Indonesian Journal of Data and Science

Volume 4 Issue 3
ISSN 2715-9936
https://doi.org/10.56705/ijodas.v4i3.110

Research Article

Shortest Route Navigation Indoors Using Digital Maps

Muhammad Arfah Asis [1,*], Muh Aliyazid Mude [2], Ririn Astiani [3], St Nadya Kurnia Prihandani [4]

[1] Universitas Muslim Indonesia, Makassar, Indonesia, muh.arfah.asis@umi.ac.id
[2] Universitas Muslim Indonesia, Makassar, Indonesia, aliyazid.mude@umi.ac.id
[3] Universitas Muslim Indonesia, Makassar, Indonesia, ririn.astiani@icloud.com
[4] Universitas Muslim Indonesia, Makassar, Indonesia, st.nady4@gmail.com

Correspondence should be addressed to Muhammad Arfah Azis; muh.arfah.asis@umi.ac.id

Received 18 October 2023; Accepted 10 December 2023; Published 31 December2023

© Authors 2023. CC BY-NC 4.0 (non-commercial use with attribution, indicate changes).
License: https://creativecommons.org/licenses/by-nc/4.0/ — Published by Indonesian Journal of Data and Science.

Abstract:

Digital maps have revolutionized our ability to navigate to desired locations by providing the shortest routes, primarily in open spaces. However, this functionality is limited to outdoor environments. This study aims to extend this capability by enabling the determination of the shortest routes within indoor spaces. The research employs the Haversine method for distance measurement and integrates the Dijkstra algorithm for route determination. The findings demonstrate the feasibility of implementing the Haversine method and the Dijkstra algorithm for route determination within enclosed spaces. It was observed that the routing machine between route nodes did not perform optimally, prompting its replacement with a polyline-based approach.

Keywords: Digital Maps, Dijkstra Algorithm, Haversine Method, Indoors, Polyline

1. Introduction:

The city's development can be seen from the increasing number of public buildings or public places being built. Public buildings can be used or visited by the public, such as hospitals, airports, zoos, beaches, and malls. One of the places that is most popular among both young and old is the mall. [1]. Some public buildings are open and some are closed [2]. Malls are included in the category of closed public buildings where many spaces can be visited inside.

As buildings develop in the city, technological developments follow. One technology that helps us visit public places is a geographic information system (GIS). GIS helps us find out the location of the place we are looking for using digital maps. One of the digital maps that is popularly used is OpenStreetMap (OSM) [3].

Digital maps can show us the closest route to the location we want to go to. But this only applies to open areas. Currently, there are many closed buildings which are very spacious. So sometimes it is very difficult to find space in it like a mall. A mall is a multi-story building that contains many shops. In Makassar, there is a mall which has an area of around 70 thousand square meters. This mall consists of 5 floors and 1 basement.

Many applications or systems help us in searching for locations, but most of them only work in open spaces. The rest are applications specifically designed for specific buildings. As in research that implements Dijkstra's algorithm to determine the shortest route in the mountains [4]. Some carry out research that calculates the distance between locations in closed spaces using the Haversine method [1]. In this research, we will implement the Haversine method and Dijkstra's algorithm to find the closest route in a closed building which will be applied to a digital map.

2. Method:

a. Research design
The stages of this research, searching for routes on a digital map, can be seen in **Figure 1**.

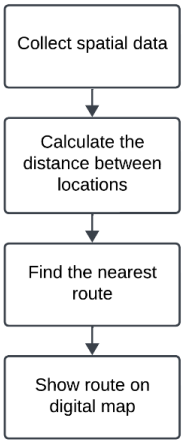

Figure 1. Stages of route finding on a digital map

The spatial data collection stage is carried out in the mall. Through this process, coordinate data is collected which includes each route node in it, as well as documenting the locations or spaces in the mall. This stage is an integral part of the mapping effort of the location structure within the mall.

After obtaining coordinate data, the next step is to calculate the distance between route nodes and connected locations. This measurement process utilizes the haversine method, a reliable method for estimating the distance between points on a spherical surface, such as measuring the distance between given geographic coordinates. This method allows accurate calculations in determining the distance between related locations in spatial data.

After obtaining the distance between connected nodes, the next step is to find the shortest route using Dijkstra's algorithm. This algorithm will calculate and determine the fastest or shortest path between previously connected nodes, ensuring the determination of the optimal route in the context of the previously measured distance.

