ADVANCED INTERFACES FOR A MULTI-OPTION REAL-TIME RIDESHARE SYSTEM

Prasuna DVG Reddy
University of California at Davis
Davis, CA

Paul P. Jovanis
University of California at Davis
Davis, CA

Ryuichi Kitamura
Kyoto University
Kyoto, Japan

ABSTRACT

Ridesharing has long been perceived as a commuting mode that requires planning and lacks the flexibility traditionally associated with commuting alternatives, such as solo driving. An efficient rideshare matching system is essential to promote ridesharing as an effective commuting alternative. This paper describes ways to increase the performance of the rideshare matching system to match not only for sequential origin-destination (O-D) and work start-time, but also pick-up and drop-off of riders using a geographic information system (GIS) platform. All trips are linked with other trip attributes that will help as part of the information database. The prototype demonstrates image-integration technology by including a user picture as a security measure. The next part of the paper talks about interfacing a voice-recognition algorithm to the ridesharing system. The last part of this paper explains how an operator can recognize a handwritten input from the user with an operator.

INTRODUCTION

For years ridesharing has been seen as a commuting mode that entails a huge database, fast matching algorithm, and a good user interface. The success of rideshare matching depends on flexibility, information content and interfaces to the system. An efficient rideshare matching system is essential to promote ridesharing as an effective commuting alternative.[1] Potential carpoolers are first required to provide information, such as names, phone numbers, home and work locations, and work start times. The rideshare agency matches the information with other participants registered in the database and identifies potential carpool partners. A match list of potential carpool partners, along with their names and telephone numbers, is mailed to the participant. Later the individual contacts people on the list to form a carpool.

A real-time rideshare matching system could offer many benefits. It will allow travelers to review rideshare options, identify best matching riders, reserve rides in advance, register details for an immediate travel request, and help in the formation of carpools at park-and-ride lots while taking real-time demand into account. "Real-time matching" is defined as a one-time rideshare match obtained for a trip either the same day or the evening before. More importantly, a real-time matching system will make ridesharing possible for commuters who do not have a regular commute schedule in terms of departure time and destination location. Further, the most important breakthrough that advanced traveler information system (ATIS) will bring to ridesharing is that matching information will ultimately be available in the vehicle.

Therefore, the availability of a real-time rideshare matching system could significantly increase the market potential of ridesharing per se, since the system would provide more options, and thereby increase rideshare opportunities.

The results of one study suggested that a comprehensive ride matching system should include the following components:[2]

1. Means for storing information on commute patterns
2. Means for matching commuters' information effectively so that more successful carpools will be formed
3. Means for informing commuters
4. Means for updating and validating information to ensure accuracy
5. Means for easy access to the system in the process of information acquiring.

In the RIDES 1990 survey, carpoolers were asked to think of ways RIDES could improve its service. Not surprisingly, the carpoolers wanted to see RIDES make the information more accurate and up-to-date, and provide longer match lists. When asked which people the ridesharers would call from the match list, most applicants selected those people with similar work hours or employment locations. Twelve percent of the survey respondents claimed that they did not try to call anyone on the list because they found no one matched their hours. Obviously, a well-designed rideshare matching system could play a significant role in determining the success of the program.[3]

Dissemination of rideshare information and other communication among carpool partners have been done almost exclusively via mail and telephone. The efficiency and attractiveness of ridesharing are greatly reduced because of the significant amount of time (usually four to five days) required for mailing match lists and contacting carpool partners. This problem could be ameliorated by implementing a real-time rideshare matching system. In such a system, ridesharers' information processing and matching procedures would be performed by computer, and matches provided to ridesharers either over the telephone, computer modem, or information kiosks. Carpools may thus be formed in much shorter times using advanced communication facilities such as fax machines, pen computers, beepers and cellular telephones instead of mailing out match lists.[2] It has been found that computerized rideshare matching systems have higher placement rates than manual ones. If one plans to implement the kind of advanced facilities just mentioned, there is a need for developing interfacing algorithms between systems that avoid manual interference. Fax machines and pen computing systems need character-recognition algorithms. Telephone interfaces need voice-recognition algorithms.

Setting the Context

Present systems are operator dependent in order to get a matching list. Users need to specify their origin, destination, and travel time to the operator through telephone or fax. The operator will enter the data into the system and generate a matching list. This matching list is delivered to the user. To make a real-time ridesharing system really real-time (on line help), advanced interfaces need to be introduced. Under this class of interfaces, the authors have developed algorithms for voice and handwriting recognitions.

These advanced interfaces are easy to use and are very compact. Handwriting interface systems will be very popular when pen-computing systems are widely available on the market. Using pen-computing technology, the user can send a request to the system by writing on the screen. In this paper, emphasis is on numeral recognition because the authors' real-time rideshare system takes only numeric input (e.g., zone numbers, travel time, etc.). Each of these interfaces has its own advantages and disadvantages. Voice-recognition technology helps in recognizing a user's commands through the telephone or a built-in speaker.

