# **DPS920/CVI620**

# **Project – Final Report Submission**

**This is a GROUP Submission, NOT an Individual Submission. ALL team members must submit the final work.**

***Please paste the resulting images, answers, … in this document.***

1. **Project Title.** What is the title of your project?

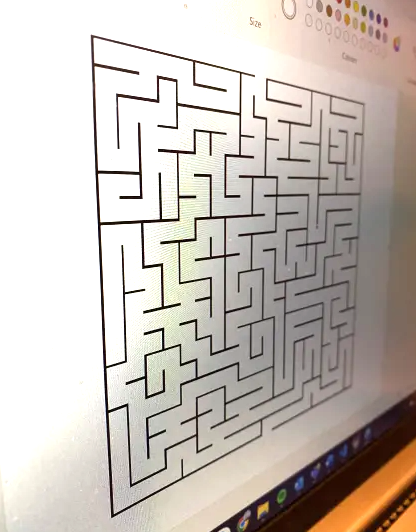
Maze Solver

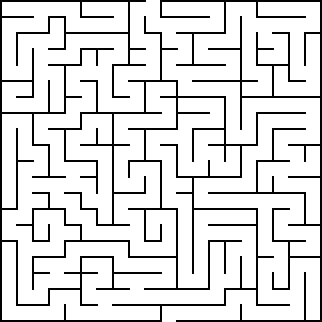
1. What is the **problem** that you are trying to solve? (Please explain in a paragraph.)

A maze is a two-dimensional structure divided into rows and columns representing cells of equal size. All cells are filled with a wall and empty spaces that represent a possible path to the endpoint. Only two cells are known, one as a starting point and one as an endpoint. Only open squares of the maze can be passed through. If a wall is meet when trying to go through the maze, a different direction has to be taken. Using computer vision techniques to detect the walls of a maze, the goal is to make a computer program to solve a maze by implementing algorithms and techniques learnt from OpenCV. The program will use video capture as input and output the same maze with lines drawn to show the solution.

1. **Dataset.** What is the dataset you used? How many images/videos does this dataset have? Please paste a few images/snapshots here.

The dataset used in this project will be a video capture capturing a maze as input and also a picture of a maze. The images below represent the test data set. The first image was captured using video capture of a maze from a webcam. The second image is a generated picture of a maze.





1. **Ground Truth.** If applicable, show samples of Ground Truth (gt) and how it was collected and/or labelled.

Ground truth was obtained simply by manually solving the mazes, and then comparing the program output to the manually solved version.

1. **Training vs Test vs Validation.**  Explain how you divided your dataset into training / test / and validation sets.

Because no machine-learning was involved in this project, no training dataset was required. All of the dataset was used for testing.

1. **Previous Works:** Explain what has been done by others and which one (if any) you are using (code or idea) as a starting point. You need to include all references at the end of this document. (Please explain in a paragraph.)

Several different maze-solving methods have been implemented by other developers in the past. Some work by converting the maze into a graph, and then traversing the graph using depth-first search or breadth-first search algorithms. The maze-solving method we chose works by detecting the maze contours, then using morphological operations on the resulting contours, such as dilation and erosion, to reveal the solution for the maze. As a starting point we used an existing implementation from <https://github.com/ahnv/MazeSolver>.

1. **Contribution and Method.** Briefly explain the method you used in this project to tackle the problem in Part 2. What are your contributions? What steps did you add to previous works? Why did you choose these steps? Detailed explanations for any team member must be included in individual reports. (Please explain in 2 paragraphs.)

The method we chose to solve mazes involves finding contours, and applying morphological operations to the resulting contours. Once the contours were found, they were stored in a binary format. You can think of the solution of a maze as a line that splits the walls of the maze into two parts. We can use morphological operations on one of these parts to reveal the maze solution. First, we dilate the contour, which expands it. The border of the shape that results from the dilation shows the solution. So, to extract the border, we erode the dilated contour, and subtract the eroded contour from the dilated contour, and the result shows the maze solution.