The final step is to display the calculated route on a digital map, referring to the results obtained from the previous stage. This enables graphical visualization of pre-calculated shortest or fastest paths, providing a clear and intuitive picture of recommended routes based on pre-processed spatial data.

b. Haversine Method

The Haversine method calculates the distance between two points by acknowledging the Earth's curved surface. It employs the Haversine Formula to determine the distance between two coordinates on the Earth's longitude and latitude, considering the curvature of the planet rather than assuming a flat plane [5]. In the Haversine method, the input variables rely on latitude and longitude under the premise of Earth's near-spherical shape. The Haversine formula calculates the distance along a great circle between two points on the Earth's surface (treated as a sphere), assuming a radius (R) of 6,367.45 km. It uses the spherical coordinates (latitude and longitude) of the two points, designated as long1, lat1, and long2, lat2, respectively, to compute this distance [6]. The Haversine Formula takes the following form [5], [7]:

$$\Delta\text{lat} = lat2 - lat1$$
$$\Delta\text{long} = long2 - long1$$
$$a = \sin^2(\Delta lat/2) + cos(lat1).cos(lat2).\sin^2(\Delta long/2)$$
$$c = 2.atan2(\sqrt{a}, \sqrt{1-a})$$
$$d = R.c$$

Notes:

R = radius of the earth is 6371(km)
Δlat = the amount of change in latitude
$\Delta long$ = the amount of change in longitude
C = axis intersection calculation
d = distance (km)
1 degree= 0.0174532925 radians

c. Dijkstra’s Algorithm

Dijkstra's algorithm stands as a graph search method specifically designed to solve the shortest path problem originating from a single source on a graph. It operates effectively in scenarios where the graph contains no negative-weighted nodes, generating a tree that represents the shortest paths from the source to all other nodes in the graph [8]. Dijkstra's algorithm falls under the category of greedy algorithms, frequently employed in solving optimization-related problems [9].

Dijkstra's algorithm resolves the challenge of discovering the shortest path between two nodes within a weighted graph, aiming to identify the path with the least overall weight, such as distance or any specified metric [10]. Dijkstra's algorithm is instrumental in determining the shortest path within a weighted graph by identifying the minimum weight. This method calculates the shortest distance between two or more points within a graph, seeking the path with the smallest overall value. For instance, let's consider a labeled directed graph, G, with vertices V(G) = {v1, v2, …, vn}, where the sought shortest path is from v1 to vn. Initiating from v1, Dijkstra's algorithm proceeds iteratively, seeking the point whose cumulative weight from point 1 is the smallest. These selected points are isolated, no longer considered in subsequent iterations. Various schemes are employed in these searches for shortest distances, including [11]:

1) Initialization
   - Define the starting node as the source node.
   - Initialize the distance from the source node to all other nodes with an infinite value.
   - The source node's distance value to itself is set to 0.
   - Determine the set of unvisited nodes.
2) Select the node with the shortest distance
   - From the set of unvisited nodes, select the node with the shortest distance from the source node.
   - Mark this node as visited.
3) Update the distance to unvisited neighbors
   - For each neighboring node of the node under consideration:
   - Compute the new distance from the source node through the node under consideration to the neighboring node.
   - If the new distance is shorter than the previous distance from the source node to the neighbor node, update the distance value.
4) Repeat steps 2 and 3
   - Repeat steps 2 and 3 until all nodes have been visited or until the destination node (if specified) has been visited.
5) Results
   - Once all nodes have been visited, the shortest distance from the source node to all other nodes is recorded.

d. OpenStreetMap

OpenStreetMap, abbreviated as OSM, is an online project that aims to create global maps that are freely and openly available to the public. The project is built entirely by a community of volunteers who contribute by conducting surveys using GPS, generating data from aerial imagery, and collecting and sharing publicly available geographic information [3]. Apart from providing access to digital maps, this platform also allows users to add markers and labels to the map, as well as determine routes and calculate the distance between two location points. This facilitates not only visual access to maps but also provides the ability to plan routes as well as practically measure distances between locations. To make it easier to use OSM, we can use a JavaScript library called Leaflet.js [12].

