Motion -II

- Structure from Motion
- Review of Motion
- Review of camera models
- Optical Flow

Structure-from-Motion

- Determining the 3-D structure of the world, and/or the motion of a camera using a sequence of images taken by a moving camera.
  - Equivalently, we can think of the world as moving and the camera as fixed.
- Like stereo, but the position of the camera isn't known (and it's more natural to use many images with little motion between them, not just two with a lot of motion).
  - We may or may not assume we know the parameters of the camera, such as its focal length.

Structure-from-Motion

- As with stereo, we can divide problem:
  - Correspondence.
  - Reconstruction.
- Again, we'll talk about reconstruction first.
  - Assume that each image contains some points,
  - we know which points match which

Motion

- Rigid and Non-Rigid Motion
- Rigid motion: all points on the object move together.
  - Their relative coordinates do not change

3D Geometry

- To model 3D scenes, we must specify:
  - Location
  - Displacement from arbitrary locations
  - Orientation

Linear Transformations

- A *linear transformation*:
  - Maps one vector to another
  - Preserves linear combinations
- Thus behavior of linear transformation is completely determined by what it does to a basis
- Turns out any linear transform can be represented by a *matrix*

Matrices

- By convention, matrix element $\mathbf{M}_{rc}$ is located at row *r* and column *c*:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} & \cdots & \mathbf{M}_{1n} \\ \mathbf{M}_{21} & \mathbf{M}_{22} & \cdots & \mathbf{M}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}_{m1} & \mathbf{M}_{m2} & \cdots & \mathbf{M}_{mn} \end{bmatrix}$$

- By convention, vectors are columns:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix}$$

Matrices

- Matrix-vector multiplication applies a linear transformation to a vector:

$$\mathbf{M} \bullet \mathbf{v} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} & \mathbf{M}_{13} \\ \mathbf{M}_{21} & \mathbf{M}_{22} & \mathbf{M}_{23} \\ \mathbf{M}_{31} & \mathbf{M}_{32} & \mathbf{M}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}$$

- Recall how to do matrix multiplication

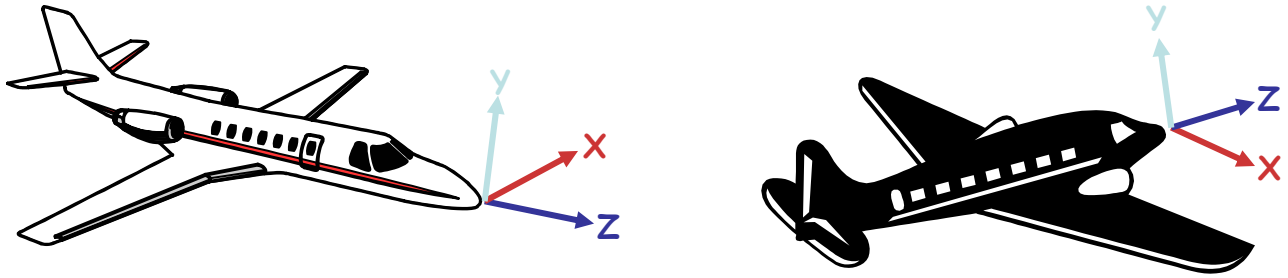
Matrix Transformations

- A *sequence* or *composition* of linear transformations corresponds to the product of the corresponding matrices
  - Note: the matrices to the *right* affect vector first
  - Note: order of matrices matters!
- Some (not all) matrices have an inverse:
- So we can reverse the transformation

$$\mathbf{M}^{-1}\left(\mathbf{M}\left(\mathbf{v}\right)\right) = \mathbf{v}$$

Transformations

- Modeling transforms
  - Size, place, scale, and rotate objects parts of the model w.r.t. each other
  - Object coordinates ⛵ world coordinates



Transformations

- All these transformations involve shifting coordinate systems (i.e., basis sets)
- That's what matrices do…
- Represent coordinates as vectors, transforms as matrices

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Multiply matrices = concatenate transforms!

Transformations

- *Homogeneous coordinates*: represent coordinates in 3 dimensions with a 4-vector
  - Denoted [x, y, z, w]
    - Note that typically w = 1 in model coordinates
  - To get 3-D coordinates, divide by w: [x', y', z'] = [x/w, y/w, z/w]
- Transformations are 4x4 matrices
- Why? To handle translation, rotation and projection uniformly

