Exploring a Raw Unprocessed Campus Map with Image Processing and Path-Finding Algorithms

*N*ithin Kumar Mundrathi
200775601
Supervisor: Dr. Julian Hough
MSc Artificial Intelligence

***Abstract*—Path-Finding is a important area to research which focuses on find the optimal path.This paper covers the study and implementation of Path Finding Algorithms on a Raw Unprocessed University Campus Map and this map is cleaned using Image processing Algorithms. The Map obtained is also segmented using Image Segmentation Algorithms into walkable segments and Non-walkable Segments. The Experimentation study is done using many path finding Algorithms and the results are obtained by comparing path of each Algorithm with a reference gold standard path and a Metric called C Metric is used. This Metric compares two different paths using the reference path's nearest neighbour's nodes as Gold Standard path.The other metrics are also chosen to compare the algorithms based on Speed, memory occupied and number of common nodes with the reference path.The study is done on Static map of constant obstacle density and single Agent environment rather than Multi agent-dynamic environment.**

***Index Terms*—Path-Finding, Image Segmentation, Image Processing**

I. Introduction

Humans take a look at an Image and instantly know what are the objects in the image.But the Image captures the 3-D world and maps the 3-D world into 2-D Image which results in loosing a little spatial information. Similarly,If a user looks at a map and the user knows where he/she is and then decide where to go. But the path to follow from origin to destination is a complex task due to the obstacles placed in the path. The obstacles maybe buildings,houses, bridges. The user usually takes help of the Sign Boards. The path from origin to destination could either follow a longer path or shorter path which depends on obstacle density and distribution. Many fast,accurate and state of the algorithms are developed to find the shortest,fastest path for a human based on the origin and destination.Along with the recent developments in Artificial Intelligence, Google Maps was improved using KD-Tree and machine Learning Algorithms.These algorithms are currently used in GPS,robotics, games.Path finding algorithms is one of the most important cores of majority of video games.The algorithm deals with finding the shortest route between the origin and destination.The algorithm has to be designed to find adequate path to reach the destination from the origin by avoiding the blockades and also finding the cost-effective path.

The Performance of these algorithms is affected by several factors like the problem size, length of path, the distribution of obstacles in the map.Path Finding can be done in two different environments-dynamic and static environments using Single Agent and Multi-agent's. Currently this paper is based on Single Agent environment in a Static Environment trying to find the shortest path between origin and destination.These path finding algorithms are generally implemented over maps which are generated in a Grid-based graph. There are also other ways to represent the map by dividing the whole grid into smaller clusters (or) smaller grids. Some other ways to represent the grids are hexagonal grid, Triangular grid, Navigation grid, Way points, tile grids.

We generate a Grid like Map based on the Given Single Image.This Image is preprocessed using Computer vision techniques and segmented such that the walkable paths are segmented in White and blockade's/non-walkable paths are segmented in Black.The Image segmentation is done using traditional computer vision techniques like Image thresholding, Dilation,erosion. The numbers in the Image need to be extracted which is done using a pre-trained Machine learning algorithm.These numbers are mapped to a specific location in the Image.example– (24 is "library"). These Numbers in the Image are to be detected and also location at which this particular number is also present.

Once Location of the Numbers(Place) is extracted, then the input to the algorithm should be the Location of place's (origin and destination).A little data preprocessing needs to be done to find the nearest walkable pixel from Location.This nearest pixel from the building would be the input to the Algoirthm. Then, the output would be the optimal path between the origin and destination in the map.A Binary map of the Image is created, in which 1 indicate the path and 0 indicate the blockades in the Map and this map is extracted from the Image.

