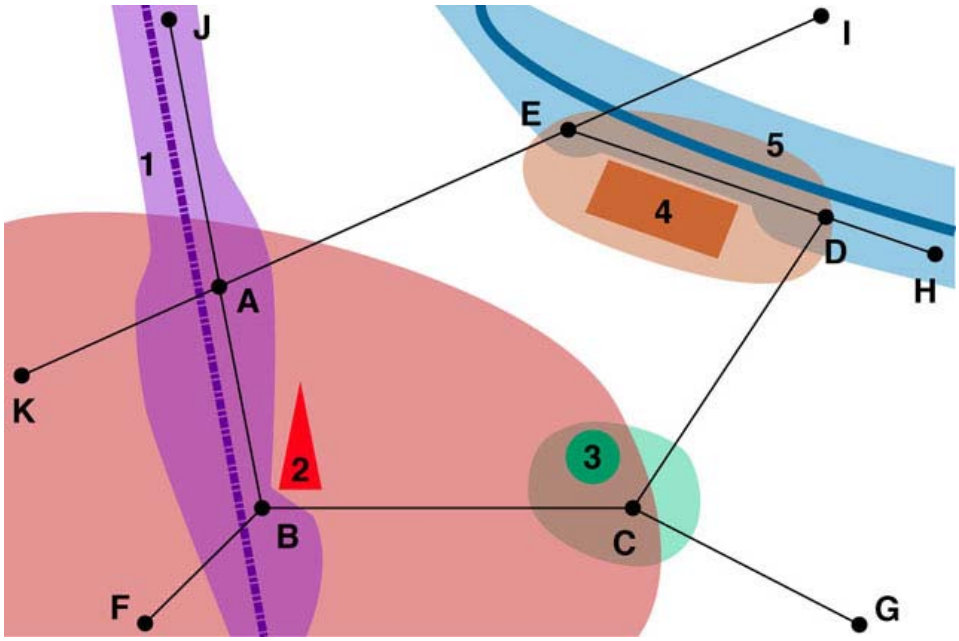


Routing by Landmarks

Urs-Jakob Rüetschi, University of Zurich
David Caduff, University of Zurich
Frank Schulz, University of Karlsruhe
Alexander Wolff, University of Karlsruhe
Sabine Timpf, University of Zurich

Conference paper STRC 2006

STRC

6th Swiss Transport Research Conference
Monte Verità / Ascona, March 15. – 17. 2006

Network Routing by Landmarks

Urs-Jakob Rüetschi
David Caduff
Sabine Timpf
Geographic Information Analysis
Department of Geography
University of Zurich
Winterthurerstr. 190
CH-8057 Zurich

Phone: +41 44 635 5255
Fax: +41 44 635 6848
email: uruetsch@geo.unizh.ch
http://www.geo.unizh.ch/gia

Frank Schulz
Alexander Wolff
Institute of Theoretical Computer Science
Department of Computer Science
University of Karlsruhe
P.O. Box 6980
D-76128 Karlsruhe

Phone: +49 721 608-4322
Fax: +49 721 608-4211
email: fschulz@ira.uka.de
http://i11www.ira.uka.de/group

March 2006

Abstract

A route in a network can be described as a sequence of nodes. Given a route, travellers need directions to follow it, which are preferably expressed as a sequence of instructions, as for instance, "face towards the tower" and "move along the river". This paper presents a method to find routes in a network with the property that they can be described by a simple sequence of instructions. The key problems that we need to solve are (1) how to attribute landmark information to the network and (2) how to find an optimal route. We approach the first problem by using landmarks as parts of instructions and mapping instructions to sets of edges in the network. The second problem can be solved by building an auxiliary graph such that a standard Dijkstra shortest path algorithm can be used to find optimal routes. Preliminary tests indicate that our approaches produce good results.

Keywords

Routing – Landmarks – Navigation – Networks

1. Navigating Networks

Networks, consisting of nodes and edges, serve as a conceptual environment for navigation. They are not the only conceptual environment (see [RT04]), but clearly the most prominent one. Navigation is defined as coordinated and goal-directed route following through space [Mo05]. Accordingly, navigation on a network means to decide at each node what outgoing edge to take next and then to move along that edge to the next node. Nodes thus play the role of "decision points". Navigation on a network also means to recognise the goal when approaching it. For the purpose of this work, we assume that the goal of the journey is a node in the network. If we further assume a simple graph underlying the network, then the edge to take next is identified by the node it leads to. Consequently, a route corresponds to a sequence of nodes in the network.

