Extracting and Solving Mazes in Images

91.423/523 Computer Vision Final Project, UMass Lowell

Anthony Salani

asalani@cs.uml.edu

Abstract

In this paper I detail a method for extracting undirected graphs from images containing distinct paths. This method leverages several techniques, such as image segmentation, color quantization, the formation of topological skeletons, and contour detection to create a series of line segments used to form an undirected graph. Once this graph is obtained, it is trivial to solve it for any two points on the graph and return the resulting path through the image. This method of path extraction can be applied to several types of images, particularly mazes, to some degree of success.

1. Introduction

The method of path extraction described in this document attempts to extract information pertaining to valid paths through an image. A path is considered anything a human would reason as a "walkable," or traversable, surface. For example, the spaces between the walls in mazes, the colored lines in a subway diagram, or the partition of road between the dotted yellow line of the median and the sidewalk, are all considered traversable surfaces. While it is possible to generalize the method described here to many different kinds of images containing paths, it is currently designed to solve the subset of image containing paths that are considered mazes, such as the ones found inside of a children's puzzle book, where paths and walls are represented using two or more relatively discrete colors, and there is almost always guaranteed to be a finish.

While the particular class of image we have chosen to solve is not in urgent need of having valid paths extracted from it, this approach could be generalized and be applied to other graphical depictions of areas. For example, the interiors of most shopping centers lack representations of their floor plan that is easily searched by a computer via methods such as A*. Mapping these areas in such a way would allow for customers to more efficiently find what they are looking for, without the chance that they might become discouraged and leave before buying anything. This is just one of many possible applications of a reliable form of path extraction.

The method of path extraction described in this paper is an exercise in image processing. First, k-means image segmentation was used as a form of color quantization. The resulting image was then made binary, where one set of pixels represented walls and the other set represented spaces between walls. After this, a skeleton of the image was created. This skeleton is then used to find the paths through the image, which are then made into a graph and can now be solved by any informed search algorithm.

2. Background

Seemingly little research has been done into the topic of path extraction. Whatever research has been done, has been mostly limited to blog posts and answers to questions of help forums such as Stack Overflow. Furthermore, most of these solutions are not very robust and work on only a very small class of mazes.

For example, one popular method of solving images is to take an image, make it binary, and then execute a search algorithm on the per-pixel level. This approach will only work well on images that are easily made binary, and images that are incredibly small. Larger images would take an incredible amount of time. Furthermore, images that require lots of data to represent complicated features such as curves would not be able to be solved by this method.

Another popular approach is to split the maze into two walls, and then perform a series of dilations, erosions and clever arithmetic to extract a path from the start of the maze to the end of the maze. This method is only works on classes of mazes that have their exits and entrances located on the border of the maze, does not yield a graph or any other easily searched data structure, and cannot find paths between any two arbitrary points in the maze.

This method builds primarily off of the process of finding topological skeletons of images[1],

3. Approach

The approach to solving this problem, as described above, is a series of image processing techniques followed by the application of a search algorithm. The image processing pipeline starts by quantizing the colors in an image, then forming a topological skeleton, then extracting edges. Once this has been done, the graph has been extracted and can be searched with A*.

3.1. Color Quantization

The step of color quantization aims to segment the walls of the maze from the paths between the walls. Color quantizations aims to decrease the number of colors used in an image. There are many different ways to perform color quantization – some range from just putting every color in the image into one of many bins evenly distributed across the RGB color space. Others rely on more complicated techniques like k-means image clustering.

In this particular application, binning colors evenly across the RGB spectrum was a non-ideal method of quantizing colors. In images where the color of the walls and spaces between the walls are distinct but similar (for example, yellow gaps and darker yellow walls), there is a chance that the walls and paths will be merged and the image will be unable to be solved correctly. This is why k-means image clustering was used. The image was clustered into 2 colors that were considered to make up the majority of the image. The resulting image was then made binary and had line cleaning applied to it.

3.2. Line Cleaning

The process of line cleaning is used frequently as a means to correct the results obtained from approach steps 3.1 and 3.3. Given that mazes are characterized by long, contiguous lines, it is a safe assumption to make that any gaps of a couple of pixels are the results of any number of possible problems, and are not intended features of the image. For example, the camera used to take a picture of the maze could be of a very low quality, lighting conditions were imperfect, or there were gaps left behind in the image after forming a topological skeleton.