Our contribution to this previous work was to try and apply the same maze-solving technique to real-life images, which may have distortions or other imperfections. This first involved creating a more complex preprocessing pipeline that can effectively extract the walls and paths of the maze, and convert the input images into a binary format to be used by the maze-solving algorithm, without losing any important details in the image. We also had to take the ideas/techniques from the original code, and re-implement them into functions that can be called from the main maze-solving pipeline.

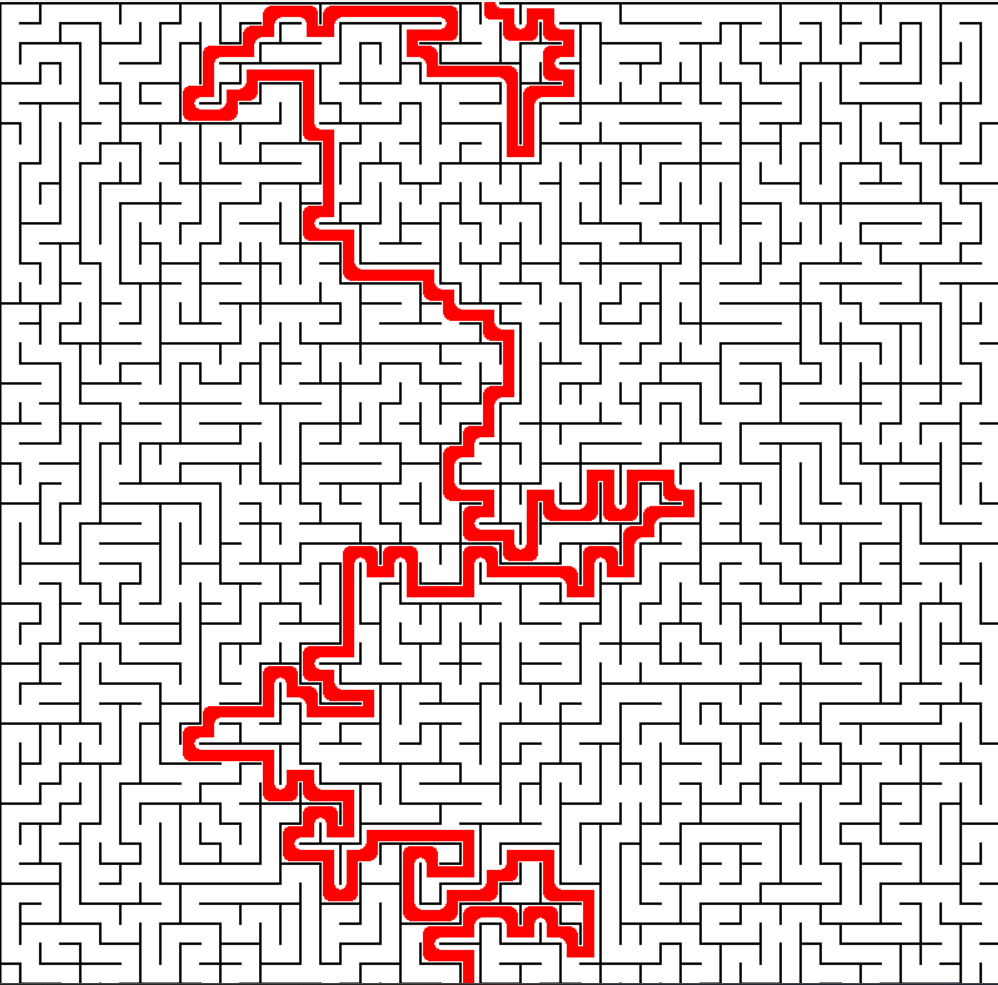
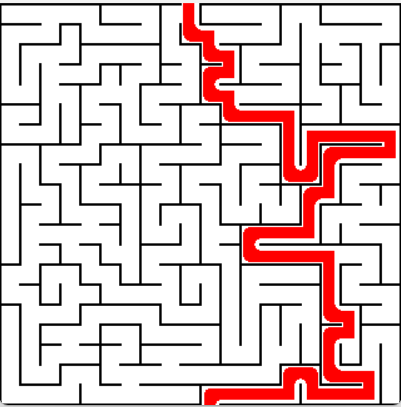
1. Explain whether you achieved what you expected with this method? If yes, why, if no, why not. (Please explain in a paragraph.)

