Robust Gesture Processing for Multimodal Interaction

Srinivas Bangalore
AT&T Labs Research
180 Park Ave
Florham Park, NJ 07932
srini@research.att.com

Michael Johnston
AT&T Labs Research
180 Park Ave
Florham Park, NJ 07932
johnston@research.att.com

ABSTRACT

With the explosive growth in mobile computing and communication over the past few years, it is possible to access almost any information from virtually anywhere. However, the efficiency and effectiveness of this interaction is severely limited by the inherent characteristics of mobile devices, including small screen size and the lack of a viable keyboard or mouse. This paper concerns the use of multimodal language processing techniques to enable interfaces combining speech and gesture input that overcome these limitations. Specifically we focus on robust processing of pen gesture inputs in a local search application and demonstrate that edit-based techniques that have proven effective in spoken language processing can also be used to overcome unexpected or errorful gesture input. We also examine the use of a bottom-up gesture aggregation technique to improve the coverage of multimodal understanding.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation (e.g. HCI)**]: User Interfaces—*input devices and strategies (e.g. mouse, touchscreen),natural language*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*language parsing and understanding*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

finite-state methods, local search, mobile, multimodal interfaces, robustness, speech-gesture integration

1. INTRODUCTION

With the explosive growth of wireless communication networks in the past few years and advances in the capabilities of mobile computing devices, it is now possible to access almost any information from virtually everywhere. However, the *efficiency* and utility of mobile information access is still severely constrained by the user interfaces of mobile devices. With small keyboards and limited screen real estate it quickly becomes cumbersome to maintain the same established techniques used in non-mobile human-computer interaction. The efficiency of human interaction with mobile devices can be dramatically improved when their user interfaces are augmented with natural modalities such as sketched pen input [20], touch input, and speech. These interfaces are most effective when they support multiple different modalities of input [3, 23, 21, 7, 11, 22, 1].

In this paper, we focus on techniques that enable multimodal interaction in the context of local search applications on mobile devices supporting speech, pen, and touch input. Multimodal interfaces are most effective if, in addition to supporting input in one modality or another, they allow users to combine multiple modalities in a single turn of interaction. For example, allowing a user to issue a command using both speech and pen modalities simultaneously. This kind of multimodal interaction requires integration and understanding of information distributed in the two modalities.

We have been investigating methods that achieve such multimodal integration and understanding over the past several years [17, 18]. In recent work, we have emphasized the issue of *robust multimodal understanding* to support practical applications [4]. The primary focus in that work, however, has been to provide robustness to speech recognition errors. In this paper, we focus on techniques to provide robustness to gesture recognition errors and highlight an extension of these techniques to *gesture aggregation*, where multiple pen gestures are interpreted as a single conceptual gesture for the purposes of multimodal integration and understanding.

The outline of the paper is as follows. Section 2, illustrates the use of multimodal interfaces for local search with examples from our prototype system. Section 3 summarizes the approach taken for multimodal language processing and Section 4 describes edit-based techniques for robust speech processing. Section 5 explores the application of these techniques to robust pen gesture processing and extensions of these techniques for gesture aggregation. Section 6 presents an experimental evaluation of the effectiveness of edit-based techniques for pen gesture processing and Section 7 concludes the paper.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICMI'08, October 20–22, 2008, Chania, Crete, Greece.
Copyright 2008 ACM 978-1-60558-198-9/08/10 ...$5.00.

2. MULTIMODAL INTERFACES FOR LOCAL SEARCH

In the modern world, whether traveling or going about their daily business, users need to access a complex and constantly changing body of information regarding restaurants, shopping, cinema and theater schedules, transportation options and timetables. This information is most valuable if it can be delivered while mobile, since users' plans change while they are mobile and the information itself is highly dynamic (e.g. train and flight timetables change, shows get cancelled, and restaurants get booked up).

In this paper, we will illustrate and evaluate our approach to multimodal information access using data and examples from the MATCH (Multimodal Access To City Help) system, a city guide and navigation system that enables mobile users to access restaurant and subway information for urban centers such as New York City and Washington, D.C. [19, 5]. However, the techniques described apply to a broad range of mobile information access and management applications beyond this particular task domain, such as apartment finding [15], setting up and interacting with map-based distributed simulations [10] or searching for hotels [8].

In MATCH, the user interacts with a graphical interface displaying restaurant listings and a dynamic map showing locations and street information. The inputs can be speech, drawings on the display with a stylus, or synchronous multimodal combinations of the two modes. The user can ask for the review, cuisine, phone number, address, or other information about restaurants and for subway directions to locations. The system responds by generating multimodal presentations synchronizing graphical callouts and animation, with synthetic speech output.

For example, a user can request to see restaurants using the spoken command *show cheap italian restaurants in chelsea*. The system will then zoom to the appropriate map location and show the locations of restaurants on the map. Alternatively, the user could give the same command multimodally by circling an area on the map and saying *show cheap italian restaurants in this neighborhood*. If the immediate environment is too noisy or public, the same command can be given completely using a pen stylus as in Figure 1(a), by circling an area and writing *cheap* and *italian*.