Finding a route that connects a given start node with a given target node is called route generation. Because there are typically many possible routes between the start and the target, route generation needs criteria for finding an optimal route. Typical criteria include shortest and fastest, but also "soft" criteria like fewest turns or most scenic are possible. If a traveller is new to an area, a route is only as good as it can be described. This indicates that there is another important "soft" criterion: the availability of a good route description.

Our goal is to find routes for which there exists a simple and unambiguous description in terms of landmarks. Such a landmark-based approach to route generation has applications in pedestrian and automobile navigation, where it will complement more traditional sign-following strategies. The inclusion of landmarks in the directions provided by navigation systems enhances comprehensibility and supports memorability by linking the network to the environment.

In order to achieve our goal, we first examine the role of landmarks in route descriptions (Section 2). Then we propose a new way to link landmarks with the network (Section 3), which provides the information we need to find routes with simple descriptions (Section 4). Finally, we discuss first results and indicate open problems for further research (Section 5).

2. Landmarks and Route Descriptions

Landmarks, as described by Lynch [Ly60], are visual points of reference that enhance the imageability of urban spaces. Specifically, landmarks are spatial objects in which observers do not enter, but rather use as reference from an external point of view. Typical examples of landmarks are towers, mountains, and monuments. For our purposes, we may simply assume that landmarks are spatially anchored objects that can easily and unambiguously be recognised by navigators. These two properties make landmarks useful for the description of routes. In many cases, however, landmarks are not point-like objects, but rather have a spatial extent (i.e., line or region). Route descriptions typically take advantage of these properties. The instruction "Along the river", for instance, refers to a landmark that has a line-like spatial extent, which can be followed. Therefore, we explicitly allow landmarks to be line- and region-like objects.

Figure 1 Two examples of typical landmarks (on the left the Kaiser Wilhelm Gedächtniskirche in Berlin and on the right a building that clearly contrasts with the environment in terms of colour).

A **route description** is a *sequence of instructions* that guides travellers along a pre-computed route. In human navigation, many different elements may be elements of route descriptions, as for instance street names, references to landmarks, proximal and directional instructions, etc. Research results indicate that some elements are better suited to the task than others

[MD01]. In particular, street names are introduced and described less often than landmarks, which highlights the cognitive importance of landmarks over street names [TD03a, TD03b].

We call a route description *landmark-based* if it makes significant use of landmarks. An alternative to landmark-based route descriptions are *network-based* route descriptions, which basically exploit the geometry of the network to make statements such as "turn left" or "go straight for 3 blocks". Such instructions work fine for grid-like networks, but are much harder to apply in irregular network structures, such as medieval town configurations. Furthermore, using landmarks in route descriptions provides some fault tolerance, allows for (limited) re-synchronisation, and is closer to human cognition. According to these considerations, a simple landmark-based route description might look as follows:

> *Face toward the Eiffel Tower and walk along the Seine until you see the Arc de Triomphe on your left.*

This example applies three different landmarks (i.e., the Eiffel Tower, the river Seine, and the Arc de Triomphe). At the same time it refers to these landmarks in three different ways, namely face toward, walk along, and walk until. These references fall into two classes: those depending on the traveller's heading (e.g., walk along) and those independent of it (e.g., face toward). Heading-independent references to landmarks allow for more robust navigation, but often it is easier to find heading-dependent references for instructions.

If we abstract away from nice wording, instructions are of the general form "move - relative to - landmark" and the set of possible route descriptions can be stated using formal grammar notation:

> description ::= { instruction }
> instruction ::= move_instr | stop_instr
> move_instr ::= "move" relative landmark
> stop_instr ::= "move" "until" landmark
> relative ::= "to" | "towards" | "past" | "along" | "across" | " through" | "around"

Here, {*A*} means zero or more instances of *A* and the vertical bar means “or”. The spatial relatives *towards*, *past*, *across*, and *through* are usually heading-independent and the relatives *along* and *around* are heading-specific. This set of spatial relatives is arbitrarily extensible, which is necessary because a system for routing and route descriptions may use any set of relatives. The term "move" was chosen for its independence of any specific mode of transportation – it can be replaced by walk or ride or even drive.

