3D Motion and Structure

COSC 470: Special Topic
Computer Vision | 3D Reconstruction
Steven Mills

Recovering Camera Motion

- The Essential matrix depends just on the rotation and translation between the cameras
- But the translation is only up to a scale
- We can recover these values by decomposing the matrix with the SVD
- Gives four solutions, but only one is sensible

Recovering Camera Motion

- We know the Essential matrix from camera calibration and the 8-point algorithm
- Take the first camera to be at the origin, pointing along the Z axis, so
  $$P_1 = K_1[I|0]$$
- We want to find *R* and *t* so that the second camera's projection matrix is
  $$P_2 = K_2[R|t]$$

Recovering Camera Motion

- It turns out (See H&Z 9.6 for details) that if

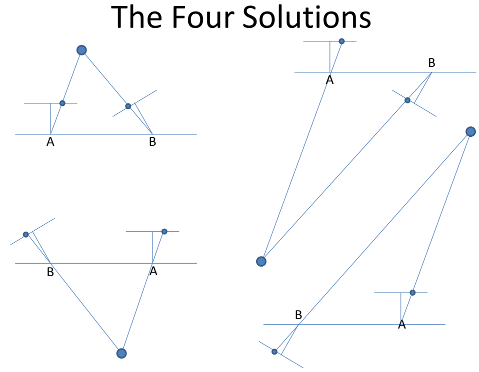
$$\mathrm{SVD}(E) = USV^T, W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Then *R* is either UWV^T or UW^TV^T, and t is some scaling of the last column of U, which we'll denote $\mathbf{u}_3$
- This leads to four possible solutions depending on our choice of *R* and whether we use $+\mathbf{u}_3$ or $-\mathbf{u}_3$

Which Solution is Correct?

- These four solutions can be checked to see if they make sense
- 3D points should be in front of both cameras
- Three of the solutions won't meet this criterion, so can be rejected
- The last solution is our answer, with $|t|$=1

The Four Solutions



Is a Point in Front of a Camera?

- Given a 3D point *X* which projects to a 2D image point *x*, is *X* in front of the camera with projection matrix *P*?
- X and x are in homogeneous co-ordinates, and so are defined up to a scale
- If the scale applied to x (the last element of *PX*) is positive, then *X* is in front of the camera
- But we need to know the 3D location, *X*…

Computing 3D Structure

- Given the camera motion and calibration we can recover the camera projection matrices
  $P_1 = K_1[I \mid 0] \qquad P_2 = K_2[R \mid t]$
- From these we can compute the 3D location of a point from the two images
- Each point corresponds to a ray in 3D space
- Errors mean these rays don't quite intersect

Computing 3D Structure

- Consider the projection into a camera
  $$\mathbf{x} = P\mathbf{X}$$
- Here **X** is the 3D point, **x** is its 2D image, and *P* is the projection matrix for the camera
- This is defined up to a scale, so as with computing Homographies we take cross products
  $$\mathbf{x} \times \mathbf{x} = \mathbf{x} \times P\mathbf{X} = 0$$

Computing 3D Structure

- This is easiest if we treat *P* as 3 separate rows
  $$(\mathbf{x} \times P)\mathbf{X} = 0$$
  $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \mathbf{X} = 0$$
  $$\begin{bmatrix} y\mathbf{p}_3^T - \mathbf{p}_2^T \\ \mathbf{p}_1^T - x\mathbf{p}_3^T \\ x\mathbf{p}_2^T - y\mathbf{p}_1^T \end{bmatrix} X = 0$$

Computing 3D Structure

- This gives us 3 equations, but only two are linearly independent, usually use the first two
  $$(x\mathbf{p}_3^T - \mathbf{p}_1^T)\mathbf{X} = 0$$
  $$(y\mathbf{p}_3^T - \mathbf{p}_2^T)\mathbf{X} = 0$$
- So the two views give us four equations, and we can solve for **X**
- Since this is a homogeneous system, we use the SVD method as before

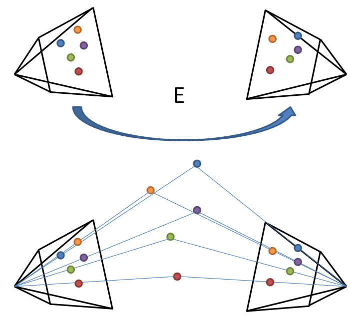
3D Reconstruction Cost Functions

- This method gives us a linear solution
- As with Homographies and Fundamental matrices, this isn't quite the error we want to minimise
- Other cost functions are available
- But the linear solution is easy to compute, and generalises well to more than 2 views

Stereo Reconstruction Summary

- Given two calibrated views of a scene
  - Match features
  - Compute F using 8-point algorithm and RANSAC
  - Compute $E = K_2^T F K_1$
  - Decompose E to get the rotation and translation
  - Projection matrices: $P_1=K_1[\ I \mid 0\ ]$, $P_2=K_2[\ R \mid t\ ]$
  - Recover 3D location of matching points

Stereo reconstruction Summary



Multi-View Reconstruction

- Suppose we have more than two cameras
- We could estimate pairwise motion, but each is defined up to a scale, and scales will change
- Instead we can do pose from 2D-3D matches
  - Each match gives us 2 equations (x and y in image)
  - We have 6 unknowns (rotation, translation)
  - We need at least 3 point matches

Perspective n Point (PnP) Pose

- Can make linear solutions using similar techniques to what we've seen
- But these are linear in the elements of *R* and *t*
- *R* has 9 elements by only 3 degrees of freedom (it is a Rotation matrix)
- So these solutions are very bad
- Other more complex solutions exist, see for example original RANSAC paper

Multi-View Reconstruction

- PnP gives us the location of the new camera
- We can then refine our 3D structure estimate
  - We can update any existing points that the new camera can see
  - We can often recover new points, seen in the new camera and exactly one other camera
- All of this comes from point correspondences between pairs of cameras

Multi-View Reconstruction