University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Duckham, M;Kulik, L;Worboys, MF

Title:
Imprecise Navigation

Date:
2003

Citation:
Duckham, M., Kulik, L. & Worboys, M. F. (2003). Imprecise Navigation. GeoInformatica, 7 (2), pp.79-94. https://doi.org/10.1023/A:1023426607262.

Publication Status:
Published

Persistent Link:
https://hdl.handle.net/11343/34956

# Imprecise navigation

Published in *Geoinformatica*, v7, n2, pp 79–94

MATT DUCKHAM duckham@spatial.maine.edu
*NCGIA, University of Maine, ME 04469-5711, USA*

LARS KULIK kulik@spatial.maine.edu
*NCGIA, University of Maine, ME 04469-5711, USA*

MICHAEL WORBOYS worboys@spatial.maine.edu
*NCGIA, University of Maine, ME 04469-5711, USA*

**Keywords:** granularity, wayfinding, navigation agent, graph theory, indiscernibility

**Abstract.** Conventional models of navigation commonly assume a navigation agent's location can be precisely determined. This paper examines the more general case, where an agent's actual location cannot be precisely determined. This paper develops a formal model of navigation under imprecision using a graph. Two key strategies for dealing with imprecision are identified and defined: *contingency* and *refinement*. A contingency strategy aims to find an instruction sequence that maximizes an agent's chances of reaching its destination. A refinement strategy aims to use knowledge gained as an agent moves through the network to disambiguate location. Examples of both strategies are empirically tested using a simulation with computerized navigation agents moving through a road network at different levels of locational imprecision. The results of the simulation indicate that both the strategies, contingency and refinement, applied individually can produce significant improvements in navigation performance under imprecision, at least at relatively fine granularities. Using both strategies in concert produced significant improvements in performance across all granularities.

## 1. Introduction

Conventional models of navigation commonly assume a navigation agent's location can be precisely determined. However, all location sensing technology is inherently imprecise, and in general precise location cannot be determined. Imprecision leads to *granularity*, where individual elements within a particular grain cannot be discerned apart.

For example, a cell phone network is a familiar imprecise location-aware technology. In order to correctly route calls, the network tracks in which network cell each cell phone is located. The cells are typically between a few hundred meters to a few kilometers in size. Conventional models of navigation would not be applicable to a person driving through a busy city armed with only a basic cell phone, since it would not be possible to discern apart the locations of different road intersections within each cell. The techniques explored within this paper are designed to be able to cope with just such a situation. These techniques use the constraints placed on an agent's movement by the transport network to augment imprecise knowledge

about location. Potentially, such techniques could be used to provide location-based services, such as navigation assistance, to users whose precise location, and perhaps precise destination, is unknown.

Following the literature review in the section §2, §3 introduces a formal model of navigation under imprecision. This formal model is extended with two key strategies, contingency and refinement, in §4. The results of an empirical study of computerized agent navigation, based on the formal model, are reported in §5. A discussion of the computational issues and the relationship to human navigation is contained in §6, followed by conclusions in §7.

### *1.1. Why imprecise?*

A basic premise of this work is that irrespective of improvements in location sensing technology, imprecision will remain an important issue for location-aware systems into the foreseeable future. Given the decreasing cost and increasing availability of low power high precision GPS devices it can seem as if there is no need to consider the effects of imprecision. In the case of GPS, obstacles such buildings or dense forest canopy attenuate GPS radio frequency (RF) signals and can lead to severe multipath problems [15]. Such problems have recently led several major US cell phone networks to abandon GPS-based location sensing in favor of augmented cell phone network-based location sensing (such as TruePosition, [33]) in their attempts to meet FCC E911 requirements (the ability to locate a cell phone to within about 50 to 100m in an emergency, see [10]).

No location-aware technology is ever likely to achieve so complete and uniform coverage that every person at any location and any time can determine their precise location. Rather, we assume that in the future a range of different location-sensing technologies, (including GPS, triangulation of RF wireless LAN signals [3], proximity to infrared beacons [34], scene analysis and computer vision [18], and inertial tracking [25]) will be needed for location aware systems in different environments (in cities, in buildings, in wilderness) and for different people. Hightower and Boriello [16] provide an extensive review of location sensing technology including a discussion of accuracy and precision issues.

One further issue is relevant to this discussion: privacy. It is likely, as location-aware systems become more commonplace, that many people will want to protect sensitive information about their location. Individuals may wish to obfuscate their precise location for privacy reasons (eg [7]) at the same time as accessing valuable location-based services. The techniques developed within this paper might be used by an individual wishing to access location-based services without revealing their precise location. In other circumstances, however, these techniques might also be used by some third party to determine more precisely the location of an individual, providing a mechanism for tracking an individual in a way that potentially could be an invasion of privacy.

In summary, even given future developments in technology, there exist many potential scenarios where an individual might wish access location-based services without using precise knowledge of location.

© Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

## 2. Background

This paper presents an empirical study of navigation under imprecision in computerized agents. However, the ultimate objectives of this work concern navigation under imprecision in human agents. There exist numerous studies of human navigation and wayfinding. The term "wayfinding", coined by Lynch [21], is often regarded as near synonymous with "navigation" (eg [17], although wayfinding is sometimes described as "non-instrumental navigation"). Studies of human cognition suggest that human navigation involves *landmark* knowledge, concerning significant locations in a space; *procedural* knowledge, concerning routes through a space; and *survey* knowledge, concerning an allocentric view of the space [26, 2, 14]. A variety of models of navigation have been proposed, addressing different aspects of human navigation, including route descriptions [30, 8, 1], landmarks [23], planning and survey knowledge [13, 32, 31].

