AIAA Guidance, Navigation, and Control Conference and Exhibit
16 - 19 August 2004, Providence, Rhode Island

AIAA 2004-5354

Natural Language Processing in the Control of Unmanned Aerial Vehicles

E. Craparo[*] and E. Feron[†]

Massachusetts Institute of Technology, Cambridge, MA 02139

This paper addresses the opportunities and challenges involved in applying natural language processing techniques to the control of unmanned aerial vehicles (UAVs). The problem of controlling an unmanned aircraft via natural language inputs is formulated as a feedback control problem. Two implementations of such systems are described. The phraseology of the existing air traffic control language is used as a base command set, and knowledge of air traffic control and airport operations, combined with existing natural language processing techniques, is used to achieve a higher recognition success rate than a traditional natural language processor designed for a more general domain of discourse would. This is the first known attempt at formalizing air traffic control phraseology for use in an unmanned system, and the first known flight of a vehicle controlled by natural language inputs. Outstanding problems and possible directions for future research are described.

I. Introduction

Unmanned aerial vehicles (UAVs) are becoming increasingly useful in military, commercial, and scientific applications. Currently, UAVs can be found performing surveillance and reconnaissance missions for the military, collecting scientific data in areas that are inhospitable or inaccessible to humans, and furthering commercial and agricultural enterprises. One of the primary needs of military and civilian users is an interface with which a single human operator can coordinate multiple UAVs with the same ease that air traffic controllers coordinate multiple aircraft.

Current interfaces between humans and UAVs do little to recognize the human affinity for verbal communication or the accepted (and empirically effective) practice of guiding aircraft through verbal commands. Existing control schemes such as pure data link and radio control greatly restrict the variety of commands that may be easily passed to a UAV. Radio control relies on continual monitoring and deliverance of low level commands, and it ignores the necessity of centralized planning in aircraft guidance. Data link, on the other hand, enables centralized control but also necessitates high controller workload and is inconsistent with current air traffic control practices at many airports. Both of these control schemes severely restrict the extent to which UAVs can be successfully integrated into the existing civil and military air traffic control system. This need not be, however - both the highly structured nature of air traffic control phraseology and the algorithmic and goal-driven nature of flight make unmanned air traffic control an ideal venue for the application and development of natural language processing technology.

II. Previous Work

The Stanford Computational Semantics Laboratory has investigated natural language guidance of unmanned aerial vehicles.[1] Their work was aimed at multi-modal communication with UAVs, e.g. communication via voice and graphical inputs. While it addressed many important issues related to task-oriented dialog, it did not do so in the context of air traffic control. Similarly, researchers at Brigham Young University have developed voice- and PDA-based UAV control interfaces and have obtained good performance on

[*]Student Member AIAA, Ph.D. candidate, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, `emily1@mit.edu`. Author to whom all correspondence should be addressed.

[†]Associate Fellow AIAA, Associate Professor of Aeronautics and Astronautics, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 77 Massachusetts Ave, M.I.T. Rm 32-D724, Cambridge, MA 02139, `feron@mit.edu`

voice command recognition,[2] although their voice command set was also developed solely for their research and was not based on air traffic control commands, nor did it allow the kind of flexibility that the air traffic control language allows.

Although little has been done to realize the possibility of controlling unmanned aerial vehicles via voice commands, some work has been done in applying natural language processing and its sister technology, speech recognition, to air traffic control for use with manned systems. This is generally done with the goal of reducing error due to miscommunication, ambiguous instructions, or errors of memory.

Cushing[3] has proposed a natural language processing system capable of acting as a "mediator" between air traffic controllers and pilots. Citing accidents due to ambiguity of commands and errors of interpretation, Cushing suggests passing communications through a processing unit capable of automatically filtering out potential sources linguistic confusion and asking the speaker for clarification if necessary. In addition to its applicability as an operational safeguard, Cushing notes, such a system might also be useful as a training device for pilots an controllers, alerting them to any potentially confusing verbal idiosyncrasies they may have. In addition, it could serve as one layer of a later human-machine interface.

Churcher *et al* intended to use speech recognition technology to automatically transcribe certain, essential parts of transmissions between the air traffic controllers and airborne pilots.[4] They claimed that these transcripts could be used air traffic control training purposes, or for relaying information to the pilot in flight and thereby reducing pilot workload in a manner similar to data link. They used IBM's off-the-shelf commercial continuous speech recognizer, and it gave them only a modest accuracy of recognition (around 30%) in its base form. However, when the device was augmented with other knowledge sources and higher levels of linguistics such as contextual information and context-free syntax, the accuracy could be greatly improved to over 70% even in noisy environments. While this result is not spectacular, it does show a large improvement over the baseline result and indicates a promising area for future work.

An important example of such high-level knowledge is the structure of a discourse. Discourse is defined as a collocated, coherent, and related group of sentences.[5] There are at least two different approaches to coherence. One is an informational approach, where relationships between sentences impose constraints such as result, explanation, parallel, and elaboration on the information in the utterances.[6] Historically, this approach has been applied predominantly to monologues between a speaker and hearers. Another approach is called intentional approach, in which utterances are understood as actions, requiring that the listeners infer the underlying goal.[7] This intentional approach has been applied mostly to dialogs.

The notion of intentional coherence in discourse plays a significant role in the air traffic control system, because the high level tasks of landing, takeoff, and maneuvering around the airport must be coordinated by a group of sentences, not just individual sentences. Even human pilots are urged to take advantage of the predictable nature of flight discourse; a student guide to voice communications published by the United States Navy gives this advice:[8]

> Know what to **expect**. As you progress through each flight, you should know what is expected to happen. If you know what is to be said ahead of time, responding correctly will be much easier.