Although real-time rideshare matching is not currently operational in the United States, field tests of such programs are planned in Texas, California and Washington State. Computer software is being developed to automate ride-matching services and make them accessible via the telephone or personal computer. The "Smart Commuter" project being carried out in Houston, Texas, will test a comprehensive employer-based matching service. It will include the ability to provide real-time carpool matches by providing GIS-based information and available bus transit services to employees. Caltrans is also conducting similar projects -- the "Smart Traveler" project in Los Angeles and the field operational test in Sacramento for real-time rideshare matching.[4] So, the time is right for investigating the different interfaces that can attract more users to rideshare systems.

This paper is divided into three parts. Part I deals with the multioptional real-time rideshare matching algorithm and geographic information system interface.

Part II describes the voice-recognition interface module design for both single (user-dependent) and multi (user-independent) user systems. Part III deals with user-written character recognition (handwriting recognition) module design.

PART I

This part of the paper investigates ways in which flexibility of the real-time rideshare matching system can be increased to provide matches not only for origin-destination and work-start time matching, but also enroute pick-up and drop-off of riders. A GIS-based system (using ARC/INFO 6.0.1) linked up with a ridesharing data base is proposed for implementing such rideshare-matching services.

This system will involve significant improvements over existing matching procedures that require an exhaustive data base search each time a match is to be made. Most existing matching systems also do not allow enroute matching due to their lack of route information. Use of this prototype in testing how system attributes (e.g., data base size, personal preferences, and matching criteria such as inconvenience tolerances) may affect the successful operation of a real-time rideshare matching program is also illustrated.

Methodology

Currently, ridesharing agencies typically match potential ridesharers by origin and destination, using criteria such as work start times, extra distances drivers would be willing to tolerate picking up and dropping off riders, and flexibility in work timings. Two methods are used for matching ridesharers -- zone-based matching and zipcode matching. The zone-based matching involves allocating each ridesharer to a traffic analysis zone, while zipcode matching requires the person to be assigned a zipcode in the data base. Zipcode matching proves to be easier, since zipcodes can be easily assigned using the participant's street address.

The computer model should be flexible enough to accommodate short-distance commuters by matching their routes with at least part of the route used by long-distance commuters. Geo-coding commuters' routes allows this sort of matching to be performed. Matching criteria required for this purpose include work start times, amount of flexibility in work timings, and distances drivers are willing to go out of their way to pick up and drop off riders. Five alternative scenarios are described:

1. Driver and rider matched by their origins and destinations
2. Driver and rider have same origin and driver drops rider on his/her way
3. Driver will pick up rider at some point of his/her commute
4. Driver will pick up and drop off the rider in his/her commute
5. Deviation from the driver's regular path.

These scenarios are illustrated in Figure 1, created as a set of layers by the GIS system, to be searched in sequence if suitable matches are not found in earlier layers. The above scenarios represent simplifications of situations that may arise in implementing flexible matches in a real-time ridesharing system. It is also possible to incorporate additional criteria (like walking distances to/from pick-up or drop-off points, and wait times) using the GIS system to calculate travel times from the spacial distribution of origins, destinations, and pick-up/drop-off points.

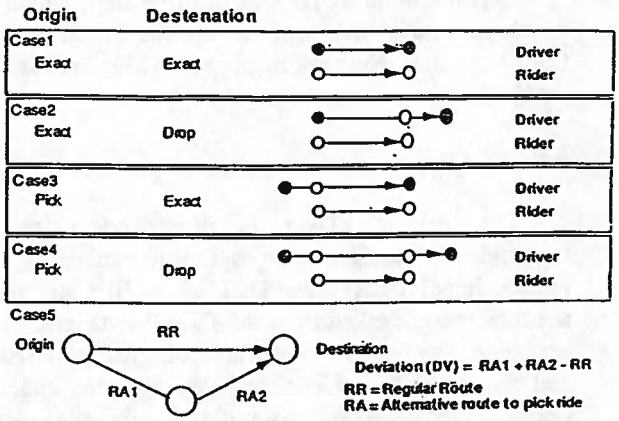

Figure 1. Alternative scenarios in rideshare matching

Another layer in this GIS based real-time rideshare matching system includes ridesharer attribute information. This information consists of the persons' picture, personal preferences (like smoking/non-smoking carpool partners), gender, and special messages, all included in the data base using ARC/INFO's Image Integration module. These types of information are provided by participants on a voluntary basis, together with other required data, like origin and destination, work start times, routes, work time flexibility, and inconvenience tolerances of distance and time.[5,6]

ARC/INFO is the front-end software to the user, with a data base containing the street network and traffic flows. The data base also contains geo-coded locations for address matching. ARC/INFO modules Network,

Image Integration and Formedit are used for this application. A more detailed discussion of the network-building procedure used in developing the real-time ride-share matching prototype is provided in the following section.