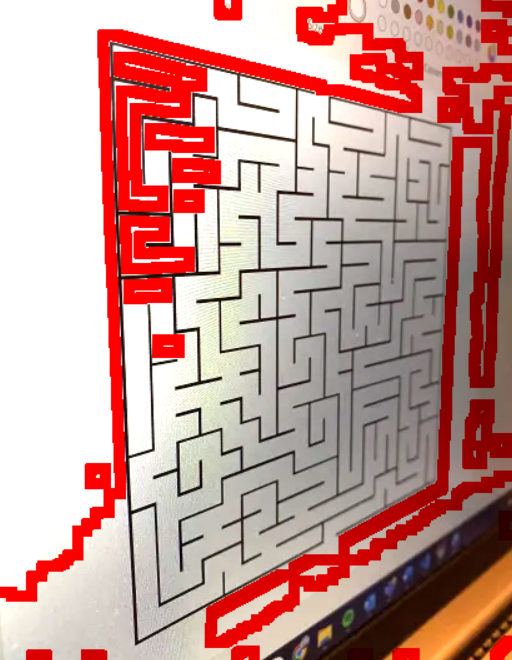
Throughout this method, we were not able to achieve everything that we expected. This was because the method we chose was very sensitive to any imperfections in the input image, as they would have a significant effect on the resulting contours of the maze walls. Although the preprocessing pipeline was able to effectively extract the maze from the image, some errors would still remain.

1. Evaluation.
   1. Please provide a performance evaluation of your method (quantitative and qualitative, if applicable). Note that you evaluate the method you proposed (Part 7) on the dataset you chose (Part 3) for the problem in part 2.