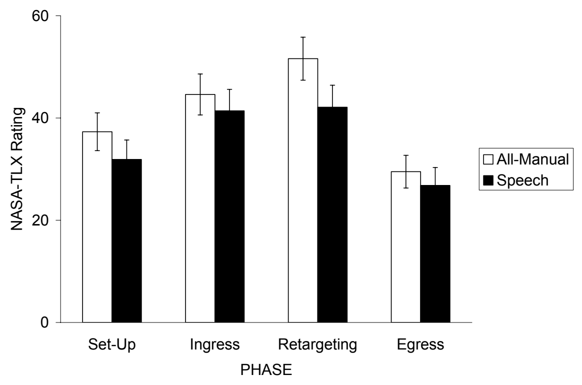
Thus, pilots are encouraged to rely on their knowledge of the established structure of both flight and communications to form their responses to ATC commands. The manual continues:

> Every conversation with a controlling agency or service follows a specific progression...proper communication involves the realization of which progression phase you are in and making correct and timely responses.

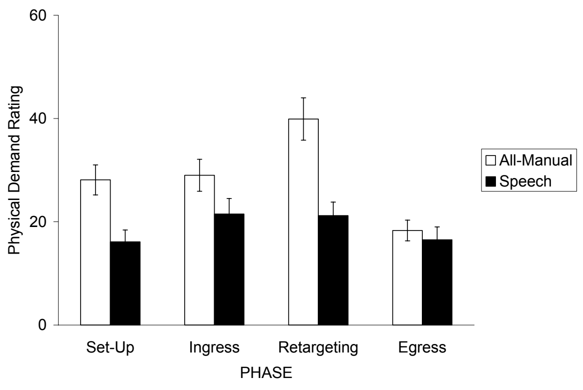
This aspect of communication is not lost on computational linguists. Grosz argued that a discourse could be represented as a composite of three interacting components: a linguistic structure, an intentional structure, and an attentional state.[7] Grosz also pointed out that task-oriented dialogs have a linguistic structure that closely parallels the intentional structure of the task being performed.[9] The fundamental notion in this observation is that a discourse has an underlying purpose which it aims to achieve, called the discourse purpose, and that each segment of a discourse also has a finer-grained purpose, called discourse segment purpose. Then they are organized in a tree-like hierarchy with two coherence relations: dominance and satisfaction-precedence. This structure helps a discourse management system understanding the intention of a speaker.

A. Speech Processing in Aviation

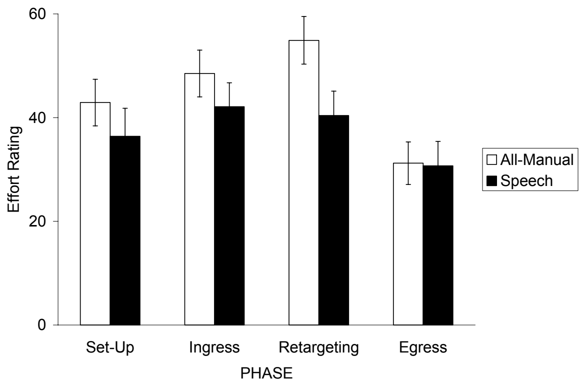
A number of new aviation systems are beginning to utilize speech recognition technology. It is widely believed that enabling voice control of various pieces of aviation automation will bring about a more efficient and less error-prone mode of operation. For instance, the Air Force has recently begun testing of voice recognition software for use in its Airborne Warning and Control System (AWACS) E-3 Sentry aircraft.[10] This technology would enable air battle managers to control their radar screens using voice commands rather than mouse, keyboard, and function key inputs. Preliminary testing has indicated that speech recognition software may enable battle managers to achieve a 40% reduction in workload, with increase accuracy and situational awareness.[10] As shown in Figure 1, experiments indicate that a consistent decrease in mental workload, physical demand, effort ratings, and necessary communication events may be realized uniformly in all phases of AWACS operation through the use of voice commands.[11]



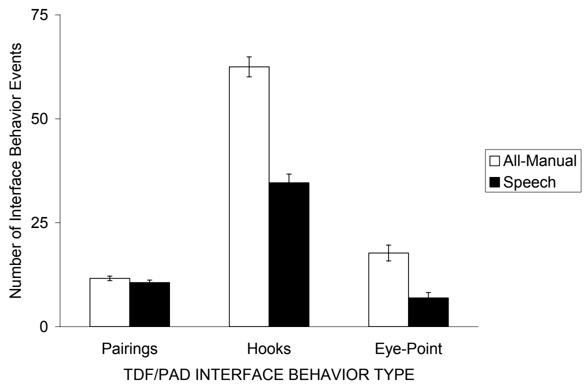
(a)



(b)



(c)



(d)

Figure 1. Mean NASA-TLX ratings and standard errors of mental workload of AWACS controllers as a function of phase and control modality (a), mean Physical Demand ratings and standard errors as a function of phase and control modality (b), mean Effort ratings and standard errors as a function of phase and control modality (c), and mean number and standard errors of three TDF/PAD interface activity measures as a function of control modality.

In addition to these measurable improvements in performance when using voice commands, AWACS controllers also had highly favorable reactions to the voice interface and indicated that they felt comfortable and at ease with the interface.[11] Many seemed to appreciate the decrease in workload and the quickness with which tasks could be accomplished via the voice interface, with one respondent noting that

> The speech controls would be an excellent addition to future AWACS interfaces. Given the fact that future command and control will be even more focused on time sensitive tasks, speech

controls will allow future [air weapons officers] the ability to handle more complex and a greater number of tasks in the same amount of time.

These positive sentiments were echoed by most participants in the experiment; as Table 1 indicates, a large percentage of participants preferred to utilize voice commands when both voice and manual commands were available to them, and as Table 2 shows, the majority of participants agreed that voice control was either "very useful" or "extremely useful" in most tasks, and all agreed that it had at least "minor utility" in all tasks.[11]

TASK	*%*
Set-up Phase, Open ATO	50
Set-up Phase, Mark Controller	100
Set-up Phase, Sort ATO	64
Set-up Phase, Set Bulls-eye	91
Ingress Phase, Hook Aircraft (Tag)	100
Ingress Phase, Check-in Aircraft	67
Ingress Phase, Open ATO	18
Ingress Phase, Hook Aircraft (Pair)	82
Retargeting Phase, Repairing	100
Switch Bulls-eye Probe Task	83
Range & Bearing Probe Task	58
ATO Question Probe Task	42
Threat Call Hooks	10

Table 1. Percentage of participants using speech to accomplish speech-enabled tasks during preference trials.

