CS 6810/7810: Wavelets and Wavelet Algorithms
Assignment 4
Curve Analysis with 1D Haar Wavelets

Vladimir Kulyukin
Department of Computer Science
Utah State University

February 4, 2017

Learning Objectives

1. 1D HWT
2. Curve Analysis
3. Object Detection

Introduction

In this assignment, we will explore some of the ideas discussed in Lecture 4. Specifically, we will investigate how 1D HWT wavelets can be used to analyze curve topologies to detect objects. While we will focus on detecting landing pads of Langstroth beehives, the same curve analysis techniques generalize to curves from other domains.


Figure 1: Pad localization: original image (left); localized pad (right)

Let us state the problem. The left image in Figure 1 is a sample image captured by a Pi camera. The right image in Figure 1 shows a detected pad. This is, of course, an ideal case. Typically, localized pads have some extra bits and pieces of grass. The end objective is to take a sample image similar to the left image in Figure 1 and localize a landing pad as close to the ideal case as possible.

Separating Grass from Hive

The first step is to separate the grass from the hive. The left image in Figure 2 shows the same original image and the right image in Figure 2 shows the grass texture separated from the hive texture. As can be seen in the right image, the grass texture is indicated by a large presence of white pixels.

I will skip, for the sake of clarity and brevity, how these grass pixels are identified. In this assignment, you will work with images like the right image in Figure 2. The zip archive `hw04_images.zip` contains all the images. I attached the archive to the announcement "CS 6810/7810: Spring 2017: Images for Assignment 4" posted around 11:00pm on February 4. The archive contains pairs of images where the images with the letter `b` are black and white images with white grass pixels corresponding to the original images whose names do not have that suffix. For example, `2015-07-27_17-42-44.png` is an original image whereas `2015-07-27_17-42-44b.png` is the corresponding black and white image with grass pixels.

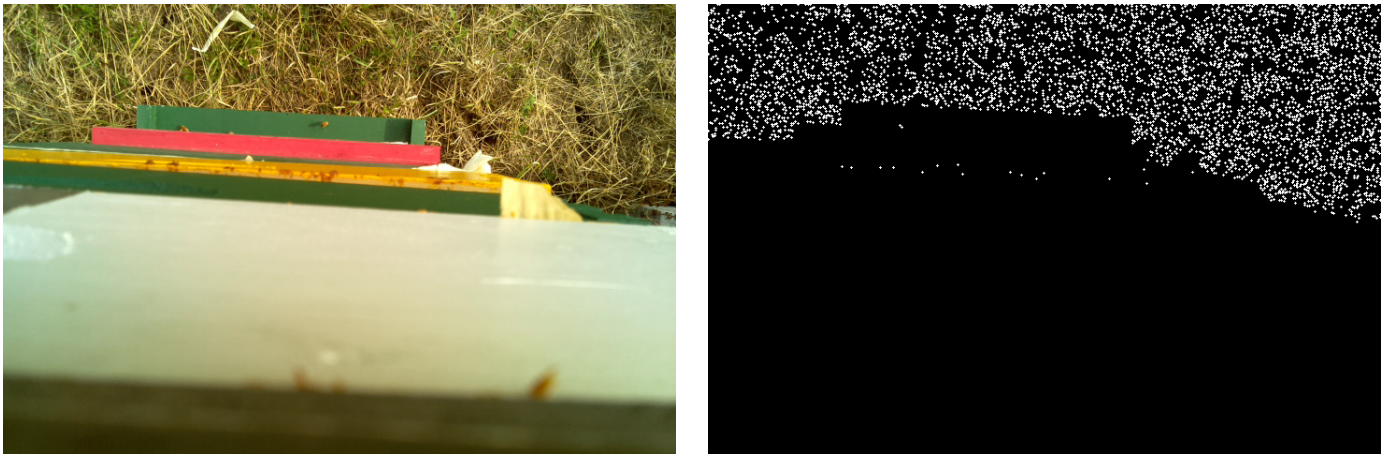

Figure 2: Detection of grass: original image (left); grass (right)

Curve Topology Analysis

Can grass pixels be used to localize landing pads? This is the question that we will explore in this assignment. We can start with outlining a conceptual border curve separating the grass texture from the hive texture. We do not have to do it programmatically. This curve is a conceptual construct that makes our analysis easier. The left image in Figure 3 shows an image with a drawn border curve.

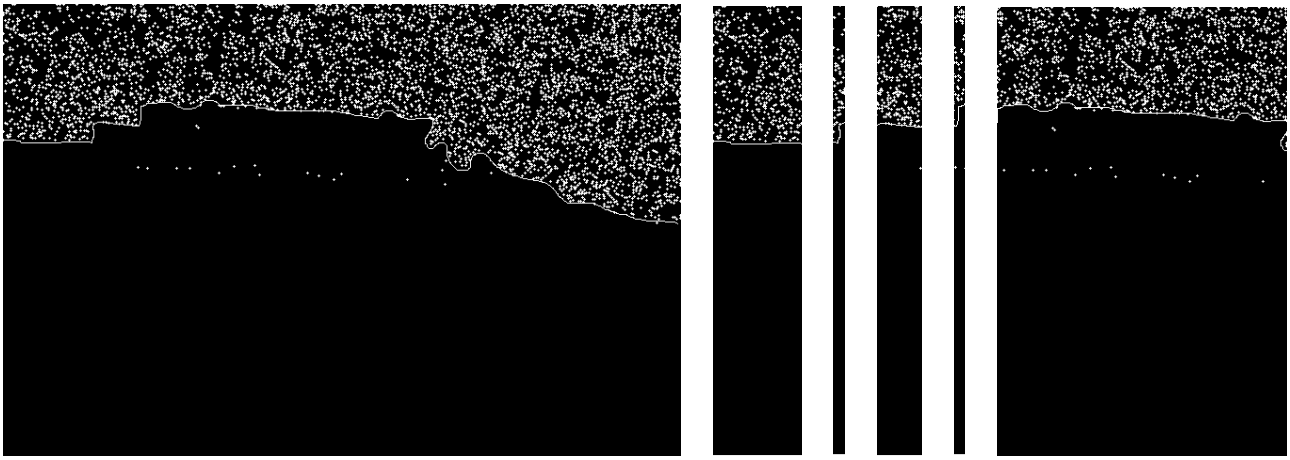

Figure 3: Curve topology: curve under grass (left); first five segments: flat, climb, flat, climb, flat