The search procedure being implemented for carpool matches is done in two stages. In the first stage, a search is conducted of the geo-coded origins and destinations to match a rideshare request with possible riders or drivers of similar origin and destination with the constraints of work start times and work time flexibility. The second stage will involve users viewing ridesharer attribute information, like photographs and personal preferences, to make a final match. In this stage, the user is allowed to select a match according to his/her taste. This would be useful for conducting simulation experiments on the amount of information required, gender effects, and acceptable criteria of time and distance inconvenience.

Network Building

Network building consists of three steps which will develop a network map with all required data layers. The three steps are: Network map, geo coding, and network feeding.

Network Map

The network layer is developed using the topologically integrated geographic encoding and referencing (TIGER) files. TIGER/Line files are the line network product taken from the TIGER data base. They are based on a vector model, are topologically consistent, and contain both locational and attribute data, including line segments, census geographic codes, and address ranges for the left and right nodes of each segment.

Geo Coding

Geo coding is a process that identifies locations on maps using addresses. Once addresses are located, the data associated with the addresses can be related to other coverage features and used for a variety of applications. With geo coding, one can get information about the best route from one address to another.

Network Feeding

The basic network is fed with different routes contained in the rideshare data base. Each link in the network will have a series of artificial links that are parts of the carpoolers' trip. Each part is called a segment. Attribute data for each segment are stored as a data structure. The data structure is similar for all segments.

All real network link data are stored in the "main" class, and artificial link data are stored in "instances" of the main class. Instances all have the same data structure, but with different values. All instances consist of data, methods, and connectivity. The data may be origin, destination, mode of transportation, starting time, capacity of car, number of passengers etc. The method is designed to check the matching characteristics between rider and driver. Connectivity is a double-linked list, in which the front pointer will have the address of forward-link instance and the end pointer will have the before-link instance address.

In the beginning of the network feeding process, all the segments are fed with proper data. Here the data base is constructed in an object-oriented fashion and not as a relational data base like existing rideshare data bases. Data can be modified and new information input into the data base at any time. Once the system finds a match for a particular trip, all the instances of that trip are eliminated. The advantage of using instances is that a new instance can be defined by inheriting data from the parent class. For more details, see Reddy et al.[7]

User Interface

The user interface is being developed on the X windowing system using Motif windows on a DEC 5000 work station, comprising a graphical interface that takes information from the rider and makes matches using the internal data base. This interface will have three windows. The main window will contain the main menu, a bar menu, and an icon manager. In this window, the user will enter his/her information. The second window will display the city street network. This window will have zooming and panning features. The last window displays all generated matches. The user interface is being programmed in Arc Micro Language (AML) of ARC/INFO. The data transfer between the matching algorithms and the GIS system are done by a preprocessor written in C++.

Features of the GIS-Based Rideshare Matching Model

The GIS-based model being developed has the following features:

1. It is a simple interface. Anyone who knows how to use a map can use this interface.

2. Users have more information about the potential rider, such as their gender, trip time, distance to walk, trip purpose, personal habits, etc.

3. Having an image of the rideshare person will solve some of the security problems and will lead to easy identification.

4. The model can function as a simulation model to investigate the critical number of participants needed to provide a suitable match, based on a matching criteria and a given spacial distribution of ridesharers.

5. The model can investigate the effect on the critical mass of participants as matching criteria are varied. These criteria include the time and distance inconvenience that is required for a participant.

PART II

The interface described above helps the operator to find a match. These GIS-based systems are suitable for public places and operator assisted systems. These systems, however, are not directly accessible to the users. This part of the paper describes work performed at the University of California at Davis on a voice operated information system (VOIS) by which users have access to a system using speech commands. The principal aims of this work are to develop a user-friendly interface for the user and to investigate the degree to which dialogue control can be used to compensate for deficiencies in existing interfaces.

User interfaces have experienced radical changes in recent years. Widespread use of bitmapped displays and the mouse have resulted in significant increase in the presentation of graphic information as well as extensive development of interfaces that require users to do practically no keyboard input. None of the existing systems, however, has addressed the basic problem of how to raise the level of interaction to a dialogue. The current state of interactive interfaces with respect to the user's role will be briefly characterized below. A thought-provoking and much more detailed, examination is found in Norman and Draper.[8]

Terveen [9] observed that a model of human conversation may be used to influence the structure of interfaces. Kant[10] also observed that the interfaces are dependent on the type of application, type of user, and locational characteristics. Sometimes, high-end systems are not suitable for all types of users. A survey of commercially available speech recognition systems has been conducted by General Motors,[11] and gives the different speech-recognition systems and their attributes.

System Overview

The VOIS system architecture is shown in Figure 2.