Most models of human navigation agents deal exclusively with navigation where an agent's location can be precisely determined. The classic TOUR model of Kuipers [19, 20], for example, relies on knowledge of precise location. More recently, Frank [11, 12] proposes an algebra of navigation derived from the TOUR model, similarly based on knowledge of precise location. Relatively few studies of imprecision in navigation have been attempted. Chown [5] has begun to examine the role of uncertainty in the PLAN model, an extension of the TOUR model. Raubal and Worboys [24] have used rough sets to model imprecise knowledge in the wayfinding process. In [24] indiscernibility in observations results in some propositions being undetermined, in addition to others being classically true or false. A different approach is taken in [4], where fuzzy set theory is used to model imprecision in a robot's location. In [4] machine learning techniques are used to improve the robot's navigation performance under imprecision. None of these approaches addresses the specific problem, examined in this paper, of delivering location-based navigation services to an agent under imprecision.

## 3. Imprecise navigation

### *3.1. Formal description of imprecise navigation*

Graphs are a natural mechanism for representing networks, such as road networks. A graph $G$ comprises a set of vertices $V$ and edges $E$ connecting those vertices. A labeled graph additionally has a label $w(e)$ associated with each edge $e \in E$. A simple mechanism for modeling the imprecision of an agent's location is to use an equivalence relation $\sim$ on the set of vertices $V$, where a location $v \in V$ cannot be discerned apart from other locations in the equivalence class of $v$, $[v] \in V/\sim$. The movement of an agent through a network under imprecision can then be represented as a sequence of equivalence classes on the vertices of a graph.

Adopting this model of imprecision in navigation has the advantage of simplicity. However, representing imprecision by an equivalence relation on $V$ assumes a discrete model of movement through a network, where edges facilitate movement

between nodes, but movement along edges is not represented. Representing the movement of an agent along an edge, such as movement along a road joining two intersections, is likely to be an important component of a more sophisticated model of imprecision in navigation.

Assuming a graph $G = (V, E)$ is simple (there are no edges from a vertex to itself and at most one edge between two different vertices), a path $p$ can be represented as a sequence of vertices $(v_0, v_1, ..., v_n)$, where each pair of vertices $v_i, v_{i+1}$ in $p$ is connected by an edge $e \in E$. A navigation agent cannot directly use paths to navigate under imprecision since they refer to precise locations within the graph. Instead, the agent must abstract away from precise locations by considering the *instructions* needed to follow a path.

For each vertex $v \in V$ we define a *labeling* function $f_v : C(v) \to L$, where $C(v)$ is the set of vertices connected to $v$ by a single edge ($C(v) := \{v' \in V | (v, v') \in E\}$) and $L$ is some set of labels. The actual elements of $L$ are arbitrary, but the cardinality of $L$ must be greater than or equal to the largest degree of any vertex $v \in V$ ($|L| \geq \texttt{sup}(\texttt{deg}(v))$ for all $v \in V$). In other words, $L$ must be at least large enough to ensure that every $f_v$ is injective (one-to-one). Using the labeling function, a sequence of instructions for following a path can be generated without directly referring to (precise) locations. For some path $p = (v_0, v_1, ..., v_n)$ and labeling functions $f_{v_0}, ..., f_{v_{n-1}}$ the instruction sequence $q$ is given by:

$$q = a_0 a_1 ... a_{n-1} \text{ where } a_i = f_{v_i}(v_{i+1})$$

Each element $a_i$ of an instruction sequence $q$ is called an instruction. The set of all instruction sequences forms a language on $L$. Crucially, while each path can be associated with a unique instruction via the labeling functions, an individual instruction can describe many different paths. Figure 1 provides an example graph annotated with labels $L = \{n, s, e, w\}$. As stated above, the choice of labels is arbitrary. Cardinal directions are used in this example only to make Figure 1 easier to interpret. From Figure 1, the label of $v_3$ from $v_0$ is $s$ ($f_{v_0}(v_3) \mapsto s$). The path $(v_6, v_7, v_3, v_4, v_8)$, for example, would be represented by the instruction sequence *enes*. This same instruction sequence would also describe the path $(v_3, v_4, v_1, v_5, v_9)$. Note that while the equivalence classes $V/\sim$ form contiguous blocks in Figure 1, akin to cells in a cell phone network, this is not a requirement of the model.

### *3.2. Basic navigation agent*

Based on the formal model above, we can provide the outline of a basic agent capable of navigation under imprecision. The basic agent is assumed to possess (precise) knowledge of the labeled graph $G = (V, E)$ and the weights associated with each edge $w : E \to \mathbb{R}^+$, the granulation $V/\sim$, the graph labeling functions $f_v : C(v) \to L$, and the destination $d \in V$. Note that in this initial work we assume that the destination $d$ is precisely known, but this assumption can easily be generalized to address cases where the destination is itself imprecise, modeled by an equivalence class $[d]$. While the agent is actually located at some vertex $s \in V$,

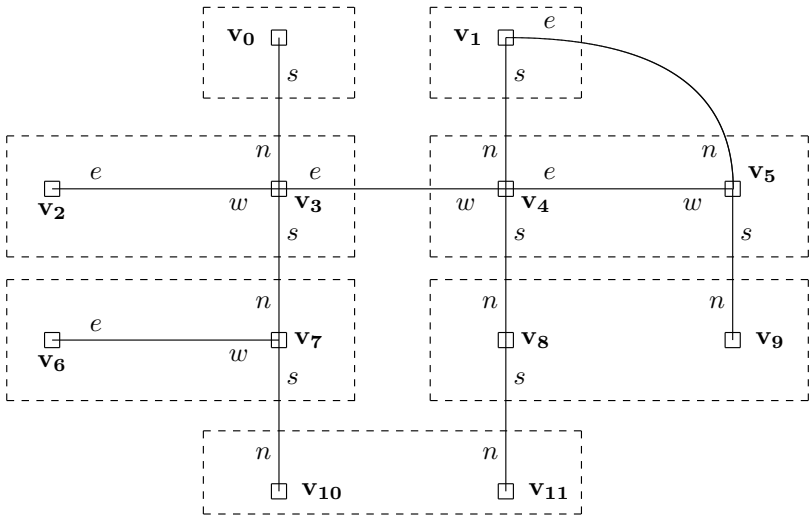



*Figure 1.* Example granulated graph annotated with labels and equivalence classes (dotted lines indicate equivalence between enclosed vertices)

it can only access imprecise information about its current location in terms of the equivalence class $[s]$.