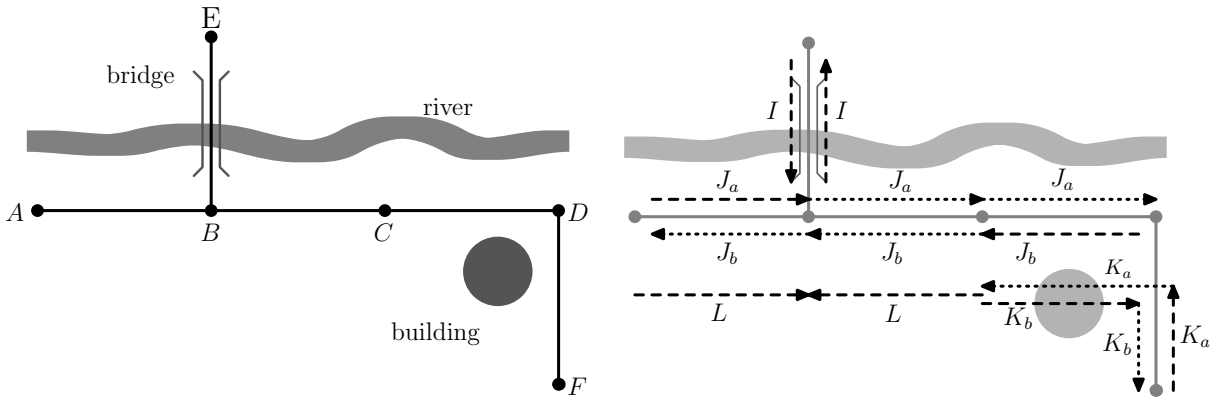
3. Linking Landmark-based Instructions to the Network

We need a way to attach landmark information to the underlying network. The primary problem is that landmarks are typically off the network and there are many different spatial relations between the network and the landmarks. We propose to cope with this problem by using complete *instructions* instead of individual landmarks, that is, we look at pairs (*spatial relative, landmark*) like "along the river" and not merely “the river”. The basic idea is that the landmark is used for both, the clear identification of a specific edge in the network from the navigator's point of view, as well as the action that is to be performed.

This bundling of landmarks with their relations to the network solves only part of the problem, for in fact we have a triple relation between the network, the landmark, and the traveller en route. For example, suppose there is a T-junction where the bar of the T is parallel to a river (e.g. edges EB, AB, and BC in Figure 2). Then, an instruction of the form "move along the river" is useless if the traveller just arrived along the stem of the T. However, once the traveller is properly oriented, this instruction can keep him busy for a long distance until another instruction stops him. An instruction like "move across the bridge" is useful whenever the traveller is close to a bridge, irrespective of the current orientation. Instructions of this type do not depend on the navigator's orientation with respect to a particular edge, as the bridge can be seen from the current point of view and hence, the next edge can clearly be identified.

These two scenarios indicate that an instruction identifies a *set of edges* in the network. That is, we translate textual instructions to sets of edges. The instruction "move across the bridge" translates to the set {BE,EB} and the instruction "move along the river" translates to the set {AB,BA,BC,CB,CD,DC}. An instruction thus is a pair *(a,E)* consisting of a text string *a* and a set of edges *E*. By this pairing, an instruction can be seen as a *description of the visual environment*.

Figure 2 A small network with nodes A,B,C,D,E,F, and three landmarks, a bridge, a river, and a building (left); to the right, the four instructions *I*="across bridge", *J*="along river", *K*="around building", and *L*="until bridge" are indicated. Indices *a* and *b* occur when an instruction is split into two.



Unfortunately, there are two problems with the mapping of the instructions to the network graph. First, in order to guarantee that the generated routes are unique, we require that any two edges of the same instruction never share a common source node. In other words, an edge *e* that belongs to the edge set *E(I)* of an instruction *I* has a unique successor *e*' in *E(I)*—if any. But this is not always the case. For example, the instruction "move along the river" from Figure 2 violates this requirement (it contains edges BA and BC, etc.). However, we can easily resolve this problem by splitting one instruction *J* into two instructions J_a and J_b with the same text string but disjoint edge sets that satisfy the required condition.

The other complication is that the usability of some of the edges of an instruction depends on the traveller's orientation, which in turn is determined by the last traversed edge. We cope with this problem by marking each edge of an instruction as being either dependent or free (i.e., independent of the traveller's previous edge).