II. Literature Review

Dijkstra. A great deal of research has been done in the field of pathfinding algorithms which involves improving the algorithms, some introducing new heuristics to the existing algorithms. Several papers have been published comparing several algorithms to a single specific existing problem and also using a single algorithm (making improvements) by using various types of existing maps. [2] George Dantzig proposed the first formulation in 1955 to find the shortest path at a Conference paper which was later published in Operations Research(Dantzig 1957). Using this paper as reference Minty (1957) [3] proposed a network using a web of strings and

knots which solves the shortest path problem. Later, Many new algorithms which are fast and consume less memory space have been introduced since the publication of Dijkstra algorithm [4] in 1959.Later A* was first described in 1968 by Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute [1]. "For a map which is represented in a grid based system, Euclidean distance is precise compared to other distance measures" Soltani, A.R. et.al. [5] reported in their thesis that: in large scale maps represented in grid's, Dijkstra fails to find an optimal path whereas A* finds an optimal path using its target based heuristics. Souissi, O. et.al [6] discuss two main methods which require path planning in the map. These methods include cases it is necessary to model the environment. The environment is modelled using probabilistic models and statistical forward planning which helps to model the environment. These algorithms are class of computational algorithms that use a repeated random sampling and approximate a value. The other type of algorithms, it isn't necessary to model the environment but it involves weighting of the nodes. So, the nodes closer to the goal gets more weight and nodes closer to the obstacles or blockade gets less weight, so that the path could be ignored as they lead to deadend. Futhermore, Souissi, O. et.al gave a list of path finding algorithms classified into specific categories. D*, D* Lite and Lifelong planning A* comes under "Classic Dynamic" category. Under "Any-Angle movement", the algorithms which comes under this category are Theta*, Field D* and Incremental Phi*. In the "Moving target" category, categorize algorithms as Generalized Adaptive A*, Generalized Fringe-Retrieving A* and Tree-AA*. In the Final category, "Sub optimal" they place algorithms as HPA*, Anytime D* and Partial Refinement A*.

III. Methodology

A. *Data set and its Description*

The current data set used for thesis is a map of Queen Mary university of London. The Map is a bird's eye view of the university. The map is represented in various colours, numbers, symbols. The specific buildings in the map are described by specific number Id's and these Id's are mapped to specific name of the buildings. The Colour code represents different buildings like administrative building, Recreation area, building under Maintenance, Grass. The Symbols in the map represent Fitness Center, Staff Car Park,Bicycle Park, Smoking area.The areas covered with grass also need to be treated as walk able part if not the algorithms would treat the grass areas as obstacles and path obtained would not be the optimal path.This would also be the same case with cycling track and Bancroft Road which would be to treat them as walk able path.In the current thesis, we try to find a path from building to building based on the Index by which the respective buildings are represented.We Ignore the Regent's Canal and park left to the Regent's Canal.We also tend to Ignore the Mile End Hospital and the areas colour coded in gray-light and dark as they do not come under the university premises..Below is the Image which represents Map of Queen Mary University Of London.

Fig. 1. Map of Queen Mary University of London.

B. *Data Preprocessing and Data Cleaning*

*1) **Extracting the number Index's of the building**:* The Image preprocessing is done using traditional image processing algorithms and techniques. Initially, the number-indexes of the respective buildings in the Image are to be extracted and their respective locations of the buildings in the Image. We tend to ignore the color code of the buildings and symbols which represent other important key locations in the map like parking,cash machine, fitness center. The number-indexes are detected using contours in the image. We find the contours using Open CV's contour function and then classifying the blobs inside the contours. The blobs inside the contours are the number-indexes which needs to be classified by a Machine Learning Algorithm.

There is a open source SVM Model which has already been trained on numbers and this algorithm has been trained on many different sizes of Images which has the numbers.This SVM model has trained on MNIST data set and currently we could use this pre-trained algorithm.So, extract the regions around the contours which would be rectangular small patches and then classify the number inside the small patch using the pre-trained algorithm.For example- the small patches extracted are of 16x16 size which has the numbers inside them. So, the Final contours which are present inside the Image are of size 16x16 or 32x32 based on the size of the Original Image (we resize the contours such that it fits the Algorithm).We could try to choose large rectangular patches but this would lead to noise in the patches which would affect the output of the algorithm.

