
Subject: Re: Finding all angles within a range of directions; an algorithm question
Posted by [Mark Zimmerman](#) on Thu, 11 Apr 2002 21:29:17 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <fd12a3f3.0204111301.41ce7b31@posting.google.com>, tbowers0@yahoo.com wrote:

```
> Say I have a 3D array of observation data of 'Brightness' over the
> hemisphere of theta,phi angles (theta=0 is straight up, phi=0 is
> North, and 3rd dim is timesteps).
>
> data = findgen(3,4,10) ;simple example with 3 theta angles, 4 phi
> angles, 10 timesteps
> theta = [0,30,60] ;a *very* simple example
> phi = [0,90,180,270]
> timestep = indgen(10)
>
> If I have a flat plate facing an arbitrary angle, how do I find all
> brightnesses that fall on its surface. In other words, all
> brightness[*,*] that come from angles within +/- 90 degree hemisphere
> of plate's surface (of course, all angles are really in radians, but
> listed here as degrees for clarity). The only solution I come up with
> requires:
>
> 1) reforming the brightness data to a list of
> theta,phi,timestep,brightness quadruplets (now a 2D array
> [4,n_thetas*n_phis*n_timesteps]; about 7 or 8 lines of code
> reform()ing and transpose()ing). Doing this cause I'll use where() in
> a moment
>
> 2) convert the theta,phi polar coordinates to x,y,z cartesian
> coordinates by:
>   brightnessAnglesCartesian = sin(brightness[0,*]) *
>   cos(brightness[1,*]), sin(brightness[0,*]) * sin(brightness[1,*]),
>   cos(brightness[0,*])
>
> 3) convert the plate's theta,phi polar coordinate facing direction
> (its 'normal') to x,y,z cartesian coordinate by same formula to create
> plateAngleCartesian, a 3-element vector
>
> 4) compute all angles psi that plate normal (plateAngleCartesian)
> makes with all brightness angles (brightnessAnglesCartesian) by:
>   psi = acos(plateAngleCartesian # brightnessAnglesCartesian) ;acos(dot
>   product of 3x1 plate angle vector and 3xN brightness angle vectors)
>
> 5) indices = where(psi le !pi/2.0)
>
> 6) Now I can work with these angles: brightness[3,indices]) = 0.0
> ;brightnesses in 4th column
```

>
> This seems awfully circuitous. I'm using an interactive interface to
> rotate the plate with the mouse so calculation speed is critical and I
> think my method is way too slow (and not very elegant either). Could
> anyone please advise on a better way to do this? One thing I've
> learned is that when dealing with angles and rotations, there are
> usually very quick and clever alternatives than my usual brutish
> approach.
>
> Many thanks in advance

One simplification: in (4), skip taking the acos of the dot products.
All of the angles you want will have positive dot products; just throw
out the negative ones.

-- Mark

Subject: Re: Finding all angles within a range of directions; an algorithm question
Posted by [Struan Gray](#) on Mon, 15 Apr 2002 13:18:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

@yahoo.com writes:

> Could anyone please advise on a better way to do this?

Construct a rotation matrix which describes a rotation of the
original angular coordinates into the 'reference frame' of the plate,
i.e. which translates theta and phi into theta* and phi* where theta*
is the angle from the plate normal.

Then just do a matrix multiply (fast in IDL) and find the items
with theta* less than 90 degrees. You can probably speed it up by not
bothering to calculate phi* at all, and do a matrix multiply with
a vector to just find theta*.

You can either keep track of the time values by suitable identity
elements in the rotation matrix, or separate out the angular
information and use where/histogram/compare* to find the indices of the
elements you want.

Struan

Subject: Re: Finding all angles within a range of directions; an algorithm question
Posted by [tbowers0](#) on Tue, 16 Apr 2002 15:56:35 GMT

Wow Struan! Your explanation was a bit beyond me. So, instead of just replying "Uhh.. Huh?" I did some surfing on rotation matrices and stuff and found the Matrix and Quaternion FAQ at <http://skal.planet-d.net/demo/matrixfaq.htm>. Educated myself a bit, but I'm still unclear. If I understand:

Struan Gray <struan.gray@sljus.lu.se wrote

- > Construct a rotation matrix which describes a rotation of the
- > original angular coordinates into the 'reference frame' of the plate,
- > i.e. which translates theta and phi into theta* and phi* where theta*
- > is the angle from the plate normal.

So I need to build a polar coord. rotation matrix for the plate normal's current 'pointing' direction, right?

I can't find a formula for this in polar coords. The above FAQ (Question 35) talks only about "Euler angles" which I think are cartesian xyz.

- > Then just do a matrix multiply (fast in IDL) and find the items
- > with theta* less than 90 degrees. You can probably speed it up by not
- > bothering to calculate phi* at all, and do a matrix multiply with
- > a vector to just find theta*.

I think you mean matrix multiply the above mentioned polar angle rotation matrix with some other matrix or vector, but I'm not sure what? All I have are 2 arrays of theta and phi. To clarify my example, I have vector of theta angles (shown across top), vector of phi azimuthal angles (shown here down left side), and 2D array of float data values for each angle.

	0	45	90	135	180
0	7.0	5.0	1.1	0.5	0.1
90	9.0	6.0	1.5	0.9	0.1
180	7.0	5.5	1.2	0.5	0.1
270	3.0	2.0	0.8	0.2	0.0

and say my plate rotation theta,phi angle is 45,0 (45 degrees from vertical and due North)

Or, in IDL speak:

```
theta = [0,45,90,135,180]
phi   = [0,90,180,270]
B = [[7.0,5.0,1.1,0.5,0.1], $
     [9.0,6.0,1.5,0.9,0.1], $
     [7.0,5.5,1.2,0.5,0.1], $
     [3.0,2.0,0.8,0.2,0.0]]
```

plateRotationAngle = [45,0]

So I need to build a polar coord. rotation matrix for plateRotationAngle and multiply this by some other matrix? And then just do something like where(result lt 90)? I'm not sure what you mean here.

- > You can either keep track of the time values by suitable
- > identity elements in the rotation matrix, or separate out the
- > angular information and use where/histogram/compare* to find
- > the indices of the elements you want.

Hmm.. not sure at all what you mean here, except the possible use of where. What's compare?

I must thank you very much Straun. This is becoming extremely educational! Many thanks for your help on this!
todd

Subject: Re: Finding all angles within a range of directions; an algorithm question
Posted by [tbowers0](#) on Tue, 16 Apr 2002 16:08:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

- > One simplification: in (4), skip taking the acos of the dot products.
- > All of the angles you want will have positive dot products; just throw
- > out the negative ones.
- >
- > -- Mark

Oh! Good point. I still would like to keep it general though because I may sometimes use a smaller 'acceptance angle'. Eg, get all values with 45 degree half angle rather than the plate's 90 degrees. But, it definitely seems appropriate to put a check on the front to see and save from unnecessary function calls, like

```
if (sensorHalfAngle eq 90) then begin
  ;just dot product and where(dotProduct ge 0.0),
  ;no need to acos()
  ...
endif else begin
  ;angle not over a flat plat, so dot product
  ;then calc. psi=acos(dotProduct), then where(psi le 90.0)
  ...
endelse
```

Thanks Mark
