Integration of Live Video in a System for Natural Language Dialog with a Robot

Erik Sandewall, Hannes Lindblom and Björn Husberg
Department of Computer Science
Linköping University
Linköping, Sweden
erisa@ida.liu.se, hanli513@student.liu.se, bjorn.husberg@home.se

Abstract

For the communication with mobile robots during their missions to locations that are inaccessible or dangerous for people, it is desirable to make use of a combination of natural language, preferably in spoken form, and a graphical presentation of what is in the robot's field of sight. The present article addresses architectural and methodological issues for such multimodal systems against the background of a system of this kind that we have developed, the WITAS Robot Dialog Environment (RDE).

1 Goals and Issues

A major reason for having mobile robots is to let them go to places where it is impossible, inconvenient, or dangerous for people to go. In this article we consider robots that are equipped with a video camera as a part of their perception system. For the communication with such robots during their missions, one would like to use a combination of spoken natural language and visual presentation of what is in the robot's field of sight. In order for the interaction to be as natural as possible, it should be possible to show the operator the actual video that is 'seen' by the robot. It should also be possible for the operator as well as the robot to refer to the moving video, both by phrases in the vocal communication and by gestures that indicate objects or areas in the passing video image. We have built a software system, the WITAS Robotic Dialog Environment (RDE) that provides major parts of these services for the English-language dialog with an unmanned helicopter (UAV). More specifically, the system provides two-way spoken dialog using entire phrases in restricted English, combined with display of live or previously recorded video and the possibility for the operator to point into the video.

In the course of designing this system we have identified a number of specific issues that will be important in any system of a similar kind. The most important issues are:

- Linguistic expressions and gestures that are used for referring to points, areas, trajectories and moving objects in the video.
- Synchronization between actual time, time referred to in the spoken interaction, and time of recording of presently displayed video during playback.
- Markup of video frames, allowing the dialog system to relate positions on a video frame to positions in the physical or simulated world of the robot's environment.
- Linguistic expressions that refer to the passing time in which the actual dialog takes place, including the impact of the time that is defined by the playing video, on the conduct of dialog.

Besides these specific issues, the overriding issue of system architecture is important and non-trivial. A system of this kind is by definition very

heterogeneous and requires the combined operation of different subsystems of very different character; in terms of design it is much more than the sum of its parts.

Therefore, although robot dialog is indeed an interesting topic for research on dialog systems, it can not be treated merely as an extension of other kinds of dialog. The purpose of the present article is to identify and discuss additional, sometimes extralinguistic aspects that must also be taken into account. We address general architectural issues for such multimodal systems, as well as the first three of the four specific problems mentioned above. The article is written from the background of our actual RDE system and the experience from developing it. The fourth issue above is also very important but will not be addressed in the present article.

2 The Robotic Dialog Environment

2.1 Outline of Architecture

The WITAS RDE software system (Robotic Dialog Environment) consists of three subsystems that in turn have several parts:

- An *Autonomous Operator's Assistant*, AOA, consisting of two parts: a *Speech and Graphics User Interface*, SGUI, and a *Dialog Manager* in a broad sense of the word.
- A *Robotic Agent* consisting of a *Robotic World*[1] that can be either the actual UAV system and the world it is flying in[2], or a simulator for this, and a *Video Server* that provides the channel from the UAV's on-board video camera to the AOA.
- A *Development Infrastructure* that provides the services that are needed for the development, demonstration, and validation of the dialog system.

An earlier version of the Dialog Manager was described in (Sandewall et al., 2003). The subsystems and parts of RDE communicate by message-passing and video-flow. In particular, the Dialog Manager and the SGUI communicate using KQML-like messages that in most cases contain a phrase in natural English, together with some protocol information. Messages from SGUI to dialog manager may also contain several alternative interpretations of a given phrase from spoken input.

The SGUI manages both an interface on the screen of the laptop or tablet that is used by the operator, and the audio input and output using a headset. At present we are using the Nuance[3] system for input and a choice of several alternatives, such as Festival[4], for the spoken output. In addition, the SGUI passively displays the video that is passed to it from the video server, while the video server in turn is actively controlled by the dialog manager. The SGUI also interprets the gestures that the operator makes on still images and on the moving video. Its interpretations of these gestures are passed to the dialog manager.

