Image Segmentation

George Tang

October 31, 2018

1 Introduction

Recall in edge detection, we found the boundaries of objects based on the change of pixel intensities. These boundaries divided, or **segmented**, an image into different regions. Here we discuss a **region segmentation**, where the image is segmented into *closed* regions based on region characteristics such as color and object classification. In this lecture, we will be focusing on the first approach. Region segmentation allows us to isolate individual parts of an image, while edge detection cannot.

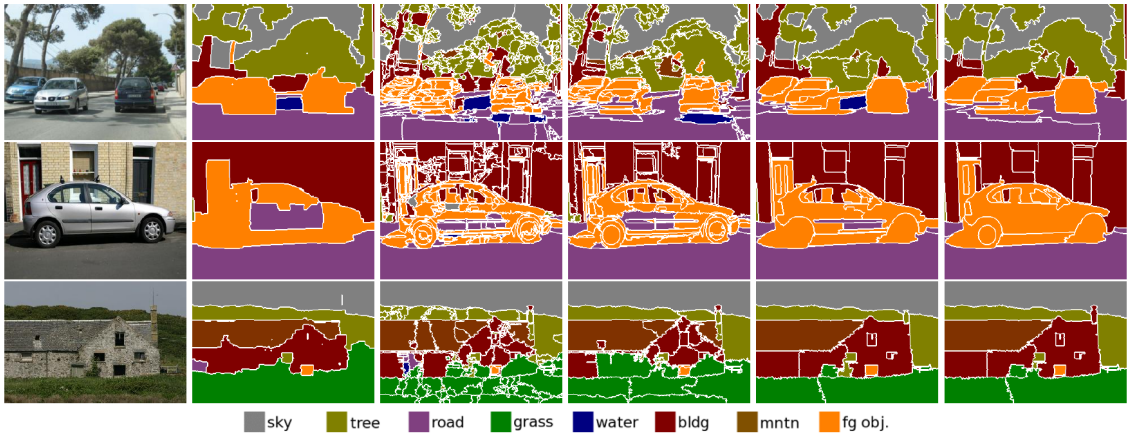


Figure 1: Comparison of region segmentation algorithms based on object classification. Notice that regions are completely separated from each other by boundaries.

2 Definition of Segmentation

Formally, an image I is partitioned into N regions if it satisfies the following properties:

- $R_1 \cup R_2 \cup R_3 ... R_{N-1} \cup R_N = I$
- $R_i \cap R_j = \phi, i \neq j$

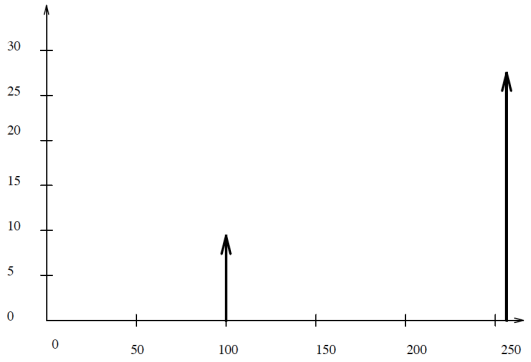
As we are segmenting the image by color, we need predicates (constraints) to determine the extent of regions. To simplify the discussion, let's begin by assuming out images are greyscale. Some examples of predicates are:

- All pixels in a region must have the same intensity.
- All pixels in a region must be within a set intensity interval.
- The standard deviation of intensities in a region must be less than a certain value

3 Thresholds and Histograms

When the image only contains one object, let's say a grey block on a black tablecloth, the greyscale image is a split into two regions. One region (the object) is represented by some intensity value, while the other (background) is represented by a different intensity value. When we plot the pixel intensities on a histogram, we see two peaks. Thus to get the two regions, we can define a single threshold T in between the two peaks of the histogram to segment the image. This special type of thresholding results in a **binary image**.

255	255	255	255	255	255
255	255	255	255	255	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	255	255	255	255



(a) **(b)**

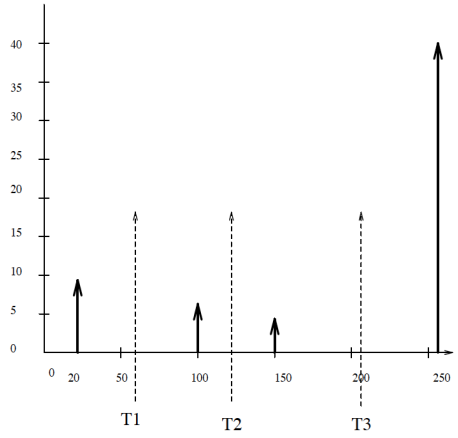
Figure 2: (a) binary image, (b) histogram

$$J(x,y) = \begin{cases} 1 & \text{if} \quad I(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

Notice that binary images can contain more than two regions (e.g. many identical objects on a background). For images that cannot be segmented into a binary image, the pixel intensity of the segmented image, $J(x,y)$, depends on the relation of the corresponding pixel intensity in the original image, $I(x,y)$, to a range of thresholds.

