Original Research Paper

Cognitively Inspired Algorithm for Imprecise Navigation

[1,2]Melissa Shahrom and [2]Zalilah Abd Aziz

[1]*Department of Infrastructure Engineering, University of Melbourne, Australia*
[2]*Universiti Teknologi MARA Selangor, Malaysia*

Article history
Received: 26-05-2016
Revised: 12-07-2016
Accepted: 13-07-2016

Corresponding Author:
Melissa Shahrom
Department of Infrastructure Engineering, University of Melbourne, Australia and Universiti Teknologi MARA Selangor, Malaysia
Email: melissashahrom@gmail.com

Abstract: This paper presents an algorithm, namely the private navigation algorithm. The aim of this algorithm is to bridge the gap between high quality navigation services and low quality of location information or imprecise data. Generally, the imprecision is due to the poor positioning technology and the algorithms use to protect location privacy. The benefits of the algorithm are at least two-fold: Firstly, it provides an efficient instructions for navigation under imprecision and secondly it supports location privacy protection while using navigation services. In common navigation systems, the navigation instructions generated are based on geometry oriented representation, e.g., shortest path which is based on the distance travelled and normally involves many turns. In human wayfinding, the navigation instruction is considered efficient if the instruction can reduce the cognitive load during the wayfinding activities as well as can guide users to a destination. The algorithm applies the simplest path computations for generating simple navigation instructions due to its ability to minimize the complexity of communicating the instructions. The research examines the efficiency of the algorithm based on several performance measurers. The research also takes into account the wayfinding heuristics such as the initial orientation and agent's behavior (passive or active), that possibly can improve agent's navigation performance. The cognitively motivated simplest cardinal direction weighting function is introduced which reflects the complexity of communicating cardinal instructions. The results show that the private navigation algorithm was efficient when it is incorporated with wayfinding heuristic for imprecise navigation.

Keywords: Cognitive, Privacy, Private Navigation Algorithm, Simplest Path, Efficient Algorithm, Wayfinding Heuristic, Cognitive Wayfinding Instruction

Introduction

This research presents a study of navigation under imprecision in the computerized agent environment. Wayfinding and navigation can be defined as a spatial problem solving. It is knowing where you are in an environment, where your intended destination is and knowing how to get there from your current location. In navigation systems, the imprecision refers to two different reasons; which are due to poor positioning technology and for protecting location privacy.

Many studies of human navigation and wayfinding exist in the literature such as in (Hochmair and Karlsson, 2005; 2000; Golledge, 1999) as well as the studies of the role of human cognition in navigation (Wiener *et al.*, 2009; Hochmair and Karlsson, 2005). A diversity of navigation models has been proposed in the literature dealing with different aspects of human navigation, comprising route descriptions (Westphal and Renz, 2011; Haque *et al.*, 2007; Richter and Duckham, 2008), providing landmarks (Duckham *et al.*, 2010; Michon and Denis, 2001; Caduff and Timpf, 2005) and also planning and survey knowledge (Werner *et al.*, 1997; Goldin and Thorndyke, 1982).

Some studies have indicated that, not only the total length of the route is important for human navigation, but the complexity of navigation instruction also plays an important role. According to Streeter *et al.* (1985), in verbal instructions, routes that are easier to describe and follow are in favor as compared to the overall length of a route. As reported in Golledge (1995), several criteria used in human route selection are ranked highly by human subjects such as the shortest distance, least time as well as the number of turns.

In cognitive wayfinding, the navigation instruction is considered efficient if the instructions can reduce the cognitive wayfinding load during the wayfinding activities in order to prevent navigation errors and guide users to a destination. Reducing cognitive wayfinding load is defined as reducing complexity of route instructions, reducing complexity of identifying decision points and computing efficient route for navigation. In short, the cognitive wayfinding refers to easy-to-follow routes and less complex navigation instructions.

There are very few research on imprecise navigation involving computer agents have been discussed in the literature, especially in the evaluation of efficient navigation instructions. One of the works on imprecise navigation is an imprecise navigation algorithm as introduced in Duckham *et al.* (2003) which uses the shortest path for constructing navigation instructions. It has been reported in their work that, this algorithm is able to generate navigation instructions with imprecise location information. The factor of providing efficient and simple navigation instructions for imprecise navigation is not taken into account and investigated in Duckham *et al.* (2003). Despite, they used common shortest path calculations for generating the paths and instructions. In common shortest path algorithms, a cost function is applied that is related to the graph's structure in its embedding geographical reference frame, such as the distance between nodes, speed of movement, or direction of travel (Brunye *et al.*, 2015). By doing this, the structure of the road network is often ignored, as its aim is to get the shortest path to the destination.

Although the former imprecise algorithm (Duckham *et al.*, 2003) has been proven to be reliable in the past research, the private navigation algorithm is developed in order to improve the performance of imprecise navigation by considering the cognitive aspect in the route computation. The aim of the private navigation algorithm is to bridge the gap between high quality navigation services and low quality of location information.

Graph Representation

Conventional navigation systems are developed based on Dijkstra algorithm which is known as the shortest path algorithm. The algorithm uses metric distances in order to identify the shortest route between an origin and a destination (Agarwal and Gupta, 2014). Metric distance is a simple measure that is used to identify the optimum path in conventional navigation systems. Recent studies have shown that minimizing turns is also an essential factor determining the route choice of users despite the shortest distance (Shahrom, 2013). Cognitively, users usually incorporate several criteria with the distance for path selection, such as the fastest route, the route with least ambiguous and the easiest route. However, these route computations cannot be developed in one simple algorithm (Shahrom, 2013). Numerous previous studies have worked on generating suitable routes for various purposes. A common approach is to apply graph search on a route network to obtain one suitable path and then to describe that path.