The dialog manager receives English phrases in textual form, and produces appropriate responses that are sent back to the SGUI to be pronounced. Apart from standard command and query behavior it contains managing multiple threads. Several versions of the dialog manager exist; please refer to (Eliasson, 2005) for recent work on this topic in our project. The dialog manager also receives messages representing the SGUI's interpretations of the user's gestures on the screen. Its interpretation of these gestures in combination with the language input results in two kinds of requests: helicopter operation requests that are sent to the Robotic World, and display requests that are sent to the video server, which in turn directs the requested video stream to the SGUI.

Both the robotic dialog situation per se and the integration with the video flow have a significant influence on the design of the dialog manager. Dialog with a robot results in multiple threads in the dialog, since both events in the robot's environment and policies requested by the operator may lead to initiatives from the robot's side in that dialog. The fact that the robot moves and acts in real time imposes real-time constraints on the dialog.

[1] We reserve the term 'environment' for the software system, and use the terms 'robot world' and its 'surroundings' for the place that the robot is in.

[2] See subsection 6.1 for additional details about the WITAS UAV system.

[3] http://www.nuance.com/

[4] http://www.cstr.ed.ac.uk/projects/festival/

Finally, the existence of a video stream concurrently with the dialog and the possibility of referring from language to video means that the dialog manager must be consistently time-aware.

Additional information about the WITAS RDE can be found via the WITAS website at
`http://www.ida.liu.se/ext/witas/`

2.2 The Real-Time of a UAV

Any robotic dialog system must take time into account, but this does not necessarily mean that everything must happen very fast. In fact, one of the observations when we started experimenting with UAV dialog scenarios was that often there is plenty of time. If it takes 20 seconds for the UAV to fly from point A to point B, then it may not matter so much whether a particular vocal response is made in two seconds or in three. In the end, there are some situations where very fast response by the dialog system is a high priority, and there are others where the system must instead 'pass the time' and indicate to its operator that it is still there while operator and dialog system are jointly waiting for the UAV to finish a particular task. The dialog system must be able to adapt to different real-time requirements and to switch gracefully to higher-priority communication tasks when needed.

2.3 Varieties of Video Input

In principle, the scene that the robot is facing can be presented either directly, using video obtained from a video camera that is mounted on the robot, or using virtual reality based on the combination of a world model and sensors mounted on the robot or in the environment. Our present system uses a composite video signal which is obtained, during actual flights, from the UAV's video camera. During simulations we use archived videos from earlier flights with our project's UAV. (Animated 'virtual reality' in closed-loop simulation is being implemented at present to serve as a development tool).

The real or synthesized video is recorded from a video camera that may sometimes be directed straight down during flights, but which may often be directed at an angle against the vertical. The coordinate transformation between a video frame and the map is therefore non-trivial and varies with time.

One particular facility in our system has turned out to be very important, namely the use of *playback*. Video that is received from the UAV is directed to the video server that is able to both forward it to the dialog system, and to accumulate it to its archive. Correspondingly, the dialog system is able to request both current video and playback from a particular time from the video server. This facility is important for several applications, but it has also been very helpful in the development work since it provides a natural way of integrating previously recorded video into simulation sessions.

3 Requirements and Methodology

3.1 System Aspects

The ultimate test of a system for dialog with a UAV is of course to carry out those dialogs during actual flights. However, this does not mean that its development can and should be performed through a large number of tests during actual flights; doing so would be very costly and inconvenient, in particular because of the safety arrangements that must surround test flights. It does not even mean that the validation and evaluation of the dialog system design should be done only through test flights. Many aspects of the system design are better verified in laboratory settings. In other words, the ability to conduct dialog during actual UAV flights is a necessary but not a sufficient requirement on the entire dialog system.