We can decompose the image into a histogram of color intensities and choose thresholds at the valleys of the histogram (remember in a histogram we can count the number of pixels in each interval) to obtain the regions. If there are N regions, there are $N-1$ thresholds, and we end up with N binary images, where the region of interest is represented by an intensity of 1 while everything else 0. From these images, we can assign each region a label and build a complete segmented image.

255	255	255	255	255	255	255	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	255	255	255	20	20
255	255	255	255	255	255	255	255
150	150	255	255	255	255	255	255
150	150	255	255	255	255	255	255



(a) **(b)**

Figure 3: (a) image with multiple regions, (b) histogram

4 Peakiness Test

Real life images are not discrete. They contain are contain a continuous spectrum of pixel intensities and noise. Thus the peaks of the histogram are not discrete. The **Peakiness Tests** decides which peaks in the histogram actually represent objects and thus regions. Good peaks are sharp and tall. The test is based on the following characteristics of a peak.

- W, P = width, height of peak
- V_a, V_b = height of two valleys on side of the peak
- N = number of pixels under peak (peak area)

The sharpness of a peak is defined as the ratio of the area of the peak to the rectangle formed by its width and height: $\frac{N}{W*P}$. Tall peaks are measured by the ratio of peak height to the average height of the adjacent valleys: $\frac{V_a+V_b}{2*P}$. The peakiness is defined as a product of these factors. If the peakiness is greater than some threshold, then the peak will be used for segmentation.

$$Peakiness = (1 - \frac{N}{W * P}) * (1 - \frac{V_a + V_b}{2 * P})$$

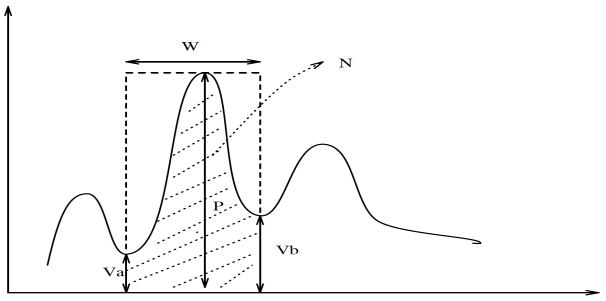


Figure 4: Components of a Histogram Peak

5 Connected Components

We N regions and N binary images, so how do we combine them into one image? Notice that we can process the binary images over a sliding window of time, meaning at the first instance, we segment the image based on the first threshold, and the at the second instance, the first and second thresholds, so on... and at the last instance, the last threshold. Let's assign each instance of time an id, and apply the sliding window. Every time we apply the thresholding, we also perform flood fill on the image (setting pixels in region to be 1), and then assign all 1 pixels the id. The end result is a fully segmented image. This algorithm runs in $O(N * P)$, where P is the number of pixels.

1. Scan the binary image left to right, top to bottom.
2. If there is an unlabeled pixel with a value of '1' assign a new label to it.
3. Recursively check the neighbors of the pixel in step 2 and assign the same label if they are unlabeled with a value of '1'.
4. Stop when all the pixels of value '1' have been labeled.

Figure 3.7: Recursive Connected Component Algorithm.

Figure 5: Recursive Algorithm

The issue with a recursive algorithm is that we are constrained by the memory of the call stack. We can also apply a two-pass sequential algorithm. This algorithm is based on **equivalence classes**, which is like

an id of a region. Let's assume in a binary image we have two regions. If we suddenly find a connection, then then we set the regions' equivalence classes equal to each other. In the end, we reassign regions with equivalent equivalence classes to a master equivalence class.

1. Scan the binary image left to right, top to bottom.
2. If an unlabeled pixel has a value of '1', assign a new label to it according to the following rules:

$$\begin{array}{cc} & 0 \\ 0 & 1 \end{array} \rightarrow \begin{array}{cc} & 0 \\ 0 & \mathrm{L} \end{array} \qquad \begin{array}{cc} & 0 \\ \mathrm{L} & 1 \end{array} \rightarrow \begin{array}{cc} & 0 \\ \mathrm{L} & \mathrm{L} \end{array}$$

$$\begin{array}{cc} & \mathrm{L} \\ 0 & 1 \end{array} \rightarrow \begin{array}{cc} & \mathrm{L} \\ 0 & \mathrm{L} \end{array} \qquad \begin{array}{cc} & \mathrm{L} \\ \mathrm{M} & 1 \end{array} \rightarrow \begin{array}{cc} & \mathrm{L} \\ \mathrm{M} & \mathrm{L} \end{array} \text{ (Set } L = M\text{).}$$

3. Determine equivalence classes of labels.
4. In the second pass, assign the same label to all elements in an equivalence class.