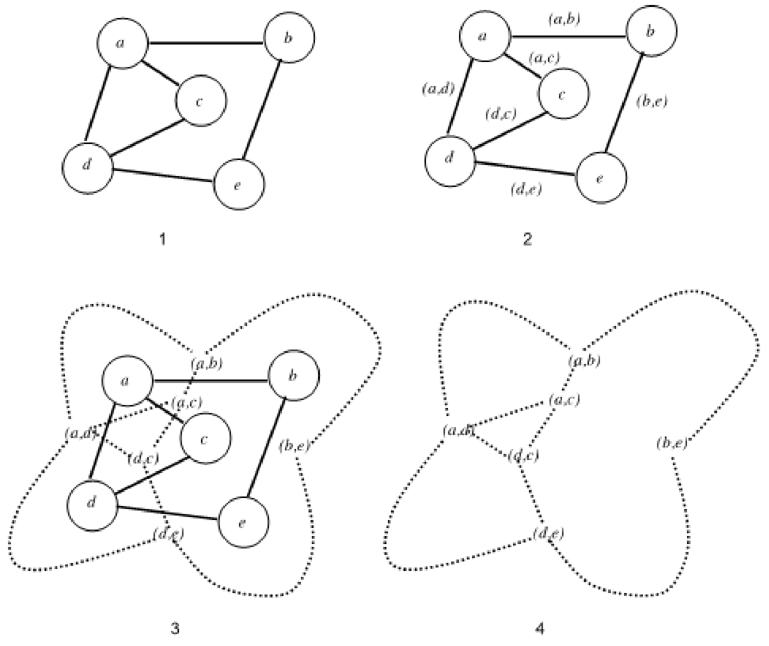


Fig. 1. Line graph construction

The graph usually represents the road's spatial information of locations and lengths, where intersections are nodes, streets are edges and streets lengths are weights. This representation is a conventional approach and has been used in several recent analyses of road networks. The navigation instruction that is based on geometry oriented representation usually generates instructions with many turns due to lack of topological analysis.

The concept regarding navigation in a graph, for informing users when to change directions and into which direction to turn next can be represented by a line graph. The line graph is used to represent each street as node and the street intersections are represented as edges. A line graph $L(G)$ is a graph in which each node of $L(G)$ represents an edge of G and two nodes of $L(G)$ are adjacent if their corresponding edges share a common node in G. The construction of a line graph is illustrated in Fig. 1:

- A graph G
- Nodes in $L(G)$ constructed from edges in G. Each node of the line graph is shown labeled with a pair of nodes of the corresponding edge in the original graph
- Added edges in $L(G)$, e.g.,: The $L(G)$ node labeled (a,c) corresponds to the edge between nodes a and c. $L(G)$ node (a,c) is adjacent to three other nodes: (a,d) and (a,b) (corresponding to edges sharing the node a) and (d,c) (corresponding to an edge sharing the node c)
- The line graph

Unlike the geometry oriented graph, edges cost of the line graph is always set to 1. Therefore, the minimum number of turn can easily be determined where it refers to the shortest topological distance. Moreover, different cost can also be applied such as to represent the complexity of describing the navigation instructions.

Cognitive Cost: Minimizing Navigational Complexity with Simplest Path

The private navigation algorithm implements the simplest path computation for generating cognitively simple navigation instructions due to its ability to minimize the complexity of communicating the instructions. In the context of personal navigation systems, the assumptions are:

- Users have wireless devices (e.g., mobile phones or PDAs) that are online via some form of wireless communication network
- Users are assumed to be able to obtain their positions using GPS technology
- Each mobile object store locally its position and only reveals its imprecise location (as a region)

Figure 2 shows the process in the private navigation algorithm. The dotted line arrows and boxes show the cognitively applied strategies for producing and interpreting route instructions for imprecise navigation. The algorithm incorporate the cognitive cost and wayfinding heuristic in the navigation process.

The routing algorithms and computer-based agent simulation environment have been developed using Java. The algorithm starts with determining the imprecise region (obfuscated) O and the desired destination d. The imprecision in this context is achieved by the number of nodes in the obfuscated region (Duckham and Kulik, 2005). A set of nodes s' is taken into account (as obfuscated region) instead of a single node from where the agent is located.

A graph G comprises a set of nodes V and edges E connecting those nodes. A weighted graph has a function $w:\varepsilon \rightarrow R^+$ associating a weight with each edge $e \in E$. Simplest path (Duckham and Kulik, 2003) is different in terms of its weighting function where it associates a weight with each pair of connected edges rather than each edge in the graph, $w:\varepsilon \rightarrow R^+$ where $\varepsilon = \{((v_i,v_j), (v_j,v_k)) \in E \times E\}$ (Fig. 4).

The idea is that, the weight is based on the complexity of information required to negotiate the decision point represented by the edge pair such as a path from v_i to v_k through intersection v_j.

A set O is a representation of an imprecise location of a user such that $s' \in O$ and $O \subseteq V$. By providing a larger obfuscation set, the spatial resolution of a location is reduced, making the users location imprecise. User's location in O is only imprecise and not inaccurate, such that in O definitely contain s'.

For navigation services, the drawbacks of having the location obfuscated or imprecise are:

- If the density of nodes is too high, the quality of service would be compromised, however
- If the density of nodes is too low, the probability of being located in that location is very high, thus the privacy protection technique applied in the privacy-aware algorithm is ineffective

Computing the simplest path instead of the shortest path enables the algorithm to generate simplest navigation instructions, due to its ability to minimize the instruction complexity. The steps involve are:

- An algorithm selects element of $s' \in O$ and $(s',v_i) \in E$
- A weight of $w(e)$ is assigned for each pair of connected edges $e \in E$ to represent the complexity of information to communicate instruction
- Path computation is based on the shortest path calculation by using simplest path cost given by the graph