More formally, an instruction $I=(a,I^f,I^d,c)$ consists of a text string *a* like "move along the river", a set of free edges I^f, a set of dependent edges I^d, and an instruction cost *c* that encodes how cognitively demanding an instruction is. Either of the two sets may be empty but they must not overlap. Note that *E(I)* is the union of I^d and I^f. To guarantee unique routes another condition must hold: the intersection $E(I_1) \cap E(I_2)$ of the edges of any two instructions I_1 and I_2 is a simple path.

This results in the following set of instructions for the example in Figure 2:

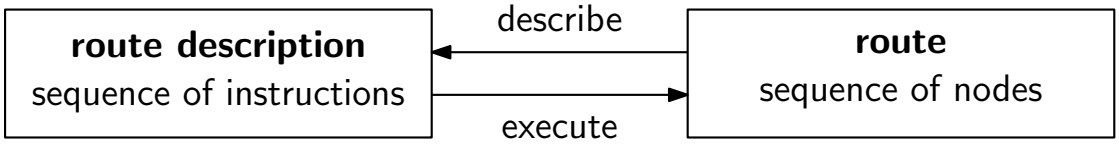
- I = ("move across bridge", {BE,EB}, ∅, 1)
- J_a = ("move along river", {AB}, {BC,CD}, 1)
- J_b = ("move along river", {DC}, {CB, BA}, 1)
- K_a = ("move around building", {FD}, {DC}, 1)
- K_b = ("move around building", {CD}, {DF}, 1)
- L = ("move until bridge", {AB,CB}, ∅, 1)

Note how two of the textual instructions have been split to ensure that there are no two edges in an instruction that share a common source node. Instruction costs are all set to one. These instructions can be used to guide a traveller from F to E (around building, along river, until bridge, across bridge), but not from E to F, because there is no free edge in "along river" that can be joined once the traveller is at node E. We can now either resort to left/right instructions or accept that this specific point is difficult from a navigation point of view.

4. Generating Routes with Simple Descriptions

Routing and route description are related in the sense that a route that cannot be described (and thus cannot be communicated) is not of much use to the traveller. Given a route description, its execution (i.e., following sequentially the instructions in the route description) either fails or results in an *induced route*. If this induced route equals the original route, then the route description is said to be *valid*. In order to get valid route descriptions, we must consider the complete route description (sequence of instructions) while generating the route (sequence of nodes); see also Figure 3.

Figure 3 Relationship between routes and route descriptions.



Route execution is an iterative process. The following procedure describes, in our terminology of instructions and edges, what a human traveller would do. Given a start node s and a sequence of instructions $(I_1,I_2,...,I_k)$, we set e to be the only free edge in I_1 with source node s. Let I_i be the current instruction. We then proceed as follows:

1. If e is one of the free edges of instruction I_{i+1}, continue with instruction I_{i+1}.

2. If e is one of the dependent edges of instruction I_{i+1} and there is a successor e' of e in I_{i+1}, append e' to the path, set e to e', and continue with instruction I_{i+1}.

3. If e has no successor in instruction I_i , then either we have reached the goal (e leads to the goal node), or we can continue with a free edge of the next instruction I_{i+1} starting from the tail of e. If none of these two cases is applicable the route description is not valid.

4. Otherwise, there is a successor e' of e; append e' to the path, set e to e', and stick to instruction I_i.

This procedure implements the strategy to switch from one instruction to the next as soon as possible, which is consistent with the basic strategy to use information as soon as it becomes available.

The route generation algorithm requires as input:

- a directed graph representing the network,
- a set of instructions, that is, tuples (s,I^f,I^d,c), and
- a source node s and a target node t

In order to find a route with a valid route description, we construct an auxiliary graph based on the original network and the available instructions, such that a standard Dijkstra shortest path algorithm [Dij59] will produce a "shortest" route along with a valid route description. The route produced shall be "shortest", both, in distance and in terms of the number and types of instructions used to describe it. To achieve this goal, the types of instructions are weighted by the cost for an instruction $c(I)$, which we assume we will have available. The basic idea of the auxiliary graph is to expand the nodes and add the edges of the instructions as additional edges to the auxiliary graph.

More precisely, the auxiliary graph is denoted by $G'=(V',E')$ with edge costs being pairs of non-negative real numbers. The first component of such a pair models the distance covered by the edge while the second component models the instruction cost.

