Finding Boxes with Cycles

Rob Owens

CSCI 440

rtowens@stcloudstate.edu

ABSTRACT

One difficult problem in computer vision is to identify objects in images. This program tests a method of examining the lines of the image for cycles which would form planes of potential objects. The success rate of this method is low as this method does not distinguish between other features of a box such as rectangular planes or parallel lines. This can cause multiple false positives for the program, particularly when background objects are made up of multiple planes similar to the box.

1 Introduction

This program is tasked with finding a cardboard box in a bitmap file (black and white image). This is accomplished through several steps with each getting more abstracted from the pixels themselves. The method illustrated here is to smooth the image, make an edge image, generate a Hough transform of the image, search for cycles in the image, and then combine the cycles to make an object.

2 The Problem

The main problem that is being attempted is identifying shapes in an image. More specifically identifying a cardboard box amongst various backgrounds. The images are stored in 8-bit bitmap (BMP) files. Each image contains a single box but may contain any number of other objects and have various backgrounds. An example image is shown below with a box on a chair in Figure 1.

A chair sitting in front of a table

Description automatically generated

Figure 1: Example starting picture

3 Processing Steps

There are 5 steps in processing an image for this problem: smoothing the image, use a Sobel operator to detect potential edges, thin the image, generate a Hough transform to generate lines, search for planes, and finally combine the planes to make a box.

3.1 Smoothing

The first step is to smooth out the image slightly to help reduce noise in the image. This is done by averaging a pixel with its surrounding 8 pixels and itself. This is used to remove small bumps but keep larger outliers mostly intact, which are generally where edges are. This step is repeated on some images where more noise is present, to reduce the amount of time it takes in latter stages o remove the same parts.

3.2 Edge Detection

The second step is to generate an edge image from the image. This is done by applying the Sobel operator to the image. The Sobel operator functions by applying two kernels to an image and then adding the results together to produce an image where the whiter a pixel is (larger number) the more of a difference that pixel has between either side of the pixel. Shown below are the two Sobel kernels used in this processing.

|  |  |  |
| --- | --- | --- |
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

|  |  |  |
| --- | --- | --- |
| +1 | 0 | -1 |
| +2 | 0 | -2 |
| +1 | 0 | -1 |

3.3 Thinning

The third step is to thin the generated edge image. This step uses the thinning algorithm from Stefanelli and Rosenfeld. The thinning algorithm starts by identifying pixels that are considered final, or necessary in order to keep the edge intact. These points are usually ones that connect 2 other pixels on opposite sides, as demonstrated in Figure 2. In these examples the final point would be the center pixel.

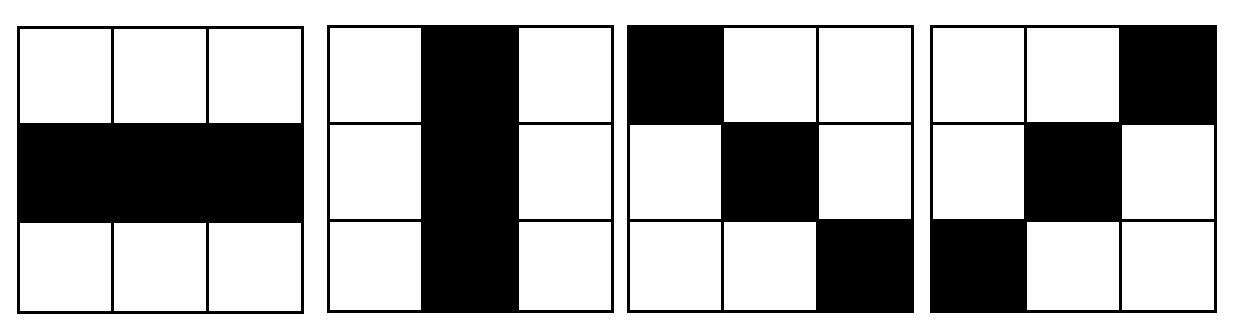


Figure 2: Example Final Points.

3.4 Hough Transform

The third step is to do a Hough transform of the image to break the image into lines. This is accomplished by iterating through the image and for each pixel, calculating a line that is perpendicular to a line that runs through this pixel. As shown below in figure 3, the calculated line is the line extending from O. Thus, each potential line in the image can be described as (r, **θ**) and is a list of pixels that represent the line. After this is done the line can be tweaked slightly to better represent the edge in the image.