Algorithm 1 gives a naive algorithm that could be used by such an agent attempting to navigate from a starting vertex $s \in V$ to a destination vertex $d \in V$. While the agent is initially located at $s$, it is unable to discern $s$ apart from other elements in the equivalence class $[s]$. Instead, the algorithm arbitrarily selects an element $s' \in [s]$ and generates the instruction sequence $q$ for the shortest path from $s'$ to $d$ (given weights $w(e)$ for each edge $e \in E$). The algorithm then executes the sequence of instructions by making the agent follow each instruction in turn, ignoring any instructions that cannot be completed (ie where at location $v$ for some instruction $a$ there exists no $v' \in V$ such that $f_v(v') \mapsto a$). When the instruction sequence has been executed, the algorithm checks to see whether the agent has reached its destination, by checking whether the (updated) location $s$ is in the equivalence class of the destination location $[d]$. If so, the agent has arrived at a location indistinguishable from $d$ and the algorithm terminates. Otherwise, the algorithm reiterates using an arbitrarily selected element of $[s]$ as the new starting vertex.

## 4. Improving imprecise navigation

Even though precise location information is unavailable to the navigation agent, it is possible to improve the basic imprecise navigation algorithm described above

**Algorithm 1:** Basic imprecise navigation algorithm

$G = (V, E)$, $w : E \to \mathbb{R}^+$, $V/\sim$, $\{f_v | \forall v \in V . f_v : C(v) \to L\}$, $s \in V$, $d \in V$
**while** $s \notin [d]$ **do**
  *// Generate instruction sequence*
  Choose arbitrary $s' \in [s]$
  Calculate shortest path $p = (v_0, v_1, ..., v_n)$ where $v_0 = s'$, $v_n = d$ given $w$
  Generate instruction sequence $q = a_0 a_1 ... a_{n-1}$ where $f_{v_i}(v_{i+1}) \mapsto a_i$
  *// Follow instruction sequence*
  **for** $j = 0$ *to* $j < |q|$ **do**
    **if** $\exists t \in V$ *such that* $f_s(t) \mapsto a_j$ **then** $s = t$

by taking advantage of the constraints the network places on movement. The basic imprecise navigation routine in Algorithm 1 has two main elements: generating an instruction sequence, and following an instruction sequence. In general, it is possible to identify two corresponding strategies for improving navigation under imprecision, termed here *contingency* and *refinement*. Contingency involves finding an instruction sequence that maximizes an agent's chances of reaching its destination. Refinement involves using additional information gained while following an instruction sequence to infer more detailed knowledge about an agent's location. Both strategies are elaborated upon below.

### *4.1. Contingency*

Since the same instruction sequence can be used to describe different paths, a contingency strategy aims to find an instruction sequence that maximizes an agent's chances of reaching its destination. In human navigation, for example, instructions such as "Continue straight on to the T-junction and then turn right", can sometimes be used for navigating from a range of different starting locations to a single destination.

To illustrate an example of contingency, consider the graph in Figure 1. An agent is located at $v_{11}$ and tries to navigate to $v_1$. Assuming edges are weighted using the Euclidean distance along each edge, the shortest path from $v_{11}$ to $v_1$ is $(v_{11}, v_8, v_4, v_1)$, with instruction sequence *nnn*. However, due to imprecision, the agent is unable to discern $v_{10}$ and $v_{11}$ apart. The shortest path from $v_{10}$ to $v_1$ is $(v_{10}, v_7, v_3, v_4, v_1)$ with instruction sequence *nnen*. Note that the instruction sequence *nnen* is also a path from $v_{11}$ to $v_1$, although this is not the shortest path. Selecting the instruction sequence *nnen* guarantees the agent will arrive at its destination from either $v_{10}$ or $v_{11}$; selecting the instruction sequence *nnn* will only lead an agent to the destination if it is located at $v_{11}$. As a result, one example of a contingency strategy would be to find and preferentially use those instruction sequences that increase an agent's chance of reaching its destination.

Algorithm 2 provides a modified example of Algorithm 1. Instead of selecting an arbitrary element of $[s]$ to use for generating an instruction sequence, the algorithm

checks the instruction sequences for all shortest paths from all elements of $[s]$ to $d$. The most successful instruction sequence (the one that when executed leads most frequently to the destination) is selected for the next stage of the algorithm.

