Anthony Poerio (adp59@pitt.edu)

CS1674: Homework 4 - Written

Due: 9/26/2016, 11:59pm

QUESTIONS

1) **How are feature distinctiveness and feature repeatability at odds with each other? Why is each of those properties desirable (in moderation)?**

   Feature **distinctiveness** means that → If a set of pixels within an image has some feature, then we are able to identify it with a high degree of certainty—without mistaking the *actual* feature for another one.

   Feature **repeatability** means that → We are able to correctly identify *the same* feature across many images, despite variations.

   Feature repeatability and distinctiveness are at odds because: we must necessarily sacrifice distinctiveness in order to generalize a desired feature, allowing us to identify it in a wide variety of circumstances. Put another way, if a feature is very highly distinctive, it cannot be very highly repeatable—otherwise, far too many images would test positive for containing the feature, and we would have false positives.

   In moderation, both qualities are desirable because distinctiveness allows us to find features with high certainty, and repeatability allows us to find features within a wide variety of imagery.

2) **Describe the process of extracting the locations of corners from images, starting with the use of image gradients.**

   In order to extract the locations of corners in an image, we start by looking through a small window within the image.

   To find image gradients, we need to compute the change of intensity in the pixels we move across, as we shift our small window in any given direction. (Practically, this can be achieved using a filter.)

Then, if shifting our window in *all directions* from some given point results in calculating a change in intensity → we know we have found a corner at this point.

More abstractly, this can be done using an algorithm called the Harris Corner detector, which is a sophisticated way of determining whether we have image gradients in all directions, indicating that we are at a corner within the image.

3) Why do we say that the Harris corner detector is not scale-invariant?

The Harris Corner Detector is not scale-invariant because it will NOT necessarily detect the **same corners** if the algorithm is run on two versions of the same image, shown at different scales.

The easiest intuitive way to understand what this means is this. A corner can be viewed as the intersection of two lines. But if we get close enough to one of the lines, and we are asked to evaluate only that part of the image, we'll know longer be able to see the intersection—and from this perspective, we're no longer looking at a corner at all.

Therefore, the corners that the Harris Corner Detector algorithm identifies ***depends upon*** the scale of the image. It is not invariant to scale. If scale were a parameter to the algorithm, it would change the outputs.

4) How can we compute a histogram of gradient orientations, when generating a SIFT descriptor?

The basic idea is that we take a 16x16 square window around the feature we are interested in and compute the gradient orientation (angle) for each pixel, over the range from $[0, 2\pi]$.

This is done by dividing the 16x16 window into a 4x5 grid of cells, then quantizing the gradient orientations by 'snapping' each gradient to the nearest of 8 angles equally dividing the range, $[0,2\pi]$. (I.e. – $0, \pi/4, \pi/2, \ldots 2$)

The actual computation finds the gradient intensity in both the X and Y directions, and it is given by this formula:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

5) **What is the effect of gradient magnitude on the histogram of gradient orientations, in the computation of the SIFT descriptor?**

With our gradient orientations in hand, we can create a histogram over edge orientations *weighted by the magnitude*. (In the example we've seen in our slide deck, we break this histograms value groups into **8 buckets**.) That is, aren't just interested in a *count* of how many pixels fell into a given bucket. We also want to know *how strong* the magnitude of each vector is.

For example, a vector oriented at the angle $\pi/2$ with magnitude 1 would only count for 1, but a vector with magnitude 10 would count for 10. This is different than how histograms typically work, as magnitude is usually not taken into account. But for the SIFT descriptor, magnitude is important, so ***we must take both angle and magnitude into consideration*** as we build our histogram.

6) **Why do we have to normalize the SIFT histogram a second time, after clipping all values to be at most 0.2?**

After we clip all the values to be at most 0.2, we lose some of the information we had about magnitude—especially those vectors whose magnitudes were very large. Because of this, we must normalize the data a second time, to account for what is, in some ways, a different data set.

Previously, the vectors were normalized taking into account the 'true' range of magnitudes. So they won't be 'normalized' with respect to a maximum value of 0.2. In this regard the vectors aren't 'normalized' at all, until we do this operation a second time, with respect our new maximum.