Rotation

Last lecture we addressed camera rotation. This lecture we will look at camera rotation. For example, moving cameras may pan (side to side), pitch (up and down), or roll (rotate about optical axis).

Rotation about y axis

Consider first the case that we *rotate the camera coordinate system* around the Y axis, with an angular velocity of Ω radians per second. Any point (X_0, Y_0, Z_0) will sweep out a circle centered on the Y axis, when represented in the camera's coordinate system. Specifically, at time t the point would be at $(X(t), Y(t), Z(t))$,

$$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & 0 & \sin(\Omega t) \\ 0 & 1 & 0 \\ -\sin(\Omega t) & 0 & \cos(\Omega t) \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

Note that if we consider the line from the origin through point (X_0, Y_0, Z_0), then that line will sweep out a cone that is centered on the Y axis. From the perspective of the camera, the path of the point relative to the camera's coordinate system is the same, whether the camera is rotating and the point is fixed in space, or the camera is fixed and the point rotates around the above circle.

What is the the path of the point when it is projected in the image plane? From geometric intuition, the path of the point will be the intersection of the above cone with the projection plane $Z = f$. This intersection will give a parabola in the image plane. The image velocity vectors will be tangent to this parabola. Let's compute what the image velocities are.

As in the translation case, we compute

$$(v_x, v_y) = \frac{d}{dt}(x(t), y(t))|_{t=0}$$

So,

$$\begin{aligned} v_x &= f\frac{d}{dt}(\frac{X_0(t)}{Z_0(t)}) \mid_{t=0} \\ &= f\frac{(\Omega Z_0^2 + \Omega X_0^2)}{Z_0^2}, \text{ since } \frac{d}{dt}X(t)|_{t=0} = \Omega Z_0, \quad \frac{d}{dt}Z(t)|_{t=0} = \Omega X_0 \\ &= f\Omega(1 + (\frac{x}{f})^2) \end{aligned}$$

So, we see that the rotation gives us two components to the x velocity. There is a constant motion, plus there is a second order motion. The first term is intuitively easy to understand: when we pan the camera, we move all the points in the same direction. The second component is less obvious. It arises because we are projecting onto a plane that is rotating with the camera.

Next, we calculate v_y. You might think that rotating about the Y axis gives no y component to the velocity, but this intuition turns out to be incorrect. (Remember, we are intersecting an image plane with a cone, and this gives a parabola in the image plane.) The y component of velocity turns

out to be:

$$\begin{aligned} v_y &= f\frac{d}{dt}(\frac{Y_0(t)}{Z_0(t)}) \mid_{t=0} \\ &= f\frac{\Omega X_0 Y_0}{Z_0^2} \\ &= \Omega\frac{xy}{f} \end{aligned}$$

which is second order in x, y. We will see an example below (see plots).

Rotation about x axis

Rotating about the x axis (called "pitch") leads to similar equations, except that the x and y axes are swapped:

$$(v_x, v_y) = \Omega(\frac{xy}{f}, f(1 + (\frac{y}{f})^2)).$$

The dominant component is the constant y component Ωf, and there are second order components there as well which play a role away from the optical axis.

Rotation about z axis

Finally, we consider the camera rotation about the optical axis. Again let the angular velocity be Ω. Then,

$$\begin{bmatrix} X_0(t) \\ Y_0(t) \\ Z_0(t) \end{bmatrix} = \begin{bmatrix} \cos(\Omega t) & -\sin(\Omega t) & 0 \\ \sin(\Omega t) & \cos(\Omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

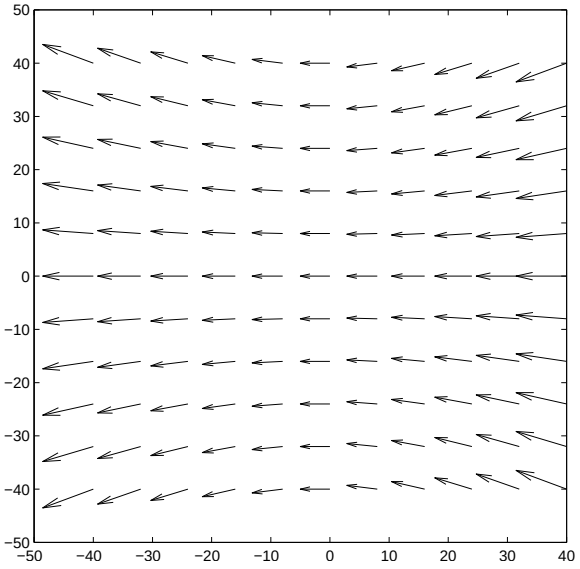
Again the point travels on a circle, this time centered on the Z axis. And again, the line from the origin through this point sweeps out a cone (centered on the Z axis).

