Subject: Re: Working with 2 partially overlapping images of different array sizes
Posted by aqueous0123 on Tue, 15 Jan 2002 19:19:06 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> wrote in message news:<on8zb4gtao.fsf@cow.physics.wisc.edu>...
>
> What you are asking for is 100% what the function INTERPOLATE() will
> do for you. The only trick is to make the X and Y arrays for the
> interpolation. You are trying to interpolate IMAGE2 onto the grid for
> IMAGE1, so what you want is an array of X values and Y values that
> express the IMAGE1 grid in the coordinate system of IMAGE2.
>
> ;; Length of image 2 in the X and Y directions
> lenx2 = max(lons2)-min(lons2) & leny2 = max(lats2)-min(lats2)
> ;; Pixel sizes of each image
> dx2 = lenx2/n_elements(lons2) & dy2 = leny2/n_elements(lats2)
>
> x1 = (lons1-lons2(0))/dx2
> y1 = (lats1-lats2(0))/dy2
>
> image3 = interpolate(image2, x1, y1, /grid)
>
> Based on what you said, you definitely don't want to use MISSING,
> because you want nearest neighbor on the outskirts where IMAGE2 is not
> defined. [If you really wanted to set to zero then you would say,
> MISSING=0.]
>
> Good luck,
> Craig

Thanks Craig.
I implemented your solution into the folowing fn.

```
function testArrays, $
 array1, $  ;input,
 array2, $  ;input
 xRange1, $  ;input. array1's longitudes
 yRange1, $  ;input. array1's latitudes
 xRange2, $  ;input. array2's longitudes
 yRange2, $  ;input. array2's latitudes
 MISSING=missing ;input, optional. fill val for interpolation

  ;paste array2 into array1 geographically, regridding array2
  ; to array1's array space dimensions

  ;x/yRange arrays should have same # elements as x/y of
  ; input arrays for this algorithm. user may have just
```

```
  ; passed lat/lon min/max values
  sz = size(array1, /dimensions)
  xRange1 = congrid(xRange1,sz[0],/interp,/minus_one)
  yRange1 = congrid(yRange1,sz[1],/interp,/minus_one)
  sz = size(array2, /dimensions)
  xRange2 = congrid(xRange2,sz[0],/interp,/minus_one)
  yRange2 = congrid(yRange2,sz[1],/interp,/minus_one)

  ;array2's deltas in...
  xDelta2 = max(xRange2)-min(xRange2) ; ...X
  yDelta2 = max(yRange2)-min(yRange2) ; ...Y

  ;Pixel sizes of each image
  sz = size(array2, /dimensions)
  xSize2 = sz[0]
  ySize2 = sz[1]
  dx2 = xDelta2 / xSize2
  dy2 = yDelta2 / ySize2

  x1 = (xRange1 - xRange2[0]) / dx2
  y1 = (yRange1 - yRange2[0]) / dy2

  array3 = interpolate(array2, x1, y1, /grid, MISSING=missing)

  return, array3
end
```

Now, I do the following to test.

```
IDL> a = indgen(4,4) ;4x4 array
IDL> b = [[100,200],[300,400]] ;2x2 array
IDL> aX = [2,5] ;a's 'longitudes' span 2 to 5
IDL> aY = [1,4]] ;a's 'latitudes' span 1 to 4
IDL> bX = [4,5] & bY = [2,3] ;b's lon/lat spans
IDL> print, testArrays(a,b, aX,aY,bX,bY)
    100     100     100     200
    100     100     100     200
    300     300     300     400
    300     300     300     400
% Program caused arithmetic error: Integer divide by 0
```

Looks good. Just like I wanted. b is in a at the right-center
and rest of a is filled with nearest neighbor.

What happens if b spans same range as a, even tho it's smaller in
array dimensions (meaning b has larger "pixels" than a.

```
IDL> aX = [2,5] & aY = [1,4] & bX = [2,5] & bY = [1,4]
IDL> print, testArrays(a,b, aX,aY,bX,bY)
    100     200     200     200
    300     400     400     400
    300     400     400     400
    300     400     400     400
```

Oops. Looks like it put b in upper left corner of a. Also, no more div by 0 error. So its easier to see, I'll test with a missing value fill, like you suggest.

```
IDL> print, testArrays(a,b, aX,aY,bX,bY, missing=-1)
    100     200      -1      -1
    300     400      -1      -1
     -1      -1      -1      -1
     -1      -1      -1      -1
```

b's data not 'stretched' over a like i would have expected. b's values ([[100,200],[300,400]]) should be in the 4 corners and all other elements interpolated between. What if my data were floats, like they'll probably be in reality.

```
IDL> print, testArrays(float(a),float(b),
float(aX),float(aY),float(bX),float(bY), missing=-1)
      100.000      166.667     -1.00000     -1.00000
      233.333      300.000     -1.00000     -1.00000
     -1.00000     -1.00000     -1.00000     -1.00000
     -1.00000     -1.00000     -1.00000     -1.00000
```

Now I'm confounded. Looks like its trying to interpolate to 4 by 4, but filled as missing all outside b's [2,2] array size.

Ok, Insert b back to the right-center, like the 1st try above, but with floats and a missing fill.

```
IDL> aX = [2,5] & aY = [1,4] & bX = [4,5] & bY = [2,3]
IDL> print, testArrays(float(a),float(b),
float(aX),float(aY),float(bX),float(bY), missing=-1)
     -1.00000     -1.00000     -1.00000     -1.00000
     -1.00000     -1.00000      100.000     -1.00000
     -1.00000     -1.00000     -1.00000     -1.00000
     -1.00000     -1.00000     -1.00000     -1.00000
% Program caused arithmetic error: Floating divide by 0
% Program caused arithmetic error: Floating illegal operand
```

Huh?