Figure 3: Example of a line represented in a Hough transform

3.5 Finding Planes

The fourth step is to find planes in the image. Planes are a cycle of lines in this case and can be found by taking any line and performing a depth first search up to depth 4, since we are looking for rectangles. This search goes to the end of a line segment and checks for nearby edge pixels that are part of a line found in step three. Then a cycle is found if starting line becomes an option again.

3.6 Combing Planes

Finally, objects can be formed by combining planes that share edges. More specifically in this case the object is a combination of 3 planes that share one edge with each other plane. As shown below, plane 1 shares an edge with 2 and 4, plane 2 shares an edge with 1 and 4.

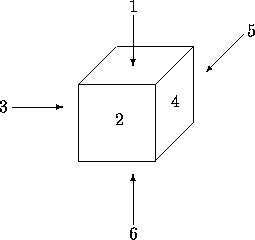


Figure 4: A cube with numbered faces

4 Implementation

The program was implemented using C++ and loads the original image into memory before the program tries to manipulate the image. The smoothing operation is implemented by averaging a pixel with its neighbors. The Sobel image is generated by applying the Sobel kernels to each pixel and adding the sum. Then each pixel is made an edge if the Sobel value is greater than the threshold (50 in this case). The Thinning Algorithm is implemented as a series of functions, with detecting each state being a function. Then the image is repeatedly thinned until there is no more thinning to be done. This will reduce the number of edge pixels to be considered later in processing and help reduce duplicate lines that represent the same edge. The Hough transform is an array of vectors which contain a list of pixels that are for the line. By using this, it allows for two things to happen: 1 is that searching in a line segment for a pixel or close pixel is vastly improved as we can use a binary search for the list and 2 finding the end of the line are very simple with the start being the first pixel and the last pixel being the last in the list. While the depth first search was not completed, a simpler algorithm was completed. This algorithm tests the number of intersections each line has and removes the dead ends. These are lines with less than 2 intersections as they can’t be a part of a cycle. This process is repeated till no more edge are being removed. This will mimic what a depth first search would do, as that would identify cycles and keep those edges, while removing the dead ends. Since the depth first search was not finished, combing the planes was also not finished and is not run in these cases.

5 Performance & Discussion

Once the program starts to run it can take some time. The most time intensive parts are generating the Hough transform and searching for the planes. Generating the Hough transform is the most computationally complex because after the array is generated, the program will also scan each line to find those that have more than a certain number of pixels and will also condense those lines down to single segments. Thus, each line in the Hough transform is either empty or a line in the image. This drastically speeds up the search in the image, needing to generally search about 700 lines versus about 200,000 lines that are possibilities with the Hough transform. This method was unable to accurately detect any boxes; however, it did not completely fail. Often it kept part of the box and part of the background as shown below in figure 5. This is the same starting image as figure 1 and was the most successful.

A picture containing object

Description automatically generated

Figure 3: After condensing lines

A close up of a logo

Description automatically generated

Figure 4: Final product of the process.

6 Discussion

The methods used here are very rough and don’t provide a consistent or efficient method for detecting boxes in images. The main issue for this method was that what was defined as a box can also be something else. There was no difference between a rectangle as a plane and a trapezoid or kite or another shape with 4 sides. The only constraint on the plane was several sides that made a cycle and combining the planes has a similar issue of just needing to form a cycle, however this produces less edge cases as the main difference is the relative sizes of the planes. By the methods described the following object would be considered a box, even though this is clearly not a box.

Figure 5: A potential false positive

It is obvious that this object is not a box and is probably not a valid three-dimensional object. Another issue is the large complexity and time it would take for the method to run. While not every line in the Hough transform is kept, on average more than 2000 lines are kept while takes about 5 minutes for the basic algorithm implemented here to scan for dead ends. Adding on how long a depth first search would be and then combining all the planes together would also add significant time. Another issue with this method as demonstrated in the final output is that several lines stacked on top of one another can make cycles with each other which causes a problem. This is more a problem with the implementation of the condensing lines algorithm than the search algorithm but is a problem. Without condensing the lines, each segment is generally only part of an edge as the edges aren’t at perfect degrees, so the Hough transform will break up the edges into multiple parts.

7 Conclusion

This program is tasked with finding a cardboard box in an image containing several objects and varied backgrounds. The method illustrated here is to smooth the image, make an edge image, generate a Hough transform of the image, search for cycles in the image, and then combine the cycles to make an object. The method laid out here works for some scenarios but in general is a poor strategy on its own as the constraints fit too many objects.

8 References

R. Stefanelli and A. Rosenfeld. 1971. Some Parallel Thinning Algorithms for Digital Pictures. J. ACM 18, 2 (April 1971), 255-264. DOI=http://dx.doi.org/10.1145/321637.321646

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00