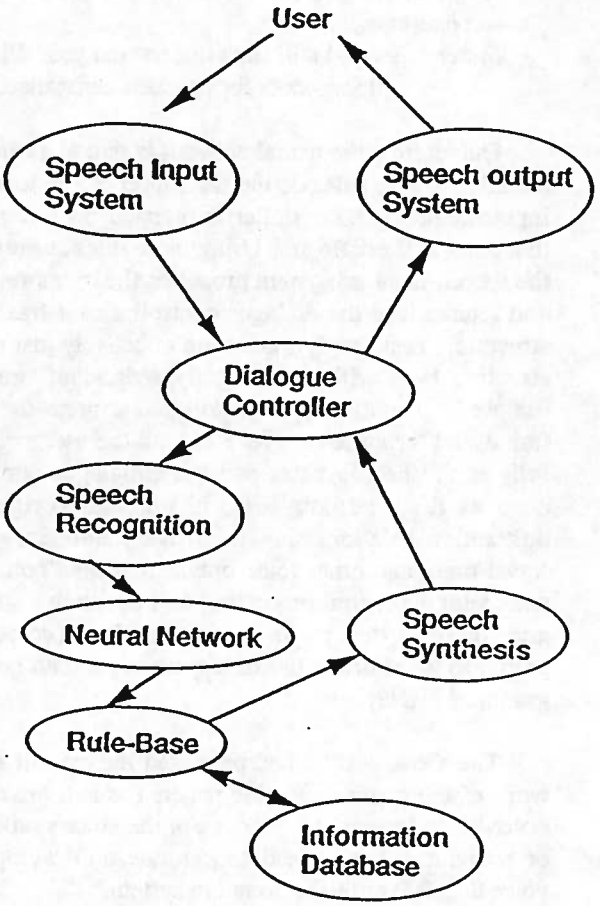

Figure 2. VOIS system design

As shown, the dialogue controller is an independent unit that interfaces with other system components. A typical dialogue transaction cycle operates as follows. The dialogue controller outputs a question to the speech-output subsystem, and simultaneously outputs a set of rules to the speech-input system. From the dialogue controller, the input is transferred to the speech-recognition module. The speech-recognition module uses a trained neural network and tries to recognize the input. If the neural network is not able to identify the input from the user, then an alternative system will be activated. For example, a worst-case dialogue may proceed as follows:

System: Where do you want to go to?
User: I have to get to Sacramento this afternoon.
System: Sorry, where to?
User: To Sacramento.
System: Sorry, I still can't understand you. Please state destination. One word only please.
User: Sacramento.

(Normally at this stage the system will understand the input. But in case it does not, the system will advise use of an alternative system. In this case, the system advises calling an operator.)

System: Sorry, I still can't understand you. Please call xxx-xxxx for operator assistance.

Output from the neural network is sent to a rule-base module. There, rules define the subset of the total user input the dialogue controller is prepared to interpret at that point in the dialogue. Using these rules as guidance, the speech-input subsystem processes the user's response and returns it to the dialogue controller as a frame-like structure. These reply frames are effectively just phrase structure trees with semantically redundant branches removed. The dialogue controller interprets the reply frame, and the cycle then repeats until the user's query is fully established. At this point, the dialogue controller accesses the local data base, in this case, origin and destination locations that the driver commonly uses, travel times and other route options. It then communicates with the central processing unit to obtain a solution and outputs the required information (voice and graphics) to the driver by voice synthesis and an optional graphical display.

The above outline had presented the overall framework of the system. In this paper, the authors do not describe the internal architecture of the voice synthesizer or recognizer, but instead concentrate on developing a voice interface to the information system.

Design Methodology

Frame

A frame consists of data and has two key properties (Figure 3). First, the fields of a frame are invoked automatically when the frame's data are needed by the system, or when data is added by the system or the user. Thus, a frame allows a dialogue system to be constructed in which the desired outcome of filling in all the required data slots is achieved without having to specify a precise order of dialogue events.

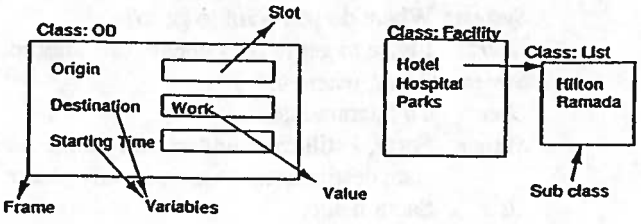

Figure 3. Textual representation of audio frame

Secondly, the frame has an associated inheritance mechanism (Figure 4). A new frame may be defined as a specialization of a more general frame. In the new frame, only those properties that need to be changed or added are specified and all the remaining properties are inherited from the existing frame. This inheritance mechanism gives direct support to the requirements for dialogue adaptation. Adaptation implies that a previous event, or one in a different context, may need modification for the present situation. In the case of dialogues, the event could refer to such cases as a fixed origin and changing destination. For example, in daily commute trips, only the departure time may change (Figure 4) and all other items remain the same as the previous day.

