Presented at Collaboration Technologies and Systems
Collaborative Robots and Human Robot Interaction Workshop 2012

A Multi-Modal Intelligent User Interface for Supervisory Control of Unmanned Platforms

Glenn Taylor, Richard Frederiksen, Jacob Crossman, Michael Quist, Patrick Theisen
Soar Technology, Inc.
Ann Arbor, MI, USA
{glenn, rdf, jcrossman, quist, patrick.theisen}@soartech.com

Abstract— **Typical human-robot interaction (HRI) is through tele-operation or point-and-click interfaces that require extensive training to become proficient and require the user's complete attention to operate. For unmanned platforms to reach their full potential, users must be able to exert supervisory control over those platforms. This requires more effective means of communication in both directions, including high-level commands given to the vehicle and meaningful feedback to the user. Our aim is to reduce the training requirements and workload needed to interact with unmanned systems effectively and to raise the level of user interaction with these systems so that supervisory control is possible. In this paper we describe an intelligent user interface, called the Smart Interaction Device (SID) that facilitates a dialogue between the user and the unmanned platform. SID works with the user to understand the user's intent, including asking any clarification questions. Once an understanding is established, SID translates that intent into the language of the platform. SID also monitors the platform's progress in order to give feedback to the user about status or problems that arise. We have incorporated multiple input modalities, including speech, gesture, and sketch as natural ways for a user to communicate with unmanned platforms. SID also provides multiple modes of feedback, including graphics, video and speech. We describe SID's architecture and some examples of its application in different domains.**

Keywords: HRI Multi-modal Interfaces, HRI Applications, Supervisory Control, Interaction Components

I. INTRODUCTION

Unmanned platforms that have been deployed in the field for military use tend to be remotely controlled through an Operator Control Unit (OCU). OCUs typically allow for low-level tele-operation (e.g., joystick control) or point-and-click-style interfaces (e.g., menus, buttons) that require fine-grained tasking. These systems currently require extensive training. US military specialties in unmanned platforms require roughly 7 months of training in classroom and practical application courses. Even with all this training, human error is still cited as the dominant cause of 80% of all UAV crashes [1]. On top of this, typical OCUs require the full attention of the user to task the robot and monitor its progress [2]. This takes away from the users' ability to maintain their own situational awareness or perform other tasks.

A current trend in human-robot interaction research is toward *supervisory control*, where the user's job changes to one of managing the robot rather than controlling it inch by inch. Sheridan [3] outlines three requirements for supervisory control: autonomy, high-level commands, and situational awareness. This occurs routinely in human teams that include a leader and a subordinate: the leader gives a high-level task, and the subordinate is given some autonomy to perform the task while at the same time keeping the leader informed of progress or problems. For supervisory control in human-robot teams, the same conditions must hold: the robot must have autonomy to perform at least some tasks, must accept high-level tasking (far above tele-operation), and must provide feedback to the user to help maintain the user's awareness. Autonomy in unmanned platforms has been slowly increasing over time, but OCUs have tended to lag behind in terms of providing high-level tasking, providing effective situational awareness to the user, and in usability generally.

Over the last few years, we have been developing the Smart Interaction Device (SID) whose purpose is to enable a user to work at the level of supervisory control using natural ways of interacting with the robot. Our approach with SID follows Sheridan's three requirements. SID takes the form of an intelligent user interface that facilitates a high-level, multi-modal interaction between a user and a robot. SID leverages the autonomy on the platform, but also adds some intelligence in the user interface itself that helps to translate high-level user commands into robot commands. SID also helps maintain user situational awareness by providing feedback to the user, either by request or based on expected protocols. The result is that the user can spend less time minding the unmanned vehicle and more time paying attention to his or her surroundings, working on other tasks, or managing multiple vehicles.

In the rest of this paper, we frame our approach in the context of other work in the field, describe the system in detail, and describe some applications of the system in a range of real and simulated unmanned platforms.

