

Multimodal Fusion

Submitted in partial fulfilment of the requirements for the

Master of Philosophy

in

Computer Speech, Text and Internet Technology

University of Cambridge
Department of Engineering

Hank Liao
Sidney Sussex College
Sidney Street
Cambridge, England CB2 3HU
H.Liao.01@cantab.net
July 24, 2002

Supervisors
M. Hickey, HP Labs Bristol
P. C. Woodland, University of Cambridge

Copyright © 2002 by Hank Liao

Abstract

Computing applications are becoming increasingly complex and pervasive as exemplified by ubiquitous microprocessors in everyday appliances and resulting feature explosion. Recent technologies like speech or gesture recognition can make such systems more natural, easy to use, and robust. Work by [Oviatt et al. 2000] and other groups have shown this to be the case. A challenge has been to fuse these different input modalities effectively to complement their natural strengths and use redundancies to improve robustness. This project looks to examine methods of combining information from different input modalities in multimodal dialogue systems.

A theoretically recogniser agnostic and domain independent multimodal framework is developed. The framework is implemented in this project with a commercial speech recogniser and a freeware gesture recogniser. Many improvements were made to the gesture recogniser mainly to handle spatial gestures. Grammar rules and a unification-style algorithm were applied to fuse the semantic frames from different modalities into candidate command frames for the application. By using a multimodal grammar and scoring, the best candidate frame is selected and sent to an application.

To validate and evaluate the framework, a multimodal tourist guide for Bristol was built. As implemented, it has zoom and pan commands to navigate the map, and queries about distance and paths between sites and locations of sites. The evaluation task was comprised of four high level goals that users were expected to carry out with minimal guidance. Despite the sparse instruction, users were enthusiastic to multimodal interaction. The unification algorithm used in conjunction with N-Best lists was a viable fusion method. It improved performance over the estimated lower bound and 1-Best case and allowed the system to improve recognition results by using only valid combinations of inputs. These observations were made though with a set of three users; this small sample size limits the extent to which conclusions can be made from the evaluation.

Acknowledgements

I want to sincerely thank the people who helped to make this an interesting project and this year in Cambridge most memorable. First all of all, I would like to express my gratitude to my thesis supervisor Marianne Hickey for providing excellent guidance, thoughts and patience throughout the development of this dissertation. Secondly, I'd to thank Phil Woodland for his time and comments. I'd also like to thank Ann Copestake and Mark Gales for their time over the course of the year and for their advice, enthusiasm in teaching, and effort to excel.

I'd like to thank those people out there who provide free knowledge and software online. The open software movement is a noble cause, and this project would not have been possible without public domain source code to leverage.

Lastly, I want to thank all the great people I've met in Cambridge for making this year a *brilliant* year. To the sheep herd, I gave a rousing, "baaaa" of approval to our crazy nights out off the court. To my friends at Harrington House One, thanks for the tea; I'll miss asking for the usual. To the other usual suspects around, you know who you are; they'll be many more hotpots wars and punting adventures to have. And to my special partner, thanks for all the support and encouragement through the days and late nights; I'll always remember the sweet memories of strawberries and champagne.

Statement

I hereby declare that this dissertation is not substantially the same as any that I have submitted or intend to submit for a degree or diploma or other qualification at any other University.

I further state that this research and its results are for the most part my own. Where I have adapted theories or results of others, I state this clearly in the text and the bibliography.

Table of Contents

1	Introduction.....	1
2	Background and Related Work	2
2.1	Automatic Speech Recognition.....	2
2.2	Automatic Pen Gesture Recognition	3
2.3	Multimodal Research.....	3
2.4	Multimodal Fusion	4
3	Multimodal System Architecture.....	8
3.1	System Description.....	8
3.2	Automatic Speech Recogniser	9
3.3	Gesture Recogniser.....	10
3.4	Multimodal Input Fuser	11
4	Multimodal System Implementation	15
4.1	Development Environment	15
4.2	System Components	16
4.3	Communication Between Components	16
4.4	SpeechSvr.....	18
4.5	GestureApp	19
4.6	InputSvr.....	23
5	Multimodal Tourist Application.....	28
5.1	Requirements.....	28
5.2	Design	29
5.3	Implementation.....	31
5.4	The Final System.....	34
6	Evaluation.....	35
6.1	Evaluation Protocol	35
6.2	Evaluation Results	35
7	Discussion	42
7.1	Framework Architecture	42
7.2	Comparison to Other Related Work	42
7.3	Limitations	43
8	Conclusions.....	44
9	Future Work and Research.....	45
	Appendix A – Evaluation Protocol.....	47
	Appendix B – Companion CD Contents.....	51
	References.....	52

1 Introduction

Computing applications are becoming increasingly complex and pervasive as exemplified by ubiquitous microprocessors in everyday appliances and resulting feature explosion. Recent technologies like speech or gesture recognition can make such systems more natural, easy to use, and robust. Work by [Oviatt et al. 2000] and other groups have shown this to be the case. A challenge has been to fuse these different input modalities effectively to complement their natural strengths and use redundancies to improve robustness. For example, speech and gesture are complementary and users employ each to their strengths – speech for commands and gestures for spatial/location referencing [Oviatt 1997]. Most research has found users prefer to interact multimodally and perform tasks more efficiently and quickly than in corresponding unimodal applications [Oviatt 1997][Vo 1998]. These results promise to make technology more intuitive and easier to use.

This project looks to examine methods of combining information from different input modalities in dialogue systems. First, a framework will be developed that is ideally reusable, relatively recogniser and domain independent and incorporates a multimodal fusion algorithm. Secondly, a suitable multimodal application is needed to test the framework. Such an application should benefit from a multimodal interface by requiring a fair sized command vocabulary and spatial operations. Lastly, the framework shall be evaluated through this application with test users.

2 Background and Related Work

This chapter provides context for the project through a brief summary of relevant multimodal research. Topics presented in the following sections include an overview of recognition in various modes such as automatic speech recognition and pen gesture recognition, multimodal research in general, and current techniques for fusing multimodal inputs.

2.1 Automatic Speech Recognition

Automatic speech recognition (ASR) is the process of mapping an audio signal to a sequence of words usually with the goal of deriving some meaning. Since the 1950's, incremental improvements have been made to advance technology from simple 10 digit recognition tasks, to large vocabulary continuous speech recognition on the desktop or through the telephone. State of the art systems can boast recognition accuracies of over 99% with short lists or yes/no tasks and over 95% on 14,000 stock quotes. This section discusses various aspects of speech recognition and identifies key ideas relevant to its use in multimodal applications.

ASR proceeds in two main steps: feature extraction and feature classification. Feature extraction entails parameterising a relatively rich 16-bit precision 8-16kHz audio signal of speech to a compact set of coefficients. For example, a front-end can for each 10ms frame of speech, use a window of 25 ms to calculate 25 mel filter bank coefficients and then further transform these into 12 cepstral coefficients plus one energy related coefficient. These coefficients and their first and second derivatives are then typically used in the subsequent pattern matching process.

This feature classification process seeks to find the most probable word sequence given the sequence of observation vectors of coefficients. This probability is a combination of the acoustic likelihood of the signal, and a linguistic *a posteriori* likelihood of the word sequence. The acoustic score is calculated from existing models. Usually words are broken up into sub-word units such as phones that are modelled with hidden Markov models (HMMs). A search is then conducted to find the best sequence of phone models to match the observation vector; in other words, the best word sequence given the acoustic signal. The linguistic score is from a model of the language and possible word sequences.

For a multimodal system, the speech recognition component will need to handle continuous speaker independent speech from a medium number of words for embedded devices, to very

large vocabulary in richer applications. Also, with the goal of making interfaces more natural and accessible, the recogniser will need to be robust to environmental noise and speech disfluencies common in spontaneous speech. Besides robustness, available computing power could be a limit to the recogniser in embedded devices, when the recogniser executes on the device as opposed to in the network.

2.2 Automatic Pen Gesture Recognition

Pen gesture recognition is the mapping of a stylus or pen strokes, on touch sensitive screens or pads to characters, symbolic commands or points. Gestures can be simple points, regions or more complicated figures composed of one or more strokes. Pen gesture input as a means to input characters and symbols in small devices such as the Palm and Visor personal digital assistants and their Palm operating system's Graffiti gesture input system. A natural extension to this is handwriting recognition, but as seen with the failure of the Apple Newton, accuracy is still poor. Research is being conducted to improve these results, as well as in 3-D gesture recognition where hand movements are tracked either through cameras or a 'data glove' to provide gesture input.

The recognition of gestures is a similar application of pattern recognition to ASR. Pen gestures can be classified as arbitrary in nature, symbolic in that they can be deduced from their meaning, or deictic in referencing an object or concept in the current context. Appropriate features must be extracted from the input signal where they are then matched to a set of models. Simple templates can be used, or more sophisticated features such as a combination of image, velocity, angle, direction, and number of strokes can be used to discriminate between gesture classes [Rubine 1991]. The resulting feature vector can then be classified using a variety of techniques such as neural networks, decision trees, or HMMs.

2.3 Multimodal Research

Research over the years in multimodal interactions began with incorporating deictic references in speech recognition applications with pointing gestures exemplified in Bolt's Put-That-There system [Bolt 1980].

[Oviatt et al. 1997] provide good insight on the nature of multimodal interactions by creating a mapping application with a variety of command types that can be issued through speech, gestures or multimodally:

- Spatial location commands, e.g. adding or moving, that are highly likely by a 2-1 factor to be expressed multimodally to specify location, size, number, shape, a point or line or area.
- Selection commands, e.g. zoom or deletion, are only occasionally executed multimodally since users prefer to say unique speech descriptors or rely on the map or dialog context, rather than using a gesture to select.
- General action commands, e.g. print or view overlay, which do not involve spatial, location or selection information, were extremely unlikely to be executed multimodally at 1% of the time.

This kind of research has lead academia to look towards more generic frameworks to handle multimodal interactions beyond simple deictic resolution.

2.4 Multimodal Fusion

Several groups have developed techniques to combine multimodal inputs at various levels with different methods. This current work is surveyed in this section.

2.4.1 Level of Input Fusion

The input signals from various modalities can be combined at several different levels in the recognition process trading more fusion and complexity, with less interaction but more simplicity. Inputs can be fused at:

The feature level. This is the lowest level of multimodal fusion where speech and gesture frames are mapped to a sequence of tokens. This is likely to be rather domain dependent, and makes it difficult to leverage established speech or gesture recognition systems as it entails designing a new modelling and decoding algorithm to fuse the different mode frames together. As [Wu et al. 1999] point out, the fusion of fairly different modes, where inputs are sampled at different time scales, can be a significant obstacle for the modelling. For example, an HMM that depends on generating observations for each mode at a state would have difficulties with speech phenomena that can be an order of magnitude less in duration than gestures. Little research has been conducted at this level of fusion, probably due to difficulty and lack of corpora, although it promises the best overall results through the tightest fusion of modalities.

The partial results or token level. In this intermediate level of fusion recognisers provide mappings of input signals to convenient modality dependent tokens for semantic interpretation. These tokens can be words, partial word sequences, or complete sentences or in the gesture case, single stroke recognition results, or complex gesture results from a combination of strokes. A multimodal grammar can be constructed to parse the multimodal token streams. At this level of fusion, the partial results of one mode could influence the results of another.

The partial interpretation level. In this highest level of fusion, recognition takes place unimodally, possibly leveraging other recognition toolkits, likely with some post processing of the token strings through semantic interpretation. A higher-level multimodal component fuses these interpretations into a single multimodal interpretation. This is the approach taken in this dissertation.

Hybrid multi-level fusion. [Vo 1998] discusses another option of combining different input streams by allowing the different modes to interact at the decoding level, where prior results influence the language model. This has been explored by [Johnston and Bangalore 2001] in their finite state transducers approach to multimodal fusion where the semantic interpretation of a gesture or partial speech utterance constrains future results in the recognition search at the token level.

2.4.2 Current Approaches

Some examples of the latest work in the fusion of multiple streams of inputs from different modalities have been conducted at CMU and OGI with their respective Jeanie (now Interact) and QuickSet projects. Jeanie is calendaring application where a user can create, modify or delete meetings involving other users and dates. Quickset is a military map application where units can be placed and manipulated on a map. Work has been progressing from fusing partial interpretations or semantic frames, to working at the lower level of partial recognition results.