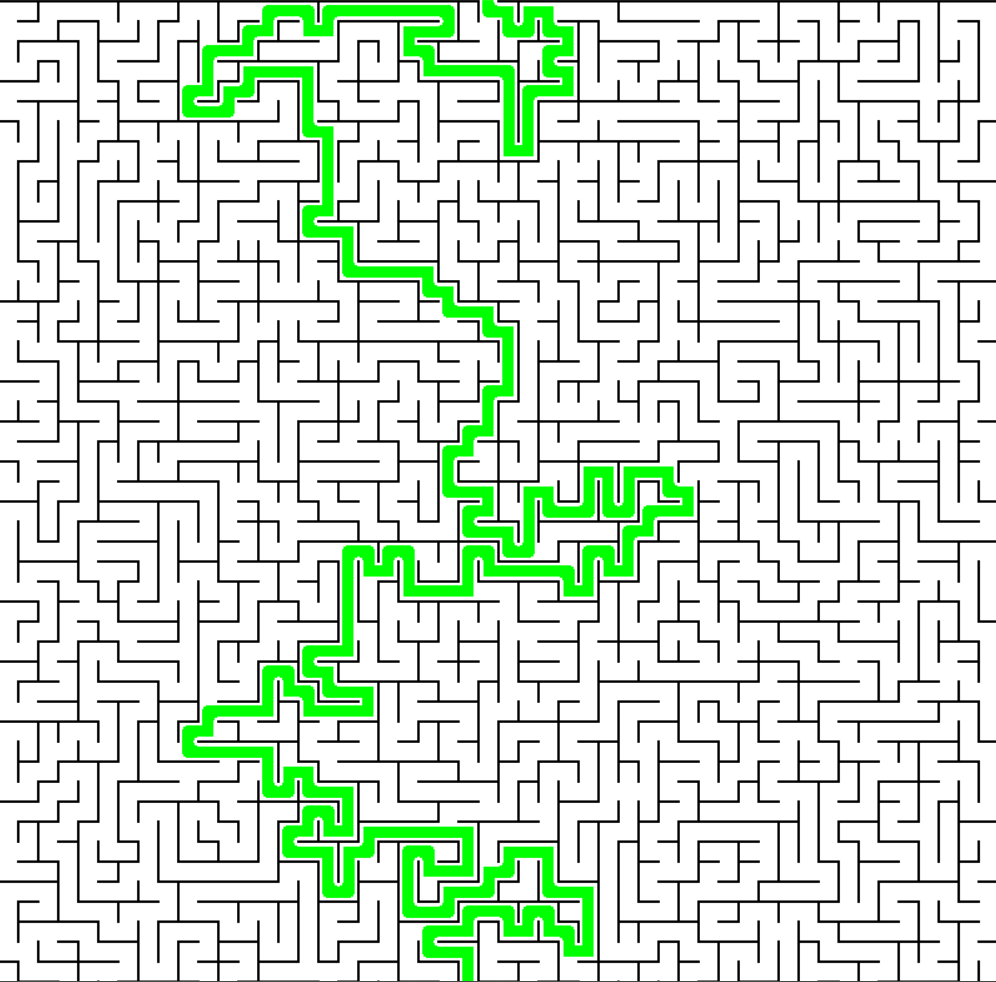
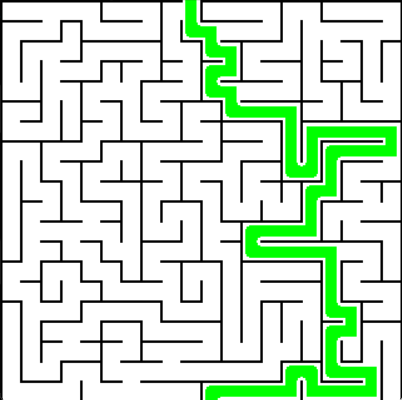
Due to the chosen maze-solving method being very sensitive to imperfections and distortions in the input image, our code did not perform very well unless the input image was perfect or near-perfect. The mazes in our dataset that were computer generated (and not real-life photographs) were able to be solved successfully.

* 1. Please provide sample results (qualitative and quantitative, if applicable).





* 1. Comparison with previous works (Part 6) needs to be included in this part as well.



1. **Code submission**. Please submit your code and required files (or a link to a shared folder) and specify instructions on how to run your code.

<https://github.com/BenLee8602/maze-solver>

see readme for instructions

1. **A list of references.** Please do \*\*\*NOT\*\*\* just include the link! Include the title, author, year, etc. See different citation styles such as APA, MLA, …: <https://pitt.libguides.com/citationhelp> For IEEE you can find more examples here: <http://libguides.murdoch.edu.au/IEEE/all>

GitHub, MazeSolver by Abhinav Dhiman

<https://github.com/ahnv/MazeSolver>

GeeksForGeeks, OpenCV Perspective Transformation

<https://www.geeksforgeeks.org/perspective-transformation-python-opencv/>

GeeksForGeeks, OpenCV Shape Detection

<https://www.geeksforgeeks.org/how-to-detect-shapes-in-images-in-python-using-opencv/>

GeeksForGeeks, Adaptive Thresholding

<https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-2-adaptive-thresholding/>

1. Add this declaration to this report:

We (group 7), Kirk Parillon and Ben Lee, declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

1. Specify what each member has done towards the completion of this work:

|  | Name | Task(s) |
| --- | --- | --- |
| 1 | Kirk Parillon | 1,2,3 |
| 2 | Ben Lee | 4-9 |
| 3 |  |  |