**Algorithm 2:** Imprecise navigation algorithm with contingency

$G = (V, E)$, $w : E \to \mathbb{R}^+$, $V/\sim$, $\{f_v | \forall v \in V . f_v : C(v) \to L\}$, $s \in V$, $d \in V$
**while** $s \notin [d]$ **do**
  *// Generate instruction sequence*
  $m = 0$
  **for** *each vertex* $s \in [s]$ **do**
    Calculate shortest path $p = (v_0, ..., v_n)$ where $v_0 \in [s]$, $v_n = d$ given $w$
    Generate instruction sequence $q' = a_0 a_1 ... a_{n-1}$ where $f_{v_i}(v_{i+1}) \mapsto a_i$
    $c = \{s' \in [s] | \, q' \text{ executed from } s' \text{ terminates in } [d]\}$
    **if** $|c| > m$ **then** $q = q'$ and $m = |c|$
  *// Follow instruction sequence*
  **for** $j = 0$ *to* $j < |q|$ **do**
    **if** $\exists t \in V$ *such that* $f_s(t) \mapsto a_j$ **then** $s = t$

### *4.2. Refinement*

An agent has the ability to sense its current imprecise location. Refinement aims to use knowledge gained as an agent moves through the network to disambiguate location. Again, an analogy exists in human navigation. Humans unsure of their current location might head in what they believe to be the general direction of their destination in the hope that particular road configurations or landmarks will enable them to discover more precisely their location as they travel.

To illustrate an example of refinement, consider again the graph in Figure 1 and an agent located at $v_{11}$ trying to navigate to $v_1$. Selecting at random one of the instruction sequences for the shortest paths from elements of the equivalence class $[v_{11}]$ to $v_1$ yields either $nnn$ or $nnen$. Whichever is selected, after executing the first instruction, $n$, the agent will arrive at $v_8$. Even though the agent cannot discern apart the location of $v_8$ from $v_9$, there exists no direct path from either $v_{10}$ or $v_{11}$ to $v_9$ (without first passing through some other vertex that can be discerned apart from $v_8$, $v_9$, $v_{10}$ and $v_{11}$). One example of a refinement strategy would be to use precise knowledge of the graph network to disambiguate location in cases such as that described above, and recalculate new routes through the space in the light of this refinement.

Algorithm 3 provides a modified example of Algorithm 1, which adopts this refinement strategy. After executing each instruction, the algorithm checks whether knowledge about the current location can be refined based on the possible paths to that location. Specifically, when the agent moves from one equivalence class to the next, the algorithm finds those vertices that can be reached using a direct path between the two equivalence classes. This information allows the agent to

refine knowledge about its location, as illustrated in the example above. When a refinement occurs, the algorithm reiterates based on the revised knowledge of location.

**Algorithm 3:** Imprecise navigation algorithm with refinement

$G = (V, E)$, $w : E \to \mathbb{R}^+$, $V/\sim$, $\{f_v | \forall v \in V. f_v : C(v) \to L\}$, $s \in V$, $d \in V$, $R = [s]$
**while** $s \notin [d]$ **do**
  *// Generate instruction sequence*
  Choose arbitrary $s' \in R$
  Calculate shortest path $p = (v_0, ..., v_n)$ where $v_0 = s'$, $v_n = d$ given $w$
  Generate instruction sequence $q = a_0 a_1 ... a_{n-1}$ where $f_{v_i}(v_{i+1}) \mapsto a_i$
  *// Follow instruction sequence*
  **for** $j = 0$ *to* $j < |q|$ **do**
    **if** $\exists t \in V$ *such that* $f_s(t) \mapsto a_j$ **then** $s = t$
    **if** $R \not\subseteq [s]$ **then**
      Find $R' = \{t \in [s] | \exists$ a path $p = (r, v_1, ..., v_n, t)$ where $r \in R$ and $v_i \in [s] \cup [r]\}$
      Update $R = R'$ and $j = |q|$

## 5. Imprecise navigation simulation

This section describes the results of a large navigation simulation programmed using Java. The simulation was conducted with imprecise driving agents using different navigation strategies through a graph at different levels of granularity. The graph used was a road network containing more than 3000 vertices from the city of Bloomington, Indiana (available from `http://www.city.bloomington.in.us/its/gis/`). During the development of the simulation, a range of different road network data sets were successfully used. The Bloomington data set was chosen since it exhibits a wide range of different network configurations (in particular a dense downtown grid network and sparser suburban networks).

Four different types of agent were used during the simulation: the basic agent (Agent B, as in Algorithm 1), an agent that used a contingency strategy to select preferred instruction sequences (Agent C, as in Algorithm 2), an agent that used a refinement strategy to disambiguate location as it moved through the network (Agent R, as in Algorithm 3), and an agent that used both contingency and refinement strategies in concert (Agent CR, a combination of Algorithms 2 and 3). As described in the previous section, each agent has access to precise information about the labeled graph $G$ and associated weights $w$, the granulation $V/\sim$, the graph labeling functions $f_v$, and the destination $d$. The simulation does model an agent's state in terms of its precise location $s$. However, the agent does not have access to this precise state, only to imprecise information about its location in terms of the equivalence class $[s]$. In the same way, a human agent who is lost will *possess*

a precise state (where he or she is actually located), but may lack precise *knowledge* about that state. The simulation randomly selects starting and destination vertices from the network. Each agent then attempts to navigate between the same starting and destination vertices. The simulation was repeated for fifty different randomly chosen pairs of starting and destination vertices. Finally, the simulation was repeated using seven different granulations (a total of 350 individual simulations by each of four driving agents).

Figure 2 shows a small portion of the simulation road network with example routes taken by each of the four different agents along with the shortest path. Agent B takes the longest and most circuitous route, even backtracking from a dead end at one point. Agents C, R and CR all take similar routes. In this case Agent C took the shortest route, followed by CR and then R. Note that while all the agents arrive at a location *indiscernible* from the destination (the regular granulation is shown as a grid in the background of Figure 2), none of the agents arrive at the *actual* destination.