Vo and Wood's work in Jeanie at CMU looked at using a frame merging strategy to fuse multimodal events [Vo and Wood 1996]. Results from different modes are parsed into partial filled frames that are continuously merged, with slot scores, to form an aggregated combined frame. A multimodal interpreter then decodes this frame to find the best hypothesis from possible slot combinations. Their calculation of 80% recognition accuracy, given perfect

multimodal interpretation and perfect recognition in both modes seems fairly poor. The method itself is also less elegant and formal than later work by Vo himself and at OGI.

Johnston's early work [Johnston 1997] at OGI with Oviatt focused on a general mechanism of fusing partial interpretations from different modes to resolve deictic references in speech to formalise past ad hoc approaches. The input modes, after full recognition, are semantically represented by underspecified typed feature structures and the multimodal fusion conducted through a unification process. Only compatible speech and gesture results are unified and the most probable unified type feature structure is taken as the action by the system. However, the notion of probability was not discussed in this paper, and the fusion of more than a binary set of events in a single command is problematic.

Later Vo [Vo 1997][Vo 1998] develops his approach for multimodal interactions further through deeper fusion of multimodal streams and a re-useable multimodal framework. This time, multimodal streams fuse at the partial results or token level. Words and gestures are aligned using a neural network topology he calls Multi-State Mutual Information Network, which is an augmented Multi-State Time Delay Neural Network with a Viterbi decoder to determine the best path through a series of words and gestures. Once the best path is found, a semantic post processor can assign slot parameters to the resulting multimodal input fragments. Beyond specifying a multimodal grammar, there is added complexity of having to train a likely domain dependent neural network and conducting pre-processing and post-processing steps. However, there are gains over the results in Vo's earlier work, and there is the added benefit of the neural network incrementally improving over time with use.

Research conducted by [Wu et al. 1999] exploring multimodal fusion yielded some statistical approaches and a hybrid approach at various levels in the recognition and understanding process. Their first approach integrates estimates of multimodal posterior probabilities of input classes for each mode from either the number of possible combinations of a given class with classes in another mode or from actual labelled data; this gives only a slight improvement. Their second approach is a bottom-up algorithm termed Members to Teams to Committee (MTC). Individual recognisers, with varying specifications, constitute members that each differently compute posterior estimates for each semantic output class. These members are combined into teams that learn the mode or prior probability of each of the members and their posterior estimates in different ways based on different training data and approaches, possibly discriminatively trained to reward correct members and penalise

incorrect ones through adjusting the mode probability. The committee level then decides the best team result through the highest confidence score. Wu et al. contribute theoretical lower and upper bounds to the multimodal recognition process, and their MTC architecture approaches this theoretical upper bound. The results are promising, but it appears the system requires much specialist knowledge to construct and employ in another application.

In [Johnston and Bangalore 2001], Johnston moves beyond his unification-based approaches and also proceeds to fuse at a lower token level than his earlier work. They choose to use a $N+1$ tape finite state transducer (FST) topology with N being the number of modes, and the additional 1, the output combined multimodal result tape. The FST is roughly equivalent to a multimodal grammar that parses the inputs and yields the output tape providing semantic information. This theoretically is more efficient than unification process and allows past inputs from one mode to influence future results; however, data is required to train the FST limiting portability.

3 Multimodal System Architecture

This section presents a general multimodal input architecture that attempts to be modular and domain independent, but more simplified than architectures such as [Vo 1998] or the MIT Galaxy hub. Most notably, dialog management components are neglected to focus on completing the system and handling multimodal inputs; the incorporation of a dialogue manager and knowledge base into this framework is straightforward. Design decisions are outlined and discussed. In Section 4, the actual implementation of this framework and fusion algorithm are described. Section 5 details the design and development of a sample application in this framework. These sections present the design and implementation aspects of this project.

3.1 System Description

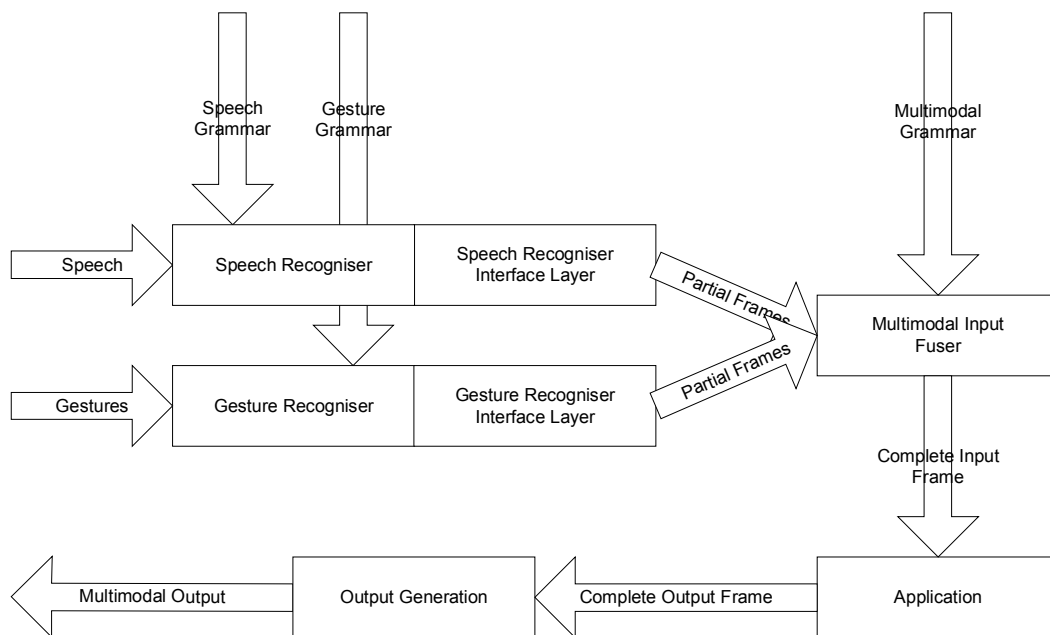


Figure 1: Multimodal System Architecture

The overall design of the system is presented at a high level in the logical diagram Figure 1 showing the interaction between components of a multimodal system.

The Multimodal System Architecture takes multimodal inputs such as speech utterances and gesture strokes from a user, and three different grammars for speech, gesture and multimodal grammars developed by the system developer. Within the framework presented lie four main components:

- A speech recogniser that transduces user words or utterances to semantic partial frames of the likely user intentions.
- A gesture recogniser that transduces user gesture strokes to semantic partial frames of the likely user intentions.
- A multimodal input fuser that takes input from various recognisers such as the speech and gesture recognisers and fuses their results into a complete semantic frame for the application.
- An application that accepts semantic frames and provides user feedback through multimodal output.

This configuration allows for clearly defined communication between components and isolation of the multimodal input fuser from specific recogniser technology and the application itself. Thus, the best recogniser technologies from different sources can be used in the overall system. In the general case, recognisers can be added to the multimodal system by writing an interface layer that converts the recogniser's output to partial frames that conform to those accepted by the Multimodal Input Fuser. Also the framework is domain mostly independent as it is designed for a generic turn-taking dialogue. To apply it to the specific application, only grammars need to be written to specify the allowable speech, gesture, and multimodal commands that are accepted. The following subsections describe each of the components in the framework in detail.

3.2 Automatic Speech Recogniser

The ASR used in this project is the commercial Nuance engine. The Nuance recogniser can handle large vocabulary, speaker independent applications, provides an interpretation engine and speech detection with barge-in, supports probabilistic finite state grammars or statistical language models, and is designed for telephony use with support for multiple languages. The Nuance recogniser is based on HMM modelling of phones with Gaussian mixture model output distributions and many optimisations to improve the efficiency of the search. The selection of the Nuance recogniser for this project is based on Nuance's reputation as one of the best commercially available recognisers for accuracy and speech detection robustness, the availability of interfaces to fuse the ASR, and ultimately the author's familiarity with it.

Nuance is primarily used in call centre based, command and control voice applications which makes it suited for this application, however there is an acoustic mismatch between the head mounted microphone used for this system and the regular land-line or mobile handsets that the acoustic models are trained with. Also, the Nuance native audio provider, i.e. for PC desktop audio, does not allow interruption of the prompt or barge-in; the use of a soft-phone and an IP based audio provider could allow for barge-in. These drawbacks are not expected to significantly impact this project.

Regarding the Speech Recognition Interface Layer, the Nuance ASR is integrated in this project through the RCEngine API and based on a sample C++ application XApp provided with the Nuance System software. XApp is designed to test the Nuance software configuration as a generic speech application that can load grammars, detect and recognise speech, connect to a recognition server, and display the final interpreted recognition results. Typically an application developer using the Nuance system writes grammars for the recogniser, an application through one of the API's such as DialogBuilder or Java SpeechObjects, and tunes the speech recognition process through parameters such as pruning or confidence thresholds or dictionary pronunciations. The grammar defines what utterances can be recognised, and the corresponding interpretations that should result. More information about the Nuance System software can be found in [Nuance 2001]. Changes to the XApp sample application, the grammar developed, and tuning are described in section 4.4.

3.3 Gesture Recogniser

The gesture recogniser used in this project is a sample C++ Win32 application called `GestureApp` written by Konstantin Boukreev to demonstrate the use of a feedforward multilayer perceptron neural network to recognise mouse gestures. An issue with `GestureApp` and many of the other gesture recognisers examined is that they are designed to recognise gesture command symbols and not spatial-location gestures such as points, lines or regions. `GestureApp` in particular also only recognises single stroke gestures e.g. symbols like arrows with separately drawn heads are not valid gestures. Other recognisers such as `Xstroke`, `SymbolCommander` and CMU's `Amulet` had problems such as being only available for Linux, lack of a programmable toolkit, and being deeply embedded in a GUI toolkit system respectively. Thus, `GestureApp` was selected for this project because of its ease of use and readily available source code, despite what is likely an inferior recognition algorithm.

The software, as available through free download on the Internet, samples mouse gestures initiated through a right click down, mouse movement and right button up event. The recognition algorithm is as follows, as adapted from [Boukreev 2001]:

1. Record a mouse path
2. Smooth a path to 16 base points
3. Transform points to absolute angles' vector
4. Compute sine and cosine for each angle
5. Pass resulting 32 value feature vector to neural network's inputs
6. Apply soft-max function on output network vector
7. Find and verify a winner

The neural network has 32 input nodes, a 32 neuron hidden layer, and 29 neuron output layer. The layers are fully connected with each other and the neural transfer function is log-sigmoid. Each of the 29 output neurons represents a gesture, and a soft-max function is applied to determine the output probability vector. Though the sum of the probabilities of all the gestures equals one, because of the nature of neural net the probabilities the network outputs are not grounded and could have dynamic range issues. The network itself is trained through standard back propagation of simulated training data where noise is added to representative point sequences of each gesture. Momentum is used in the training process where the learning rate starts fast and is slowly reduced. *GestureApp* allows one to train and save a neural network and then reload it at a later time.

Once an appropriate network is trained to recognise a set of chosen gestures, an interpreter is needed to map raw gesture results, i.e. symbols recognised, to partial interpretation frames e.g. a zigzag gesture could represent a [CMD: **zoom**] feature structure, whereas a line would need to return coordinate information about the gesture. Beyond gestures mapping to commands, spatial gestures need to be generically interpreted; point, region and line can be communicated with the following feature structures:

[POINT: **x;y**]

[REGION: **left;top;right;bottom**]

[LINE: **start.x;start.y;end.x;end.y**]

3.4 Multimodal Input Fuser

The Multimodal Input Fuser as shown in Figure 1, has clearly defined interfaces to receive partial frames from recognisers, load a multimodal grammar, and output a complete command frame making it fairly recogniser and application independent. The Multimodal Input Fuser fuses partial frames in a unification-based manner. A feature structure, which is a

set of attribute-value pairs, and associated probability and confidence scores comprise a frame. With the possible attributes defined in the other speech and gesture grammars, the multimodal grammar defines how feature structures can what fused frames are valid for the given application.

3.4.1 Unification

The usage of unification here is similar to the approach taken in [Johnston et al. 1997]. Unification of frames can be considered the union of the attribute-value pairs in the feature structures to produce the most common set of pairs, the addition of log probabilities and averaging of confidence scores. Some specific unification operations follow to clarify.