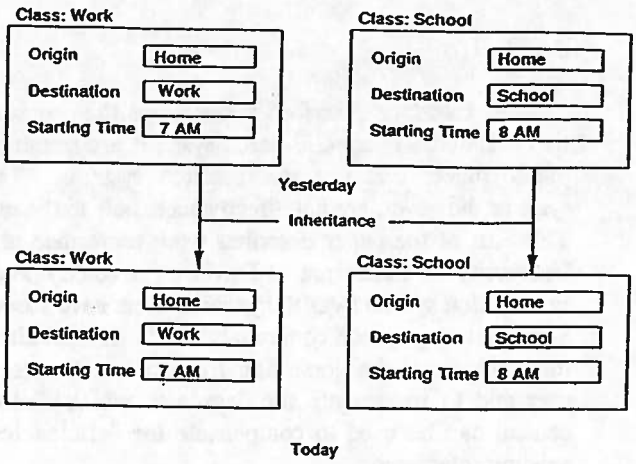

Figure 4. Inherited mechanism of audio frame

Speech Recognition

The speech-recognition subsystem performs a frequency and amplitude analysis of spoken signals and compares them with stored representations of words and phrases that are recognizable by the system. The process of speech recognition consists of several steps. The first is feature extraction. It represents the time-amplitude-frequency characteristics of the signal digitally, reducing data flow and analysis to manageable proportions. The next step is comparison of the input signal with the stored reference patterns. This requires time synchronization of input signal and stored patterns, one of the biggest problems in speech recognition. Usually alignment methods establish the beginning and end of vocabulary entries using an energy criterion: The beginning is signaled by the onset of significant speech energy, and termination by the speech energy dropping below some threshold. The input pattern is now ready for comparison. For each vocabulary entry, a measure of similarity is computed. The reference word that yields the highest similarity rating is the "recognized" word, except that the speech-recognition subsystem will apply

a rejection criterion (less than a threshold value) to prevent incorrect recognitions. If this criterion is not achieved, the input will be rejected.

Synthetic neural networks are an attractive solution to the above problem, since they can learn to perform classification from labeled training data and do not require knowledge of statistical models. This research also investigates the feasibility of using synthetic neural networks for identification of a human voice.

Two approaches are followed in using this system. The first is called VOIS-D and is applied to a single user (speaker dependent system). It may be installed in his/her car or at home. The second approach is called VOIS-I and is applied to multiple users (speaker-independent systems). It is installed in public places like airports and roadside kiosks.

The neural network used in the both systems consists of an input layer (Figure 5), a hidden layer, and an output layer. There are 500 input nodes with each node representing speech energy at each time interval. The hidden layer has simulated nodes for from 1/10th of the total nodes to the total number of nodes. Presently there are 31 output nodes (total vocabulary) in VOIS-D. Each node represents a word. This network takes a large amount of time to be trained.

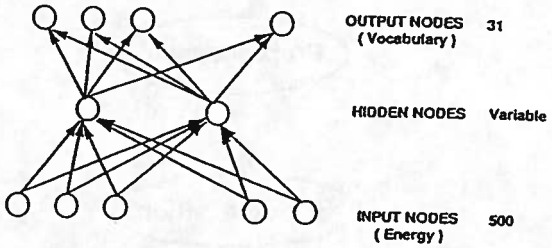

Figure 5. Neural network design

Alternatively, the same network is divided into smaller pieces and designed into a step-by-step process in VOIS-I. In this case, the driver can hear the output of the system and choose one of them. For example, if the VOIS-I system lists hotels, parks, and bus stops in the step-by-step process, the driver has to choose one of these options. In the case of VOIS-D, the system driver can say any input, like "Hotel Hilton" at any stage of acquiring information.

In recent testing with the prototype, it was found that step-by-step networks showed very good results because the number of words to be recognized by each network is fewer compared with those using a single network. The other advantage in the step-by-step network is that if the system needed to be extended (increase the number of words in the data base), one has only to train a small network. In a single network, one would have to retrain the whole network. During the training of the neural network, the output nodes are set to "1" if the node is chosen and "0" otherwise. During the testing (or prediction) phase, the node that is chosen has the greatest value among all output nodes. The highest value from the output node should have a minimum of 0.5. The learning model used to train the network is the back-propagation method. Once the neural network recognizes a word, it will be sent as an input to the rule base.

Rule Base

Once the speech-recognition subsystem sorts out the word spoken by the driver, the next step is to process the word. This task is performed by a rule-base module. The module contains a large set of rules, called a knowledge base, which checks what action has to be taken depending on input received from the neural network. The final system will have multiple inputs from different neural networks (step-by-step) all are stored in a frame. Each frame is the input to a rule base that has full information about requests made by the driver.

Voice Synthesizer

This module is simple compared with voice-recognition modules. It reads the frame/text input from the rule base and, using a synthesized voice, conveys the message to the driver. This module has another function than just delivering speech; that is the repeat voice output. This function enables the driver to replay the most recently spoken voice message.

