Anthony Poerio (adp59@pitt.edu)

CS1674: Homework 6 - Written

Due: 10/24/2016, 11:59pm

QUESTIONS

1) **List the steps involved in detecting an edge.**

   To find an edge, we know we need to search for a location in the image where there is *rapid change* in our pixel values.

   With this in mind, we know we can look at the gradients (or first derivative) of pixel values in our image (in any given direction) to determine how 'quickly' pixel values are changing.

   The steps are: First, run a smoothing filter to minimize noise in the image itself and ensure we are getting the most accurate gradient values. Then, compute the image gradients in both the X and Y directions. And finally, perform some sort of function to ensure which local maxima are actually edges and *not noise*. This can be done with **thresholding** (how I'd do it), or using more advanced techniques such as hysteresis that are not covering in great detail.

2) **What two steps do we iterate between when trying to cluster with the k-means algorithm?**

   To start, we randomly initialize k-cluster locations. We also randomly guess the cluster center in each of the clusters we have initialized. Then, we iterate between 2 different steps.

   Step 1: From each separate data point, find the cluster center it's closest to. We can imagine that this closest cluster center 'owns' that data point.

   Step 2: Look at all of the points we have assigned within each our clusters, and set the **new cluster center** to be the location equal to the mean of all points within the cluster selected (across all dimensions).

   If the cluster center *has changed* after finding the mean and assigning our center to it, repeat from Step 1.

When a cluster center is no longer moving after taking the mean of the cluster as a whole, we have converged upon the 'true' center of that cluster, and we can end the algorithm.

3) Name one difference between means-shift and k-means.

In the **mean-shift** algorithm, the user does not select some set number of cluster centers (as in **k-means**)—rather we select only a 'window size'.

We then initialize windows at all of our feature points, and shift the 'center' of each cluster window towards its '**center of mass**' until equilibrium is reached, much like in k-means.

However, as a chief difference:

- In **k-means,** we *randomly select* cluster centers and then move them around until equilibrium is found.
- In **means-shift**, we *must have already identified feature points we are interested in*, and then we find the cluster centers starting from those identified points.

In this way, means shift requires more information to get started, but results may be better in some cases.

4) Why is edge detection insufficient when we need to find lines in an image?

The biggest problem is that the difference filters used as a first step in edge detection respond strongly to random noise, which are NOT true edges.

Then, when we true to fit a line against the results of our difference filter, using the SSD calculation—the SSD line can be thrown ***very far*** off track (relative to the 'true' line) in an attempt to minimize the effects of a noisy outlier.

For this reason, we need algorithms like RANSAC and Hough to minimize the effects of these outliers and suppress them as we fit lines in our image.

5) **Intuitively, rather than formally, what is the underlying idea behind the Hough transform? In other words, why does the Hough transform make sense, as a way to deal with outliers when finding lines?**

The Hough Transform is almost like an exhaustive search of the image space from a small set of points, which ***might*** be part of a line. More specifically, given a point which could be on a potential line, it 'checks' the other points that might also be on that line with it (along the same gradient). Then, it assigns a higher value if more points are indeed of similar intensity change on that same gradient.

The way I'm thinking about is this: The Hough Transform takes potential points on a line and systematically checks the other points which ***should be*** on that line with it, trying to find corroborating evidence that a line exists, and that our selected point is a part of it.

For points that are outliers, very few other points in image space will corroborate their status as an element on a 'line', when we check them. More intuitively we can't have a line with just *one point*. We need a series of points **all** along the *same gradient*.

That's what the Hough Transform checks—do we have a **set of points** all in a row, that could form line with high probability—do we have a larger set of points that seem like an *extended gradient*? If yes—then we assign a high value to that set of points in Hough Space. If no, we assign a low value. When we're all done with the algorithm, the points with the highest values in Hough Space correspond to lines with higher probability.