Similarly, if the user says *phone numbers for these two restaurants* and circles two restaurants as in Figure 1(b) [A], the system will draw a callout with the restaurant name and number and say, for example *Time Cafe can be reached at 212-533-7000*, for each restaurant in turn (Figure 1(b) [B]). If the immediate environment is too noisy or public, the same command can be given completely in pen by circling the restaurants and writing *phone* (Figure 1(c)).

3. MULTIMODAL PROCESSING

In our system, multimodal integration and understanding is performed by a single component (MMFST) which takes as input a word lattice from speech recognition and/or a gesture lattice which is a combination of results from handwriting recognition and gesture recognition (Figure 2). This component uses a cascade of finite state operations [17, 19] to align and integrate the content in the word and gesture lattices and produces as output a meaning lattice which is passed on to a multimodal dialog manager for further processing.

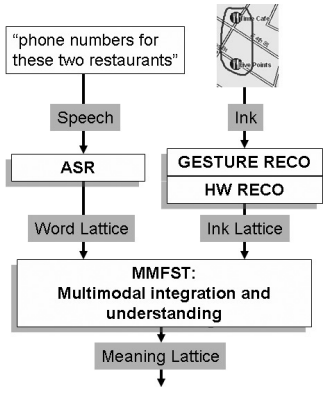


Figure 2: Multimodal understanding component

In the example above where the user says *phone for these two restaurants* while circling two restaurants (Figure 1(a) [A]), assume the speech recognizer returns the word lattice in Figure 3 (Speech). The gesture recognition component also returns a lattice (Figure 3, Gesture) indicating that the user's ink is either a selection of two restaurants or a geographical area. The multimodal integration and understanding component (MMFST) combines these two input lattices into a lattice representing their combined meaning, Figure 3 (Meaning). This is passed to a multimodal dialog manager and from there back to the user interface where it results in the display in Figure 1(a) [B] and coordinated TTS output.

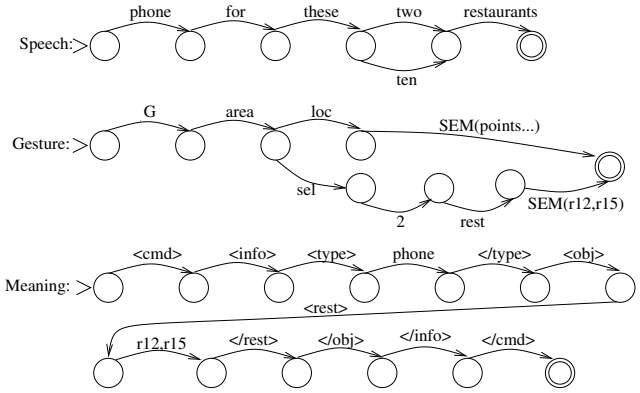
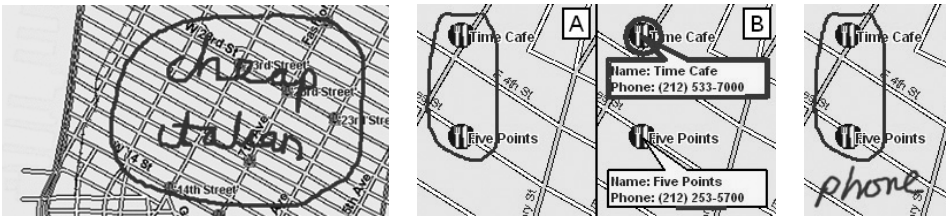


Figure 3: Multimodal example

The alignment of speech and gesture and relation to their combined meaning is captured in a single declarative multimodal grammar representation [17, 18]. The non-terminals of the multimodal grammar are atomic symbols but each terminal contains three components W:G:M corresponding to the n input streams and one output stream, where W is for the spoken language input stream, G is for the gesture input stream, and M is for the combined meaning output stream. The epsilon symbol (ϵ) is used to indicate when one of these streams is empty within a given terminal. In addition to the gesture symbols (*G area loc ...*), G contains a symbol *SEM* used as a placeholder for specific content. Figure 4 contains a small fragment of the multimodal grammar used for MATCH, which includes coverage for commands such as that in Figure 3.



(a) (b) (c)

Figure 1: (a): Unimodal pen command (b): Two area gestures (c): Phone command in pen

The multimodal grammar can be compiled into a finite-state device operating over two input streams (speech and gesture) and one output stream (meaning). The symbols on the transition arcs of the FSA correspond to the terminals of the multimodal grammar. For the sake of illustration here and in the following examples we will only show the portion of the three tape finite-state device which corresponds to the *DEICNP* rule in the grammar in Figure 4. The corresponding finite-state device is shown in Figure 5. This three tape machine is then factored into two transducers: $\mathcal{R}$:$G \rightarrow W$ and $\mathcal{T}$:$(G \times W) \rightarrow M$. $\mathcal{R}$:$G \rightarrow W$ (e.g. Figure 6) aligns the speech and gesture streams through a composition with the speech and gesture input lattices (G o (G:W o W)). The result of this operation is then factored onto a single tape and composed with $\mathcal{T}$:$(G \times W) \rightarrow M$ (e.g. Figure 7) resulting in a transducer G_W:M. Essentially the three tape transducer is simulated by increasing the alphabet size by adding composite multimodal symbols that include both gesture and speech information. A lattice of possible meanings is derived by projecting on the output of G_W:M. For detailed description of this approach see [18] and [5].