Verify for yourself that:

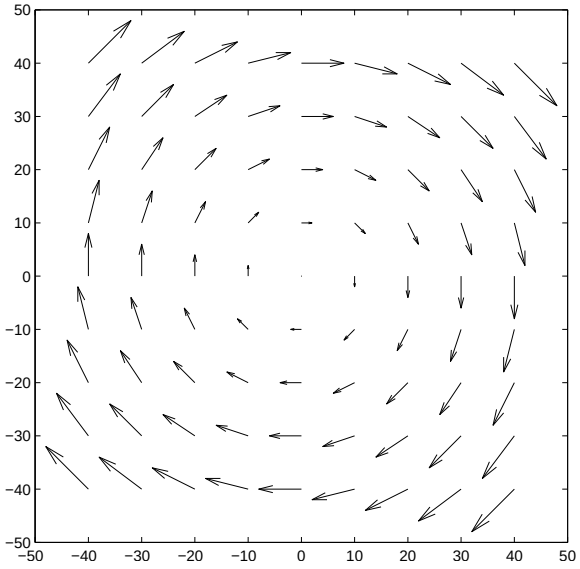
$$(v_x, v_y) = \Omega(-y, x)$$

This defines a vector field such that the motion is along circles and the image speed of a point is proportional to the radius of the circle on which an image point lies.

A key difference between the motion fields that are caused by camera translation versus camera rotation is that, whereas the translation fields depend on the depth of points in the scene, *the rotation fields do not*. This is interesting (and perhaps surprising). Panning or rolling the camera gives you motion, but the motion can you nothing about how far away the scene points are. To get information about the distance to scene points from motion, you would need to translate the camera.



$\boldsymbol{\Omega} = (0, 1, 0)$
pan



$\boldsymbol{\Omega} = (0, 0, 1)$
roll

Rotation matrices (some review, some new stuff)

In general, by a *3D rotation matrix*[1], we will just mean a real 3×3 invertible matrix $\mathbf{R}$ such that $\det \mathbf{R} = 1$ and

$$\mathbf{R}^T = \mathbf{R}^{-1}$$

i.e.

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I},$$

where $\mathbf{I}$ is the identity matrix. Note that the rows and columns of $\mathbf{R}$ are orthogonal to each other and of unit length, i.e. *orthonormal*.

Rotation matrices preserve the length of vectors, as well as the angle between two vectors. To see this, let $\mathbf{p}_1$ and $\mathbf{p}_2$ be two vectors, possibly the same. Then

$$(\mathbf{R}\mathbf{p}_1) \cdot (\mathbf{R}\mathbf{p}_2) = \mathbf{p}_1^T\mathbf{R}^T\mathbf{R}\mathbf{p}_2 = \mathbf{p}_1^T\mathbf{p}_2 = \mathbf{p}_1 \cdot \mathbf{p}_2.$$

If $\mathbf{p}_1 = \mathbf{p}_2$ then we see vector length is preserved. More generally, since the dot product of unit vectors is just the cosine of the angle between them, we see that the angle between vectors is preserved.

Why did we require that $\det \mathbf{R} = 1$? An example of an orthonormal matrix that does *not* have this property is:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[1]You may recall from your linear algebra background that such matrices are special cases of *unitary* matrices (which can have complex elements and can have determinant -1).

This matrix performs a mirror reflection of the scene about the yz plane. The length of vectors is preserved, as is the angle between any two vectors. But the matrix is not a rotation.

Another orthonormal matrix that fails the "det $\mathbf{R} = 1$" condition is

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which swaps the first two variables. This is a reflection, rather than a rotation.

Consider a matrix $\mathbf{R}$ whose rows define the directions of the camera's XYZ axes, written in some other coordinate system. For example, we might have a "world coordinate system" defined by gravity and two other directions, as we discussed last class in the context of vanishing points.