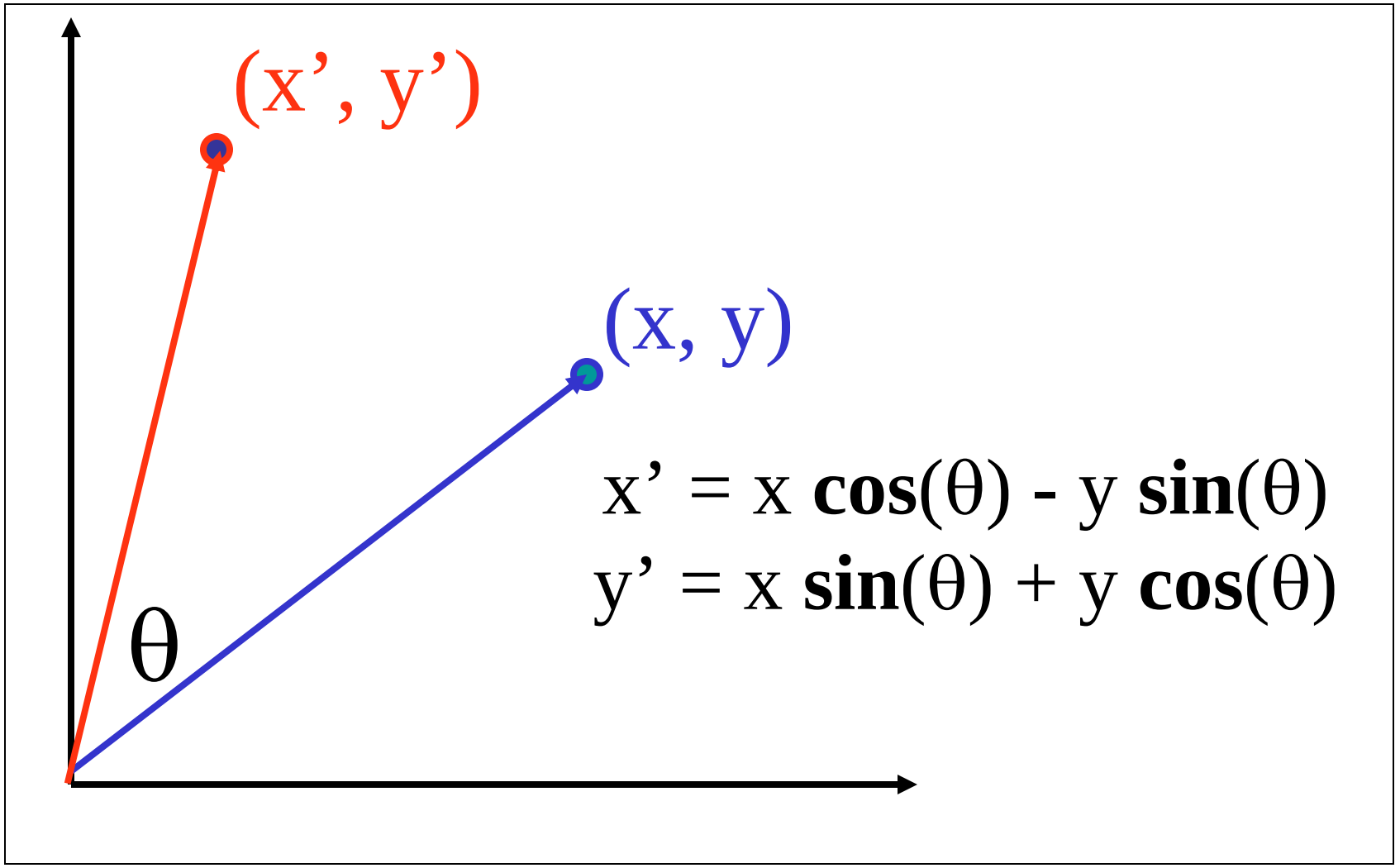
Translations

- For convenience we usually describe objects in relation to their own coordinate system
  - Solar system example
- We can *translate* or move points to a new position by adding offsets to their coordinates:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