II. PRIOR WORK

As mentioned, most OCU systems today are tele-operation or point-and-click interfaces. Some of the more advanced HRI systems in military robot research have experimented with speech interfaces, but typically these are characterized by one-shot interactions: the user issues a command and the system performs the task, perhaps with minimal acknowledgment [4]. This interaction is still fairly close to the primitives of the platform, and does not include the kinds of dialogue that one

This work was partially funded by the Office of Naval Research.

would see in similar human communication. In contrast, our goal is to approach the language that people use when communicating to each other in these situations.

Academic research has gone much further in terms of interactive robotic systems. There has been extensive work on dialogue systems for robots, including work in direction-giving [5] and spatial language [6]. Most similar to SID in terms of architecture is the WITAS dialogue system [7]. WITAS focused on UAV operations, specifically multi-threaded conversations and coordination of joint activities such as target tracking. SID shares much in common with the WITAS architecture, including similar core components for dialogue management and task representation. However, rather than the multi-agent paradigm used by WITAS, we take a knowledge-based system approach, using the Soar cognitive architecture [8, 9] as the basis for SID. Also, whereas WITAS supported speech with mouse clicks to indicate positions for a UAV to fly, SID supports other modalities natural to these domains such as sketching and gesture simultaneous with speech.

The work of Oviatt and her colleagues is well-known among multi-modal interfaces, exemplified in systems such as QuickSet [10, 11] that combines speech and pen-based input. In SID, we borrow from the underlying multi-modal fusion algorithms in this work, as described by Johnson et al [12]. We implement these algorithms within the Soar architecture as a way to perform very fast unification and to leverage multiple sources of knowledge both for the fusion of input sources and for the resolution of multiple interpretations.

There has been a recent boom in "natural interfaces" made possible by inexpensive sensor systems like the Microsoft Kinect. This has spawned numerous research and demonstration systems that show simple gestures like "raised/lowered arm" to trigger a UAV to take off or land (for example, [13]). Rather than inventing new protocols, our approach has been much more user-focused in that we have worked to discover how our expected users interact in today's operations and use that as a starting point for understanding how users would like to interact with unmanned platforms.

III. Background: Natural Interaction

Our goal in building HRI systems is to provide natural ways for users to interact with unmanned platforms. What constitutes natural interaction? We look to human interaction to help answer this question, and the answer varies depending on the domain and on the situation. For face-to-face contact, people naturally use speech and gesture to communicate. For non-face-to-face communication, gesture becomes less useful and speech or text/chat more prevalent. People often use shared artifacts like maps to refer to or even drawn on, especially in spatially oriented domains. Furthermore, mixing modes together has been shown to be natural and often more efficient in spatially oriented domains [14]. In addition to leveraging the work of others, we have conducted our own interviews and observations of our expected users performing the kinds of tasks that the military hopes will soon include unmanned platforms, in naturalistic settings and in Wizard of Oz studies. We expect that if robots can interact in ways similar to how people interact in these settings, those robots will be easier to use and require less training for users.

Another aspect of naturalness comes from dialogue itself. Human communication happens over time, by way of multiple communicative acts. Dialogue is a way to mitigate the frailties of human communication so that a task gets done. Dialogue is so important in human communication that it is embedded in military communications as a matter of doctrine: a speaker issues a command, and the hearer repeats it back to make sure everything was understood properly. This process helps increase the robustness of communication and understanding between participants, and is an essential element to carry over to human-machine interfaces.

Because people use dialogues regularly, they expect certain forms of interaction, whether it is in the back and forth protocols of command-and-acknowledge, in the use of clarification questions, or in the brevity afforded by using referring expressions (e.g., pronouns) to minimize the amount of language needed. Especially in supervisory control in human teams (e.g., lead/subordinate roles), there are expectations of acknowledgments and feedback as tasks progress. When these expectations are broken, situation awareness and task completion suffers. Dialogue has a secondary effect of decreasing the workload of participants. By allowing references to earlier parts of the conversations, there is less work for the speaker. Hearers also expect this kind of brevity, and have to work harder when it is not present in the dialogue [14]. To be considered natural, HRI systems must meet these human expectations about conversations.

