Image Segmentation

Mihir Patel

May 2017

1 Introduction

In recent years, computer vision has become an increasingly popular field of computer science research. Such a field has vast potential for automating many difficult tasks from product quality to item searching and can lead to many future things such as augmented reality. With the advance of GPUs in recent years, machine learning, specifically convolutional neural networks, have finally the point where most items can be classified as well as if not better than humans. For example, handwriting classification on the MNIST dataset has surpassed 99.7 percent accuracy, whereas most humans can't beat 99. Powerful deep networks such as ImageNet have also made strong progress on general classification of any picture instead of just looking for specific things.

A big issue however that still remains in this field is segmentation. Essentially, the issue is that while machine learning works very well in classifying objects, it has to first know what an object is so it knows where to look. A naive approach of simply checking every possible location with a network filter does not work well in that it is $O(n^3)$ for each object. This is similar to how a brain functions, where first an image is split into objects (along with several other steps, such as deciding what is moving) before classification occurs.

2 Edge Detection

One of the first ideas was using edges to help distinguish objects. This is done by checking for a change in value across a specific pixel. A sobel matrix is first applied to each pixel, which identifies vertical and horizontal change in pixel values. If the sum of the absolute values is above a threshold, this is considered to be an edge as it has a significant change in color. In addition, the arctangent of the vertical value over the horizontal value can give a theta perpendicular to the contour.

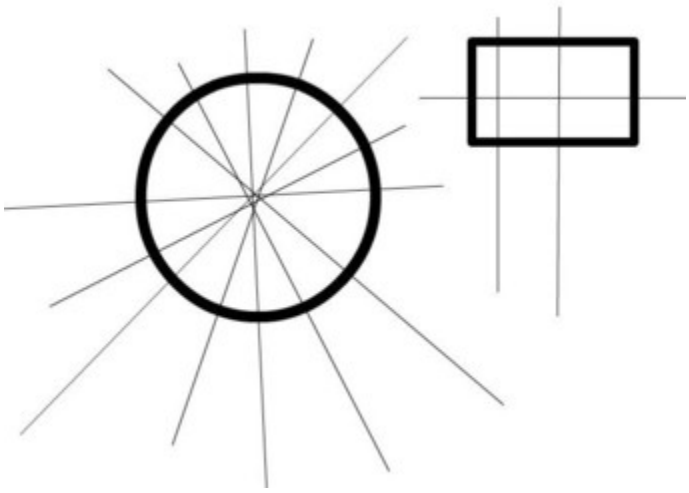
$$S = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

This method turns out to be very effective for most cases, however is restricted in its very local approach. One area in which this fails is smoother transitions of colors, such as in a sunset or human skin. As a result, a more commen approach is to use guassian matrices. Essentially, a normal or guassian distribution is created in two dimensions and extended outward based on a determined standard deviation, allowing for the input of further away pixels albeit at diminishing values to observe larger patterns.
So we now have these values for each pixel determining its change in color. However, how do we know what is considered a significant change? What do we use to determine our threshold? The short answer is this can't be automated. In most cases, some human input is required based on the problem or dataset to determine a threshold. There are howevr improvements that have been made for better functionality. Firstly, the notion of strong and weak edges is used. Essentially, a two threshold approach is used where one higher threshold determines definitive lines and a second lower one finds things that are likely lines. If a weak line is connected to a strong line, then it is included and considered to be part of the image. The other advancement is that image sampling methods are used to modify the threshold. For example, the standard deviation might be multiplied by some constant and added to the threshold to adjust it to account for varying images.

$$T' = T + a * \sigma$$

3 Feature Detection

From these edges, we can features to look at specific things. This is particularly useful if we are looking for a target object, such as a face, which has a very particular shape. One common method is the Hough Circle Transform, which looks at the lines perpendicular to the edges. As a circle's edges are always perpendicular to a ray from the center to the edge, high convergences of these lines can be seen at the circles. The issue that arises however is that this is very inefficient in terms of computation and memory ($O(n^3)$) as each point and its corresponding perpendicular line is looped over and all points must store how many lines pass through it along with the distance to calculate radius.



In order to deal with the issues slow speed, probabalistic Hough transforms are used. This algorithm instead uses 3 points to generate an ellipse. If this is similar to previously found ellipses, it averages them and increases the value of that ellipse. Otherwise it creates a new entry for this ellipse. This continues until an ellipse breaks a specific threshold.

4 Clustering Methods

So far, we have approached this problem as a way of finding the border of regions. However, sometimes such an approach fails as there aren't clear distinctions in regions. As a result, clustering methods are used. Essentially, the idea of clustering methods is to compress the data into regions that are defined by a particular characteristic instead of by their edge. A common example of this is k-means. In this algorithm, K cluster locations are selected randomly selected. Each pixel is then assigned a cluster based on factors such as distance, color similarity, and other factors of interest. A new average center of this set of pixels is found based on assignment, and the process is repeated with the new locations until the algorithm stabilizes.