  - Note that this translates all points uniformly

2-D Rotation



2-D Rotations

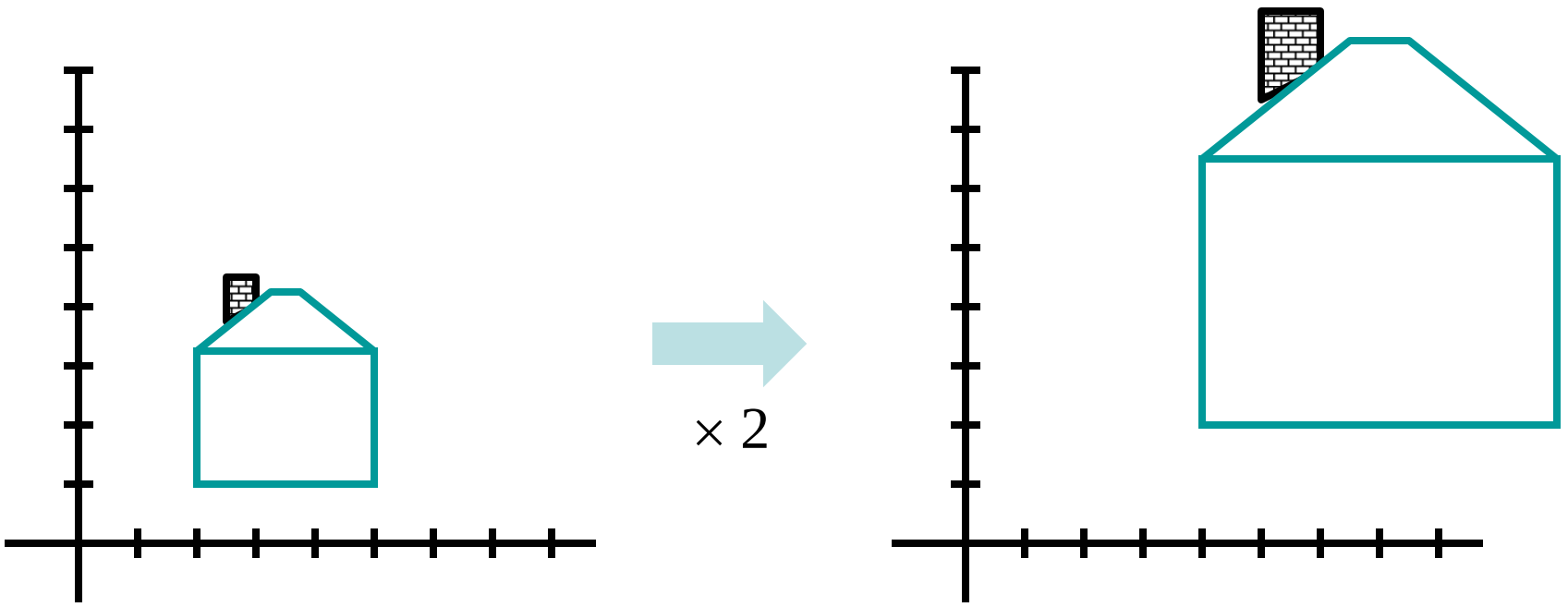
- Rotations in 2-D are easy:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- 3-D is more complicated
  - Need to specify an *axis of rotation*
  - Common pedagogy: express rotation about this axis as the composition of *canonical rotations*
    - Canonical rotations: rotation about X-axis, Y-axis, Z-axis

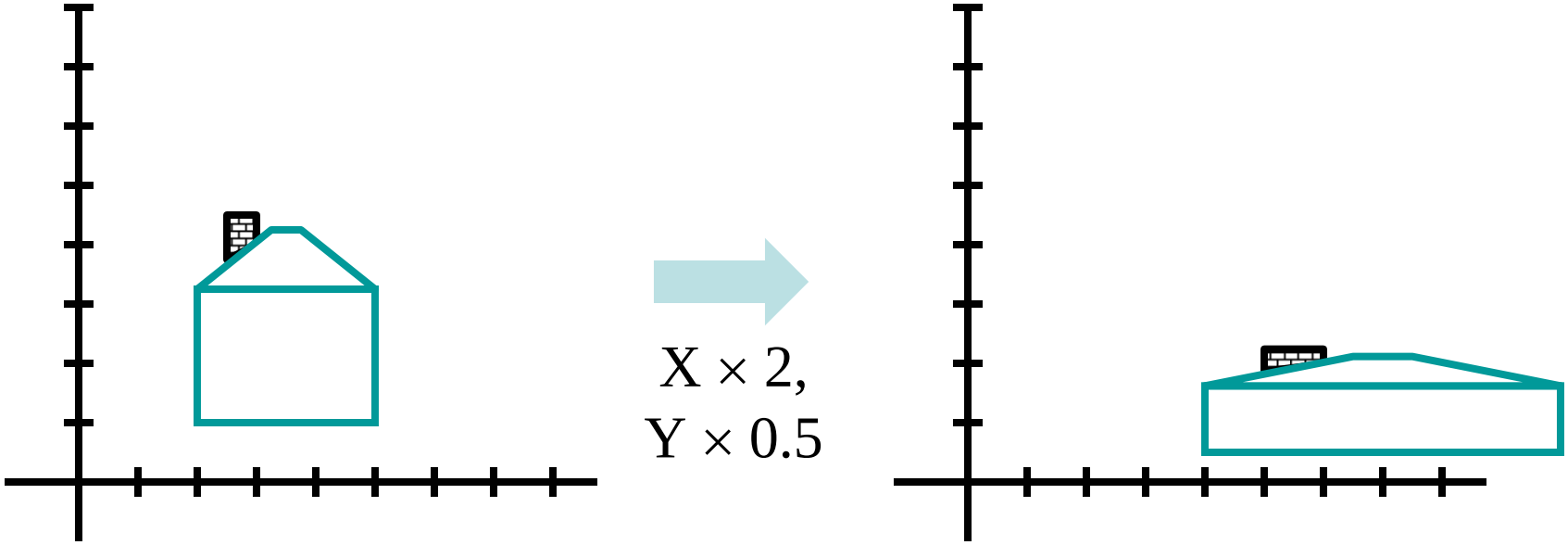
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