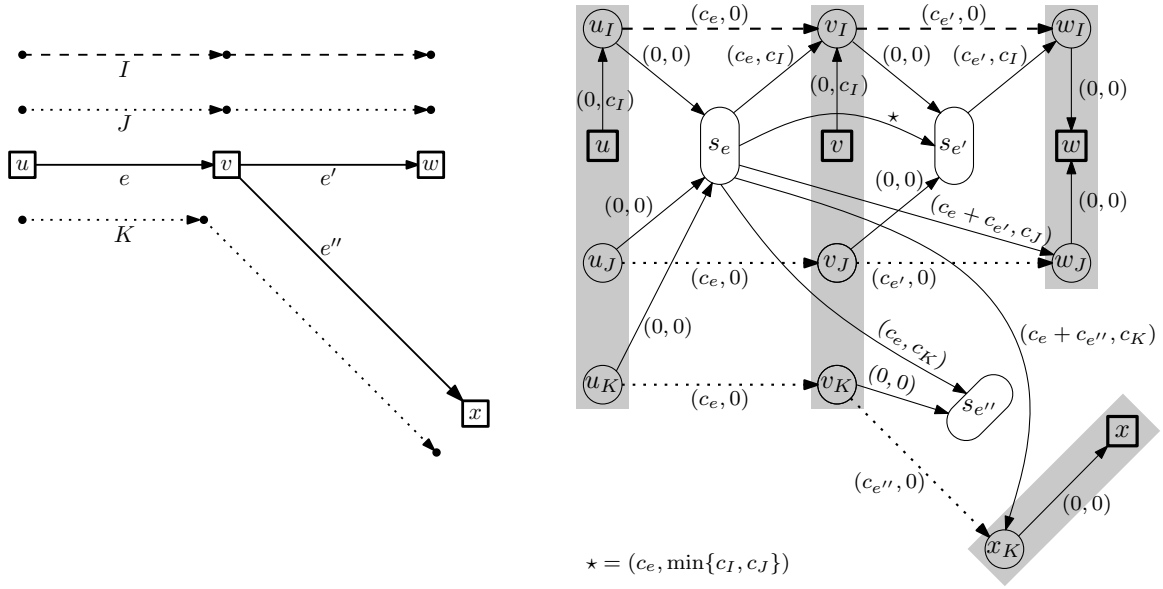
Each node v in the original graph G is replaced by a set of nodes, see the shaded boxes in Figure 4. First, for each node v in G there is a copy of v in G', see the bold square nodes in Figure 4. Second, for each instruction I that contains one of the incoming or outgoing edges of v there is a node v_I in G', see the circular nodes in Figure 4. Third, G' contains a node s_e for each edge e of G, see the rectangles with rounded corners in Figure 4. This node makes it possible to switch on e from one instruction to another.

Let $e=(u,v)\in E$ be an edge in the original graph and let $\bar{I}$ denote the set of all instructions. We insert five types of edges in G'. In Figure 4, free edges of an instruction are marked by dashed line segments and dependent edges by dotted line segments. All edges are annotated by their costs.

- *Start edges*. For each instruction I with $e\in I^f$ we create in G' the edge (u,u_I) with cost $(0,c(I))$.
- *Final edges*. For each instruction I with $e\in E(I)$ that does not contain an outgoing edge from v we introduce the edge (u_I,u) with cost (0,0).
- *Instruction edges*. For each instruction I with $e\in E(I)$ we introduce the edge (u_I,v_I) with cost $(c(e),0)$.

- *Switch edges*. For each instruction I with $e \in E(I)$ we introduce the edge (u_I, s_e) with cost (0,0). In addition, if $e \in I^f$ we introduce the edge (s_e, v_I) with cost $(c(e), c(I))$. Otherwise (if $e \in I^d$), we introduce the edge (s_e, w_I) with cost $(c(e)+c(e'), c(I))$ if there is an edge $e'=(v,w) \in E(I)$.
- *Double-switch edges*. For each edge $e'=(v,w)$ in G with $I(e')=\{I \in \bar{I} \mid \{e,e'\} \subseteq E(I)\}$ being not empty we introduce the edge $(s_e, s_{e'})$ with cost $(c(e), \min_{I \in I(e')} c(I))$.

Figure 4 A simple graph with four nodes and three instructions (left) and the corresponding auxiliary graph (right).