Similar technologies to the voice-activated AWACS workstations have been proposed for use in "interactive kneeboards" intended for use by Navy fighter pilots, with the goal of achieving a comparable reduction in workload and "head-down time" and an increase in accuracy.[12]

It is reasonable to believe that the benefits of voice-recognition technology that have been observed in systems such as the AWACS workstations and voice-activated kneeboards will also be found in UAV control architectures.

Utility Rating Question	*Waste of Effort*	*Minor Utility*	*Somewhat Useful*	*Very Useful*	*Extremely Useful*
Interacting with ATO	0	1	3	7	1
Interacting with TDF/PAD situation display	0	0	9	3	0
Responding to bearing & range query	0	2	2	6	1
Pairing friendly tracks against targets	0	1	0	3	8

Table 2. Number of participants that selected each response category for four utility rating questions on a post-experimental speech interface survey form.

III. Language Processing in Feedback Control

The block diagram shown in Figure 2 is a schematic diagram of a feedback control system that incorporates natural language processing. As in the current air traffic control system, a human controller is present to issue directives based on an aircraft's current state and the controller's intentions. Once these verbal commands are processed by the natural language processing unit, they are translated into a set of high-level goals and constraints that are then passed on to the aircraft's planning algorithms. These planning algorithms then generate a sequence of maneuvers for the aircraft. The natural language processing unit is also capable of generating a verbal reply to the user, either stating the perceived goal or requesting additional information. It is worth noting that natural language processing is about parsing sentences and understanding them; it is not about speech recognition, although the two can easily build upon each other.

IV. Implementations

MIT has taken part in creating two implementations of systems that utilize natural language processing as the primary means of communicating with an unmanned aerial vehicle. The first of these implementations translated text inputs into commands that were sent to simulated aircraft in an airport environment, with surrounding airspace. The second also utilized text inputs, and it was used in a flight demonstration at NASA Dryden.

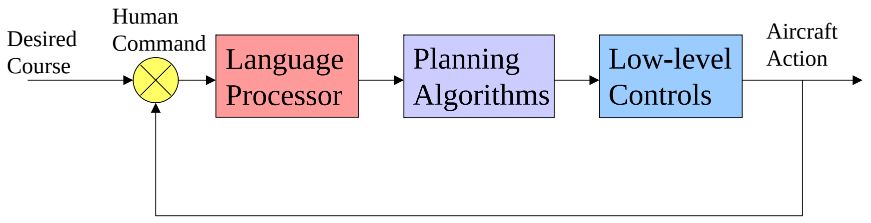


Figure 2. Incorporation of a natural language processor into a feedback control loop.

A. Simulation System

1. System Overview

The simulation system is comprised of five distinct modules as shown in Figure 3. A user playing the role of a ground and air traffic controller can issue text commands to the preprocessor, which then passes the edited commands to the sentence parser. The sentence parser recognizes a set of sentence structures, and converts sentences into standardized verb templates. These verb templates are then passed to the discourse manager. In the discourse manager, consecutive verb templates are analyzed for feasibility and inconsistencies. Inconsistencies are reported to the user in much the same way a that pilot would ask a controller for clarifications. If no inconsistencies are found, the verb templates generated by the discourse manager are transformed into primitive commands for the aircraft. For this simulation, a database maintains the current and immediate past states of the airport and the various aircraft in the system. It is consulted when the discourse manager checks for inconsistencies, and it is updated by the discourse manager when new commands arrive. Finally, a graphical simulator displays the airport and aircraft states, and it maintains the simulation clock. The two main elements of this system, the parser and discourse manager, are described in more detail below.

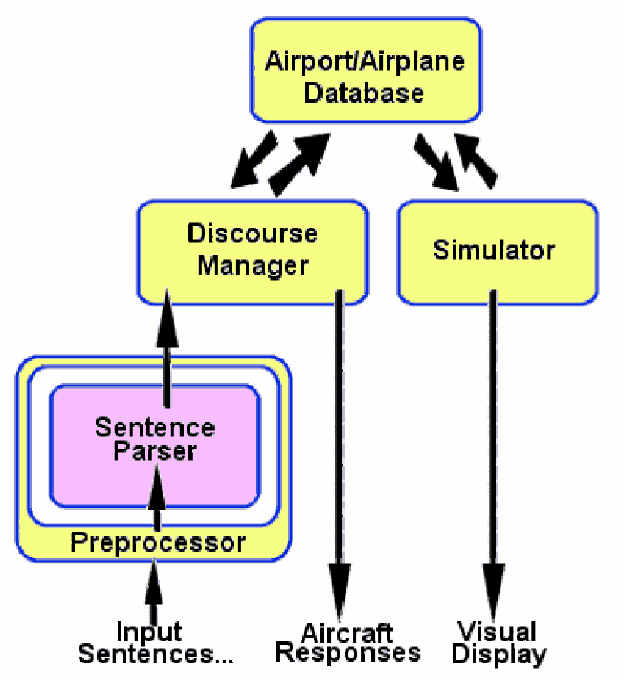


Figure 3. System layout.

2. Parser

Parsing is the recognition and assignment of structure to a string. Syntactic parsing consists of the recognition of sentences and the assignment of syntactic structure to them. Thus, in order to parse a sentence, each of its constituent parts must be given one or more labels, some formal syntactic structure for these labels must be defined, and the sentence must be processed in such a way that one or more parse trees are assigned to it.