Let the three rows of $\mathbf{R}$ be $\mathbf{u}$,$\mathbf{v}$,$\mathbf{n}$. The third row $\mathbf{n} = (n_x, n_y, n_z)$ is the unit vector in the direction of the optical axis ("Z axis") of the camera, represented in world coordinates. The other two rows are unit vectors in the direction of the camera's x and y axes which are perpendicular to the n vector. These define the "left-right" and "up-down" directions of the camera. We can write the 3×3 rotation matrix as:

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix} .$$

Note that this matrix maps $\mathbf{u}$, $\mathbf{v}$, $\mathbf{n}$ to (1,0,0), (0,1,0), (0,0,1) respectively.

From linear algebra, you can verify that a 3D rotation matrix (as defined earlier) must have at least one real eigenvalue and, because rotation matrices preserve vector lengths, this eigenvalue must be 1. The eigenvector $\mathbf{p}$ that corresponds to this eigenvalue satisfies

$$\mathbf{p} = \mathbf{R}\mathbf{p} .$$

This vector doesn't change under the rotation. Thus, this vector is the axis of rotation defined by the matrix $\mathbf{R}$.

Most of the above should be familiar to you, and you should review your linear algebra and make sure you understand it.

Next, suppose you are given a unit vector $\mathbf{a}$ which is to be the axis of rotation, and you are given an angle θ. How would you construct the rotation matrix that rotates about axis $\mathbf{a}$ by an angle θ ?

To perform this construction, we first introduce some notation. Observe that the cross product of two vectors defines a linear transformation, namely

$$\mathbf{a} \times \mathbf{b} == \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & a_1 \\ a_2 & -a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

so we let $[\mathbf{a}]_\times$ denote the 3×3 matrix in the above equation,

$$[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}.$$

If we want to rotate a general vector $\mathbf{b}$ about axis $\mathbf{a}$ by angle θ, we can decompose $\mathbf{b}$ into two vectors, one parallel to $\mathbf{a}$ and one perpendicular to $\mathbf{a}$

$$\mathbf{b} = \mathbf{a}\mathbf{a}^T\mathbf{b} + (\mathbf{I} - \mathbf{a}\mathbf{a}^T)\mathbf{b}$$

where we are assuming $\mathbf{a}$ is of unit length. (Note that $\mathbf{a}\mathbf{a}^T$ is a 3×3 matrix.) The first component is unaffected by the rotation, since we are rotating it about its own axis. For the second component, the rotation can be written as the sum

$$\cos\theta \ (\mathbf{I} - \mathbf{a}\mathbf{a}^T)\mathbf{b} + \sin\theta \ [\mathbf{a}]_\times(\mathbf{I} - \mathbf{a}\mathbf{a}^T)\mathbf{b})$$

which you can verify is equal to

$$\cos\theta \ (\mathbf{I} - \mathbf{a}\mathbf{a}^T\mathbf{b}) + \sin\theta \ [\mathbf{a}]_\times\mathbf{b}.$$

The above sum is just the rotation of the component vector $(\mathbf{I}-\mathbf{a}\mathbf{a}^T\mathbf{b})$ within the plane perpendicular to $\mathbf{a}$. Thus, the rotation operator can be written as the sum of three matrices,

$$\mathbf{R} = \mathbf{a}\mathbf{a}^T + \cos\theta \ (\mathbf{I} - \mathbf{a}\mathbf{a}^T) + \sin\theta \ [\mathbf{a}]_\times.$$

This formula for a rotation is a slightly simplifed version from what I wrote on the slides. Another way of writing the same formula is *Rodriguez'es formula for rotation* (which you can look up on the web if you like....)

Homogeneous coordinates

We next consider an alternative way to represent translations and rotations which allows them both to be expressed as a matrix multiplication. This has certain computational and notational conveniences, and eventually leads us to a wider and interesting class of transformations.