4. ROBUST SPEECH PROCESSING USING EDIT MACHINES

Like other grammar-based approaches, multimodal language processing based on declarative grammars can be brittle with respect to unexpected or errorful inputs. On the speech side, the brittleness of using a grammar as a language model for recognition can be alleviated by instead building statistical language models (SLMs) that capture the distribution of the user's interactions in an application domain. However, to be effective SLMs need to be trained on large amounts of spoken interactions collected in that domain – a tedious task in itself, in speech-only systems but an often insurmountable task in multimodal systems. The problem we face is how to make multimodal systems more robust to disfluent or unexpected spoken language in applications for which little or no training data is available. In [4, 5], we show how this challenge can be addressed using a number of techniques including sampling data from the multimodal grammar, and adaptation of data from different domains in order to build SLMs for the speech recognizer (ASR).

The second source of brittleness in a grammar-based multimodal/unimodal interactive system is in the assignment of meaning to the multimodal output. In our grammar-based multimodal system, the grammar serves as the speech-gesture alignment model and assigns a meaning representation to the multimodal input. Failure to parse a multimodal input implies that the speech and gesture inputs could not be fused together and consequently could not be assigned a meaning representation. This can result from unexpected or errorful strings in either the speech or gesture of input or unexpected alignments of speech and gesture. For more robust multimodal understanding, more flexible mechanisms must be employed in the integration and the meaning assignment phases. Robustness in such cases is achieved by either (a) modifying the parser to accommodate for unparsable substrings in the input [12, 2, 24] or (b) modifying the meaning representation it can be learnt as a classification task using robust machine learning techniques as is done in large scale human-machine dialog systems (e.g. [14]).

In [4, 5] we describe a technique that overcomes unexpected inputs or errors in the speech input stream with the finite-state multimodal language processing framework and does not require training data. The basic idea is that if the ASR output cannot be assigned a meaning then to transform it into the *closest* sentence that can be assigned a meaning by the grammar. The transformation is achieved using edit

S	→	ϵ:ϵ:<cmd> CMD ϵ:ϵ:</cmd>
CMD	→	ϵ:ϵ:<show> SHOW ϵ:ϵ:</show>
CMD	→	ϵ:ϵ:<info> INFO ϵ:ϵ:</info>
SHOW	→	show:ϵ:ϵ ϵ:ϵ:<rest> ϵ:ϵ:<cuis> CUISINE ϵ:ϵ:</cuis> restaurants:ϵ:ϵ (ϵ:ϵ:<loc> LOCPP ϵ:ϵ:</loc>) ϵ:ϵ:</rest>
INFO	→	ϵ:ϵ:<type> TYPE ϵ:ϵ:<type> for:ϵ:ϵ ϵ:ϵ:<obj> DEICNP ϵ:ϵ:</obj>
CUISINE	→	italian:ϵ:italian | chinese:ϵ:chinese | new:ϵ:ϵ american:ϵ:new_american ...
LOCPP	→	in:ϵ:ϵ LOCNP
LOCPP	→	here:G:ϵ ϵ:area:ϵ ϵ:loc:ϵ ϵ:SEM:SEM
LOCNP	→	ϵ:ϵ:<zone> ZONE ϵ:ϵ:</zone>
ZONE	→	chelsea:ϵ:chelsea | soho:ϵ:soho | tribeca:ϵ:tribeca ...
TYPE	→	phone:ϵ:ϵ numbers:ϵ:phone | review:ϵ:review | address:ϵ:address
DEICNP	→	DDETSG ϵ:area:ϵ ϵ:sel:ϵ ϵ:1:ϵ HEADSG
DEICNP	→	DDETPL ϵ:area:ϵ ϵ:sel:ϵ NUMPL HEADPL
DDETPL	→	these:G:ϵ | those:G:ϵ
DDETSG	→	this:G:ϵ | that:G:ϵ
HEADSG	→	restaurant:rest:<rest> ϵ:SEM:SEM ϵ:ϵ:</rest>
HEADPL	→	restaurants:rest:<rest> ϵ:SEM:SEM ϵ:ϵ:</rest>
NUMPL	→	two:2:ϵ | three:3:ϵ ... ten:10:ϵ