Before a sentence may be assigned a particular structure, each of its constituent words must be identified and labeled. A lexicon provides a repository of words in a given language, and their relevant syntactic characteristics. A lexicon of words relevant to air traffic control would contain entries such as

$$\begin{aligned}&Number \longrightarrow one \mid two \mid three \mid ...\\ &Determiner \longrightarrow a \mid an \mid the \mid ...\\ &Noun \longrightarrow contact \mid runway \mid taxiway \mid ...\\ &Verb \longrightarrow contact \mid taxi \mid turn \mid ...\\ &...\end{aligned}$$

The lexicon for the simulation system described in this paper contains the verbs relevant to air traffic control, as well as new classes of words for airplane names, taxiway names, units of measurement, and numbers that conform to conventional air traffic control phraseology.

In addition to a system for placing labels on each word in a sentence, it is also necessary to describe the ways in which these words are related to one another. A *grammar* is a set of rules describing the ways in which terminals (words) can be represented by non-terminals (equivalence classes such as noun phrase (denoted in this paper as NP), verb phrase (VP), and sentence (S)).

There are many possible grammatical frameworks, but the ones used most commonly by computational linguists are those found in the Chomsky hierarchy (see Table 3).[5] Here, types of grammars are arranged in descending order of complexity. That is, a Type 0 grammar is capable of defining all languages defined by Types 1, 2, and 3, as well as some languages that cannot be defined by the other types of grammars, while Type 1 grammars can define all languages defined by Type 2 and Type 3 grammars, but not all languages defined by Type 0 grammars.

Type	**Common Name**	**Rule Skeleton**	**Linguistic Example**
0	Turing Equivalent	$\alpha \longrightarrow \beta$; $\alpha \neq \epsilon$	Augmented Transition Networks
1	Context Sensitive	$\alpha A\beta \longrightarrow \alpha\gamma\beta$; $\gamma \neq \epsilon$	Tree-Adjoining Grammars
2	Context Free	$A \longrightarrow \gamma$	Phrase Structure Grammars
3	Regular	$A \longrightarrow xB$ or $A \longrightarrow x$	Finite State Automata

Table 3. The Chomsky Hierarchy. A and B denote a non-terminals, α, β, and γ denote strings of terminals and non-terminals (possibly empty, except where indicated), x denotes a string of terminals, and ϵ denotes the empty string.

As Table 3 indicates, the various types of grammars are characterized by the forms of grammar rules that are allowed. Type 0 grammars are unrestricted; any non-empty string may be written as any other string. The class of languages defined by Type 0 grammars are the recursively enumerable languages, i.e. those that may be enumerated by a Turing machine.

In context-sensitive grammars, a non-terminal may be written as any non-empty string of terminals and non-terminals, provided it is in a certain context. It has been demonstrated that some natural languages, such as Swiss German, are context-sensitive[13].[14]

Context-free grammars allow a non-terminal to be written as a string of terminals and non-terminals, or possibly the empty string. Many natural languages appear to be described by context-free grammars, or at least closely approximated by them.

Finally, the regular grammars allow a non-terminal to be written as a string of terminals followed by at most one non-terminal. They are equivalent to regular expressions and so can be described by finite state automata.

From the point of view of someone wishing to formally describe a given language, the prospect of a finite-state grammar may be attractive. Finite-state grammars, however, are incapable of capturing some aspects of the English language. Chomsky[15] has shown that a language can be generated by a finite state automaton if and only if it can be generated by a context-free grammar that does not have any center-embedded recursions, i.e.

$$A \longrightarrow \alpha AB$$

For example, the following sentences may be constructed using center-embedded recursions:

The man stole the car.
The man the policeman chased stole the car.
The man the policeman the woman called chased stole the car.
etc...

Another example involves sentences embedded in other sentences. Let S denote an English sentence. Then, another valid sentence is

The person who said S was incorrect.

This sentence may also be center-embedded within another sentence, and this process may continue indefinitely.

While difficult to read and understand[a], these sentences are grammatically correct according to the English syntax. Thus, while finite-state grammars can model many aspects of English and can often provide good approximations of it, not all English syntax can be modeled with a finite-state grammar.

It appears that English is, in fact, a context-free language.[5] No known analysis has been made of the air traffic control language. However, it is assumed that "aviation English," as a more constrained subset of English, is at most context-free as well.

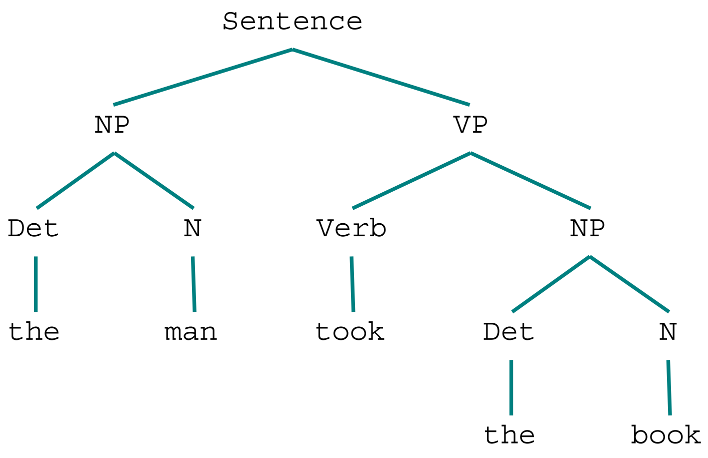


Figure 4. The first context-free grammar parse tree (Chomsky, 1956).

A context-free grammar is formally defined as a 4-tuple[5] consisting of

1. A set of non-terminal symbols N
2. A set of terminal symbols Σ such that $\Sigma \cap N = \phi$
3. A set of productions P, where each is of the form $A \longrightarrow \alpha$, $A \in N$, $\alpha \in (\Sigma \cup N)*$
4. A start symbol S

In the simulation implementation, N and part of Σ are defined by the lexicon, while P and the remainder of Σ are contained in the syntactic rules. S is implicitly contained in the syntactic rules as well.