- *How can we represent this in matrix form?*

Scaling

- Scaling operation:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} ax \\ by \\ cz \end{bmatrix}$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

2-D Rotation

- This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- 3-D is more complicated
  - Need to specify an *axis of rotation*
  - Simple cases: rotation about X, Y, Z axes

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the Z-axis?*
  - Build it coordinate-by-coordinate

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the Y-axis?*
  - Build it coordinate-by-coordinate

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3-D Rotation

- *What does the 3-D rotation matrix look like for a rotation about the X-axis?*
  - Build it coordinate-by-coordinate

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Translation Matrices?

- We can composite scale matrices just as we did rotation matrices
- But how to represent translation as a matrix?
- Answer: with *homogeneous coordinates*

Homogeneous Coordinates

- *Homogeneous coordinates*: represent coordinates in 3 dimensions with a 4-vector

$$(x, y, z) = \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

(Note that typically w = 1 in object coordinates)

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make transformation operations *much* easier
- They allow us to string transformations together easily
- Our transformation matrices are now 4x4:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:

$$\mathbf{R}_y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Coordinates

- *How can we represent translation as a 4x4 matrix?*
- A: Using the rightmost column:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rigid-Body Transforms

- Goal: object coordinates ⛵ world coordinates
- Idea: use only transformations that preserve the shape of the object
  - *Rigid-body* or *Euclidean transforms*
  - Includes rotation, translation, and scale
- To reiterate: we will represent points as column vectors:

$$(x, y, z) = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Vectors and Matrices

- Vector algebra operations can be expressed in this matrix form
  - Dot product:

$$\mathbf{a} \bullet \mathbf{b} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \alpha$$

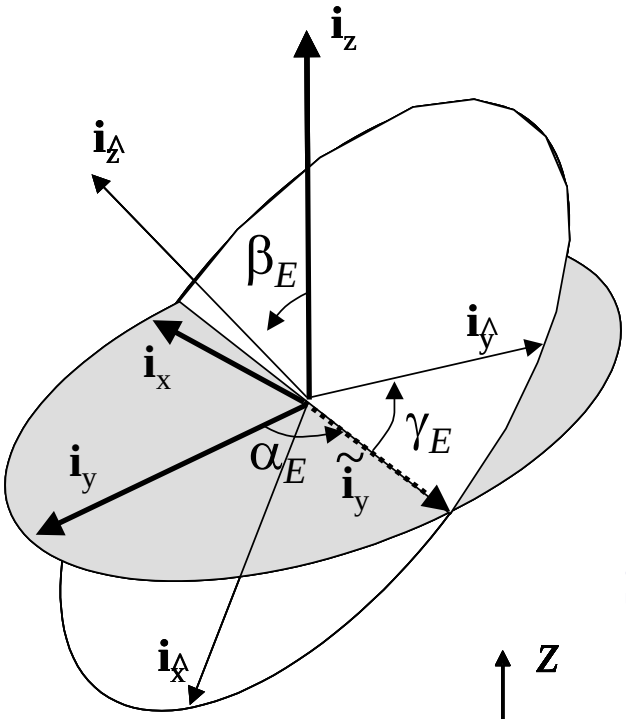
  - Cross product:
    - Note: use right-hand rule!