3. Result and Discussion:

Data collection is done manually by dragging the digital map marker. We created a route point and location on Floor 1 of Nipah Mall Makassar. The first-floor route points can be seen in **Figure 2**.

Figure 2. Route nodes and location in the First Floor

Figure 2 shows eight route nodes and twenty location nodes. Route points are marked with green circles from A to H and locations in red. Once the location data and route points are available, the next step is to calculate the distance between the route and location. At this stage, we apply the Haversine method to calculate the distance between 2 locations. The calculation results using the Haversine method can be seen in **Table 1**.

Table 1. Distance of each node

Edges	Node 1		Node 2		Distance (m)
	Latitude	*Longitude*	*Latitude*	*Longitude*	
A – B	-5,139352	119,449724	-5,139395	119,449858	15,59
B – C	-5,139395	119,449858	-5,139416	119,450196	37,50
C – D	-5,139416	119,450196	-5,138765	119,450457	78,00
D – E	-5,138765	119,450457	-5,138326	119,450615	51,77
E – F	-5,138326	119,450615	-5,138265	119,450357	29,32
F – G	-5,138265	119,450357	-5,138383	119,450110	30,29
G – H	-5,138383	119,450110	-5,138562	119,450250	25,19
G – A	-5,138383	119,450110	-5,139352	119,449724	116,00

In **Table 1**, there are eight edges or sides that connect two points. Next, apply the Dijkstra Method to find the closest route. For example, there is a mall visitor who is at TimeZone and wants to go to the toilet. So, the first thing to do is calculate the distance between the two locations and the nearest node. The results were obtained. The distance between the TimeZone location and point C is 18.28 meters. The distance between the toilet location and point F is 16.28 meters and with point G is 27.36 meters. An image of the distance to each route point and location can be seen in **Figure 3**.

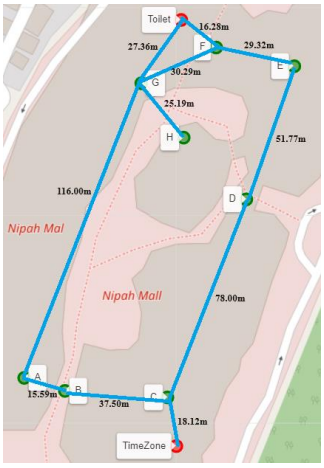

Figure 3. Distance between each location and route node

After determining the distance, the next step is implementing the Dijkstra algorithm. The calculation starts from the TimeZone or starting point which I call node Z to the Toilet or end point which is called point T.

Initialization, starting with the initial node Z. The distance from Z to itself is 0, and the distance to all other nodes is initialized as infinity (∞).

Step 1, node Z is only connected to node C with a distance of 18.12 meters, so the shortest distance selected is from node Z to C.

Step 2, calculations start from the node selected in step 1, namely node C. Node C is connected to two nodes, node B with a distance of 37.5 meters, and node D with a distance of 78 meters. The total distance is obtained by adding the distance of node Z to node C and the distance of node C to the connected node. The result is node Z to C to B is 55.62 meters and Z to C to D is 96.12 meters. The shortest distance from step two is the distance from node C to node B so the node selected for step two is Z to C to B.

Step 3. Calculations start from the node selected in step 2, namely node B. Node B is only connected to Node A apart from the selected node (node C). The total distance from Z to C to B to A is 71.21 meters. This is smaller than the sum of the distances to nodes that have not been selected. So the next selected node is node A.

Step 4. Calculations start from the node selected in step 3, namely node A. Node A is only connected to Node G apart from the selected node (node B). The total distance from Z to C to B to A to G is 187.21 meters. This is greater than the sum of the distances to the node that has not been selected, namely node D. So the next selected node is node D.

Step 5. Calculations start from the node selected in step 4, namely node D. Node D is only connected to Node E apart from the selected node (node C). The total distance from Z to C to D to E is 147.89 meters. This is smaller than the sum of the distances to nodes that have not been selected. So the next selected node is node E.

Step 6. Calculations start from the node selected in step 5, namely node E. Node E is only connected to Node F apart from the selected node (node D). The total distance from Z to C to D to E to F is 177.21 meters. This is smaller than the sum of the distances to nodes that have not been selected. So the next selected node is node F.