For both the development and the validation of the system it is useful to identify a few distinct and, as it turns out, fairly independent subtasks:

1. Solving the equipment problems that arise when computer-based dialog is to be performed at the airfield, working outdoors: taking into account the audio disturbances from the helicopter noise and the wind, as well as the difficulties of using a laptop or tablet in full daylight; handling wireless transmissions between the UAV itself, the UAV base station, and a nomadic operator; arranging for the operator to carry the necessary computer equipment in backpack style for easy walking, arranging for spectators to be able

to hear and see the multimedia dialog, and others more.

2. Implementing the software for interfacing the dialog system to the UAV control system, so that the dialog system can receive sensor information from the UAV on the appropriate level and also send commands to the UAV.

3. Implementing a simulator that interfaces to the dialog system in the same way and across the same interface as the actual UAV does.

4. Implementing the interactive video subsystem in such a way that it can be run both with video that arrives in the course of the session (closed loop) and with previously recorded video that has been archived on the video server.

5. Implementing a dialog system that is able to operate in a laboratory setting, using simulators, video servers, etc.

6. Integrating the above into a system that convincingly demonstrates well functioning dialog during actual flights.

Implementations of all these tasks were completed and integrated in time so that the dialog system could be demonstrated as a part of the main WITAS demo in October, 2003 in the presence of an international evaluation committee. However, this was done using an early version of the dialog system and a very early version of the video system that did not provide for gesture input. Considerable additional development has been done since the main demo but exclusively in a laboratory setting.

3.2 How Often Should we Fly?

The degree of interdependence or independence between the tasks mentioned above is an important question with respect to the development methodology. Concretely speaking: given that we have verified in late 2003 that our early dialog system worked together with the flying helicopter, and given that we have continued to develop and extend the system using a simulated robotic agent, how often do we need to test the dialog system in actual test flights in order to convince ourselves, and our colleagues, that the entire system is viable and that the proposed design is to be considered as valid?

The artificial intelligence community is traditionally skeptical towards simulations, and many ground-robot projects work with a tight testing loop. It is frequently argued that simulations do not (or can not) capture the full variability of what may happen in real-world situations, which suggests that test runs are needed continuously during the development work.

This argument does not automatically carry over to the case of a UAV robot, however. To begin with, every flight experiment is fairly complex and requires considerable preparation due to the complexity of the equipment and the obligatory safety measures, so that the overhead of working with very frequent tests would be forbidding. Secondly, the world of flying objects is very structured anyway. The possibility of "a lot of unexpected things happening" is just not permitted; civil aviation authorities would certainly not allow these devices for general use if that were the state of the current technology. The UAV per se must be strictly designed and strictly modelled and its conformity to the model must be validated stringently. Under these conditions it is natural that the development of a dialog system can largely proceed in the laboratory setting, using a simulation system that correctly models the possible and anticipated behaviors of a correct UAV, including its possible fault conditions. The verification that the dialog equipment is functional in the outdoor setting at the airfield must also be done, or course, but to a large extent it can be factored out as a separate issue.

3.3 Obtaining Information from Sensors for the Dialog

We have now argued that the ability of the dialog system to give commands to the UAV during actual flights does not need to be tested so often, and that most of the time it can be safely replaced by simulations, provided that consistency of interfaces and other elementary software discipline is applied. Unfortunately, the same does not apply for the information from sensors, and in particular for the interpretation of video input. In principle,

the dialog system should rely on the capability of the on-board UAV system to interpret the video signal in combination with other sensor data, providing it with the information about the Robotic World that it needs for the dialog.

In practice, however, the ability of the on-board WITAS system to provide this information is fairly restricted. If the dialog is restricted to those topics that are possible with the actual sensor-provided information, then it will be quite limited. Conversely, it has been easy to extend the dialog system so that the dialog can also cover many topics for which the required sensor information is just not available.

This situation can be met in a number of different ways. One possible reaction may be to postpone the work on the dialog system for a number of years, until the sensor information has become available. This is the only possibility if one insists that only those results that have been demonstrated in actual flights are of interest. However, applying the principle of concurrent engineering, it is arguably better to proceed with the development of the dialog technology while using prerecorded video (realistic, but sacrificing the closed control loop) with manual interpretation, as well as a simulator for the Robotic World (closed control loop, but virtual reality instead of real video) as substitutes for testing during actual flights. It is not too difficult after all to define a plausible interface for the connection between the dialog system and the forthcoming video interpretation system, and both sides may of course participate in specifying the interface between them.