A deictic reference generates an attribute requiring a parameter that needs to be subsumed by a more specific value of the same type e.g. a general locative speech deictic as in ‘*zoom in here*’, could be filled by a point gesture:

$$\left[\begin{array}{l} \text{CMD: } \mathbf{zoom} \\ \text{POINT: } \square_{\text{point}} \end{array} \right] \sqcap \left[\text{POINT: } \mathbf{5;11} \right] \rightarrow \left[\begin{array}{l} \text{CMD: } \mathbf{zoom} \\ \text{POINT: } \mathbf{[5;11]_{\text{point}}} \end{array} \right]$$

where \sqcap is the symbol for unification. The \square_{point} represents a placeholder for a point value. If the same attribute has different, conflicting values or types, then the frames cannot unify and unification fails e.g.

$$\left[\text{CMD: } \mathbf{zoom} \right] \sqcap \left[\text{CMD: } \mathbf{pan} \right] \rightarrow \emptyset$$

One modification to this may be to examine the confidences of each the frames. If one is far surer than the other or in other words one has a low confidence, it could be rejected, and the result would be the more confident frame rather than failed unification. Otherwise, where attributes are unlike, attribute-value pairs are aggregated:

$$\left[\text{LOCATION: } \square_{\text{point}} \right] \sqcap \left[\text{LINE: } \mathbf{\{5;11\},\{6,13\}} \right] \rightarrow \left[\begin{array}{l} \text{LOCATION: } \square_{\text{point}} \\ \text{LINE: } \mathbf{\{5,11\},\{6,13\}} \end{array} \right]$$

3.4.2 Grammar Rules and Feature Structure Production

As Johnston outlines in [Johnston 1998], additional means to combine feature structures are needed to effectively handle more than two inputs. Similarly, this project incorporates a grammar rule to manage a command with multiple gestures, however Johnston uses recursive subcategorization to elegantly allow for multiple gestures to be linked. This simple non-recursive ternary rule $CMD \rightarrow CMD LOC LOC$ expands a command frame to one that accepts

location complements. This is invoked when, for example, a user says *'how far'* with two pointing gestures:

[CMD: **DISTANCE**] \sqcap [POINT: **5;11**] \sqcap [POINT: **6;13**] \rightarrow

CMD: DISTANCE
START: [POINT: 5;11]
END: [POINT: 6;13]

In this way, the Multimodal Input Fuser can build a unified command frame from a variety of different and number of interpretation frames. In section 4.6, the implementation of the Multimodal Input Fuser, actual details of the unification algorithm and the gathering of frames from the recogniser are discussed and limitations noted.

3.4.3 The Use of Ranked Result Lists

Instead of attempting to only unify the best result from each modality into a possible complete command, this project examines using N-Best lists of the top N interpreted partial result frames from each of the recognisers. It is possible that the top result is incorrect, but further down in the list lies the correct one. Two frames returned lower in the lists could unify where higher ranked frames do not unify or parse through the grammar. Probability calculations are necessary to rank unified frames for combinations of results in N-Best lists. It is hypothesised that through using unification and grammar constraints with N-best lists improved performance can be achieved.

3.4.4 Handling Other Input Modalities

Any generic multimodal framework needs to account for how additional modes of interaction can be incorporated. In this architecture, multiple recognisers can be integrated to send partial interpretation frames according to the interface to the Multimodal Input Fuser. It may be more useful though to consider a further level of abstraction where inputs are divided into linguistic and spatial-temporal fusers as shown in Figure 2.

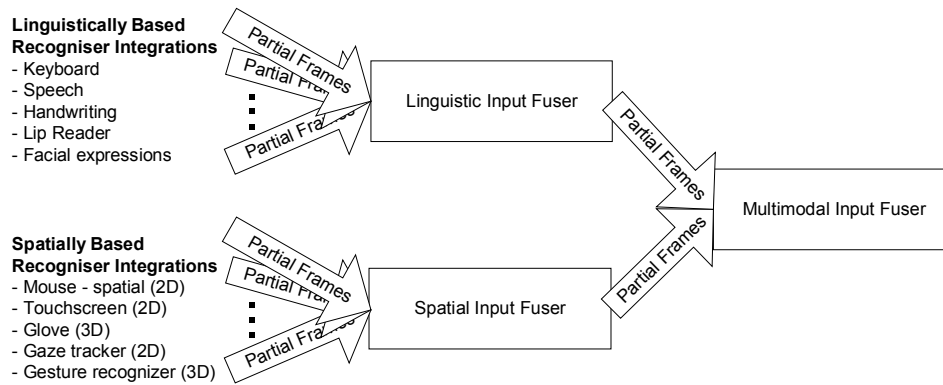


Figure 2: Handling other input modalities

The advantage of this would be to leverage the possibility that linguistically based recognisers could share the same grammar and interpreter. A similar case could be made for spatially based recognisers. This would ease the burden for developers of multimodal applications and could help reinforce redundancies within certain modes of interactions.

4 Multimodal System Implementation

In the previous section, the overall design of the multimodal framework was discussed. In this section, the implementation of the design is presented. The development environment, each of these components, and their communication protocols are described in detail in the following subsections. The implementation source code can be found on the companion CD.

4.1 Development Environment

This project was developed in C++ on 500-600 MHz Intel Pentium based machines using the Windows 2000 operating system and compiled with Microsoft Visual C++ 6.0.

4.1.1 Software Platform

Much of the graphical user interface elements are platform dependent. Since the majority of the code base in this project is leveraged from existing source code, there are a variety of different styles, functions and libraries used to implement similar features in different components; for example `GestureApp` is based on Microsoft Active Template Library (ATL), whilst other applications use Microsoft's Foundation Classes (MFC). The multithreading and socket communication also make Windows platform dependent calls as described in the Microsoft Platform SDK. Generally though, the data structures are platform independent and based on Standard Template Library (STL) data types.

4.1.2 Hardware Platform

To execute all the components of the multimodal system in the project on one machine, the computer should run at a minimum of 500 MHz, have at least 256 MB of memory and 500 MB of disk space. These are mostly requirements to execute the speech recogniser software. Also, this project is designed to be used with a touch sensitive screen for the gesturing. The Fujitsu Stylistic LT-P600 pen tablet is an ideal platform and is used for this project. Alternatives solutions could be to run the recognition software on a different machine in a network allowing the tablet computer to require less powerful hardware or to use a combination touch screen and screen peripheral device attached to a conventional PC console through the mouse and monitor connections.

4.2 System Components

The logical system architecture shown in Figure 1 in the previous section was divided into several different software components for implementation as shown in Figure 3. Each component is a free running application process in the Windows environment. The recogniser integrations are embedded into two separate components: the `SpeechSvr` and the `GestureApp` for speech and gesture recognition respectively. The interpreted recognition results or partial frames are sent to the core of the Multimodal Input System, embodied in the `InputSvr` application. Once the `InputSvr` application has fused the partial frames into complete valid command frames, this is sent to the application part of the `GestureApp` component. The application code in the `GestureApp` component then determines the output result and can send commands to the prompt playback portion of the `SpeechSvr`, or perform display updates to the map.

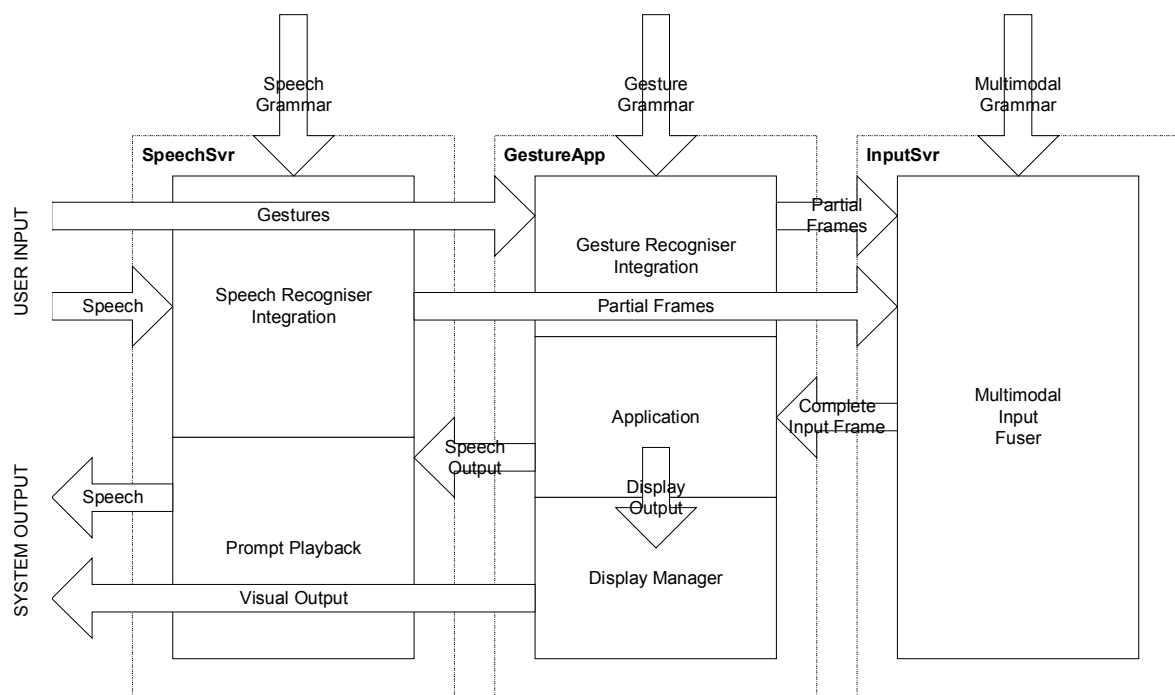


Figure 3: System components and interactions

4.3 Communication Between Components

The communication between all the components uses WinSock, a Windows implementation of Berkeley TCP/IP sockets. Freeware source code called `ws-util` was used to provide the low-level socket management of packaging, sending and receiving bytes. High-level functions were added to create a separate listener thread for components to function as servers listening for messages and to package message sending into one function requiring an

IP address, port and string message. The communication mechanism as implemented is inefficient in that socket streams are repeatedly closed and reopened for every message as opposed to maintaining the socket connection while both components are open. Also as a consequence, streams are in effect used only in one way, and so two streams and two socket ports are needed for two-way communications, when one would suffice.

The communications do perform adequately despite these drawbacks and allow for each of the components to exist on different machines across a network if necessary, however all the components can also (and are for this project) be run on one standalone machine. The socket communication functionality is implemented in the shared `ws-util.cpp` `basic-client.cpp` source files and the component specific `threaded-server.cpp` file. The components communicate using messages with delimited tokens; the message formats will be discussed in further detail with each component that produces them.

The recogniser integrations provide two types of messages: `StartXxxxRec` and `EndXxxxRec`. There is a necessity to differentiate recognition results from different modes for timing; this will be discussed later in section 4.6.1. The format of the messages is as follows:

```
Start[RecType]Rec,[Date],[Time recognition started]
```

and

```
End[RecType]Rec, [Date],[Time recognition ended]
[result number],[result string],[confidence],[log probability],
  [number of interpretations that follow]
[1st interpretation, 1st feature],[1st interpretation, 1st feature's
  value], [1st interpretation, 2nd feature],[1st interpretation, 2nd
  feature's value], ... (row ends with '@@')
[2nd interpretation, 1st feature],[ 2nd interpretation, 1st feature's
  value], [1st interpretation, 2nd feature],[ 2nd interpretation,
  2nd feature's value], ... (row ends with '@@')
(the number of interpretation rows must equal the [number of
  interpretations that follow]
(next result, etc.)
```

[RecType] refers to the modality generating the message, in this case Speech or Gesture. Specific examples of this messaging are provided in section 5.

To test the messaging and various component interfaces, another component called `TestSvr` was written to use the client and server functions to allow custom messages to be sent to the other components.

