as
noted above. The spatial domain method works fine, but it is very slow when
the target pattern gets large. If the things you are trying to track are
small (less than about 7x7 pixels), the filtering approach will work fine.
A word of caution - no matter how it is done, the cross correlation method
assumes that the target has not changed, and in particular has not rotated
as it moved. In many cases, such as surveillance tracking or recognition,
it is necessary to use many different targets corresponding to all possible
orientations of the object.

From vokey@hg.uleth.ca reply on nih-image@soils.umn.edu

"Isn't plain old 'filtering' (ie convolution) fairly similar to
cross-correlation, except for a constant multiple?"

Yup, for the most part. If the images (image and "filter") are normalized, it
is the correlation coefficient. Unnormalized, its magnitude is

affected by

both SUM (image)^2 and SUM (filter)^2, but in either case is a measure of

    ij   ij   ij   ij

similarity.

Background subtraction notes

From A. Eke <ekeand@kisk.sote.hu> posting on nih-image@soils.umn.edu

     Try multiple smoothing of the image, which effectively works
     as defocusing it. Then subtract the defocused image from the
     original. This yiled an image of no backgroung gradient. I
     do not know about some other disturbing details in the
     background that may bother you.

From Edward J. Huff huff@mcclb0.med.nyu.edu posting on nih-image@soils.umn.edu

Since you have the opportunity of collecting some more images from the
sample, you should try to get an image that reflects the illumination
intensity. Then this image can be subtracted from, or better, divided
into, the original image. The best algorithm I have found so far is the
flat field correction. You have to collect a dark image (no light on the

camera, and adjust the gain and offset so that this image does not give all
zeros) and a bright image (out of focus may do the job, or move the in
focus sample around a lot while collecting the image) which reflects the
amount of illumination at each pixel. The calculation is (sample - dark) *
coef / (bright - dark). The Image 1.54UMX which is coming out soon has
this calculation in 16 bits (or you can buy e.g. IPLab). The integration
time for the each pair of images being subtracted must be the same. If
your bright image is collected for a different time than the sample, then
it needs its own dark image.

For fluorescence microscopy images, this calculation is absolutely
essential, because it converts the image from a record of how much light
struck the camera at each pixel into a record of how much dye was present
at each point of the sample. The assumption is that the amount of excited
light from each dye molecule is proportional to the amount of illumination.

Uusing background subtraction is inadequate if the illumination varies by
more than a few percent. For instance, if the illumination is twice as
strong at the center, then the after background subtraction, a spot of dye
at the center will look twice as bright as the same spot moved to the edge.
But after flat field correction, the two spots will look the same brightness.

Noise removal

From jvanheld@dbmdec5.ulb.ac.be (Jacques VAN HELDEN)
posting on nih-image@soils.umn.edu

One way to remove noise from a gray scale image is to apply a median filter.
This filter replaces each pixel from the source image by the median value of
its neighbours. The effect is to remove all points that are darker

or
brighter than their neighbours, and thus to remove noise. By fixing the X and
Y size of the median filter (the number of neighbours to take into
consideration is X*Y), you determine the size of the noise you want to
remove.

There is a median filter routine in the QuantileCrest version of nih-image,
which is on zippy in the contrib dir. The name of the routine is XYQuantile,
in the User menu. You should try different sizes of the filter to eliminate
your noise. Note that the filter is much slower with bigger XY dimensions.
But it works.

However, if you have anyway to binarise the image for measurement, it is surely
faster and easier to remove noise from the binary image (as John Russ proposed)
than from the gray scale image.

From (russ@mat.mte.ncsu.edu) posting on nih-image@soils.umn.edu

>A gray scale microscopic image of cells. In certain cells in the field there is
>a darkly stained structure of irregular shape that I want to measure the area
>of. When I threshold for the stained material I get a lot of non

structure
>pixels througout the image (noise). The area of the noise is 6 pixels or less
>while the structures are much larger. I would like to filter out the noise. What
>I have tried is making a duplicate of the image and make it binary and then
>erode with a coefficient of 5 several times which gets rid of almost all of the
>noise and then AND the eroded image with the density sliced image, but I find
>that I have lost a significant portion of the strutures of interest. I think a
>filter based on size might solve my problem - or does anyone have other
>suggestions.