The parser contained in this system is an extension of the Earley context free parser,[17] which utilizes a dynamic programming approach to eliminate the problems and efficiencies encountered by top-down or bottom-up parsers. In general, dynamic programming algorithms systematically solve and store the solutions to all sub-problems needed to solve the overall problem. In the case of parsing, this means that the various subtrees for all parts of the input are discovered once, and then looked up on subsequent reparses. Because there is a great deal of backtracking and regularity in the search space inherent in parsing, this can result

[a]It has been noted[16] that many of the constructions used to prove the complexity of grammars of natural languages also tend to cause severe difficulty in human comprehension. Thus, while it may not be possible to generate all *grammatical* English strings without resorting to a complex grammar, the subset of those strings that is *easily understood* by humans could possibly be generated by a simplified grammar, perhaps even a finite state grammar.

in a significant gain in time efficiency. In addition, because all of the possible subtrees are stored in a single chart, all possible parses may be retrieved, including ambiguous ones.

In a goal-oriented language such as air traffic control, it is useful to formalize commands in terms of the action they are meant to induce, as well as any qualifications that may exist on this action. For example, some command verbs, such as “contact,” generally take one or two arguments (e.g. “contact ground” or “contact tower on 119.7”), while others may take an undetermined number of arguments (e.g. “taxi to runway five,” “taxi to runway five via sierra,” “taxi to runway five via sierra and echo,” etc.). To capture this predicate-argument relationship between command verbs and their arguments, it is useful to employ a subcategorization frame of the type used in natural language processing to encode relationships between words and their complements.[5] While this project focuses on the relationships between command verbs and their arguments, it is also possible to create subcategorization frames for other parts of speech as well.

The verb templates generated by this system’s parser are rather specialized - they output information with relevant headings based on the verb being parsed.

For example, the input sentence “taxi to runway three via zulu” generates

```
[OUTPUT](go :to (runway :num 3) :via (taxiway :num zulu) :agent you)
```

while the sentence “turn three two zero” generates

```
[OUTPUT] (turn :heading (heading :to 320) :agent you)
```

This actor-action-object framework facilitates the interpretation of parser outputs by those parts of the system that must translate these outputs into actions.

3. Discourse Manager

Because the parser’s context-free grammar rules were intentionally made as general as possible, many nonsensical sentences are parsed. For example, the same rules that would allow the parser to handle a sentence such as “follow that Continental to the runway” might also allow it to accept a sentence such as “follow that temperature to the runway.” Additionally, a sentence accepted by the parser may be semantically acceptable in isolation, but not when taken into context with surrounding sentences or the state of the aircraft. For example, the command “climb and maintain ten thousand” is semantically acceptable at the sentence level, but is inappropriate if the aircraft is already at an altitude of twelve thousand feet. It was to alleviate these problems that the discourse manager was created.

When the discourse manager (shown schematically in Figure 6) receives a verb template from the parser, it performs a semantic interpretation of the verb template to determine what action to take. If it finds the requested action to be reasonable and unambiguous, it updates the states of aircraft in the system database accordingly and reads back this update to the controller. However, if the discourse manager finds the command to be ill-posed (that is, it is nonsensical, inappropriate for the circumstances, or inconsistent with previous commands), it generates a response message requesting clarification or another command. The discourse manager takes the context of the input into account, and rejects a command requesting an action in conflict with the context even if the input is syntactically and semantically valid at the sentence level.

As shown in Figure 6, this module has three internal stages: a preprocessor, a set of dispatch functions, and a set of handler functions. The preprocessor transforms the arguments of a given verb template into a data structure containing (record, value) pairs, and invokes an appropriate dispatch function depending on the verb. Recall that the lexicon contains verb entries, such as “climb,” “descend,” “taxi,” “remain,” and “maintain.” Each verb entry has its own dispatch function in the form of dispatch_*(). For example, any command containing verb “maintain” is processed by dispatch_maintain().

It is the responsibility of these dispatch functions to further parse their arguments, and invoke the appropriate actions (such as changing altitude, or updating a value in the system’s database). At this stage, the dispatch functions can detect and reject messages that are semantically valid in the sentence level, but are not coherent in the greater context.

First in the dispatch functions’ analysis of a command is the notion of history. For example, there exist commands that nullify the effects of the previous command, such as “cancel that”, “never mind”, and “let’s not do that.” In order to resolve the meanings of “that”, it is necessary to keep track of the history of commands. In this system, it is assumed that only the immediately preceding command will be referenced, and so one-command history is kept. An example of the handling of a cancellation command is as follows:

