ECE 4522 MATLAB Assignment 3

Leomar Durán
College of Engineering
Temple University
Philadelphia, PA
leomar.duran@temple.edu

***Abstract*—This assignment introduces the backward difference system, and uses cascading to produce an edge detector. This works because the difference of the current value and the previous are zero if they were equal, and larger if they are more different. The backward difference system is produced in 3 steps, and is first tested. We finally apply the edge detector to a photograph of a barcode to prepare it for decoding.**

***Index Terms*—convolution, filter, backward difference system, difference system, edge detection, threshold, barcode, encode, decode**

I. Objectives

The objectives of this report is to show the use of a backward difference system, implement an edge detection filter and use this filter to read a barcode.

After this lab, students will be able to apply a difference system, and explain what it does.

II. Methods

`./Ece4522/MatlabAssignment3/B1.m` demonstrates and tests the first 3 steps of the edge detection filter. More specifically, it

1) creates a sample signal, a square wave with a 10-sample duty cycle, a period of 30 samples, and an amplitude of 255.
2) uses the backward difference system modeled by (1) to find the edges of the sample signal, where its value changes.

$$y[n] := x[n] - x[n-1]. \tag{1}$$

3) checks the absolute value of this resulting signal against a 60 % tolerance to create a sparse location signal.
4) compresses the location signal by removing all ZERO values using the `find()` function.

Rather than using a sample signal, `./Ece4522/MatlabAssignment3/B2.m` first reads from a photo of a barcode, and samples the middle row as the input signal.

After compressing the location signal,

1) the signal is once again pass through a backward difference system.
2) the values of this signal are then regularized to between 1 and 4 by dividing by the base width for that 59-edge interval.
3) finally, the first end code 1-1-1 is located, and the next 59 values from and including that end point are cut out and decoded.

`./Ece4522/MatlabAssignment3/B2OFF.m` additionally, works on a rotated image. This is done by finding the angle of rotation θ and taking its negative complement $\theta - \frac{\pi}{2}$. Each coordinate in the matrix is then multiplied by $R(\theta)$, and the value of the original pixel coordinate is assigned to the new one.

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}.$$

III. Results

A. Part B.1 Edge Detection and Location in 1-D Signals

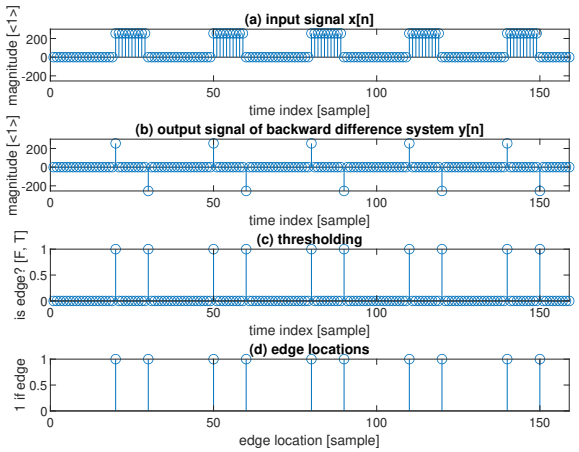

Fig. 1. (a) the input square wave signal, (b) the result of the backward difference system, (c) the result of thresholding by checking the absolute value against a 60 % tolerance, and (d) the output compressed edge locations.

B. Part B.2 Bar Code Detection and Decoding

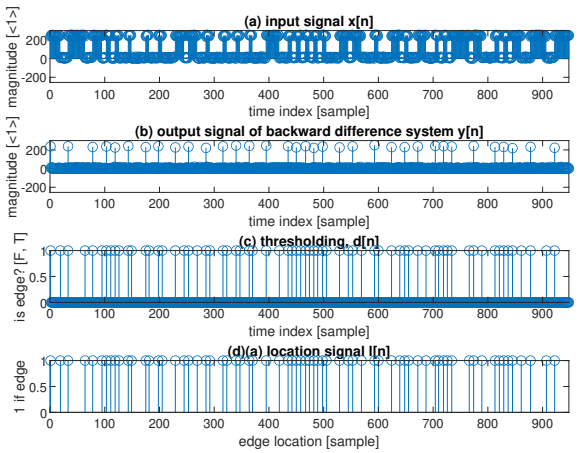

Fig. 2. (a) the input square signal read from the barcode image, (b) the result of the backward difference system, (c) the result of thresholding by checking the absolute value against a 60 % tolerance, and (d) the output compressed edge locations.

