# QUALITATIVE ROUTING INSTRUCTIONS IN INDOOR SPACE

Eliseo Clementini[1]*, Valentina D'Orazio[2]

[1] Dept. of Industrial and Information Engineering and Economics
University of L'Aquila, L'Aquila, Italy - eliseo.clementini@univaq.it
[2] Dept. of Information Engineering, Computer Science and Mathematics
University of L'Aquila, L'Aquila, Italy - valentina.dorazio1@student.univaq.it

**Commission IV**

**KEY WORDS:** indoor navigation, routing algorithms, qualitative spatial reasoning, wayfinding, landmarks

**ABSTRACT:**

This paper discusses the generation of routing instructions in indoor space. We develop a user-friendly application that guides a pedestrian to reach a desired destination within a typical building floor in a simply and quickly way. Starting from a suitable navigation graph of the floor, the application considers qualitative reasoning techniques to produce indications such as "go slightly further" or "keep on the right" and adds a semantic level that provides indications on the visible landmarks along the route, such as "continue after the reception desk on the right". This characteristic is fundamental when the users do not have a previous knowledge of the building, as they can be reassured by the presence of landmarks that help to understand if the taken direction is the correct one. The algorithm proposed on this paper is able to generate navigation instructions closer to what would be the typical indications of a human guide. We evaluate the result for some case studies of building floors.

## 1. INTRODUCTION

In recent years, digital road navigation has revolutionized the way of moving from a place to another improving the experience of traveling. Today, even if we perfectly know the path to be followed to reach a destination, we prefer to use an outdoor navigation system because we can verify, for example, driving conditions, active speed cameras or roadblocks in real time. Concerning indoor navigation, systems are less present in society and only prototypical in some contexts (Zlatanova et al., 2020). Notably, a proposal of standard for indoor spatial information has been proposed by the Open Geospatial Consortium (OGC, 2020). The main technological reason of lesser development is that satellite signals are too weak to get inside buildings and the resolution of common hand-held devices would be too low to precisely identify a position in walking space. Actually, indoor localization could work through Wi-Fi networks and mobile telephone base stations, but such an approach fails when there is a high concentration of users between router and receiver (usually a smartphone) due to signal breakups. Other factors that influence the signal can be the presence of metal objects such as shelves or metal armatures (Lymberopoulos et al., 2015).

For these reasons, creating indoor navigation systems requires greater effort than outdoor navigation due to technological issues that may vary in each type of building. Further, indoor navigation has its own peculiarities since it is directed to pedestrians that are moving on a free space, contrary to cars that are moving in a more constrained network. Pedestrians don't directly see a predefined line network where they are obliged to proceed, but they tend to erratically move in open space: corridors are more naturally seen as a one possible direction path, but larger spaces such as halls and open floors might constitute a confusing framework to move and find a direction (Stoffel et al., 2008).

* Corresponding author

In this paper, we do not focus on the technological problem of identifying user position. We assume that it is possible to identify the initial location and that the latter can be matched to a navigation graph. Therefore, a first problem in building an indoor navigation system is to define a good navigation network that can be superimposed to a floor map. There are in the literature many proposals of how extracting the network that can be subdivided in two main approaches: generating the medial axis of the indoor space (Yao, Rokne, 1991, Lee, 2004) or considering the visibility graph (Taneja et al., 2011). There are hence hybrid approaches that tries to take advantages of the two and reduce critical factors (Mortari et al., 2019). We believe that in order to generate human-like directions instructions, we should start from a navigation network that would mimic as much as possible the most likely movements of people inside the indoor space. As a basis, we use the navigation network proposed in (Clementini, Pagliaro, 2020), where a straight skeleton of the map is combined with shortpaths based on visibility between nodes. Further other points of interest, such as doors, can be added to such a network (see an example in Figure 1).

The first contribution we give in this paper is about the algorithms to be applied in the data structure. In particular, (1) the Dijkstra's algorithm for finding the minimum route among all available routes; (2) the change of reference system from the coordinate-base geometric representation to the egocentric reference system of the user ("go straight", "turn right", "turn left"); (3) finding close-by points of interest during navigation, such as the rooms encountered in the aisle in order to confirm that the path being taken is the right one.

