CS 280 Computer Vision
Spring 2018 Efros, Malik, Yu Lecture 19

1 Lecture

Correspondence

Application to SfM

As a review, in the **structure from motion** (SfM) problem we are given many images and want to (a) figure out where they were taken from, and (b) build a 3D model of the scene. The idea is that by moving around the scene, the cameras give us the information we need to recreate 3D structure.

In a simplified setup, the input is a collection of images with pairwise points $(u_{i,j}, v_{i,j})$ [corresponding to scene point $p_{i,j}$] already provided. The output is then

- **structure:** a 3D location x_i for each point p_i
- **motion:** camera parameters R_j, t_j (and possibly a calibration matrix K_j)

There are i features (i.e. scene points) and j frames. Our goal is to minimize **reprojection error**, meaning "given our newfound 3D point x and our camera parameters, we should be able to project x back into each image and get the original point p."

Camera calibration and triangulation: imagine we had the 3D points and their matches as 2D points p in an image. The *camera calibration problem* is to recover the camera parameters from these known 3D points and/or calibration objects. *We know the structure; we want to find the parameters.*

There are two types of parameters:

- **internal (intrinsic):** the parameters of the device itself: focal length, optical center, aspect ratio.... These are intrinsic to the camera.
- **external (extrinsic):** the parameters of the camera with respect to the world, i.e. the pose (position and orientation). These are a function of where/how the device is placed.

Note: it only makes sense to talk about extrinsic parameters if we have multiple images.

The simplest way to perform camera calibration is to solve for the projection matrix which converts 3D points to (u, v) image plane coordinates. We can place a known object in the scene, identify correspondences between the image and the scene, and then compute the mapping from the scene to the image.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \approx \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The problem is that the intrinsic & extrinsic parameters will then be mushed up into the projection matrix, and it won't be clear how to separate them. So the other way to do it is to *explicitly* solve for the parameters.

Among other things, we have the extrinsic translation T of the optical center from the origin in world coordinates, the extrinsic rotation R of the image plane, the intrinsic focal length f, the intrinsic principal point (x'_c, y'_c), and the intrinsic pixel size (s_x, s_y). We can decompose the projection matrix into a product

of matrices that make these values explicit. If the original projection equation is

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \Pi \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

then

$$\begin{aligned} \Pi &= \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{3\times 3} & 0_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} I_{3\times 3} & T_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix} \\ &= \text{intrinsics} \cdot \text{projection} \cdot \text{rotation} \cdot \text{translation} \end{aligned}$$

We can solve this system using nonlinear optimization.

What if we knew the camera parameters (both intrinsic and extrinsic) and a single point's correspondences between multiple images, and we wanted to compute the 3D location of that point? This is just a **triangulation** problem.

Back to SfM: the trick is that we want to do both of these things (calibration and triangulation) at the same time. *This is what SfM does.* Note: in SfM we're mainly thinking about calibrating extrinsics, because we assume the intrinsics already known (we can just read them off of EXIF files or something). But the bottom line is that we *calibrate the extrinsics* and *get the structure* – the 3D positions – at the same time.

Before performing SfM, we must (1) *detect features* (e.g. using SIFT) and (2) *match features* across the entire collection of images (e.g. using NN + RANSAC, where we're estimating the fundamental matrix between each pair of images and also tracking the matches across many images).

Then we can actually do SfM: solve for the geometry of the camera (a 3D position c_i, a 3D orientation R_i, and the focal length) and the 3D points (a collection of X_j's). To do so, we minimize the sum of squared reprojection errors:

$$g(X, R, T) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| P(X_i, R_j, t_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

where X is a 3D point. i corresponds to a location; j corresponds to a camera. $P(X_i, R_j, t_j)$ is the projection of point i into the frame of camera j. $(u_{i,j}, v_{i,j})$ is the original location of point i in the frame of camera j. *Note: some points aren't visible in some images, and we don't want to penalize things that aren't even seen. Accordingly, w_{ij} is an indicator variable expressing whether point i is visible in image j.*

Minimizing this function is called **bundle adjustment**.

Incremental SfM: initialization is important. It's difficult to initialize all of the cameras at once. Luckily, SfM with two cameras is relatively straightforward, and it's easy to add new cameras to an existing model. We can do SfM with just a few cameras at first, then add cameras one-by-one and recompute.

Application to Instance Retrieval

Recall the *multi-view matching* problem: we have two images and we find the correspondences between them (which in turn gives us structure, motion, etc.). We can also frame this as a *search for instances*. Maybe we have one image, and we want to find similar images in a huge dataset. *This is also a type of matching.* A good image will have lots of correspondences, and a bad one will have very few.

In this way, correspondences are important for **image (instance) retrieval**.