The next sections describe our technical approach to making natural interfaces for supervisory control, in the form of our Smart Interaction Device (SID), and the applications of SID to various robotic platforms.

IV. Approach: Intelligence in the Interface

Our general approach to building natural interaction systems for unmanned platforms has been to make the user interface smarter, in the tradition of intelligent user interfaces [15]. The user interface is "smart" in the sense that it lets a user specify higher-level input and it works to understand that input in user terms, then translates that input into robot terms. The system also interprets user inputs across multiple input modalities, and combines them to a single unified meaning. The goal is to let the user think and communicate at the level of supervisory control tasks without worrying about the details of how a particular unmanned platform has to be told to perform that task.

To accommodate the kinds of high-level interactions that occur in these domains, an intelligent user interface must have some core capabilities for managing dialogue with a user, translating the user commands into robot-level tasks, and monitoring the progress of the robot to ensure that it is proceeding correctly and to provide feedback to the user. As shown in Figure 1, SID consists of a domain-independent set of core modules ("SID Core") that are designed to fulfill these requirements. SID additionally includes input-device-specific, domain-specific and platform-specific layers, each of which may be customized to a particular application. Different ways of

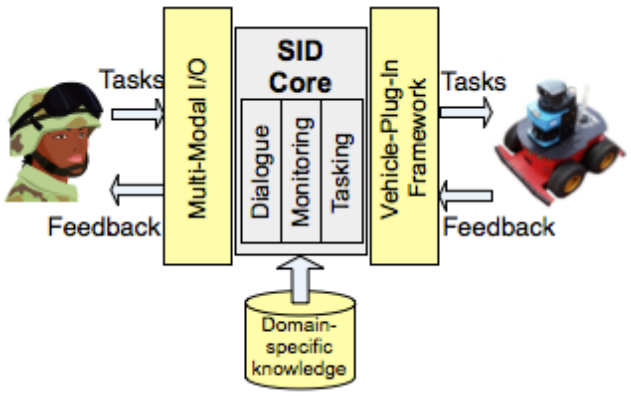

Figure 1: High Level Architecture of the Smart Interaction Device (SID)

interacting with the robot and different robot capabilities necessitate different user-facing and robot-facing software interfaces.

Domain-*independent* knowledge in SID consists of rules for how to manage dialogues, how to break a task down into finer tasks, and how to maintain situational awareness about a platform. Domain-*dependent* knowledge consists of rules for how translate a particular user command to a particular platform-level command, or how to interpret particular input modes in a given domain.

As illustrated in Figure 1, the main behavior components in SID Core are the Dialogue Manager, and Monitor, and the Tasker:

- The **Dialogue Manager** is responsible for all interactions with the user, including interpreting user intent, asking clarification, and providing feedback to the user about the status of the robotic platform
- The **Monitor** is responsible for maintaining awareness of the state of the robotic platform, especially as it pertains to fulfilling the user's intent
- The **Tasker** is responsible for translating user's intent into commands for the robotic platform.

These modules work over a common shared memory representation that maintains the state of the vehicle, the state of the conversation, and the current plan assigned to the vehicle. This allows all modules in SID Core to have access to the task definition and progress as task execution proceeds. This shared representation also allows SID to engage in a dialogue with the user before, during, and after task execution.

From a functional perspective, SID is a goal-directed system composed of multiple behavior modules, each responsible for the different capabilities mentioned above. This is somewhat similar to the "agent-based" approaches of some conversational dialogue systems [16, 17]. However, rather than multiple agents providing services as seen in other systems, the Soar Cognitive Architecture provides the main unifying framework for SID's behavior modules. Soar provides a set of core knowledge representations, reasoning and learning processes that can be used for building adaptive knowledge-based systems without having to build these features from scratch [9]. Soar can also be used in a multi-agent paradigm; however, here we chose to implement the behavior modules within a single Soar agent.

