Multimodal Maps: An Agent-based Approach

Adam CHEYER and Luc JULIA

SRI International
333 Ravenswood Ave
Menlo Park, CA 94025 - USA

ABSTRACT

In this paper, we present a prototype map-based application that accepts handwritten, verbal and gestural requests issued in a synergistic fashion. The application is distinguished by its comparatively rich natural language capabilities, access to existing data sources including the World Wide Web, and a mobile handheld interface. To implement the described application, a hierarchical distributed network of heterogeneous software agents was augmented by appropriate functionality for developing synergistic multimodal applications.

KEYWORDS

Agent Architecture, Multimodal Interface, Map, Distributed Application

1. INTRODUCTION

Maps were once prized possessions enabling the adventurous to navigate unfamiliar lands. Nowadays, maps are given away free at gas stations, and their esteemed worth has greatly diminished. However, as more of the physical world is given electronic representation via the Internet, Compuserve or other online information services, new advances in human computer interfaces may make it again fashionable, fun and efficient to navigate using maps.

One developing research area that may have a significant impact on map-base interfaces is that of multimodal dialog systems; researchers now realize that some tasks are more easily expressed by voice, others by handwriting or typing, still others by gestures or direct manipulation. Ideally, these input modalities can be combined synergistically [1], allowing a free and natural mixture of the modes.

Although synergistic multimodal interfaces are starting to appear [3,6], existing architectures provide insufficient functionality to support certain requirements of our project:

- State-of-the-art natural language and speech recognition systems should be incorporated into the system in order to accept complex linguistic input. Current synergistic multimodal applications can often only accept very simple commands such as "put that there."
- The user interface should be light and fast enough to run on a handheld PDA while able to access applications and data that may require a more powerful machine.
- The multimodal interface should be able to access a wide variety of data sources, including information stored in HTML form on the World Wide Web.

The organization of this paper is as follows: Section two describes a prototype multimodal map-based application for the domain of travel planning. Sections three and four describe the agent-based approach we used to implement the application and the work on which it is based. Section five presents a brief comparison with other related work in the field. And finally, section six will outline our conclusions and future directions.

2. A MULTIMODAL MAP APPLICATION

Keeping in mind the desired objectives listed in the previous section, we developed a prototype multimodal application for the travel planning domain. As illustrated in Figure 1, the user is presented with a pen sensitive map display which allows drawn gestures and written natural language statements to be combined with spoken input. As opposed to a static paper map, the location, resolution, and data presented by the map change, according to the requests of the user. Objects of interest, such as restaurants, movie theaters, hotels, tourist sites, municipal buildings, etc. are displayed as icons. The user may ask the map to perform various actions. For example :

- *distance calculation* : e.g. "How far is the hotel from Fisherman's Wharf?"
- *object location* : e.g. "Where is the nearest pos office?"

- *filtering* : e.g. "Display only the French restaurant within 1 mile of this hotel."
- *information retrieval* : e.g. "Show me all availabl information about Alcatraz."

The application also makes use of multimodal (multimedia) output as well as input: video, text, sound and voice can all be combined when presenting an answer to a query.

During input, requests can be entered using gestures (e.g. arrows, lines, circles, cross-out or delete marks, etc.), handwriting, voice, or a combination of pen and voice. For instance, in order to calculate the distance between two points on the map, a command may be issued using:

- *gesture*, by simply drawing a line between the two points of interest.
- *handwriting*, by writing "What is the distance from the post office to the hotel?"
- *voice*, by speaking the same request.
- *synergistic combination of pen and voice*, by speaking "What is the distance from here to this hotel?" while simultaneously indicating the specified locations by pointing or circling.

Notice that in our example of synergistic combination of pen and voice, the arguments to the verb "distance" can be specified before, at the same time, or shortly after the vocalization of the request to calculate the distance. If a user's request is ambiguous or underspecified, the system will wait several seconds and then issue a prompt requesting additional information.