Once the numbers are classified, we represent each contour by their number and the location of the contour is given by the output of contour function.The center of the contour is treated as location of the number of Index's. In the end, the location of the all the buildings in the Map is stored in a dictionary with the value as the Location and key as the number-index. The number indexes are to be manually mapped to the respective building names and the location of building in the map. The output present after Data preprocessing are the Image which represents the map, the dictionary which maps the location, number-index and the name of the buildings.

*2) **Segmentation of Image**:* Image segmentation is one of the techniques used to locate the objects and their boundaries

in images. Many methods are introduced to perform segmentation based on low level elements in the Images like pixel intensity, textures, and pixel density. The Segmentation could also be done with many Image Segmentation Techniques like Watershed Algorithm, (Image Segmentation algorithms), Deep Learning techniques like using U-Net (which needs a labelled data set).Among all the Image segmentation algorithms, the most simpler Image segmentation called threshold segmentation is used in this thesis. So, threshold segmentation is categorized under region based segmentation(combine pixels with similar colour which forms a region). Each region is also separated from other neighbouring regions by boundary pixels. In this method, we find a threshold pixel value for the whole image and then label all the pixel's based on this threshold.The other types of Segmentation are Edge Based Segmentation, Segmentation using Clustering and Supervised Clustering (require a labelled data) [9].As, there are many colours in the Image, thresholding is done based in the colour codes. examples as the buildings are color coded with violet, red and orange. These buildings could be treated as obstacles in the map by converting the pixels with above color codes into black pixels. So, Obstacles are created in the map using the specific building color codes by Threshold Segmentation.

Once the Image Segmentation is done, we need to preprocess and clean the Image such that the walk able segments in the Image should be clearly segmented from the non-walk able fragments in the Image.The final output is that the Walk able segment in the Image are to be segmented in White Color and the Non-Walk able fragments in Black. There would be too much noise in the Segmented output. To remove this noise, Image Filtering is done.By using Image processing techniques like Dilation, thresholding, erosion, the edges and noise inside the segmented parts are filled based on the density around the segmented parts. The image is labelled with walk-able segments labelled in white pixels and all the other non-walk able fragments are in various pixel colors rather than white.

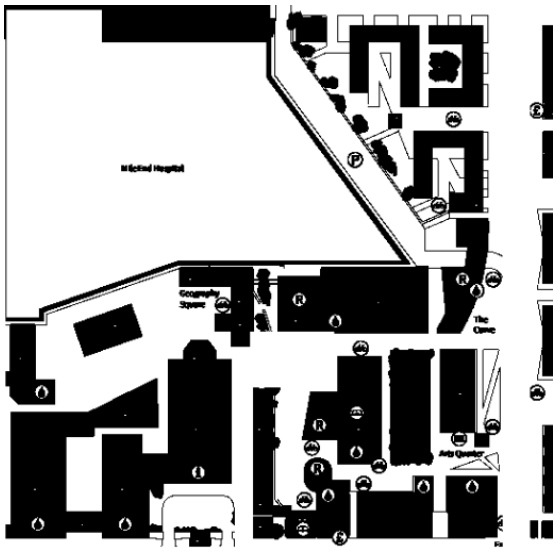

Fig. 2. Noisy Segmented Output

*3) **Edge Cases**:* The Segmented output which we have after preprocessing is not the perfect segmented Image. A still more rigorous segmentation is needed which covers all the boundary and corner cases like the building extensions(under paths through the building) covers the walk able parts , the bridges also include the walk able parts, the cycling tracks also include the walk able parts, the symbols like Parking, recreation facilities and the Bancroft road in the Mile end passes through an building underpass(so, in a 3D-world the building and road at the same location will be tough to represent in a 2D image). The Circles which include the key locations like parking, water fountains, smoking area, fitness center and library are to be removed. (They could also be indexed with additional numbers and be represented in the Image). The buildings are represented as an entity at a single location rather than a certain region inside the Image. The grass is represented as walk-able segments and should be converted to white pixels. The grass provides a lot of obstacle density, so the final path obtained by the algorithm would not be a optimal path.To solve this problem,we do replace the Grass with White pixels based on color encoding of grass pixels.