The results of the experiment are summarized in Figure 3. The graph shows the average normalized distance traveled for each agent at each level of granularity. Granularity is measured in terms of the average number of vertices per equivalence class, plotted on a log scale for clarity. The granulations used in the simulation were regular square grid partitions at different cell sizes, covering the extent of the road network, in much the same way as the cells of a cell phone network can be thought of as a partition on space. This partition induces equivalence classes on the vertices $V/\sim$. The normalized distance is the ratio of the actual distance traveled by an agent to the length of the shortest path between start and destination vertices. The normalized distance is a measure of the performance of each agent: larger normalized distances indicate the agent had to travel further to reach its destination, meaning worse performance.

The normalized distance is a good way of comparing the performance of the different agents, but is not suitable for comparing the performance of an agent with the shortest path. Occasionally, agents can achieve normalized distances of less than 1 (meaning the path the agent took was shorter than the shortest path). This occurs because an agent may stop once it has reached a vertex indistinguishable from the destination. Sometimes, particularly with coarser granulations, the vertex the agent stops at is a little nearer the starting point that the actual destination. Indeed, while Figure 3 shows an increase in normalized distance with granularity at finer granularities, the chart display a pronounced *decrease* in normalized distance with the two coarsest granularities. This is likely to be a result of the discrepancy between the destination and the actual location reached by an agent, indistinguishable from the destination. As the granularity gets very coarse, this discrepancy tends to get larger, and the normalized distance traveled by all of the agents decreases.

The graph in Figure 3 shows that in general, Agents C and R perform better than Agent B, while Agent CR performs better than any other agent. To test whether the performance of the improved agents (Agents C, R, and CR) was significantly better than that of the basic agent (Agent B) a t-test of normalized distances for the

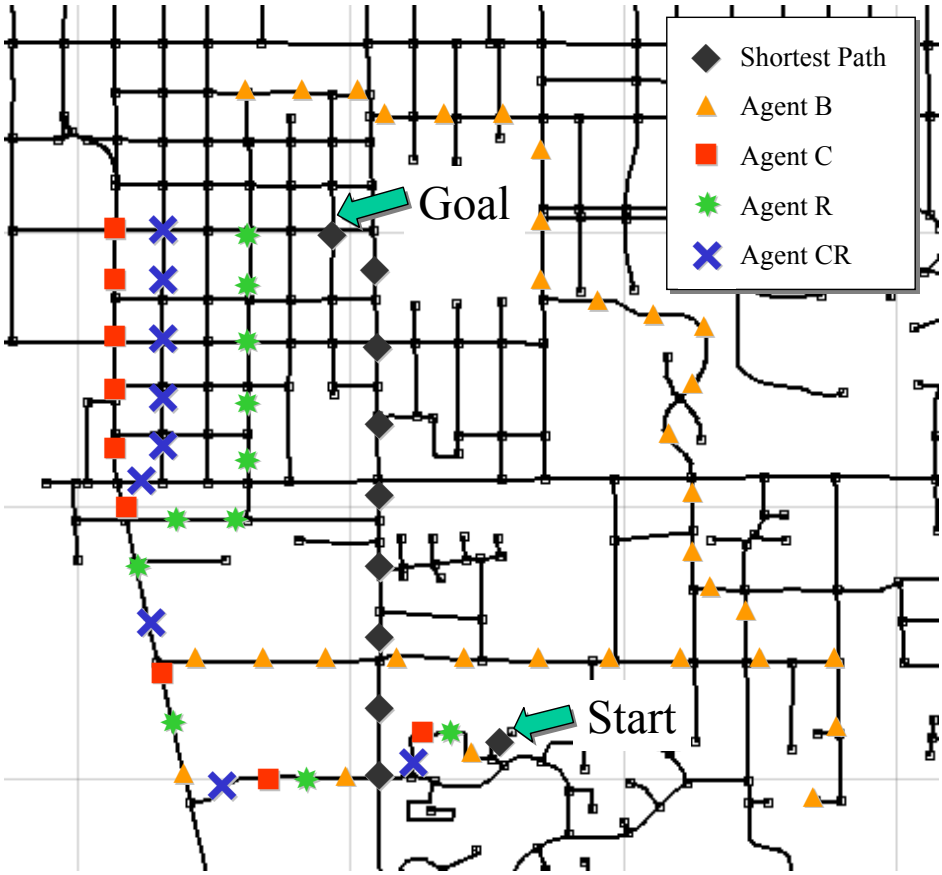



*Figure 2.* Example routes taken by four different agents, compared with shortest path