After, having all the elements at disposal, we extract the routing directions. In this part, we transform geometric information in qualitative information: distances are expressed with natural language expressions such as "go ahead for a while" or "get past the director office"; directions are expressed as "turn slightly left". Often the intermediate nodes of the network will be too close each other to represent a significant movement

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-VI-4-W2-2020-47-2020  

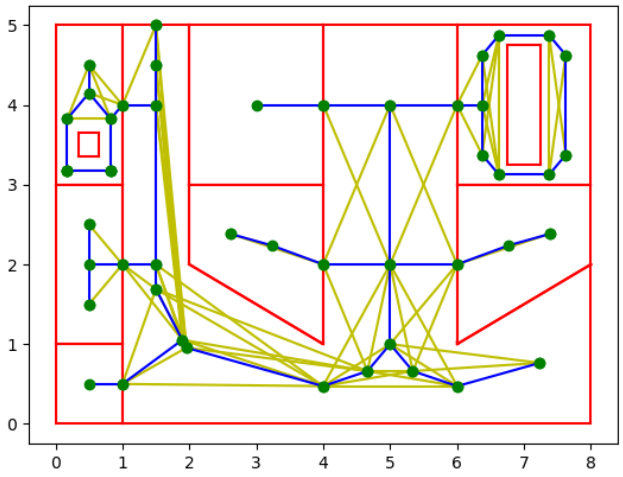

Figure 1. The navigation network for a case study floor. Edges are of two kinds: coming from the straight skeleton (in blue) and from the visibility graph (in green)

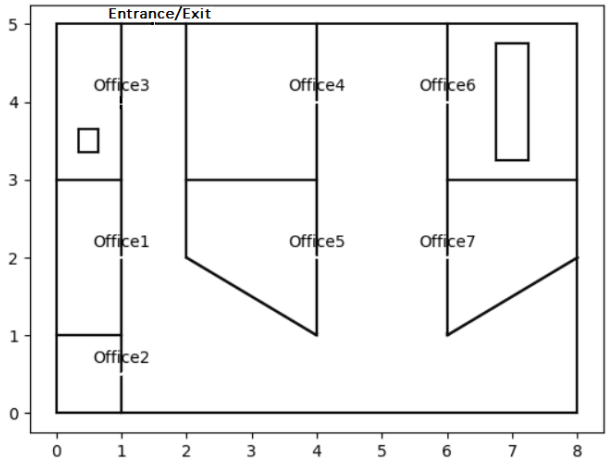


Figure 2. Association between nodes and text names to identify office doors and main entrance

(e.g., nodes at distance of few meters or less): in this case, the suggested routing instruction should be computed as a combination of qualitative distances and directions (Clementini et al., 1997). For example, two consecutive "turn slightly left" could form a "turn left" if the two turning points are very close each other. Combination of qualitative relations are examples of qualitative spatial reasoning (Clementini, 2013, Clementini, Cohn, 2014, Bartie et al., 2013, Fogliaroni, Clementini, 2015).

The problem of generating route directions has already been dealt with in the literature (Richter, Klippel, 2005, Allen, 1997, Dale et al., 2002, Allen, 2000, Cuayahuitl et al., 2010, Russo et al., 2014, Fellner et al., 2017, Zang et al., 2018): the novelty of the proposed approach is the use of a more user-friendly navigation network and the use of qualitative reasoning techniques.

The rest of the paper is organized as follows. In Section 2, we discuss the preliminary part on navigation algorithms. In Section 3, we describe the extraction of routing directions. In Section 4, we describe a pilot implementation of the system. In Section 5, we give some concluding remarks.

## 2. ALGORITHMS AND DATA STRUCTURES

To model possible paths in indoor space, we adopt a graph whose nodes are meaningful positions in the space (entrance, turning points, doors) and arcs are the main connections between nodes. Let us refer as a running example to the floor map in Figure 1. As mentioned before, the navigation graph has been extracted with the algorithms illustrated in (Clementini, Pagliaro, 2020). In this graph, nodes are depicted with blue points: some of these points represent doors (the ones falling on red walls) , while other are simply turning points. Arcs obtained with the straight skeleton are depicted in blue and arcs obtained by the visibility graph are depicted in green. For the scope of navigation, there is no distinction between these two species of arcs.

