Subject: Re: Mapping image into a polar-square coordinate
Posted by pgrigis on Wed, 09 Jul 2008 13:29:21 GMT
View Forum Message <> Reply to Message

I suggest using 2-dimensional interpolation (for instance, "bilinear")
to interpolate x,y data to radius and angle. You only need to call it
once, so it should be fast.

Ciao,
Paolo

cmejiapr...@gmail.com wrote:
> Hi programmers,
>
> I have an image and I want to map an annulus of it (matrix 981X 981)
> onto a rectangular axes whose columns are the angle, and the rows are
> the radius to the central pixel. I tried:
>
> ;data has the image
> xx1 = findgen(4096,10)*0.
> for i=0,1023 do begin
> roll=i*360./4096.
> SB=rot(data,-roll,1,490.5,490.5,cubic=-0.5,missing=-1,/pivot)
> xx1[i,*]=SB[50:59,490]
> for j=0,9 do xx1[i+2048,j]=SB[930-j,490]
> xx1[i+3072,*]=SB[490,50:59]
> for j=0,9 do xx1[i+1024,j]=SB[490,930-j]
> endfor
>
> But it takes too long to run, i need something faster. Any advise?
> Thanks

Subject: Re: Mapping image into a polar-square coordinate
Posted by pgrigis on Wed, 09 Jul 2008 14:47:05 GMT
View Forum Message <> Reply to Message

seems like my previous post got lost...
anyway I am suggesting to use interpolation (i.e. BILINEAR)
to convert from cartesian to polar coordinates.
That should be pretty fast.

Ciao,
Paolo

cmejiapr...@gmail.com wrote:
> Hi programmers,
>

> I have an image and I want to map an annulus of it (matrix 981X 981)
> onto a rectangular axes whose columns are the angle, and the rows are
> the radius to the central pixel. I tried:
>
> ;data has the image
> xx1 = findgen(4096,10)*0.
> for i=0,1023 do begin
> roll=i*360./4096.
> SB=rot(data,-roll,1,490.5,490.5,cubic=-0.5,missing=-1,/pivot)
> xx1[i,*]=SB[50:59,490]
> for j=0,9 do xx1[i+2048,j]=SB[930-j,490]
> xx1[i+3072,*]=SB[490,50:59]
> for j=0,9 do xx1[i+1024,j]=SB[490,930-j]
> endfor
>
> But it takes too long to run, i need something faster. Any advise?
> Thanks

Subject: Re: Mapping image into a polar-square coordinate
Posted by Camilo Mejia on Wed, 09 Jul 2008 19:01:12 GMT
View Forum Message <> Reply to Message

On Jul 9, 7:47 am, pgri...@gmail.com wrote:
> seems like my previous post got lost...
> anyway I am suggesting to use interpolation (i.e. BILINEAR)
> to convert from cartesian to polar coordinates.
> That should be pretty fast.
>
> Ciao,
> Paolo
>
> cmejiapr...@gmail.com wrote:
>> Hi programmers,
>
>> I have an image and I want to map an annulus of it (matrix 981X 981)
>> onto a rectangular axes whose columns are the angle, and the rows are
>> the radius to the central pixel. I tried:
>
>> ;data has the image
>> xx1 = findgen(4096,10)*0.
>> for i=0,1023 do begin
>> roll=i*360./4096.
>> SB=rot(data,-roll,1,490.5,490.5,cubic=-0.5,missing=-1,/pivot)
>> xx1[i,*]=SB[50:59,490]
>> for j=0,9 do xx1[i+2048,j]=SB[930-j,490]
>> xx1[i+3072,*]=SB[490,50:59]
>> for j=0,9 do xx1[i+1024,j]=SB[490,930-j]

>> endfor
>
>> But it takes too long to run, i need something faster. Any advise?
>> Thanks

yeah, but i dont know how to extract a rectangular matrix which rows
are radius and columns are angles

Subject: Re: Mapping image into a polar-square coordinate
Posted by jschwab@gmail.com on Wed, 09 Jul 2008 19:43:58 GMT
View Forum Message <> Reply to Message

> yeah, but i dont know how to extract a rectangular matrix which rows
> are radius and columns are angles

Paolo's suggestion of bilinear is a good one.

The best thing to do is construct a polar coordinate system and then
transform that into a rectangular system that is equivalent to your
pixel indices.

Suppose there is a rectangular coordinate system, centered on the
middle pixel of your 981 x 981 data. Then if we want to extract the
annulus which is between 100 and 200 pixels from the center, we could
do something like this.

---

image: 981 x 981 (same as your ``data'' array)
new_image: 4096 x 10

;; first construct the equivalent polar coordinates

min_r = 100.0
max_r = 200.0

;; this is theta = [0, 2*pi)
new_th = rebin(dindgen(4096) / 4096d * (2d * !dpi), 4096, 10)

;; this is r = [r_min, r_max]
new_r = rebin(transpose((max_r - min_r) * dindgen(10) / 9d + min_r),
4096, 10)

;; now convert to rectangular coordinates
;; and shift such that the origin lies not at the center
;; but at image[0,0]

```
new_x = new_r * cos(new_th) + 490.0
new_y = new_r * sin(new_th) + 490.0

;; new_x and new_y are fractional pixel coordinates
;; use bilinear to extract the values

new_img = bilinear(image, new_x ,new_y)
```

Hope that helps,
Josiah

Subject: Re: Mapping image into a polar-square coordinate
Posted by Camilo Mejia on Wed, 09 Jul 2008 20:00:21 GMT
View Forum Message <> Reply to Message

On Jul 9, 12:43 pm, "jsch...@gmail.com" <jsch...@gmail.com> wrote:
>> yeah, but i dont know how to extract a rectangular matrix which rows
>> are radius and columns are angles
>
> Paolo's suggestion of bilinear is a good one.
>
> The best thing to do is construct a polar coordinate system and then
> transform that into a rectangular system that is equivalent to your
> pixel indices.
>
> Suppose there is a rectangular coordinate system, centered on the
> middle pixel of your 981 x 981 data. Then if we want to extract the
> annulus which is between 100 and 200 pixels from the center, we could
> do something like this.
>
> -----
>
> image: 981 x 981 (same as your ``data'' array)
> new_image: 4096 x 10
>
> ;; first construct the equivalent polar coordinates
>
> min_r = 100.0
> max_r = 200.0
>
> ;; this is theta = [0, 2*pi)
> new_th = rebin(dindgen(4096) / 4096d * (2d * !dpi), 4096, 10)
>
> ;; this is r = [r_min, r_max]
> new_r = rebin(transpose((max_r - min_r) * dindgen(10) / 9d + min_r),

> 4096, 10)
>
> ;; now convert to rectangular coordinates
> ;; and shift such that the origin lies not at the center
> ;; but at image[0,0]
>
> new_x = new_r * cos(new_th) + 490.0
> new_y = new_r * sin(new_th) + 490.0
>
> ;; new_x and new_y are fractional pixel coordinates
> ;; use bilinear to extract the values
>
> new_img = bilinear(image, new_x ,new_y)
>
> -----
>
> Hope that helps,
> Josiah

Thanks a lot Josiah and Paolo, it works awesome

Camilo