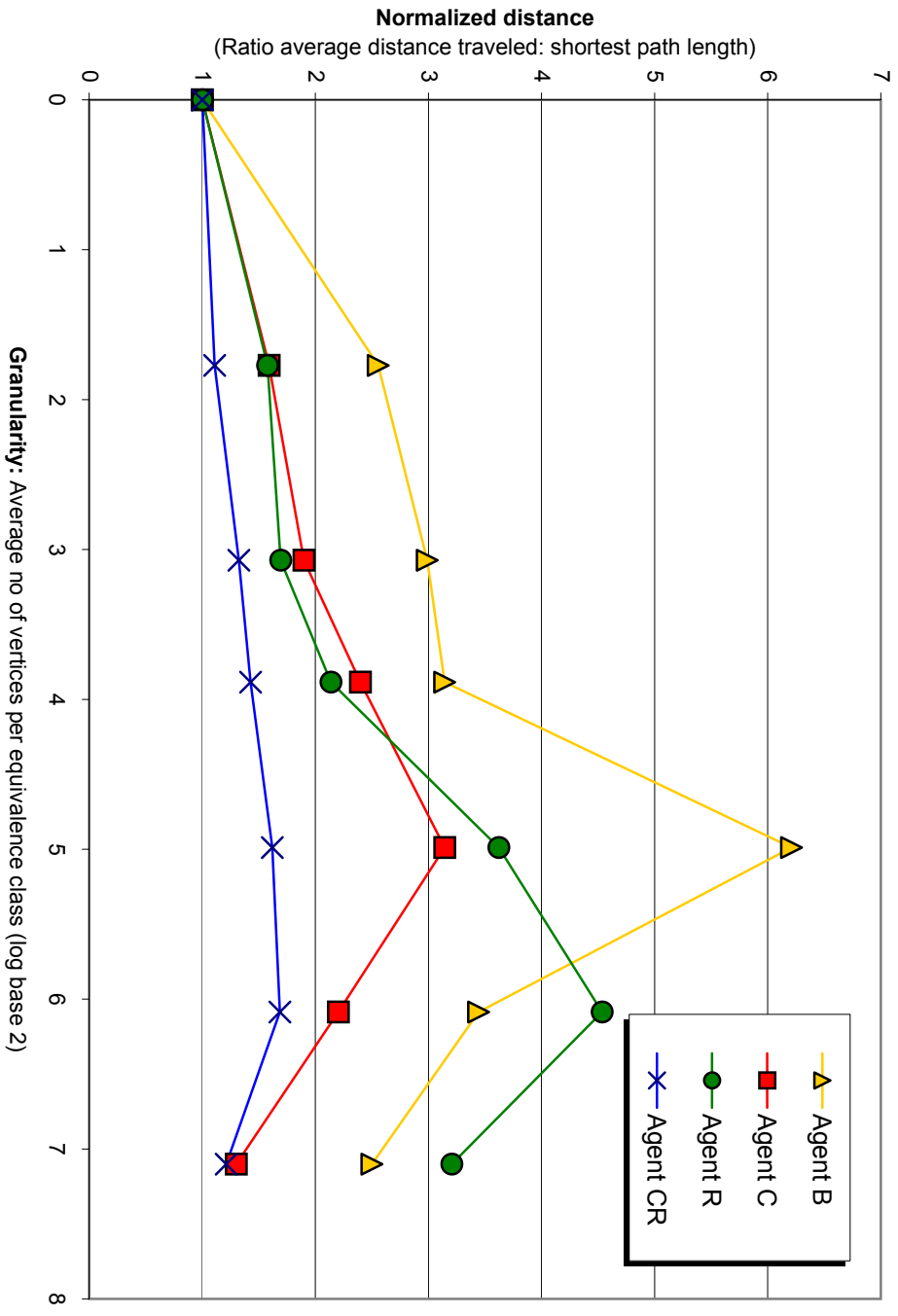



*Figure 3.* Results of imprecise navigation simulation

*Table 1.* Results of one-tailed t-distribution significance test for difference in performance of Agents C, R and CR from Agent B. The table shows for each granularity level (measured in mean number of vertices per equivalence class) whether an agent's performance was significantly different from that of the Agent B at the 5% level (S=significant, NS=not significant).

| | 1.0 | 3.4 | 8.4 | 14.8 | 31.7 | 67.9 | 138.1 |
|---|---|---|---|---|---|---|---|
| Agent C | NS | S | S | S | NS | S | S |
| Agent R | NS | S | S | S | NS | NS | NS |
| Agent CR | NS | S | S | S | S | S | S |

populations of 50 simulations for each agent at each granularity level was conducted (Table 1). The null hypothesis was that there was no significant difference between the performance of the improved agents over that of agent B. At the finest level of granularity each equivalence class contained only one vertex. At this granularity all the navigation algorithms decay gracefully to a conventional shortest path algorithm (normalized distance traveled equals 1). As a result, all the agents achieve a normalized distance of 1 at the finest granularity, and there were no significant differences between performance of any agents. For all the other granularities agent CR was found to exhibit significant improvements at the 5% level. Agents C and R also showed significant improvements at the finer granularities, but not at some of the coarser ones. For example, Agent R produced no significant improvement at over Agent B for the three coarsest granularities. Further significance tests, not shown in Table 1, also indicated that Agent CR performs significantly better than either Agents C or R, except at the coarsest granularity.

To summarize, the results of the simulation indicate that the strategies of contingency and refinement applied individually can both produce significant improvements in navigation performance under imprecision, at least at relatively fine granularities. Using both strategies in concert produced significant improvements in performance at all granularities tested where some indiscernibility existed.

## 6. Discussion

### *6.1. Computational issues*

The time complexity of the algorithms involved in imprecise navigation is dominated by the shortest path calculation. Time complexity can be an issue in imprecise navigation, as a result of the high number of shortest path recalculations. The Java implementation described above uses Dijkstra's algorithm to calculate shortest paths. The naive Dijkstra's algorithm is $O(n^2)$, where $n$ is the number of vertices in the graph, although a variety of optimizations exist to improve its performance [6]. The key advantage of using Dijkstra's algorithm in this context over related shortest path algorithms, such as the A* algorithm, is that it calculates the shortest

path from a vertex to all other vertices in a graph. This is ideal in the case of the contingency strategy, where an agent must calculate the shortest path from all starting vertices in $[s]$ to a single destination $d$. By reversing the start and destination nodes this calculation can be achieved with a single pass of Dijkstra's algorithm.

Agents need to recalculate the shortest path when they come to the end of an instruction sequence that does not bring them to a vertex indistinguishable from the destination, or when Agents R and CR gain refined knowledge about their location. In general, all the agents need to recalculate the shortest path more often as the granulation coarsens. Agents R and CR need to recalculate the shortest path more often than B and C, but Agent CR reaches its destination more efficiently than Agents B or C at least partially offsetting this additional computational burden.