Efficient navigation implies the minimal effort to reach the destination. Hereafter, we will use the distance between nodes as the measure of effort. The approach can be easily modified to take into account other measures like time of locomotion or the presence of architectural barriers. Therefore, we use the Dijkstra's algorithm (Dijkstra, 1959) to extrapolate the shortest path from a source node to a destination node. In our study case, the nodes of the network that can be reached by navigation are the "door nodes" of the floor. These nodes are associated with a text label with the aim of indicating the meaning of the node in a given building. For example, a door node can represent an office, an entrance or an exit (Figure 2).

The selection of the shorter route is the first step to give indications. Afterwards, we need to transform information on an egocentric frame of reference to discern between right and left directions. Moreover, we need to retrieve information about what surrounds the user along the way.

### 2.1 Dijkstra's algorithm

Dijkstra's algorithm is used to search for shortest paths in a graph with or without ordering, cyclic and with non-negative weights on the arcs. Let us consider a graph with $n$ nodes and two particular nodes, the source node and the destination node. Then, the weight of the arc that joins the nodes $j$ and $k$ is retrieved by the function $dist(j, k)$. The main strategy of the algorithm is to calculate the distances from the source node to its neighbors and to propagate the calculus to neighbors of neighbors and so on, till all nodes are considered. In the pseudocode (see Algorithm 1), the set $Q$ contains the nodes still to be scanned. At the end of the algorithm, each node of the graph will be associated with two data: $totalDist[i]$ indicating the total weight of the shortest path from the source node to node $i$ and $previousNode[i]$ indicating the node preceding the node $i$ in such a shortest path. Therefore, each node will store the information on the distance from the source node and the direction to take to reach it. To identify the shortest path to reach the destination node, we consider the destination node and backtrack all the preceding nodes with $previousNode[]$ till we reach the source node.

For our purposes, the Dijkstra algorithm has been applied to the whole navigation network to find the shortest route from the starting position to the destination. The source node and the destination node correspond to doors of the map. The weight of the arcs correspond to the Euclidean distance between the nodes. The result of the application of the Dijkstra algorithm from the entrance to office 4 is shown in Figure 3.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-VI-4-W2-2020-47-2020 | © Authors 2020. CC BY 4.0 License. 

**Algorithm 1** Dijkstra's algorithm

```
1:  function DIJKSTRA(Graph, sourceNode)
2:      for node v in Graph do
3:          totalDist[v] := infinite;
4:          previousNode[v] := undefined;
5:      totalDist[sourceNode] := 0 ;
6:      Q := set of all nodes in the Graph;
7:      while Q is not empty do
8:          u := node in Q with the shortest totalDist[];
9:          remove u from Q;
10:         if totalDist[u] = infinite then
11:             break;
12:         for neighbour v of u with v ∈ Q do
13:             route := totalDist[u] + dist(u, v);
14:             if route < totalDist[v] then
15:                 totalDist[v] := route ;
16:                 previousNode[v] := u ;
17:     return totalDist[], previousNode[];
```

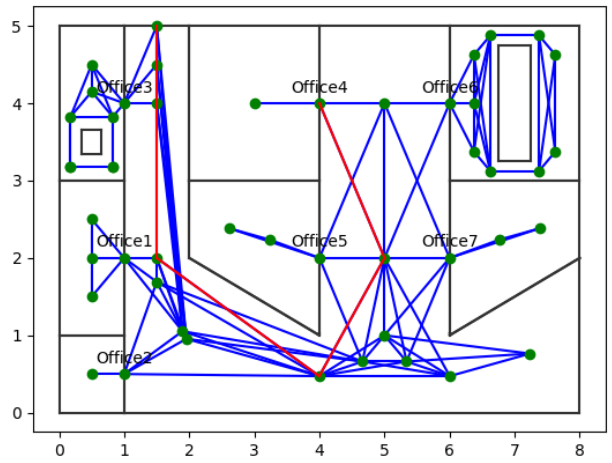


Figure 3. Minimum path to reach office 4 from the entrance as calculated by the Dijkstra algorithm

### 2.2 Finding orientation

Starting from the shortest path found by the Dijkstra algorithm, we need a change in the reference system to find directions from the egocentric point of view of a user moving in the path. To determine right or left turns, we consider three consecutive nodes in the minimum path, $node1$, $node2$, $node3$, and we construct two vectors $\vec{v} = node2 - node1$ and $\vec{u} = node3 - node2$ (see Algorithm 2). Then, we find the oriented angle between those two vectors by the the scalar product. To determine the orientation of this angle, we use the cross product of the two vectors: being in two dimensions, the component of the cross product relative to $k$ is null. The sign of the cross product will be positive for counterclockwise configurations of the three consecutive nodes and negative for clockwise ones (Figure 4). Therefore, from a positive sign we can infer the "turn left" direction and from a negative sign we can infer the "turn right " direction.

