CS 280 Computer Vision
Spring 2018 Efros, Malik, Yu Lecture 18

1 Lecture

Correspondence and Interest Points

Last time we looked at stereo correspondence; let's now examine a more general type of correspondence, i.e. matching any kind of feature across different visual representations of a scene. *Everything* depends on this: stereopsis, optical flow, structure from motion, recognition...

Keypoint Matching

What if we want to find correspondences across very different views? Maybe we want to stitch together panoramic photos, which we know to be related by a homography. In this case, we want to find correspondences so we can in turn compute our homography.

One way to do this is through **keypoint matching**. We separate our problem into three distinct steps: (1) *detecting the points that would provide good matches* (interest points; points that would be easy to find in the other image; "corners") and then (2) *extracting feature descriptors for the area around each interest point.* These descriptors can be used for matching i.e. (3) *determining correspondences between descriptors in two views.*

1. **Corner detection.** At every point in the image, we want to ask "is it a corner or not?" A corner is a region that is easy to localize, i.e. one that looks different from its neighbors. If we wiggle the window around, its contents should change significantly.

   If we have a window W with weighting function $w(x, y)$ (e.g. a 2D Gaussian), along with an intensity function I, then we can define the change in the window's appearance for the shift u, v according to an SSD metric as

   $$E(u, v) = \sum_{x,y \in W} w(x, y)[I(x + u, y + v) - I(x, y)]^2$$

   However, this is way too slow to compute naively. So instead we want an approximation. Since we're only looking at a local region, we can just take the gradient (I_x, I_y) at each point. The justification for this is based on the Taylor series expansion of a function at a point; we only need a couple of terms if we only care about things being around the center.

   The local quadratic approximation of the error surface $E(u, v)$ in the neighborhood of $(0, 0)$ is given by the second-order Taylor expansion.

   $$\begin{aligned}
   E(u, v) &\approx E(0, 0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0, 0) \\ E_v(0, 0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
   &= \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
   &= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y \in W} w(x, y) I_x^2(x, y) & \sum_{x,y \in W} w(x, y) I_x(x, y) I_y(x, y) \\ \sum_{x,y \in W} w(x, y) I_x(x, y) I_y(x, y) & \sum_{x,y \in W} w(x, y) I_y^2(x, y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
   &= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}
   \end{aligned}$$

where

$$M = \sum_{x,y \in W} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

i.e. a 2×2 *second moment matrix* of image derivatives summed over some neighborhood.

In summary, we approximate distinctiveness with local autocorrelation, then approximate local autocorrelation with a quadratic form (the second moment matrix). We can quantify distinctiveness ("corner"-ness) as a function of the two eigenvalues λ_1, λ_2 of the second moment matrix. (We want both eigenvalues to be large; otherwise we probably just have an edge or a low-texture region.) Note: we don't actually need to compute the eigenvalues, as the determinant and the trace are good enough.

$$\begin{aligned} \text{corner response for window } W &= \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \\ &= \det(M) - \alpha \cdot \text{tr}(M)^2 \end{aligned}$$

This is called **Harris corner detection**. In order to filter for only the best (and spread-out) corners, we should also perform adaptive non-maximal suppression.

2. **Feature descriptors.** We need to provide context for each corner, in order to make sure we match each specific one. To make our descriptors rotationally invariant, we can rotate each descriptor to the direction of the gradient of the corner. We should also subsample and normalize our descriptors in order to make them more robust to changes.

   Note: we have characterized a MOPS descriptor vector, which is essentially a simpler version of SIFT.

3. **Matching.** So far, it's all been one image at a time. But now we will bring in pairs and actually find the correspondences. This might not be easy, because a lot of the interest points in one image might not even exist in the other image.

   We should therefore involve a threshold: for every patch, find the closest patch to it *above some threshold*. We can find all patches that match within some threshold. But how do we pick this threshold? One idea is to use symmetry: a pair of patches should be a good match in both directions.

   The trick that works amazingly well is the "Russian greedy grandma algorithm." With this kind of matching, a point has either zero or one correct match in the other image. Now, the relevant analogy: if a woman can't choose between two suitors, maybe she shouldn't be marrying either. *If there are two potential matches that we can't decide between, maybe we shouldn't choose either.* If there's only one match for us, we want the first guy – the best match – to be *way* better than the second-best match.

   So our threshold should be based on the ratio between the first and second nearest neighbor.

   Another problem: if we have even one incorrect match, our homography is hosed. We need to compute a homography that ignores all *false positives.* To do so, we can use RANSAC (random sample consensus). We determine the transformation associated with a random set of correspondences, and count how many correspondences (here **inliers**) agree with the transformation. Then we repeat. At the end, we pick the transformation with the greatest number of inliers.