Subject: INTERPOLATE function - Question
Posted by dmfl0590 on Wed, 09 Mar 2016 12:34:01 GMT
View Forum Message <> Reply to Message

Hi all

I wrote the following code because I'm interested to understand how the INTERPOLATE (bilinear) function works.

```
Big = randomu(2,136,136)
nint = size(Big, /dimensions)

Small = fltarr(4,4)
Small[0,0]=0.1
n = size(Small, /dimensions)
n = n[1:*]

X = (n[0]-1)*findgen(nint[0])/(nint[0]-1E)
Y = (n[0]-1)*findgen(nint[0])/(nint[0]-1E)

Small_int = fltarr(nint[0],nint[1])
Small_int = INTERPOLATE(reform(Small[*,*]), X, Y, /GRID)
```

The Small array which is the array I want to interpolate has only one non-zero entry. When I interpolated from [4,4] to [136,136] I noticed that Small_int[0:44,0:44] its the non-zero part of the matrix (2025 non-zero pixels), i.e. that part of matrix affected by interpolation.

I did the same test but this time the Small=[8,8]. I interpolated to [136,136] and I got 400 non-zero pixels (i.e. Small_int[0:19,0:19]).

Does anyone knows if it's possible to know how many pixels will be affected when we interpolate our matrices?

Thanks in advance

Subject: Re: INTERPOLATE function - Question
Posted by wlandsman on Wed, 09 Mar 2016 19:52:40 GMT
View Forum Message <> Reply to Message

On Wednesday, March 9, 2016 at 7:34:04 AM UTC-5, dmfl...@gmail.com wrote:
> Hi all
>
> I wrote the following code because I'm interested to understand how the INTERPOLATE (bilinear) function works.
>
> Big = randomu(2,136,136)
> nint = size(Big, /dimensions)

>
> Small = fltarr(4,4)
> Small[0,0]=0.1
> n = size(Small, /dimensions)
> n = n[1:*]
>
> X = (n[0]-1)*findgen(nint[0])/(nint[0]-1E)
> Y = (n[0]-1)*findgen(nint[0])/(nint[0]-1E)
>
> Small_int = fltarr(nint[0],nint[1])
> Small_int = INTERPOLATE(reform(Small[*,*]), X, Y, /GRID)
>
> The Small array which is the array I want to interpolate has only one non-zero entry. When I interpolated from [4,4] to [136,136] I noticed that Small_int[0:44,0:44] its the non-zero part of the matrix (2025 non-zero pixels), i.e. that part of matrix affected by interpolation.
>

Your formulae for X and Y are incorrect. The output values currently go from 0 to 3 whereas you want them to go from 0 to 4. (In one dimension "0" refers to the left edge of the first pixel, and "4" refers to the right edge of the last pixel.)

Y = n[0]*findgen(nint[0])/nint[0]

if you do this you will the find same percentage of non-zero pixels in the small array as the big one. In your case, 1 out of 4 pixels should be nonzero so 136/4. =34 pixels in each dimension, small_int[0:33,0:33]

Subject: Re: INTERPOLATE function - Question
Posted by dmfl0590 on Thu, 10 Mar 2016 19:44:16 GMT
View Forum Message <> Reply to Message

Hi

Thank you for pointing out this. I changed the code but I still have something that I don't understand.

When Small[0,0]=0.1
I got 34x34=1156 nonzero pixels i.e. small_int[0:33,0:33] nonzero

When I changed the Small[1,0]=0.1 I didn't got 1156 nonzero pixels as I was expecting (i.e. small_int[34:67,34:67] nonzero). Please see the code below:

Big = randomu(2,136,136)
nint = size(Big, /dimensions)

Small = fltarr(4,4)
Small[1,0]=0.1

```
n = size(Small, /dimensions)
n = n[0:*]

X = n[0]*findgen(nint[0])/nint[0]
Y = n[0]*findgen(nint[0])/nint[0]

Small_int = fltarr(nint[0],nint[1])
Small_int = INTERPOLATE(reform(Small[*,*]), X, Y, /GRID)

index = WHERE(small_int gt 0, count)
print, count

> IDL  2278
```

Also for Small[3,3]=0.1 I got 4489 nonzero pixels (the other entries in the Small matrix are zero). I was expecting small_int[101:135,101:135] to be the nonzero part (34 pixels in each direction). How can we explain that? Maybe I did something wrong again...

Subject: Re: INTERPOLATE function - Question
Posted by wlandsman on Thu, 10 Mar 2016 21:43:06 GMT
View Forum Message <> Reply to Message

On Wednesday, March 9, 2016 at 2:52:45 PM UTC-5, wlandsman wrote:

>
> if you do this you will the find same percentage of non-zero pixels in the small array as the big one. In your case, 1 out of 4 pixels should be nonzero so 136/4. =34 pixels in each dimension, small_int[0:33,0:33]

OK, I will somewhat modify my previous answer. In general, there is no reason to expect the same fraction of non-zero pixels when you are *interpolating*. If you want to have the same fraction then use

```
small_int = rebin(small,nint[0],nint[1], /sample)
or
small_int = congrid(small,nint[0],nint[1] )
```

But any fractional pixel position less than 1 pixel away will be interpolated and thus non-zero. So in dimension if small[1] is non-zero, then any pixel position between 0 and 2 will be interpolated.

The corner pixel [0,0] is a special case, because all interpolated positions will have a X,Y value less than one. (Remember that [0,0] specifies the lower left hand corner of a pixel, with [0.5, 0.5] in the middle.)