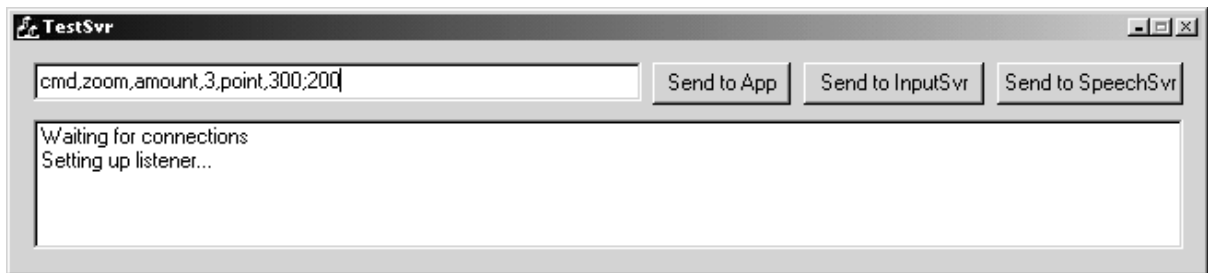


Figure 4: TestSvr interface test application

4.4 SpeechSvr

As mentioned in the previous section, the `SpeechSvr` is based on the `XApp` sample programming example provided with the Nuance system software. The `SpeechSvr` is the end client in the Nuance System architecture as pictured in Figure 5. It handles speech through the recognition client (`recclient`), which detects and passes speech to the `resource-manager`, which then finds a recognition server (`recserver`) to process the audio. The `recserver` signals the application of different events in the recognition process; eventually the application calls some Nuance API functions to retrieve results.

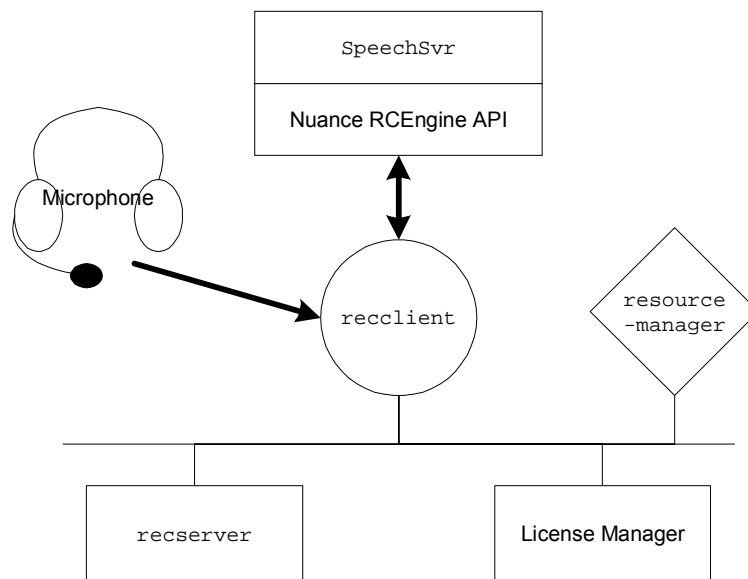


Figure 5: SpeechSvr in Nuance System architecture

Changes made to the base code include incorporating the previously mentioned socket communications functionality, incorporation of call logging and utterance recording, the removal of a memory leaks in the existing code (though some still exist), and cosmetic changes to the interface to reflect the new usage of the component. The logging mechanism records the audio of the utterance plus timing information, recognition and interpretation results, and the confidence scores. Also, with the diversity of accents present in the subject

pool, the grammar was compiled against two different acoustic models English.UK.1.6 and English.America.1. These are what Nuance calls single pass models which are more memory and CPU efficient but less robust and accurate; with the small grammar this was deemed a good trade-off. These two different versions of the grammar can be loaded through the two different SpeechSvr start-up scripts `go.speechsvr.uk` and `go.speechsvr.us` respectively.

New functions added to the `SpeechSvr`, which can be called by other components through the socket messaging, include `NewAudioChannel()` that will start a new call logging session, `Play()` that allows other components to only queue and play prompts, and `PlayAndRecognize()` that allows other components to queue and play prompts and start the recognition process. The play and recognise functions though are not fully implemented to block and so a call to them while playback or recognition is occurring causes a fatal error.

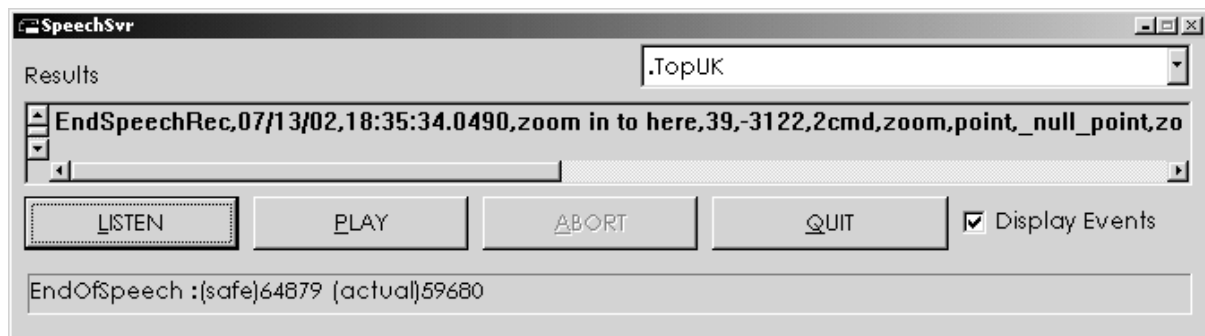


Figure 6: SpeechSvr application window

The key additional functionality is the handling of the `StartofSpeech` event to signal the `InputSvr` of the start of recognition process, and handling of the `RecognitionStopped` event. The `RecognitionStopped` event indicates that speech has ended, and the recserver has completed processing the utterance and has a recognition result. At this point, additional code was added to retrieve the probability of a result using `RecResultTotalProbability()` and to process the Nuance interpretation list using `RecResult` and `NLResult` functions to retrieve the attribute-value pairs to send to the `InputSvr`.

4.5 GestureApp

The `GestureApp` component provides three main functions: gesture recognition, the tourist application functionality and control over the client display window. Ideally, these three

features could be separated as outlined in the Figure 1. However, since the mouse events for a client window, the client window display, and the application itself are relatively tightly coupled, separating these functions would be involved, though possible, hence time was spent on other aspects of the project. The major changes to `GestureApp` beyond the original source include modifying the gesture recogniser, changing the gesture alphabet, writing a gesture interpreter and adding the application code and graphical map manipulation routines.

4.5.1 Gesture Recognition Improvements

The original `GestureApp` application is shown in Figure 7. As shown, it captures a mouse path through `WM_MOUSEMOVE` events, reduces the path to 16 points (the darker points among the light) and outputs a top three list of most likely gestures. The menus allow you to train, test, recognise, load and save neural nets.

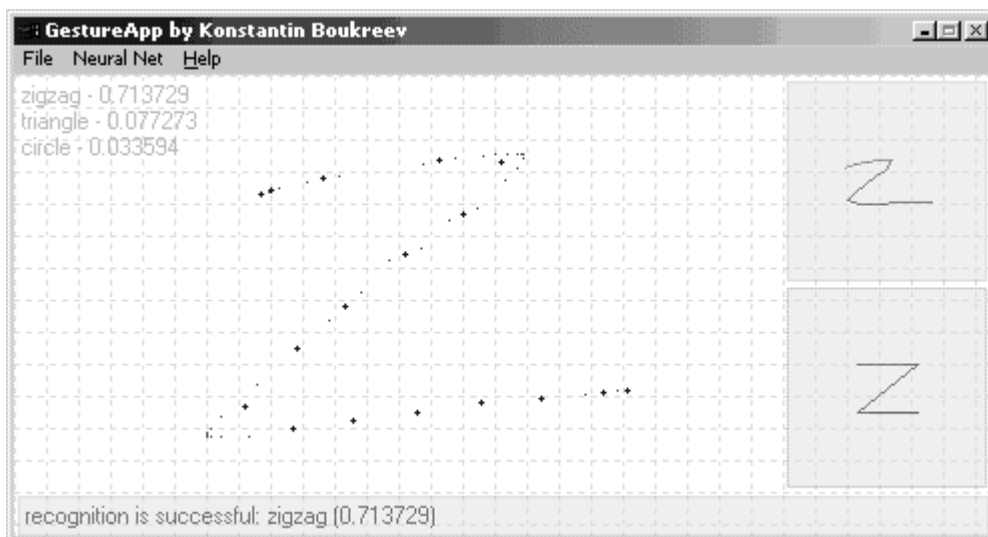


Figure 7: Snapshot of original `GestureApp`

However, there were many modifications required to incorporate the gesture recogniser into this project. First, the mouse path capture was changed from using the right button to left for pen tablet use. The original algorithm only worked with 16 mouse point samples or more; an interpolation algorithm to stretch the path to more points was written since fast gesturing, e.g. common to drawing lines, tended to have only a few points causing the original algorithm to reject the gesture. An issue with the pen tablet is that gestures tend to be performed quickly causing poor gesture path acquisition due to windows `WM_MOUSEMOVE` messages being sampled too slowly. There does not appear to be an easy method to increase the sampling rate of mouse movements.

The algorithm to stretch the path in the `Board::TransformPath()` function is:

```
// overstretch path pts by 1.5 times, then reduce pts to smooth
// stretches path by finding longest distance between points, and
// adding a pt at the median
while (loop through the path with the number of points < 16*1.5)
{
    for (loop through all points)
        find the largest distance between two points in the path

    insert a new point in between the largest distance
}

// smooth path down
while (loop through the path with the number of points > 16)
{
    for (loop through all points)
        find the smallest distance between two points in the path

    erase one point, and shift the other to the middle of distance
}
```

While this algorithm works well in practice to add resolution to the gesture path in practice, it cannot make up for fundamental problem of the slow sampling rate. For example, if a circle is sampled at only three points, the algorithm will interpolate these points linearly, and thus a triangular path will result instead of a curvilinear one. One way to mitigate this is to provide user feedback by drawing the raw mouse points captured; the user can then see if they are moving the pen too quickly and adjust accordingly. It was decided that displaying the gesture path on the screen was too distracting, however this was purely a subjective decision.

The gesture recogniser was also augmented to recognise points; in the original algorithm point gestures would have too few samples to be recognised. When two mouse points are recorded, one at the mouse button down and another at the mouse up, this results in a point gesture and interpretation at 100% probability and confidence. An improvement would be to possibly smooth the probability of being a point reference to a line or region based on the number of sample points generated or the total distance between all points. Even better would be to incorporate the recognition of point gestures in the neural net, possibly through the addition of scale dependent features. Also, decreasing the number of points needed for a gesture increases the chances of a false touch on the screen being interpreted as a gesture.

Lastly, a gesture recognition confidence metric was devised:

$$C(n) = \frac{P(n) + (P(n) - P(n+1))}{2} \times 100$$

where

n represents the rank of a result in an N - Best list

C(n) represents the confidence of result n, and

P(n) represents the probability of result n

This balances and compares the probability with the probability of the next best result. High probability results, with the probability of next best result relatively low, have a high confidence; high probability, with the next best also having a high probability, reduces confidence. Although non-standard, this seems to work fairly well in practice.

4.5.2 Gesture Symbols, Interpreter and Grammar

The original twenty-nine symbols recognised by *GestureApp* are shown in Figure 8. As mentioned previously, these are mainly symbols, however for this project spatially oriented gestures like lines, regions, and points need to be recognised.

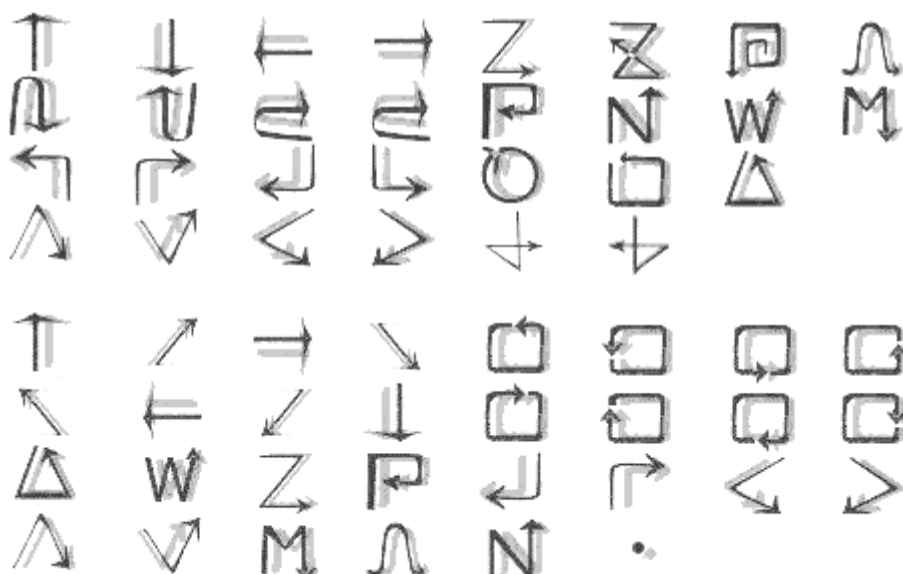


Figure 8: Twenty-nine gestures originally recognised by *GestureApp* (top) and the modified gesture alphabet of thirty (bottom)

The original gesture alphabet was modified to add four diagonal lines for the directions up-left, up-right, down-left and down-right, as well as six more region gestures that start at 3 o'clock, 6 o'clock and 9 o'clock clockwise and counter-clockwise, reserving sixteen gestures

symbols for spatial recognition. The neural net training data was modified, a new neural net trained using 9000 training cycles and otherwise default values, tested and saved. The addition of these gesture and the removal of 10 unused gestures proved in informal empirical testing to adequately capture the spatial gestures of the line, region and point variety.