The user interface runs on pen-equipped PC's or a Dauphin handheld PDA using either a microphone or a telephone for voice input. The interface is connected either by modem or ethernet to a server machine which will manage database access, natural language processing and speech recognition for the application. The result is a mobile system that provides a synergistic pen/voice interface to remote databases.

In general, the speed of the system is quite acceptable. For gestural commands, which are handled locally on the user interface machine, a response is produced in less than one second. For handwritten commands, the time to recognize the handwriting, process the English query, access a database and begin to display the results on the user interface is less than three seconds (assuming an ethernet connection and good network response). Solutions to verbal commands are displayed in three to seven seconds after the end of speech has been detected; partial feedback indicating the current status of the speech recognition is provided earlier.

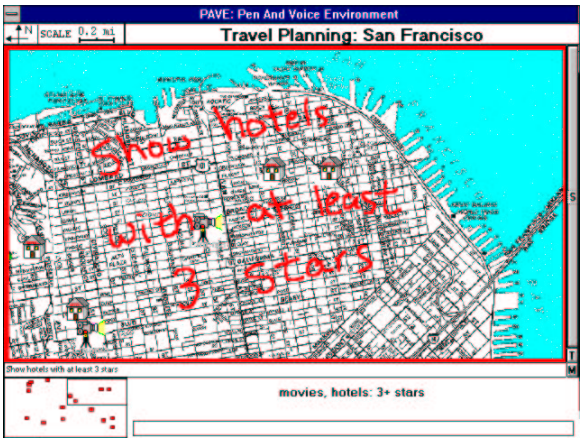

Figure 1: Multimodal Application for Travel Planning

3. BUILDING BLOCKS

In order to create the application described in the previous section, we chose to augment a proven agent- based architecture with functionalities developed for a synergistically multimodal application. The result is a flexible methodology for designing and implementing distributed multimodal applications.

3.1 Open Agent Architecture

The Open Agent Architecture (OAA) [2] provides a framework for coordinating a society of agents which interact to solve problems for the user. Through the use of agents, the OAA provides distributed access to commercial applications, such as mail systems, calendar programs, databases, etc.

The Open Agent Architecture possesses several properties which make it a good candidate for our needs:

- An Interagent Communication Language (ICL) and Query Protocol have been developed, allowing agents to communicate among themselves. Agents can run on different platforms and be implemented in a variety of programming languages.
- Several natural language systems have been integrated into the OAA which convert English into the Interagent Communication Language. In addition, a speech recognition agent has been developed to provide transparent access to the Corona speech recognition system.

- The agent architecture has been used to provide natural language and agent access to various heterogeneous data and knowledge sources.
- Agent interaction is very fine-grained. The architecture was designed so that a number of agents can work together, when appropriate in parallel, to produce a fast response to a query.

The architecture for the OAA, based loosely on Schwartz's FLiPSiDE system[10], uses a hierarchical configuration where client agents connect to a "facilitator" server. Facilitators provide content-based message routing, global data management, and process coordination for their set of connected agents. Facilitators can, in turn, be connected as clients of other facilitators. Each facilitator records the published functionality of their sub-agents, and when queries arrive in Interagent Communication Language form, they are responsible for breaking apart any complex queries and for distributing goals to the appropriate agents. An agent solving a goal may require supporting information and the agent architecture provides numerous means of requesting data from other agents or from the user.

The Open Agent Architecture provides the capability of accessing distributed knowledge sources through natural language and voice, but it is lacking integration with a synergistic multimodal interface.

3.2 TAPAGE

TAPAGE (edition de Tableaux par la Parole et la Geste) is a synergistic pen/voice system for designing and correcting tables.

To capture signals emitted during a user's interaction, TAPAGE integrates a set of modality agents, each responsible for a very specialized kind of signal [3]. The modality agents are connected to an "interpret agent" which is responsible for combining the inputs across all modalities to form a valid command for the application. The interpret agent receives filtered results from the modality agents, sorts the information into the correct fields, performs type-checking on the arguments, and prompts the user for any missing information, according to the model of the interaction. The interpret agent is also responsible for merging the data streams sent by the modality agents, and for resolving ambiguities among them, based on the its knowledge of the application's internal state. Another function of the interpret agent is to produce reflexes: reflexes are actions output at the interface level without involving the functional core of the application.

