CS 280 Computer Vision
Spring 2018 Efros, Malik, Yu Lecture 16

1 Lecture

Epipolar Geometry

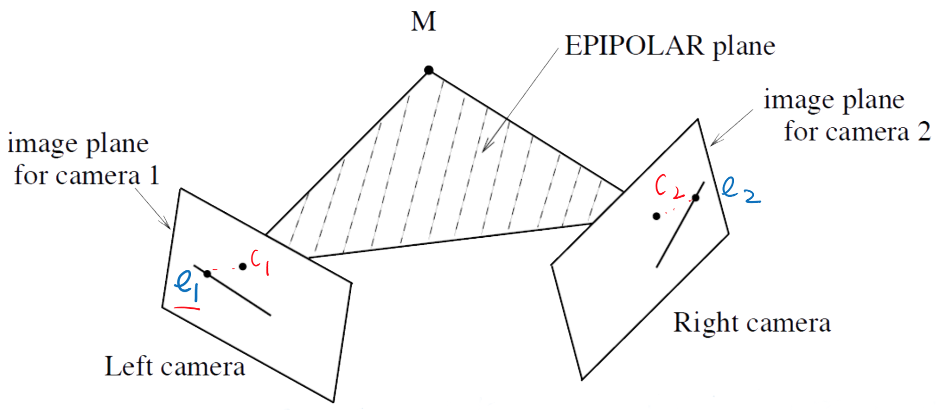
Continuing from last time, we consider a setup in which two pinhole cameras are related to each other by an arbitrary rotation R and translation t. We want to work out how we can recover 3D depth in this situation.

A point M in the world forms a plane with the centers of projection C_1, C_2 of the left and right cameras. (C_1 is the pinhole of the left camera; C_2 is the pinhole of the right camera.) This plane is called the **epipolar plane** which contains M. There is an epipolar plane for every point M. Note that every epipolar plane has one line in common: the line which joins the centers of projection of the two cameras.

C_1 is in front of the image plane for the left camera. The distance between the two is f. Likewise, C_2 is in front of the image plane for the right camera. We call M_1 and M_2 the projections of the point M in each camera. Therefore, the points (M, C_1, M_1) form a straight line.

The **epipolar line** for each camera is the intersection of the epipolar plane (M, C_1, C_2) with the image plane. If we extend the line (C_1, C_2) in both directions, then at some point it will intersect the left image plane. This intersection is the left **epipole** e_1. Similarly, we have a right epipole e_2. (e_1 is the image of the second camera's pinhole in the first camera.)

Since (C_1, C_2) is in every epipolar plane, all epipolar lines in the first camera must go through e_1.



source: slides from Jitendra Malik's CS 280 lecture

The general case: we have n points (X_i, Y_i, Z_i) in the world, which means we have n projections (x_i, y_i) in camera 1 and n projections (x'_i, y'_i) in camera 2. We can measure all of the projections, and are interested in reconstructing the world coordinates. This is the **structure from motion** problem.

At the same time, we need to figure out the rotation and translation between the two cameras.

Eight-Point Algorithm

- First we must find n corresponding points in the two views, i.e. projections of identical points in the scene, e.g. M_1 and M_2 in the previous section. There are other algorithms to determine these points, but for now we will take them as givens.
- If we have at least eight point correspondences, we can find the E (essential) matrix, which encodes the translation and rotation as $E = \hat{T}R$, where $\hat{T}$ is the skew-symmetric matrix corresponding to the translation vector t.
- After we have the E matrix, we can factorize it to get R and t. At this point, the geometry is known and we can recover depth by triangulation (just by extending rays out and seeing where they intersect).
- Once we have an approximate reconstruction, we can refine it via the process of **bundle adjustment** (which is really just global least-squares).

Finally, we can use the algorithm as a whole in order to build 3D models of objects in the world given multiple views.

Projective Transformations

Recall: projective transformations are a general family of transformations which includes affine transforms and perspective projections. They are *linear transforms* if we use homogeneous coordinates. In homogeneous coordinates, scalar nonzero multiples of a point, i.e.

$$\begin{bmatrix}\lambda x_1 & \ldots & \lambda x_n\end{bmatrix}^T \text{ as compared to } \begin{bmatrix}x_1 & \ldots & x_n\end{bmatrix}^T$$

are regarded as the same point in P^{n-1} projective space. Note that these are lines through the origin in n-dimensional space. To go from homogeneous coordinates to ordinary Euclidean coordinates, we just divide out by the final coordinate. To go the other way, we just include a 1 as the homogeneous coordinate.

