Finding the Box with an Image Processing System

Andrew Morris

Saint Cloud State University  
733 13th Ave SE  
St. Cloud, MN 56304  
7633819141

moan1303@stcloudstate.edu

**ABSTRACT**

This program was designed to find a cardboard box in an 8-bit greyscale image. Images were processed with different techniques such as: Histogram Equalization, Simulated Gaussian, variable window noise reduction, Kirsch Operations, Stefanelli & Rosenfeld’s Thinning Algorithm, and the Hough Transform. Images were processed using different combinations of the aforementioned operations through trial and error. The techniques implemented produced satisfactory results in producing a line image.

However, identifying any of the boxes was not achieved. Given more time, the program would identify 3 sets of parallel lines in a window, and draw lines over the box to identify the box within the digital image.

# INTRODUCTION

According to Ballard and Brown, Computer vision is “the construction of explicit meaningful descriptions of physical objects from images”. [1] This refers to image understanding, which is different than image processing.

Building a vision system has been a significant area of research in the field of Artificial Intelligence. The goal is take principles of human vision, and apply them to digital images. How does a human recognize a doorway when he or she walks down a corridor? What distinguishes the edge of the road when driving in a car? How can a human read a handwritten letter, turning the inaccurate squiggles into a string of characters? Of course, humans do not obtain information only by reading and talking; a great deal of what we know comes from direct observation of the world around us. [2] This is perhaps the greatest challenge in the field of computer vision.

This task of image understanding can be broken down into further steps. Beginning with a digitized image, reduce the noise to prepare for edge detection. Next, detect the edges of objects like a pencil sketch, and group these edges into lines. Thereafter, segment the lines to outline regions. Afterwards, describe the surfaces the regions depict. Subsequently, group surfaces into objects. Finally, objects are then to be identified.

Computer Vision is computationally strenuous. Also, there are other challenges, in general, when analyzing a digitized image. First of all, the formation of an image loses information about the scene in which it was taken. Second, information is lost within each pixel because a pixel only holds the average brightness over its part of the visual field.

# THE PROBLEM

The objective of this system is to identify a cardboard box in an 8-bit greyscale bitmap. After identifying the box, the box should be labeled or outlined.

Seven images were presented. Also provided were code segments, and other useful information related to bitmap image manipulation. These included copying a bitmap file, saving a bitmap, and extracting header information.



Figure 1. An example of one of the images containing the cardboard box.

With the given goal, the system is to be made to identify the box shape. Such a system consists of processing steps that includes Histogram equalization, smoothing, edge-detection, thinning, noise-reduction, and Hough transform.

# PROCESSING STEPS

In order to define accurate lines within each image, several processing techniques were implemented. In the design of the processing steps, it came as no surprise that the order in which processing operations were applied was relevant to any success.

Some of the operations described include: Gaussian distribution smoothing, Kirsch Operators in edge detection, Stefanelli and Rosenfeld’s Thinning algorithm, variable window noise reduction, and Hough Transform line detection algorithms.

### Histogram Equalization

Histogram Equalization is a series of operations that increases the contrast over most of an image for the purpose of finding edges. This is achieved through an algorithm that distributes the amount of each greyscale value evenly throughout the image. The end result is an image with a uniform distribution of greyscale values. Notice in Figure 2 how the contrast is increased after the application of Histogram equalization.

Figure 2. Left: The box before Histogram Equalization. Right: The box after Histogram Equalization has been applied.

### Simulated Gaussian Smoothing

The smoothing method takes a 3x3 window around each pixel, and weighs the values according to Figure 3. The summation of these weighted values is divided by the total weights. The center pixel’s value is the result of this operation. It can be seen that the new pixel value will be an average value over the window. Performing this operation decreases the amount of extreme local maxima or minima in pixel values throughout the image. Figure 1 shows the weights assigned to each pixel.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Figure 3: Weights of Smoothing operation (center is the current pixel)

### Kirsch Operations

For edge detection 3x3 Kirsch operators were used. These operators are not based on a strict window size. In fact, Kirsch mentioned in his proposition of these operators that the operators are flexible in their usage [3]. Because of the 3x3 operator I had to make four passes over the image; one for each operator. No weighting was added to the Kirsch operators.