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \mathbf{c}$$

$$\mathbf{a} \bullet \mathbf{c} = \mathbf{0}$$

$$\mathbf{b} \bullet \mathbf{c} = \mathbf{0}$$

Rotation of coordinates



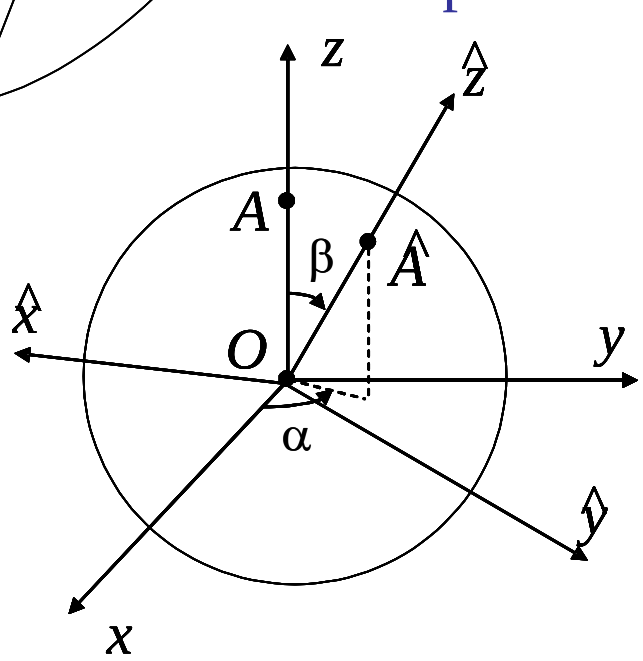
Rotation Matrix

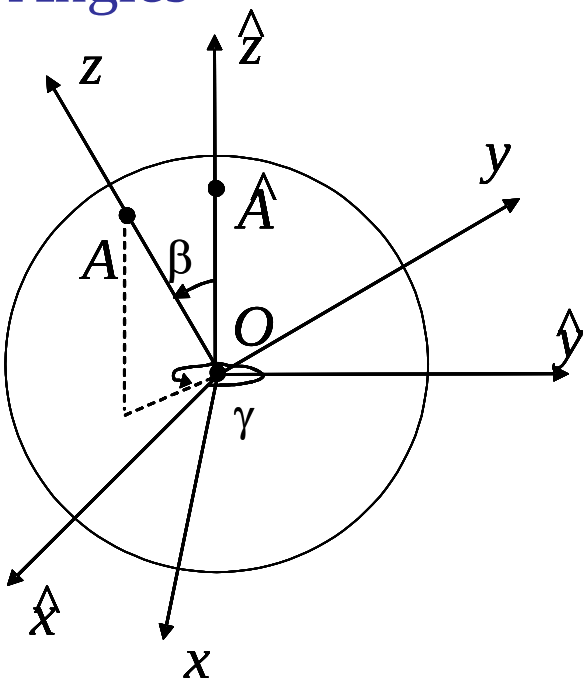
$$Q = \begin{bmatrix} \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{x}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{y}} \cdot \mathbf{i}_z \\ \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_x & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_y & \mathbf{i}_{\hat{z}} \cdot \mathbf{i}_z \end{bmatrix}$$

Euler Angles

$$\alpha_E = \pi - \alpha, \quad \beta_E = \beta, \quad \gamma_E = \gamma.$$

Spherical Polar Angles





The Camera

- These parameters are encapsulated in a *projection matrix*
  - Homogeneous coordinates of the world a 4 vector
  - Homogeneous coordinates of the image a 3 vector
  - So the projection operation is a 3×4 matrix

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{Bmatrix} X \\ Y \\ Z \\ 1 \end{Bmatrix}$$

Perspective

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

Weak Perspective

$$\begin{pmatrix} U \\ V \\ \end{pmatrix} = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ & & & \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

Reconstruction

- A lot harder than with stereo.
- Start with simpler case: scaled orthographic projection (weak perspective).
  - Recall, in this we remove the z coordinate and scale all x and y coordinates the same amount.

First: Represent motion

- We'll talk about a fixed camera, and moving object.
- Key point:

Some matrix

$$S = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{pmatrix}$$

Points

$$P = \begin{pmatrix} x_1 & x_2 & & & x_n \\ y_1 & y_2 & . & . & y_n \\ z_1 & z_2 & & & z_n \\ 1 & 1 & & & 1 \end{pmatrix}$$

The image (ignoring *z*)

$$I = \begin{pmatrix} X_1 & X_2 & . & . & X_n \\ Y_1 & Y_2 & & & Y_n \end{pmatrix}$$

Then: $I = SP$

- Objects represented as a moving set of points
- These are projected into the image
- Projection is represented with Matrix multiplication
    - take inner product between each row of S and each point.
    - First row of S produces X coordinates, while second row produces Y.
- Projection model is “scaled orthographic” here
    - from now on we ignore third row of S.
- Translation occurs with t_x and t_y; t_z does not matter
- Scaling encoded with a scale factor in S.
- The rest of S accounts for rotation

Examples:

- S = [s, 0, 0, 0;
  0, s, 0, 0];

  This is just projection, with scaling by s.

- S = [s, 0, 0, s*tx;
  0, s, 0, s*ty];

This is translation by (tx,ty,something), projection, and scaling.

Structure-from-Motion

- S encodes:
  - Projection: only two lines
  - Scaling, since S can have a scale factor.
  - Translation, by tx and ty.
  - Rotation:

$$I = SP$$

Putting it Together

Scale

3D Translation

$$s\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\end{pmatrix}\begin{pmatrix}r_{1,1} & r_{1,2} & r_{1,3} & 0\\ r_{2,1} & r_{2,2} & r_{2,3} & 0\\ r_{3,1} & r_{3,2} & r_{3,3} & 0\\ 0 & 0 & 0 & 1\end{pmatrix}P$$

Projection

3D Rotation

$$\equiv\begin{pmatrix}s_{1,1} & s_{1,2} & s_{1,3} & st_x\\ s_{2,1} & s_{2,2} & s_{2,3} & st_y\end{pmatrix}P$$

where

$$(s_{1,1}, s_{1,2}, s_{1,3}) \bullet (s_{2,1}, s_{2,2}, s_{2,3}) = 0$$

$$\left\|(s_{1,1}, s_{1,2}, s_{1,3})\right\| = \left\|(s_{2,1}, s_{2,2}, s_{2,3})\right\|$$

We can just write st_x as t_x and st_y as t_y.