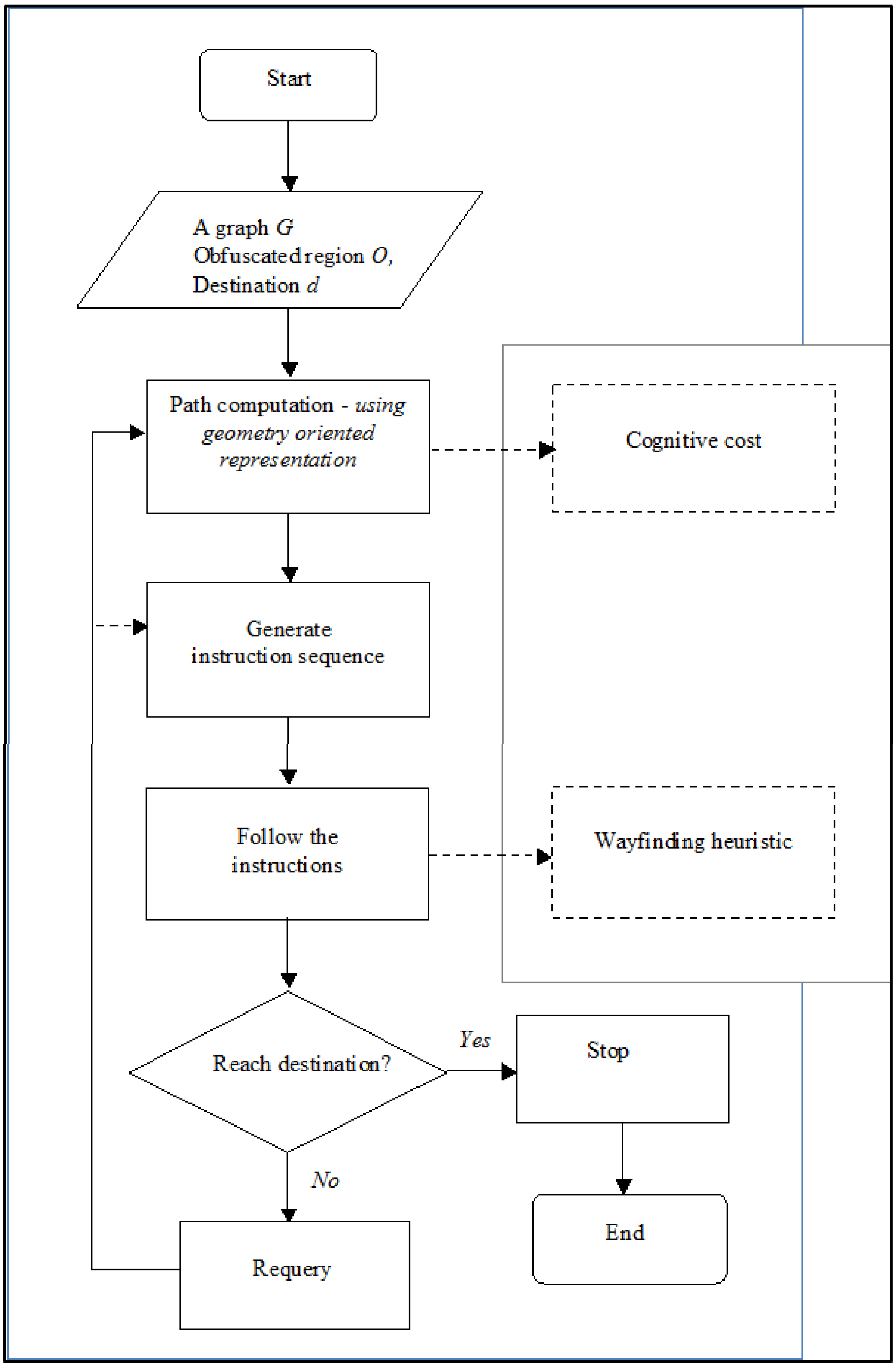


Fig. 2. The dotted line arrows and boxes show the improved strategies for producing and interpreting route instructions

```
Define input: A graph $G = (V, E)$ is a connected, simple and directed graph with the obfuscated location of the client $O \subseteq V$ with an agen imprecise location $s' \in O$ and $d \in V$ is the destination node; $\mathcal{E}$ is the set of pairs of directed edges that share their middle node, $\mathcal{E} = \{((v_i, v_j), (v_j, v_k)) \in E \times E\}$; $w : \mathcal{E} \to \mathbb{R}^+$ is the graph weighting function; $c_s : E \to \mathbb{R}^+$ stores the weights of the simplest path from obfuscated nodes $s'$ and $S$ is a set of visited edges; $ce$ holds the number of executable instruction from $s'$; $Q$ stores the instructions.

INITIALIZATION:
Initialize $c_s(e) = \infty$ for all $e \in E$
Define region $O$ and $s' \in O$

PATH COMPUTATION:
while $s' \neq d$ do
    for all $s' \in O$ do
        for all $(s', v_i) \in E$ do
            set $c_s(s', v_i) = 0$
        while $|E \backslash S| > 0$ do
            Find $e \in E \backslash S$ such that $c_s(e)$ is minimized
            Add $e$ to $S$
            for all $e' \in E \backslash S$ do
                if $(e, e') \in \mathcal{E}$ then
                    set $c_s(e') = \min(c_s(e'), c_s(e) + w(e, e'))$
        Generate the simplest path
        $td = d$
        while $td \neq s'$ do
            find $(v_i, td) \in E$ where $c_s(v_i, td)$ is minimum
            prepend node $v_i$ to path $p = (v_0, v_1, ...v_n)$
            Set $td = v_i$
            Generate instruction sequence based on $p$, where $q' = a_0 a_1 ... a_{n-1}$

            SELECTION OF INSTRUCTION:
            $f$=0;
            for all $s' \in O$ do
                $ce = s' \in O$ —$q'$ executed from $s'$ terminates at $d$
            if $|ce| > f$ then
                $Q = q'$ and $f = |ce|$
```

Fig. 3. Private navigation algorithm

The algorithm (Fig. 3) sets all edges in the obfuscated region with zero weight. The route generation is based on the shortest path calculation by using simplest path cost given by the graph. The algorithm uses the single-source shortest path computation for generating simplest path from all nodes in obfuscated region s' and calculate the minimum cost of going from s' to other nodes in the graph through multiple edges to get the simplest path. To generate the simplest path p, the algorithm iterate backwards through the edges v_t and choosing the edges $(v_t, td) \in E$ where $c_s(v_t, td)$ is minimum. Based on p, algorithm generates instruction q' for all $s' \in O$ to d, where a is the instruction and 0,1........n-1 is the sequence, n is the number of instruction.