### 2.3 Finding the points of interest along the path

The use of landmarks has been widely indicated as a way of improving the cognitive adequacy of navigation instructions in navigation (Richter, 2013). In our approach, we enrich navigation instructions with points of interest that the user meets along the selected path with the purpose of confirming that the user is on the right track. As an example, in our study case we limited the points of interest to nodes representing doors of the building, but any other node representing features of different

**Algorithm 2** Algorithm for orientation and sign

1: **function** CALCULATION OF ORIENTATION ANGLE AND SIGN($node1, node2, node3$)
2: $v = (node2_x - node1_x, node2_y - node1_y, 0)$
3: $u = (node3_x - node2_x, node3_y - node2_y, 0)$
4: $\theta = cos^{-1}[(\vec{u} * \vec{v}) / \|u\| * \|v\|]$
5: $sign = det \begin{bmatrix} i & j & k \\ v_x & u_x & 0 \\ v_y & u_y & 0 \end{bmatrix} = v_x * u_y - v_y * u_x$
6: return angle, sign

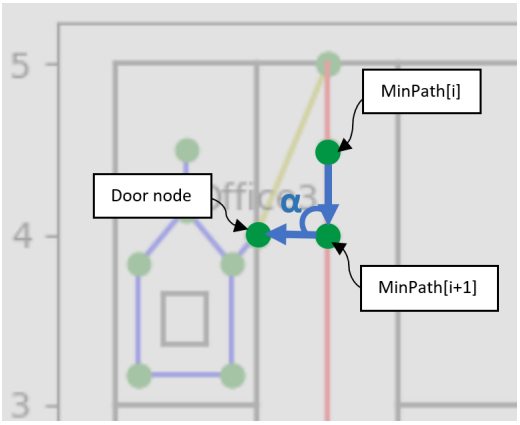


Figure 4. Cross product of two vectors in three consecutive nodes of the network to determine turning direction

kind can be easily added to the data structure. A straightforward solution may be implemented by considering a circular buffer around nodes encountered along the path that connects the source node with the destination node to verify the existence of door nodes that fall within the buffers (Figure 5). In our experiment, the radius of the circular buffer was taken bigger than the average width of the corridors, but it can be modified to satisfy the dimensions of the building of interest. In Algorithm 3, we make use of a data structure called the "room-door connection matrix", which indicates for each room the number of doors connecting the room. In each room, we calculate the Euclidean distance from the current node in the path to the set of doors of the room to discover rooms that are near the node. For such doors, we find the orientation along the path, e.g., by saying that Office 3 is visible on the right when moving along the path (see Figure 5).

**Algorithm 3** Find doors close by a node

```
1:  function CALCULATION OF DOORS CLOSE BY A NODE(node, radius)
2:      Find the room where the node is located
3:      for door in room-door connection matrix do
4:          distance= sqrt((x_door − x_node)^2 + (y_door − y_node)^2)
                ▷ distance between the door and the node
5:          if distance < radius and distance > 0 then
6:              calculate sign by Algorithm 2
7:              if sign < 0 then
8:                  door is on the right
9:              else if sign > 0 then
10:                 door is on the left
```

## 3. ALGORITHM FOR PROVIDING INDICATIONS

The qualitative system of directions that we are using is structured as in Figure 6. Coming from node $A$ and reaching node $B$, the next direction to take is one in the set {*right*, *slightly*

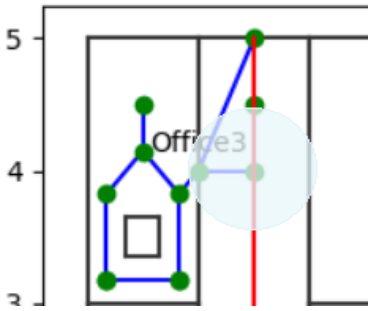


Figure 5. Circular buffer around a current node of the path to establish whether there are rooms (or door nodes) encountered along the way