Below is the representation of the Kirsch operator that detects vertical edges.

*K* is considered an edge element if:

) > Threshold

|  |  |  |
| --- | --- | --- |
| L1 |  | R1 |
| L2 | K | R2 |
| L3 |  | R3 |

Figure 4: Vertical Edge Kirsch Operator

### Variable Window Noise Reduction

Noise reduction is the process of eliminating unwanted pixels when processing an image. It is beneficial to remove unwanted pixels because the unwanted pixels reduce the effectiveness of other detection and processing operations. The Variable Window Noise Reduction take a center pixel, then scans the specified perimeter. If a pixel is encountered during this scan, then it is possible that this frame of pixels can belong to a desired line. If no pixels are encountered in the scan, then erase all of the elements in the window. We can assume from that point that the cluster of pixels is not beneficial for building a line.

The benefits of such an algorithm is that hanging edge elements are preserved while dense clusters of noise are removed.

An obvious precondition of this algorithm is that desired edges need to be mostly continuous, and the window used is not so large as to eliminate large areas.

Ex: In Figure 5, the three pixels in the window would be eliminated in the noise reduction.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Figure 5. An example of a 5x5 window.

### Stefanelli & Rosenfeld’s Thinning algorithm

To thin the edge image, the Thinning algorithm developed by Stefanelli and Rosenfeld was used [4]. Through a series of iterations, the edge image is thinned as the line image grows. The resulting image is made up of final points. Final points are pixels that, if removed, would break a line. This way only critical pixels for defining a line are kept. By continuing to define these final points, and eliminating contour points, the final result is an image that is completely comprised of final points. This means, all lines in the image are one pixel wide.

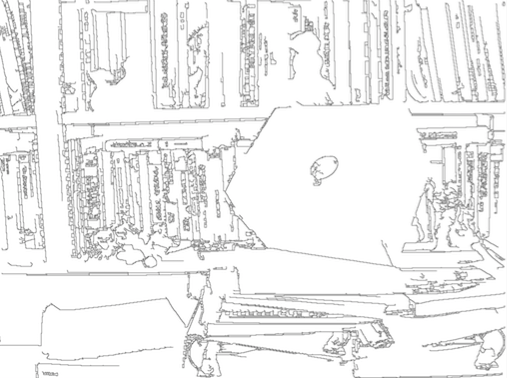


Figure 6: An example of an image after the thinning algorithm has been applied.

### Hough Transformation

Hough Transformation was used to identify lines in the thinned image. The Hough Transformation is helpful in the fact that the algorithm uses global information to identify lines. The algorithm uses the possible lines resulting from all line elements, as defined by the thinning algorithm.

Each pixel “votes” for a line that to which it can belong. This process is augmented by keeping track of the coordinates of each pixel that voted for the individual lines. This helped track the termination points of each line, and the pixels relevant in each of the proposed lines. The “votes” for each line are considered, and then compared to some threshold. A line that has a sufficient amount of votes is considered to be a line.

The angle was increased by two degrees for each vote. Each edge element votes for ninety different lines where the current pixel is the axis.

The intent was to define an equation that takes the total number of votes into account, and defines the threshold based on these statistics. However, tuning for each image ultimately produced the best line images.

# IMPLEMENTATION

For this implementation, the C++ programming language was chosen. The choice was made on the grounds of performance, and language familiarity.

The order in which these techniques were implemented varied greatly. Through many trials it was found that for certain operators, running twice through the iterations helped greatly.

Originally, histogram equalization was included in the processing steps. After several trials it was determined the histogram equalization made the box faces more similar. Therefore, the faces of the box would not be distinguished from each other. In some of the images, the histogram seemed effective, but in others more noise was introduced.