Figure 3.8: Sequential Connected Component Algorithm.

Figure 6: Sequential Algorithm

The final image segmentation algorithm is presented below:

1. Compute the histogram of a given image.
2. Smooth the histogram by averaging to remove small peaks.
3. Identify candidate peaks and valleys in the histogram.
4. Detect good peaks by applying peakiness test.
5. Segment the image using thresholds at the valleys.
6. Apply connected component algorithm.

Figure 3.9: Steps in segmentation using histogram.

Figure 7: Image segmentation Algorithm

6 Region Growing

Often image segmentation yields too many regions (see Figure 1) that we want to merge into larger, more representative regions. Imagine we have the initial segmented image, or **seed segmentation**. We can refine it through region growing algorithms.

The simplest region growing algorithm is **split and merge**. We have a predicate (e.g. same pixel intensity) that determines if two regions are merged or split. There are also a lot of other region growing algorithms, and many of them involve **graph theory**.

1. Split region R into four adjacent regions (quadrants) if $Predicate(R) = false$.
2. Merge any two adjacent regions R_1 and R_2 if $Predicate(R_1 \cup R_2) = true$.
3. Stop when no further merging and splitting are possible.

Figure 3.13: Split and Merge Algorithm.

Figure 8: Split Merge Algorithm

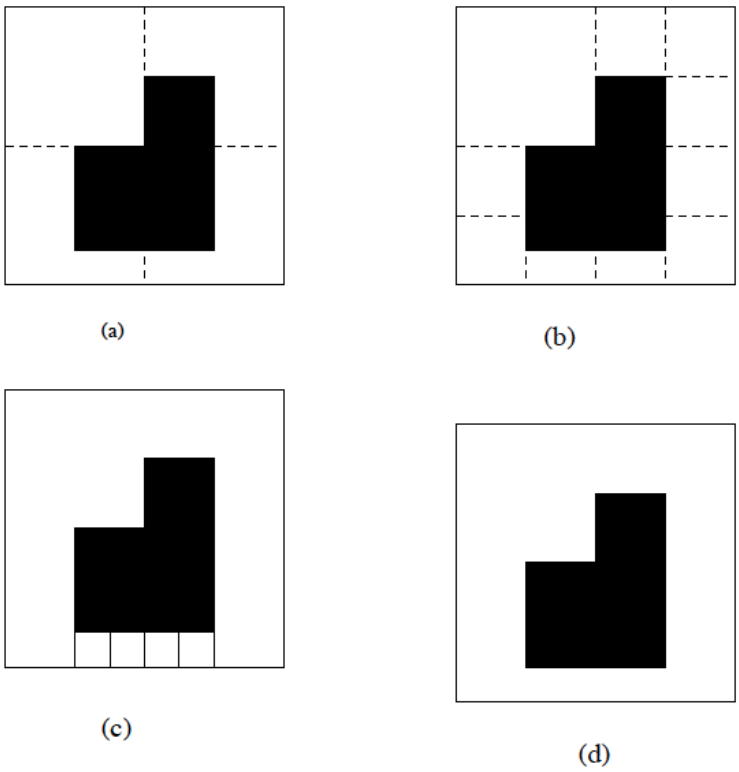


Figure 9: Demonstration of the Split Merge Algorithm

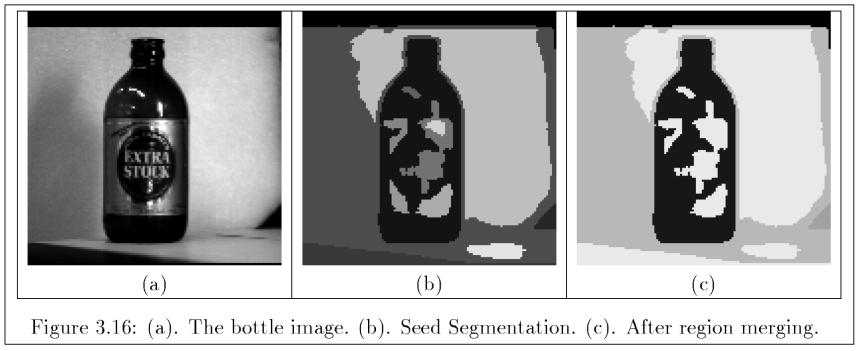


Figure 10: Entire Image Segmentation Process

7 Applications

The applications of image segmentation are numerous. First, because image segmentation divides the image into different regions with similar properties, it can aid in locating objects and classify objects. In traffic control systems or autonomous vehicles, it can help pinpoint objects in the surroundings. In the medical field, image segmentation allows doctors to see different areas of the body and observe blood flow.

In this lecture we covered only the basics of image segmentation. Modern segmentation is usually accomplished using convolutional neural networks or robust computer vision algorithms, but many of the concepts are the same.