Project Update (MVP)

Damanvir Singh | Munib Syed

Original planned features

* Indoor navigation
* Calculating optimal/near-optimal paths
* Floor plan reading and processing
* IPS (if included) using Wi-Fi triangulation
* Universal application (works inside any building)
* Web application (works on most platforms)

Revised features

* Indoor navigation
* Calculating optimal/near-optimal paths using a sufficient path finding algorithm like A\* or Concurrent Dijkstra’s
* Eraser tool or fill tool to account for unwanted or wanted wireframe obstacles.
* Floor plan reading and processing - using ray-casting/ray-tracing algorithm in OpenCV
* IPS (if included) using Wi-Fi triangulation
* Universal application (works inside any building and on multiple floors)
* Web application (works on most platforms if they have a web browser)
* User manual

Features demonstrated by the MVP

* Floor Plan Reading - obstacle detection on floor plans using ray-casting/ray-tracing algorithm, which currently works for all non-wireframe obstacles.
* Floor Plan Reading - placing nodes and edges (using image coordinates) while avoiding obstacles, which works unless the obstacles are extremely thin (in which case they will likely be missed by the algorithm).
* Web Application - custom maps can be used with Leaflet.js using a gem called tileup.
* Web Application - nodes can be displayed on the map as markers using the same node coordinates given from ray-casting/ray-tracing algorithm.
* Path-finding using Dijkstra’s algorithm.

Justification for features not included in the MVP

* IPS - this feature needs more extensive research to be implemented. Plus, this feature is not a core feature.
* Path finding using either A\* or Concurrent Dijkstra’s algorithm - this feature is not yet included as both of these path-finding algorithms require testing and research to conclude which one serves our purposes better. Therefore, the algorithm in use for path finding so far is Dijkstra’s algorithm, which serves our purposes fine but is not the most efficient algorithm when it comes to running time.
* The fill-tool/eraser tool - is a method of optimizing the ray-tracing algorithm to work with a wide variety of maps. It has been shown in the MVP that the ray-tracing algorithm is capable of accurately detecting obstacles in a simple map, so accounting for more difficult maps through optimization is not a necessary step at this stage.
* Universal application - so far the application can work inside most buildings as our algorithm for floor-plan reading is not perfect yet. Also, the application cannot work on multiple floors yet as this features requires more testing and research.
* User manual - not complete yet because the rest of the project must be complete in order to write a comprehensive manual.

Justification for features added/removed

Added Features:

* Eraser tool/fill-tool - while neither of these features have been coded yet, it is clear that they are very necessary for dealing with some more complex obstacles, as ray-tracing gives incorrect results when dealing with “wireframe” obstacles, i.e. ones that are not filled with a solid colour.
* A proposed feature is a method to optimize the jump value depending on the map size and obstacle size. This would optimize the floor-plan reading algorithm to make fewer mistakes.
* One proposed feature is vertical ray-tracing. This will again help to deal with some more complex maps. For instance, horizontal ray-tracing (what we are using now) may miss some obstacles that are very thin (horizontally). Similarly, vertical ray-tracing may miss some obstacles that are very thin, vertically. If both can somehow be used in conjunction, then more obstacles can be accounted for.
* A user manual will necessary as it will teach the users on how to use the application(s).

Removed Features:

* IPS - **if** time and technology does not allow.
* Obstacle detection via corner detection - this was an unviable method due to its self-referential nature.

Instructions to use the MVP:

Web application and Path finding

1. In order to view the map/actual product, open up both index.html and index2.html. Both files are different in terms of the content they have in them. index2.html is the latest file with the most progress, so it is recommended to use index2.html.
2. In index2.html, it is visible what our final product will look like as most of the features including nodes, shortest and path custom mapping are displayed.

Computer Vision:

1. The computer\_vision folder contains the ray-tracing algorithm (rayTraceClean.py). It can be tested by inputting your own .png file and uncommenting appropriate parts of the code. Have a recent version of OpenCV installed. This folder also contains two test images, testfile.png and testfile2.png.