Display Unit

This unit has a minimal graphic utility from displaying some types of graphics. This helps the driver determine his/her present location and intended destination. Sometimes, if a driver does not understand the synthesis message, the display terminal will help in clarifying information.

Preliminary Observations

The following observations were noted when the existing prototype developed was tested by different people:

- Vocal commands of drivers are often incomplete. Omissions do not follow normal grammatical or linguistic rules and seem to reflect

the user's assumptions about the internal structure of the system. Omissions can occur at any point of the command structure. For example, instead of starting, "Hotel Hilton," the user will frequently state the abbreviated command, "Hilton," or "Hilton Hotel." Users often skip words or change the order of words.

- Analog processes, like searching for a hotel or opening the next information, are always broken down into discrete steps. From a theoretical point of view one would expect drivers to use commands like: "All hotels, next, next, next," or, "Hotels and name." Instead, they use the command, "All hotels," and expect the system to list all the hotels.

- Surrounding noise will sometimes disturb the word recognition.

- People have initial problems understanding the use of the system, but as time goes on, they become more acquainted with its operation.

PART III

During the past few decades, there has been a growing interest among scientists in inventing systems that can imitate the functionality of a human in various aspects of his/her activities. Handwriting recognition is one of the oldest and most challenging among them. Intensive research has been done in the area and over time the field has attracted increasing amounts of talent, as evidenced by the growing number of papers and technical reports in the area. Automated postal-zip-code reading, bank-check-amount verification, office automation and computerized storage of old handwritten material are just a few possible applications of this field.

Various approaches to handwritten-digit recognition is being investigated here with the aim of being able to recognize a handwritten request for a matching list received from a fax machine or pen-computing system. Discussed below is the architecture for one of the more promising approaches along with some of the initial experimental studies on the system. The system is being further enhanced and additional results will be reported in future papers.

Architecture

The architecture for the prototype handwritten-numerals-recognition system has evolved from the system proposed in 1991 by Wang and Gupta[12] with several major differences and enhancements. One of the important differences is that the present system is for off-line recognition, while the system discussed in Wang and Gupta is for on-line situations.

The architecture for the system is shown in Figure 6. The system takes off-line numeric input from the user about origin and destination zipcode and about travel time. Then, the whole input is segmented into single characters. All these characters will be in a bitmap format. Each pixel in the bitmap can posses a value of either "1" or "0." These bitmaps are usually scanned from the input documentary by a digital scanner and passed through a segmentation algorithm to get binary bitmaps for segmented digits.

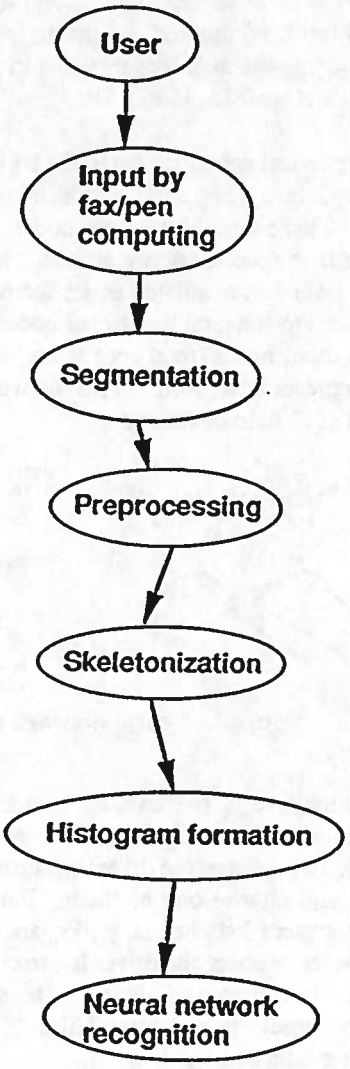

Figure 6. Architecture of handwriting recognitition

Preprocessing

The standard preprocessing steps involved in pattern-recognition systems have been incorporated in

this stage. Filling of isolated holes and small notches and linkages of small brakes or broken lines are achieved through the use of smoothing procedures involving de-magnification followed by re-magnification.

The bitmap is de-magnified to a suitably small size and then re-magnified to the required size. The de-magnification routine moves a window of size k1 x k2 across the binary bitmap and uses some rules to obtain a single binary value for the area spanned by the window. Re-magnification involves a similar process. The algorithm relies on the observation that, in the recognition of numerals, the most important and distinguishing features are the gross features. More discussion of this method is documented in Nagendraprasad.[13]

Skeletonization

Skeletonization is the process by which the input patterns are transformed to thin line drawings, ideally one pixel in thickness. A good skeleton retains the connectivity and structural features of the original pattern. Rideshare participants use different pens to write their requests to the system by fax. The skeletonization process standardizes the thickness of the digit. Figure 7 shows sample bitmaping of numerals and their corresponding skeletonized bitmaps. The accuracy of this method is dependent on the skeletonization algorithm. In application the authors used the algorithm presented in Nagendraprasad et al.[14]

