Subject: Re: Image warping in IDL
Posted by Wox on Tue, 14 Nov 2006 16:20:52 GMT
View Forum Message <> Reply to Message

> On Fri, 10 Nov 2006 15:51:34 -0700, JD Smith <jdsmith@as.arizona.edu>
> wrote:
>
> <snip>
> If you code this up, let us know whether it was faster.
>

Version: IDL 6.2
OS: WinXP
CPU: Intel Pentium 4A, 2018 MHz (20 x 101)
Mem: 512MB

Benchmark forward mapping of array(1200x1200). Repeated 10 times,
average seconds:

Obtaining the fractional coord, using 2D spline: 0.435900

Different forward mappings:
Dummy method: 0.212500
Fant's resampling(double loop): 36.6000
Drizzle like algo. (empty pixels): 8.18290
Drizzle like algo. (trigrid empty pixels): 9.92030

The mapping seems smooth enough to omit interpolating the "empty
pixels". However in general, it should probably be used. So your
suggestion speeds it up 3/4 times! So the message seems to be the same
as always:"Use histogram." :-)

Maybe some code for the ones who are interested:

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;Obtaining the fractional coord
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
imgs=size(*img)
seval=indgen(imgs[1])
teval=indgen(imgs[2])
xmap=seval#replicate(1b,imgs[2])
ymap=replicate(1b,imgs[1])#teval

; Correct: xmap,ymap in CCD => calculate where they are in corrected
```

```
frame
; X-distortion
 xmap+=bsplineint2Dcp(tck1.uvec,tck1.vvec,tck1.p,tck1.q,tck1.
n,tck1.m,tck1.h,tck1.k,seval,teval,tck1.CP)
; Y-distortion
 ymap+=bsplineint2Dcp(tck2.uvec,tck2.vvec,tck2.p,tck2.q,tck2.
n,tck2.m,tck2.h,tck2.k,seval,teval,tck2.CP)



;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;Dummy method
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
temp=*img
(*img)[*]=0
(*img)[xmap,ymap]=temporary(temp) ; => black pixels + duplicates
overwritten!!!



;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;Fant's resampling
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
interimg=ptr_new(*img)

; loop over rows
for i=0,imgs[2]-1 do $
resample_array, xmap[*,i], img, interimg, imgs[1], 1, i*imgs[1]
; loop over columns
for i=0,imgs[1]-1 do $
resample_array, ymap[i,*], interimg, img, imgs[2], imgs[1], i

ptr_free,interimg



;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;Drizzle like algo
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
>   ; Intermediate image: add boarder of two pixels
>   interimg=MAKE_ARRAY(imgs[1]+4,imgs[2]+4,type=size(*img,/type ))
>  fsum=interimg
>  fsum[*]=0
>
>  imgsinter=imgs+4
```

```
>  npixmax=imgs[1]*imgs[2]
>
>  ; H1: Group fractional pixels per pixel
>  v=[reform(xmap,1,npixmax),reform(ymap,1,npixmax)]
>   h1=hist_nd(v,1,MIN=[0,0],MAX=[imgs[1]-1,imgs[2]-1],REVERSE_I NDICES=r1)
>  ; h1 has dimensions of output image, and each pixel contains the number of pixels mapped into
it
>  ; Fractional pixels that map in pixel i: v[*,r1[r1[i] : r1[i+1]-1]]
>
>  ; H2: Histogram of number of fractional pixels per pixel
>  h2=histogram(h1,omax=omax2,omin=omin2,REVERSE_INDICES=r2)
>
>  ; Loop over the different fractional pixel counts:
>  ; handle repeat counts >0
>  i0 = omin2>1
>  for i=i0,omax2-omin2 do begin
>   if r2[i] ne r2[i+1] then begin
>    ;-----------Get output pixels + values to assign:-------------
>
>    ; indices in h1 with pixel count= i+omin2
>    indh1=r2[r2[i] : r2[i+1]-1]
>
>    ; indices in v with pixel count= i+omin2
>    ; since we know the pixel count:
>    ; r1[indh1] : r1[indh1+1]-1
>    ; <>
>    ; r1[indh1] : r1[indh1]+pixelcount-1
>    nc=omin2+i
>    indh1=rebin(r1[indh1],h2[i],nc,/SAMPLE)+ $
>     rebin(lindgen(1,nc),h2[i],nc,/SAMPLE)
>    npix=h2[i]*nc
>    indh1=r1[indh1]
>
>    ; Devide pixels in 9 neighbors (5 will get 0% of pixel)
>    ; first pixel of the 4 receiving output pixels
>    outpix=floor(v[*,indh1])
>    ; difference between fractional pixel and first pixel
>    dxy=v[*,indh1]-outpix
>    ; area of 4 rectangles (total area=1)
>    f_pix=[dxy[0,*]*dxy[1,*], (1.-dxy[0,*])*dxy[1,*], $
>     dxy[0,*]*(1.-dxy[1,*]), (1.-dxy[0,*])*(1.-dxy[1,*])]
>    off_x=rebin([3,2,3,2],4,npix) ; [1,0,1,0]+2 for boarder
>    off_y=rebin([3,3,2,2],4,npix) ; [1,1,0,0]+2 for boarder
>
>    ; For each input pixel: 4 output pixels + values
>    ; Pixels that fall off: clip (double boarder will catch them)
>    off_x=0>(rebin(outpix[0,*],4,npix)+off_x)<(imgsinter[1]-1)
>    off_y=0>(rebin(outpix[1,*],4,npix)+off_y)<(imgsinter[2]-1)
```