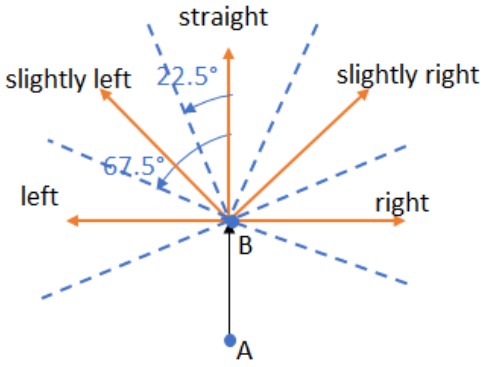


Figure 6. The qualitative system of directions

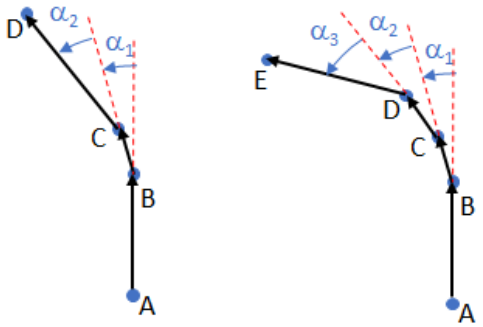


Figure 7. Composition of qualitative directions

*right*, *straight*, *slightly left*, *left*}. We assigned such qualitative directions to an equiangular conic domain. Assuming the angle of the straight direction to be $0^\circ$ degrees and directions to be measured anticlockwise, the qualitative directions are assigned to the intervals $[-112.5^\circ .. -67.5^\circ]$, $[-67.5^\circ .. -22.5^\circ]$, $[-22.5^\circ .. +22.5^\circ]$, $[+22.5^\circ .. +67.5^\circ]$, $[+67.5^\circ .. +112.5^\circ]$.

If the distance between nodes is shorter than a given threshold, we adopt a composition of angles before determining the qualitative direction. The threshold is qualitatively identified as *very close* and as an example to run the algorithms it has been set up to 1 meter. For example, as illustrated in Figure 7, if the distance from node $B$ to node $C$ is *very close*, i.e., $dist(B, C) =$ *very close*, the qualitative direction associated to angle $\alpha_1$ is ignored and the next node $D$ is considered. If $dist(C, D) >$ *very close*, then the angle $\alpha_1 + \alpha_2$ determines the qualitative direction to take. If the node $D$ as well is very close to $C$, then the composition is repeated. Therefore, if $dist(C, D) =$ *very close*, the next node $E$ is considered and the angle $\alpha_1 + \alpha_2 + \alpha_3$ determines the qualitative direction. The process is repeated if the sequence of very close nodes is longer, till an overall distance corresponding to another qualitative distance distinction is reached. We call *close* such a qualitative distance, which corresponds to the minimum sum of *very close* distances for which we give an explicit direction information. In our experiment, the *close* distance has been set up to 2 meters.

**Algorithm 4** Algorithm for providing indications

**function** FINDINDICATIONS($minimumPath$)
  $accAngle \leftarrow 0$ ▷ sum of angles
  $accDist \leftarrow 0$ ▷ sum of distances
  $indications \leftarrow []$
    ▷ list containing [qualitative distance, qualitative direction, points of interest]
  $InitP0(minimumPath)$
    ▷ introduces a first point P0 in minimumPath that is before P1 at distance very close and with the same direction as P1-P2
  $n \leftarrow len(minimumPath)$
  **for** $i = 0, i <= n-2, i++$ **do**
    $p1 \leftarrow minimumPath[i]$
    $p2 \leftarrow minimumPath[i+1]$
    $p3 \leftarrow minimumPath[i+2]$
    $angle \leftarrow signedAngle(p1, p2, p3)$
    $dist \leftarrow distance(p2, p3)$
    **for** doors in room **do**
      ▷ check for doors of the given room
      $POI \leftarrow$ calculation of doors close by ($p3$)
        ▷ Algorithm 3
    $accAngle = angle + accAngle$
    $accDist = dist + accdist$
    **if** $qdist(accDist) =$ very close **then**
      pass
    **else if** $qdist(accDist) =$ close **then**
      insert($indications$, "move a little bit", $qdir(accAngle)$, $POI$)
      $accAngle \leftarrow 0$
      $accDist \leftarrow 0$
    **else if** $qdist(accDist) >$ close **then**
      insert($indications$, $approx(accDist)$, $qdir(accAngle)$, $POI$)
      $accAngle \leftarrow 0$
      $accDist \leftarrow 0$