The circles in the Image are removed by combining few Image processing techniques. So, first find the circles at which location are they present and color the whole area inside the circle using nearby pixels. If the circle surrounded by walk able segments, fill the circle by white pixels and if surrounded by non-walk able segments, fill it by black pixels. Initially we extract a small rectangular patch which has the circular symbols. The rectangular patches are of size 20x20 and these patches do have circle inside them. Then, the circles inside these rectangular patches are found using Open CV-Hough circle's function which try to identify all the circles in the rectangular patch. Each circle found is represented by their location and the radius. The next step includes to convert the rectangular patch into Gray scale. In the Gray scale rectangular patch, find the center of circle and check if the pixel surrounding the circle are white means the circle symbol in a walk-able region. So, replace the circular region with white pixels/black pixels depending on the surrounding neighbourhood pixels.

Fig. 3. Rectangular Patch with a Symbol

*4) **Nearest walkable pixel around building**:* The whole building region could not represented with specific single location and also as a single number. Instead when the algorithm try to find a path from the whole building region,it initially finds the nearest walk able pixel surround the building region and assumes this pixel as the origin and starts to find

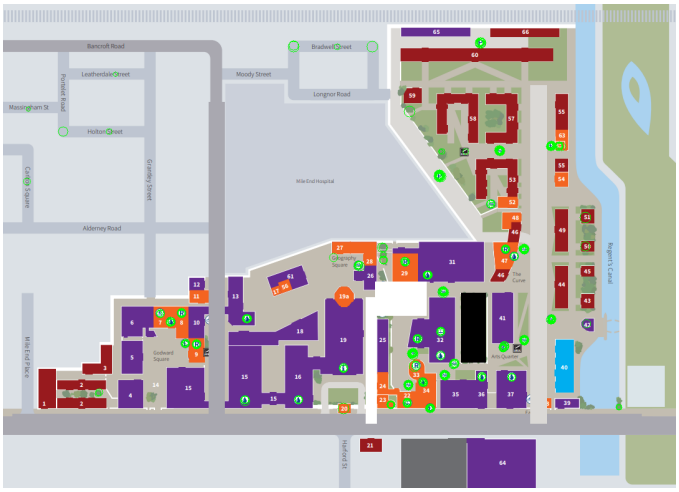

Fig. 4. Image with all detected Symbols

a path to the destination. It works the same with the pixels near the destination building.We find a path from nearest walkable pixel at origin's building to the nearest walkable pixel at the destination building.A new dictionary is created which finds the nearest walk able pixel from the building. The origin will be this nearest pixel from the building. The optimal path starts from the nearest pixel to the destination given by the user.This dictionary stores all the nearest walkable pixel from the respective building and we use this dictionary as the starting point to the algorithms in finding the optimal path to the destination.The Image could be cleaned near perfectly by labelling the grass work as walkable path and this eliminates and decreases the obstacle Density.

Fig. 5. Final Segmented Output

The segmented Image is preprocessed, cleaned and can be used in the path finding algorithms.

IV. Algorithms to Path Finding

The Image finally has the walk able segments and non-walk able fragments segmented such that the path finding algorithms finds a way through the walk able segments from the origin to destination. The output of the path finding algorithms is a pathway which maps the way from the Origin and Destination. The user inputs the origin and destination (example in google maps), the origin could be the user's location and destination could be any place in the University, the user wants to go.For example, the origin could be a library and destination could be Queen's building. The input will be converted to number-indexes based on the key-value pairs of the dictionary which has the location of the nearest walkable pixel around the building.The path finding algorithms like A*, BFS and DFS finds a path for the user from the origin to Destination. The algorithms find the shortest path through the walk able segments in the map. The Algorithms clearly maps out a path in walk able areas and the non-walk able fragments are being treated like Blockade's.