Figure 4: Multimodal grammar fragment

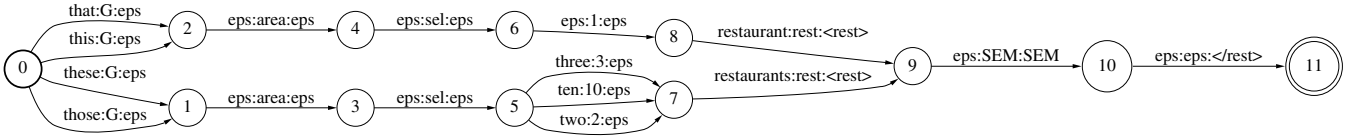


Figure 5: Multimodal three-tape FSA

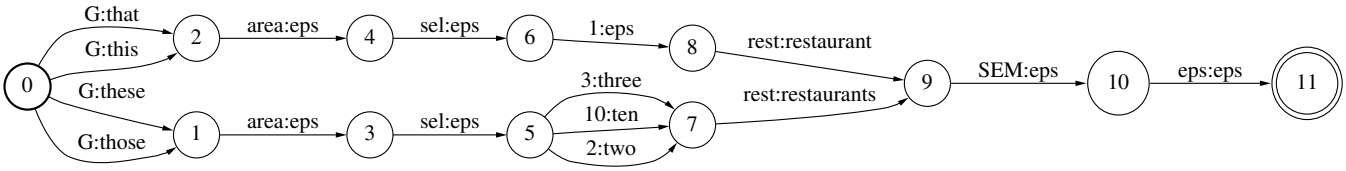


Figure 6: Gesture/speech alignment transducer

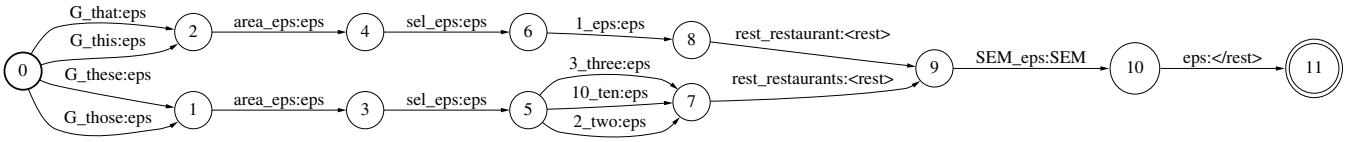


Figure 7: Gesture/speech to meaning transducer

operations such as substitution, deletion and insertion of words. The possible edits on the ASR output are encoded as an edit finite-state transducer (FST) with substitution, insertion, deletion and identity arcs and incorporated into the sequence of finite-state operations. These operations could be either word-based or phone-based and are associated with a cost. The word-edit transducer (λ_{edit}^{W}) coerces the set of strings ($\mathcal{S}$) encoded in the lattice resulting from ASR ($\lambda_{\mathcal{S}}$) to the closest strings in the speech component of the multimodal grammar (λ_W) that can be assigned an interpretation. We are interested in the string with the least cost sequence of edits ($argmin$) that can be assigned an interpretation by the grammar. This can be achieved by composition ($\circ$) of transducers followed by a search for the least cost path through a weighted transducer as shown below.

$$s^* = \underset{s \in \mathcal{S}}{argmin}\ \lambda_{\mathcal{S}} \circ \lambda_{edit}^{W} \circ \lambda_W \qquad (1)$$

As an example in this domain the ASR output *find me cheap restaurants thai restaurants in the the upper east side* might be mapped to *find me cheap thai restaurants in the upper east side*. The edit machine described in [4] is essentially a finite-state implementation of the algorithm to compute the Levenshtein distance. It allows for unlimited insertion, deletion, and substitution of any word for another (Figure 8). The costs of insertion, deletion, and substitution are set as equal, except for members of classes such as price (*expensive*), cuisine (*turkish*) etc., which are assigned a higher cost for deletion and substitution. Membership of these high cost classes of words is determined automatically from the underlying application database. [5] presents some variants to this basic edit FST that are computationally more attractive for use on ASR lattices. One such variant limits the number of edits allowed on an ASR output to a predefined number based on the application domain. A second variant has more fine tuning of costs based on the application database.

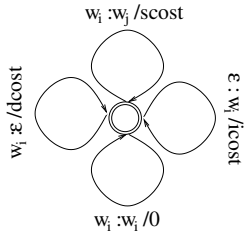


Figure 8: Basic edit machine

5. ROBUST GESTURE PROCESSING

For applications such as the local search task described here recognition of pen gestures generally has lower error rate than speech recognition given smaller vocabulary size and less sensitivity to extraneous noise[1]. Even so, gesture misrecognitions and incompleteness of the multimodal grammar in specifying speech and gesture alignments contribute to the number of utterances not being assigned a meaning. In this section we explore techniques for overcoming unexpected or errorful gesture input streams.

The edit-based technique used on speech utterances proved to be effective in improving the robustness of multimodal understanding. However, unlike a speech utterance, which is represented simply as a sequence of words, gesture strings are represented using a structured representation which captures various different properties of the gesture. The basic form of this representation is: *G FORM MEANING (NUMBER TYPE) SEM. FORM* indicates the physical form of the gesture, and has values such as *area*, *point*, *line*, and *arrow*. *MEANING* provides a rough characterization of the specific meaning of that form; for example, an *area* can be

[1]Note however that in applications involving freehand sketching [1] or 3D gesture input using computer vision gesture recognition is considerably more complex and may be as or more prone to errors than speech.