Affine Structure from Motion

$$\begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & t_x \\ s_{2,1} & s_{2,2} & s_{2,3} & t_y \end{pmatrix} P$$

where

$$(s_{1,1}, s_{1,2}, s_{1,3}) \bullet (s_{2,1}, s_{2,2}, s_{2,3}) = 0$$

$$\left\| (s_{1,1}, s_{1,2}, s_{1,3}) \right\| = \left\| (s_{2,1}, s_{2,2}, s_{2,3}) \right\|$$

Affine Structure-from-Motion: Two Frames (1)

- Stack image data of corresponding points in an image measurement matrix
- Object coordinates are what we seek to find
- The projection equations for the two frames, including the relative motion between them is in the matrix S

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Affine Structure-from-Motion: Two Frames (2)

To make things easy, suppose:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Affine Structure-from-Motion: Two Frames (3)

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Looking at the first four points, we get:

$$\begin{pmatrix} u_1^1 & u_2^1 & u_3^1 & u_4^1 \\ v_1^1 & v_2^1 & v_3^1 & v_4^1 \\ u_1^2 & u_2^2 & u_3^2 & u_4^2 \\ v_1^2 & v_2^2 & v_3^2 & v_4^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Affine Structure-from-Motion: Two Frames (4)

$$\begin{pmatrix} u_1^1 & u_2^1 & u_3^1 & u_4^1 \\ v_1^1 & v_2^1 & v_3^1 & v_4^1 \\ u_1^2 & u_2^2 & u_3^2 & u_4^2 \\ v_1^2 & v_2^2 & v_3^2 & v_4^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

We can solve for motion by inverting matrix of points.

Or, explicitly, we see that first column on left (images of first point) give the translations. After solving for these, we can solve for the each column of the *s* components of the motion using the images of each point, in turn.

Affine Structure-from-Motion: Two Frames

(5)

$$\begin{pmatrix} u_k^1 \\ v_k^1 \\ u_k^2 \\ v_k^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} a_k \\ b_k \\ c_k \\ 1 \end{pmatrix}$$

Once we know the motion, we can use the images of another point to solve for the structure. We have four linear equations, with three unknowns.

Affine Structure-from-Motion: Two Frames

(6) Suppose we just know where the *k*'th point is in image 1.

$$\begin{pmatrix} u_k^1 \\ v_k^1 \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} a_k \\ b_k \\ c_k \\ 1 \end{pmatrix}$$

Then, we can use the first two equations to write a_k and b_k as linear in c_k. The final two equations lead to two linear equations in the missing values and c_k. If we eliminate c_k we get one linear equation in the missing values. This means the unknown point lies on a known line. That is, we recover the epipolar constraint. Furthermore, these lines are all parallel.

Affine Structure-from-Motion: Two Frames (7)

But, what if the first four points aren't so simple?

Then we define *A* so that:

$$A\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

This is always possible as long as the points aren't coplanar.

Affine Structure-from-Motion: Two Frames (8)

Then, given:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

We have:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1} A \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

And:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1} \begin{pmatrix} 0 & 1 & 0 & 0 & . & . & . & x_n \\ 0 & 0 & 1 & 0 & & & & y_n \\ 0 & 0 & 0 & 1 & & & & z_n \\ 1 & 1 & 1 & 1 & & & & 1 \end{pmatrix}$$

Affine Structure-from-Motion: Two Frames (9)

Given:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1} \begin{pmatrix} 0 & 1 & 0 & 0 & . & . & . & x_n \\ 0 & 0 & 1 & 0 & & & & y_n \\ 0 & 0 & 0 & 1 & & & & z_n \\ 1 & 1 & 1 & 1 & & & & 1 \end{pmatrix}$$

Then we just pretend that:

$$\begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1}$$

is our motion, and solve as before.