Ultimately, the order of operations consisted of:

1. Smoothing
2. Smoothing
3. Kirsch
4. Noise Reduction (11x11 window)
5. Noise Reduction (25x25 window)
6. Thinning
7. Hough Transform
8. Reprint pixels for elected lines
9. Noise Reduction (50x50 window)
10. Hough Transform

First, the program started by smoothing over the image. Doing this twice sufficiently eliminated most extrema in the image. The initial smoothing tended to leave isolated influxes in pixel values. This behavior is exhibited in the fact that the smoothing operator only views a 3x3 window. Performing this operation a second time, the initial value of each pixel is reflected from it’s surrounding values; further dampening sharp isolated changes. This helped eliminate initial noise for the Kirsch operator.

Then, upon the first use of the Kirsch operator, it was found that the 3x3 window still introduced too much noise. After implementing the noise reduction, however, the noise left behind by the Kirsch operators were cleaned up nicely.

In the noise reduction steps, a two step noise reduction was used. The goal of the first step is to eliminate small isolated edge elements that would interrupt the noise reduction in the 25x25 window. In Figure 7, the desirable effects of the two step noise reduction can be seen. The first step, apply 11x11 noise reduction, then apply 25x25 noise reduction.

Figure. 7: Left: original Kirsch image. Middle: 11x11 Noise reduction. Right: 25x25 Noise reduction

After noise reduction, Stefanelli and Rosenfeld’s thinning algorithm was employed. This operation yielded adequate results. A problem with this algorithm is that very thick edges tend to get small bubbles in them. When an edge is too thick, the thinning algorithm will not define the center of an edge to be an edge element. This causes the thinning algorithm to define the lines surrounding these holes in the edges as final points. To combat this issue, a preliminary run of the Hough Transform was applied, then the pixels that voted for the elected lines were reprinted. This tended to break up thinned lines that were established. Then by further noise reduction, smaller clusters of bubbles could be eliminated (See figure 8).

Figure 8. Left: Thinned image. Right: Reprinted pixels after Hough Transform

After the reprinted Hough Transform, then two more iterations of Noise reduction were performed to eliminate these clumped areas of bubbles.

Through one more iteration of Hough Transform with a smaller threshold, a satisfactory line image was obtained.



Figure 9. The final line image.

Finally, the approach to finish box detection was to use the idea of parallel lines. Due to time constraints, this was never implemented.

# PERFORMANCE

After comprehensive manipulation of the images for the box-detection while making cumulative progress by investigation, eventual images were collected through implementation. A few were delineated above. Since BMP files are a lossless and uncompressed formatted to store graphics file, they can be quite large. Undertaking two-dimensional array of image data desired numerous operations for respective pixel refinement. Administering those operations obligated quite a dollop of computer’s memory as foreseen.

# DISCUSSION

The approach can be altered to identify other shapes resting on its geometry. Mostly, any shapes with lines, rather than curves would be more appropriate with least amount of modifications. The latter might require extensive refactoring. This is due largely in part to the Hough Transform’s focus on defining straight lines.

# CONCLUSION

Computer vision, is a very significant area of research in Artificial intelligence. As humans, our anatomy is intricately arranged to make vision seem trivial. However, creating a program to replicate these properties is very difficult.

This program attempts to identify a cardboard box in a series of greyscale images. These images contained a box placed in a variety of settings. In order to extract meaning from an image, it must first be processed. Gaussian smoothing, Kirsch operators for edge detection, variable window noise reduction, Stefanelli and Rosenfeld’s thinning algorithm, and Hough Transform were used.

Ultimately, attempts to find the box in any of the images were futile. However, through processing steps a very desirable line image for box detection was produced.

# REFERENCES

1. Dana H. Ballard and Christopher M. Brown. 1982. *Computer Vision* (1st. ed.). Prentice Hall, Upper Saddle River, NJ.
2. Matt Ginsberg. 1993. *Essentials of Artificial Intelligence* (1st. ed.). Morgan Kaufman Publishers, Inc. San Francisco, CA
3. Kirsch, R. 1970. Computer determination of the constituent structure of biological images. *A Applied Mathematical Division, Nat. Bureau of Standards.* (Sept. 1970), 795-825. DOI= <http://www.mel.nist.gov/msidlibrary/doc/kirsch_1971_comp_deter.pdf>.
4. Stefanelli and Rosenfeld, 1971. *Some parallel thinning algorithms for digital pictures.* J. ACM, 18 (1971), pp. 255-264