Ideal Pathfinding

ABDIRAHMAN YASSIN AND GAYMER BARRIOS
Wilfrid Laurier University, Physics and Computer Department Waterloo, CP468, Ontario, Canada N2L 3C5

Abstract

An image is provided as input. The pathfinding is performed by a general search algorithm which applies different heuristics functions to generate output, a png file, showing the path from the starting point to the goal. A grayscale map of Wilfrid Laurier University, Waterloo campus, is used to show the path between two buildings. This path could be the shortests, the least costly, or a combination of both. The algorithm considers a darker pixel to be the cheapest. The program was developed using Python.

Index Terms - pathfinding, heuristics, A*

1. Introduction

1.1 Pathfinding

Pathfinding is the plotting of the shortest route between two points. "Pathfinding is closely related to the shortest path problem which examines how to identify the path that best meets some criteria (shortest, cheapest, fastest, etc) between two points in a large network. In this project, we will find the shortest path between two points given a map, starting position and a goal [1]".

1.1 Heuristics

Heuristic is any approach to solve a problem that's believed to be optimal. It's not guaranteed to be perfect or rational but instead sufficient for reaching an optimal goal. Also, A heuristic function or simply a heuristic, is a function that ranks alternatives in search algorithm at each branching step based on the available information to decide which branch to follow. For example, it may approximate the exact solution. In this project, we employ heuristics to optimize path finding.

1.3 Challenges faced during development

We faced several challenges while trying to figure out what to do with what we've learned so far in class. The main challenge was coming up with the idea of the project, then we realized that we could solve one of the problems we face when walking between different buildings here at Laurier. We always had a discussion on which path is nearer. In one of our discussions, we got an idea on how to determine the shortest path to take between different buildings here at Laurier. After several days of research, we found out that there's a symposium on Educational Advances in Artificial intelligence [2] which guides and gives students projects on AI and helps them to complete projects giving suggestions on how to approach solving the problems in that project.

2. Algorithm explanations

2.1 General Search

In our project, we use a general search as a base to implement greedy best first search, A* search, Beam search and Human search. A general search will visit each node of the search space. For each node it visits, it will keep track of the nodes that can be explored, then by picking a next node from the current reachable nodes, it can find the goal. Our project then adds an evaluation function made up of a heuristic function and cost function. By modifying these functions, the general search algorithm can be used to show the differences in pathfinding between different heuristics.

2.2 Greedy Best First Search

An informed search algorithm that expands the node that appears to be closest to the goal. In this case, the evaluation function equals the estimated cost from the current node to the goal. If we let n be the current node, then F(n) = H(n) give us the equation for this heuristic. This heuristic does not care about the cost of the path. It is considered to be an incomplete heuristic because it can get stuck in loops and is not Optimal since it is not guaranteed to return the shortests path to the goal.

2.3 A* Search

The second informed search in this project is A*. A* improves on Greedy Best First Search heuristic by taking into consideration not only an estimated cost to reach the goal but also the cost so far to get to the current node.This give us an evaluation function F(n) = H(n) + G(n) In contrast with Greedy Best First Search, A* will not get stuck in loops and it will find an Optimal path to the goal.

2.4 Beam Search

“Beam search is a heuristic algorithm that explores a graph by expanding the most promising node in a limited set [3]”. This search is an optimization of best-first search. In our project, the beam-search tries to detect obstacles and chooses the most efficient and least costly path given an obstacle. Instead of generating multiple paths like best-first search, it explicitly tells you which path is the least costly by implementing the heuristics defined.Also, beam search uses beam width which is a parameter in the beam search algorithm which determines how many of the best partial solutions to evaluate.This reduces the search space by eliminating candidates to reduce the memory and time requirements, this is achieved through pruning. One of the weaknesses of the beam search is that it gets too close to the object.Also, another weaknesses is that when you prune, you might eliminate good paths, or paths that might be more efficient than the ones used in the search.

3. The Program

The main window of the program allows the user to select the heuristic to use and the image file to use as input. The following refers to Fig.1.

3.1 The Number of Iterations

This argument controls the number of iterations the program will run on each input image file. Each iteration corresponds to a path found and drawn on the output image file.