Suppose we wish to translate all points (X, Y, Z) by adding some constant vector (t_x, t_y, t_z) to all coordinates. So how do we do it? The trick is to write out scene points (X, Y, Z) as points in $\Re^4$, and we do so by tacking on a fourth coordinate with value 1. We can then translate by performing a matrix multiplication:

$$\begin{bmatrix} X + t_x \\ Y + t_y \\ Z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

We can perform rotations using a 4×4 matrix as well. Rotations matrices go into the upper-left 3×3 corner of the 4×4 matrix:

$$\begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e. nothing interesting happens in the fourth row or column.

Notice that we can also scale the three coordinates:

$$\begin{bmatrix} \sigma_X X \\ \sigma_Y Y \\ \sigma_Z Z \\ 1 \end{bmatrix} = \begin{bmatrix} \sigma_X & 0 & 0 & 0 \\ 0 & \sigma_Y & 0 & 0 \\ 0 & 0 & \sigma_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

This scaling doesn't come up so much in 3D, but it does come up when we do 2D transformations.

We have represented a 3D point (X, Y, Z) as a point $(X, Y, Z, 1)$ in $\Re^4$. We now generalize this by allowing ourselves to represent (X, Y, Z) as any 4D vector of the form (wX, wY, wZ, w) as long as $w \neq 0$. Note that the set of points

$$\{ \ (wX, wY, wZ, w) \ : \ w \neq 0 \ \}$$

is a line in $\Re^4$ that passes through the origin and through the point $(X, Y, Z, 1)$ in $\Re^4$. That is, we are associating each point in $\Re^3$ with a line in $\Re^4$, in particular, a line that passes through the origin. This representation is called *homogeneous coordinates*.

Is this generalization consistent with the 4×4 rotation, translation, and scaling matrices which we introduced above? Yes, it is. For any $\mathbf{X} = (X, Y, Z, 1)$ and indeed for any 4×4 matrix $\mathbf{M}$, we have

$$w(\mathbf{MX}) = \mathbf{M}(w\mathbf{X}),$$

Thus, multiplying each component of the vector $\mathbf{X}$ by w and transforming by $\mathbf{M}$ yields the same vector as transforming $\mathbf{X}$ itself by $\mathbf{M}$ and then multiplying each component of $\mathbf{M}$ $\mathbf{X}$ by w. But multiplying each component of a 4D vector $\mathbf{X}$ by w doesn't change how that vector is transformed.

Points at infinity

We have considered points (wX, wY, wZ, w) under the condition that $w \neq 0$, and such a point is associated with $(X, Y, Z) \in \Re^3$. What happens if we let $w = 0$? Does this mean anything? Consider (X, Y, Z, ϵ) where $\epsilon > 0$, and so

$$(X, Y, Z, \epsilon) \equiv (\frac{X}{\epsilon}, \frac{Y}{\epsilon}, \frac{Z}{\epsilon}, 1).$$

If we let $\epsilon \to 0$, then the corresponding 3D point $(\frac{X}{\epsilon}, \frac{Y}{\epsilon}, \frac{Z}{\epsilon})$ goes to infinity, and stays along the line from the origin through the point $(X, Y, Z, 1)$. We thus identify $lim_{\epsilon \to 0} \ (X, Y, Z, \epsilon)$ with a 3D point at infinity, namely a point in direction (X, Y, Z).

What happens to a point at infinity when we perform a rotation, translation, or scaling? Since the bottom row of each of these 4×4 matrices is (0,0,0,1), it is easy to see that these transformations map points at infinity to points at infinity. In particular,

- a translation matrix does not affect a point at infinity; i.e. it behaves the same as the identity matrix;
- a rotation matrix maps a point at infinity in exactly the same way it maps a finite point, namely, $(X, Y, Z, 1)$ rotates to $(X', Y', Z', 1)$ if and only if $(X, Y, Z, 0)$ rotates to $(X', Y', Z', 0)$.
- a scale matrix maps a point at infinity in exactly the same way it maps a finite point, namely, $(X, Y, Z, 1)$ maps to $(\sigma_X X, \sigma_Y Y, \sigma_Z Z, 1)$ if and only if $(X, Y, Z, 0)$ scales to $(\sigma_X X, \sigma_Y Y, \sigma_Z Z, 0)$.

We sometimes interpret points at infinity as *direction vectors*, that is, they have a direction but no position. One must be careful in refering to them in this way, though, since vectors have a length whereas points at infinity $(X, Y, Z, 0)$ do not have a length.