Algorithm 4 describes the overall technique for constructing the list of indications for navigation. We define two acceptance functions that allow passing from the metric domain to the qualitative domain: the function $qdir$ returns a qualitative value for direction, given the angle measure, and the function $qdist$ returns the qualitative value for distance, given the metric distance. The $mimimumPath$ returned by the Dijkstra algorithm is a list of $n$ vertices. Initially, to this list is added a fictitious point $P0$ that is at distance *very close* to the first vertex $P1$ and with the same direction as the first arc $P1, P2$. The fictitious vertex $P0$ allows the algorithm to find the initial direction to follow, without influencing the distance and angle of $P1, P2$. The list of indications for navigating the network is made up of triples (distance information, orientation information, points of interest). Therefore, the distance information is skipped if it is up to *very close*. Then, it is "move a little bit" in the case the distance range is up to *close*. For longer distances, the actual metric distance is transformed by a function $approx$ that returns an approximated expression such as "about 5 meters" or "about 10 meters".

Examples of triples in the returned $indications$ list could be:

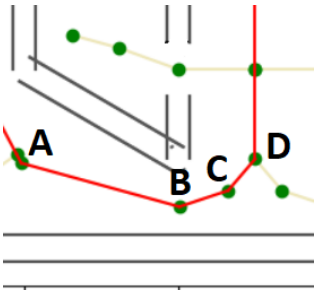


Figure 8. Sequence of very close distances

- ("move a little bit", "right", "Office 3 at right")
- ("about 5 meters", "straight", "Office 5 at left")
- ("about 10 meters", "slightly left", "Office 4 at right")

Correspondingly, we designed a module that reorganize the previous list in more suitable natural language expressions, which for the same examples would return the following sentences:

- move a little bit on the right till you see Office 3 on your right;
- go straight for about 5 meters till you see Office 5 on your left;
- turn slightly on the left and proceed for about 10 meters till you see Office 4 on your right.

The qualitative distinctions that have been adopted for distances are essentially three: the *very close* distance that represents a distance that cannot reasonably correspond to an indication of movement for the moving agent. For example, if moving by foot, something shorter than a step must be disregarded. Then, the distance *close* represents a distance that can correspond to an actual movement, but it is not perceived as a remarkable movement or it does not require sensible effort by the moving agent. Finally, the negation of *close* is any distance bigger than *close* that definitely requires a movement in a given direction. For *non-close* distances, we preferred the adoption of a semi-quantitative space, translating the distance measurements to expressions of the kind *about 5 meters*, *about 10 meters*, and so on.

The absorption of small distances in larger ranges is necessary to avoid false orientation indications as well. Consider the example in Figure 8 with three nodes $B$, $C$, $D$ at *very close* distance each other, depending on the curvature, the individual orientation indications would correspond to *slightly left* or even *straight*, while adding up the *very close* distances to the next significant range we would obtain a single indication of turning *left*, which is less ambiguous for the user.

The application of Algorithm 4 in the case of the path from the entrance to Office 4 (Figure 3) is the following indications list:

- ("move a little bit", "straight", "Office 3 at right")
- ("about 2 meters", "straight", "Office 1 at right")
- ("about 2 meters", "slightly left", "no POI")
- ("about 2 meters", "left", "Office 5 at left")
- ("about 2 meters", "slightly left", "Office 4")

Correspondingly, the indications are rearranged as follows:

- go straight a little bit till you see Office 3 on your right;
- go straight for about 2 meters till you see Office 1 on your right;
- turn slightly on the left and proceed for about 2 meters;

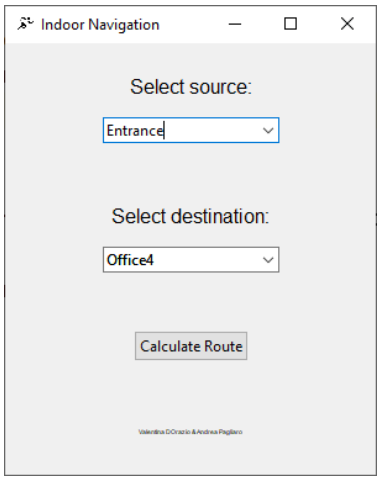


Figure 9. User dialog that prompts the user to choose source and destination

- turn left and proceed for about 2 meters till you see Office 5 on your left;
- turn slightly on the left and proceed for about 2 meters till you reach Office 4.

