Subject: Re: Image warping in IDL
Posted by JD Smith on Wed, 08 Nov 2006 17:07:40 GMT
View Forum Message <> Reply to Message

On Wed, 08 Nov 2006 15:50:58 +0100, Wox wrote:

> On Wed, 8 Nov 2006 07:01:29 -0700, David Fanning <news@dfanning.com>
> wrote:
>
>> ...but I'll have
>> to study the question for a few more days to understand
>> just what you want here. :-)
>
> The problem is not straitforward. Let me try again:
>
> 1. I have two arrays with the same dimensions:
> - input image with pixels [Xi,Yi]: this contains my image
> - output image with pixels [Xo,Yo]: this "will" contain the warped
> image
>
> 2. There are two ways of warping:
> a. Inverse warping: You have two surfaces (Xo,Yo)->Xi and (Xo,Yo)->Yi
> where (Xo,Yo) are irregular and non-integer. First these two surfaces
> are evaluated for the pixels of the output image, i.e. regular-integer
> (Xo,Yo). Now we know where each output pixel is located in the input
> image (hence the name "inverse" warping). These locations are
> non-integer, so we have to INTERPOLATE (bilinear, cubic, whatever...).
>
> => gridding + interpolation
>
> b. Forward warping: You have two surfaces (Xi,Yi)->Xo and (Xi,Yi)->Yo
> where (Xi,Yi) are irregular and non-integer. First these two surfaces
> are evaluated for the pixels of the input image, i.e. regular-integer
> (Xi,Yi). Now we know where each input pixel is located in the output
> image (hence the name "forward" warping). These locations are
> non-integer, so we have to RESAMPLE.
>
> => gridding + resampling

I don't see how forward and reverse mapping in this context are any
different from each other. You have two images, with an irregular
grid of matching "anchor" points mapping between them (I'm visualizing
warping two eyes to two other positions in an image). To map *all*
points of the input image to the output image, you triangulate
(TRIANGULATE) that irregular grid (eyes, nose, ears, etc.), lay the
triangulation down on the output surface, and use interpolation
(TRIGRID) to figure out how points get mapped from input->output in
the space between anchor points. Then INTERPOLATE actually samples

the input image at those traingulated output positions. This simply describes the steps WARP_TRI does for you (you could also do them yourself easily).

This operation is symmetric under interchange of the sets of anchor points in the input and output images. Simply swap them, and that's you're reverse map. Certainly you don't need to loop over pixels yourself. Perhaps you were looking for a convenient way to calculate both forward and reverse maps at once?

JD