A. *Performance of the Algorithm*

The path finding algorithms are to be tested on a family of maps like random maps, real world maps, floor maps and maze maps (used in games). The performance of all these algorithms may differ comparing them with types of maps, but most of the papers use random maps (or) Maze maps(games).The final analysis being results in a biased analysis if tested on one set of Map. So, if the Algorithm does need to work good on all the above maps. The maps generated must be complex enough and should be different from each other to eliminate the bias in evaluating the performance of the algorithms. The factors which affects the performance of the algorithm are the obstacle density , distribution of obstacles which are the shape/region of the buildings. The percentage of obstacle density is a factor which also affects the analysis of these algorithms. The distribution of obstacles should also not be random which are mostly generated by random maps, but they should be in a organized way. The performance needs to be evaluated by increasing the obstacle density, by decreasing it and by changing the distributional shapes of obstacle density. The distribution of obstacles should be in such a way that there should be a path which can clearly be plotted from the origin and Destination(increase in object density and changing obstacle distribution may block the path from origin to destination).

The uncleaned Image could have an bigger effect on the Obstacle Density and distribution. This could create random obstacles at random locations with irregular shape/distribution which effects the path finding algorithms. A clean well segmented Image preprocessed using the Image processing techniques will result in a good path and shortest path from origin to destination.

B. *Uninformed Searches*

This is a blind searching Algorithm. These search algorithms explore the search space without any knowledge of the

goal. The algorithms does not spend any time in Planning and do not use any heuristics which need to be calculated.

*1) **Breadth First Search**:* Breadth First search is a type of Uninformed Search(the algorithm performs search without any information about the goal.).These uninformed algorithms explore the whole search space and terminates once the goal is found. In BFS,the Agent explores all the sub nodes that are reachable from a parent node before exploring any of the subnodes. So, all the nodes at the Depth (D-1) have to be explore before we reach a node at Depth(D). This Algorithm is easy to implement but time consuming and require heavy memory requirements.

*2) **Depth First Search**:* Requirements. The Depth first search explores a single sub node from a parent node as far as possible until a terminal state is reached. Following the consequent sub-node to each parent node.they require not heavy memory requirements .

*3) **Dijkstra**:* Dijkstra is the predecessor to A*. The sub node is choosen based on its distance from the Parent Node. Initially, we select a Parent node and add it to the queue. Later add all the neighbouring nodes to Queue-parent node. Now, Sort the queue based on its distance from the origin. This is a pathfinding algorithm which finds the shortest path from one node to other node. It's a variant of breadth first search.The ordering of the next nodes to be visited is based on distance to start node which is used as heuristic.

C. *Informed Searches*

Informed search uses the knowledge about the goal and also destination when exploring the search space.The measures used are problem map, estimated costs and estimate to goal location.

*1) **BFS-Best First Search**:* BFS uses the knowledge about the goal(destination) to explore the search space.This knowledge gives the algorithm which sub-node is more promising to explore from the parent node.

*2) **A***:* It is a variant of BFS.the Nodes are present in a 2D map where each node has neighbouring nodes.This algorithm also uses heuristic used in Dijkstra along with a additional heuristic which is how close it is to the node.This heuristic is calculated to a node and it represents its utility with respect to the destination. This allows the algorithm to reject exploring regions with high obstacle density and explore more promising regions. A* keeps a list of open nodes which could be reachable from already explored nodes.Now, estimate the distance from the open nodes, to the target. Finally, the node to explore will be the node with smallest value for the heuristic(sum of distance from origin + sum of distance to the Target).

As the map we use is a grid world, the heuristic we use should be close to real distance but based on the 4-way (or) 8-way connectivity. So, there are also many measures which we use to calculate the distance but the one which is more closer to the real world distance is the Euclidean distance rather than Manhattan distance (or) Chebyshev distance.the heuristic distance chosen should not overestimate the distance to the goal., then we could accept it.So, the use of this heuristic(distance from the target) will allow the A* to direct towards the destination faster compared to Dijkstra.