Notice that the last indication needed some modification since it corresponds to the reached destination.

## 4. IMPLEMENTATION

The algorithm has been tested in Python by making use of the following libraries:

1. **Networkx**: a Python library for the management of graphs and networks. Notably, it contains an implementation of the Dijkstra's algorithm;
2. **Matplotlib**: a library for plotting graphics in association with the NumPy mathematical library;
3. **Tkinter**: a library for creating graphical user interfaces (GUIs). Although several libraries for GUIs exist, Tkinter still maintains a primacy of lightness and stability.

The implementation requires in input the coordinates of each room and the position of doors. The user dialog (Figure 9) allows the selection of a path from the source point to the destination point. By clicking on "Calculate Route", a window opens for viewing the route to follow (e.g., the one in Figure 3) and another one with a description of the steps to be taken to reach the destination (Figure 10). The indications provide concrete references inside the building typical of a human guide, as looking at the path with the user's eyes. Expressions like "go straight till you reach Office 1 on your right" makes the direction unambiguous and comforts users about being in the right direction.

## 5. CONCLUSION

The research and use of outdoor positioning and navigation technologies have seen constant and exponential growth. Based on this success, there have been attempts to implement these technologies in indoor spaces, leading to numerous studies. Indoor navigation applications have great potential to encourage pedestrians interaction and orientation with mobile devices;

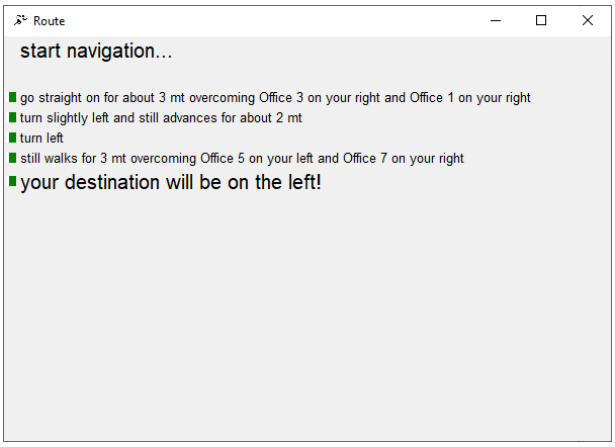


Figure 10. Indications window returned to the user

hence, a user-friendly solution is essential for their success (Sakpere et al., 2017).

For this reason, the paper proposes a high-level solution for indoor navigation problems independent of the technology used, where in addition to the typical geometric indications, it offers concrete references within the building to help the pedestrian to move correctly inside an unknown place.

In a navigation system, it is essential to know the starting point and the destination point of the pedestrian and the information necessary to determine the best route from the initial position to the destination. Starting from a plan of a building and from a graph, it is possible to determine the orientation through the algorithms proposed in this paper, which returns the basic indications (such as "go straight", "turn slightly to the left", and so on) and it is possible to identify the landmarks along the route. Qualitative reasoning techniques improve the result.

The algorithms were implemented in Python language with the help of external libraries.

## REFERENCES

Allen, G. L., 1997. From knowledge to words to wayfinding: Issues in the production and comprehension of route directions. *International Conference on Spatial Information Theory: A Theoretical Basis for GIS, COSIT '97*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 363–372.

Allen, G. L., 2000. Principles and practices for communicating route knowledge. *Applied Cognitive Psychology*, 14(4), 333–359.

Bartie, P., Clementini, E., Reitsma, F., 2013. A qualitative model for describing the arrangement of visible cityscape objects from an egocentric viewpoint. *Computers, Environment and Urban Systems*, 38(1), 21-34.

Clementini, E., 2013. Directional relations and frames of reference. *GeoInformatica*, 17(2), 235-255.

Clementini, E., Cohn, A., 2014. RCC*-9 and CBM*. M. Duckham, E. Pebesma, K. Stewart, A. Frank (eds), *GIScience 2014: Geographic Information Science*, Lecture Notes in Computer Science, 8728, Springer, Cham, 349–365.

