Journal of Engineering and Technology

ISSN 2180-3811 eISSN 2289-814X https://jet.utem.edu.my/jet/index

DEVELOPMENT OF A LIBRARY NAVIGATION SYSTEM USING DIJKSTRA'S ALGORITHM

A. A. M. Salleh[1] and K. Abdulrahim*[1]

[1] Department of Electronic/Electrical Engineering, Faculty of Engineering and Built Environment, Universiti Sains Islam Malaysia, 71800, Nilai, Malaysia.

corresponding: khairiabdulrahim@usim.edu.my

Article history:
Received Date:
27 March 2023
Revised Date:
10 June 2023
Accepted Date:
21 June 2023

Keywords:
Navigation System, Library, Dijkstra's Algorithm, Simulation, Android Studio

Abstract— The aim of this article is to explore the potential of indoor navigation systems for library users, an area that has not been fully explored in previous research. The design of library buildings, which typically feature multiple floors and hundreds of book racks, poses a challenge for users to locate specific books or articles. As traditional navigation signals such as Global Positioning System signals have been found to be unreliable for indoor navigation, this article proposes a navigation system using the shortest distance algorithm to assist library users. To test the proposed system, we implemented the Dijkstra algorithm in a navigation software prototype using the real USIM library layout

as a case study. The algorithm was able to assist users in navigating the library even with obstacles, by providing the shortest path from a given starting point to a desired destination. The results of this study demonstrate the potential for indoor navigation systems in libraries and highlight the effectiveness of the Dijkstra algorithm in providing accurate and efficient route planning for users. This research provides a foundation for future studies exploring the application of navigation systems in other indoor environments and the development of more sophisticated algorithms to further improve navigation accuracy and efficiency.

I. Introduction

Localization techniques such as Wi-Fi [1], Bluetooth [2], RFID [3], and computer vision [4] are commonly used for positioning users within libraries. However, each technique has limitations, including signal interference, accuracy issues, and high implementation costs. Mapping and floorplan generation techniques, such as SLAM [5], are crucial for creating accurate representations of library spaces. However, challenges arise due to the dynamic nature of libraries, making real-time mapping and updating a complex task. Route planning algorithms, such as Dijkstra's algorithm [6] or RRT (Rapidly exploring Random Trees) [7], help users navigate through libraries efficiently. Nonetheless, these algorithms may struggle [8] with considering dynamic obstacles, multi-floor navigation, and providing personalized routes. This becomes the key focus of the work in this article. User interfaces, such as graphical, AR, and voice-based interfaces, aim to enhance user experience, but they may have limitations in terms of usability, accessibility, and user acceptance. Integrating emerging technologies like

Internet of Things, IoT [9], beacons, and Artificial Intelligence, AI [10] can bring innovative solutions to library navigation, but challenges such as interoperability, scalability, and privacy concerns must be addressed.

The majority of the world's libraries are multi-floor and quite large, presenting a significant challenge for unfamiliar visitors such as new students or foreign patrons. For example, the Bodleian Library at the University of Oxford [11] has eleven floors with various blocks, making navigation a daunting task. Similarly, the Thomas Fisher Rare Book Library [12] at the University of Toronto, while medium-sized, can still take days or even weeks to navigate due to its vast collection of books. As a result, library patrons spend more time looking for books, while strangers have even more difficulty finding them. To address this issue, route planning can be used to help navigate these complex library systems.

This article revisits the development of a library navigation system that uses Dijkstra's algorithm to provide the shortest and fastest route to the desired destination, while also allowing for adjustments to be made in response to dynamic barriers, such as moving book racks. The article also describes the validation of the algorithm through various scenarios and obstacles, as well as the implementation and performance of the algorithm in a prototype navigation system.

The article is structured as follows: the materials and methodologies used are outlined, including the system's flow architecture, prototype development, and simulation testing. The results and discussion are then presented, followed by a conclusion and discussion of future work.

II. Material and Methodology

A. The system's architecture and interface concept

In this proposed library navigation system, the system architecture consists of two core roles: user and administrator. Figure 1 shows how the system flow architecture works.

