Amanda Hosler
8/18/2004

NUMACK's Route Planning: Automatically Generating a Route and its Natural Language Description

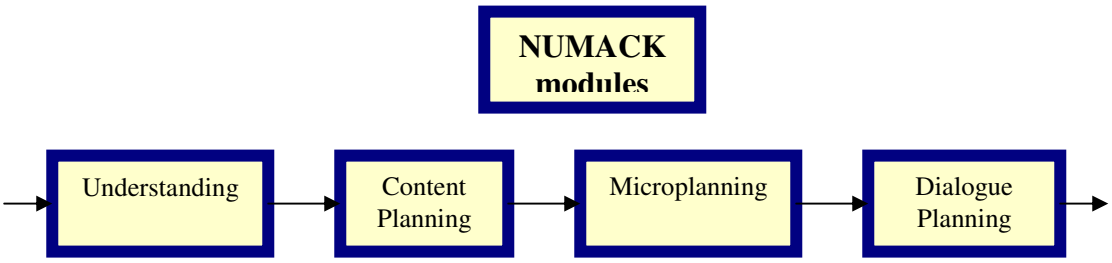
Statement of Problem

NUMACK is an Embodied Conversational Agent (ECA). An ECA is a type of user/computer human-computer interface; a virtual being that communicates with humans using speech and gesture, rather than using the standard screen and keyboard interface. ECAs are interesting because from a research standpoint, programming an ECA and watching how it interacts with humans shows the programmer what actions are more natural, making the process a learning experience on human behavior.

A central research contribution from NUMACK is the study of the relationship between gesture and speech and the use of gesture in a spatial information context. This topic is interesting because many people use gesture when communicating spatial information, so we can learn whether gestures are saying something that speech is not, what are the meanings that different types of gestures express, etc. NUMACK will interact with humans by answering questions and giving directions around the Northwestern University campus. He'll use speech recognition and head-tracking to understand a question, determine the answer, and use speech and gesture in his response. In terms of HCI, NUMACK could help increase usability in the ECA interface, since information being conveyed over multiple channels will increase redundancy which improves understanding.

NUMACK's response to the user will be created using extended versions of the SPUD and BEAT systems. SPUD is a natural language generation system (Stone and Doran, 1997). Paul Tepper, who has worked on SPUD, is implementing and extending SPUD to express spatial information and gesture. He is also implementing a hierarchical structure in prolog of a spatial information knowledge base. Stefan Kopp implemented his previous work—automatic inverse kinematics in animation graphics—to a skeletal structure, which he also created, for NUMACK. He is also working on extending and implementing BEAT, which is a gesture and speech generator that focuses on timing those actions using a pipelining approach, for NUMACK.

Figure 1.



NUMACK has an understanding module that recognizes the meaning of the user's speech and eye gaze input. The understanding module then sends a message to the content planner module, phrasing the input in a way the content planner can understand. The content planner module answers the input, and sends a coded message of that answer to the microplanner module. Microplanning then makes the message more language detailed, chooses appropriate lexicon, sentence structure, etc., and then collaborates with the dialogue planner to create the discourse. An area of NUMACK that needs more attention is content planning.

A content planning module consists of a knowledge representation or knowledge database, and usually some planning, i.e. a list of actions that does something with the relevant knowledge [7]. A simple content planner could be sent the message "what is A", look at its knowledge database and see "A = star" and return "star". In that case planning isn't required. The input passed to NUMACK's content planner is a little more complex. In general, the input will look like, "how do I get from A to B". The content planner has to provide an answer; in this case, the route from A to B. My work addresses this issue of content planning for routes, in particular the kind of content planning that can be translated in NUMACK's later modules into both speech and gesture. The remainder of this paper describes my work on implementing a route planner for NUMACK.

Previous Research

Computer generated routes and direction giving language have become a relatively popular research subject and popular among everyday people, so a lot of time has gone into getting the most usability out of them. Here are some things we know about them:

- Computer generated routes have become very popular and demanded. Route planners such as Mapquest, Microsoft AutoRoute Great Britain, and Microsoft Streets and Trips automatically generate a map showing the route from A to B. However, these route maps are often more difficult to use then hand drawn maps because they do not distinguish between essential and extraneous information, and therefore lack clarity [1]. Essential information is considered by Agrawala and Stolte (2001) to be, in the instance of driving directions, a list of road names and turn directions. The mentioned route planners include a map of the entire surrounding area, creating clutter that interferes with essential information.

- A study conducted by P.-E. Michon and M. Denis shows that spoken route directions include numerous references to landmarks, and that when confronted by directions that only include a list of street names and turn directions, people react to the absence of landmarks [6]. Furthermore, the same study showed that landmarks are used most often at the start and end of a route, when reorienting, and when large open spaces occur along the route.