The TAPAGE system can accept multimodal input, but it is not a distributed system; its functional core is fixed. In TAPAGE, the set of linguistic input is limited to a *verb object argument* format.

4. SYNTHESIS

In the Open Agent Architecture, agents are distributed entities that can run on different machines, and communicate together to solve a task for the user. In TAPAGE, agents are used to provide streams of input to a central interpret process, responsible for merging incoming data. A generalization of these two types of agents could be :

Macro Agents: contain some knowledge and ability to reason about a domain, and can answer or make queries to other macro agents using the Interagent Communication Language.

Micro Agents: are responsible for handling a single input or output data stream, either filtering the signal to or from a hierarchically superior "interpret" agent.

The network architecture that we used was hierarchical at two resolutions – micro agents are connected to a superior macro agent, and macro agents are connected in turn to a facilitator agent. In both cases, a server is responsible for the supervision of its client sub-agents.

In order to describe our implementation, we will first give a description of each agent used in our application and then illustrate the flow of communication among agents produced by a user's request.

Speech Recognition (SR) Agent: The SR agent provides a mapping from the Interagent Communication Language to the API for the Corona speech recognition system, a continuous speech speaker independent recognizer based on Hidden Markov Model technology. This macro agent is also responsible for supervising a child micro agent whose task is to control the speech data stream. The SR agent can provide feedback to an interface agent as to the current status and progress of the micro agent (e.g. "listening", "end of speech detected", etc.) This agent is written in C.

Natural Language (NL) Parser Agent: translates English expressions into the Interagent Communication Language (ICL). For a more complete description of the ICL, see [2]. The NL agent we selected for our application is the simplest of those integrated into the OAA. It is written in Prolog using Definite Clause Grammars, and supports a distributed vocabulary; each agent dynamically adds word definitions as it connects to the network. A current project is underway to integrate the Gemini natural language system, a robust bottom up parser and semantic interpreter specifically designed for use in Spo-

ken Language Understanding projects.

Database Agents: Database agents can reside at local or remote locations and can be grouped hierarchically according to content. Micro agents can be connected to database agents to monitor relevant positions or events in real time. In our travel planning application, database agents provide maps for each city, as well as icons, vocabulary and information about available hotels, restaurants, movies, theaters, municipal buildings and tourist attractions. Three types of databases were used: Prolog databases, X.500 hierarchical databases, and data loaded automatically by scanning HTML pages from the World Wide Web. In one instance, a local newspaper provides weekly updates to its Mosaic-accessible list of current movie times and reviews, as well as adding several new restaurant reviews to a growing collection; this information is extracted by an HTML reading database agent and made accessible to the agent architecture. Descriptions and addresses of new restaurants are presented to the user on request and the user can choose to add them to the permanent database by specifying positional coordinates on the map ("add this new restaurant here"), information lacking in the HTML database.

Domain Agent: This agent encodes knowledge as to what actions must be performed to resolve each possible type of ICL request in its particular domain. For a given ICL logical form, the domain agent can verify argument types, supply default values, and resolve argument references. Some argument references are descriptive ("How far is it to the hotel on Emerson Street?"); in this case, a domain agent will try to resolve the definite reference by sending database agent requests. Other references, particularly when contextual or deictic, are resolved by the user interface agent ("What are the rates for this hotel"). Once arguments to a query have been resolved, the domain agent coordinates the actions and calculations necessary to produce the result of the request.

Interface Agent: This macro agent is responsible for managing what is currently being displayed to the user, and for accepting the user's multimodal input. The Interface Agent also coordinates client modality agents and resolves ambiguities among them : handwriting and gestures are interpreted locally by micro agents and combined with results from the speech recognition agent, running on a remote speech server. The handwriting micro-agent interfaces with the Microsoft PenWindows API and accesses a handwriting recognizer by CIC Corporation. The gesture micro- agent accesses recognition algorithms developed for TAPAGE.