The auxiliary network has complexity $O(\Delta \cdot |E| + |\bar{I}|)$, where $\Delta = \max_{v \in V} \deg(v)$ is the maximum degree of G and $|\bar{I}| = \sum_{I \in \bar{I}} |E(I)|$. It can be shown that valid route descriptions along with the induced routes correspond to paths in the auxiliary graph. Hence, we can apply a shortest-path algorithm like Dijkstra's algorithm to the auxiliary graph in order to compute the desired route description. The sequence of instructions that makes up the route description can be found by tracing the route in the auxiliary graph and adding instructions to the route description as they come up along the route.

Applying Dijkstra's algorithm to the graph G' with edge costs being only the first component of c' yields the shortest route with respect to the original costs c among all routes with a valid route description. The source and target nodes for Dijkstra's algorithm in G' are the

corresponding copies of the source and target of the query in G. Analogously, the route with the best description can be determined when the edge costs equal the second component of c'. Using pairs as edge costs yields lexicographically first solutions: For example, if the first component of the pair denotes the instruction cost and the second one the original cost, among all routes with best description cost one with shortest distance is computed.

5. Discussion and Outlook

In this paper, we propose a new approach to route generation that produces a description of the route along with the route itself. This is motivated by the observation that a route is only as good as it can be described. The route description produced consists of instructions that refer to landmarks, for it is known that landmarks are useful to travellers. The generation of the route and its description builds on the concept of instructions that in turn identify edges in the network. For any single instruction can guide the traveller along several edges, we have nicely implemented what is known as *chunking* (one piece of information identifies an extended "chunk" of a route) [KTH03]. Both, the use of landmarks and the form of chunking that emerges from our approach are known to correspond closely to human cognition and thus should be more easily usable than route descriptions that build on the network geometry alone.

Research in wayfinding and navigation has shown that travellers are prepared to take suboptimal routes in terms of travel time if these routes are potentially easier to describe and to follow [SV86, SVW85]. Hence, it is not a problem if our routes are somewhat longer than the geometrically shortest route. We plan to extend the algorithm to another parameter as input that controls how much longer the generated route can be as compared to the shortest possible route. This will no longer lead to a unique optimal solution, but rather to a set of Pareto-optimal solutions.

Other open questions remain, for example, is there a way to automatically deduce instructions and the associated cost and edge sets from surveying or other existing data? How well does the algorithm perform on large data sets? And finally, how do our route descriptions compare with route descriptions produced by humans?

Acknowledgments

A. Wolff acknowledges support by the German Research Foundation (DFG grant WO 758/4-2). U.J. Rüetschi and D. Caduff acknowledge support by the Swiss National Science Foundation (grant 205120-109621).

References

Dij59 Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, pages 269-271, 1959.

KTH03 Klippel, A., Tappe, H., Habel, C.: Pictorial Representations of Routes: Chunking Route Segments during Comprehension. Spatial Cognition 2003: 11-33

Ly60 Lynch, K. (1960): The Image of the City. The MIT Press.

Mo05 Montello, D. R. (2005). Navigation. In P. Shah & A. Miyake (Eds.), The Cambridge handbook of visuospatial thinking (pp. 257-294). Cambridge: Cambridge University Press.

MD01 Michon, P. E. and Denis, M. (2001): When and Why Are Visual Landmarks Used in Giving Directions? In Spatial Information Theory, COSIT, edited by Montelo, D.R., Springer Verlag, LNCS 2205.

RT04 Rüetschi, U.J. and Timpf, S. (2005): Modelling Wayfinding in Public Transport: Network Space and Scene Space. Pages 24-41 in Spatial Cognition IV, LNAI 3343. Springer-Verlag Berlin Heidelberg, 2005.

SV86 Streeter, L. A., & Vitello, D. (1986). A profile of driver's map-reading abilities. Human Factors, 28, 223-239.

SVW85 Streeter, L. A., Vitello, D., & Wonsiewicz, S. A. (1985). How to Tell People Where to Go: Comparing Navigational Aids. International Journal of Man-Machine Studies, 22(5), 549-562.

TD03a Tom, A., & Denis, M. (2003a). Referring to landmark or street information in route directions: What difference does it make? In International Conference on Spatial Information Theory, COSIT (Eds, Kuhn, W., Worboys, M. and Timpf, S.) Springer-Verlag, Kartause Ittingen, Switzerland, pp. 362-374.

TD03b Tom, A. & Denis, M. (2003b). Language and spatial cognition: comparing the roles of landmarks and street names in route instructions. In Applied Cognitive Psychology, vol. 18, no. 9, 1213-1230.