- Route finding is guided by our visual perception, i.e. we see our surroundings and decide where to go when following a route, so features like landmarks are natural guidelines in finding the way [5]. NUMACK will be communicating face to face via natural language, so directional, temporal, and spatial language are all useful in describing the route. For instance, "turn to the right", "go until the end of the fence", and "the fence will be on your right".

- The best route for one person may not be the best route for another [9]. This can be extended to the best route descriptions. Person X may be familiar with a different landmark than person Y, or person X may need more information when some specific landmark has already been mentioned.

- There is a difference between using specific landmarks and using regions or areas. For instance, naming a specific building or buildings that someone will pass on their right vs. saying "there will be several buildings on your right". Sometimes it is more effective in a route description to point out regions of areas vs. specific placefs [10].

Perhaps the most relevant previous work is that done by Christopher Habel concerning multimodal route instructions [4]. His paper features what to communicate, or what to say, when turning a route into a natural language description. The NUMACK architecture also uses an incremental approach to the end to end system, similar to Habel's INC. I don't know if his system is completed. But this work with NUMACK is focused on creating a program which generates the route from A to B and also returns landmarks, and directional and spatial information to be passed on to a microplanner module to plan speech and gesture for an Embodied Conversational Agent.

My Approach

NUMACKs content plannerwill consist of a knowledge database, including facts about the domain. It will also create a plan for the knowledge representation. This means that it will iterate through a route planning algorithm, and plan the route based on information about routes found through previous work, and empirical subject data. It will then plan the knowledge representation of the route using an algorithm that expands the route into a description of the route made up of directional changes and location changes. This will be passed to the microplanner to map these descriptions onto words that will be used during dialogue.

On the first iteration of the project, we went through the videotaped data of 28 subjects giving directions for a few specific routes at Northwestern University. Each subject gave directions for Frances Searle to the Allen Center. Other routes included Allen to Norris, Allen to the Observatory, each of those places to University Hall, and to the Arch. Landmarks that the subjects widely used were noted as key landmarks for those routes. For this iteration of the project, we focus on the route described from Frances Searle to the Allen Center. As shown in Figure 1, the most popular landmarks for this route included Cook Hall, the parking lot directly to the East of Frances Searle, and the parking lot directly to the West of the Allen Center. The road going around to the left of Frances Searle and the grassy area between the end of that road and the parking lot are both mentioned half of the time. In fact, for almost all the data, parking lots are mentioned when they are visible from a first person perspective of the route, suggesting that parking lots are generally used as landmarks in this setting. Landmarks were also mentioned most around points of reorientation, or the beginning or end of a route, as Michon and Denis found.

Figure 2.

Frances Searle to Allen Center			
Path	**Landmark**	**Number**	**Percentage**
	DNE Sign	1	5.56%
left around Searle to p1 to p2	Roundabout	6	33.33%
count = 18 people out of 28	Alleyway/cul-de-sac	9	50.00%
Chose this path	Bushes	4	22.22%
From Searle to Allen.	Garbage cans	5	27.78%
Percentages are out of 18	Cook	10	55.56%
	Pancoe	1	5.56%
	Grass Patch / Path	9	50.00%
	ParkingLot 1 (east of Searle)	18	100.00%
	Field	2	11.11%
	Road	7	38.89%
	SPAC	1	5.56%
	Construction	7	38.89%
	Lampposts	1	5.56%
	Parking Lot 2 (west of Allen)	17	94.44%

The route from Frances Searle to Cook and the landmarks mentioned most have been added to a prolog knowledge database, along with about 10 other landmarks / buildings that were also mentioned in other routes, and about 15 other points in the route. These were added to create multiple potential routes so that the route planner has multiple route possibilities that it has to decide between. This database was created using Prolog because SPUD is already set up in Prolog and at this point it will be most effective to create the entire knowledge database in the same language, and eliminate the need for a communication module. Prolog is a logic programming language, so it holds a database of facts and rules, and the user queries whether something is true or false using a command line. Prolog is ideal for search problems like this one, because information

about a graph is easily kept in the database, and easily updated. When a query about a route is the input to the program, Prolog simply searches the database to see if the necessary facts are there to create the desired route. This means a lot less code is needed to do same thing that it would take several classes to do in Java or C++, because of the structure that those languages require. However, while the Prolog code can be done in one file, this might mean something harder to read for anyone who is not the author, because the structure of the different classes is not forced. Some structure can be achieved by splitting the Prolog code into a couple different well organized files.