TABLE 1: EXAMPLE MULTI-MODAL DIALOGUES WITH SID

Multi-Modal Dialogue	Dialogue Moves
User: Move along route blue and tell me when you're done.	Directive (move); Request for information
SID: Roger, moving along route blue; will report completion.	Acknowledgement; commitment to report
(time passes while robot moves)	
SID: Okay, completed route blue.	Inform (fulfills earlier commitment)
User: Drive over to the vehicle.	Directive (move)
SID: Which vehicle? I know of two.	Request for clarification (subdialogue)
User: *That* one (while pointing)	Clarification (subdialogue)
SID: Roger, driving to vehicle 1.	Acknowledgement
User: There are enemies in *this* location (user sketches an area).	Inform
SID: Roger, rerouting to avoid enemies.	Acknowledgement; inform of new plan

These modules of SID Core are discussed in detail below.

A. Multi-Modal Dialogue Manager

Broadly speaking, the key role of the Dialogue Manager (DM) in this system is to manage the interaction between the robot and the user over time to help facilitate robot tasking and feedback to the user. Interactions may play out through a back-and-forth sequence of commands, requests, clarifications, and responses. To illustrate the kinds of multi-modal dialogues that can occur with SID, some examples are shown in Table 1.

For a user to provide a robot with high-level tasking, the Dialogue Manager must come to an understanding of the user's intent. Intent understanding incorporates different sources of knowledge, including knowledge of the mission, of the domain, and of the conversation up to that point. The same input may have different meanings in different contexts. This process also requires resolving references or ambiguities within a user's input. References might be to objects, locations, or people in the environment or to previous utterances in the dialogue, which the DM must keep track of to help to resolve. Ambiguities can occur when a user refers to an object in the environment or a previous topic in the dialogue without enough specification to narrow down to a single thing the user is talking about. If the DM does not have enough information to fully understand the input, it may request more information from the user.

By definition, dialogues occur over time, including commands, acknowledgements, answers, and even sub-dialogues to resolve problems like ambiguities. Also, in a given mission, there may be multiple conversations that occur based on how the mission plays out. Every turn that any participant takes in a

conversation is called a *dialogue move*. The DM keeps track of these conversations and the connections between dialogue moves, maintaining the temporal aspects of the moves as well as the individual threads of conversation. The DM also keeps track of unanswered questions or requests, by either the user or the system, and works to fulfill these information needs.

The basic algorithm of the Dialogue Manager is as follows:

1) Classification: infer the type kind of dialogue move from the current input and current dialogue state (based on the taxonomy of [18])
2) Attachment: identify the input's dialogue thread (is it part of an existing dialogue, or a new one?)
3) Reference Resolution: resolve any references to prior inputs (e.g., pronouns) or external references (e.g., objects in the world)
4) Clarification (as needed): resolve any unresolved problems by asking the user for clarification; relate any answer back to the original dialogue move
5) Command construction: based on the dialogue move and whole dialogue context, construct a concise description of the command for execution (including robot moves or subsequent dialogue moves such as acknowledging the command)

This process is slightly more complicated when using multiple modes of input. In SID, user inputs may come from multiple sources simultaneously, for example, through speech and gesture. SID must deliberate about what these two inputs mean independently and if they mean anything together. This is process is called multi-modal fusion [10]. We derive our approach to multi-modal fusion from frame-based unification approaches [12]. In SID, input from any single source is represented as a *semantic frame* that has some meaning on its own, but may not be sufficient to understand the total input. A semantic frame is a typed feature structure that is abstracted from the particular input device but still contains the core information of the input. Unification rules serve to combine inputs from different sources in meaningful ways that are constrained by factors such as contents and timing. Our unification algorithm uses Soar's Rete algorithm [19] to efficiently handle the resolution of variables and constraints.

Consider a command like "Move over there," spoken with a simultaneous pointing gesture. The semantic frames for these individual modes are illustrated in Figure 2. The spoken utterance contains a "move" command, with a timestamp and an input source, but not enough information to tell where to move (lacks a *location*). This is illustrated in Figure 2 (top). Similarly, a pointing gesture representation includes parameters such as the type of gesture (pointing) of where the user is pointing, but without a command associated with it (shown in Figure 2 (bottom)). SID does not have enough information to do anything with either one of these semantic frames individually, so the system considers integrating the two to see if they make sense together.