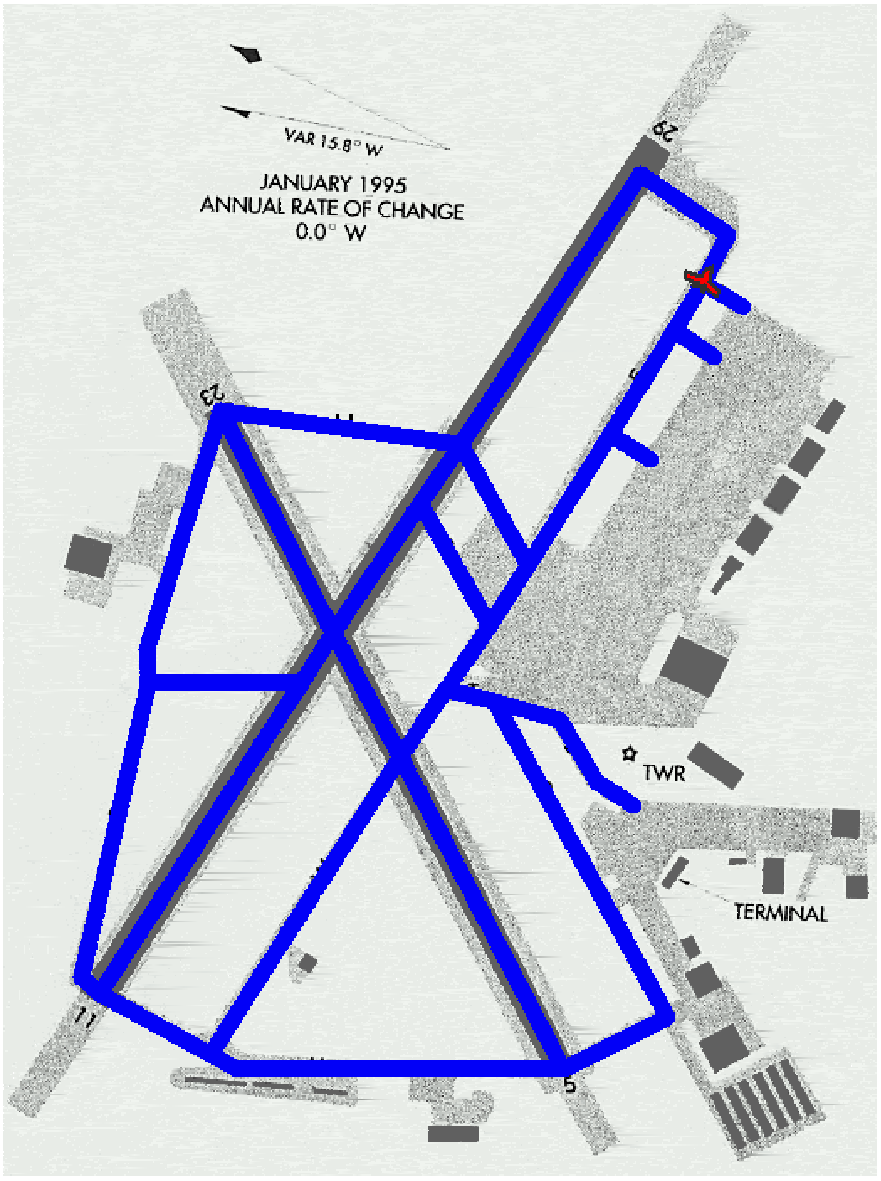


Figure 5. Screenshot of the near-airport environment in which users may control simulated aircraft via natural language inputs.

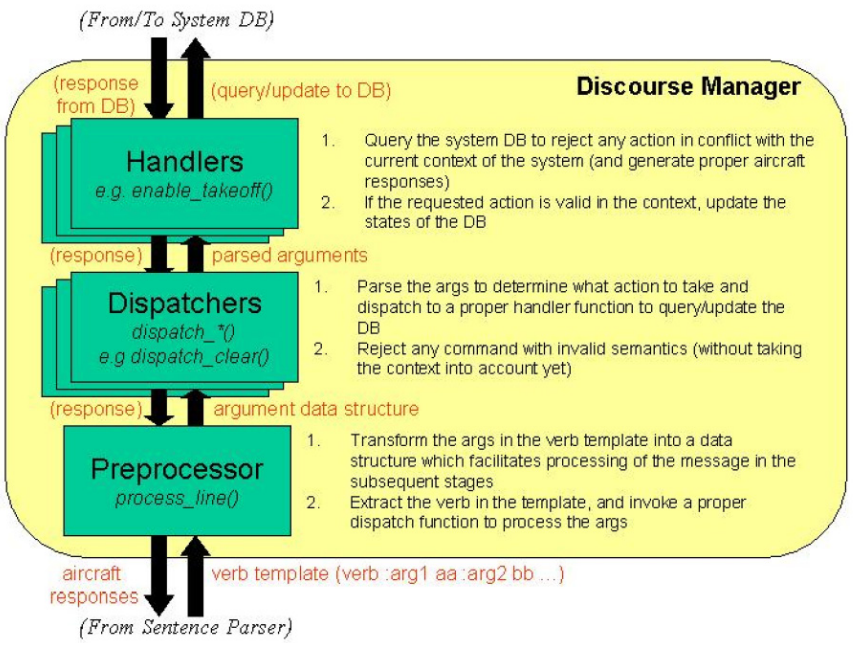

(From/To System DB)
(response from DB)
(query/update to DB)
Discourse Manager
Handlers
e.g. enable_takeoff()
1. Query the system DB to reject any action in conflict with the current context of the system (and generate proper aircraft responses)
2. If the requested action is valid in the context, update the states of the DB
(response)
parsed arguments
Dispatchers
dispatch_*()
e.g dispatch_clear()
1. Parse the args to determine what action to take and dispatch to a proper handler function to query/update the DB
2. Reject any command with invalid semantics (without taking the context into account yet)
(response)
argument data structure
Preprocessor
process_line()
1. Transform the args in the verb template into a data structure which facilitates processing of the message in the subsequent stages
2. Extract the verb in the template, and invoke a proper dispatch function to process the args
aircraft responses
verb template (verb :arg1 aa :arg2 bb ...)
(From Sentence Parser)

Figure 6. The discourse manager.

```
TOWER: Delta, clear to taxi to two-niner.
PLANE: Roger.
TOWER: Cancel that.
PLANE: Roger.  Please specify a new destination.
```

A second stage of analysis in the discourse manager deals with implicit references. For example, there are commands such as "remain this frequency" that require contextual knowledge for complete comprehension. Other commands may reference procedures or locations uniquely relevant to a particular airport, or perhaps current weather conditions. Thus, the dispatcher can query the airport/aircraft database for clarification of some commands.

The notions set forth by Grosz *et al*[7] for dealing with the structure of a discourse are particularly relevant in the design of the discourse manager. This preliminary system deals with two dominant intentions: takeoff and landing. These intentional structures determine the response of the system given an input utterance. That is, the utterance is mapped to a corresponding (intermediate) intention, and handlers consult the database to see whether or not all the prerequisites have been met. For example, it does not make sense to issue ground-specific commands to an aircraft in mid-air, and vice versa. Nonsensical or inapplicable commands are rejected and the user is requested to make a new command.