4 World and Video

4.1 Ontology of the UAV Domain

The surroundings of our UAV is defined to be road traffic phenomena on the ground. Other aerial vehicles besides the UAV itself are not considered, and the ontology for ground phenomena is based on roads, vehicles that move along those roads, road crossings, buildings, persons, and a few other major types. The vehicle and building types are subdivided into subtypes, and they may have named parts such as the roof of a building. Objects of these types as well as their parts may be characterized with elementary properties, such as color and building material. There are the obvious actions for the UAV: take off, land, fly to point X (described as e.g. a building, a street intersection, or a person), follow vehicle A, fly along road R, and so on. Similarly for the observed ground vehicles there are actions such as ”arrive at point X”, ”overtake vehicle B”, and so forth.

This ontology and repertoire of phrases was first developed for vocal-only communication. As graphic interaction was added, we decided to begin with the following four types of gestures:

- Indicate a particular point in the image, for example for a fly-to command
- Indicate a particular area in the image, for example for a command to survey the area or to not fly over it
- Indicate a particular trajectory in the image, for example a segment of a road that the UAV is to fly along, or patrol back and forth
- Indicate a particular vehicle or other moving object that is part of a query or command to the UAV, for example that the UAV should catch up with it.

There are more usages of these gestures than one may notice at first. For example, the trajectory gesture is also useful for specifying the likely current or past position of a particular ground vehicle that one wishes to designate.

Gesture input is made using the touch screen of a tablet, or using a mouse on a conventional screen[5]. The gesture part of the SGUI interprets the movements of the pen or the mouse, and attempts to classify the input according to these four cases. The gesture type and the position and size parameters characterizing it are sent to the Dialog Manager. The gesture input is sometimes ambiguous, however, and it is then necessary to combine it with the spoken input in order to make the correct analysis. In such cases the SGUI sends the list of the alternatives to the Dialog Manager and allows it to make the choice.

[5]We acquired touch-screen tablets for this purpose but found that for development purposes it was more convenient to work with a mouse and an ordinary screen.

4.2 Synchronization issues

Consider a particular time when the user indicates an item in the live video and utters an accompanying phrase. The time of speech and the time of the gesture are used to connect those two speech acts, and they are therefore recorded independently. Their contents and timestamps must be stored, since there are situations where the later dialog makes reference back to one or the other of them. Furthermore, if the video is in playback mode so that the interaction refers to an earlier time then additional timepoints are involved.

The gesture only specifies points and figures in the coordinate system of frames in the video; it must be translated into the corresponding coordinates in the physical world, from which one can also derive what object is being referenced, for example a building, or a vehicle. It is therefore important that the time of display of a particular frame can be related to the exact *time of recording* of that frame. For this purpose, our video server puts a timestamp into every frame that arrives to it from the video source[6]. This timestamp is in the video frame itself, so that timestamped video can be archived and forwarded using standard video formats, and it is insensitive to the video encoding methods. When the SGUI interprets the input gestures of the user, it identifies both the gesture itself and the timestamps of the successive frames where the gesture was made.

4.3 Markup of Video Frames

Besides accounting for time, the system must also account for the coordinate transformation between the surfaces of the successive video frames on one hand and the Robotic World on the other. During actual UAV flights this information must come from the video interpretation and other sensor data interpretation that is done in the UAV itself. For archived videos from previous UAV flights it is possible to add it more or less manually, and for simulations with adjoining visualization these parameters can be generated as a by-product of the visualizer.

[6]Other approaches have been studied, in particular using the Microsoft "media time", but they were found not to give enough accuracy.

To be precise, there are two tasks that our system expects the data analysis in the UAV to perform: relating each frame to the proper coordinates in the physical world, and identifying the positions of moving objects, such as road vehicles, in the successive video frames. In fact, all objects in the world that one may wish to refer to in the user-system interactions except stationary objects that are known to the system's geographical model, must be recognized and reported by the data analysis system.