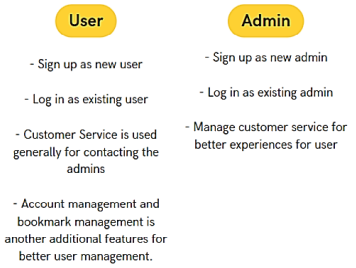

Figure 1: System Flow Architecture

A library navigation system is designed to provide a streamlined experience for library users, allowing them to easily find books and articles within the library. Both the user and administrator roles in this system have similar features for registration and login, but also have unique features tailored to their specific needs. Administrators can manage customer service to provide a better experience for users, while users have access to customer service benefits such as problem reporting and helpdesk support. Additionally, users can use tools such as account management and bookmark management to better organize their navigation experience.

The flowchart for this library navigation system illustrates that the major component of the software has its own database, integrated with MySQL system software. The system provides a range of functionality for both user and administrator roles, with a focus on efficient navigation within the library. As the system continues to evolve, it will need to work with the MySQL software database, connecting to the library book database and location. This will enable the system to better organize and separate the two roles, providing an even more streamlined and effective navigation experience. Figure 2 below shows the flowchart for the library navigation system.

Once the system setup is complete, the navigation system interface is established as follows: The prototype system's interface is the simplest, with only a distance indicator, origin and destination indicators, and a search bar for the specified location. An example is shown in Figure 3.

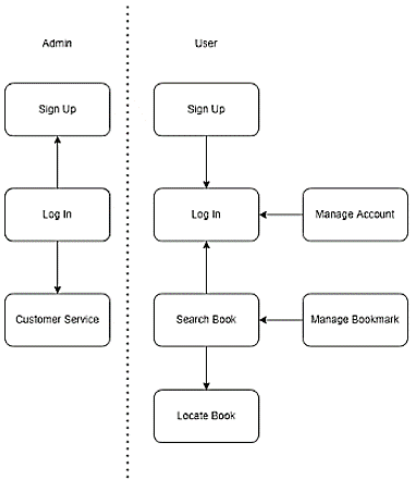

Figure 2: System Flow Architecture's Flowchart

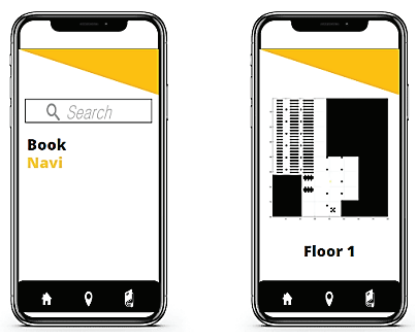

Figure 3: Interface Concept

The Android Studio software [13] was utilized to create the user interface for the library navigation system. Users can easily search for their desired location using the search panel provided in the interface. To integrate map layouts into the system, .png files are used instead of a full code algorithm. The map layout used in this work was created using the MATLAB software, where obstacles such as bookshelves, tables and chairs, walls, and columns are encoded as blank spaces that cannot be accessed by the algorithm. The map layout model used for simulations in the MATLAB software was the main area of the Universiti Sains Islam Malaysia (USIM) library on the first floor. Figure 4 showcases the map layout model that was developed for the purpose of simulations running in the MATLAB software.

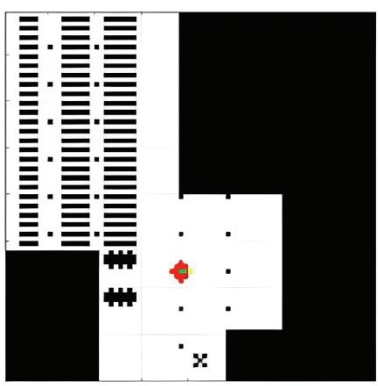

Figure 4: Map Layout Model for the Simulation System

In the MATLAB simulation software, the map layout displays several key elements to aid in navigation. The user's starting point, or source, is represented by a vivid green box,

while the intended destination is marked with a bright yellow box. The region that the algorithm is calculating is shown in red, while the area computed by the Dijkstra algorithm is highlighted in a subdued gray. Obstacles in the area, such as bookshelves, tables, chairs, walls, and columns, are indicated by black boxes. All of these components are integrated into the MATLAB software, providing an intuitive and user-friendly interface for simulation purposes.