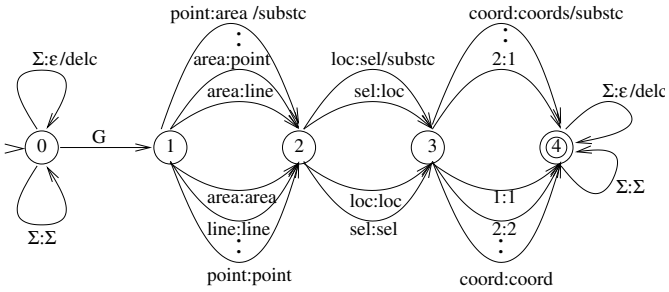


Figure 9: A finite-state transducer for editing gestures

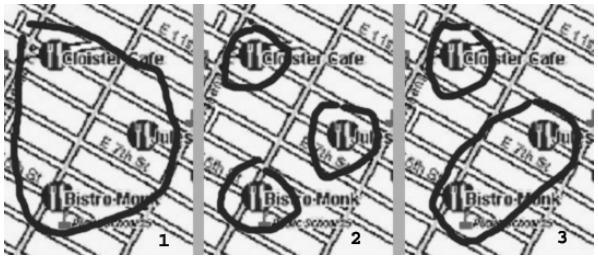


Figure 10: Multiple ways to select three restaurants

either a *loc* (location) or a *sel* (selection), indicating the difference between gestures which delimit a spatial location on the screen and gestures which select specific displayed icons. *NUMBER* and *TYPE* are only found with *sel*. They indicate the number of entities selected (*1,2,3, many*) and the specific type of entity (e.g. *rest* (restaurant) or *thtr* (theatre)). Since the gesture is represented as a lattice rather than a single sequence of symbols, imprecise gestures where it is not clear which of several entities are selected can be represented as multiple different paths through the lattice. Likelihood of each possible selection is captured using relative costs on each path, based on factors such as distance from the gesture to the selected item.

Editing a gesture representation implies allowing for replacements within the value set. We initially adopt a simple approach that allows for substitution and deletion of values for each attribute, in addition to the deletion of any gesture. We did not allow for general insertion of gestures since it is not clear which content to attribute to a newly inserted gesture. One of the problems is that if you have, for example, a selection of two items and you want to increase it to three selected items it is not clear a priori which entity to add as the third item.

As in the case of speech, the edit operations for gesture editing can be encoded as a finite-state transducer (Figure 9). In this gesture edit transducer *delc* represents the delection cost and *substc* the substitution cost. The costs used in the gesture edit transducer can be learned from data or hand crafted based on knowledge of the application. In our initial implementation and evaluation the assigned costs are equal. In ongoing work we are exploring the use of other features of the gesture and context to set costs for edit operations. For example, substituting a line with a point should be sensitive to the number of points that make up the line, with lines consisting of very few points having a lower cost for substitution with a point. Note also that the combination with speech plays an important role in which edits are applied since the speech input will only align with certain paths in the gesture lattice.

Table 1 is an illustrative example of the role of gesture editing in overcoming errors. In this case the user has drawn an area which has been misrecognized as a line. Also, after the area, there was a spurious pen tap which has been recognized as a point gesture. The gesture edit transducer allows for substitution of *line* with *area* and for deletion of the spurious point gesture (among other possible edits). In this case the speech is *chinese restaurants here* which the multimodal grammar aligns with an area gesture and so an edit path is chosen where the line is substituted with area and the point deleted.

Speech	chinese restaurants here
Gesture	G line loc SEM G point loc SEM
Possible gesture after editing	G area loc SEM

Table 1: Gesture editing

5.1 Insertion of Gestures through Aggregation

One kind of gesture editing that supports insertion is gesture aggregation, introduced in [19] and expanded on here. Gesture aggregation allows for insertion of paths in the gesture lattice which correspond to combinations of existing temporally adjacent gestures. These insertions are possible because they have a well defined meaning based on the values of the combining gestures. These insertions allow for alignment and integration of deictic expressions with sequences of gestures which are not specified in the multimodal grammar. This overcomes problems identified in [16] regarding multimodal understanding and integration of deictic numeral expressions such as *these three restaurants*. The problem is that for a particular spoken phrase there are a multitude of different lexical choices of gesture and combinations of gestures that can be used to select the specified plurality of entities (e.g. three) and all of these need to be integrated with the spoken phrase. For example, as illustrated in Figure 10, the user might circle all three restaurants with a single pen stroke, circle each in turn, or circle a group of two and group of one. The split into a group of two and a group of one is most likely if some natural feature intervenes (a road or river) or if there is some other entity inbetween which the user does not wish to select.

In our implementation, gesture aggregation serves as a bottom-up pre-processing phase on the gesture input lattice. A gesture aggregation algorithm traverses the gesture input lattice and adds new sequences of arcs which represent combinations of adjacent gestures of identical type. The operation of the gesture aggregation algorithm is described in pseudo-code in Algorithm 1 below. The function $plurality()$ retrieves the number of entities in a selection gesture, for example, for a selection of two entities $g1$, $plurality(g1) = 2$. The function $type()$ yields the type of the gesture, for example *rest* for a restaurant selection gesture. The function $specific\_content()$ yields the specific IDs.