Multi-modal fusion happens by way of domain-specific unification rules similar those described by Johnston, et al [12]. An example of an integration rule is given in Figure 3, combining the semantic frames from Figure 2. The unification rule contains a left hand side (LHS) that specifies the components required for integration – in this example, a movement command and a gesture that specifies a direction. Part of the LHS of the integration rule includes temporal constraints that ensure the gesture occurred within the time bounds given with the command. On the right-hand side (RHS), we have the specification of the semantics for the unified command.

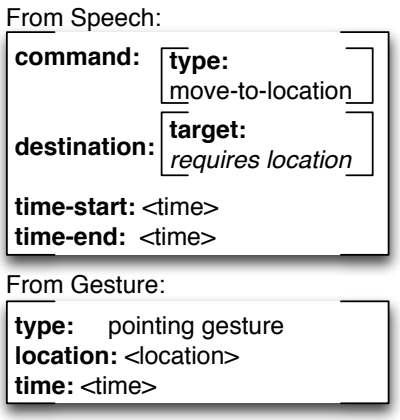

Figure 2: Typed semantic frame representations for a "Move to Location" Command (top) and a pointing gesture (bottom)

Unification in Soar happens naturally as part of Soar's rule matching algorithm. In fact, the style of integration rule shown in Figure 3 can be directly implemented as a single rule in Soar. Adding new combinations of multiple modes is simply a process of adding domain-specific rules to assert those combinations. Because there may be multiple possible interpretations of individual and combined modalities, this rule asserts a new

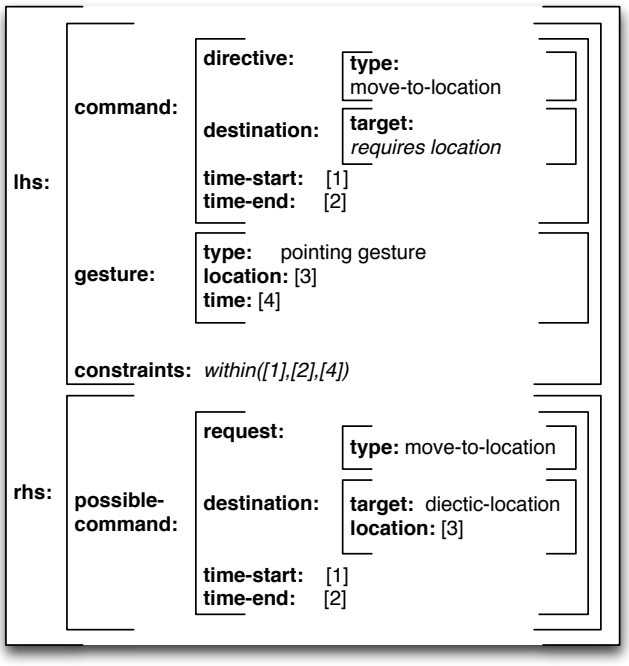

Figure 3: Semantic Frame Integration Rule for Combining a Spoken Command with a Deictic Gesture

"possible command" that might be considered along with other interpretations. In these cases, SID uses knowledge-based preferences to select between the different interpretations of the input. The selected command interpretation is then handed to the Dialogue Manager for interpretation within the context of the dialogue and the larger situation.

In addition to the process of understanding user inputs, the Dialogue Manager is also responsible for responding to user questions, asking clarifying questions, or providing status updates. To generate this feedback, the Dialogue Manager constructs an utterance using template-based generation [20] and creates a representative dialogue move in the dialogue thread to keep track of the system's participation in the conversation. This utterance is then handed to the speech generation system to verbalize.