The sequential nature of following navigation instructions means that shortest path calculations might easily be optimized. An optimized shortest path algorithm might provide an agent with instructions incrementally, using the time taken by the agent to execute each instruction to complete more of the calculation. This is clearly a potential strategy in human navigation where each instruction might take seconds, minutes or even hours to complete. Another strategy would be to precompute and cache the results of an all-pairs shortest path (APSP) algorithm, such as the Floyd-Warshall algorithm. This would reduce shortest path calculation to a intensive one-off calculation of the shortest paths between all pairs of vertices in a graph. Most APSP algorithms have a time complexity of $O(n^3)$.

Finally, while it was not used directly in this paper Stell [27] has developed an approach to granulation in graphs that could prove very useful in imprecise navigation. Stell suggests a range of different classifiers capable of generating new granulated graphs based on equivalence relation for vertices and/or for edges in a graph. It is possible that such a granulated graph could be used for "high-level" reasoning about navigation and even for reducing the time complexity of the shortest path computation. A discussion of some of the key issues surrounding generalization of graphs can be found in [28]. The main obstacle to using a granulated graph for shortest path calculations is deciding what weights to assign to granular edges that arise from the amalgamation of precise weighted edges, although several possibilities suggest themselves and will be explored in further research.

### *6.2. Human navigation issues*

Several key issues not covered in this paper will need to be addressed before the results of this work find practical application in human navigation, enumerated below.

- First, the system described does not incorporate mistakes. A degree of fault tolerance is in-built into the system, since navigation agents sometimes select sub-optimal or incorrect instruction sequences. However, human navigation agents do make mistakes when executing instructions, particularly when the vagueness and ambiguity of human instructions is taken into account. Future work must address this issue.

- Second, landmarks are an important feature of human navigation, used for orientation or to provide reassurance and feedback (eg [23]). Deriving landmarks under imprecision is difficult. One possibility is to use visibility analysis to identify landmarks that can be seen from most or all locations within a cell, (eg tall buildings like the CN tower in Toronto). Alternatively, spatially extended landmarks might be used (eg Oxford Street in London) where the landmark itself induces an equivalence relation on point locations.

- Third, the constraints provided by the network form the basis of the contingency and refinement strategies. These strategies can be expected to yield further performance improvements with the introduction of more constraints, such as using a directed graph. However, some navigation tasks, such as navigation on foot, are less constrained and as a result are likely to be harder to achieve under imprecision.

- Fourth, distance traveled is assumed to be a good measure of agent performance in this study. However, in human navigation the *simplicity* of instructions often needs to be taken into account. Based on [30, 29], Mark [22] developed a navigation application that weighted shortest paths according to how simple (easy to follow) the resulting instructions were. The issue of performance, both in terms of how far an agent travels and how simple an instruction sequence is to execute, needs to be addressed in future research.

- Fifth, and finally, the system described here is not very interactive. In human navigation systems, targeted human user interaction could be a valuable source of information. Simple questions to the human user, such as "Can you see the CN Tower?", might help with orientation or in refining location.

## 7. Conclusions and further work

This paper describes initial work on navigation under imprecision. It details a formal model of imprecise navigation, and identifies two key strategies for effective navigation under imprecision: contingency and refinement. A simulation with navigation agents using these strategies indicates that both strategies can produce significant improvements in an agent's navigation performance especially if used in combination.

While this paper has been concerned primarily with navigation in computerized agents, the results have clear implications for navigation of human agents. Current work in progress is attempting to develop human imprecise navigation applications, based on extensions of the empirical work described in this paper. This work is focused on using verbal interfaces to achieve navigation under imprecision. Verbal interfaces are ideal for this task as unlike most visual maps, they allow instructions to be delivered in an explicitly sequential form, and provide a natural mechanism for communicating imprecision. Future work will also begin to address vagueness in human spatial verbal predicates, such as "near", "far", "left", and "right" (eg [35, 9]).

**Acknowledgements**

This work was conducted under the SmartMaps project, supported by the National Imagery and Mapping Agency, grant no NMA201-01-1-2003. The authors would like to acknowledge invaluable input from Max Egenhofer, David Mark, Martin Raubal, and Sabine Timpf. This research has also benefitted from collaboration with Jörg-Rudiger Sack at Carleton University, Ottawa (NSERC supported Collaborative Research and Development grant).