Line cleaning is done by retrieving line segments from a probabilistic Hough transform. The line segments that are retrieved have a very restrictive upper and lower threshold on line length. Because the imperfections are often only a couple of pixels wide and frequently occur on thin lines, minimum line length and maximum line length are clipped to very small values. The number of votes required to consider a line a part of the image is also incredibly low. These numbers are not the result of any careful mathematical analysis, but rather from a process of trial and error on different binary images. Also, line cleaning after different steps used different parameters.

Once the line segments are obtained, they are drawn on the image. This process is repeated several times. The end result of the transform is almost identical to the image prior to the transform, but the small pixel-sized imperfections have been filled in, allowing the next step in the pipeline to execute more optimally.

3.3. Formation of a Topological Skeleton

A topological skeleton is a binary image that represents another binary image, but each region is shrunken down so it is represented as a series of pixel-wide lines that approximately bisect the original area. This process is also described as “thinning” the original image, which is much more descriptive of the end result.

There are several methods of forming a topological skeleton. The method chosen in the described implementation of path extraction is simply a set of morphological transforms on the original binary image. The algorithm used is as follows:

```
elem = 3x3 ellipse kernel
img = original binary image
skel = empty image

while img is not empty:
  temp = copy of img
  dilate temp by kernel
  erode temp by kernel
  temp = temp & img
  skeleton = temp | skeleton
  erode img by kernel
```

The only adjustment that needed to be made to the original algorithm is that the input image had to be inverted, because there is an assumption being made in the program that the walls are a darker color than the paths.

3.4. Edge Extraction and Graph Building

Once the skeleton of the image has been obtained, the next step is to construct the graph. Edges are extracted using contour detection, and then put into bins even distributed across the image. In this particular implementation, a bin size of 4 was ideal, as it joined together the most line segments without creating a path that overlapped a non-traversable surface, such as a wall. Bins are formed using the following formula:

$$f(x,k) = x - (x \bmod k) + (k/2)$$

where x is a point along either the x or y axis, and k is the bin size.

3.5. Application of a Search Algorithm

Once the graph has been built, the search algorithm can be applied. The method implemented uses A*, but a different search algorithm could easily be used.

Before the search algorithm is applied, a pair of start and end points are obtained from the user of the application. Because the user is not guaranteed to have clicked on a node in the graph, the closest neighbor is found for each point clicked. These new coordinates are treated as the new start and end points for solving the maze.

3.6. Submitted Code

Table 1 Summary of Submitted Code

Filename	Summary	Author
main.py	Program driver, applies image processing techniques described	Anthony Salani
maze.py	Contains code for sections 3.1 through 3.4	Anthony Salani
graphs.py	Undirected graph data structure	Anthony Salani
util.py	Code for resizing images intelligently and converting lists of points to transitions	Anthony Salani

4. Dataset

The dataset used consists of about 30 mazes that attempt to span the spectrum of possible mazes. We have identified several possible axes along which a maze can exist:

- colorful vs. black and white
- curved vs. angled
- photographed vs. computer generated
- abstract vs. physical (hedge or corn mazes)
- flat vs. overlapping
- large vs. small
- thick walls vs. thin walls
- thick paths vs. thin paths
- dark walls vs. dark paths

There are also a collection of images that do not depict mazes, but instead depict different objects or scenes that contain paths that should be recognizable.

5. Evaluation

The method described in this document works well on most of the abstract class of mazes. Even abstract mazes that are photographs of pieces of paper construct graphs that are reflective of the maze. However, it is less common to find a maze that is entirely accurate. Most mazes generate graphs that are not complete. A graph that is not complete often has one or two holes in the graph, often at large or otherwise open intersections. Due to the nature and location of the gaps, this often renders mazes with gaps to be unsolvable in the manner that author intended.

The class of mazes that work the best are mazes that are abstract, computer generated, flat, large, and with walls that are slightly smaller and darker than the paths that they demarcate. Whether or not the image is purely binary to start or is colorful, or if is curved, angle or both are entirely inconsequential to the success of the path extraction.