The algorithm uses a priority queue data structure. In priority queue, a node with high priority is served before a node with low priority. If two nodes have the same priority, they are served according to their order in the queue. It removes the node that has the path with the lowest discovered weight to s. This lowest discovered path will change as the graph is explored:

- Set the cost of going to s' = 0 and the cost of going to every other edge to 1
- Let (s', v_i) be equal to e
- For each unvisited neighbor e' of e; if the cost of going to e and then directly to e' is smaller than the currently known minimum cost of getting to e', update the cost of getting to e' to be the cost of getting to e plus the cost of getting from there to e'
- Mark e as visited and put in S. (The distance associated with it is now final and minimal)
- If there are no more unvisited nodes, stop and return
- Let e' be equal to the next-smallest tentative distance and go to step 3

Figure 5 illustrates the process of selecting and generating the navigation instruction in the private navigation algorithm. Assume that there are three nodes in the obfuscated set O, $s' = \{v_1, v_2, v_3\}$.

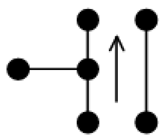
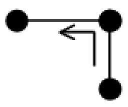
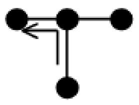
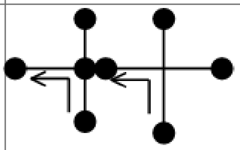
straight on		1 slot
turn (not at the intersection)		4 slots
turn left or right at T-intersection		6 slots
Turn left or right at intersection		5 + deg (v) slots

Fig. 4. Simplest path weight (Duckham and Kulik, 2003)

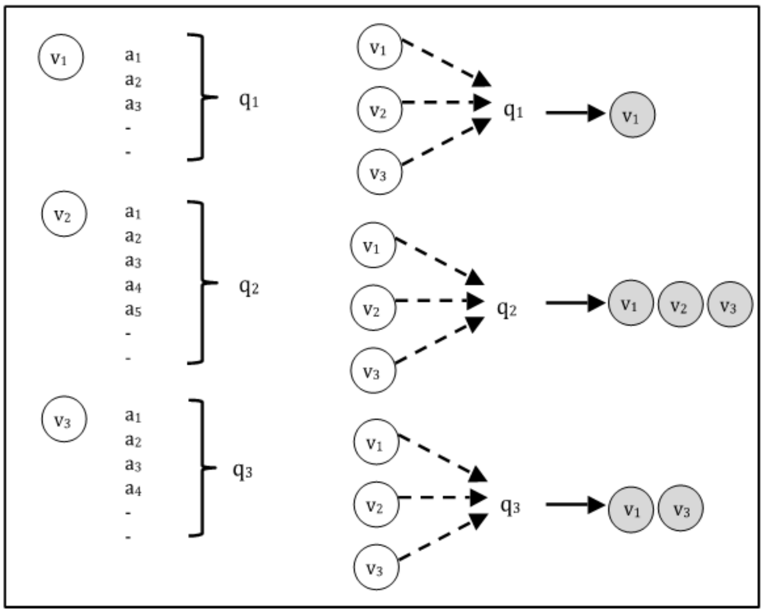


Fig. 5. Generating instruction sequence

Each of these nodes has a series of navigation instructions, a_1, a_2, ...a_n with $q' = \{q_1, q_2, q_3\}$ where q' is the instructions generated based on the simplest path computation from all s' to d. For example, the instruction sequence from v_1 to d is q_1 which consist of $\{a_1, a_2, a_3, ...a_n\}$ where n is the number of instruction. Three sets of simplest path instructions are generated, q_1, q_2 and q_3. The algorithm executes these instructions one by one from all nodes in the obfuscated set; v_1,v_2 and v_3. The one with most frequently leads to destination (in this illustration, q_2 is the most successful instruction sequence with 100% success) is chosen as a navigation instructions and is placed in Q.

Finally, the instruction Q is used as the navigation instruction from obfuscated region O to destination d. For one-time query (static) analysis, once the instruction sequence is finalized, an agent follows the instruction sequence and the algorithm is terminated once the destination is detected and stopped when the instruction is not executable. In a real wayfinding task, successful wayfinding corresponds to the agent's ability to reach destination from a start by following a sequence of instructions given.

Computational Complexity

This section explains the time complexity of the algorithm in terms of how long the program runs. The efficiency of the algorithm depends on how much time it takes to execute and provide relevant result by analyzing the time with respect to increase in input elements. In the private navigation algorithm, the statement corresponds to the extract minimum operation of Dijkstra's algorithm takes $2|E|$ steps as it has to check all the edges of the graph once, to get the value of minimum cost. The private navigation algorithm runs in $O(|E| + |V| \log|V|)$ where E is the number of edges, due to the implementation of priority queue in the Dijkstra's operations. The private navigation algorithm has the time complexity issues as contributed by the path computation.

The repetitive process is used for computing the minimum complexity of every edge connected to the selected edges. In order to compute the complexity of an intersection, the algorithm needs to know the orientation of all other edges connected to the selected edge which leads to $2|E|$ steps. Both operations happen $|E|$ times, which leads to a total number of $|E|(2|E|+2|E|)$ steps. However, since geographical networks are sparse graphs with a small limited number of roads at an intersection, the number of steps reduces to approximately $|E|(2|E|)$, which leads to a time complexity of $O(|E|^2)$. The operation of the simplest path algorithm can be seen as a mapping from the original graph G to a graph $G' = (E', \varepsilon)$, where E' is the set of edges E [17]. In the worst case which requires a totally connected graph, the graph G' could have as many as $|E| = V(|V|\text{-}1)$ edges, leading to a complexity of $O(|V|^4)$ for the simplest path algorithm. Since most of geographical networks can be considered to be planar graphs, which have a maximum number of $|E| = 3(|V|\text{-}2)$ edges, the complexity of the simplest path algorithm is the same as shortest path algorithm which is $O(|V|^2)$.

Finally, the algorithms iteratively simulate the navigation instruction in order to choose the most efficient navigation instruction that can bring most agents to destination. The private navigation algorithm have a time complexity of $O(|V|^2)$ because the time execution is directly proportional to the size of the obfuscated region.