Two approaches may be useful:
1. as you say, measure everything and discard features based on some size

criterion (area, length, etc.) This is probably the fastest unless the

presence of too many little things bogs down the measurement process

2. use erosion with a suitable coefficient, but then follow it with an

equivalent number of dilations. (Opening). This will restore the area of

the larger features, more or less, but it will smooth their shapes somewhat.

From "Norm Hurst" <norm_hurst@maca.sarnoff.com> posting on nih-image@soils.umn.edu

Here's a macro for "coring" an image. First the image is high-pass filtered,

then these highs are "cored", which means values near zero (you tell it how

near) are set to zero, and values outside that range are moved closer to zero

by the amount of the coring width.

A low-pass image is formed by subtracting the highs from the original, and the

cored highs are added to the lows.

This approach works well if your image has edges separated by noise-infested

flat areas.

Grey dilation and erosion

From jvanheld@dbmdec5.ulb.ac.be (Jacques VAN HELDEN)
posting on nih-image@soils.umn.edu

I also wrote a routine which is an extension of median filtering.

It allows filtering on larger arrays than 3x3 (up to 11x11, but it is
then very slow). Rectangular arrays are allowed (you can filter
along
either X or Y axis for instance, which is convenient for
"anisotropic"
images).

The routine also allows to choose the darkest, lightest, median or
any
intermediate value.
It is availaible on zippy, in pub/image/contrib.
The name of the file is QuantileFiltering.sit.hqx .

From jmt1ntg@fencer.cis.dsto.gov.au posting on nih-
image@soils.umn.edu

The code for these filtershas been done in a more general way
by Jaques Van
Helden. I have tried it out and it seems to work well. The
approach adopted
by Jaques allows you to choose any size of array from 3x3 to
11x11 and also
to choose rectangular arrays. It also allows you to choose the
pixel whose
intensity is in any position in the sorted set of intensity values
for the
array of pixels you are looking at. That is if you want the
brightest,

second brightest, third brightest .... you may have what you
want.
If you duplicate two copies of an image, and take the brightest
quantile
of one and the darkest quantile of the other and then copy one of
these
filtered images, paste it onto the other, and then use the subtract
button
in the paste control, you will display an image which has the
edges
enhanced.

The only thing we need now is for these items to be runnable from macro
calls.

From jvanheld@dbmdec5.ulb.ac.be (Jacques VAN HELDEN)
posting on nih-image@soils.umn.edu

> The only thing we need now is for these items to be runnable from macro
> calls.

It is possible to call the Quantile functions from a macro;

XY Quantile: UserCode(1, FilterW, FilterH, Quantile);
Z Quantile: UserCode(2,Quantile,0,0);

I added a new funtion in the zippy file: Crest Pathway.
the file name is now Image_1.51_QuantCrest.sit.hqx

Crest Pathway: UserCode(3,radius,angle,length);

The crest pathway is a function that allow you to trace contours.
It traces a line following a crest in an image, by reiterately
stepping from a point to its darkest neighbour.

The user is asked to enter 3 parameters:

- Averaging distance: to avoid influence of noisy pixels, you can ask
the routine to average the gray level over a chosen distance, and to

choose the direction with the darkest average.

- Maximal angle: this parameters indicates the maximal deviation that
the path can take at each step. Enter 1 for 2 x 45 deg (right and left),
2 for 2 x 90 deg, and 3 for 2 x 135 deg.

- Maximal path length: indicates the number of reiterations of the
routine. Note that the routine stops before if the pathway is cycling.
A value of 0 for the maximal pathway indicates that the routine stops
after having accomplished a cycling pathway. The "tail" is then
automatically removed and only the cycle remains.

Filters for color imaging

From "Tarja Peromaa" <peromaa@gra-gw.hut.fi> reply on nih-image@soils.umn.edu

>However, I am not sure of the best filter combination
>for capturing "real life" color images. Intuitively, one could think that
>the best filters are those with spectra close to the eye color pigments. But
>if I understood, red and green pigment are largely overlapping.
>Alternatively, one could choose three totally non-overlapping filters, but
>would the image then look like what we see ?