Clementini, E., Di Felice, P., Hernández, D., 1997. Qualitative Representation of Positional Information. *Artificial Intelligence*, 95(2), 317–356.

Clementini, E., Pagliaro, A., 2020. The construction of a network for indoor navigation. *6th International Conference on Geographical Information Systems Theory, Applications and Management, GISTAM 2020*, SCITEPRESS - Science and Technology Publications, 254–261.

Cuayahuitl, H., Dethlefs, N., Richter, K.-F., Tenbrink, T., Bateman, J., 2010. A dialogue system for indoor wayfinding using text-based natural language. *International Journal of Computational Linguistics and Applications*, 1(1-2), 285–304.

Dale, R., Geldof, S., Prost, J.-P., 2002. Generating more natural route descriptions. *Proceedings of the 2002 Australasian natural language processing workshop*, 41–48.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), 269–271.

Fellner, I., Huang, H., Gartner, G., 2017. "Turn Left after the WC, and Use the Lift to Go to the 2nd Floor" — Generation of Landmark-Based Route Instructions for Indoor Navigation. *ISPRS International Journal of Geo-Information*, 6(6), 183.

Fogliaroni, P., Clementini, E., 2015. Modeling visibility in 3D space: A qualitative frame of reference. M. Breunig, M. Al-Doori, E. Butwilowski, P. Kuper, J. Benner, K. Haefele (eds), *9th International 3DGeoInfo Conference, 2014*, Lecture Notes in Geoinformation and Cartography, Springer, Cham, 243–258.

Lee, J., 2004. A spatial access-oriented implementation of a 3D GIS topological data model for urban entities. *GeoInformatica*, 8(3), 237-264.

Lymberopoulos, D., Liu, J., Yang, X., Choudhury, R. R., Handziski, V., Sen, S., 2015. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, ACM, New York, NY, USA, 178–189.

Mortari, F., Clementini, E., Zlatanova, S., Liu, L., 2019. An indoor navigation model and its network extraction. *Applied Geomatics*, 11(4), 413-427.

OGC, 2020. IndoorGML: OGC Standard for Indoor Spatial Information. http://www.indoorgml.net/ (accessed July 6, 2020).

Richter, K.-F., 2013. Prospects and challenges of landmarks in navigation services. M. Raubal, D. Mark, A. Frank (eds), *Cognitive and Linguistic Aspects of Geographic Space*, Lecture Notes in Geoinformation and Cartography, Springer, Berlin, Heidelberg, 83–97.

Richter, K.-F., Klippel, A., 2005. A model for context-specific route directions. C. Freksa, K. M., B. Krieg-Brückner, B. Nebel, T. Barkowsky (eds), *Spatial Cognition IV. Reasoning, Action, Interaction. Spatial Cognition 2004*, Lecture Notes in Computer Science, 3343, Springer, Berlin, Heidelberg, 58–78.

Russo, D., Zlatanova, S., Clementini, E., 2014. Route directions generation using visible landmarks. *6th ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness, ISA 2014*, 1–8.

Sakpere, W., Adeyeye Oshin, M., Mlitwa, N., 2017. A State-of-the-Art Survey of Indoor Positioning and Navigation Systems and Technologies. *South African Computer Journal*, 29, 145.

Stoffel, E.-P., Schoder, K., Ohlbach, H. J., 2008. Applying hierarchical graphs to pedestrian indoor navigation. *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, ACM, New York, NY, USA, 419–422.

Taneja, S., Akinci, B., Garrett, J. H., Soibelman, L., East, B., 2011. *Transforming IFC-Based Building Layout Information into a Geometric Topology Network for Indoor Navigation Assistance*. ASCE, 315–322.

Yao, C., Rokne, J., 1991. A straightforward algorithm for computing the medial axis of a simple polygon. *International Journal of Computer Mathematics*, 39(1-2), 51-60.

Zang, X., Vázquez, M., Niebles, J. C., Soto, A., Savarese, S., 2018. Behavioral indoor navigation with natural language directions. *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, Association for Computing Machinery, New York, NY, USA, 283–284.

Zlatanova, S., Yan, J., Wang, Y., Diakité, A., Isikdag, U., Sithole, G., Barton, J., 2020. Spaces in Spatial Science and Urban Applications—State of the Art Review. *ISPRS International Journal of Geo-Information*, 9(1), 58.