3DROTATE

Consider the picture as if it were on a horizontal rectangle locate on the Z=0 plane with its upper left corner at the origin. This means that points on the picture have values of (X,Y,0)

Consider the perspective transformation as if it were the process of capturing a picture by a frame camera located a distance Zc=f above the picture, where f is the focal length of the camera, which is determined by the fov (field of view) defined by the diagonal dimension of the image. So that

tan(fov/2) = (sqrt(width^2 + height^2)) / (2 * f)

or

f = diag / (2 * tan(pef * fov / 2))

where fov = the equivalent fov for 35mm picture frame whose dimensions are 36mm x 24mm. Thus

fov = 180 * atan(36/24) / pi (which is approx. 56 degrees)

and we have added pef into the equation as the perspective exaggeration factor, thus increasing or decreasing the effective fov used to calculate f.

Note that the camera is looking straight down along the -Z direction.

The perspective equations in 3D are defined as:

(x,y,f) = M (X',Y',Z'-Zc)

which includes an implicit divide of the first two terms by the third.

where M is the camera orientation matrix, which is the identity matrix with M22=-1. Because the camera is looking straight down, M is a reflection about Z. That is the camera sees coordinates with +Z values as closer and with -Z values as further away.

Thus M is:

M0 = (1 0 0)
M1 = (0 1 0)
M2 = (0 0 -1)

Also

(X',Y',Z') = R (X,Y,0) are the rotated points determined by the composite rotation matrix from the three rotations, pan, tilt, and roll (in any order). We define these three rotation angles as:

pan = right hand positive rotation about Y axis
tilt = right hand negative rotation about X axis
roll = right hand positive rotation about Z axis

Thus the three rotation matrices become:

Rp0 = (cospan 0 sinpan)
Rp1 = (0 1 0)
Rp2 = (-sinpan 0 cospan)

Rt0 = (1 0 0)
Rt1 = (0 costilt sintilt)
Rt2 = (0 -sintilt costilt)

Rr0 = (cosroll sinroll 0)
Rr1 = (-sinroll cosroll 0)
Rr2 = (0 0 1)

So now we can express the perspective equation as:

(x,y,f) = M R (X,Y,0)

But to avoid a divide by zero in the implicit divide (in the final equations below), we must convert (X,Y,0) to (X,Y,1). To do this we note that

(X,Y,0) = (X,Y,1) - (0,0,1)

Thus the perspective equations become:

(x,y,f) = M {R [(X,Y,1) - (0,0,1)] - (0,0,Zc)}

But

R [(X,Y,1) - (0,0,1)] = R [II (X,Y,1) - S (X,Y,1)]

where I is the identity matrix

II0 = (1 0 0)
II1 = (0 1 0)
II2 = (0 0 1)

And S is a matrix of all zeros except for S22=1

S0 = (0 0 0)
S1 = (0 0 0)
S2 = (0 0 1)

So that combining we get

IMS = (II - S)

Or

IMS0 = (1 0 0)
IMS1 = (0 1 0)
IMS2 = (0 0 0)

So the perspective equations become:

$(x,y,f) = M \{R\ IMS\ (X,Y,1) - D\ (X,Y,1)\}$

where D is a matrix of all zeros except for D22=Zc=f.

Thus the perspective equations become:

$(x,y,f) = M\ [(R\ IMS) - D]\ (X,Y,1) = M\ T\ (X,Y,1)$

where

(R IMS) is just the Rotation matrix R with its third column all zeros and T is then simply R with its third column just (0, 0, -f).

Thus

T0 = (R00 R01 0)
T1 = (R10 R11 0)
T2 = (R20 R21 -f)

Now we want to convert (x,y,f) to (u,v,1) pixels for output coordinates and we want to convert (X,Y,1) to (i,j,1) pixels for input coordinates.

These last two transformations are just affine transformations, namely:

$(x,y,f) = A\ (u,v,1)$

where

$x = sx * (u - du)$
$y = -sy * (v - dv)$ (as lines increase downward)

where we do the offset before the scaling to get results to come out right.

and

(X,Y,1) = B (i,j,1)

where

X = (i – di)
y = - (j – dj) (as lines increase downward)

(we will ignore a change of scale in the input)

So that B is just a matrix of offsets

B0=(1 0 -di)
B1=(0 -1 dj)
B2=(0 0 1)

where

idx = user supplied input image i offset relative to the picture center
idy = user supplied input image j offset relative to the picture center

di = idx + (width – 1)/2
dj = idy + (height – 1)/2

An similarly we have

A0 = (sx 0 sx*(-du-di))
A1 = (0 -sy sy*(dv+dj))
A2 = (0 0 -f)

where

du = odx = user supplied output image u offset relative to the picture center
dv = ody = user supplied output image v offset relative to the picture center

sx is the x output scale factor (defined by user supplied zoom)
sy is the y output scale factor (defined by user supplied zoom)

sx = sy = 1/zoom for zoom positive
sx = sy = - zoom for zoom negative

So that the perspective transformation equations become

A (u,v,1) = M T B (i,j,1)

or

(u,v,1) = A^{-1} M T B (i,j,1)

where A^{-1} is the inverse matrix of A.

We can compute A^{-1} simply enough manually from the adjoint matrix (or the matrix of cofactors of A) divided by the determinant of A.

Thus we get

A^{-1}0 = (1/sx 0 -A02/(sx * f))
A^{-1}1 = (0 -1/sy A12/(sy * f))
A^{-1}2 = (0 0 1/f)

But as M is also a nearly empty matrix, we might as well do the matrix multiply A^{-1} M manually to get

A^{-1}M0 = (1/sx 0 A02/(sx * f))
A^{-1}M1 = (0 -1/sy -A12/(sy * f))
A^{-1}M2 = (0 0 -1/f)

Thus the perspective transformation equation becomes:

(u,v,1) = P (i,j,1)

where

$P = A^{-1} M T B$

So we just need to do the matrix multiplies on these four matrices.

Then we invert P to get $Q = P^{-1}$ to get the inverse transformation matrix. This again can be achieved from the adjoint method.