Subject: Image warping in IDL
Posted by Wox on Wed, 08 Nov 2006 12:53:58 GMT
View Forum Message <> Reply to Message

Hello,

I have a question concerning image warping: "Is there a fast way of doing forward mapping in IDL?"

To make myself clear, some remarks:

1. IDL's WARP_TRI uses inverse mapping. It uses triangulation and surface interpolation to get a "non-integer" position in the original image for each "integer" position, ie. pixel, in the output (destination) image. The reverse mapping now involves surface interpolation of the original image at the non-integer positions.

2. The triangulation and surface interpolation step is already done, so that leaves only the mapping. However the non-integer positions in the "output" image were calculated in the triang-interpol step for each integer position, i.e. pixel, in the "original" image.
Additionally, this "triang-interpol" step uses other techniques, using external information (spline coefficients) that can't be changed.

3. Because of step 2, forward mapping has to be performed.

Possible answers to the question above (+ why there's a problem):

1. Just do the forward mapping the hard way:

```
; img: original 2D array
;loop over rows
for i=0,nrow-1 do $
resamplearr, xmap[*,i], img, interimg, ncol, 1, i*ncol

; loop over columns
for i=0,ncol-1 do $
resamplearr, ymap[i,*], interimg, img, nrow, ncol, i
; img: destination array
```

So this loops over all rows, resampeling them separate, creating an intermediate image. Then it does the same for all columns of the intermediate. For 2000x2000 arrays, you can imagine how slow it is.

2. Another possibility would be, when we have the xmap and ymap of the forward mapping, i.e. pixel [0,0] mapped to [xmap[0,0],ymap[0,0]]

...etc., to convert them for inverse mapping, i.e.
[xmap[0,0],ymap[0,0]] mapped to [0,0] ...etc.

The problem is how? You could do this:
img = WARP_TRI(xmap, ymap, indgen(ncol)#replicate(1b,nrow),
replicate(1b,ncol)#indgen(nrow) , img)

But here the number of controlpoints equals the number of pixels in img. Except for the speed, the general idea seems strange. You use the "triang-interpol" step on the "triang-interpol", if you know what I mean.

3. Maybe there is a way of making solution 2 faster. In remark 2, I stated that the "triang-interpol" is already done using external parameters. I can't change the parameters (I'm not calculating them) but maybe I could convert these parameters so "inverse" mapping xmap and ymap are calculated. The problem is, I wouldn't know how. To be specific, these parameters are coefficients of 2D splines, one for x and one for y.

This text is getting longer and longer. My apologies for this. At this point I would have to thank you for reading this in the first place. Thanks!

So the initial question becomes now:
1. Is there a way of converting forward mapping splines to inverse splines?
2. If not, is there a fast way of doing forward mapping (cfr. solution 1)?

I hope this is making any sense. And thanks again.