Essentially what this algorithm does is perform closure on the gesture lattice of a function which combines adjacent gestures of identical type. For each pair of adjacent gestures in the lattice which are of identical type, a new gesture is added to lattice. This new gesture starts at start state of the first gesture and ends at the end state of the second gesture. Its plurality is equal to the sum of the pluralities

of the combining gestures. The specific content for the new gesture (lists of identifiers of selected objects) results from appending the specific contents of the two combining gestures. This operation feeds itself so that sequences of more than two gestures of identical type can be combined.

Algorithm 1 Gesture aggregation

$P \Leftarrow$ the list of all paths through the gesture lattice GL
while $P \neq \emptyset$ **do**
  $p \Leftarrow pop(P)$
  $G \Leftarrow$the list of gestures in path p
  $i \Leftarrow 1$
  while $i < length(G)$ **do**
    if $g[i]$ and $g[i+1]$ are both selection gestures **then**
      if $type(g[i]) == type(g[i+1])$ **then**
        $plurality \Leftarrow plurality(g[i]) + plurality(g[i+1])$
        $start \Leftarrow start\_state(g[i])$
        $end \Leftarrow end\_state(g[i+1])$
        $type \Leftarrow type(g[i])$
        $specific \Leftarrow append(specific\_content(g[i]),$
                $specific\_content(g[i+1])$
        $g' \Leftarrow$ G area sel *plurality type specific*
        Add g' to GL starting at state *start* and ending at state *end*
        $p' \Leftarrow$ the path p but with the arcs from *start* to *end* replaced with g'
        push p' onto P
        $i \Leftarrow i + 1$
      end if
    end if
  end while
end while

For the example of three selection gestures on single restaurants as in Figure 10(2), the gesture lattice before aggregation is as in Figure 11(a). After aggregation the gesture lattice is as in Figure 11(b). Three new sequences of arcs have been added. The first from state 3 to state 8 results from the combination of the first two gestures. The second from state 14 to state 24 from the combination of the last two gestures, and the third from state 3 to state 24 from the combination of all three gestures. The resulting lattice after the gesture aggregation algorithm has applied is shown in Figure 11(b). Note that minimization has been applied to collapse identical paths.

A spoken expression such as *these three restaurants* is aligned with the gesture symbol sequence *G area sel 3 rest SEM* in the multimodal grammar. This will be able to combine not just with a single gesture containing three restaurants but also with our example gesture lattice, since aggregation adds the path: *G area sel 3 rest [id1,id2,id3].*

We term this kind of aggregation *type specific aggregation*. The aggregation process can be extended to support *type non-specific aggregation* to support cases where users refer to sets of objects of mixed types and select them using multiple gestures. For example in the case where the user says *tell me about these two* and circles a restaurant and then a theatre, *non-type specific aggregation* applies to combine the two gestures into an aggregate of mixed type *G area sel 2 mix [(id1,id2)]* and this is able to combine with *these two.* For applications with a richer ontology with multiple levels of hierarchy, the *type non-specific aggregation* should assign the aggregate to the lowest common subtype of the set of entities being aggregated. In order to differentiate the original sequence of gestures that the user made from the aggregate, paths added through aggregation can be assigned additional cost.

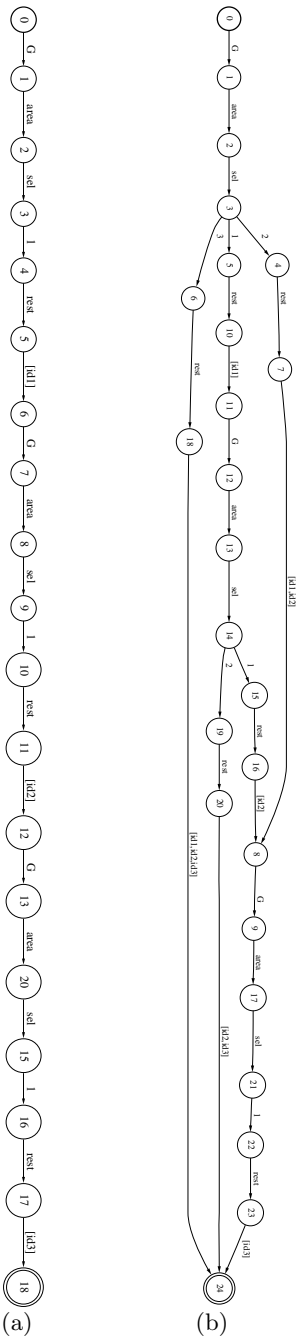


Figure 11: Aggregated lattice

6. DATA AND EXPERIMENTS

To evaluate the approach, a corpus of multimodal data was collected in a laboratory setting from a gender balanced set of sixteen first time users. The subjects were AT&T personnel with no prior knowledge of the system and no experience building spoken or multimodal systems. A total of

833 user interactions (218 multimodal / 491 speech-only / 124 pen-only) resulting from six sample task scenarios were collected and annotated. The subjects completed a series of six sample task scenarios of varying complexity. These involved finding restaurants of various types and getting their names, numbers, addresses, or reviews, and getting subway directions between locations. Users were not prompted to use particular modalities or commands and so the choice of modality reflects a natural distribution of this task domain. Their inputs were then annotated for speech, gesture, and meaning using a multimodal log annotation tool [13].