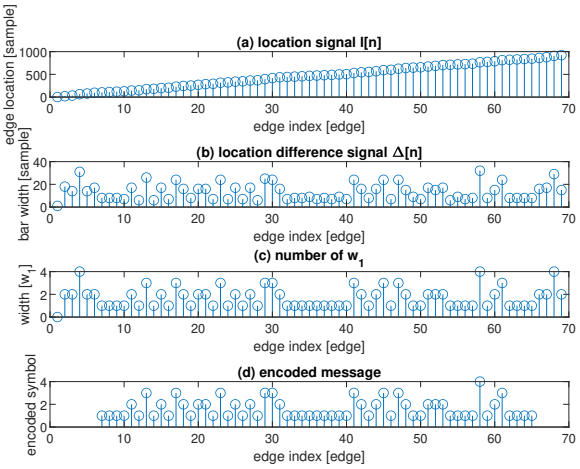

Fig. 3. (a) the location signal, (b) the result of running the location signal through the backward difference system, (c) $\Delta[m_0, n]$ in terms of w_1 and (d) the cropped out message from part (c).

The results of `./Ece4522/MatlabAssignment3/B1.m` show

```
decodeMessage =

     8     8     2     7     8     0     4
     5     0     1     6     5
```

These were the digits expected from the original barcode.

C. Part OFFv3. Bar Code Rotation

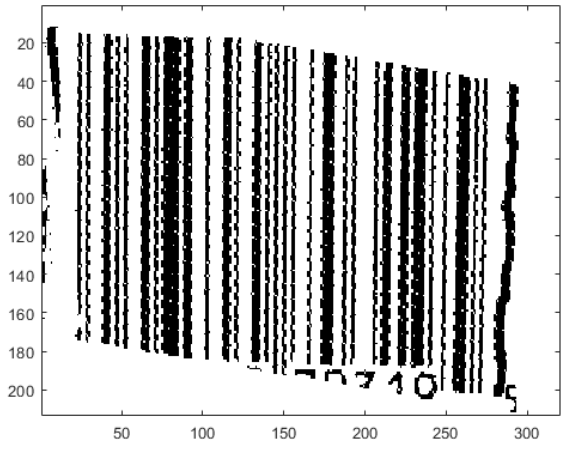

Fig. 4. Unrotated image from OFFv3.png. This is the image created in the first of part OFFv3.

IV. Insights

A. Part B.1 Edge Detection and Location in 1-D Signals

The signal in Fig. 1(b) is created by subtracting the present value of $x[m_0, n]$ in part (a) from its value 1 sample ago as represented by (1). This has the result of ZEROing out in $y[m_0, n]$ where $x[m_0, n]$ is flat because these values are constant, and the difference between a constant and itself is ZERO. It also emphasizes edges because their result is non-ZERO. These appear as impulses. They are positive impulses when the $x[n] > x[n-1]$ and negative impulses when $x[n] < x[n-1]$.

Along with the thresholding in Fig. 1(c), this marks the edge immediately after the transition because of the reference to $x[n-1]$.

B. Part B.2 Bar Code Detection and Decoding

Since $\Delta[m_0, n]$ represents the differences in the locations of edges, the vertical axis in Fig. 3 shows the width of the bar enclosed by those edges. Ignoring the first ZERO height, we can see 4 distinct heights of approximately $8.2, 16.4, 24.6$, and 32.8 samples.

Now we can justify that the total width of a signal is 95 times the base width of the shortest bar. This is because there are 12 digits each with width 7, 2 end delimiters of width 3, and 1 middle separator of width 5. So the message is $(12\,\text{digits} \times 7w_1/\text{digits}) + (2\,\text{ends} \times 3w_1/\text{ends}) + (1\,\text{separator} \times 5w_1/\text{separator}) = 95w_1$. Also, from part **B.2** of the lab manual, we know to expected 59 bars in the barcode message. Thus, we can expect any interval of 59 bars to be of size $95w_1$. By dividing these by 95, we find the base width of the thinnest bar w_1.

V. Conclusion

We have implemented and tested a backward difference system. Then we have shown how two backward difference systems can be used to implement an edge detection system, and further used that edge detection system to help decode a barcode.

As the resulting value of `decodeMessage` shows, we were successfully able to find the encoded message in the photograph of the barcode.

In Part OFFv3, we were able to recreate the unrotated image. However, it seems that the quality is off because it is not good enough to decode using the same algorithm. Maybe another filter needs to be applied such as sharpening.