An affine transformation is represented as

$$\begin{bmatrix}X' \\ Y' \\ W'\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

A perspective projection is represented as

$$\begin{bmatrix}1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0\end{bmatrix}\begin{bmatrix}X \\ Y \\ Z \\ 1\end{bmatrix} = \begin{bmatrix}X \\ Y \\ Z/f\end{bmatrix} = \begin{bmatrix}fX/Z \\ fY/Z \\ 1\end{bmatrix}$$

A general projective transformation (from P^2 to P^2) is represented as

$$\begin{bmatrix}x_1' \\ x_2' \\ x_3'\end{bmatrix} = \begin{bmatrix}p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33}\end{bmatrix}\begin{bmatrix}x_1 \\ x_2 \\ x_3\end{bmatrix}$$

If we were to scale the matrix by a constant nonzero factor, every entry in the output would be multiplied by the same factor. However, this would have no effect – it'd still be the same canonical Euclidean point. Therefore, the matrix really only has 8 independent numbers (though it pretends to have 9); scalar multiples don't count, and we can treat one of the entries of the matrix as being a scalar multiple.

Essential Matrix Constraint

Let us return to epipolar geometry. Call $x_1 \in \mathbb{R}^3$ and $x_2 \in \mathbb{R}^3$ the homogeneous coordinates of M_1 and M_2. Then

$$x_2^T \hat{T} R x_1 = 0 \implies x_2^T E x_1 = 0$$

x_1 and x_2 are measurable (in each camera's coordinates), as they're just the projection of a point onto each camera's image plane. We will have one linear equation $x_2^T E x_1 = 0$ for every pair of corresponding points, and we can use these to determine the nine unknowns of E. We'll need at least eight point correspondences, since there are only eight independent entries. (This is the reason for the name "eight-point algorithm.") The more points the better – more points will help counteract noise.

Essential Matrix Factorization

After we have solved for E, how do we extract R and t? We have

$$E = \hat{T}R$$

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix} R$$

This requires an algorithm too, but it's just an exercise in linear algebra – not worth covering in lecture. Szeliski's book has a nice exposition of it.

Summary

The eight-point algorithm allows us to recover the essential matrix and thereby 3D structure from two views with an unknown relative orientation (R, t). If we have more than two cameras, we can combine information from all of them in a global coordinate system and use the bundle adjustment routine, which is a kind of least-squares minimization of the reprojection error.

- *The idea there*: use the eight-point algorithm to estimate the essential matrix, giving us an initial guess for the positions of the cameras and the points in 3D. Given the estimated 3D position of each point, predict 2D image plane positions for any camera in which it is visible. Then adjust the positions of the cameras and the points in 3D such that they best predict what we see in the images (by minimizing the squared error between the predicted and actual positions, over all cameras and all points).

Note: there is also the **fundamental matrix** F, which is similar to the essential matrix but uses pixel coordinates instead of camera coordinates. With the essential matrix, we assume the cameras are calibrated according to known intrinsic matrices K_1 and K_2. The fundamental matrix is related to the essential matrix as $F = (K_2^T)^{-1} E K_1^{-1}$. If we don't know the intrinsics, we can estimate the fundamental matrix instead.

SLAM is structure from motion in the setting of robotics.

Solving for Stereo Correspondence

Say we have two images of the same scene from different views, along with a point in the first image. What is the corresponding point in the second image? It can only be on the corresponding epipolar line, so we want to search over those pixels and find the best match. Note: we need R and t (and by extension E) to calculate the epipolar line, and we need some correspondences to find E. Hence we do the matching in stages. We get a few good points, which we match and use to estimate R and t, and then we try to find the correspondences for all of the remaining points.

We will consider different possible geometries. The first involves cameras with parallel optical axes and image planes parallel to the baseline. If the camera centers are at the same height and the focal lengths are the same, then the epipolar lines will fall along the horizontal scan lines of the images.

Even if the cameras are actually rotated with respect to each other, we can re-project the image planes onto a common plane parallel to the line between the optical centers. This process is called **rectification**, and of course requires knowledge of R. After this transformation, the pixel motion for correspondences will be horizontal.