# What did you learn in CSE 152?

Owen Jow

December 21, 2018

## 1 Low-Level Vision

After a multi-week review of linear algebra, you were given an introduction to *filtering*, one of the most basic computer vision operations. Filtering refers to a transformation on each pixel intensity as a function of neighboring pixel intensities. In the linear case, it can be seen as sliding a kernel of weights over the image and taking dot products to determine each pixel's new value. This is known as a *correlation* or *convolution* operation, the latter being when the kernel is flipped.

Note that I have been speaking in regard to the spatial domain so far. We can also view images as denizens of the frequency domain, which arises from the Fourier notion that we can represent any function in terms of sines and cosines. It turns out that an image is just a function (from $x, y$ to intensity), so it can be represented in terms of sines and cosines too. Specifically, a 2D image can be represented through a *Fourier transform* as a weighted sum of 2D sines and cosines of different frequencies.[1] Those weights and frequencies we can plot in a *frequency domain image*, which is a function from $x$-frequency, $y$-frequency to the weight of the corresponding sinusoid.

We can directly tie frequency manipulation back to linear filtering using the *convolution theorem*, which tells us that convolution in the spatial domain is equivalent to point-wise multiplication in the frequency domain. Hence we can filter by converting everything into the frequency domain via FFT,[2] multiplying, and then converting back via inverse FFT.

Next, you learned about *feature matching*, which here means "finding points in 2+ images which represent the same point in 3D." This is a very important topic in computer vision because the ability to robustly determine correspondences in different views allows you to solve a large number of problems, including but not at all limited to optical flow and SfM.[3] Commonly, feature matching is broken into three steps: keypoint localization, local descriptor extraction, and descriptor matching.

There are many types of interest points; in this class, we covered Harris corners. A *corner* in an image is a point around which there are strong derivatives in two directions. Accordingly, it is distinctive and hopefully localizable in different views of a scene (because it's not going to look the same as, e.g., all other points on an edge, or all other points in a flat region).

To find corners, threshold based on corner response at each location. A *Harris* corner is one detected according to the Harris corner response function $\lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$, where $\lambda_1$ and $\lambda_2$ are the eigenvalues of the second moment matrix

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

[1]For an image, *frequency* is cycles in intensity per pixel – as opposed to per second. Look up "spatial frequency!"

[2]*Fast Fourier transform*: an $n \log n$ algorithm to compute the discrete Fourier transform.

[3]*Structure from motion*: 3D structure and camera pose estimation from multiple views of a scene.

The idea is that we want the eigenvalues of that matrix to be similar and large, which signifies that the gradients in the window around the point are strong in two different directions.

Once we have keypoints, we ought to extract descriptors based on their surrounding windows, since matching just by comparing single pixel intensities is prone to failure. A popular choice of descriptor is SIFT, which in a simplified sense is a normalized histogram of oriented gradients. It is invariant to a lot of window transformations, including orientation and brightness changes.

Finally, we can match the descriptors based on some metric, e.g. L2 distance. When identifying matches, it helps to make sure the best match is much better than the second-best match – so we'll accept matches based on the ratio of *best match distance* to *second-best match distance.*

## 2 Motion

The Lucas-Kanade optical flow method solves for the $(u, v)$ motion of a window which best adheres to brightness constancy across two frames. Note that our standard brightness constancy equation $I(x, y, t-1) = I(x+u, y+v, t)$ assumes pure translation within a neighborhood; we can use other motion models if we so desire.

Based on three assumptions (brightness constancy, small motion, and spatial coherence), the system of equations we end up having to solve for $u$ and $v$ is

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ \vdots & \vdots \\ I_x(\mathbf{p}_n) & I_y(\mathbf{p}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ \vdots \\ I_t(\mathbf{p}_n) \end{bmatrix}$$

(The linearized brightness constancy equation for a single pixel is $I_x u + I_y v = -I_t$.)

## Acknowledgments

This document was directly inspired by Yining Liu's "what have you learned so far?" writeups.