Experimental Setting

A Victoria, Australia road network dataset was used in the simulation. The experiments aimed to improve the navigation performance by adding heuristics in wayfinding such as to perform the initial orientation, using cardinal direction for communicating instructions and agent responses when following the instructions. A simple instruction set was used in order to test the algorithm, comprising only basic directions; *straight*, *right* and *left*, for relative directions. The obfuscated (imprecise) regions were then chosen randomly based on the number of nodes (obfuscated level) in the region. The level of obfuscation starts with a single node to 6 nodes in a set, then, extended to a larger size of 10, 15, 20 and 25 nodes in order to see the difference in the performance. Each algorithm performed 100 iterations per experiment (10 start regions and 10 destination ×10 obfuscation level) and their success rates and stopping distances were recorded.

Improving Imprecise Navigation Performance with Initial Orientation

Spatial orientation is the ability to establish a position in space relative to a particular destination. The initial orientation of a user from the start node to the destination is important for successful navigation especially for imprecise navigation, in which perhaps the user can make correct navigation decisions when following the navigation instructions. In this research, the initial orientation information is given to users by computing a direction from a start node to a destination node. The basic entity of the initial orientation is a position of a destination given by a coordinates. An initial orientation is computed, given (s',d) where the origin is at s' (a node in obfuscated set) and heading to d.

The initial orientation is the heading of an agent from the origin, which refers to which edge should the agent, has to start with. Routing algorithms provide initial orientation information and the agent chooses which edge to start based on the minimum bearing from the origin to the destination. In Fig. 6, e_2 is chosen due to a_1 has the smallest angle from the origin to the destination. Without initial orientation knowledge, agents choose the starting edge randomly.

The result from the analysis shows that, agents with the initial orientation (*SimpRelOP*) perceived higher success rate as compared to agents without initial orientation knowledge (*SimpNP*). With initial orientation, *SimpRelOP* achieved 100% success rate as compared to *SimpNP* (0%) at level 1 (Fig. 10).

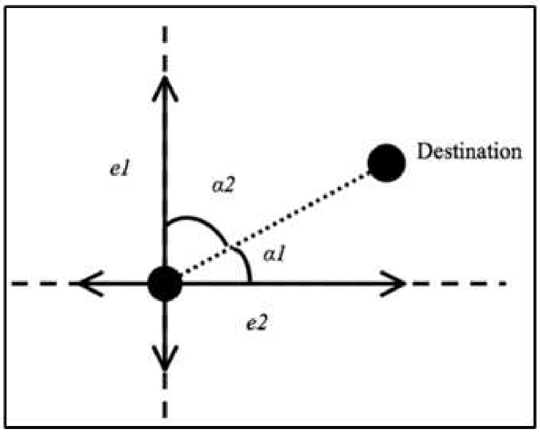


Fig. 6. Initial orientation with least-angle strategy (Hochmair and Karlsson, 2005)

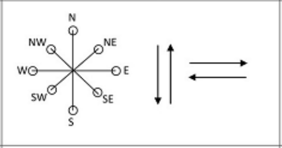
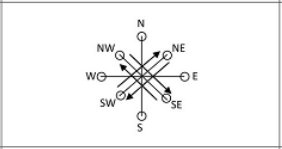
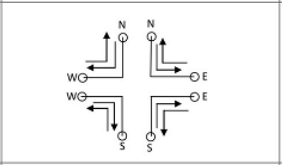
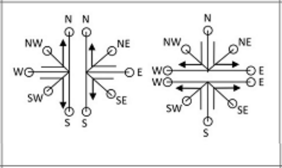
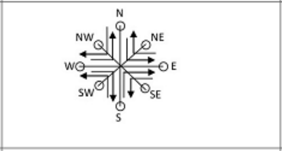
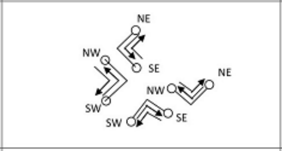
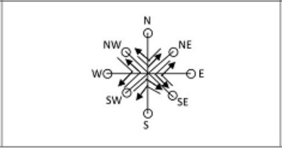
Instruction	Illustration	Weight
Continue *north/south/east/west*		1
Continue *north west/ north east/ south west/ south east* (not in cardinal direction)		3
Turn *west/east/north/south*		3
Turn *west/east/north/south* at T-junction		3
Turn *west/east/north/south* at other junction		5
Turn *north west/ north east/ south west/ south east* (not in cardinal direction)		5
Turn *north west/ north east/ south west/ south east* (not in cardinal direction) at other junction		7

Fig. 7. Simplest cardinal direction weighting function (Shahrom, 2013)

An increase of 26.5% of an average mean of success rates when the agent was given the initial orientation at the beginning of its navigation. The results from the statistical test also indicate that there was a significant difference in success rate between *SimpRelOP* and *SimpNP* ($p = 0.001 < 0.05$).

There was also an improvement in the percentage of frequency of normalized Euclidean stopping distance, when initial orientation was given. Without initial orientation, 36.67% of the agents stopped at a normalized Euclidean stopping distance > = 1. The increase in navigation performance can be seen where the frequency of normalized stopping distance at range > = 1 reduced to 3.33% for *SimpRelOP* and thus the frequency moved to a higher level, which is in the range of 0.4-0.6 of normalized Euclidean stopping distance. These mean differences were significant based on the statistical tests. Thus, it can be concluded that there is a significant difference in normalized Euclidean stopping distance between *SimpRelOP* and *SimpNP* ($p = 0.01 < 0.05$) where *SimpNP* stopped farther than *SimpRelOP*.

However, the analysis on the normalized network stopping distance showed although *SimpRelOP* performed better than *SimpNP*, with an average difference of 200 meters in stopping distance, however, no significant difference between *SimpRelOP* and *SimpNP* when the statistical test was conducted. The tremendous performance of the agents when initial orientation was given shows that, giving efficient direction is important for navigation under imprecision.

Cognitive Cost Heuristic: Simplest Cardinal Direction Weighting Function

Brunye *et al*. (2015) reported that real navigation is faster and more accurate when following cardinal direction. In this experiment, comparisons were made based on which navigation instruction can efficiently lead agents to the destination, by using either cardinal direction (*SimpCardOP*) or relative direction (*SimpRelOP*). For this purpose, an agent was assumed to have knowledge about compass direction to execute the instructions.