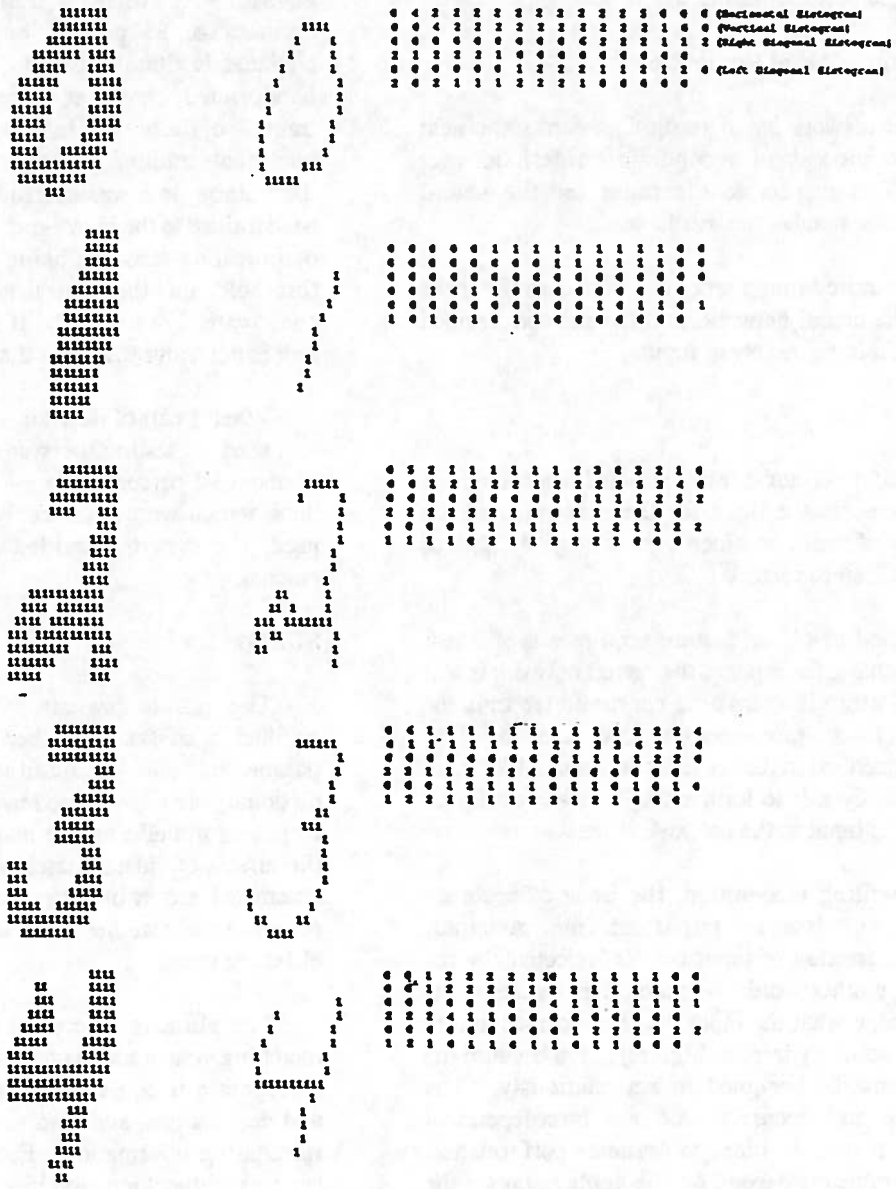

Figure 7. Bitmaps, skeletonized versions, and histograms of numerals

Histogram Formation

A histogram is a vector extracted from the bitmap using simple operations. It consists of a vector of numbers in which each number is obtained as the summation of the number of pixels along a particular direction. The set of histograms for a digit consists of four vectors: Vertical histogram, horizontal histogram, right-diagonal histogram, and left-diagonal histogram. Thus, the i^{th} element in the vertical histogram is the sum of the pixels of value "1" in i^{th} column of the bitmap. The other histograms are similarly formed. Figure 7 shows the histogram for a sample set of skeletonized digits. Though the concepts of histograms may appear simplistic, a neural network can be trained to recognize them with high accuracy as discussed next.

Neural Network-Based Recognizer

A neural network-based recognizer forms the next stage in the process of recognizing rideshare user requests. With respect to the input for the neural network, two approaches are available:

1. The entire bitmap array can be fed as the input to the neural network, and the network trained over a large variety of inputs.

   or

2. A feature vector can be extracted from the input to serve as the input for the network, and the network can be trained over a large number of such feature vectors.

The method involving feature vectors was adopted. Under this method, the input to the neural network is less complex, and hence it is less time-consuming to train the network. The feature vectors used are the four histograms described in the previous section. All of them are placed side-by-side to form a single large vector that was used as the input to the network.