The information about each video frame constitutes a *markup* for that frame. In on-line mode the markup will be generated continuously by the data analysis system or the simulator; in playback mode it is possible to compute and archive the markup beforehand so that it is available when needed. In our case we worked almost exclusively with playback and archived markup, except for one occasion where a demonstration of on-line use was made. During that demo we used persons as "image analyzers" that produced the markup in real time by looking at the video and tracking reference points on their screens.

The markup sequence is parallel with the video frame sequence in the sense that each frame has its own markup. However, it is not necessary to send a continuous flow of markup information from the video server to the SGUI, since most of the time it would not be used. Instead, when the SGUI receives a gesture into the video being displayed, it identifies the timestamps of the frames being pointed into, and requests the accompanying markup of those frames from the video server.

4.4 Preliminary Gesture Interpretation

The structure of the message flow and the responsibilities of the respective software modules should now be clear. One specific practical point deserves to be mentioned, concerning the disambiguation of the input gestures. Each gesture is assumed to be in one of the four types mentioned above, and the exact choice of gesture type is sometimes dependent on the accompanying phrase. For example, a gesture showing three-quarters of a circle may either designate an area or a trajectory. In principle, it should therefore be necessary to first send incoming phrases through

language processing in the dialog manager before the gesture can be interpreted.

On the other hand, some interactive situations may require very rapid response. We have therefore adopted the following shortcut. The main part of the SGUI anyway receives input sentences in written form from the speech analyzer. The parsing of these sentences takes place in the dialog manager. However, in many cases the accompanying sentences are very simple, such as ”fly here”. Therefore, the SGUI is equipped with a list of standard phrases that it recognizes immediately by itself, and if a gesture is accompanied by such a standard phrase then its type can be decided at once.

Furthermore, the SGUI is defined to make such speech-gesture combination even in cases where the interpretation of the speech remains uncertain. It then sends the available information about the speech input, its assumptions about that speech input, and its resulting interpretation of the gesture to the dialog manager. If the latter should decide that the SGUI's interpretation of the speech was incorrect then it sends a message back to the SGUI asking it for a new interpretation of the gesture based on the alternative classification.

It might be argued that this solution is an artifact of a too strong separation between the SGUI part and the Dialog Manager part of the system, and that a more ”agent-oriented” architecture with many processes that send messages back and forth would have been a better way of handling the problem. We do not share that opinion: the present implementation does not have any particular disadvantage, and the separation of SGUI and Dialog Management as two distinct blocks has a performance advantage since it allows the SGUI to be implemented with strict consideration of real-time constraints while the Dialog Manager can give priority to the symbolic side of the computation.

5 Actual-Time Considerations

One of the interesting issues for this system is that several aspects of time must be taken into account in an effective way. We have already mentioned the connection between time of display and time of recording of the video, which is administrated by the SGUI using the timestamps. In addition, there are a number of computational and transmission delays that must be properly accounted for. The time where a spoken input phrase is concluded is not necessarily the same as the time when the pointing gesture is concluded. The times when those two speech acts have been interpreted in their respective computations need not coincide either. These aspects have been taken into consideration throughout our system, for example by keeping track of exact time of speech.

At present our system does not combine speech and gestures for output to the user, but only for input from the user. Output is speech only, or text only if the speech facility is disabled. When we proceed to two-way speech and gesture combinations it will be even more important to have full control of actual time, and for the system to choose its speech acts within the limitations of available time.

6 Related Work

6.1 The WITAS Projects

WITAS, the Wallenberg Laboratory for Information Technology and Autonomous Systems, is engaged in goal-directed basic research in the area of intelligent autonomous vehicles and other autonomous systems. Its main project focuses on the development of an airborne computer system that is able to make rational decisions about the continued operation of the aircraft, based on various sources of knowledge including pre-stored geographical knowledge, knowledge obtained from vision sensors, and knowledge communicated to it by data link.