After testing the prolog code on the first facts entered in the knowledge base more information, that is route points and landmarks, should be coded into the database. We were going to try to determine a theory about picking routes and key landmarks based on the empirical data supplied by the videotaped subjects. Do subjects describe how to get from A to B using the shortest route, the popular route, or the most accessible route, etc. What kinds of patterns are seen in the key landmarks picked, etc. However, the experiment was held to get good spatial gestures from the subjects, not to get good information about routes and landmarks. For some of the subjects, if they didn't know where exactly one of the route stops was (like Allen), one of the experimenters would point out the way to get there, thus biasing all the route data for the experiments. However, the subjects had to walk to each of the places and determine what landmarks they were going to use to describe the route to someone else, so the landmark data can still be used. The second iteration of coding relies on our having theories about landmarks.

So far the route planning knowledge base contains about 16 different labeled vertices that are actually part of the path, i.e. a location on the path en route to some building or landmark. These points on the path itself are represented as nodes of a graph, and the paths from one point to its neighbors are represented as edges, with the distance between the two nodes as the edge weight. The database also contains some knowledge about these nodes and edges, for instance relative location between nodes and landmarks (north, west, etc.). Right now the knowledge base contains about 12 A's and a B's (landmarks that someone at the Northwestern U campus may want directions to, i.e. buildings, the lagoon, etc.). The database created so far includes a few classes of facts that are hard coded into the knowledge base. These are:

- `vertex(X)` where X is a vertex in the graph, and a point in the path that the route traveler may walk over while traversing the route.
- `neighbors(X, NN)` where X is a vertex in the graph, and NN is a list of vertices that share an edge with X.
- `coordinates(X, Cx, Cy)` where X is a vertex in the graph OR a landmark and Cx and Cy are the x and y coordinates of X.
- `rel_dir(A, B, C)` where A is a vertex in the graph, B is a landmark and C is the direction (e.g. north, west) of B from A.
- `path_by_entrance(A, B, C)` where A is a building landmark, B is the direction (e.g. north, west) of the door on the building, and C is the vertex in the graph near the door of the building.

- `abs_dir_to_rel(A, B, C)` where A is the current absolute direction that a route traveler is facing, B is the new absolute direction that the route traveler turns to face, and C is the relative direction (e.g. left, right) that the traveler will turn.

There are a set of rules that make a route finder implemented using the A* algorithm. This algorithm is widely known to be an optimal search, as long as the heuristic is consistently an underestimate [7]. In this prolog code the search is optimal because the heuristic simply calculates the distance straight from the current node to the final node, which is the least possible distance it could travel to get to the final node, clearly an underestimate. This has been tested first using the route from Frances Searle to the Allen Center. There are already many different routes that can be queried from the command line, as long as the necessary information for path points and landmarks are already set up in the knowledge database.

Here a finding was interesting. Before implementing the A* algorithm, a lowest cost first search algorithm was being used, i.e. there wasn't a heuristic implemented yet, it was only following the cost of the path so far. When running the program querying the route from Frances Searle to the Allen Center, a list of routes was outputted in the same order as the user data from most to least commonly described route. But when implementing the A* algorithm, the order that the routes were outputted changed slightly, so that it didn't mirror the user data. But as stated before, the experiment was conducted to college gesture data, not information about picking routes, and this particular route was especially biased, because the experimenter told several people how to go from the start point to the end point.

Figure 3.

User queries:

```
| ?- find_route(frances_searle, allen, Route).
Route = [p1, p6, p4] ?
yes
```

Code:

```
find_route(A, B, Route):-
        route_start_and_end(A, B, X, Y),
        assert(goal(Y)),
        heuristic(X, Y, C),
        search([node(X, [], 0, C)], BackwardsRoute), !,
        reverse(BackwardsRoute, Route).
```

As shown in figure 3, the query input asks for directions from one landmark to another (buildings most likely), and using route_start_and_end/4 Prolog finds the appropriate path node to start the route, and the appropriate end node to be the goal node, the ending

point. Using assert/1, the fact that Y is a goal node is entered into the knowledge database. Then the heuristic is calculated from the start node to the finish one. At each node, the heuristic to the goal node is calculated. The cut (!) is used so that after the first route is returned, if the user wants to be presented with a different route, the program answers 'no' instead of giving the next route. With the search rule, the route is returned in reverse order, calling for the quick reverse of the route before returning it with the query.

Figure 4.

User queries:

```
| ?- describe_route(frances_searle, allen, Describe).
Describe =
[start_location(frances_searle,direction(behind)),change_
orientation(left),change_location(hogan,direction(left)),
change_orientation(left),change_location(pancoe,direction
(left))] ? yes
```

Code:

```
describe_route(A, B, Describe):-
        find_route(A, B, Route),
        face_direction(A, Route, Point, NRoute, Cur_Face_Dir),
        list_dir_loc_changes(Cur_Face_Dir, Point, NRoute, Describe).
```

Once the route exists, (after find_route(A, B, Route)) the program gathers knowledge about the route from the knowledge database and attaches the directional and spatial information to each step in the route. The rule face_direction/5 returns the Cur_Face_Dir, that is the current absolute direction that the route traveler faces at the beginning of the route. From this information, the face direction can change at each route point, and the turn direction can be noted with it (i.e. left, right). Also, location changes can also be noted by using the database fact rel_dir(A, B, C) to attach a landmark to the route description at each point in the route. This concludes the first iteration of the project. This iteration is useful in terms of programming because it will figure out the way the information will look and be passed into and out of the program, and it should be ready at this point to be set up with the end to end system.