For evaluation of our approach we focus on *concept accuracy*. We developed an approach, similar to [9, 6], in which the meaning representation, in our case XML, is transformed into a *sorted* flat list of attribute-value pairs indicating the core contentful concepts of each command. The *attribute-value* meaning representation normalizes over multiple different XML representations which correspond to the same underlying meaning. For example, *phone and address* and *address and phone* receive different XML representations but the same attribute-value representation. For the example *phone number of this restaurant*, the XML representation and corresponding sorted attribute value representation (A/V) are as in Table 2. *Concept Sentence Accuracy* measures the number of user inputs for which the system got the meaning completely right (This metric is called *Sentence Understanding* in Ciaramella [9]).

XML	<cmd><info><type>phone</type><obj> <rest>[r12,r15]</rest></obj></info></cmd>
A/V	`cmd : info type : phone object : selection`

Table 2: Attribute value representation

We conducted two experiments evaluating the effectiveness of the gesture edit machine on this corpus. We first examined the impact of gesture editing on accuracy specifically within the subset of the data which is multimodal. In order to factor out the impact of speech errors, in this first experiment, we leave out examples where the speech reference string is out of grammar and use the transcribed string along with the gesture lattice from the gesture recognizer as inputs to the multimodal integration and understanding system. This dataset consisted of 174 multimodal utterances that were covered by the speech grammar. For 55.4% of the utterances, we obtained the identical attribute-value meaning representation as the human transcribed meaning representation. Applying the gesture edit transducer on the gesture recognition lattices, and then integrating the result with the transcribed speech utterance, produced a significant improvement in the accuracy of the attribute-value meaning representation. For 68.9% of the 174 multimodal utterances, we obtained the identical attribute-value meaning representation as the human transcribed meaning representation, a 22.5% relative improvement in the robustness of the system that can be directly attributed to robustness in gesture integration and understanding.

In our next experiment, we examined the impact of gesture editing on the performance of the system as a whole, where both the speech and gesture edits are allowed and we operate on the recognized speech rather than the transcription. In Equation 2, we formulate the process of editing the speech and gesture lattices. The speech lattice ($\lambda_{\mathcal{S}}$) encodes the set of strings $\mathcal{S}$ from the speech recognizer which is edited using the word-edit machine (λ^{W}_{edit}). The gesture lattice ($\lambda_{\mathcal{G}}$) encodes the set of gestures $\mathcal{G}$ from the gesture recognizer which is edited with the gesture-edit machine (λ^{G}_{edit}). The edited lattices are aligned using the $\mathcal{R}$:$G \rightarrow W$ transducer introduced in Section 2. We then search for the least cost path (with minimum number of edits) ($argmin$) that pairs the edited speech string with the edited gesture string.

$$(g^*, s^*) = \underset{(g,s)\in\mathcal{G}\times\mathcal{S}}{argmin} \; (\lambda_{\mathcal{G}} \circ \lambda^{G}_{edit}) \circ \mathcal{R} \circ (\lambda_{\mathcal{S}} \circ \lambda^{W}_{edit}) \qquad (2)$$

In Table 3, we summarize the overall accuracy improvements from applying both speech and gesture editing techniques. On the 709 speech-only and multimodal utterances, the grammar-based multimodal integration and understanding model achieves a concept accuracy of 38.9% using the 1-best ASR and gesture recognizer output. This low accuracy is to be expected given that a large proportion of the user's utterances are not encoded in the multimodal grammar. On editing the ASR 1-best output using the edit machine shown in Figure 8 and integrating with unedited gesture lattice, we obtain a concept accuracy of 51.5%. Using the speech edit machine that incorporates knowledge about the application database, the concept accuracy improves to 63.2%. And finally, incorporating gesture edits in conjunction with speech edits improves the concept accuracy to 64.5%. While there is a significant improvement in concept accuracy attributable to gesture editing, the impact of gesture editing is smaller than that seen from speech editing However, this is somewhat to be expected since far more of the data is speech only than multimodal and in this domain there are considerably more recognition errors and out of grammar utterances in the speech stream than the gesture stream.

There are a number of other methods for achieving robustness to speech and gesture recognition errors as mentioned in Section 4. Most of these techniques are not declarative in nature and the application knowledge is tightly integrated into the program for achieving robustness. We found that encoding the edit operations in a declarative manner as a finite-state transducer allowed us to naturally extend these techniques to apply on speech and gesture lattices. This approach is particularly attractive for multimodal language processing since lattice representation of recognition hypotheses from each of the input modes is the most efficient way to exploit mutual compensation across these modes. In future work, we would like to explore learning from data how to balance gesture editing and speech editing based on the relative reliabilities of the two modalities.

Approach	Concept Sentence Accuracy
Grammar-based	38.9%
Speech-edit	51.5%
Tuned-speech-edit	63.2%
Tuned-speech+Gesture-edit	64.5%

Table 3: Concept accuracy using speech and gesture recognition output

7. CONCLUSIONS

Multimodal interfaces have tremendous potential to increase the usability and utility of mobile information services such as local search. To reach this potential, robust meth-

ods for multimodal integration and understanding are required that can be authored without access to large amounts of training data before deployment. We have shown here how techniques initially developed for overcoming errors and unexpected strings in the speech input, can be applied to gesture processing, resulting in overall improvement in the robustness and effectiveness of finite-state mechanisms for multimodal understanding and integration.