```
>    outpix=off_x + imgsinter[1] * off_y
>    add=rebin(reform((*img)[indh1],1,npix),4,npix)*f_pix
>
>    ;-----------Handle multiple indices in:----------
>    ; > interimg[outpix]+=add
>    ; > fsum[outpix]+=f_pix
>
>    ; H3: find duplicate outpix's
>     h3=histogram(outpix,omax=omax3,omin=omin3,REVERSE_INDICES=r3 )
>    ; H4
>    ; skip repeat count 0 (by setting min=1)
>     h4=histogram(h3,omax=omax4,omin=omin4,min=1,REVERSE_INDICES= r4)
>    ; handle repeat count 1
>    if r4[0] ne r4[1] then begin
>     ; indices in h3 with repeat count= 1
>     indh3=r4[r4[0] : r4[1]-1]
>     ; indices in outpix with with repeat count= 1
>     indh3=r3[r3[indh3]]
>     ; Add
>     interimg[outpix[indh3[*,0]]]+=add[indh3]
>     fsum[outpix[indh3[*,0]]]+=f_pix[indh3]
>    endif
>    ; handle repeat count >1
>    for j=1,omax4-omin4 do begin
>     if r4[j] ne r4[j+1] then begin
>      ; indices in h3 with repeat count= j+omin4
>      indh3=r4[r4[j] : r4[j+1]-1]
>
>      ; indices in outpix with with repeat count= j+omin4
>      nc=omin4+j
>      indh3=rebin(r3[indh3],h4[j],nc,/SAMPLE)+ $
>       rebin(lindgen(1,nc),h4[j],nc,/SAMPLE)
>      npix=h4[j]*nc
>      indh3=r3[indh3]
>
>      ; Total values for duplicate indices and add
>      interimg[outpix[indh3[*,0]]]+=total(add[indh3],2)
>      fsum[outpix[indh3[*,0]]]+=total(f_pix[indh3],2)
>     endif
>    endfor
>    ;----------------------------
>   endif
>  endfor
>  ; fractional area-weighted average
>  outpix=fsum eq 0
>  fsum+=outpix
>  interimg/=fsum
>
```

```
>  ; cut boarders before interpolation
>  interimg=interimg[2:2+imgs[1],2:2+imgs[2]]
>  outpix=outpix[2:2+imgs[1],2:2+imgs[2]]
>
>  ; interpolation of the pixels that didn't receive anything
>  h4=histogram(outpix,min=1,REVERSE_INDICES=r4)
>  if r4[0] ne r4[1] then begin
>   ind=r4[r4[0] : r4[1]-1]
>   xmap2=ind/imgs[1]
>   ymap2=ind mod imgs[1]
>
>   TRIANGULATE, xmap2, ymap2, tr, bounds
>   gs = [1,1] ;Grid spacing: for pixels -> 1 in each direction
>   b = [0,0, imgs[1]-1, imgs[2]-1] ;Bounds: 0,0,maxx,maxy
>
>   interimg=TRIGRID(xmap2, ymap2,interimg[xmap2, ymap2],tr, gs, b, /QUINT,EXTRA = bounds)
>  endif
>
>
>  ; Replace original
>  (*img)=temporary(interimg)
```