B. Implementation of Dijkstra's Algorithm

To utilize Dijkstra's algorithm, one must possess a fundamental comprehension of its operational procedures. The algorithm has had widespread use in applications such as packet switching software for computer communications [14]. While the original Dijkstra algorithm was employed to identify the shortest route between two nodes, a more general form involves the selection of a single node to serve as the "source" node. This allows for the computation of paths between the source node and all other nodes within the network, resulting in the construction of a shortest-path tree. To understand how it works, the basic algorithmic model of the temporal algorithm is shown in Equation (1) below:

$$O((|E|+|V|)\log|V|) \quad (1)$$

where:
O = Time Algorithm
E = Edges
V = Vertices

The execution time bound for Dijkstra's algorithm on a graph with edges E and vertices V is given as a function of the number of edges represented by $|E|$. The number of vertices represented by $|V|$ uses Big-O notation. The data structure used to represent the set Q largely determines the bounds on complexity. Since $|E|$, the upper limit can be set as follows: It is $O($ 〚\|V|〛 ^2) in each graph that is tightened, but this relaxation ignores the possibility that different upper bounds on $|E|$ apply to some problems.

The code used consists of Dijkstra's algorithm with calculations integrated in MATLAB. The 'j' in the code

represents the number of columns and the 'i' represents the number of rows. Then we use the *distanceFromStart* function written in the time direction to calculate the probabilities from the source to the destination for each step. We assume a constant user step and is equivalent to each iteration. One pixel of map is equivalent to one step of a user taken. Although this is a very simplified approach, it is consistent with the main goal of the navigation system, which is to find the shortest route to the destination. From there, the algorithm then completes the location that tracks the shortest distance based on how long it takes to get there. An example of code used to simulate Dijkstra's algorithm is shown in Figure 5.

```
if j ~= 1
    if map(i, j-1) ~= 2 && (map(i, j-1) ~= 3 && map(i, j-1) ~= 5)
        t = distanceFromStart(current)+cost(i, j-1);
        if t < distanceFromStart(i, j-1)
            distanceFromStart(i, j-1) = t;
            parent(i, j-1) = current;
        end
    end
end
if i ~= 1
    if map(i-1, j) ~= 2 && (map(i-1, j) ~= 3 && map(i-1, j) ~= 5)
        t = distanceFromStart(current)+cost(i-1, j);
        if t < distanceFromStart(i-1, j)
            distanceFromStart(i-1, j) = t;
            parent(i-1, j) = current;
        end
    end
end
if j ~= ncols
    if map(i, j+1) ~= 2 && (map(i, j+1) ~= 3 && map(i, j+1) ~= 5)
        t = distanceFromStart(current)+cost(i, j+1);
        if t < distanceFromStart(i, j+1)
            distanceFromStart(i, j+1) = t;
            parent(i, j+1) = current;
        end
    end
end
if i ~= nrows
    if map(i+1, j) ~= 2 && (map(i+1, j) ~= 3 && map(i+1, j) ~= 5)
        t = distanceFromStart(current)+cost(i+1, j);
        if t < distanceFromStart(i+1, j)
            distanceFromStart(i+1, j) = t;
            parent(i+1, j) = current;
        end
    end
end

distanceFromStart(current) = Inf;
numExpanded = numExpanded + 1;
```

Figure 5: An example of Dijkstra's Algorithm's code used

Incorporating the code effectively into the prototype system requires rigorous testing of its functionality. This includes comprehending the functional prerequisites, recognizing the test inputs, evaluating the expected outcomes, and executing the test code. A proven methodology for carrying out these processes is Agile software development, which is widely adopted by software developers and organizations. However, elaborating on this approach is beyond the purview of this article.

III. Results and Discussion

The software development environment, Android Studio, was utilized to design the Android app. To test the performance of the algorithm and analyze its potential benefits and drawbacks, various scenarios were employed in MATLAB. Dijkstra's study emphasizes two main aspects: interface design and system flow architecture. The Android Studio was employed to develop the interface concept, while the board layout was simulated using MATLAB software. The subsequent step involved thorough testing to put the algorithms through their paces.