The variants of A* are Iterative Deepening A*, Hierarchial Pathfinding A*,

V. Experimentation

The experimentation is done on Google Colab which is a cloud based server used for running python scripts. The following experimentation is done on Image of size (776 x 1080 x 3).where the horizontal axis is 776 and vertical axis is 1080 and Image has 3 channels.The following experimentation's are done on following algorithms like Breadth First Search, Dijkstra, Best First Search and AStar.

The parameters chosen for experimentation are done Number of operations the algorithm takes to find the optimal path,Run time taken by the Algorithm to reach the optimal path, Path length-which counts the total number of nodes(pixels) covered by the algorithm during the path to the destination from the origin. The distance measure used by the algorithm could also be better parameter and experimentation is done using different distance measures like Manhattan distance and Squared distance.

The last parameter is C metric - this is a metric which different from the other one's. Currently Astar is the best algorithm when measuring to path Length from origin to destination.The C Metric would also be the total number of common nodes present between the nodes of comparable algorithm and the neighbouring nodes of reference algorithm.This C metric has a each path for a respective route following certain number of nodes which forms a path when grouped together.So, Breadth First Search(BFS) also has a path. This current metric(BFS, Astar) compares the both paths assuming the Astar as the best. Assuming AStar is the best, We check how many nodes in the path obtained by BFS are present in the path obtained by Astar and in the neighbouring nodes of the AStar Path.

$$C\_Metric = \frac{Total\ no.\ of\ Common\ Nodes\ between\ RA\ and\ CA}{Total\ number\ of\ Nodes\ in\ RA} \quad (1)$$

RA = Reference Algorithm
CA = Comparable Algorithm

So, Neighbouring nodes are selected based on a kernel(7x7).A node in the path obtained by AStar is the center pixel of the kernel and all the other surrounding pixels around the center are the neighbouring nodes. So, when comparing the two paths where one is a reference path and other is a path obtained by algorithm which needs to be compared with the reference path. Choose a single node from the path in the algorithm and check if this node is present in the neighbourhood nodes of the reference algorithm which is Astar. The final measure obtained is the count of total number of common nodes between by the algorithm and the reference algorithm-AStar.It could also be assumed that what

is percentage of common path between the algorithm and reference algorithm(AStar.)

So, Below are many tables which describe the experimentation of one single path.So, each table consists of user's input(origin and destination), and we run the multiple algorithm's for the same path between origin and destination.We take a note of all the parameter's like number opearations needed to reach the destination, what is the path Length(total number of nodes), What is the run time of the algorithm and metric which measures the percentage of algorithm's common path with the reference algorithm's path.

TABLE I
PATH(1-66) AND ITS RESULTS

Algorithm Type	**(1-66)- Path between Index 1 and 66**			
	Operations	***Path Length***	***Run Time(sec)***	***C metric***
BFS	188043	1411	4.97	17.36
Dijkstra	193279	1138	17.2	71.8
BestFirst	34584	1818	10.9	8.03
AStar	80267	1138	19.7	100

[a].

TABLE II
PATH(6-65) AND ITS RESULTS

Algorithm Type	**(6-65)- Path between Index 6 and 65**			
	Operations	***Path Length***	***Run Time (sec)***	***C metric***
BFS	173520	1406	4.76	35.9
Dijkstra	180177	1189	15.5	59.9
BestFirst	32621	1287	8.25	30.38
AStar	94864	1189	17	100

[a]

TABLE III
PATH (20-59) AND ITS RESULTS

Algorithm Type	**(20-59)- Path between Index 20 and 59**			
	Operations	***Path Length***	***Run Time (sec)***	***C metric***
BFS	132543	1025	3.94	38.6
Dijkstra	134757	825	12.6	66.06
BestFirst	18932	1070	6.36	39.6
AStar	72787	825	18.1	100

[a]Sample of a Table footnote.

The Below fig(6). shows the common path between two algorithms. The one in white would be the reference algorithm and the other in blue would be the comparable algorithm. The fig shows the common nodes visited by the both the algorithms during their path from the Origin to Destination.