A gesture interpreter was written to process the returned results and provide additional spatial information depending on the result. For lines, start and end points were also provided in the interpretations; for regions, a bounding box marked by the top-left and bottom-right corners coordinates; and for points, the point location. The interpreter allows multiple symbols to be mapped to the same interpretation e.g. 'Z' and 'M' can be mapped to zoom, and 'P' to pan. In a more refined system, this mapping could be done in a grammar; however, in this case a static finite state machine is embedded in the function `_Main::InterpretBoard` in the source code.

As well, the interpreter should only return the most highly probable interpretations with no duplicate results; this is not the case due to time constraints. For example, since regions are recognised through eight region gestures differentiated by starting point and direction as shown in Figure 8, it is possible a single gesture could result in a N-Best list of region, circle, region because it matches well to two different region patterns in the neural network. This pushes down other different possible, although less likely, results down the N-Best list where they would not be used because of the duplicate result; in this example with a N-Best length of three, the fourth result would not be used although really only two different interpretations are returned instead of the N-Best list length of three.

4.6 InputSvr

The `InputSvr` performs the main task of gathering inputs from a variety of recognition servers and fusing them into a complete frame for the application to execute.

4.6.1 Multimodal Input Acquisition

The `InputSvr` needs to determine when a series of multimodal inputs is complete signalling the end of the user's turn and dialogue act allowing the application to respond. Once the `InputSvr` has completed fusion, it is ready for a next set of events. The application must trigger the `SpeechSvr` to start listening. The diagram in Figure 9 shows the messages sent to the `InputSvr` from the various recognisers and timers that are present when the user enters two point gestures and says *'how far is it from there to here'*.

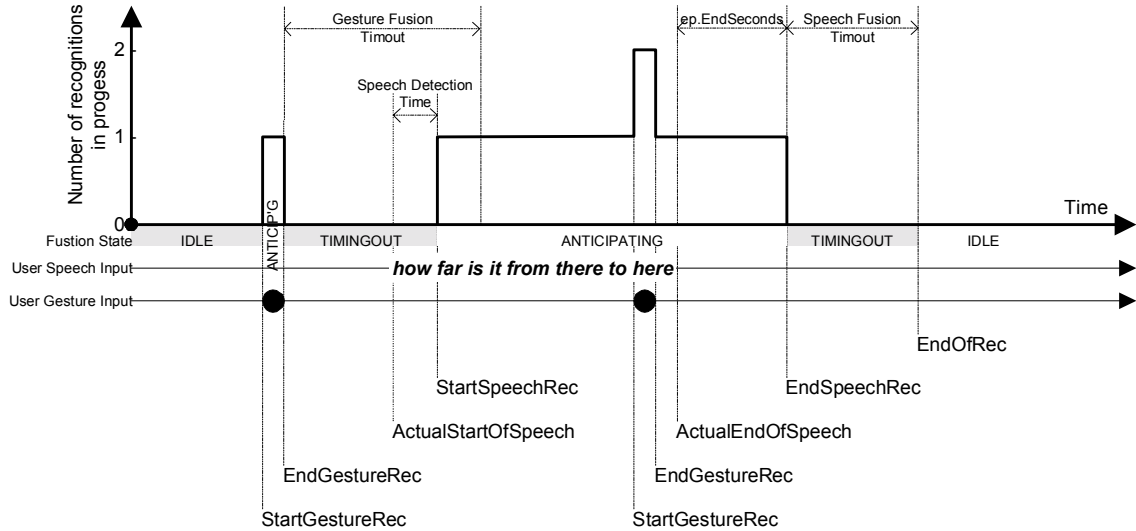


Figure 9: Sample timing diagram

The essence of the algorithm to capture multimodal inputs is to increment a counter for every recognition process that is started, and to decrement every time one reports ending. When the counter is decremented to zero, a timer is started depending whether the end recognition message came from the gesture recogniser or speech. This is necessary because the speech recogniser also has an `ep.EndSeconds` length to ensure the proper detection of the end of speech, whereas the gesture recogniser lacks this delay. The `ep.EndSeconds` parameter is set at 2.0 seconds, the `FUSION_TIMEOUT_SPEECH` timeout interval at 1000 milliseconds, whereas the `FUSION_TIMEOUT_GESTURE` is set at 2000 milliseconds; this was found to work adequately but could be tuned to decrease latency. The Windows timer `ID_FUSIONTIMEOUT` is set at the timeout interval for the modality. When it times out, as indicated by the `EndOfRec` in Figure 9, the `FusionCallback` function is called which performs the final fusion routines, checks for complete frames, finds the best complete command frame, and whether to accept and send this frame to the application or to send a rejection. If while the timer is timing out another start recognition event is received, the timer is reset. An additional refinement could be made to take into account the delay in receiving the start of speech recognition message because of the time it takes the recognition software to acknowledge that speech has started.

4.6.2 Data Structures

There are a few classes that aggregate and wrap mostly STL data types. The most basic structure represents feature structures, `TFS`, which is a hash table of `string` keys and `string` hash values. An interpretation has a result string, a `TFS`, and associated probability

and confidence scores. Each partial interpretation in N-Best lists sent by the recogniser integrations is stored in a list of `_interpretation` structures of type `_interpretation_list`. These are aggregated into a list of multimodal event frames stored in a structure of type `_mmevent_list`. The following is a listing of the main data types used in the multimodal fusion:

```
// Shared/globals.h
//feature structure
typedef std::map<string, string, std::less< string > > TFS;

class _interpretation {
public:
    string      m_result;    // raw recognition result
    TFS         m_tfs;      // feature structure
    int         m_conf;     // confidence
    float       m_prob;     // probability

    void Print(std::ostream *os);
};

typedef std::list<_interpretation> _interpretation_list;

class _mmevent {
public:
    _interpretation_list m_interpretations;
};

typedef std::list<_mmevent> _mmevent_list;
```

4.6.3 Multimodal Fusion Algorithm

The `InputSvr` collects partial interpretation frames, parses them, and unifies them in the monolithic function `SocketHandler`. The high level steps are:

```
while receiving partial frames
    parse socket stream into interpretation list of partial frames
      of TFSs
    unify each incoming partial frames with current list of fused
      frames and store in new list
    add frame log probabilities and confidences
    new list becomes current list

reject frames that do not parse in multimodal grammar as incomplete
find most highly probable complete frame
accept and send only if confidence above rejection threshold
```

The `InputSvr` starts out by initialising all the fusion variables with the function `FusionReset`. This also seeds the variable `fused` of type `_mmevent_list` with underspecified frames for future partial interpretations to unify with to implement the ternary grammar composition rule. These underspecified frames act as a grammar for the system to interpret how context free frames such as regions should be interpreted with command frames like *'zoom in'* that do not specify how to use additional frames if available:

$$\begin{bmatrix} \text{CMD:} & \square \\ \text{START:} & \square_{\text{point}} \\ \text{END:} & \square_{\text{point}} \end{bmatrix} \sqcap \begin{bmatrix} \text{CMD:} & \mathbf{zoom} \\ \text{ZOOM:} & \mathbf{in} \end{bmatrix} \sqcap [\text{POINT: } \mathbf{5;11}] \sqcap [\text{POINT: } \mathbf{5;11}] \rightarrow \\
\begin{bmatrix} \text{CMD:} & \mathbf{zoom} \\ \text{ZOOM:} & \mathbf{in} \\ \text{START:} & \mathbf{5;11} \\ \text{END:} & \mathbf{6;13} \end{bmatrix}$$

Here, without the initial underspecified frame, the zoom command cannot incorporate the two point frames to yield a command frame to zoom in the box bounded by the two points. One could define an extra interpretation in the speech grammar that requires these two point features, however this mechanism provides a general elegant way to do so.

When a recogniser integration sends a message to the `InputSvr`, it is eventually received by the `SocketHandler`. A `StartXxxxRecognition` message increments the counter `nRecognitionProcessesInProgress` and changes the fusion state to `anticipating` as shown previously in Figure 9. An `EndXxxxRecognition` message decrements the counter and possibly changes the fusion state. The speech recogniser is set to return ten results and the gesture recogniser three. The `InputSvr` also parses the message and builds a `mm_event` structure for each result in the N-Best list, which holds an interpretation list `m_interpretations` of possible TFS interpretations of the result and associated probability `m_prob` and confidence score `m_conf`. Then each TFS of the interpretation of each result in `mm_event` is unified, if possible using the `Unify` function, with each TFS in the global static fused list of interpretations forming an ongoing list of progressing unifications. In the previous example, this represents first unifying the 1st underspecified frame, with the 2nd zoom command frame, then unifying this result with the point frame, and then unifying that result with the final point frame.

The `Unify` function actually performs two steps: either a variable filling action or standard unification. For the former it calls a subroutine `VariableFill` to fill `_null_type` values for any features that match the type in the other frame. The `_null` prefix is used to determine whether a value is filled or not, and the following text after an underscore is the type of value that should fill the feature e.g. `_null_point` indicates that the value should be filled with a point TFS. Otherwise standard unification is attempted. If they have no duplicate attributes,

or one or more attributes are underspecified with a value of `_null` then the sets of feature structures are unified through the removal of underspecified attributes and a set union operation. If however, there are features with conflicting values, the unification fails and resulting feature structure is empty.

It should be noted that the `VariableFill` operates slightly different than what may be expected. The entire feature value pair is not inserted; only the value replaces the `_null` placeholder for convenience:

$$[\text{LOCATION: } _point] \sqcap [\text{POINT: } \mathbf{5;11}] \rightarrow [\text{LOCATION: } \mathbf{5;11}]$$

Finally, when the fusion timer expires and the multimodal turn complete, the multimodal grammar decides what unified frames are valid command frames for the application through some parameter checking. From the set of valid frames, the one with maximum probability exceeding a confidence threshold is sent to the application. The confidence threshold is set at 25 for this application, which worked well in practice, although is not finely tuned at all.

Overall, the unification algorithm is significantly simplified than the application in [Johnston 1998] or typical unification grammars used for language modelling mostly due to time constraints. There is no support for re-entrancy, co-referencing, or inheritance, and feature values cannot be another feature structure. However the current application does not demand a more sophisticated implementation.

5 Multimodal Tourist Application

An application is needed to demonstrate and evaluate this multimodal framework that elicits multimodal commands and naturally combines linguistic and spatial information, yet is simple enough to build quickly in a prototype form in the short project time frame. Most research conducted in multimodal systems use map applications; it is so with this project. The application chosen for this project is an Interactive Tourist Guide for Bristol and is called iTour. It should be noted that this application is very similar to the QUICKTOUR design example in [Vo 1998]. Other applications considered include a Paint program or game interface to Doom or Warcraft; however the former, while a good spatial command and control task, has individual commands that are not typically multimodal, and the latter, too intensive to implement in the time frame of the project.

In this section the needs of the application are outlined, and a suitable design formulated. The details of how this design is implemented are presented, and the necessary domain specific inputs to the multimodal system framework discussed.