A. Through Multiple Obstacles

The testing of algorithms involves a range of simulation scenarios to gain a thorough understanding of their operations. This can be a complex process, encompassing both basic tests and more intricate simulations.

Figure 6 illustrates a scenario in which the algorithm calculates a route without obstacles. From this information, the shortest distance to reach our destination would be via the right path.

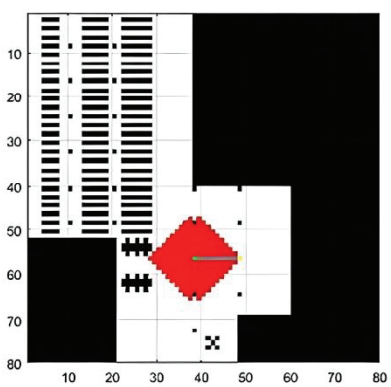

Figure 6: Simulation Without Obstacle

From Figure 7, the algorithm attempted to avoid the obstacle in the column closest to the goal. This was due to a code snippet where 'j' column appears first, which caused the program logic to calculate and proceed from this column instead of moving towards the right as expected.

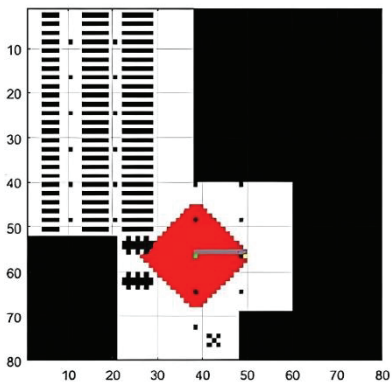

Figure 7: Simulation Through Obstacle

From Figure 8, the algorithm has decided which path to take based on its current location and surroundings. This situation indicates that the algorithm is trying to move towards a goal in a direct parallel manner; however, because both paths have equal distance (hence +1), it instead chose to go left, despite this being less desirable. In other words, due to how the code was written, map (i, j-1) comes before map (i, j+1), indicating that the algorithm prefers going left (which is the negative region), rather than to the right (which is positive region).

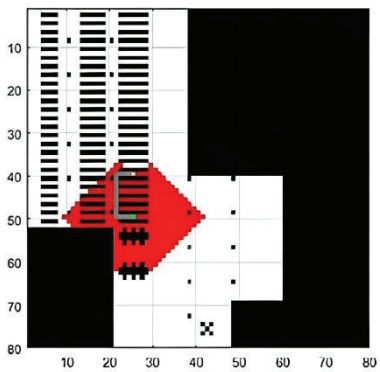

Figure 8: Simulation Through Multiple Obstacle Situation 1

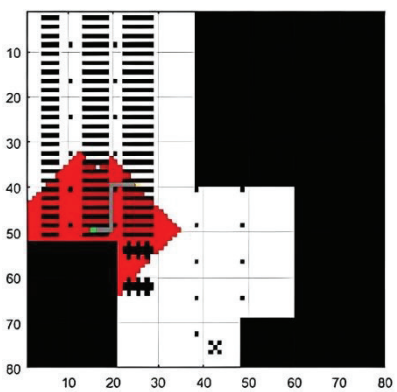

Figure 9: Simulation Through Multiple Obstacle Situation 2

This particular scenario depicted in Figure 9 is merely an illustration of the obstacles that

the algorithm encounters when combining the aforementioned methods. In this case, the algorithm aims to navigate from the bottom-left to the top-right, but it chooses the upward direction instead of the rightward direction, which is the more optimal path. This is due to the way the encoding was programmed.

B. Through Different Scenario

The presented Figure 10 depicts different scenarios in which the algorithm is expected to navigate through multiple obstacles and floors to reach the intended goal. Unfortunately, MATLAB was unable to integrate various map layouts, leading to a challenge in accurately simulating the algorithm's behavior. To overcome this limitation, a proposed solution involving the use of a stairway between floors 1 and 2 is shown in the obstacle diagram. This idea aims to eliminate the multi-floor complexity and facilitate the algorithm's navigation through the area.