Affine Structure-from-Motion: Two Frames (10)

This means that we can never determine the exact 3D structure of the scene. We can only determine it up to some transformation, *A*. Since if a structure and motion explains the points:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

So does another image of the form:

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \left(\begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1} \right) \left(A \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix} \right)$$

Affine Structure-from-Motion: Two Frames

(11)

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \end{pmatrix} = \left(\begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \end{pmatrix} A^{-1} \right) \left(A \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix} \right)$$

Note that *A* has the form:

A corresponds to translation of the points, plus a linear transformation.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

For example, there is clearly a translational ambiguity in recovering the points. We can't tell the difference between two point sets that are identical up to a translation when we only see them after they undergo an unknown translation. Similarly, there's clearly a rotational ambiguity. The rest of the ambiguity is a stretching in an unknown direction.

Affine Structure-from-Motion: A lot of frames (1)

$$\begin{pmatrix} u_1^1 & u_2^1 & . & . & . & u_n^1 \\ v_1^1 & v_2^1 & & & & v_n^1 \\ u_1^2 & u_2^2 & & & & u_n^2 \\ v_1^2 & v_2^2 & & & & v_n^2 \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ u_1^m & u_2^m & & & & u_n^m \\ v_1^m & v_2^m & . & . & . & v_n^m \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 & t_x^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 & t_y^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 & t_x^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 & t_y^2 \\ & . & & \\ & . & & \\ & . & & \\ s_{1,1}^m & s_{1,2}^m & s_{1,3}^m & t_x^m \\ s_{2,1}^m & s_{2,2}^m & s_{2,3}^m & t_y^m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

I S P

First Step: Solve for Translation (1)

- This is trivial, because we can pick a simple origin.
  - World origin is arbitrary.
  - Example: We can assume first point is at origin.
    - Rotation then doesn’t effect that point.
    - All its motion is translation.
  - Better to pick center of mass as origin.
    - Average of all points.
    - This also averages all noise.

Specifically, we can never tell where the world points were to begin with. Adding one to every x coordinate in P and then subtracting 1 in every tx is undetectable.

So, wlog we can assume that sum(P(k,:)) = 0 for k from 1 to 3, ie., sum(x1 … xn) = 0, sum(y1…yn) = 0, sum(z1 … zn) = 0.

Rotation doesn't move the origin, which is now the center of mass. Neither does scaled orthographic projection. So, this only moves from translation.

Explicitly, we assume sum(p) = (0,0,0)^T. Then: sum(s*R(p)) = s*R(sum(p)) = s*R(0,0,0)^T = (0,0,0)^T. (^T means transpose).

More explicitly, suppose sum(p) = (0,0,0,n)^T. Then, sum(R*P) = R*(sum(P)) = R*(0,0,0,n)^T = (0,0,0,n)^T. Sum(T*R*P) = T*(0,0,0,n)^T = (ntx,nty,ntz,n)^T. (Or just look at the 2x4 projection matrix). If we subtract tx or ty from every row, then the residual is (s11,s12,s13;s21,s22,s23)*P. I = s part of matrix + t part of matrix.

$$s\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\end{pmatrix}\begin{pmatrix}r_{1,1} & r_{1,2} & r_{1,3} & 0\\ r_{2,1} & r_{2,2} & r_{2,3} & 0\\ r_{3,1} & r_{3,2} & r_{3,3} & 0\\ 0 & 0 & 0 & 1\end{pmatrix}P$$

Even more explicitly. Consider the first row of the image matrix I. Average together all the entries in this row. This gives us:

sum((s{1,1},s{1,2},s{1,3})*(x_i,y_i,z_i) + tx)/n

= (s{1,1},s{1,2},s{1,3})*sum(x_i,y_i,z_i)/n + tx

= (s{1,1},s{1,2},s{1,3})*(0,0,0) + tx = tx.

So we've solved for tx. If we subtract tx from every element in the first row of I, we remove the effects of translation.

First Step: Solve for Translation (2)

WLOG: $\sum_{i=1}^{n} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

$$\bar{u}^i = \sum_{k=1}^{n} u_k^i \Big/ n$$

$$\bar{v}^i = \sum_{k=1}^{n} v_k^i \Big/ n$$

$$\tilde{I} = \begin{pmatrix} u_1^1 - \bar{u}^1 & u_2^1 - \bar{u}^1 & \cdots & u_n^1 - \bar{u}^1 \\ v_1^1 - \bar{v}^1 & v_2^1 - \bar{v}^1 & & v_n^1 - \bar{v}^1 \\ u_1^2 - \bar{u}^2 & u_2^2 - \bar{u}^2 & & u_n^2 - \bar{u}^2 \\ v_1^2 - \bar{v}^2 & v_2^2 - \bar{v}^2 & & v_n^2 - \bar{v}^2 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ u_1^m - \bar{u}^m & u_2^m - \bar{u}^m & & u_n^m - \bar{u}^m \\ v_1^m - \bar{v}^m & v_2^m - \bar{v}^m & \cdots & v_n^m - \bar{v}^m \end{pmatrix}$$

