**CHAPTER 2**

**REVIEW OF RELATED LITERATURE AND STUDIES**

In this chapter lies studies related to and supports our research*.*

**Related Systems**

**1. Foreign Systems**

**a. The A\* Algorithm**

In the given study, the researcher used the A\* Algorithm in order to find the shortest path to an objective through obstacles. The researcher takes into consideration four factors that can affect the AI path decision: (1) the AI starting point, (2) the destination point, (3) the obstacles, and (4) the “cost score” for each possible path. This cost score is calculated based on the series of movements that the algorithm predicts. For instance, the algorithm finds that two routes A and B have the same travel distance of 15m. Route A requires a change in elevation and 5 directional changes. Route B requires only two directional changes. From the algorithm, we can conclude that route A would yield a lower “cost score” and will be marked more desirable by the AI.

**b. Occlusion Culling**

In [3D computer graphics](https://en.wikipedia.org/wiki/3D_computer_graphics), **hidden surface determination** (also known as **hidden surface removal** (**HSR**), **occlusion culling** (**OC**) or **visible surface determination** (**VSD**)) is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint.[1] An Occlusion Culling method used here involves dividing the entire map area into sections. Sections are rendered individually. The sections to be rendered are determined by which sections are visible to the player camera. Although this works a bit less efficiently since sections are large and that entire sections may not even be visible. This means that even if just 1/8th of a section is visible, the entire section will still be rendered. We’d suggest that the researcher change his/her OC method (or maybe include this in our research scope.)

**2. Local Systems**

**a. The A\* Algorithm**

This research uses the A\* Algorithm the same way as the previous study, this time making use of it in an interface to help the visually impaired by applying “shortest path”. The user (a visually-impaired individual) has a specialized crane that contains a microprocessor, sensors, and a transmitter. They will also possess a headset that will relay instructions on how to proceed from a remote server. This server is what contains the application logic programmed in MATLAB and Microsoft’s C#.

A brief explanation of its working process is that the user’s crane will help pinpoint the current location of the user. The sensors embedded in it will detect the location of obstacles and clear waypoints. The crane then transmits these data to the server, who then uses the A\* algorithm to find quickest way to the user-defined destination (via the headset), taking into account the user’s current location, the defined destination, the waypoints, and the obstacles.

This study, however, is limited to indoor use, as per the researchers.

**b. Shortest Path**

The way this works is that the system scans the environment. Using the sensors on the crane, the system finds all the obstacles, waypoints, and the user’s current location. The user’s desired destination is fed to the server using the headset. Once all of the data is gathered, the server gathers the data and assigns the user location and destination, and all of the waypoints and obstacles to their own individual nodes. The nodes are then mapped as a matrix and their cell locations are determined by their real world locations. Once the matrix is built, the A\* Algorithm may now come into play, finding all possible paths defined by the waypoints and avoiding obstacles. Each successful path is weighed by their “movement cost” (“F” cost, as defined by the researchers). The path with the smallest movement cost is considered the shortest path is then converted into a vocal message to be transmitted to the user’s headset.