3.2 Updating the Terrain

This argument controls how dark a pixel gets after each iteration. The minimum of this number is 1 and the maximum is 255. When the number is low, a path would take more iterations for it to become visible, while making this number higher, would make paths to become darker after fewer iterations.

3.3 Selecting Heuristics

This section allows the user to select one or more heuristics to be used in the input image file(s). For each image file and heuristic, there will be an output image file.

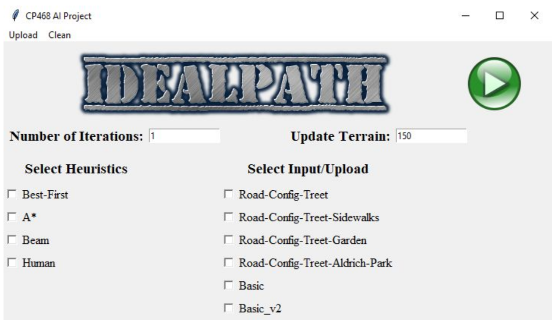
3.4 Select Input/Upload

This section allows the user to select one or more input image files for the program to use. The input image files are .bmp format.

It must contain the following:

- A blue pixel representing the starting pixel
- A green pixel representing the goal pixel
- Red pixels to outline obstacles

There could be multiple blue and green pixels present in an image file. The program randomly chooses one starting point and one ending goal if they are available. The default image files can be selected from the GUI, but the user can use the menu/upload option to select any input image.



Number of Iterations: 1
Update Terrain: 150

Select Heuristics
- ☐ Best-First
- ☐ A*
- ☐ Beam
- ☐ Human

Select Input/Upload
- ☐ Road-Config-Treet
- ☐ Road-Config-Treet-Sidewalks
- ☐ Road-Config-Treet-Garden
- ☐ Road-Config-Treet-Aldrich-Park
- ☐ Basic
- ☐ Basic_v2

Fig. 1. The GUI of the Project

3.5 The console and output images

While the program is running, the console shows the selected input image files and the heuristics selected. For each path that the program finds, it outputs the length of the path. The output images are .png output files that reflect 1 or more paths that were found between the Starting and Goal points.

The GUI also allows the user to delete all previous output files.

4. Output during testing

4.1 The input images

The following .bmp image files are used as input to test the algorithms in a smaller problem compared to the Wilfrid Laurier Campus.

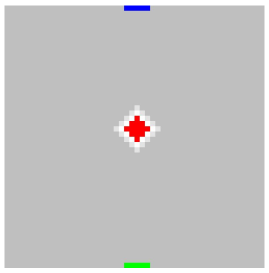


Fig. 2. A tree representation

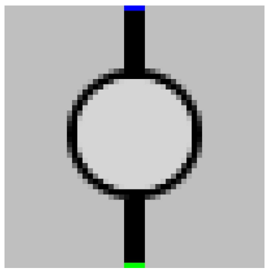


Fig. 3. A garden representation

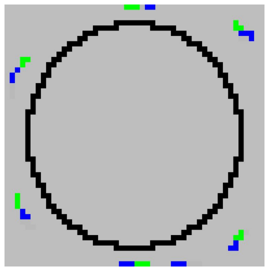


Fig. 4. A garden with multiple destinations

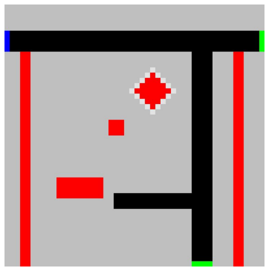


Fig. 5. A location with obstacles and sidewalks

4.2 Greedy best first search output images

On Fig.6 and Fig.7, we can see that using greedy best results in a path straight to the goal. Fig.7 has a circle which represents a sidewalk with a lower cost, but this greedy heuristic just ignores this lower cost path and cuts straight through to the goal.

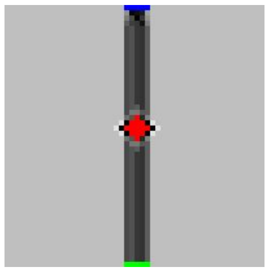


Fig. 6. Output from greedy best first

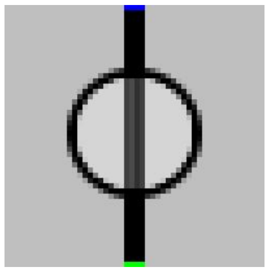


Fig. 7. Output from greedy best first

For Fig. 8, this heuristic makes a lot of straight lines path, once again ignoring the cheapest path around the black circle. Lastly, Fig. 9 has a noticeable path from the blue pixels to the green bottom right pixel. The paths to the green pixel on the top right corner, are not visible due to the black pixels.