First Step: Solve for Translation (3)

$$\begin{pmatrix} \tilde{u}_1^1 & \tilde{u}_2^1 & \cdots & \tilde{u}_n^1 \\ \tilde{v}_1^1 & \tilde{v}_2^1 & & \tilde{v}_n^1 \\ \tilde{u}_1^2 & \tilde{u}_2^2 & & \tilde{u}_n^2 \\ \tilde{v}_1^2 & \tilde{v}_2^2 & & \tilde{v}_n^2 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \tilde{u}_1^m & \tilde{u}_2^m & & \tilde{u}_n^m \\ \tilde{v}_1^m & \tilde{v}_2^m & \cdots & \tilde{v}_n^m \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ s_{1,1}^m & s_{1,2}^m & s_{1,3}^m \\ s_{2,1}^m & s_{2,2}^m & s_{2,3}^m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & & y_n \\ z_1 & z_2 & & z_n \end{pmatrix}$$

As if by magic, there's no translation.

Rank Theorem

$\tilde{I}$ has rank 3.

$$\begin{pmatrix} \tilde{u}_1^1 & \tilde{u}_2^1 & . & . & . & \tilde{u}_n^1 \\ \tilde{v}_1^1 & \tilde{v}_2^1 & & & & \tilde{v}_n^1 \\ \tilde{u}_1^2 & \tilde{u}_2^2 & & & & \tilde{u}_n^2 \\ \tilde{v}_1^2 & \tilde{v}_2^2 & & & & \tilde{v}_n^2 \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ \tilde{u}_1^m & \tilde{u}_2^m & & & & \tilde{u}_n^m \\ \tilde{v}_1^m & \tilde{v}_2^m & . & . & . & \tilde{v}_n^m \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 \\ & . & \\ & . & \\ & . & \\ s_{1,1}^m & s_{1,2}^m & s_{1,3}^m \\ s_{2,1}^m & s_{2,2}^m & s_{2,3}^m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \end{pmatrix}$$

$\tilde{I}$ S

This means there are 3 vectors such that every row of $\tilde{I}$ is a linear combination of these vectors. These vectors are the rows of P.

Solve for S

$$\tilde{I} = UDV$$

- SVD is made to do this.

D is diagonal with non-increasing values.

U and V have orthonormal rows.

Ignoring values that get set to 0, we have U(:,1:3) for S, and

D(1:3,1:3)*V(1:3,:) for P.

Linear Ambiguity (as before)

$\tilde{I}$ = U(:,1:3) * D(1:3,1:3) * V(1:3,:)

$\tilde{\tilde{I}}$ = (U(:,1:3) * A) * (inv(A) *D(1:3,1:3) * V(1:3,:))

Noise

$\tilde{I}$

- $\tilde{I}$ has full rank.
- Best solution is to estimate I that's as near to as possible, with estimate of I having rank 3.
- Our current method does this.

Weak Perspective Motion

$$\begin{pmatrix} \tilde{u}_1^1 & \tilde{u}_2^1 & . & . & . & \tilde{u}_n^1 \\ \tilde{v}_1^1 & \tilde{v}_2^1 & & & & \tilde{v}_n^1 \\ \tilde{u}_1^2 & \tilde{u}_2^2 & & & & \tilde{u}_n^2 \\ \tilde{v}_1^2 & \tilde{v}_2^2 & & & & \tilde{v}_n^2 \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ \tilde{u}_1^m & \tilde{u}_2^m & & & & \tilde{u}_n^m \\ \tilde{v}_1^m & \tilde{v}_2^m & . & . & . & \tilde{v}_n^m \end{pmatrix} = \begin{pmatrix} s_{1,1}^1 & s_{1,2}^1 & s_{1,3}^1 \\ s_{2,1}^1 & s_{2,2}^1 & s_{2,3}^1 \\ s_{1,1}^2 & s_{1,2}^2 & s_{1,3}^2 \\ s_{2,1}^2 & s_{2,2}^2 & s_{2,3}^2 \\ & . & \\ & . & \\ & . & \\ s_{1,1}^m & s_{1,2}^m & s_{1,3}^m \\ s_{2,1}^m & s_{2,2}^m & s_{2,3}^m \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \end{pmatrix}$$

$\tilde{I}$ S

$\tilde{I}$ =(U(:,1:3)*A)*(inv(A) *D(1:3,1:3)*V(1:3,:))

Row 2k and 2k+1 of S should be orthogonal. All rows should be unit vectors.

(Push all scale into P).

Choose A so (U(:,1:3) * A) satisfies these conditions.

Related problems we won’t cover

- Missing data.
- Points with different, known noise.
- Multiple moving objects.

Final Messages

- Structure-from-motion for points can be reduced to linear algebra.
- Epipolar constraint reemerges.
- SVD important.
- Rank Theorem says the images a scene produces aren’t complicated (also important for recognition).