Figure 7, presents a new model for the simplest cardinal direction-weighting function (Shahrom, 2013), which is constructed based on the complexity of communicating cardinal instructions. This model implements the simplest path strategy for computing cardinal direction travelling costs. The weight is chosen to reflect the amount of complexity of information required to describe the decision points.

Based on work by Mark (1986), the instructions are classified into frames that have several slots for different properties of an instruction. The numbers of the slots are used as the weighting function to measure the information content of the instruction. In Fig. 7, each weight holds information about whether the direction is moving straight from the cardinal direction (1 slot), moving not from the cardinal direction (3 slots), turn at a cardinal direction (3 slots), turn not from cardinal direction, or if the direction involves junction or intersection >2 (2 slots). For example, the instruction "turn west/east/north/south at T-junction" is weighted "3" because the junction is easy to recognize and not possible to overshoot at T-junction, whereas, "turn west/east/north/south at other junctions" is weighted "5" because it involves two information; turn at cardinal direction (3 slots) and involves junction or intersection >2 (2 slots). The weight is given high when the cardinal instruction complexity is high, in which the instruction involves the intersection.

There was an improvement on the performance when cardinal direction was used in communicating the navigation instruction by using *SimpCardOP* algorithm. At most levels, the difference in success rates can be seen between the two directions. The result shows, an increase of 13.1% for *SimpCardOP* in the success rate. From the statistical test, *SimpCardOP* scored more than 50% success rate with 13% higher than *SimpRelOP*. The result shows that, there is a significant change in the average mean value, especially between *SimpCardOP* and *SimpRelOP* ($p = 0.00 < 0.05$). The results on success rate also concluded that cardinal direction is an efficient direction to be used in the private navigation algorithm.

The improvement in the navigation performance can also be seen in the frequency distribution of normalized Euclidean stopping distance. The maximum frequency distribution of normalized Euclidean stopping distance increased to an average range 0.1-0.3 of normalized Euclidean stopping distance for *SimCardOP*. The results from analysis test shows that there was a significant difference in the normalized Euclidean stopping distance for all algorithms. Thus, there is a significant change in the average mean value for both tests for *SimpCardOP* and *SimpRelOP* ($p = 0.00 < 0.05$). Since the average mean of *SimpCardOP* was lesser than the mean *SimpRelOP*, it can be concluded that cardinal direction is more efficient for navigation under imprecision in terms of normalized Euclidean stopping distance as compared to relative direction.

The normalized network distance also showed a significant difference between *SimpCardOP* and *SimpRelOP*, ($p = 0.00 < 0.05$). *SimpCardOP* has brought agents closer 200 m more than *SimpRelOP* to the destination. These results indicated that, with cardinal direction, most agents stopped closer to destinations as compared to the relative direction for imprecise navigation in terms of normalized network stopping distance.

Agent's Behavior in Wayfinding Affects Navigation Performance

This experiment was conducted in order to test whether an active agent can perform better than a passive agent for imprecise navigation. For the comparison purposes, the algorithm with relative and cardinal directions were used in the experiments and the agents' performances were recorded and compared.

Navigation agent algorithm in Fig. 8 can be used by an agent to navigate from a starting node $s \in V$ to a destination node $d \in V$. The algorithms generate the instruction sequence *Seq* from s to d where s is the start node. The algorithms execute the sequence of instructions by making the agent follow each instruction.

For every executable instruction $i \rightarrow t$, the location s is updated to t where t is a stop node. However, if the instruction is not executable and the agent is *active*, the agent would hold any instruction that cannot be completed at that time and keep moving on until the instruction can be executed. If the agent is *passive*, it stops immediately $s = null$ and the algorithm terminates. When the instruction sequence has been executed, the algorithm checks to see whether the agent has reached its destination d. If so, the agent has arrived at a location d and the algorithm terminates.

Figure 9 illustrates the agent's wayfinding environment. It is modeled through 7 nodes with a static simulated environment. The agent's task is to find the way from an obfuscated region (contains two nodes, a and b) to node e.

Example, $Seq = [S, L]$, where $S = Straight$, $L = Left$. For the *passive* agent that start from a, the path is (a, b, c, *stop*) whereas from b, the path is (b, c, d, e). The agent from a fails to reach destination when following the instruction in a passive mode because the agent could not complete the instruction sequence. However, if the agent is *active*, it can successfully reach destination e, where it continues straight on at node c and then performs the next instruction L when it reaches d. The behavior of an agent, either passive or active, determines the result of the wayfinding activities.

The results from the experiments show that the private navigation algorithm works well in both relative (*SimpRelOA)* and cardinal directions (*SimpCardOA*) especially when an agent was active. Caduff and Timpf (2005) discussed different models of agents, namely strict and weak, to interpret a robust route instruction. In their model, strict agent stops when the instruction is not executable, whereas a weak agent moves straight on and executes the instruction at the next possibility. They reported that, their weak agent produces a more robust route and performs better in navigation, which is similar to the *active* agent in this research. Based on the analyses, *SimpCardOA* was the most efficient algorithm, which is not only it can achieve a high percentage of success, but it can also bring agent closer to the destination (Fig. 10). When the agent was moving "straight on" if the instruction was not executable and then try to execute the instruction whenever possible, it was expected to see improvement in the navigation performance. The algorithms that run with active agents, increased in success rate performance. The consistency can be seen in the navigation's performance where the success rates for all tests increased with active agents and the result shows positive significant difference ($p = 0.00 < 0.05$). Both *SimpCardOA* and *SimpRelOA* scored above 51% of success rates with this condition.

```
Define input: A graph $G = (V, E)$ is a connected, simple and directed graph with
$s \in V$ is the starting node and $d \in V$ is the destination node; where $i$ is an instruction
and $Seq$ is an instruction sequence; $t$ is a stop node and $u$ is the node after the execution
of straight on action.

while $s \neq d$ do
    for $(i = 0; i < Seq; i++)$ do
        if $s \in V$ and $s \neq null$ then
            Execute instruction i
            if $i \mapsto t$ then
                $s = t$
                else
                // Passive agent
                $s = null$
                // Active agent
                $s = u$
                while $s = u$ do
                    if $i \mapsto t$ then
                        $s = t$
                        else
                        $s = u$
```