The absorption spectra of cone pigments are indeed overlapping.

In addition,

the spectra is weak at the blue end of spectrum. See for example Cornsweet
(1970) Visual Perception, s. 171 for psychophysically derived estimates of
the spectra.

One possibility is to use CIE color matching response functions (or
"tri-stimulus curves") as reference. (However, the spectral response of your
camera is probably non-ideal, so although Your filters were ideal, the
camera would distort the measures.) These functions are used when perceived
color is modelled (CIELab, CIELuv). I think that there are some scanners (or
they are under development) that have filters with this sensitivity.

Color separation (used when color pictures are printed using halftoning) is
performed using approximately gaussian filters (I don't know whether ideal
profiles were better), with some overlap and the average spectral
absorptions being 440, 540 and 620 nm. Color separation filters are
narrowish compared to tristimulus filters.

If the spectra overlap considerably, there leads to a situation, where
saturated colors are difficult to reproduce, because tight and wide spectra
of original colours appear similar to the filters. For saturated

colors,
tight transmission spectra of the color separation filters are requested,
but then the system does not distinguish all the colors (those remaining
between the filters.) (Saarelma & Oittinen 1994: Fundamentals of printing
technology. Helsinki University of Technology, Laboratory of Graphic Arts
Technology, s. 95). (Probably this book also handles these questions, Hunt
1987: The reproduction of color. Fountain Press: England.)

The SBIG ST-6 ccd-camera performes color imaging with a color filter wheel,
where the passbands are 585-680, 496-585, <380-502 nm. This instrument,
however, is ment for astronomical purposes, so the bands may not optimal for
other purposes.

From russ@mat.mte.ncsu.edu reply on nih-image@soils.umn.edu

>Does anyone knows how to choose the best spectra for the R, G, and B
>filters ?

"Best" is a tough word. As a practical matter, I have for a long time been
acquiring separate RGB planes through Kodak Wratten filters 25=Red,
58=Green, and 47 or 47B=Blue, and then combining them as

color planes in
Photoshop. One advantage of using these filters is that you can get them at
practically any camera store, they are cheap, and available as plastic
sheets that can be cut to fit. Much simpler than trying to find custom
dichroic filters, or getting glass filters in the right size.

Many of my images are "real world" rather than microscope, and so the
criteria for judging the truth of the colors are probably more critical
(visual observers are used to seeing by "white" sunlight, but microscopists
know that the color values are easily changed by the color temp of the
light source, or filters, etc.). In most cases, the simple combination of
the three planes produces a quite viewable and acceptable color balance
that observers find realistic. In cases where the color balance appears to
be "off" it usually appears just a little too blue, and a simple adjustment
with the Photoshop controls can be made.

From Nick Safford (NSafford@scigen.co.uk) reply on nih-image@soils.umn.edu

The eye has dreadful sensors in it, as far as red-green discrimination is
concerned. [If you look at the spectral sensitivity of the various cells
(measured in a rather indelicate experiment involving a liquidizer and a
darkened room) the raw data give a distribution of spectral sensitivities
such that, in some cases, you'd be hard pushed to decide whether to class a
given cell as red or green sensitive]. The brain (or retina) sorts this
stuff out, but if you have a free choice of filters to use, you'll benefit
from better separated channels.

Irritatingly, I'm not going to make specific recommendations, but I'd
suggest you could look at what filters would be best at separating the
colors you're trying to distinguish. There are tricolor filters available
(have a look in the Wratten catalog from Kodak). These are optimised for
interacting with film, but they do a good general purpose job. I don't know
if you can readily get them in a form suitable for putting in you
microscope. On the other hand, if you're trying to distinguish

(say) two
stains, you could just pick a filter that blocks one of them and not the
other. I guess I'd call it pseudocolor microscopy (sorry, I just made it
up, but I'm sure it's been done before and has a name). Then you might need
only two exposures (or even one) to do the job. The results wouldn't be
unlike the infra-red satellite images you sometimes see.

PS - I don't have the exact reference for spectral sensitivity of the human
eye, but I think it was one of two books called "Human Color Vision"
(published in New York) and "Colour Vision: physiology and psychology" (or
something of the like, possibly published in Britain, given the spelling).