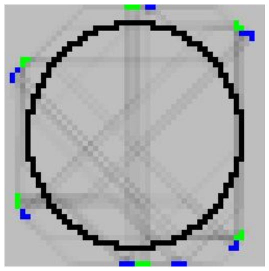


Fig. 8. Output from greedy best first

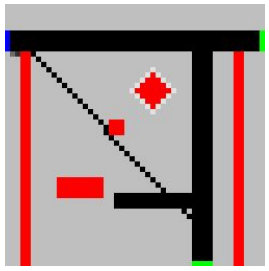


Fig. 9. Output from greedy best first

4.3 A*search output images

From this set of output, the most interesting ones are Fig.10 and Fig.11. When we compare Fig.10 and Fig.6, we can see that A* consistently stayed on a single path, as a result this path is the shortest and least costly. In Fig.11, A* focused on making a least costly path, as a result, there are no paths cutting through the circle and all paths were made on the black pixels which represent sidewalks. The other outputs are similar to the other heuristics and are put here for comparison.

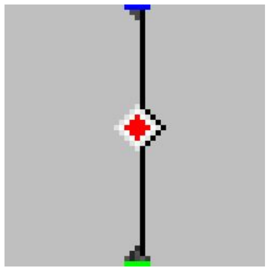


Fig. 10. Output from A*

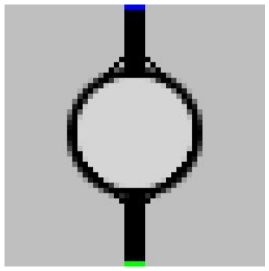


Fig. 11. Output from A*

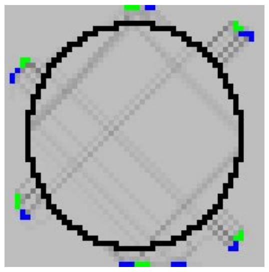


Fig. 12. Output from A*

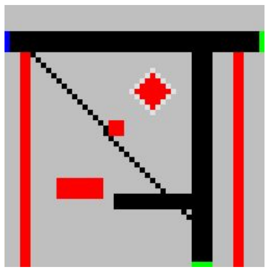


Fig. 13. Output from A*

4.4 Beam search output images

In Fig. 14, the starting point is given as the blue color and the goal is the green color. When beam search is applied, a path is generated avoiding the obstacle between the initial point and the goal on the map.

For Fig. 16, the beam search generates multiple paths since there are multiple starting points and goals. In addition, there are no obstacles on the input map, hence multiple paths.

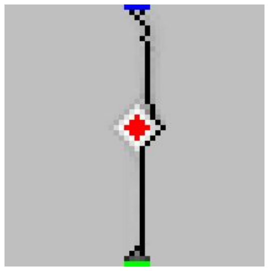


Fig. 14. Output from Beam

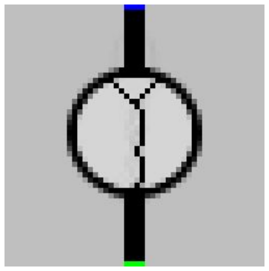


Fig. 15. Output from Beam

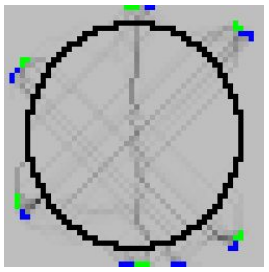


Fig. 16. Output from Beam

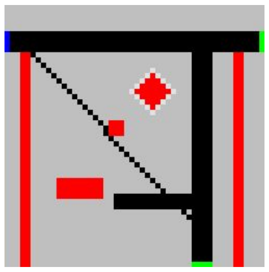


Fig. 17. Output from Beam

4.5 Human search output images

This search uses heuristics to avoid/ keep a distance from the object and then get back to the path. This is how a human would get from point A to point B, humans will keep a distance away from obstacles as shown in Fig. 18 and Fig. 21.