Fig. 8. Navigation agent

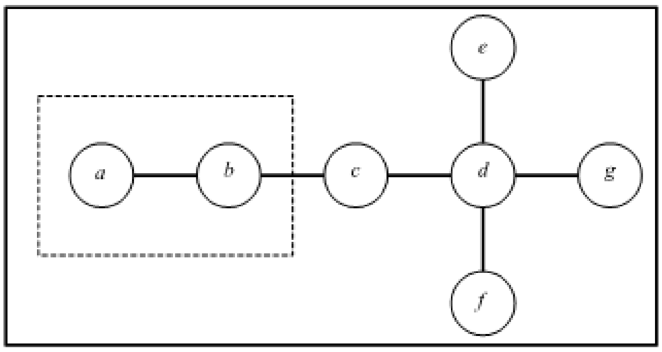


Fig. 9. Following navigation instructions

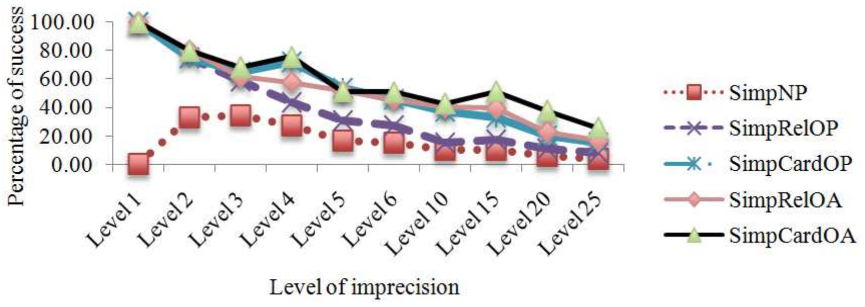


Fig. 10. Percentage of success

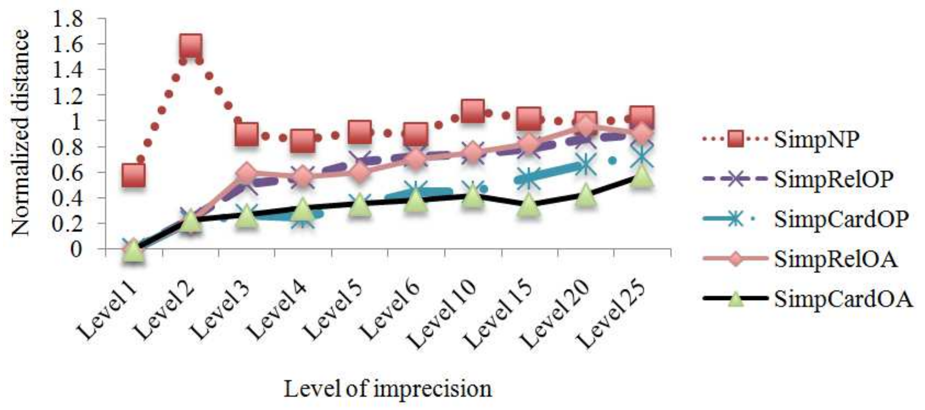


Fig. 11. Normalized euclidean stopping distance

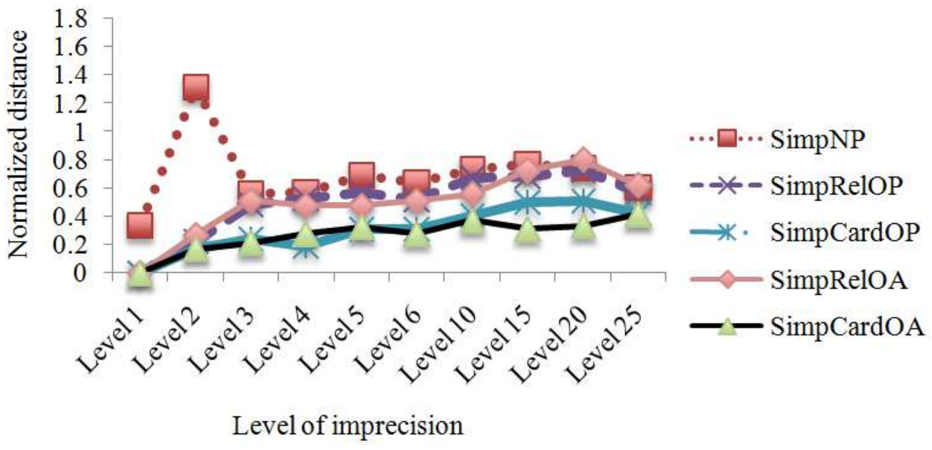


Fig. 12. Normalized network stopping distance

However, the increase in the performance of normalized Euclidean stopping distance failed to be proven statistically for *SimpCardOA* and *SimpRelOA* (Fig. 11). Thus, there was no significant difference in the stopping distance when the agent was active; *SimpCardOA* ($p = 0.1 > 0.05$) and *SimpRelOA* ($p = 0.5 > 0.05$). Active agents in *SimpCardOA* ($p = 0.2 > 0.05$) and *SimpRelOA* ($p = 0.9 > 0.05$), did not affect the difference in normalized network stopping distance. The results showed that there were no significant differences in the normalized network stopping distance when the agent was active (Fig. 12).

Conclusion

The work described in this research has been concerned with the development of a cognitively motivated private navigation algorithm for imprecise navigation. The research aims at bridging the gap between providing high quality personal navigation services and low quality location information. The private navigation algorithm relies on the topological representation and takes into account the simplicity for communicating navigation instruction in human wayfinding.

The algorithm was further improved by applying a new cost function called a simplest cardinal direction weighting function. This weighting function has been introduced for reducing cognitive load and decreasing wayfinding errors for imprecise navigation. Reducing cognitive load means providing a user with the minimum amount of information required to find their way. Technically, this research should enable computing routes that are more aligned with human cognition for wayfinding and generating route instructions by using imprecise location information. The algorithm proposed can also be applied for privacy-aware navigation services which not only can protect user's location privacy but also receive efficient navigation instructions. With agent simulation, it is possible to determine where people face wayfinding difficulties, why they face them and how wayfinding information and design have to be changed to avoid such difficulties. Moreover, the testing of different navigation ideas and theories before implementation in the real world can result in major economic benefits.