An important task for the interface agent is to record which objects of each type are currently salient, in order to resolve contextual references such as "the hotel" or "where I was before." Deictic references are resolved by gestural or direct manipulation commands. If no such indication is currently specified, the user interface agent waits long enough to give the user an opportunity to supply the value, and then prompts the user for it.

We shall now give an example of the distributed interaction of agents for a specific query. In the following example, all communication among agents passes transparently through a facilitator agent in an undirected fashion; this process is left out of the description for brevity.

1. A user speaks: "How far is the restaurant from thi hotel?"
2. The speech recognition agent monitors the status and results from its micro agent, sending feedback received by the user interface agent. When the string is recognized, a translation is requested.
3. The English request is received by the NL agent and translated into ICL form.
4. The appropriate domain agent receives the ICL distance request containing one definite and one deictic reference and asks for resolution of these references.
5. The interface agent uses contextual structures to find what "the restaurant" refers to, and waits for the user to make a gesture indicating "the hotel", issuing prompts if necessary.
6. When the references have been resolved, the domain agent sends database requests asking for the coordinates of the items in question. It then calculates the distance according to the scale of the currently displayed map, and requests the user interface to produce output displaying the result of the calculation.

5. RELATED WORK

This section will be completed in the final version of the paper. It will include comparisons and contrasts with the PAC [6], KQML, KIF and the Knowledge Sharing Initiative work [4,5], the MASCOS multiagent system [8] and several commercial map packages.

6. CONCLUSIONS

By augmenting an existing agent-based architecture with concepts necessary for synergistic multimodal input, we were able to rapidly develop a map-based application for a travel planning task. The resulting application has met our initial requirements:

a mobile, synergistic pen/voice interface providing good natural language access to heterogeneous distributed knowledge sources. The approach used was general and should provide a good basis for developing synergistic multimodal applications for other domains.

In the near future, we will continue to verify and extend our approach by building other multimodal applications. We are interested in generalizing the methodology even further; work has already begun on an agent-building tool which will simplify and automate many of the details of developing new agents and domains.

REFERENCES

1. BELLIK, Y. and TEIL, D. "Les types de multimodalites", In Proc. IIM'92 (Paris), pp. 22-28.
2. COHEN, P.R., CHEYER, A., WANG, M. and BAEG, S.C. "An Open Agent Architecture". In Proc. AAAI'94 - SA (Stanford), pp. 1-8.
3. FAURE, C. and JULIA, L. "An Agent-Based Architecture for a Multimodal Interface". In Proc. AAAI'94 - IM4S (Stanford), pp. 82-86.
4. GENESERETH, M. and SINGH, N.P. "A knowledge sharing approach to software interoperation". Computer Science Department, Stanford University, unpublished ms., 1994.
5. McGUIRE, J.G. and al. "SHADE: Technology for knowledge-based collaborative engineering". Journal of Concurrent Engineering: Applications and research, Vol 1, No 2, 1993.
6. NIGAY, L. and COUTAZ, J. "A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion". In Proc. InterCHI'93 (Amsterdam), ACM Press, pp. 172-178.
7. OVIATT, S. "Toward Empirically-Based Design of Multimodal Dialogue Systems". In Proc. AAAI'94 - IM4S (Stanford), pp. 30-36.
8. PARK, S.K., Choi J.M., Myeong-Wuk J., Lee G.L., and Lim Y.H. "MASCOS : A Multi-Agent System as the Computer Secretary". Submitted for publication.
9. PFAFF, G. and TEN HAGEN, P.J.W. *Seeheim workshop on User Interface Management Systems* (Berlin), Springer- Verlag.
10. SCHWARTZ, D.G. "Cooperating heterogeneous systems: A blackboard-based meta approach". Technical Report 93-112, Center for Automation and Intelligent Systems Research, Case Western Reserve University, Cleveland Ohio, April 1993. Unpublished PhD. thesis.