In handwriting recognition, the issue of accuracy versus rejection plays an important role. Accuracy represents the fraction of input that is "rejected" by the recognizer. In other words the recognizer expresses its inability to judge what the input is. In most recognition systems high accuracy implies high rejection because the recognizer must be designed to act cautiously. The rejection rate and accuracy rate are interdependent parameters that must be tuned to optimize performance. These two parameters depend on the application for the which the system is being built. In the present case, it is the rideshare system.

Implementation and Results

The system is first trained on a local data base to build a dictionary and to tune the neural network. The input is preprocessed and skeletonized. The skeletonized input is then fed to the histogram algorithm, which extracts a histogram vector from its input and feeds it to the neural network. If the network rejects a particular vector, then the corresponding skeletonized pattern is fed to the dictionary-matching algorithm.[15]

The algorithm was implemented in C on a PC. The neural-network architecture consisted of 94 units in the input layer, 15 units in the hidden layer, and 10 units in the output layer. The standard back-propagation algorithm was used for training. It was trained to an accuracy of 93 percent on 3,000 patterns from the National Institute of Standards and Technology (NIST) handprinted character data base. While excessive training of the net can lead to poor generalization ability, inadequate training can result in poor accuracy. As such, the training rate was carefully optimized. The net has been trained to the above-mentioned accuracy and further optimization tests are being conducted. The rejection threshold for the neural-network output units was conservatively set at 0.7. If the activation level for the unit is not above 0.7, then that output unit is rejected.

When a part of the training data (about 200 patterns) was used for testing the system, it provided an accuracy of about 92 percent. When a set of 200 patterns of test data, which were distinct from the training data, was used, the system provided an accuracy of about 88 percent.

SUMMARY

The results presented here are based on tests conducted so far. Further optimization of network parameters and availability of a more elaborate dictionary are expected to lead to higher accuracies. The emphasis in building the above systems is on studying the effects of riders' matching techniques and also the automated access of data bases. Insights gained in the course of these studies are being utilized to create a more elaborate system.

The ultimate objectives of the real-time rideshare matching system are to increase the system's flexibility to incorporate irregular work schedules, varying origins and destinations, and also to speed up dissemination of ridesharing information. Futuristic rideshare matching systems will include in-vehicle navigation and communication technology to enable enroute notification of possible ridesharing opportunities.

REFERENCES

1. K.J. Dueker and B.O. Bair, "Ride Sharing: Psychological Factors," *ASCE Transportation Engineering Journal* (November 1977).

2. S.J. Beroldo, "Ridematching System Effectiveness: A Coast-to-Coast Perspective," *Transportation Research Record* No. 1321 (Washington, DC: TRB National Research Council, 1991): 7-12.

3. M. Thayer, "A Re-Examination of RIDES' Data Base," RIDES for Bay Area Commuters, Inc. (February 1991).

4. L.N. Labell, "Advanced Public Transportation Systems: The State of the Art, Update 1992," U.S. Department of Transportation (April 1992).

5. P.D. Reddy, P.W. Grant, R. Kitamura, and P.P. Jovanis, "Image Integration and Network Applications in Transportation," presented at the Spatial Data Analysis Conference, Davis, CA (August 6, 1993).

6. P.D. Reddy, P.W. Grant, R. Kitamura, and P.P. Jovanis, "Pre-Trip Planning Using Image Integration and Networking," published in *13th Annual ESRI User Conference* (Palm Springs, CA: May 1993).

7. P.D. Reddy, P.W. Grant, R. Kitamura, and P.P. Jovanis, "Multi-Optional Real-Time Rideshare Matching Using GIS," published in *14th Annual ESRI User Conference* (Palm Springs, CA: May 1994).

8. D.A. Norman and S.W. Draper (eds.), *User-Centered System Design: New Perspective on Human-Computer Interaction* (Hillsdale, NJ: 1986).

9. L. Terveen, "Resources for Person-Computer Collaboration," working notes from the *AAAI Spring Symposium on Knowledge-Based Human-Computer Communication* (1990): pp. 132-135.

10. E. Kant, "Interactive Problem Solving Using Task Configuration and Control," *IEEE Expert* Vol. 3, No. 4 (1986): 285-313.

11. M.S. Schaeffer, "Speech Recognition Technology Survey," GM Research Report No. 89-27-77/g7690-001 (1989).

12. P.S. Wang and A. Gupta, "An Improved Structural Technique for Automated Recognition of Handprinted Symbols," accepted for publication in *International Journal of Pattern Recognition and Artificial Intelligence* (1993).

13. M.V. Nagendraprasad, "Automated Recognition of Handwritten Numerals: Issues and Algorithms," Master thesis submitted to the Massachusetts Institute of Technology (1991).

14. M.V. Nagendraprasad, P.S. Wang, and A. Gupta, "An Improved Algorithm for Thinning Binary Digital Patterns," International Financial Services Research Center discussion paper (1991).

15. D.P. Reddy and A. Gupta, "A Hybrid Approach to Typewritten Numeral Recognition," International Financial Services Research Center discussion paper (1991).