Acknowledgement

The authors would like to extend their appreciation to Prof. Matt Duckham for several related research discussions.

Funding Information

The authors have no support or funding to report.

Author's Contributions

Melissa Shahrom: Led the study, conducted the experiments and did all the analysis. She also produced the manuscript in its original form and revised it into its final form.

Zalilah Abd Aziz: Reviewed the draft manuscript.

Ethics

We confirm that this manuscript has not been published elsewhere and is not considered for another journal.

References

Agarwal, U. and V. Gupta, 2014. Network routing algorithm using genetic algorithm and compare with route guidance algorithm. Int. J. Sci. Res. Eng. Technol.

Brunye, T.T., Z.A. Collier, J. Cantelon, A. Holmes and M.D. Wood *et al*., 2015. Strategies for selecting routes through real-world environments: Relative topography, Initial route straightness and cardinal direction. PLoS ONE, 10: e0124404-e0124404. PMID: 25992685

Caduff, D. and S. Timpf, 2005. The Landmark Spider: Representing Landmark Knowledge for Wayfinding Tasks. In: Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance. Barkowsky, T., C. Freksa, M. Hegarty and R. Lowe (Eds.), Menlo Park, CA, pp: 30-35.

Duckham, M. and L. Kulik, 2003. Simplest Paths: Automated Route Selection for Navigation. In: Spatial Information Theory: Foundations of Geographic Information Science, W. Kuhn, M.F. Worboys and S. Timpf (Eds.), pp: 169-185.

Duckham, M. and L. Kulik, 2005. A Formal Model of Obfuscation and Negotiation for Location Privacy. In: Pervasive Computing, Gellersen, H.W., R. Want and A. Schmidt, Springer, pp: 243-251.

Duckham, M., L. Kulik and M. Worboys, 2003. Imprecise navigation. Geoinformatica, 7: 79-94. DOI: 10.1023/A:1023426607262

Duckham, M., S. Winter and M. Robinson, 2010. Including landmarks in routing instructions. J. Locat. Based Services, 4: 28-52. DOI: 10.1080/17489721003785602

Goldin, S.E. and P.W. Thorndyke, 1982. Simulating navigation for spatial knowledge acquisition. Hum. Factors: J. Hum. Factors Ergonom. Society, 24: 457-471. DOI: 10.1177/001872088202400407

Golledge, R.G., 1995. Path selection and route preference in human navigation: A progress report. Proceedings of the International Conference COSIT, Sept. 21-23, Semmering, Austria, pp: 207-222. DOI: 10.1007/3-540-60392-1_14

Golledge, R.G., 1999. Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes. 1st Edn., JHU Press, Baltimore, ISBN-10: 080185993X, pp: 428.

Haque, S., L. Kulik and A. Klippel, 2007. Algorithms for reliable navigation and wayfinding. Proceedings of the International Conference on Spatial Cognition V: Reasoning, Action, Interaction, (RAI' 07), ACM, pp: 308-326.

Hochmair, H. and A.U. Frank, 2000. Influence of estimation errors on wayfinding-decisions in unknown street networks-analyzing the least-angle strategy. Spatial Cognit. Comput., 2: 283-313. DOI: 10.1023/A:1015566423907

Hochmair, H.H. and V. Karlsson, 2005. Investigation of Preference between the Least-Angle Strategy and the Initial Segment Strategy for Route Selection in Unknown Environments. In: Spatial Cognition IV, Reasoning, Action, Interaction, Freksa, C., M. Knauff, B. Krieg-Brückner, B. Nebel and T. Barkowsky (Eds.), Springer, Berlin, pp: 79-97.

Mark, D.M., 1986. Automated route selection for navigation. IEEE Aerospace Electr. Syst. Magaz., 1: 2-5. DOI: 10.1109/MAES.1986.5005198

Michon, P. and M. Denis, 2001. When and why Referring to Visual Landmarks in Direction Giving. In: Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, Freksa, C. and D.M. Mark, (Eds.), Springer, Berlin, ISBN-10: 3540483845, pp: 292-305.

Richter, K.F. and M. Duckham, 2008. Simplest Instructions: Finding Easy-To-Describe Routes for Navigation. In: Geographic Information Science, T.J. Cova, H.J. Miller and M.F. Goodchild (Eds.), Springer, pp: 274-289.

Shahrom, M., 2013. Relative and cardinal directions for privacy-aware personal navigation services: A comparison towards navigation efficiency. Proceedings of the 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Apr. 2-5, IEEE Xplore Press, Melbourne, VIC, pp: 426-431. DOI: 10.1109/ISSNIP.2013.6529828

Streeter, L.A., D. Vitello and S.A. Wonsiewicz, 1985. How to tell people where to go: Comparing navigational aids. Int. J. Man-Machine Stud., 22: 549-562. DOI: 10.1016/S0020-7373(85)80017-1

Werner, S., B. Krieg-Bruckner, H.A. Mallot, K. Schweizer and C. Freksa, 1997. Spatial Cognition: The Role of Landmark, Route and Survey Knowledge in Human and Robot Navigation. In: Informatik '97 Informatik als Innovationsmotor: 27. Jahrestagung der Gesellschaft für, Jarke, M., K. Pasedach and K. Pohl (Eds.), Informatik Aachen, Springer-Verlag, Berlin, ISBN-10: 3642608310, pp: 41-50.

Westphal, M. and J. Renz, 2011. Evaluating and minimizing ambiguities in qualitative route instructions. Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Nov. 01-04, Chicago, IL, USA, pp: 171-180. DOI: 10.1145/2093973.2093997

Wiener, J.M., S.J. Buchner and C. Holscher, 2009. Taxonomy of human wayfinding tasks: A knowledge-based approach. Spatial Cognit. Comput: Interdisciplinary J., 9: 152-165. DOI: 10.1080/13875860902906496