**References**

1. Agrawala, M. and C. Stolte: 2000, 'A Design and Implementation for Effective Computer-Generated route maps'. In: *AAAI Symposium on Smart Graphics*. pp. 61–65.
2. Allen, G. and K. Kirasic: 1985, 'Effects of the Cognitive Organization of Route Knowledge on Judgments of Macrospatial Distances'. *Memory and Cognition* **3**, 218–227.
3. Bahl, P. and V. Padmanabhan: 2000, 'RADAR: An in-building RF-based user lcoation and tracking system'. In: *Proceedings IEEE INFOCOM 2000*, Vol. 2. pp. 775–784.
4. Busquets, D., R. López de Móntarez, C. Sierra, and T. Dietterich: 2002, 'A multi-agent architecture integrating learning and fuzzy techniques for landmark-based robot navigation'. In: M. Escrig, F. Toledo, and E. Golobardes (eds.): *Proceedings Fifth Catalonian Conference on AI CCIA 2002*. pp. 269–281.
5. Chown, E.: 1999, 'Making Predictions in an Uncertain World: Environmental Structure and Cognitive Maps'. *Adaptive Behaviour* **7**(1).
6. Cormen, T., C. Leiserson, R. Rivest, and C. Stein: 2001, *Introduction to Algorithms*. McGraw-Hill, second edition.
7. Cuellar, J., J. Morris, and D. Mulligan: 2003, 'Geopriv requirements'. Internet Engineering Task Force (IETF) Internet Draft. `http://www.ietf.org/ids.by.wg/geopriv.html`.
8. Denis, M., F. Pazzaglia, C. Cornoldi, and L. Bertolo: 1999, 'Spatial discourse and navigation: An analysis of route directions in the city of Venice'. *Applied Cognitive Psychology* **13**, 145–174.
9. Duckham, M. and M. Worboys: 2001, 'Computational Structure in Three-Valued Nearness Relations'. In: D. Montello (ed.): *Spatial Information Theory: Foundations of Geographic Information Science (COSIT 2001)*. Berlin, pp. 79–91.
10. Federal Communications Commission (FCC): 2003, 'Enhanced 911'. `http://www.fcc.gov/911/enhanced`. Updated: January 3rd 2003.
11. Frank, A. U.: 2000, 'Communication with maps: A formalized model'. In: C. Freksa, W. Brauer, C. Habel, and K. F. Wender (eds.): *Spatial Cognition II (Int. Workshop on Maps and Diagrammatical Representations of the Environment, Hamburg)*, Vol. 1849 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer-Verlag, pp. 80–99.
12. Frank, A. U.: 2003, 'Pragmatic Information Content—How to Measure the Information in a Route Description'. In: M. Duckham, M. F. Goodchild, and M. F. Worboys (eds.): *Foundations in Geographic Information Science*. London: Taylor & Francis, pp. 47–68.
13. Goldin, S. and P. Thorndyke: 1982, 'Simulating Navigation for Spatial Knowledge Acquisition'. *Human Factors* **24**(4), 457–471.
14. Golledge, R.: 1992, 'Place recognition and wayfinding: Making sense of space'. *Geoforum* **23**(2), 199–214.
15. Harrington, A.: 2000, 'GPS/GIS integration: Lessen atmospheric and multipath GPS errors'. *GeoWorld* **13**(12), 26–27.
16. Hightower, J. and G. Boriello: 2001, 'Location systems for ubiquitous computing'. *IEEE Computer* **34**(8), 57–66.
17. Hochmair, H.: 2002, 'The wayfinding metaphor—comparing the semantics of wayfinding in the physical world and the WWW'. Ph.D. thesis, Institute for Geoinformation, Technical University Vienna.

18. Krumm, J., S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer: 2000, 'Multi-Camera Multi-Person Tracking for EasyLiving'. In: *Proceedings Third IEEE Workshop on Visual Surveillance VS2000*. pp. 3–10.
19. Kuipers, B.: 1977, 'Representing Knowledge of Large-Scale Space'. Ph.D. thesis, Mathematics Department, Massachusetts Institute of Technology. Technical Report 418, M.I.T. Artificial Intelligence Laboratory.
20. Kuipers, B.: 1978, 'Modelling spatial knowledge'. *Cognitive Science* **2**, 129–153.
21. Lynch, K.: 1960, *The image of the city*. Cambridge, Massachusetts: Harvard University Press.
22. Mark, D.: 1986, 'Automated route selection for navigation'. *IEEE Aerospace and Electronic Systems Magazine* **1**(9), 2–5.
23. Raubal, M. and S. Winter: 2002, 'Enriching Wayfinding Instructions with Local Landmarks'. In: M. Egenhofer and D. Mark (eds.): *Geographic Information Science—Second International Conference GIScience 2002*. Berlin.
24. Raubal, M. and M. Worboys: 1999, 'A formal model of the process of wayfinding in built environments'. In: C. Freksa and D. Mark (eds.): *Proceedings of the International Conference on Spatial Information Theory*. Berlin, pp. 381–401.
25. Scott-Young, S. and A. Kealy: 2002, 'An intelligent navigation solution for land mobile location based services'. *Journal of Navigation* **55**, 225–240.
26. Siegel, A. and S. White: 1975, 'The development of Spatial Representations of Large-Scale Environments'. In: H. Reese (ed.): *Advances in Child Development and Behavior*, Vol. 10. New York: Academic Press, pp. 9–55.
27. Stell, J.: 1999, 'Granulation for Graphs'. In: C. Freksa and D. Mark (eds.): *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. COSIT'99*. Berlin, pp. 417–432.
28. Stell, J. and M. Worboys: 1999, 'Generalizing graphs using amalgamation and selection'. In: R. Guting, D. Papadial, and F. Lochovsky (eds.): *Advances in Spatial Databases: 6th International Symposium SSD'99*, No. 1651 in Lecture Notes in Computer Science. Berlin: Springer-Verlag, pp. 19–32.
29. Streeter, L. and D. Vitello: 1986, 'A profile of driver's map-reading abilities'. *Human Factors* **28**(2), 223–239.
30. Streeter, L., D. Vitello, and S. Wonsiewicz: 1985, 'How to tell people where to go: Comparing navigational aids'. *International Journal of Man Machine Interaction* **22**, 549–562.
31. Timpf, S.: 2002, 'Ontologies of Wayfinding: A traveler's perspective'. *Networks and Spatial Economics* **2**(1), 9–33.
32. Timpf, S., G. Volta, D. Pollock, and M. Egenhofer: 1992, 'A conceptual model of wayfinding using multiple levels of abstraction'. In: A. Frank, I. Campari, and U. Formentini (eds.): *Theories and methods of spatio-temporal reasoning in geographic space*, No. 639 in Lecture Notes in Computer Science. Berlin: Springer-Verlag, pp. 348–367.
33. TruePosition: 2003, 'TruePosition Web Site'. `http://www.trueposition.com`. Last accessed: January 23rd 2003.
34. Want, R., A. Hopper, V. Falcao, and J. Gibbons: 1992, 'The Active Badge location system'. *ACM Transactions on Information Systems* **10**(1), 91–102.
35. Worboys, M.: 2001, 'Nearness relations in environmental space'. *International Journal of Geographical Information Science* **15**(7), 633–651.