The last stage of the discourse manager is a group of handler functions that actually update the state of the system. In the general case, a very sophisticated (and often intractable) discourse manager is needed to resolve references and to determine the intentional structure of dialogs. However, the air traffic control language has evolved in such a way that much of the ambiguity inherent in human languages has been eliminated. The air traffic control domain is also highly goal-oriented, which permits the assumption that virtually all utterances will be either directive or informative. Without this constraint, the problem of designing a robust discourse manager becomes much more complicated[7181920].[1]

A screenshot of the simulation is shown in Figure 5.

B. Hardware Implementation

Following our initial effort, a second natural language interface system was constructed for use in a hardware implementation, outlined in Figure 7.[21] The overall architecture for this system is described further in Ref. 21.The language processing system in this implementation was similar to the simulation system in terms of is basic elements, with added safeguards to ensure a safe operating experience. It was also controlled from the air rather than from the ground, and thus utilized a command set more akin to lead-wingman interaction than to air traffic control phraseology. As part of this mission, the pilot and/or weapon system officer was given the capability to interact with the UAV via natural language. As such, the goal of the system was to minimize the workload on the operator, while efficiently and effectively communicating with the UAV.

While this type of system helps to simplify the operator interface with the unmanned system, analyzing natural language inputs such as the sentences generated by a weapon system officer and/or pilot presents challenges similar to traditional natural language parsing. Recall that parsing is the process that takes an input sentence, for instance

"Proceed to location in minimum fuel"

and converts it into a formal representation. This formal representation is traditionally a tree structure (as described for the simulation implementation) which can in turn be translated into an explicit formal command. In this experiment, parsing first applied entity extraction to all the individual concepts (e.g. "Eagle 1" or "Echo-Charlie 5") and then combined these concepts through cascades of finite-state transducers using techniques derived from those described in Ref. 22. The vocabulary of this particular experiment was much smaller than for generic application which reduces the level of ambiguity and makes parsing than on more open text. However, compared to other information processing tasks, this deployment required a particular emphasis on the safety of the parsing process. The stability of the runtime module is achieved by shifting the complexity of the system toward the offline model compilation phase (for which there is much less stringent stability requirements). This makes the runtime process much simpler. In addition, the runtime process consists mostly of finite-state operations whose algorithm can be proven correct and for which the input finite-state machines can be checked for particular formal properties. This approach also has the additional benefit of providing very efficient processing time (and the processing time can also be bounded explicitly).

As a result, the natural language interface module was designed to use a natural language parser to provide an operator with the ability to communicate the UAV using normal sentence commands (in English for this demonstration). For example, if the FW operators are notified of a potential threat, the FW operators could command the UAV to search a pre-defined region containing the potential by using the following sample dialog:

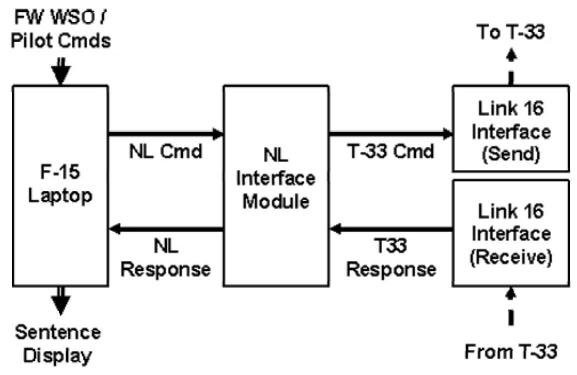


Figure 7. Natural language interface in the flight test environment.

FW: "UAV 7, this is Eagle 3."

UAV: "Go ahead, Eagle 3."

FW: "Add New Mission Task. Proceed to location Echo-Charlie 5 in Minimum Fuel. Search this region for threats and wait for further instructions after the task is completed"

UAV: "Roger. Acknowledge task information - proceeding to location Echo-Charlie 5."

FW: "Eagle 3, out."

In this dialog, there are four major components to the conversation:

1: *Introduction* The object beginning the conversation (the FW WSO in this situation) identifies the object who is to receive and react to this message (which is the UAV in this case).

2: *Response / Verification* The object receiving the introduction message or command responds with a verification that it received and understood the incoming message.

3: *Command / Question* This message contains the "information-carrying" portion of the conversation and requires a detailed response from the receiving party.

4: *Closure* This message signals the end of the conversation. In our scenario, the FW (which represents the commanding officer) will always end the conversation with "*(FW WSO Identifier)*, out." for a definitive end to the conversation.

In the conversation above, the command and response portions require the vehicle to send a message with information about the current status of the mission, task, etc. Therefore, we developed a protocol for questions, commands and their associated responses - for both sentences (for the human user) and their coded representations (for the machine) - to ensure that the messages communicated between both agents are accurate representations of the conversation. For this particular application, we observed that the introduction and closure statements are used to identify the parties involved in the conversation. Therefore, since there are only two parties (the FW WSO and UAV) involved in this demonstration, the system will not send these messages over the communications link to simplify the communications protocol for this system.

V. Future Work

This work may be extended in both theoretical and practical areas. While the system described above functions well in typical operational settings, it is not equipped to handle emergency situations. Aside from being able to respond to unusual commands that it may receive during an emergency, a UAV should also be sensitive to emergencies involving other aircraft. This is an example of an important area for further research – dialog-based aircraft state estimation. For an aircraft operating in a shared airspace, many external factors dictate the ways in which that aircraft should operate and communicate. For instance, a verbose status report should not be given to an air traffic controller who is already trying to deal with many other aircraft at once. Similarly, an aircraft should proceed cautiously if an emergency situation is detected at or near an airport. Ideally, these operational protocols and nuances of etiquette should be observed by unmanned aircraft as readily as they are by human pilots. For this to occur, the UAV will need to have an accurate model of the other aircraft that are operating near it, including their approximate positions and operational states. Because in many cases the only contact that aircraft have with one another is through communication channels, it is reasonable to attempt to make state estimations based on dialogs between aircraft and air traffic controllers.