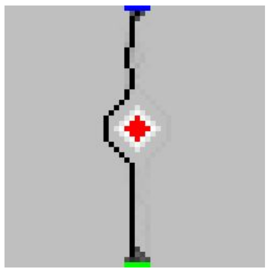


Fig. 18. Output from Human

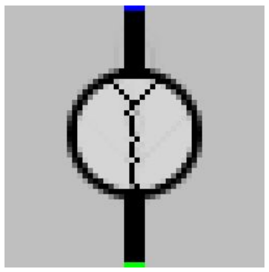


Fig. 19. Output from Human

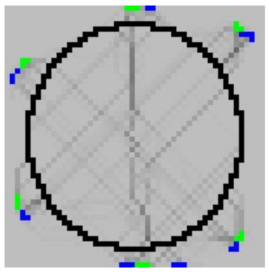


Fig. 20. Output from Human

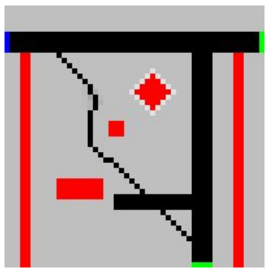


Fig. 21. Output from Human

5. Results from the WLU Campus

The goal Find out what are the shortests path to move from building A to building B in the Wilfrid Laurier University Campus.

Using snazzy maps, we were able to obtain a labeless, grayscale map of the campus. The building outlines are red pixels to represent impassable structures

The grayscale is used to represent cost. White is more costly than black. Sidewalks have a gray colour to represent cheaper cost.

At the moment, the only heuristic we can use to find paths using this image file is greedy best first search. The other heuristics such as A*, Beam and Human, take too long since the search space is huge. Fig.22 shows two paths that were found from the science building to the CP468 Class in the Schlegel building. The greedy best first search path stays close to the outline of buildings.

If A* could be applied to this image, we would expect paths to be made using sidewalks, since they have a cheaper cost. The Human search would behave like A* while keeping a distance from the buildings.

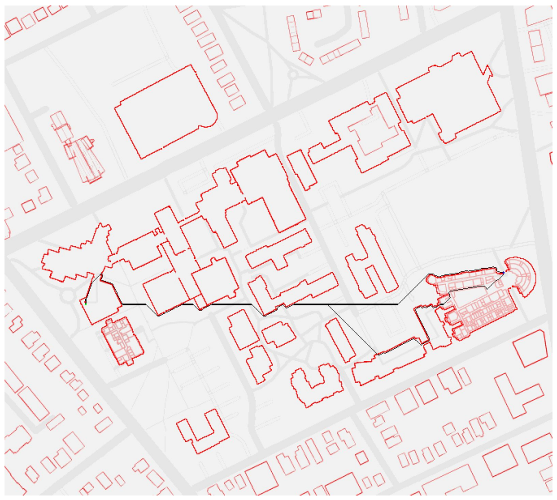


Fig. 22. Wilfrid Laurier Waterloo Campus

6. Project weakness and areas of improvement

At the moment, each pixel is used as a node so the size of the image makes the search of A* slower. It would be more efficient to modify the graph to have waypoints in order to increase the efficiency. One way to achieve that is to have nodes whenever we make a turn.

Getting a more accurate representation of the WLU campus by including elevations, more obstacles such as trees so that we can get a more accurate pathfinding.

7. Conclusion

Good pathfinding algorithms should give us the shortest and the least costly path to the goal. Obstacles and different maps with different starting and goal positions were tested in the program to experiment the effectiveness of the heuristics used in the project.

As per the output of the program, the results were consistent with our expectations. However, the program takes more time when a more complicated/ large maps are passed to it. Therefore, the performance of the program can be improved to make it more efficient.

REFERENCES

[1] Wikipedia, "Pathfinding".[online], available: https://en.wikipedia.org/wiki/Pathfinding. [Accessed 31 Jul. 2019].

[2] Educational Advances in Artificial Intelligence "Project Archive". [online], available: http://modelai.gettysburg.edu/ [Accessed June 23.2019].

[3] Wikipedia, "Pathfinding".[online], available: https://en.wikipedia.org/wiki/Beam_search. . [Accessed 16 Jul. 2019.].