The current fig(7-10) images show the path from start to the destination for the Astar algorithm.The five images shows five different paths from which comprises of the most Complex paths of the algorithm. The PATH (1-66) comprises of the longest path in the whole Map. The other paths like (6-65) and and (20-59) have a lot of obstacle density through their path. The PATH(39-9) is the shortest path among the five.

TABLE IV
PATH(39-9) AND ITS RESULTS

Algorithm Type	**(39-9)- Path between Index 39 and 9**			
	Operations	***Path Length***	***Run Time (sec)***	***C metric***
BFS	106705	685	3.77	29.63
Dijkstra	113201	620	12.5	91.6
BestFirst	728	643	3.53	22.08
AStar	8801	620	4.63	100

[a]

TABLE V
PATH(46-33) AND ITS RESULTS

Algorithm Type	**(46-33)- Path between Index 46 and 33**			
	Operations	***Path Length***	***Run Time (sec)***	***C metric***
BFS	35765	327	3.42	46.17
Dijkstra	35522	257	5.73	80.93
BestFirst	514	280	2.89	61.78
AStar	7420	257	3.9	100

[a]

VI. ANALYSIS AND DISCUSSION

The algorithm BFS-Breadth First Search is fastest algorithm and takes less time to reach the destination from the origin.But the number of operations made Breadth First Search algorithms are huge when compared to every other algorithm. The path length also covered by the algorithm is large when compared to other algorithms. This BFS although takes large number of operations and has the largest path length,its the fastest algorithm because it doesn't take much time to make a

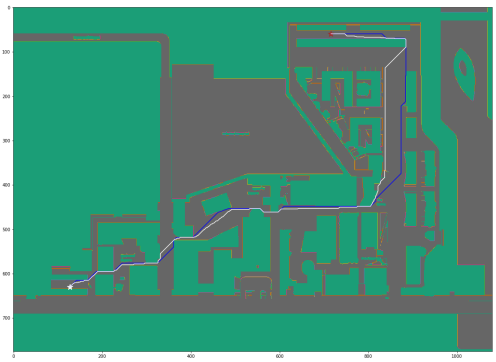

Fig. 6. PATH(2-65)- Common Paths for two Algorithms

Fig. 7. PATH(1-66)- AStar Implementation

Fig. 9. PATH(20-59)- AStar Implementation

Fig. 8. PATH(6-65)- AStar Implementation

Fig. 10. PATH(39-9)- AStar Implementation

operation and to visit a node. BFS doesn't use any heuristics to make a operation and it explores the nodes using based on the Breadth tree search algorithm.

Best First Search-BeFS takes the smallest number of operations to find the goal for the complex paths and also the short paths. The number of operations required fro BeFS differ by a large amount of (100 to 1000's) when comapred to every other algorithm.BeFS is the fastest algorithm(runtime) for the simpler simple path(small path-PATH(39-9) and PATH(46-33)), but when it comes to complex paths it performs better(faster) than AStar in terms of runtime but slower than BFS. BeFS and BFS doesnot follow the best Path and has a bad C metric when compared to AStar(shortest path). BeFS and BSS doesn't follow the optimal path and tries to find a different path for the simpler Paths and also for the complex Paths.

A. *Common Path*

Based on the following figures and tables, the AStar and Dijkstra share the most common path when compared using the C metric.The C Metric of AStar and Dijkstra is high as they have a common path rather than when compared with AStar as reference and other algorithms as comparable algorithms.When we choose the reference algorithm as AStar,Dijkstra has the best common path with AStar. The reason for this could be that both of the algorithms are built on similar heuristic.So, the method of finding the optimal path based on the heuristic could also be similar. AStar is built on the heuristic of Dijkstra and made better using an additional metric(distance from the goal and origin). whereas Dijkstra uses only distance from the origin.Due to the AStar's better heuristic , AStar takes less number of operations to reach the goal from the origin unlike Dijkstra. Dijkstra even takes a

Fig. 11. Common Path-AStar and Dijkstra

Fig. 12. UnCommon Path-BeFS and AStar

large number of operations to find a simple path(small path-PATH(39-9) and PATH(46-33)) unlike AStar which takes small number of Operations.