B. Tasker

To fulfill the goal of enabling high-level tasking for supervisory control, SID must be able to translate user inputs into commands that the unmanned platform can perform. This is the job of the Tasker. This translation happens in two phases. First is a translation from the user's command into a domain-level specification, independent of the particular platform that is being used. For example, we have been working in what we call the "tactical maneuver" domain, which captures behaviors such as moving to waypoints, following routes, patrolling areas, providing acknowledgments, reporting progress, etc. Note that these include both movement tasks and dialogue acts such as communicating back to the user. The result of this translation process is a multi-threaded hierarchical task network (HTN) that represents the user's intent in a set of steps that must be performed to accomplish that intent. The multi-threaded HTN captures steps that can be performed independently (in different threads), as well as any hierarchal and temporal ordering inherent in the task. The HTN terminates in a set of primitive actions in the domain ("domain primitives"). This domain-level intermediate form is useful from two perspective of discussion with the user. It provides a description of what the robot is doing in a language closer to the user, versus the low-level messaging that occurs at the platform level. Also, since the domain specification is independent of the platform, this level of description for a task will be the same across multiple platforms.

The second phase of translation is a refinement of this domain-level HTN specification (domain primitives) to the particulars of the platform (platform primitives). Every platform will have its idiosyncrasies, whether in terms of what tasks it can perform, the protocols/languages it can communicate in, or the particular messages it accepts. Platform-specific rules expand domain primitives in the HTN into a set of platform primitives. The refinement process is specific to each platform, and is informed by an ontology that indicates the capabilities of the target platform in terms of tasks and messages. For example, for a robot that uses JAUS messaging, a *move-to-waypoint* domain primitive would eventually be translated to a JAUS *SetGlobalWaypoint* message.

In some cases, the platform cannot perform the desired domain primitive. Suppose a vehicle did not know how to follow routes, but could move to individual waypoints. In this case, the Tasker would refine the *follow-route* domain primitive to a series of *move-to-waypoint* primitives, essentially implementing the *follow-route* command within the HTN. In other cases, there may not be a possible refinement that the platform can execute; for example, acknowledging a command. In these cases, these steps in the HTN are assigned to a SID Core – namely, the Dialogue Manager (DM) for communication with the user and the Monitor for tracking platform progress.

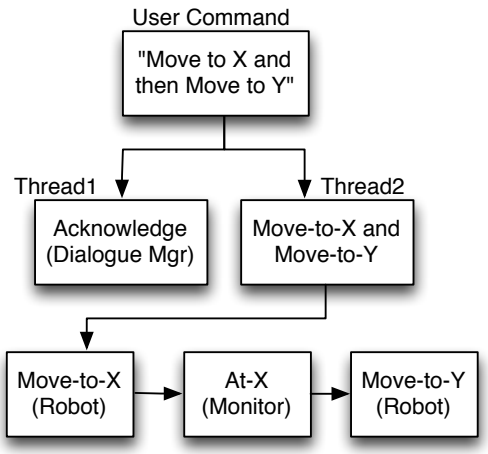

Figure 4: Example HTN with assigned roles for each node

An example HTN is shown in Figure 4. At the top is the command from the user (e.g., "Move to X and then Move to Y"). This command is executed as two threads, one for acknowledging the command and the other for the actual movement tasks. In this case, the robot cannot be given a multi-step movement command so must be issued two individual move-to-waypoint commands, and SID must monitor for completion of the first before giving the second.

Once the HTN has been fully refined, the Tasker turns to doling out these fine-grained commands to their assignees. Robot commands are sent to the robot via a plug-in that handles the low-level particulars of the language/networking protocols. The Dialogue Manager typically handles communication tasks. The Monitor, described next, handles tasks that relate to maintaining awareness about the robot.

C. Monitor

The Monitor's responsibility is to maintain an updated picture of the robot's status with respect to its tasking and including other basic information such as battery life or error states. Part of this is performed automatically, independent of any user tasking, so that SID itself has an update picture of the platform's status. In other cases, the HTN generated by the Tasker includes tasks specifically assigned to the Monitor to help keep track of the vehicle's progress in fulfilling the user's command. In the example HTN in Figure 4, the "At-X" step is assigned to the Monitor to have it watch for when the robot has achieved waypoint X. When that occurs, the execution can continue to the next movement command. The process of monitoring for vehicle status is also platform specific. Some unmanned platforms may generate signals indicating completion of a task, in which case the Monitor can simply watch for those signals. In other cases, the vehicle makes no such reports, and the Monitor

has to infer completion from other indicators; for example, the robot being within some range of the target.