Step 7. The calculation starts from the node selected in step 6, namely node F. Apart from the selected node (node E), node F is connected to Node G with a distance of 30.29 meters and T (the final node) with a distance of 16.28 meters. So the total distance from Z to C to D to E to F to G is 207.50. and Z to C to D to E to F to T is 193.49 meters. The distance from Z to C to D to E to F to G is greater than the total distance from Z to C to B to A to G, namely 187.21 meters, so the total distance to G is taken from the results of step four. In this step, a route to the destination has been found but the value obtained is still greater than the node that has not been selected, namely G. So the next selected node is G.

Step 8. The calculation starts from the node selected in step 7, namely node G. Apart from the selected nodes (nodes A and F), node G is connected to Node H with a distance of 25.19 meters and T with a distance of 27.36 meters. So the total distance from Z to C to B to A to G to H is 214.40 meters and from Z to C to B to A to G to T is 193.49 meters. In this step, the route to the destination node is also found. But the total distance to the destination node in step 7 is smaller and there are no more nodes that have not been selected whose total distance is smaller than this result. So the route search result from node Z to T is Z to C to D to E to F to T with a total distance of 193.49 meters. The values for each step can be seen in **Table 2**.

Table 2. Shortest route search results using the Dijkstra algorithm

Step	Nodes	Z	A	B	C	D	E	F	G	H	T
1	Z	0,00	∞	∞	18,12 Z	∞	∞	∞	∞	∞	∞
2	C		∞	55,62 C	18,12 Z	96,12 C	∞	∞	∞	∞	∞
3	B		71,21 B	55,62 C		96,12 C	∞	∞	∞	∞	∞
4	A		71,21 B			96,12 C	∞	∞	187,21 A	∞	∞
5	D					96,12 C	147,89 D	∞	187,21 A	∞	∞
6	E						147,89 D	177,21 E	187,21 A	∞	∞

Step	Nodes	Z	A	B	C	D	E	F	G	H	T
7	F							177,21 E	187,21 A	∞	193,49 F
8	G								187,21 A	212,40 G	193,49 F
Result	T										**193,49 F**

After getting the closest route, the next step is to create a route on the digital map. The following displays the routing from location Z to C to D to E to F to T using the routing machine in **Figure 4**.

Figure 4. Results of using a digital map routing machine indoors

In the results shown in **Figure 4**, we use a digital map routing machine to connect the selected route nodes. But the routing machine always directs out of the indoor area of the mall. This result was not as expected. The expected result should be that the routing machine simply draws a straight line between each route node. So, we replaced this method by using the polyline feature on digital maps. The results of using polylines can be seen in **Figure 5**.

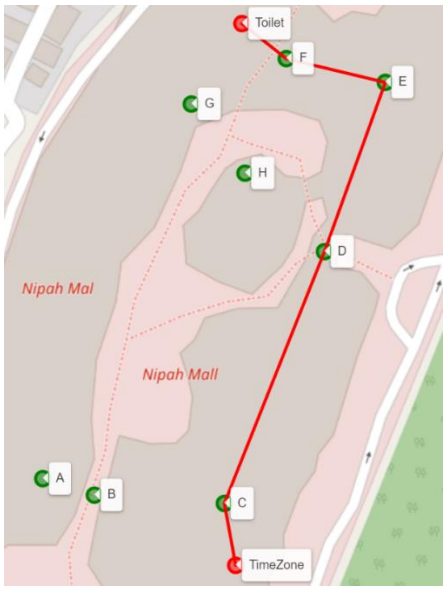

Figure 5. Polyline at route node

In **Figure 5**, the use of a polyline can connect the starting node to the route node that is taken to the final node. So that determining routes in space on digital maps can be done.

4. Conclusion:

Based on the results of this research, it can be concluded that determining the shortest route in an enclosed space is possible. The Haversine method for measuring the distance between nodes and the Dijkstra algorithm for finding the shortest route in a room such as a mall have proven effective in determining the closest route. However, using routing engines on digital maps does not provide optimal results. Therefore, we use the polyline feature as an

alternative to display the route between two locations on a digital map with more satisfactory results. We hope that the results of this research can provide a valuable contribution to the development of effective and accurate route-finding applications indoors.