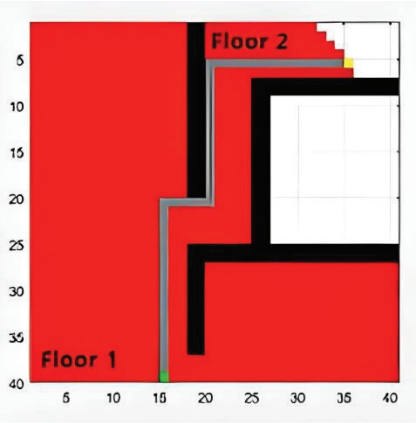

Figure 10: Simulation Through Different Scenario

IV. Conclusion

This article delves into the utilization of Dijkstra's algorithm for the development of a library navigation system. The simulation conducted exposes certain limitations of the algorithm, including its inability to calculate non-shortest diagonal paths and the challenge of combining multiple layouts for comprehensive testing. Despite these limitations, the algorithm showcases satisfactory results in basic simulations.

In conclusion, the simulation effectively demonstrates that, given a well-designed floor plan, Dijkstra's algorithm can accurately predict shortest

routes and address navigation challenges within a library. Although the algorithm has its limitations, it proves to be a simple, efficient, and accurate solution for short-range travel, particularly within library settings. Overall, the simulation serves as a valuable tool for testing the performance of Dijkstra's algorithm and lays the foundation for the development of a functional library navigation system prototype.

V. Acknowledgement

The research leading to this paper was partly supported by Universiti Sains Islam Malaysia through research code: PPPI/FKAB/0121/USIM/17321.

VI. References

[1] Rizk, H., Yamaguchi, H., Youssef, M., & Higashino, T. (2023). Laser range scanners for enabling zero-overhead wifi-based indoor localization system. ACM Transactions on Spatial Algorithms and Systems, 9(1), 1-25.

[2] Szyc, K., Nikodem, M., & Zdunek, M. (2023). Bluetooth low energy indoor localization for large industrial areas and limited infrastructure. Ad Hoc Networks, 139, 103024.

[3] Bi, S., Wang, C., Shen, J., Xiang, W., Ni, W., Wang, X., ... & Gong, Y. (2023). A Novel RFID Localization Approach to Smart Self-Service Borrowing and Returning System. CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES, 135(1), 527-538.

[4] Roy, A. M., Bhaduri, J., Kumar, T., & Raj, K. (2023). WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. Ecological Informatics, 75, 101919.

[5] Shao, S. (2023). A Monocular SLAM System Based on the ORB Features. In 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA) (pp. 1221-1231). IEEE.

[6] Dharmasiri, P., Kavalchuk, I., & Akbari, M. (2020). Novel implementation of multiple automated ground vehicles traffic real time control algorithm for warehouse operations: djikstra approach. Operations and Supply Chain Management: An International Journal, 13(4), 396-405.

[7] Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011, May). Anytime motion planning using the RRT. In 2011 IEEE international conference on robotics and automation (pp.

1478-1483). IEEE.

[8] Alqahtani, E. J., Alshamrani, F. H., Syed, H. F., & Alhaidari, F. A. (2018, April). Survey on algorithms and techniques for indoor navigation systems. In 2018 21st Saudi Computer Society National Computer Conference (NCC) (pp. 1-9). IEEE.

[9] Madakam, S., Lake, V., Lake, V., & Lake, V. (2015). Internet of Things (IoT): A literature review. Journal of Computer and Communications, 3(05), 164.

[10] El-Sheimy, N., & Li, Y. (2021). Indoor navigation: State of the art and future trends. Satellite Navigation, 2(1), 1-23.

[11] Raw, B. C. (1984). The construction of Oxford, Bodleian Library, Junius II. Anglo-Saxon England, 13, 187-207.

[12] Whyte, J. (2017, June). Preservation planning and workflows for digital holdings at the Thomas Fisher Rare Book Library. In 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL) (pp. 1-0). IEEE.

[13] Van Drongelen, M. (2015). Android studio cookbook. Packt Publishing Ltd.

[14] Ya-qiong, Z., & Yun-rui, L. (2016). A routing protocol for wireless sensor networks using K-means and Dijkstra algorithm. International Journal of Advanced Media and Communication, 6(2-4), 109-121.