5.1 Requirements

This interactive tour guide iTour provides two general functions for users:

1. *Adjusting the map view.* The user can either zoom in or out of the current view into a point or region by an optional magnification factor, or shift the view up, down, left or right, or in the direction of a drawn line.
2. *Route planning.* The user can ask for the distance or the best path between two locations as indicated by points, regions, place names, or drawing a line between the two points. As well, the user can ask for the map to display the location of a specific site.

This particular task should elicit a good variety of multimodal commands as described by [Oviatt et al. 1997] and [Vo 1998]. However, it could quickly lead to complex utterances for speech recogniser with complex PP attachments e.g. *in, along, within, around, ...* causing many out of grammar utterances. Also, there may be a tendency to use iTour as a question answering system i.e. to simply ask the system to answer the task question posed. The speech and gesture modes are mostly complementary as noted in [Oviatt et al. 1997], with minimal

redundancy in the multimodal commands compared other mode pairs, such as speech and lip movements, which can be mostly redundant. This will likely limit the amount of recognition improvement from combining the two modes, but gains should be seen in efficiency or user satisfaction from more natural interactions.

5.1.1 Use Cases

To demonstrate the possible usages of iTour, these are some example use cases of the system with sample user responses:

Use Case 1

You are at Berkeley Square in the West End. Try to find an attraction within 1 kilometre of where you are, the name of it, and how far it is from you.

'Where is the West End?'

'Zoom in here' (user points at West End)

'How far is it from here to here?' (user gestures a line between the two places)

Use Case 2

What are the names of the restaurants along the route Hotwell Road - Anchor Road - and Park Street?

'Zoom in to here' (user points to Hotwell Road)

'Move right'

'Move this way' (user draws upward)

Use Case 3

Find the attractions near Clifton Village and the shortest path to visit the three.

'Zoom out'

'Zoom in to here' (user points to Clifton Village)

'How do I get from here to there?' (user points between the bridge and the visitor centre)

'How do I get from here to here?' (user points between the bridge and observatory)

'How do I get from there to here?' (user points between the visitor centre and observatory)

5.2 Design

iTour supports the following commands in Table 1 to meet the two general required functions outlined previously in 5.1 and to enable users to conduct the scenarios in 5.1.1.

Command	Parameter Slot	Description
ZoomIn	Point	A centre point to zoom about.
	Region	A bounding box of where to zoom.
	Amount	The zoom factor.
ZoomOut	Amount	The zoom factor.
Pan	Direction	Either up, down, left or right.
	Line	A line indicating the direction of pan.
QueryLocation	Place	The index of a specific site.
QueryPath	Start	A start location for the path.
	End	An end location for the path.
QueryDistance	Start	A start location for the distance.
	End	An end location for the distance.

Table 1: iTour command descriptions

The application also requires some basic form of dialogue management. The current view of the map provides some form of memory of the current dialogue state. Paths that are displayed stay present until the application is terminated, hence to repeatedly test the application, restarting it will be necessary. A diagram outlining the flow of the dialog is shown in Figure 10. The application sends `PlayAndRecognize` messages to play the prompts shown, and then perform recognition after the results are played.

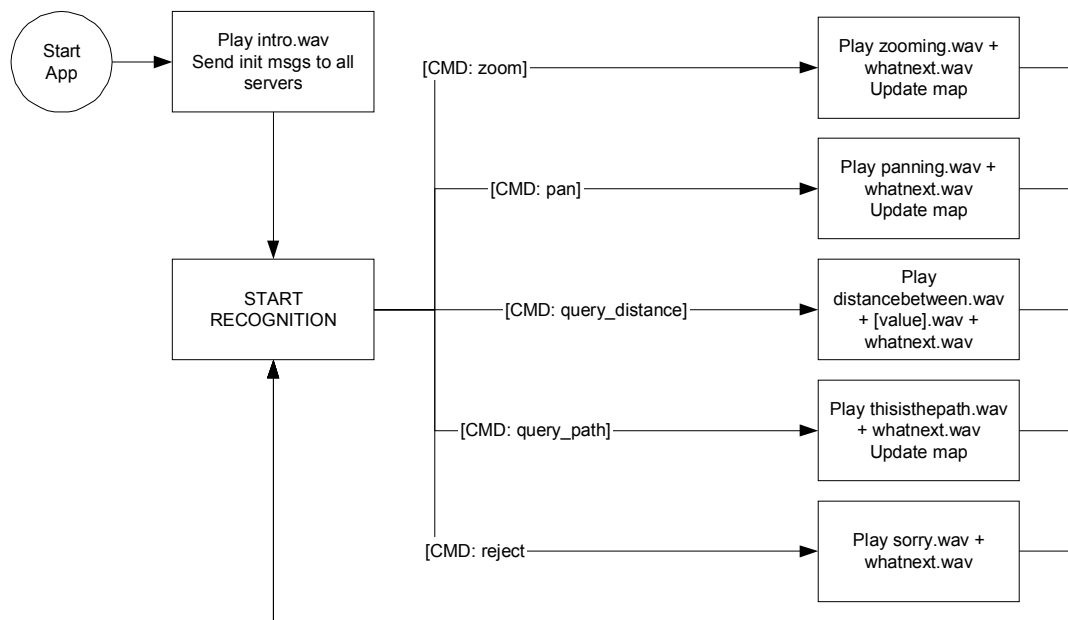


Figure 10: Dialogue flow diagram

There is no barge-in in this application. The application depends on the `InputSvr` to handle the `START_RECOGNITION` messages sent by the recognition servers and to provide a feature structure as the command result. This result is then processed by the application to

formulate a proper response that is given to the user, after which the whole dialogue turn starts again. A *'quit'* phrase, not shown in Figure 10, allows the user to quit the application, although the Escape key or Alt-F4 shortcut work as well.

5.3 Implementation

The application is a subcomponent of `GestureApp`. It consists primarily of a map stored as a 4 MB bitmap in memory with some metadata about the current view window on the map, the location of various attractions, their distances, and how to draw paths between some of the places. The application subcomponent listens on a socket for API commands on a separate thread. When a message is received, the command frame is parsed into a feature structure. Since it is assumed that the frame received is valid due to the checking in the `InputSvr`, only the necessary attributes in the feature structure are checked to invoke a command.

The zoom and pan commands only manipulate the view window on the bitmap. The distance function only calculates the city block distance between two points and echoes a rounded distance value. The display path function is mocked up for only the few places outlined in Use Case 3.

5.3.1 Speech Application Grammar

The Nuance `recserver` requires a recognition package formed by a grammar and acoustic model to perform recognition against. The full grammar listing can be found in the accompanying CD. The grammar supports the main commands zoom and pan, and search, path and distance queries. Some examples of these are respectively:

'Zoom in to here.'

'I want to move to the right.'

'Where is Berkeley Square?'

'How do I get from here to the Clifton Suspension Bridge?'

'How far is Cabot Tower from Temple Meads rail station please?'

These phrases yield the following Nuance NL interpretations:

```
{<cmd zoom> <point _null_point> <zoom in>}
{<cmd pan> <direction right>}
{<cmd query_search> <place 14>}
{<cmd query_path> <start _null_point> <end 3>}
{<cmd query_distance> <end 8> <start 10>}
```

The first utterance would send two messages to the InputSvr, one at the start of speech, and one after recognition stopped:

```
Received data at 07/01/02,05:20:59.429 is:
- start -
StartSpeechRec,07/01/02,05:20:59.429
- end -
```

```
Received data at 07/01/02,05:21:01.693 is:
- start -
EndSpeechRec,07/01/02,05:21:01.683
0, zoom in to here, 71, -3200, 2
cmd, zoom, point, _null_point, zoom, in, @@
cmd, zoom, region, _null_region, zoom, in, @@
1, zoom in, 17, -13200, 1
cmd, zoom, zoom, in, @@
2, zoom in two times, 12, -15700, 1
cmd, zoom, zoom, in, amount, 2@@
- end -
```

These two messages indicate the recognition process took about 2.2 seconds, producing a list of three possible results, ordered by confidence, and the top one being the correct result with a confidence of 71, and a log probability of -3200. The confidence-scoring algorithm is specific and internal to recognisers; being a commercial entity, Nuance does not reveal it. The confidence score is a rating between 0 and 100 of how sure the recogniser is that the result is correct with 0 being no confidence, and 100 being perfectly confident. The log probability varies from negative infinity (or the negative numerical limit of the data type, in this case float) and 0. The best result also has two interpretations: *this* can be interpreted as referring to a point or region. The generation of multiple interpretations represents multiple parses with different interpretations in the speech grammar for the same phrase. The other two results have lower confidences and probabilities, and only one interpretation each. The top result from the recogniser yields the following feature structures, one for each of the two interpretations:

$$\begin{bmatrix} \text{CMD:} & \mathbf{zoom} \\ \text{ZOOM:} & \mathbf{in} \\ \text{POINT:} & \mathbf{[]_{point}} \end{bmatrix} \text{ and } \begin{bmatrix} \text{CMD:} & \mathbf{zoom} \\ \text{ZOOM:} & \mathbf{in} \\ \text{REGION:} & \mathbf{[]_{region}} \end{bmatrix}$$

An improvement in the communication of recognition results to the `InputSvr` would be to use XML similar to the W3C Natural Language Semantics Markup Language. This would improve the readability of communications and improve the overall robustness in the systems with well-defined messages and checking of ‘wellformedness’ in the XML. [Johnston and Bangalore 2001] also prescribe the use of XML to encode the meaning representation.

5.3.2 Gesture Application Grammar

The final gesture alphabet of twenty-nine gestures plus the point with their interpretations is shown in Figure 11. The use of twenty-nine symbols is arbitrary; the neural net can handle fewer or more symbols, and some gestures are not used but kept as possible gestures to be used later, and distracters to improve rejection. The interpretations to specific results are statically encoded in the function `_Main::InterpretBoard` in `GestureApp` as opposed to using an external grammar file due to the limitations stated previously.

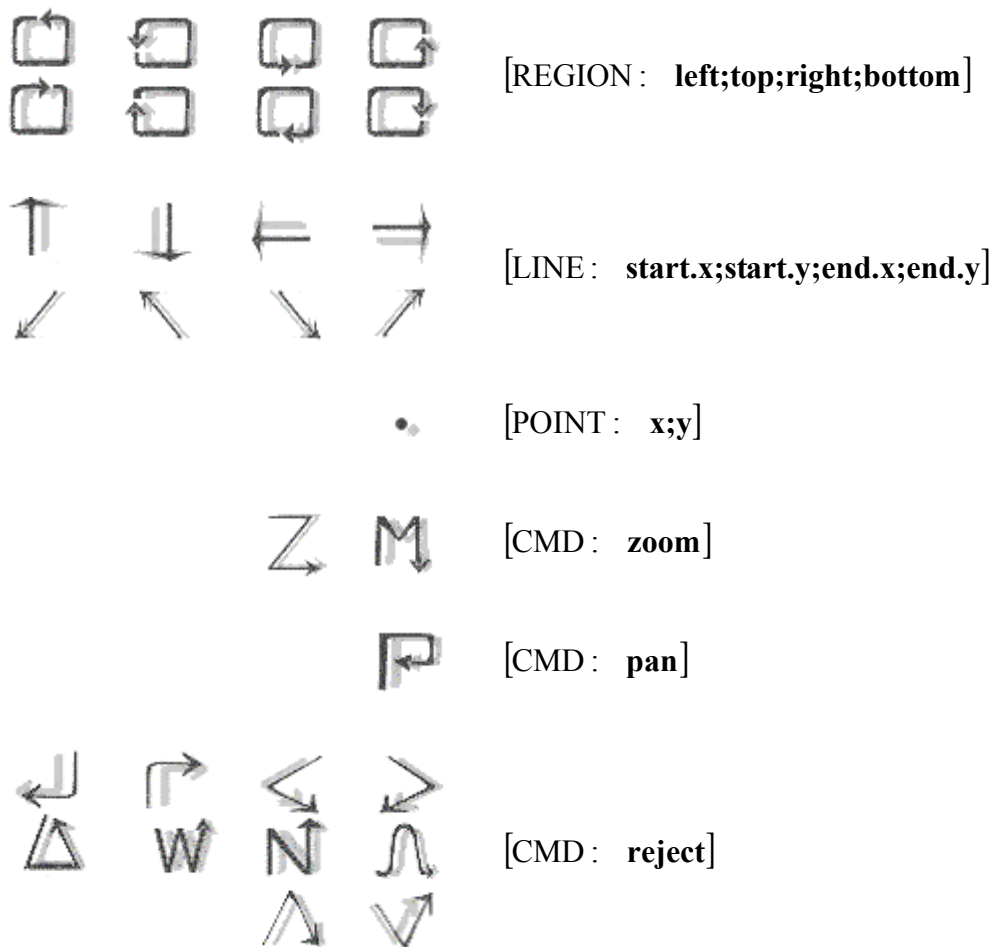


Figure 11: Final gesture alphabet with interpretations

5.3.3 Multimodal Grammar

The unification process specified in 4.6.3 is quite generative and thus can form feature structures that are not valid commands. Once the input acquisition process is complete, the `ProcessFrame` function is called to validate frames. Frames that are incomplete, that is one or more features have `_null` values or lack a `cmd` feature, are rejected. Then a finite state machine processes the TFS to ensure that the TFS matches a command the application can execute.