V. Error Handling

There are myriad ways in which things can go awry when understanding the user or in performance of the task. On the input recognition and understanding side alone, a litany of different problems can occur: the speech recognition engine can completely fail to recognize the user's input; the speech recognizer might hear only a part of the user's input, so return an incorrect or incomplete parse; the user can say something outside the grammar; the gesture recognition system can fail to recognize a gesture or user can forget to gesture altogether; the user can accidentally give confusing inputs such as "follow this point"; the user can give ambiguous inputs that cannot immediately be resolved into a precise command for the unmanned platform; etc. On the task execution side, there are analogous problems that arise with autonomous systems: a path-planner may fail to find a path to perform a task; the vehicle may work its way into a corner that it cannot get out of; etc.

While we cannot anticipate every possible problem, we take a staged approach to dealing with some of these errors in input processing and execution. SID tries to use as much as it can from the user input to understand what the user meant, including the context from the conversation up to that point. Where it does not have enough information to construct a robot task, it will ask for clarification, using what it does know to compose a meaningful question back to the user. For example, if the user says "Go to this point" without indicating a destination, SID knows the command (go-to-point) but not the target, so can ask for clarification with "Which point do you want me to go to?" Another stage in error handling occurs when resolving user inputs to SID's knowledge of the environment. For example, if the user refers to a location that the system does not know about, it can reply with, "I don't know about that location. Where do you want me to go?" Or if the user's input is ambiguous and can resolve to multiple targets, SID essentially asks, "I know of several like that. Which one do you want me to go to?" Finally, if SID cannot make any sense of the user's input – for example, from the recognizer not returning a parse to parsing only incomplete input, SID falls back to a generic "say again?" message. In this case, the user must simply re-state the input.

These are fairly simple strategies so far, and the system tends to fall to the "say again" more than we would like. We plan to make SID more helpful in the requests back to the user, and in the types of clarifications the user can provide. We are experimenting with different speech recognizers and language processors to give SID the best possible chance of making sense of the user's input. There can also be cases where the back-and-forth with the user seems to be not making any progress; ideally, SID would be able to recognize this and switch to a more guided user interaction strategy to help the dialogue along.

The platform itself can also have problems executing the given task, for example, being unable to plan to the given destination, or getting stuck where it cannot find its way out of a corner. In the former case if the platform can notify SID that it cannot generate a plan, the SID can convey this to the user. In other cases, the vehicle may simply keep trying to find a path by exploring different routes, without ever recognizing that it cannot actually reach the destination. So far, SID does not recognize these situations, and relies on the user to recognize and step in with new instructions. In the worst case, SID provides tele-op control (either verbally – "turn around", "back up" – or with a virtual joystick on a hand-held device) to allow the user to take over the robot for a brief period to get the robot where it might be able to recover.

While these methods of a generic "say again" or tele-op are unappealing from a natural user interface perspective, they are necessary fallbacks in these kinds of systems where input recognizers or autonomy algorithms do not perform 100% of the time. Generally, our approach is on the one hand to use dialogue as the basis for communicating with the user to make the interaction successful, and other the other hand, allow the user to step in to make these low-level changes by exception.

VI. Application Domains

We have used SID in a few application domains, spanning simulated unmanned air vehicles (UAVs), real UAVs, and real ground vehicles. Physically, SID may take the form of an actual physical device (e.g., a portable tablet computer or a smartphone), or may be "device-less" in the sense that the user is not holding anything but relying on sensors on the unmanned platform. In this section, we describe two different use cases: unmanned ground vehicles and unmanned helicopters.