When it comes to the long complex path's which have a huge obstacle density, Astar's take a long time to run compared to Dijkstra, but for simple path(small path-PATH(39-9) and PATH(46-33)), AStar runs very fast compared to Dijkstra.So,before AStar visits a node, it calculates the heuristics and makes a decision whether to perform this operation to this node or not. So, time taken to make an operation is based on the obstacle density and the heuristic.We do consider the obstacle density from the node to the goal and from the node to the origin for AStar unlike Dijkstra which only considers the obstacle density only from node to origin and ignores obstacle density from node to destination.

B. Uncommon Path

Best First Search always takes the most uncommon path when compared to AStar or other algorithms as reference. This Weird behaviour of Best First Search is observed when choosing the simple path(small path-PATH(39-9) and PATH(46-33)) and also Complex path's. The algorithm's name itself suggest its the best in search algorithm when chosen number of operations as measure.Whereas, the other algorithms do have atleast a little amount of common path when compared to each other unlike when compared with Best First Search.

VII. Future Work

A* is the advanced algorithm being used mostly in the path finding algorithms. But recent advancements in computer science has led to more powerful path finding algorithms which make a better use of memory, time,number of expanded nodes, length of path using powerful CPU's . path finding algorithms like IDA*, Theta*, HPA*. For image segmentation, we could possibly use recent advanced deep learning algorithms like FAST-RCNN for segmentation which gives a clear way to segment and separate the walkable and non-walkable paths in the Image.

The input given to the algorithm are the Name of origin location in the map and the name of destination. Instead the input could be the voice of the speaker and the speaker audio pronunciation of the origin and the destination. The algorithm understands the audio and maps the audio to the ID of the location.

There is a need to detect the Hangover bridges and building extension where the road passes under the building.The labelling of the bridges is done using the arrows such that its easy for the user to understand that its a bridge and a there is a underpass road passing under the bridge.

VIII. Conclusion

The experimentation done using Path Finding Algorithms on a Raw unprocessed campus map proved to be achievable and effective when using Image processing techniques and Image Segmentation.The additional metric-(C Metric) chosen to compare the paths of algorithms finds the similarity of a algorithms path with Gold standard path. This metric is also proven to be impressive when comparing two different Algorithms.Thus,Image Processing on Raw unprocessed Campus Map helps the path-finding algorithm reach the destination from the origin avoiding the obstacles.

IX. Acknowledgement

I would like to thank Dr. Julian Hough for being supportive and his uplifting attitude all through the Project Sessions. The

given recommendations were pivotal, which helped in better outcome of paper and implementation of the algorithms.

REFERENCES

[1] P.E. Hart, N.J. Nilsson, and B. Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4. 4 (2): 100–107.

[2] DANTZIG, G.B., 1957, Discrete-variable extremum problems. Operations Research, 5, pp. 266-277.

[3] MINTY, G., 1957, A comment on the shortest route problem. Operations Research, 5, pp. 724.

[4] DIJKSTRA, E.W., 1959, A note on two problems in connexion with graphs. Numerische Mathematik, 1, pp. 269-271.

[5] A.R. Soltani, H. Tawfik, J.Y. Goulermas, and T. Fernando. Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms. Advanced Engineering Informatics, 16(4):291–303, October 2002.

[6] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, andP. Feyzeau. Path planning: A 2013 survey. In Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM), pages 1–8, October 2013.

[7] U. A. S. Iskandar, N. M. Diah and M. Ismail, "Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games," 2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), 2020.

[8] Harinder Kaur Sidhu, "Performance Evaluation of Pathfinding", School of Computer Science, University of Windsor.

[9] Song Yuheng, Yan Hao, "Image Segmentation Algorithms Overview",1. SiChuan University

[10] Marr D, Hildreth E. Theory of edge detection[J]. Proceedings of the Royal Society of London B: Biological Sciences, 1980, 207(1167): 187-217.