Subject: fitting an ellipsoid to a 3D volume using eigenanalysis
Posted by nixrajguru on Fri, 12 Nov 2004 16:11:10 GMT
View Forum Message <> Reply to Message

There's a nice piece of code in the dfanning site for fitting an ellipse to 2D data. I tried to extend it to try and fit an ellipsoid to a 3D points ...

It seems that the idea is to find an ellipsoid whose center is at the center of mass and whose axes represent the standard deviation of the points in the principal axis directions.

Is there an algorithm / code for fitting an ellipsoid to 3D points available? I hope someone can shed light on this topic.

I was having trouble manipulating the eigen values and eigen vectots to get the correct orientation in 3d...

Here's what I wrote based on the 2D implementation on dfanning.com:

```
;*****************************************************
npts = N_Elements(indices)

; Calculate the mass distribution tensor.

i11 = Total(yy[indices]^2) / npts
i22 = Total(xx[indices]^2) / npts
i33 = Total(zz[indices]^2) / npts
i12 = Total(xx[indices] * yy[indices]) / npts
i13 = Total(yy[indices] * zz[indices]) / npts
i23 = Total(xx[indices] * zz[indices]) / npts
tensor = [[i11, i12, i13],[i12,i22, i23],[i13,i23, i23]]
;tensor = [[i11, i12],[i12,i22]]
;Find the eigenvalues and eigenvectors of the tensor.
stop
evals = Eigenql(tensor, Eigenvectors=evecs)

; The semi-major and semi-minor axes of the ellipse
; are obtained from the eigenvalues.
semimajor = Sqrt(evals[0]) * 2.0
semiminor = Sqrt(evals[1]) * 2.0
semiinter = sqrt(evals[2]) * 2.0

; We want the actual axes lengths.

major = semimajor * 2.0
minor = semiminor * 2.0
intermed = semiinter *  2.0
semiAxes = [semimajor, semiminor, semiinter]
```

```
axes = [major, minor, intermed]

; The orientation of the ellipse is obtained from the first
eigenvector.

evec1 = evecs[*,0]
evec2 = evecs[*,1]
evec3 = evecs[*,2]
;These are the angles obtained from the eigen vectors:
orientation1 = ATAN(evec1[1], evec1[0]) * 180. / !Pi - 90.0
orientation2 = ATAN(evec2[1], evec2[0]) * 180. / !Pi ;- 90.0
orientation3 = ATAN(evec3[1], evec3[0]) * 180. / !Pi ;- 90.0
Npoints = 200
; Divide a circle into Npoints.
theta = 2 * !PI * (Findgen(npoints) / (npoints-1))
;phi is between 0 and pi...divide into Npoints
phi = !PI * (Findgen(npoints) / (npoints-1))

; Parameterized equation of ellipse.
x =  semimajor * Cos(theta) * Sin(phi)
y =  semiminor * Sin(theta) * Sin (phi)
z =  semiinter * Cos(phi)

; Position angle in radians.
t1 = orientation1 / !RADEG
t2 = orientation2 / !RADEG
t3 = orientation3 / !RADEG

; Sin and cos of angle.
cos_t1 = Cos(t1)
sin_t1 = Sin(t1)
cos_t2 = Cos(t2)
sin_t2 = Sin(t2)
cos_t3 = Cos(t3)
sin_t3 = Sin(t3)

; Rotate to desired position angle.
xprime = xcm + (x * (cos_t1 * cos_t3 - sin_t1 * sin_t2 * sin_t3)) $
   - (y * sin_t1 * cos_t2) $
   + (z * (cos_t1 * sin_t3 - sin_t1 * sin_t2 * cos_t3))

yprime = ycm + (x * (sin_t1 * cos_t3 - cos_t1 * cos_t2 * sin_t3)) $
   + (y * (cos_t1 * cos_t2)) $
   + (z * (sin_t1 * sin_t3 - cos_t1 * sin_t2 * cos_t3))

zprime = zcm - x * (cos_t2 * sin_t3) + y * sin_t2 + z * (cos_t2 *
sin_t3)
```

```
; Extract the points to return.
pts = FltArr(3, N_Elements(xprime))
pts[0,*] = xprime
pts[1,*] = yprime
pts[2,*] = zprime

;*********************************
```