Another area for further research about the structure of the air traffic control language would involve the notion of memory in air traffic control dialogs. How long should information be retained in the course of a dialog, and when should it be considered obsolete? Patterns of memory based on time difference between utterances, situational context of utterances, and perhaps relative importance or urgency of utterances may be present.

Future additions and improvements to an implementation of this system might include a multiple-aircraft interfacing capability that allows the controller to address all or part of a fleet of aircraft, as well as an individual craft; a database manager capable of incorporating information extracted from such sources as automated weather and airport advisories and transmissions between other aircraft ("party line information") as well as transmissions to that aircraft; and further additions to the corpus of known sentences and a more sophisticated discourse manager capable of time-sensitive discourse and improved and expanded intentional inference.

In addition to the voice recognition required for a hardware implementation, there are also other practical considerations to take into account. For example, a UAV utilizing a language processing system will need to be able to decide when to start and stop listening for a command. At first glance the standard air traffic control procedure appears adequate: When the name of the UAV is heard, this marks the beginning of a command utterance. A long pause or another aircraft's call sign may indicate the end. However, this strategy is not foolproof - for example, it may happen that the UAV hears someone talking about it rather than to it. Thus, a more robust algorithm, or possibly a slight modification to air traffic control procedures for UAVs, is required.

A. Immediate Future Work

Currently, we are developing an indoor experimental, real-time test bed for studying the application of natural language processing techniques to air traffic control and developing practical solutions for real-time operations. This test bed will feature hardware and simulated mission components which perform tasks in multiple real-time environments. This system will allow a user playing the role of a forward air traffic controller to operate up to five UAVs using voice commands based on the modern air traffic control language.

The language processing capability will be demonstrated on cases of interest. Both military and civil cases will be considered. Some possible scenarios might include typical near-airport operations, ground control, en route operations, emergency procedures, and military operation scenarios involving both traditional tactics and more novel UAV-oriented schemes. Demonstration cases will include single UAVs and UAV fleets, as well as heterogeneous fleets composed of manned and unmanned aircraft.

VI. Conclusions

Many issues remain to be investigated before a system of this nature may be successfully deployed in an operational environment. However, if the potential benefits of the natural language processing technology described in this paper are realized, the utility of UAVs will be greatly increased. Aside from allowing current users of UAVs to interact more easily with the vehicles, this technology could open up entire new venues for the use of UAVs.

VII. Acknowledgments

This work was partially supported by Air Force Research Laboratory Grant F33615-01-C-1850 (Safe Operation of Multi-Vehicle Systems), by MURI Grant 009628-001, and by an NDSEG fellowship sponsored by the Department of Defense. The authors would also like to thank Emmanuel Roche and Robert Berwick for their guidance, advice, and contributions.

References

[1]Doherty, P., Granlund, G., Kuchcinski, K., Sandewall, E., Nordberg, K., Skarman, E., and Wiklund, J., "The WITAS Unmanned Aerial Vehicle Project," *Proceedings ECAI*, 2000.

[2]Quigley, M., Goodrich, M., and Beard, R., *Semi-Autonomous Human-UAV Interfaces for Fixed-Wing Mini-UAVs*.

[3]Cushing, S., *Fatal Words: Communication Clashes and Aircraft Crashes*, The University of Chicago Press, 1994.

[4]Churcher, G. E., Ateall, E. S., and Souter, C., "Dialogues in Air Traffic Control," *Proceedings of the 11th Twente Workshop on Language Technology*.

[5]Jurafsky, D. and Martin, J. H., *Speech and Language Processing*, Prentice Hall, 2000.

[6]Hobbs, J. R., "Coherence and Coreference," *Cognitive Science*, 1979.

[7]Grosz, B. and Sidner, C. L., "Attention, Intentions, and the Structure of Discourse," *Computational Linguistics*, 1986.

[8]United States Navy, N. A. T. C., *Voice Communications: Student Guide; CNATRA P-806 (REV. 4-98)*, Prentice Hall, Corpus Christi, TX.

[9]Grosz, B., "The Representation and Use of Focus in Dialogue Understanding," 1977.

[10]Mayer, D., "AWACS Voice Recognition May Enhance Accuracy," *AFMC News Service Release 0334*, March 2004.

[11]Vidulich, M. A., Nelson, W. T., Bolia, R. S., Guilliams, N. M., McLaughlin, A. B., and Donnelly, B. P., "An Evaluation of Speech Controls for AWACS Weapons Directors," .

[12]"Pilot interviews, Training Squadron VT-86." *Personal communication*, 2004.

[13]Huybregts, R., "The Weak Inadequacy of Context-Free Phrase Structure Grammars," *In de Haan, G.; Trommele, M.; and Zonneveld, W. (Eds.), Van Periferie naar Kern*, Foris, Dordrecht, 1984.

[14]Shieber, S. M., "Evidence Against the Context-Freeness of Natural Language," *Linguistics and Philosophy*, 1985.

[15]Chomsky, N., "On Certain Formal Properties of Grammars," *Information and Control*, 1959.

[16]Pullum, G. K. and Gazdar, G., "Natural Languages and Context-Free Languages," *Linguistics and Philosophy*, 1982.

[17]Earley, J., "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, 1970.

[18]Litman, D. and Allen, J., "A Plan Recognition Model for Subdialogues in Conversation," *Cognitive Science*, 1987.

[19]Carberry, S., *Plan Recognition in Natural Language Dialog*, MIT Press, 1990.

[20]Passonneau, R. and Litman, D. J., "Intention-based Segmentation: Human Reliability and Correlation with Linguistic Cues," *Association for Computational Linguistics Conference*, Columbus, OH, 1993.

[21]Valenti, M., Schouwenaars, T., Kuwata, Y., Feron, E., How, J., , and Paunicka, J., "Development and Implementation of a Manned Vehicle - UAV Mission System," *AIAA Journal of Aerospace Computing, Information, and Communication*, 2004.

[22]Roche, E., "Parsing with Finite-State Transducers," *Finite-State Language Processing*, edited by E. Roche and Y. Schabes, MIT Press, 1997.