Polarising filters with CCD's

From Don Seltzer <Don_Seltzer@qmlink.draper.com> reply on nih-image@soils.umn.edu

Specular Reflections
>Try a polarising filter over the camera lens. Rotate this until the
>specular reflections are cut out.

Warning: Most CCD cameras are responsive to near IR wavelengths. Many
polarizing filters, particularly the plastic type, do not polarize in this range. I

learned the hard way. After setting up filters on both the light source and
camera, and verifying by eye that the glare was eliminated, I was still getting
a camera image with a glare reflection. My solution was to insert an additional
blue or yellow filter in front of the camera, passing only the polarized visible
wavelengths.

Infra-red & Coomassie blue filters

There are filters you can place in front of a camera lens to give complete rejection of the IR (>700nm) wavelengths. This is often crucial in video acquisition using CCD cameras. Filters which show 0% transmittance above 700 to 800 nm are available and recommended over those which show 10% or more transmission at these wavelengths. There are also several filters that compensate for the poor video response in the wavelengths associated with Coomassie blue stains. The graph below represents a pass of Coomassie brilliant blue in typical solution through a spectrophotometer. The peak absorbance is about 585 nm.
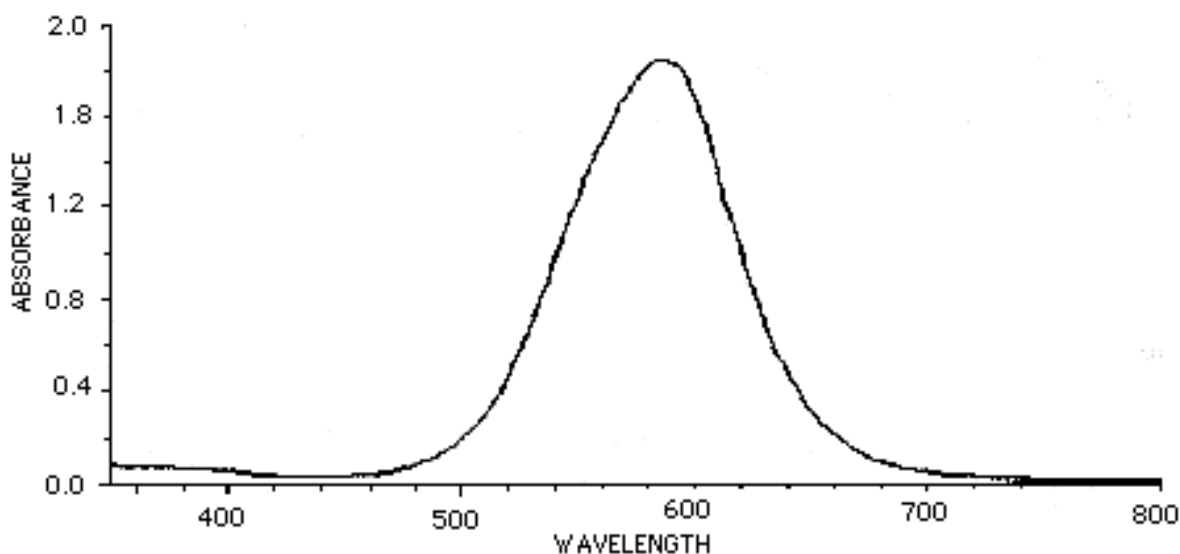
Corian Corporation sells filters filters at 630 nm and 560 nm, both of which produce excellent results when imaging Coomassie blue. This is true when sufficient lighting exists. Corian also sells a complete IR suppressing filter for about $200. Filter holders can be purchased at photography stores or fabricated. Tiffen sells 52 and 62 mm ring attachments which fit on typical camera lenses. These are also available at photography stores.

Corian Corp
1-508-429-5065
73 Jeffrey Ave.
Holliston, MA 01746-2082

FR-400-S Complete IR surpressing filter
P10-630-R and P10-560-R, 630nm and 560nm filters

Useful handheld debounced pushbutton

The circuit below is a useful handheld external trigger for frame grabbers or other devices. It can be adapted in a number of ways to debounce the signal from an instrument to the quickcapture or other frame grabber. The top half is for battery power, the bottom half does the debouncing.