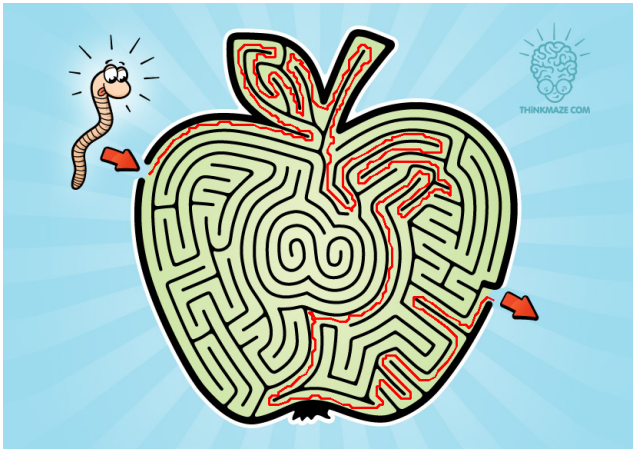

Figure N: A maze that had a complete path extracted from it.

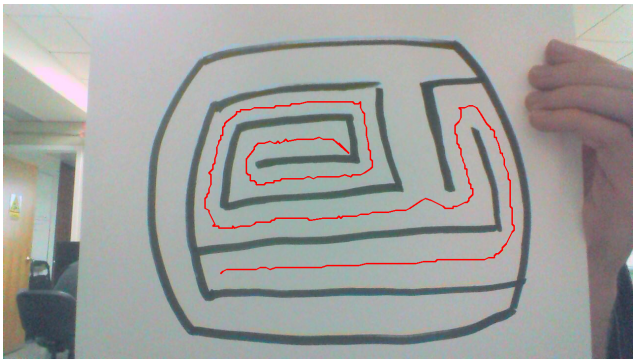

Figure N2: Maze with incomplete graph still capable of pathfinding.

The method described in this document cannot handle three classes of mazes. Those that are overlapping, with disproportionately small paths or walls, or mazes where the paths are darker than the walls are all not solved by this method. Overlapping is the hardest to overcome. See Figure M for an example of a maze with overlapping paths. Most mazes that contain segments where one path overlaps another are often hard to solved, even for people. Every maze has a different syntax it used to describe such occurrences. Developing a proper algorithm to handle this would be an exercise in machine learning, and would potentially require substantial changes to other parts of the method as well.

Figure M: A generated skeleton with holes.

Figure M2: Attempted pathfinding from two arbitrary locations on an incomplete graph.

The weakest link in the method is the process of thinning the image down to a one pixel line. The morphological transform approach to image thinning often leaves large gaps, particularly in areas with a large number of intersecting paths. These problems could potentially be alleviated by choosing a better thinning algorithm. In particular, the Zhang-Suen thinning algorithm seems as if it would be a good fit [1], and would not leave large gaps. Alternatively, the thinning step could be skipped in favor of generating navigation meshes from the color quantization step [2]. These alternative approaches were considered but not acted upon, as we lacked the time to properly implement either. Also, the speed of Python, which was used for the implementation of the method, left much to be desired in regards to performing many operations on OpenCV matrices. This was the reason the Zhang-Suen approach to thinning was not used.

6. Conclusion

Extracting paths from images is not an impossible task. It has very tangible benefits in the domain of solving mazes represented as images over previous work done. For example, the method described above takes several seconds to process the image and extract a graph, and takes almost no time at all to perform an A* search on the resulting graph. Meanwhile, a pixel-based search approach would take slightly less time to process the image, but much longer to perform subsequent searches. The morphological transform approach also has several key weakness that path extraction simply does not have. While the results here are not amazing, and there is high chance of not being able to solve the maze, there is the potential that this could easily be remedied through the use of navigation meshes over undirected graphs, or by using a different thinning algorithm.

7. Team Roles

I, Anthony Salani, have done all of the writing, data collection and programming. I have written over 2,000 lines of Python, of which close to 400 are still actively being used in the application.

References

[1] L. Lam, S. W. Lee, and C. Y. Suen. Thinning methodologies – a comprehensive survey. Supplied as additional material lam-lee-survey.pdf.

[2] M. Kallmann. Shortest paths with abritrary clearance from navigation meshes. Supplied as additional material 10-sca-tripath.pdf