We can take a closer look at the topological structure of this curve. Going left to right, the curve starts with a flat segment. A flat segment may go slightly down or slightly up but does not have any sharp climbs or declines. This flat segment is shown in the second black and white stripe image from the left. The next topological segment is a sharp climb shown in the third image from the left. This climb is followed by another flat segment shown in the fourth image from the left. This is followed by another climb in the fifth image from the left. The sixth image from the left shows a rather long flat.

The next four segments are shown in Figure 4. There is a decline shown in the second stripe image from the left. Then there is a flat shown in the third image from the left. This flat is followed by a climb and a decline shown in the fourth and fifth images from the left, respectively.

Computing Curve Topology

How can one compute elements of the curve topology outlined in the previous section? We can use the notion of horizontal and vertical projections. Let us define $C[i, j]$ to be the count of white pixels, where a white pixel is defined as any pixel whose value is above some threshold (e.g., 200), in all columns from i to j. If $i = j$, then $C[i, j]$ is the count of white pixels in column i. Effectively, $C[i, j]$ is a histogram bar whose height is equal to the count of its white pixels.

Suppose we have decided on how wide our individual bins will be. Then we can take as input an image like the right image in Figure 2 and move left to right through the image computing all $C[i, j]$ for each valid value of i and j. If all $C[i, j]$ counts are placed orderly, i.e., from left to right, in an array, they constitute the vertical projection of a given image. Similarly, we define $R[i, j]$ to be the count of white pixels in all rows from i to j. If $i = j$, then $R[i, j]$ is the count of white pixels in row i. $R[i, j]$ is a histogram bar whose height is equal to the count of its white pixels. All $R[i, j]$ counts placed orderly into an array constitute the horizontal project of a given image.

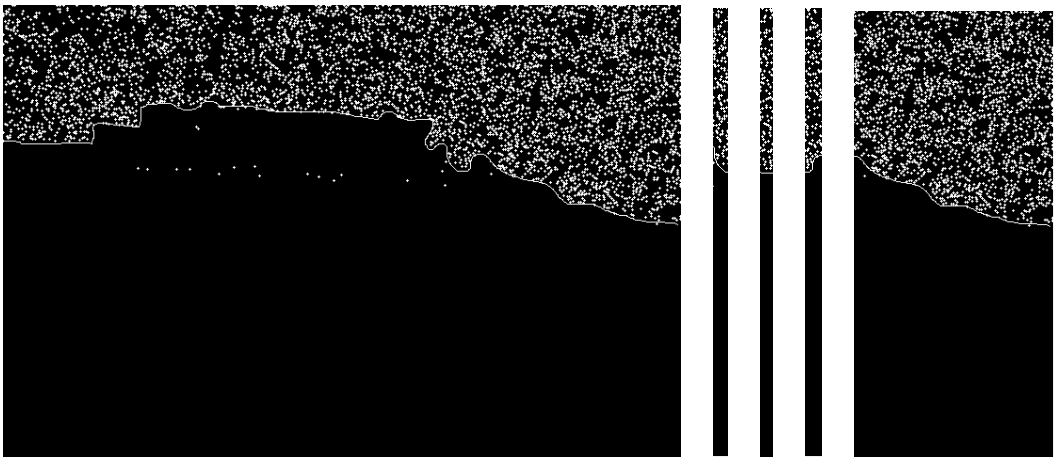

Figure 4: Curve topology: curve under grass (left); next four segments: decline, flat, climb, decline

The 1D HWT can be applied to the vertical and horizontal projections to identify the flats, climbs, and declines by means of wavelets. Why do we need to identify flats, climbs, and declines in the first place? Take another look at the left image in Figure 3. You probably notice two ladder steps. Once you can identify the second ladder step, you are on well on your way to localizing the pad. Obviously, the length of both horizontal and vertical projections must be equal to an integral power of 2. So, when you implement your solution, you may need to pad your horizontal and vertical projections with 0's. Another research variable to investigate is how many iterations of 1D HWT allow you to identify the curve topology best. Toward that end, implement a Java class `PadLocalizer` that has the following two methods:

```
public static void localizePad(String originalImageFile,
                               String grassImageFile,
                               String outputImageFile) {}
public static void localizePadInDir(String originalDir,
                                    String grassDir,
                                    String outputDir) {}
```

The first method takes an original image like the left image in Figure 2, a black and white grass pixel image like the right image in Figure 2, and a path to an output image. The method does the 1D HWT based analysis of the image in the second argument to localize the pad, draws the boundary lines of the localized pad in the original image and saves the image with the localized pad in the image specified by the third argument. The second method takes three directories: a directory with original images, a directory with black and white grass pixel images, and a directory where images with drawn localized pads are saved.

What To Submit

Submit your PadLocalizer.java or PadLocalizer.py via Canvas. Also, submit a two page pdf writeup describing your algorithm and your findings. Briefly address the following questions in your writeup. Did you use both vertical and horizontal projections or just one type of projection? Which bin widths gave you the best localization performance? How many iterations worked best for you? What was your greatest challenge?

Happy Hacking!