In general, for future extensibility, a multimodal grammar could be written to specify those TFS that are valid; currently this is statically encoded in the source code. The `ProcessFrame` function could use the grammar to parse each candidate frame; those that do are valid frames. The specification of a grammar formalism and development of a multimodal grammar parser were not incorporated in this project.

5.4 The Final System

The photo in Figure 12 shows the complete system in use, with the user gesturing on the touch screen with a pen stylus and talking to it through a Plantronics headphone with mounted microphone. The map is zoomed in, with the Cabot Tower and @Bristol marked, and a path between them shown.



Figure 12: The iTour system in use

6 Evaluation

The section presents the results of the evaluation of the multimodal fusion algorithm developed in the context of the map application. The goals of the evaluation are to reaffirm previous findings about multimodal systems in the context of this system and examine the performance of the multimodal fusion algorithm developed.

6.1 Evaluation Protocol

To evaluate the multimodal framework and application, three volunteer students were asked to perform a set of tasks with iTour. An introductory form, task list and questionnaire were given; these can be found in Appendix A. The protocol attempted to be idealistic in giving user end goals to achieve, without specifying the method to do so. It was expected that users would follow a similar set of steps to reach the end goals. If they failed thrice to complete the current task, users were asked to move to the next task, to mimic real world frustration and bailouts. Users were also asked to perform the task multimodally, and then with either gesture or speech only, for the tasks that could be done in the single mode. The introductory sheet only provided a minimal number of sample phrases to use, and no explicit instructions on how to interact with the system multimodally.

While this methodology mimics more closely the situation that this type of system would be used in reality, there are several drawbacks. Users can be confused by the open ended tasks and generate a lot of out of vocabulary utterances and gestures. Worse still, the user could fail completely to interact successfully with the system and not finish any tasks. The system, as implemented, has a limited grammar and minimal dialogue management, which can only aggravate errors for novice users. The alternative, used for example by [Vo 1998], is to get scripted results with a series of clear tasks that only require one dialogue turn each. This would yield more well formed commands and less out of vocabulary items, however it limits the spontaneous variety of user utterances and the reality of the interaction.

In the end, it was decided to test the system in a more realistic fashion.

6.2 Evaluation Results

The multimodal system was evaluated qualitatively and quantitatively. The system recorded most messages between components, timing information, the gesture paths, speech utterances, the partial interpretations of each, the inputs to the `InputSvr`, fused candidate

frames, and the final decision sent to the application. There was difficulty in using more subjects due to time and the lack of infrastructure to easily transcribe the gesture, speech and more acutely the complete multimodal turn. The questionnaire provided some qualitative sense of what the subjects felt about multimodal interaction to compare with results from other papers. The relative few numbers of subjects and possible transcription errors compromises the results, however some useful analysis and conclusions can be made from the evaluation. A summary of the evaluation data collected is tabulated in Table 2. Un-useable items were those that were involved in unexplained system errors where results were not returned and so the system could not be evaluated in these instances.

Total speech utterances	130
Total useable speech utterances	114
Total gesture symbols	80
Total multimodal turns	144
Total useable multimodal turns	128

Table 2: Summary of evaluation data collected

The following accuracy metrics are all computed only with in grammar utterances, i.e. those utterances whose transcription parses in the relevant grammar. Out of grammar utterances are those that do not parse in the relevant grammar.

6.2.1 Speech Recognition Accuracy

The speech recognition performance is summarised in Table 3. Although not brilliant, the accuracy is sufficient for operation given the acoustic model mismatch, and the amount of breath noise in the audio from either the poor microphone or microphone positioning. Around have the utterances had breath noise, and from the author’s experience, this was likely the largest contributing factor to the recognition problems; in other data, such as through the phone, there is typically not so much breath noise. It would have been useful to test more thoroughly the best position for the microphone, possibly further off to the side, before evaluation. With an iteration of the grammar, some tuning of the recognition parameters, but mostly speech detection tuning and microphone selection and refinement, the command accuracy should be able to reach at least 90%. More specific methods to improve performance like N-gram language modelling or adaptation were not explored.

Sentence accuracy (%)	60
Command accuracy (%)	78
Command Oracle accuracy with N-Best list (%)	85
Out of grammar rate (%)	32

Table 3: Speech recognition performance

Sentence accuracy is the rate at which sentences returned by the recogniser exactly matched the transcription. [Vo 1998] experienced rather poor sentence accuracy rates which mitigated his results; the 60% in this case is better than the 21.2% from the JANUS recogniser although Vo includes out of grammar utterances, and has more functionality and a larger vocabulary size of approximately 500 words compared to 300 in this project. However, Vo basically used a scripted protocol with more tasks, while in this project the tasks are fewer but more open. The command accuracy represents the accuracy at which the recogniser correctly interprets the entire utterance; the word string need not be entirely recognised correctly, just the interpretation needs to match the transcription interpretation. The command Oracle accuracy with an N-Best list of ten shows that in some instances, the correct command interpretation is not the first ranked result, but can be found somewhere lower in the list. These were primarily *'zoom in'* to *'zoom there'* substitutions that have different interpretations.

6.2.2 Gesture Recognition Performance

The gesture recognition performance is summarised in Table 4. The gesture recogniser only recognises single strokes hence sentence accuracy is not a suitable metric. Symbol accuracy is quoted, but is almost equivalent to command accuracy since there are only a few sentences and symbol/word variations that do not affect the interpretation. It is almost equivalent, in that entire symbols could be substituted, e.g. 'M' and 'Z', which yield the same command interpretation; this does not occur in the data set.

Symbol accuracy (%)	77
Symbol Oracle accuracy with N-Best list (%)	82
Symbol accuracy, hypothetical (%)	92
Symbol Oracle accuracy, hypothetical with N-Best list (%)	96
Out of grammar rate (%)	4

Table 4: Gesture recognition performance

The symbol accuracy is poorer than expected. Upon inspection of the data, the majority of errors were due to the point recognition and path stretching algorithms. In these cases, users would slowly and deliberately gesture a point, causing four or five mouse move messages, rather than the two points from a quick point gesture. The path-stretching algorithm would then futilely expand these relatively few four or five points to form a full path of points. Improving the point detection algorithm to recognise this different realization of point gesture would theoretically yield the more respectable hypothetical symbol accuracy shown in the table above. It is interesting to note that, although a small data set, almost all symbols produced were single strokes, the one exception a fully drawn arrow with line and head in two strokes, without informing the user of this limitation. The out of grammar rate thus seems quite low; the out of grammar items were mostly fully drawn arrows, shown in Figure 13, not specified in the training set of the gesture recogniser. In these figures, the gesture drawing direction is from light to dark.

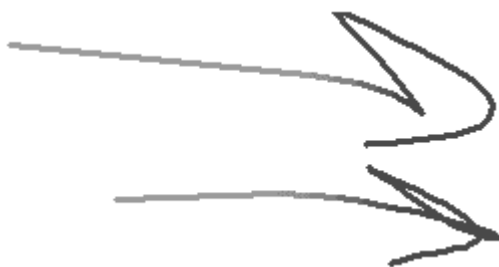


Figure 13: Unrecognised right arrows



Figure 14: Abstract references to a location

The N-Best list of three is useful in the gesture recogniser with region and other commands having some confusion especially in the small regions shown in Figure 14 that match poorly the training data. The gesture interpreter could be improved since it sometimes returns one or

two results instead of three and the recognition of a point is a hard decision instead of a soft probabilistic one. Situations where a gesture is ambiguous as to whether it is a line or point result in an N-Best list that contains either a point or a line, but not both; using an N-Best list cannot help in this case, but could if there is a soft decision between classifying a stroke as a point or other gesture.

6.2.3 Multimodal Recognition Performance

The gesture recognition performance is summarised in Table 5. The figures are adjusted after taking into account an error in the software that prevented the speech N-Best list from being used properly.

Multimodal accuracy, 1-Best (%)	68
Multimodal accuracy, N-Best (%)	74
Multimodal commands issued (%)	49
Out of grammar rate (%)	38

Table 5: Multimodal Recognition Performance

The multimodal accuracy is fair, given the speech recognition accuracy; the overall accuracy only degrades slightly with the combination of gesture. The use of the N-best list does improve the relative accuracy by almost 20% on this small data set. The N-Best list provided alternatives that were correct when the top ranked choice did not unify with other inputs, or did not parse in the multimodal grammar. One example was when the user said ‘*zoom in*’ but the top choice was ‘*zoom in there*’ due to breath noise. This does not parse in the grammar, but further down the N-best list, ‘*zoom in*’ can be found and does parse in the grammar, yielding a successful result. Another example from the data occurs when the user says ‘how do I get from here to there’ and circles two areas. The circles are like those shown in Figure 14 resulting in the gesture recogniser giving some other symbols like ‘P’ or ‘M’ the top result. However these do not unify with the frame from speech, but lower in the N-best list, a ‘region’ interpretation can be found that does unify. When the user was asked to use either speech or gesture about half the time (49%) turns were multimodal, otherwise they were entirely speech.

From a user standpoint, the system performs rather poorly due to all the out of grammar utterances (38%). It seemed the system was fairly good at rejecting of out of grammar turns.

The overall system out of grammar rate does not including system errors. Such system errors include speech-too-early due to lack of barge-in, some issues with the gesture recogniser returning zero for probability and confidence scores, and the speech recogniser rejecting utterances although the threshold is set to zero. With the system only recognizing at about a 75% rate, the system as a whole is only successfully doing what the user asks less than half the time.

Modality	Accuracy (%)	
	1-Best	N-best
Calculated multimodal	56	66
Empirical multimodal	68	74
Hypothetical multimodal	70	81

Table 6: Multimodal accuracy results and estimates

In Table 6 the multimodal accuracy is presented with some calculated hypothesised results. The multimodal accuracy is not directly comparable with the individual modality accuracy rates as they involve different actions; the multimodal turn is a composition of speech commands and gestures to spatial references. Thus the calculated multimodal accuracy estimates a lower bound of what the multimodal accuracy should be given independence of the individual recogniser accuracy rates computed in this way:

$$M = S \times G^{1.25}$$

where

M represents the multimodal accuracy,

S represents the speech recognition command accuracy, and

G represents the gesture recognition accuracy

The 1.25 arrives from the statistic that approximately 1.25 gestures are drawn every multimodal turn. This calculation is also an Oracle rate; idealistic in that it is assumed the correct interpretation can be extracted from the N-Best list, which is not always the case.

The empirical N-Best accuracy is higher than the calculated in both 1-Best and N-Best calculations assuming independence of modalities. One would expect the 1-Best result not to improve much, but the N-Best to improve results more. Perhaps this is due to the small sample size. It does show though that the use of N-Best lists and unification allowed the

system to improve recognition results by using only valid combinations of inputs. This indicates there is some redundant information, primarily interaction constraints, between the speech and gesture modes e.g. a stroke is constrained to be a spatial-locative gesture if the speech contains a command. The hypothetical multimodal calculation is similar to the calculated rate but uses the hypothetical gesture rates as opposed to the empirical. One would expect if the gesture recogniser were improved, that the overall system would exceed in performance the hypothetical multimodal N-best rate. This would bring the overall performance near to acceptable rates for general use.

6.2.4 Qualitative Results

Users were unanimous in their overall preference of interacting multimodally. One user had contradictory opinions that it was not complicated to use speech and gesture, but not natural or faster; this sentiment could be due to the many errors the system had. The other two users were more clear in their preference for using multimodal commands over just a single mode, finding it less difficult, more natural, faster and less complicated. Not surprisingly, all were somewhat confused at the beginning as to what they could say, contributing to the many out of grammar speech utterances.

