
Subject: Re: correlation between images

Posted by [Wout De Nolf](#) on Fri, 03 Apr 2009 12:01:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

So this is what I came up with so far (see code below) thanks to Brian's suggestions.

To say whether 2 images are "the same", I subdivided "img2" in subimages. For each subimage I try to find its best correlating position in "img1" with a loop over `c_correlate` while shifting the subimage over `img1`.

The criteria for "images are the same" are:

1. A correlation tolerance level (`Rtol=0.9`): each subimage must correlate better than `Rtol` with some part of `img1`
2. The subimage positions in `img1` must be regularly aligned with respect to each other within a pixel tolerances (`shifftol=0.01*imgsize`)

I have a question for you: when the criteria above are met, would you believe the images are the same? If not, do you have a better suggestion?

If you wonder what images I'm talking about, it ain't all roses like in the example. I have a distribution map of the copper content in a fragment of a painting. The second distribution map represents the azurite content (azurite is a copper containing pigment). Both should be the same (correlated) if there are no other copper containing pigments present (or if the other pigments are mixed with azurite, so having the same distribution). I want to prove that this is the case.

CODE:

```
;-----  
pro test  
path = Filepath(Subdir=['examples', 'data'], 'rose.jpg')  
read_jpeg, path, img, /true  
img = 0.3*Reform(img[0,*,*]) + 0.59*Reform(img[1,*,*]) +  
0.11*Reform(img[0,*,*])  
  
kernelSize = [10,10]  
kernel = REPLICATE((1./(kernelSize[0]*kernelSize[1])), $  
    kernelSize[0], kernelSize[1])  
img2= CONVOL(img, kernel, /CENTER, /EDGE_TRUNCATE)
```

```

print,image_equal(img,img2,/outid)
end;pro test
;-----
function
  image_equal,img1,img2,npix=npix,shifftol=shifftol,Rtol=Rtol, outid=outid
; Image offsets or scales don't matter
; npix: subimage pixels for cross-correlation
; shifftol: subimage shift tolerance
; Rtol: cross-correlation tolerance

s1=size(img1,/dim)
s2=size(img2,/dim)
msize=s1[0]<s1[1]<s2[0]<s2[1]

if not keyword_set(npix) then npix=fix(msize*0.4)>10 ; 40% of the size
npix<=msize

if n_elements(shifftol) eq 0 then shifftol=(msize*0.01)>1 ; 1% of the
size
if not keyword_set(Rtol) then Rtol=0.9

; Subimages in img2
nsub=s2/npix
nx=nsub[0]
ny=nsub[1]
x0=npix*indgen(nx)
x1=[x0[1:],s2[0]]-1
y0=npix*indgen(ny)
y1=[y0[1:],s2[1]]-1

; img2 subimages in img1
xoff=lonarr(nsub)
yoff=xoff
xyccor=fltarr(nsub)
if keyword_set(outid) then img2recon=img1*0

; Cross-correlate subimages of img2 with img1
for i=0,nx-1 do $
  for j=0,ny-1 do begin
    sub=img2[x0[i]:x1[i],y0[j]:y1[j]]
    ssub=size(sub,/dim)-1

    ; Number of sub-shifts in img1
    noffx=s1[0]-ssub[0]
    noffy=s1[1]-ssub[1]
    ccor=fltarr(noffx,noffy)

```

```

; Correlate sub with img1
for k=0,noffx-1 do $
  for l=0,noffy-1 do $
    ccor[k,l]=c_correlate(sub,img1[k:k+ssub[0],l:l+ssub[1]],0)

; Sub image offset and cross-correlation
mccor=max(ccor,moff)
k=moff mod noffx
l=moff/noffx

xoff[i,j]=k
yoff[i,j]=l
xyccor[i,j]=mccor

if keyword_set(outid) then begin
  img2recon[k,l]=sub
  print,'Progress: ',(i*ny+j+1.)/(nx*ny)*100,'% '
endif
endfor

; Check whether img2 and img1 are equal
bsame=total(xyccor lt Rtol,/pres) eq 0
bsame and= total(rebin(total(xoff,2)/ny,nx,ny)-xoff gt shifttol,/pres)
eq 0
bsame and= total(rebin(reform(total(yoff,1),1,ny)/nx,nx,ny)-yoff gt
shifttol,/pres) eq 0

if keyword_set(outid) then begin
  window
  tvscl,img1,0
  tvscl,img2,1
  tvscl,img2recon,2
  tvscl,img2-img2recon,3

  xyouts,0.1,0.7,'img1',/normal,color=100
  xyouts,0.3,0.7,'img2',/normal,color=100
  xyouts,0.5,0.7,'reconstructed img2',/normal,color=100
  xyouts,0.7,0.7,'img1 - reconstructed img2',/normal,color=100
  isurface,xyccor
endif

return,bsame
end;function image_equal

```