Acknowledgments

We would like to extend our heartfelt gratitude to Lembaga Penelitian dan Pengembangan Sumberdaya (LP2S) Universitas Muslim Indonesia (UMI) for their invaluable support, facilities, and research funding. Their contributions have been instrumental in propelling this research toward greater strides. Without their assistance, the achievements and progress made in this research would not have been possible. We deeply appreciate the opportunities provided by LP2S UMI and are thankful for their dedication to supporting the advancement of knowledge and innovation.

References:

[1] A. A. Manrang, Y. Salim, and M. A. Asis, "Rancang Bangun Aplikasi Mall Maps Berbasis Mobile Menggunakan Metode Euclidean Distance," *Bul. Sist. Inf. dan Teknol. Islam*, vol. 3, no. 4, pp. 301–310, 2022, doi: 10.33096/busiti.v3i4.1355.

[2] D. Hantono and N. Aziza, "Peran Ruang Publik pada Kantor Rukun Warga terhadap Aktivitas Masyarakat di Kelurahan Kebon Pala Jakarta Timur," *J. Arsit. ALUR*, vol. 3, no. 2, pp. 44–52, 2020.

[3] M. Marsujitullah and M. A. Asis, "Integrasi Peta Digital pada Sistem Informasi Lahan Pertanian Kabupaten Merauke, Indonesia," *Bul. Sist. Inf. dan Teknol. Islam*, vol. 3, no. 1, pp. 1–6, Feb. 2022, doi: 10.33096/busiti.v3i1.1097.

[4] F. Mahdia and F. Noviyanto, "Pemanfaatan Google Maps Api Untuk Pembangunan Sistem Informasi Manajemen Bantuan Logistik Pasca Bencana Alam Berbasis Mobile web," vol. 1, pp. 162–171, 2013.

[5] R. H. D. Putra, H. Sujiani, and N. Safriadi, "Penerapan Metode Haversine Formula Pada Sistem Informasi Geografis Pengukuran Luas Tanah," *J. Sist. dan Teknol. Inf.*, vol. 10, no. 2, pp. 1262–1270, 2015.

[6] A. N. Abadi Nugroho, "Penerapan Metode Haversine Formula Untuk Penentuan Titik Kumpul pada Aplikasi Tanggap Bencana," *Metik J.*, vol. 4, no. 2, pp. 69–75, 2020, doi: 10.47002/metik.v4i2.190.

[7] P. Dauni, M. D. Firdaus, R. Asfariani, M. I. N. Saputra, A. A. Hidayat, and W. B. Zulfikar, "Implementation of Haversine formula for school location tracking," *J. Phys. Conf. Ser.*, vol. 1402, no. 7, 2019, doi: 10.1088/1742-6596/1402/7/077028.

[8] D. Rachmawati and L. Gustin, "Analysis of Dijkstra's Algorithm and A∗ Algorithm in Shortest Path Problem," *J. Phys. Conf. Ser.*, vol. 1566, no. 1, 2020, doi: 10.1088/1742-6596/1566/1/012061.

[9] M. Chatrin Bunaen, H. Pratiwi, and Y. Finsensia Riti, "Penerapan Algoritma Dijkstra Untuk Menentukan Rute Terpendek Dari Pusat Kota Surabaya Ke Tempat Bersejarah," *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 4, no. 1, pp. 213–223, 2022.

[10] C. S. Rahayu, W. Gata, S. Rahayu, A. Salim, and A. Budiarto, "Penerapan Algoritma Dijkstra Dalam Penentuan Lintasan Terpendek Menuju Upt. Puskesmas Cilodong Kota Depok," *J. Tek. Inform.*, vol. 14, no. 1, pp. 81–92, 2021, doi: 10.15408/jti.v14i1.18721.

[11] M. S. Yusuf, H. M. Az-zahra, and D. H. Apriyanti, "Implementasi Algoritma Dijkstra Dalam Menemukan Jarak Terdekat Dari Lokasi Pengguna Ke Tanaman Yang Di Tuju Berbasis Android (Studi Kasus di Kebun Raya Purwodadi)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 12, pp. 1779–1781, 2017.

[12] I. A. Marleni and A. Gunaryati, "Presensi Karyawan Berbasis Web dengan Fitur Lokasi Leaflet JS menggunakan Laravel," *J. JTIK (Jurnal Teknol. Inf. dan Komunikasi)*, vol. 7, no. 3, pp. 479–485, Jul. 2023, doi: 10.35870/jtik.v7i3.947.