Subject: Re: simple deconvolution
Posted by James[2] on Tue, 22 Feb 2011 23:15:26 GMT
View Forum Message <> Reply to Message

On Feb 22, 7:00 am, chris <rog...@googlemail.com> wrote:
> Hi folks,
> I want to implement an image deconvolution into a larger package. The
> following code performs either the Iterative Wiener (by A.W.
> Stevenson) or the Richardson-Lucy deconvolution, but both go wrong for
> the recovery of both smoothed images and smoothed images plus noise .
> I'm a little bit confused about that. Maybe somebody could help me?
> The implemented CONVOLVE comes from the Astrolib. I'm using IDL 8 and
> the code is not optimised as you can see :)

It would help a lot if you commented your code. Then we could get an
idea of what you're trying to do, and have a better chance of
identifying why it doesn't work.

I'll intersperse my comments before the lines of code they refer to.

```
> function cr_deconv,im,psf,method,small=small
>         sz1 = size(im,/dimensions)
>         sz2 = size(psf,/dimensions)
```

This line would be more clear as: if ~keyword_set(small) then small =
1e-5

```
>         small=~n_elements(small)?1e-5:small
```

I'm not sure why you test for equality in the dimensions here. This
code would produce an error or unexpected results if the PSF is bigger
than the image: you'd get negative indices. It looks like you're
trying to say 'if the PSF and the image aren't the same size, make a
new array the size of the image with the psf centered in it.' Is this
correct?

```
> if total(sz1 eq sz2) ne 0 then begin
>         p=fltarr(sz1)
>         p[(sz1[0]/2)-(sz2[0]/2) ,(sz1[1]/2)-(sz2[1]/2)]=psf
> endif
```

This next line would produce an error if the previous if..then block
didn't run. P would be an undefined variable.

```
>         p/=total(psf)
>         p[where(p lt small)]=small
> if method eq 'iwiener' then begin
>          psf_fft=fft(p)
```

The next line throws away the sign: small negative values are changed
to a small positive value. I don't know if this matters, but it seems
like it might.

```
>          psf_fft[where(abs(psf_fft) lt small)]=small
```

Is the colon at the end of this line supposed to be a semicolon, for a
comment?

```
>          snr=mean(median(im,3))/stddev(im-median(im,3)) : snr
>          pc=psf_fft*conj(p)
```

Same thing with the small negative values here...

```
>          pc[where(abs(pc) lt small)]=small
>          filter=pc
>          filter/=(filter+1./snr)
```

... and here.

```
>          filter[where(abs(filter) lt small)]=small
>          res=abs(fft(filter*fft(im)/psf_fft,/inverse))
>  for i=0l,iter-1l do begin
>          res+=abs(fft((fft(convolve(i eq 0?im:res,p)-im)/psf_fft)*$
>          (pc/(pc+(1./snr))),/inverse))
>          snr=mean(median(res,3))/stddev(res-median(res,3))
>  endfor
>  else begin
>           corr_kernel=rot(p,180)
>           for i=0l,iter-1l do $
>           res=(i eq 0?im:res)*convolve(im/convolve(i eq 0?
>  im:res,p),corr_kernel)
>  endelse
>  return,res
>  end
```

I don't know enough about deconvolution algorithms to help with the
parts inside the for loops. But it seems possible that a programming
error is causing problems, rather than an incorrect approach
mathematically.

Also, I think you should ease up on the ternary (? :) operator a
little bit. It's useful for making concise expressions now and then,
but in general the if..then..else block makes more understandable code.