The major part of the project addresses the UAV Technologies and is described e.g. in (Doherty et al., 2000; Doherty, 2004). The other part of the project concerns robotic dialog, in particular between a human operator and a UAV. Dialog activities in WITAS were organized as a project at Stanford during 2000-2002 and as a new project in Linköping since 2002. The work reported here is from the WITAS-Linköping Dialog Project.

6.2 Other Dialog Systems

The present article has addressed multimodal dialog with a robot using spoken language and

live video. Many earlier projects have addressed robotic dialog without the graphic modality or with still-image graphics without the live video aspect.

The KANTRA system by Lueth, Laengle, et al (Lueth et al., 1994) was a relatively early system providing natural-language communication for commanding a mobile ground robot. The report does not mention any use of graphics in this system.

Multimodal dialog systems that combine spoken language with still images include in particular the SmartKom (Reithinger et al., 2003; Herzog et al.,) and MATCH (Johnston et al., 2002) systems. These systems do not address robotic dialog since their task is to provide information for a mobile operator. The WITAS-Stanford dialog system of Lemon, Peters, et al (Lemon et al., 2002) is still one of the few published examples of a multimodal robotic dialog system, but its graphic capability is limited to specifying a point in a fixed, maplike aerial photograph. It therefore does not consider the problems of dealing with moving video and the resulting real-time and other issues.

Acknowledgements

The Video Server part of the RDE was designed and implemented by Björn Husberg, replacing an earlier video-archive system with fewer facilities. The extended SGUI that provides support for the Video Server, allows the user to point into the live video, and performs the necessary coordinate transformations etc. was designed and implemented by Hannes Lindblom as a M.Sc. thesis project. Erik Sandewall directs the WITAS Dialog Project including the work described here.

The entire WITAS RDE is the result of joint work with major contributions by Malin Alsén, Peter Andersson, Karolina Eliasson, Susanna Monemar and Tobias Nurmiranta as well as additional M.Sc. students, besides the present authors.

The support of the Wallenberg Foundation for the research reported here is gratefully acknowledged.

References

Patrick Doherty, Gosta Granlund, Kris Kuchcinski, Erik Sandewall, Klas Nordberg, Erik Skarman, and Johan Wiklund. 2000. The witas unmanned aerial vehicle project. In *Proc. 14th European Conference on Artificial Intelligence*, pages 747–755.

Patrick Doherty. 2004. Advanced research with autonomous unmanned aerial vehicles. In *Proc. 9th International Conference on Knowledge Representation and Reasoning*.

Karolina Eliasson. 2005. Integrating a discourse model with a learning case-based reasoning system. In *Proceedings of DIALOR-05*.

G. Herzog, H. Kirchmann, S. Merten, A. Ndiaye, and P. Poller. MULTIPLATFORM testbed: An integration platform for multimodal dialog systems. In H. Cunningham and J. Patrick, editors, *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*.

Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marily Walker, Steve Whittaker, and Preetam Maloor. 2002. MATCH: An architecture for multimodal dialog systems. In *Proc. 40th Annual Meeting of the Association for Computational Linguistics*, pages 376–383.

Oliver Lemon, Alexander Gruenstein, and Stanley Peters. 2002. Collaborative activities and multitasking in dialogue systems. *Traitement Automatique des Langues (TAL)*, 43(2):131 – 154. Special Issue on Dialogue.

T.C. Lueth, Th. Laengle, G. Herzog, E. Stopp, and U. Rembold. 1994. KANTRA human-machine interaction for intelligent robots using natural language. In *Proceedings of the 3rd IEEE International Workshop on Robot and Human Communication, RO-MAN'94*, pages 106–111.

Norbert Reithinger, Gerd Herzog, and Alassane Ndiaye. 2003. Situated multimodal interaction in SmartKom. *Computers and Graphics*, 27(6):899–903.

Erik Sandewall, Patrick Doherty, Oliver Lemon, and Stanley Peters. 2003. Real-time dialogues with the WITAS unmanned aerial vehicle. In Andreas Günter, editor, *Annual German Conference on AI*, pages 52–63.