About half the time users would either issue the gesture before the onset of speech. This seemed to differ across commands and users. One user tended to issue the gesture first, and then speak; another tended to gesture at the same time as speaking; the last one gestured first when saying '*zoom in*', but would gesture at the same time when using '*move this way*'. Overall, deictic references were said about two thirds of the time, and when uttered, there was more of a tendency to draw the gesture during the speech.

7 Discussion

This section discusses the overall multimodal architecture and the domain dependence of the approach taken, compares this work with other similar research, and outlines limitations in the methodology.

7.1 Framework Architecture

The application is quite distinct and separate from the Multimodal Input System. In general, as the design indicates, only grammars need to be written for the different modes and the `InputSvr`. The multimodal grammar is actually an application grammar that ensures that the final accepted command frame is acceptable to the application. There is some recogniser specific code in the `InputSvr`, but this could be generalised by moving it to the integration. Otherwise the `InputSvr` is quite independent to the application and recognition servers that interface it. This was exemplified, when towards the end of the project, when application functionality was being added, only modifications to the speech, gesture and multimodal grammars were necessary.

7.2 Comparison to Other Related Work

The overall accuracy rates for the speech and gesture are better than those in [Vo 1998]'s map application. This is likely due to a smaller overall command set, speech grammar size and gesture vocabulary. It is difficult to compare the interpretation accuracy i.e. combined multimodal accuracy as Vo calculates this with out of grammar utterances and only does error analysis on the transcribed interpretation accuracy. There is no calculation of accuracy with in grammar utterances and speech and gesture recognition errors.

Similarly, it is hard to compare directly with the work in [Wu et al. 1999] as the task is significantly different and accuracy rates of the different recognisers are unknown. The task in Wu appears more difficult with complex gestures and more functionality, although it's likely that Quickset is much more finely tuned than the system in this project. However the results are similar in that there are performance gains over the accuracy rates when assuming modality independence from some redundancy in the channels.

The usage of deictic references compares similarly to the findings of other research. All deictic utterances involved terms such as 'this', 'that', 'here' and 'there'. There were no instances of redundancy where users would gesture at a location and say its name. This can

be compared to [Oviatt et al. 1997] which had similar findings that most (96%) deictic references were of this spatial locative type and that the gesture was drawn before the deictic reference was spoken 43% of the time, but found most multimodal utterances did not have a spoken deictic (59%) where the inverse (33%) was found in this project. This difference could be due to the evaluation protocol, different commands available and the small sample size.

7.3 Limitations

The project as designed has several limitations besides the limited implementation. The multimodal grammar cannot be easily extended for recursion to elegantly handle multiple references e.g. *'how do I get from here to here to here to...'* which one user asked. The use of subcategorisation lists in [Johnston 1998] can manage these sorts of commands. Another limitation is that the algorithm is tailored to a turn taking dialogue, and continuous dictation would not be possible with the input acquisition and frame unification steps tied together as they are currently. Lastly, the small sample size also limits the extent to which conclusions can be made from the evaluation.

8 Conclusions

This project demonstrated the development of a framework to fuse multimodal inputs, a simple multimodal interactive tour guide called iTour to demonstrate the architecture, and a small evaluation of the overall system.

The multimodal framework developed is theoretically recogniser agnostic although in practice this may not be the case as the specified interface may not be sufficient for the variety of possible input modes. The Nuance speech recogniser performed adequately with little tuning although the audio captured was very poor due mainly to poor microphone placement. The improvements made to the gesture recogniser to deal with spatial gestures worked fairly well as demonstrated by the recognition performance in the evaluation although further enhancements could be made.

The unification-style algorithm used in conjunction with N-Best lists was a viable fusion method. It improved performance over the 1-Best case, and performs comparably to the results in [Vo 1998] and [Wu 1999]. It was found that the use of N-Best lists and unification allowed the system to improve recognition results, over the lower bound, by using only valid combinations of inputs. Also, users appreciated the naturalness speed and style of multimodal interaction. However the major limitations in this evaluation are the narrow application functionality and small sample size. With few testers the observations hold little weight and with the simple command set, the task complexity is reduced making it easier for the various recognition processes and the evaluation less realistic. Nevertheless, the system produced provides a good foundation for further enhancements and research to be conducted upon.

9 Future Work and Research

This project was carried out in a rather short and limited time frame. With more time, many improvements could have been made. A better gesture recogniser could have been used, or the existing software could be improved through faster sampling of the pointer and better recognition of spatial entities like points, fully drawn arrows and small scribbled regions. Also the tablet digitiser itself was rather insensitive which caused a few issues with gesturing. The speech recognition software could be tuned further, and the grammar improved. The most gains would be had by improving the audio quality by possibly changing microphones, adjusting the microphone placement, audio sampling rate, and speech detection parameters to better suit the acoustic models and limit breath noise. Users noted system response was slow; the latency could be improved through tuning of the timing parameters or perhaps adding grammar feedback to end the timer faster if the current set of collected inputs yields a complete command. Formalisation of the interface between the `InputSvr` to other recognisers could make it easier to add more or change them.

The overall framework could be modified to accept grammar files for the gesture and multimodal grammars and hardcoded parameters changed to runtime settable variables found in files or the registry. Furthermore, The burden falls on the developer to properly enumerate and use attributes from the other grammars, although a tool that does consistency checking on the set of attributes used by the multimodal grammar against those found in the various recogniser grammars could be developed. It may be difficult though to robustly extract attribute names from a variety of different grammar and interpreter formats.

On the application and evaluation side, it would be useful to expand the command set to match Vo's to have a richer and more varied interaction, and to run the evaluation on more users. Only a limited set of gestures and phrases were used; more functions and users would test the fusion algorithm more thoroughly. Overall the application would benefit from better dialogue management and error recovery. Some interesting strategies could be employed to fill high confidence, incomplete frames like forms or prompt users more specifically on how to correct errors.

Lastly, tools could be developed to help transcribe the multimodal turns and replay them. A step further would be to have the ability to rerun data through the framework so that multimodal experiments can be conducted to test out different fusion algorithms. This would

likely require changes to the logging procedure as implemented. Also, besides increasing the user set size and application functionality, additional metrics could be calculated such as task completion rate and task completion time compared to unimodal interactions. It would be useful to spend more time collecting more data with a more finely tuned system, more users and more specific tasks.

Appendix A – Evaluation Protocol

The following Instructions, Task List, and Questionnaire pages were given to users complete to form the evaluation method of this system. These forms were modified slightly from the testing to reflect improvements needed for clarity suggested in the actual evaluation. The original filled forms from the evaluation were scanned and are on the companion CD.

Instructions

Introduction

This is a simple interactive tourist guide application. Pretend you are a new tourist to Bristol, and need some ideas to tour around this up and coming Western England city. You can use this system by talking to it and using the pen to point and gesture on the touch screen.

Say something, and use the screen at the same time! For example,

'Zoom in', 'Zoom in to here'
'Move right', 'Move this way'

'Where is Berkeley Square?'
'How far is it from here to there?'
'How do I get from there to there?'

Read this or skip on and fill in the blanks! This is a system built for an M.Phil thesis project called 'Multimodal Fusion' by Hank Liao. Multimodal implies that you can interact with the system using different modes such as talking to it and drawing on the screen. It is fusion in that you can do both at the same time and the system should fuse the information you give appropriately.

Some Information

Date and time: _____

Your initials please: _____

What city and country are you originally from?

Are you a native speaker of English? YES NO

Where did you grow up (learn) your English?

Canada	USA	London	Scotland
Southern England	Northern England	Other	

NOTES

Task List

You will be told to the following four tasks, two different times for each condition. The order you should do them in is:

1. _____
2. _____

Task 1

You are at **Berkeley Square** in the **West End**. *Enlarge* the map to find the name of the attraction south of (below) you?

How *far* is it from you?

Task 2

Zoom out and locate the **railway station** in the southeast, bottom right of the map. *Enlarge* the map in that area.

Pan through the map from the **railway station** to the **West End**.

Task 3

Zoom into Park Street in the West End by speaking the command and circling the area.

Look at the names of the restaurants along the route of **Park Street** down south to **Anchor Road** and then to west along **Hotwell Road** all the way to the bridge.



marks a restaurant

Task 4

Find the attractions near **Clifton Village** and get the map to display a path to visit the three.



marks an attraction.

Say 'Exit' when you are done the four tasks of one condition. Please repeat this again for the other conditions. You need not fill in the information again.

Appendix B – Companion CD Contents

The CD accompanying this thesis contains the source code for the various components in this project, grammar files, recorded log data, prompts, and executable applications. The directories are:

```
{TORONTO: /hliao/Multimodal/}$ ls
GestureApp/ - main gesture recogniser and map application
GestureTrans/ - viewer for gestures recorded in gesture log files
InputSvr/ - Multimodal Fusion server
Shared/ - shared source code and headers
SpeechSvr/ - speech integration
TestSvr/ - test application to send messages to other components
grammar/ - speech grammar, use compile.bat to compile
log/ - logged data from evaluation
prompts/ - application audio prompt files, with listing prompts.txt
release/ - location of executable files
```

To execute the application on a Windows 2000 machine with Nuance System 7.0.4 or 8 installed, the `release` and `prompts` directories need to be copied to the C: drive with the same directory structure as on the CD. Some paths are hard-coded in the source code. The `release/readme.txt` outlines the steps to start the application:

1. Run `go.basic.bat`
2. Run `go.speech.svr.uk` or `go.speech.svr.us` (depending on accent)
3. Run `InputSvr.exe`
4. Run `GestureApp.exe`

References

- K. Boukreev. 2001. Mouse Gestures Recognition. Available at:
<http://www.generation5.org/aisolutions/gestureapp.shtml> generation5.org
- S. Bangalore and M. Johnston. 2000. Fusing Multimodal Language Processing with Speech Recognition. In *Proceedings of International Conference on Spoken Language Processing, Beijing, China*.
- A. Copestake and E.J. Briscoe. 2002. *Computer Speech, Text and Internet Technology Speech Language II Lecture Notes*. University of Cambridge.
- M. Johnston and S. Bangalore. 2001. Finite-state methods for multimodal parsing and fusion. In *Finite-state Methods Workshop, ESSLLI Summer School on Logic Language and Information*, Helsinki, Finland.
- M. Johnston. 1998. Unification-based Multimodal Parsing. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, (COLING-ACL 98)*, pp. 624-630.
- M. Johnston, P. R. Cohen, D. McGee, S.L. Oviatt, J.A. Pittman, and I. Smith. 1997. Unification-based multimodal fusion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 281-288.
- S. Kettebekov and R. Sharma. 2000. Understanding Gestures in Multimodal Human Computer Interaction. In *International Journal of Artificial Intelligence Tools, September 2000*.
- Nuance Communications, Inc. 2001. Nuance Speech Recognition System, Version 7.0, Introduction to the Nuance System. Menlo Park, CA.
- S.L. Oviatt, A. DeAngeli, and K. Kuhn. 1997. Integration and Synchronization of Input Modes During Multimodal Human-Computer Interaction. In *Proceedings of ACM Conference on Human Factors in Computing Systems, (CHI'97)*, pp. 415-422.
- S.L. Oviatt, P.R. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro. 2000. Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions. In *Human Computer Interaction, vol. 15, no. 4*, pp. 263-322.
- D.H. Rubine. 1991. The Automatic Recognition of Gestures. PhD Thesis CMU-CS-91-202, Carnegie Mellon University, Pittsburgh, PA, USA.
- M.T. Vo, and C. Wood. 1996. Building an Application Framework for Speech and Pen Input Fusion in Multimodal Learning Interfaces. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'96), Vol. 6*, pp. 3545-3548.

M.T. Vo. 1998. A Framework and Toolkit for the Construction of Multimodal Learning Interfaces. PhD Thesis CMU-CS-98-129, Carnegie Mellon University, Pittsburgh, PA, USA.

L. Wu, S.L. Oviatt, P.R. Cohen. 1999. Multimodal Fusion -- A Statistical View. In *IEEE Transactions on Multimedia*, Vol. 1, No. 4, pp. 334-341.