The output generated from the program should be directly compatible with NUMACK's microplanning module. The route description should be able to be mapped to specific words that will be brought up for dialogue between NUMACK and the user. This has not been set up and tested yet. Similarly, this spatial and directional route information being returned will be input to determine NUMACKs gestures during speech.

The End Goal

The product will be a program written in prolog that takes two arguments, starting and ending points A and B, and finds a list of nodes which consist of the route from A to B. It will then collect directional, spatial, and temporal information from the knowledge database to return with the route. The program needs to be fast, because NUMACK will need to be able to answer these questions in real time. It also needs to be very easily readable, since this is the first development of the project and might need to be looked at and changed many times.

This will be successful if communication from this content planning module to NUMACK's microplanning module is easy and smooth, and if the information is indeed what the microplanner needs to know.

Future Work

As stated earlier the route description should be able to be mapped to specific words that will be brought up for dialogue between NUMACK and the user. This has not been set up and tested yet, and this is the next step that needs to be done to help the end to end system. A similar set up needs to be created for mapping to gesture planning.

On the second iteration of the project, many more landmarks should be added to the database, which are picked according to the patterns we found in the subjects. From [6] we know that landmarks are 3D and 2D objects that are pointed out near the beginning and end of the route, and when changing direction or coming upon an open space where it is not clear which direction to go. Many path points also need to be added to the knowledge database, creating a larger domain, and many more routes that NUMACK can describe. In some instances they can be added to clean the knowledge that already exists by creating a more accurate representation of the points that need to be traversed in a path

During NUMACK's dialogue with the user, his discourse history should be able to assess which landmarks and directional, temporal, and spatial language need to be elaborated (for instance, if NUMACK has already said "a parking lot will be on your right" and the user asks for more information, he can then say "the lake will be in front of you" or "a statue will be on your left"). Should this be the responsibility of the content planner or the microplanner? Perhaps a query can be sent to the content planner which then iterates through the route and pulls up more, or different, landmarks at the elaboration point. This would certainly be useful, and this also is imperative for gesture.

In the future it would be nice to automatically generate the knowledge database on the basis of a map, that is, read in the map, and come out with path points and landmarks, coordinates, etc.

Resources

1. Agrawala, M., & Stolte, C. (2001). *Rendering Effective Route Maps: Improving Usability Through Generalization*. In Proceedings of the 28th annual Conf. on Computer Graphics and Interactive Techniques, pp. 241-249.

2. Clocksin, W. F., & Mellish, C. S. Programming in Prolog. Fourth Edition. Sringer-Verlag Berlin Heidelberg New York. 1994.

3. Duboue, P., & McKeown, K. (2002). *Content Planner Construction via Evolutionary Algorithms and a Corpus-Based Fitness Function*. In Proceedings of the Second International Natural Language Generation Conference.

4. Habel, C. (2003). *Incremental generation of multimodal route instructions*. Technical Report- American Association for Artificial Intelligence, pp. 44-51.

5. Maaß, W., & Wazinski, P. (1993). Gerd Herzog. *VITRA GUIDE: Multimodal Route Descriptions for Computer Assisted Vehicle Navigation*. In Proceedings of the Sixth Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-93, pp. 144-147, Edinburgh, Scotland.

6. Michon, P.E., Denis, M. (2001). *When and Why Are Visual Landmarks Used in Giving Directions?* COSIT 2001, pp. 292-305.

7. Poole, D., Mackworth, A., & Goebel, R. Computational Intelligence: A Logical Approach. Oxford University Press, Inc. New York, 1998. ch. 4.

8. Reiter, E., & Dale, R. Building Natural Language Generation Systems. Cambridge University Press, 1999. Ch 4.

9. Rogers, S., Fiechter, C. N., & Langley, P. (1999). *An Adaptive Interactive Agent for Route Advice*. In Proceedings of the third annual conference on Autonomous Agents, pages 198-205, Seattle, WA, 1999.

10. Sterling, L., & Shapiro, E. The Art of Prolog. Massachusetts Institute of Technology Press. 1994

11. Wiener, J. M., Mallot, H. A. (2003). *Route Planning in Hierarchically Structured Environments: From Places to Regions*. EuroCogSci (2003).