8. ACKNOWLEDGMENTS

We thank Patrick Ehlen, Helen Hastie, Candace Kamm, Preetam Maloor, Amanda Stent, Gunaranjan Vasireddy, Marilyn Walker, and Steve Whittaker for their contributions to the MATCH system. Thanks also to the anonymous reviewers for their helpful comments.

9. REFERENCES

[1] A. Adler and R. Davis. Speech and sketching for multimodal design. In *Proceedings of 9th International Conference on Intelligent User Interfaces*, pages 214–216. ACM Press, 2004.

[2] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Towards conversational human-computer interaction. *AI Magazine*, 22(4):27–38, December 2001.

[3] E. André. Natural language in multimedia/multimodal systems. In R. Mitkov, editor, *Handbook of Computational Linguistics*, pages 650–669. Oxford University Press, 2002.

[4] S. Bangalore and M. Johnston. Balancing data-driven and rule-based approaches in the context of a multimodal conversational system. In *Proceedings of North American Association for Computational Linguistics/Human Language Technology*, pages 33–40, Boston, USA, 2004.

[5] S. Bangalore and M. Johnston. Robust understanding in multimodal interfaces. *Accepted for publication in Computational Linguistics*, 2008.

[6] M. Boros, W. Eckert, F. Gallwitz, G. Gŏrz, G. Hanrieder, and H. Niemann. Towards understanding spontaneous speech: word accuracy vs. concept accuracy. In *Proceedings of International Conference on Spoken Language Processing*, pages 41–44, Philadelphia, USA, 1996.

[7] J. Cassell. Embodied conversational agents: Representation and intelligence in user interface. In *AI Magazine*, volume 22, pages 67–83, 2001.

[8] A. Cheyer and L. Julia. Multimodal Maps: An Agent-Based Approach. *Lecture Notes in Computer Science*, 1374:103–113, 1998.

[9] A. Ciaramella. A Prototype Performance Evaluation Report. Technical Report WP8000-D3, Project Esprit 2218 SUNDIAL, 1993.

[10] P. Cohen, M. Johnston, D. McGee, S. L. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. Multimodal interaction for distributed interactive simulation. In M. Maybury and W. Wahlster, editors, *Readings in Intelligent Interfaces*, pages 562–571. Morgan Kaufmann Publishers, 1998.

[11] P. R. Cohen, M. Johnston, D. McGee, S. L. Oviatt, J. Clow, and I. Smith. The efficiency of multimodal interaction: a case study. In *Proceedings of International Conference on Spoken Language Processing*, pages 249–252, Sydney, Australia, 1998.

[12] J. Dowding, J. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. GEMINI: A natural language system for spoken-language understanding. In *Proceedings of Association for Computational Linguistics*, pages 54–61, Columbus, OH, USA, 1993.

[13] P. Ehlen, M. Johnston, and G. Vasireddy. Collecting mobile multimodal data for MATCH. In *Proceedings of International Conference on Spoken Language Processing*, pages 2557–2560, Denver, CO, USA, 2002.

[14] A. L. Gorin, G. Riccardi, and J. H. Wright. How May I Help You? *Speech Communication*, 23(1-2):113–127, 1997.

[15] J. Gustafson, L. Bell, J. Beskow, J. Boye, R. Carlson, J. Edlund, B. Granstrm, D. House, and M. Wirén. Adapt - a multimodal conversational dialogue system in an apartment domain. In *International Conference on Spoken Language Processing*, pages 134–137, Beijing, China, 2000.

[16] M. Johnston. Deixis and conjunction in multimodal systems. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 362–368, Saarbrücken, Germany, 2000.

[17] M. Johnston and S. Bangalore. Finite-state multimodal parsing and understanding. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 369–375, Saarbrücken, Germany, 2000.

[18] M. Johnston and S. Bangalore. Finite-state multimodal integration and understanding. *Journal of Natural Language Engineering*, 11(2):159–187, 2005.

[19] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. MATCH: An architecture for multimodal dialog systems. In *Proceedings of Association of Computational Linguistics*, pages 376–383, Philadelphia, PA, USA, 2002.

[20] J. A. Landay and B. A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, 34(3):56–64, March 2001.

[21] S. Oviatt. Multimodal interactive maps: Designing for human performance. *Human-Computer Interaction*, 12(1):93–129, 1997.

[22] S. Oviatt. Mutual disambiguation of recognition errors in a multimodal architecture. In *Proceedings of the Conference on Human Factors in Computing Systems: CHI'99*, pages 576–583, Pittsburgh, PA, USA, 1999. ACM Press.

[23] W. Wahlster. SmartKom: Fusion and fission of speech, gestures, and facial expressions. In *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, pages 213–225, Kyoto, Japan, 2002.

[24] W. Ward. Understanding spontaneous speech: the Phoenix system. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 365–367, Washington, D.C., USA, 1991.