CSci 4270 and 6270
Computational Vision,
Spring Semester, 2021
Lecture 10 Exercise — Interpolation
Due: Wednesday March 3, 2021 at 11:59pm EST

As we discussed in class today (Monday), interpolation is a key step in accurate computation of location and orientation from discrete quantities, even if they are estimated from quite a few pixels. In this lecture exercise, I'd like you to work through the details of the SIFT interpolations — the interpolation for voting into an angle histogram, the voting in the horizontal direction and the voting into the vertical direction. We are focusing on the interpolation step **after** the step of mapping a pixel location and gradient vector into the descriptor coordinate system established at a keypoint. We let u', v', I'_u, and I'_v be the mapped pixel location and derivatives. (See more explanation below.) We will not include the product the of the gradient magnitude and the Gaussian weight so as to focus on the weights for interpolation. To further clarify, note that u' and v' will each be in the range $[0, 4)$ and the actual keypoint location itself would be $u' = v' = 2$.

Your script should read in a sequence of u', v', I'_u and I'_v values and generate three lines of output for each:

1. The index of the closest orientation bin (in the range 0..7), the weight (the "interpolation weight") for that bin, and the index of the second closest bin and the weight for that bin. The weights should sum to 1.

2. The index of the closest bin in the horizontal u' direction and the weight for that bin, and the index of the second closet bin in the horizontal direction and the weight for that bin. However, only output this second bin if its index is in the valid range $[0, \ldots, 3]$. (You may assume without checking that the closest bin will be in the range $[0, \ldots, 3]$.) If both weights are output they should sum to 1.

3. Repeat the previous step but for the vertical v' direction.

Output of the weights should be accurate to two decimal positions.

At this point the exercise is complete.

The rest of this is to clarify the discussion from class and is not necessary to solve the lecture exercise. The SIFT descriptor histogram bins are a 3 dimensional array, for example in NumPy they might be

```
h = np.zeros((4, 4, 3))
```

Suppose the keypoint is at location (x, y) with orientation θ and scale σ. For each pixel (u, v) in the region around (x, y), with partial derivative $I_x(u, v)$ and $I_y(u, v)$, the mapping into the SIFT descriptor coordinate system is

$$\begin{aligned} u' &= \frac{(u-x)\cos\theta + (v-y)\sin\theta}{2\sigma} + 2 \\ v' &= \frac{-(u-x)\sin\theta + (v-y)\cos\theta}{2\sigma} + 2, \end{aligned}$$

where the 2σ is the width of each spatial bin. Adding 2 shifts the origin to the upper left corner of the region, making it easier to index the bins. Next, we map the derivative into the keypoint descriptor coordinate system as

$$\begin{aligned} I'_u &= I_x(u,v)\cos\theta + I_y(u,v)\sin\theta \\ I'_v &= -I_x(u,v)\sin\theta + I_y(u,v)\cos\theta. \end{aligned}$$

We let $g = ({I'_u}^2 + {I'_v}^2)^{1/2}$ be the gradient magnitude. Finally, we scale this by a Gaussian function of the distance of u' and v' from the keypoint location; call this value w_d. The u', v' and rotated derivatives are fed into your code.

Now for the actual voting. Let `u_tuples` be the one or two pairs of horizontal bin indices and weights output by your code. Also, let `v_tuples` be the one or two pairs of vertical bin indices, and let `angle_tuples` be the two pairs of angle bin indices. Then the voting for just the single pixel that was mapped into the keypoint descriptor coordinate system is

```
for iu, wu in u_tuples:
    for iv, wv in v_tuples:
        for ia, wa in angle_tuples:
            h[iu, iv, ia] += wu * wv * wa * w_d * g
```

It combines the three interpolation weights, the distance weighting, and the gradient magnitude, voting in up to eight bins, just for this one pixel. This is repeated across all pixels in the region. Once voting is done `h` is unraveled into a 128 component vector and normalized as described in the notes. This is repeated across all detected keypoints.