A. Unmanned Ground Vehicles

The tasks of interest in this use case are mobility tasks such as delivering supplies to remote users or patrolling an area. We have connected SID to two different ground robot: a P3-AT Pioneer and a robot custom-built by the University of Michigan's APRIL laboratory for the international MAGIC competition [21]. Both use different communication protocols but are capable of performing similar domain primitives, such as moving to points and following routes. For these use cases, we have used speech and gesture in face-to-face communication and speech and sketch for remote communication. The role of gesture in these cases has been solely for pointing (deictic gestures), in support of spoken utterances (for example, "go over *there*" with a pointing gesture). The robots maintain internal maps that can be labeled by the user, and which provide reference locations for speech and gesture. With SID as a facilitator, the robot has knowledge of its mission, can answer questions about its mission as it progresses, and can even interact with multiple users during its mission.

An example is shown in Figure 5 (top), using a smartphone with accelerometer and compass used both as a pointing device and as a simulated radio. We have also used a Microsoft Kinect to capture similar pointing gestures. In the first use case, there is no display for the user; all feedback from the robot is via generated speech and through the robot's own mobility. Alternatively, Figure 5 (bottom) shows a tablet computer as the SID user interface that includes a video feed and a map view as it is generated by the robot, both of which can be sketched on to communicate with the ground vehicles. Examples user inputs on the tablet display include "Follow this route" while sketch-

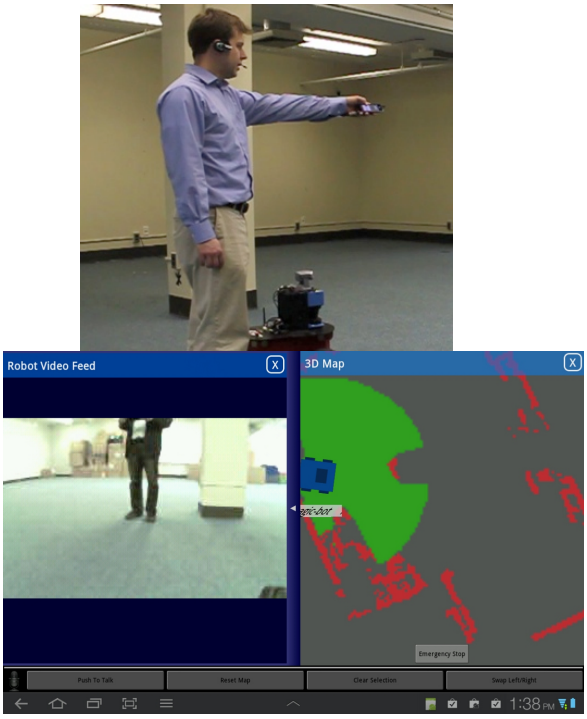

Figure 5: (top) A user gesturing while speaking to a ground robot: "Go over there"; (bottom) a tablet display with video feed and top-down map views

ing a route to follow on the map, or "Go over to this point" while indicating a destination on the video display.

B. *Real and Simulated Unmanned Helicopters*

We have also connected SID to both simulated and real rotary-wing unmanned air vehicles. With these, the focus is on remote communication using speech and sketch on a tablet computer. The task in this use case is autonomous landing at a selected landing site. Here we assume greater autonomy on the part of the aircraft – the user is not directing the aircraft so much as helping guide UAV on its last leg inbound to land safely. Along the way, the user may provide a marker indicating the landing site, or may provide information about the area surrounding the landing site such as the location of obstacles or other threats. With this information, the UAV can plan around these obstacles and, finally, choose to land.

In the simulated UAV case, we are using an open source simulation environment called SimJr we developed for prototyping autonomous behaviors [22]. Figure 6 illustrates a user interacting via a tablet computer with a simulated UAV running in SimJr. In the top pane, the user says, "There are enemies in this location" while sketching on the screen. SID fuses the two input streams together and, based on knowledge of the domain, assumes that enemy areas are to be avoided, so interprets the user input as defining a no-fly area and passes that no-fly area to the UAV planner. In the lower pane, SID indicates the constructed no-fly area on the map and provides feedback to the user about the newly constructed route around the designated no-fly area.

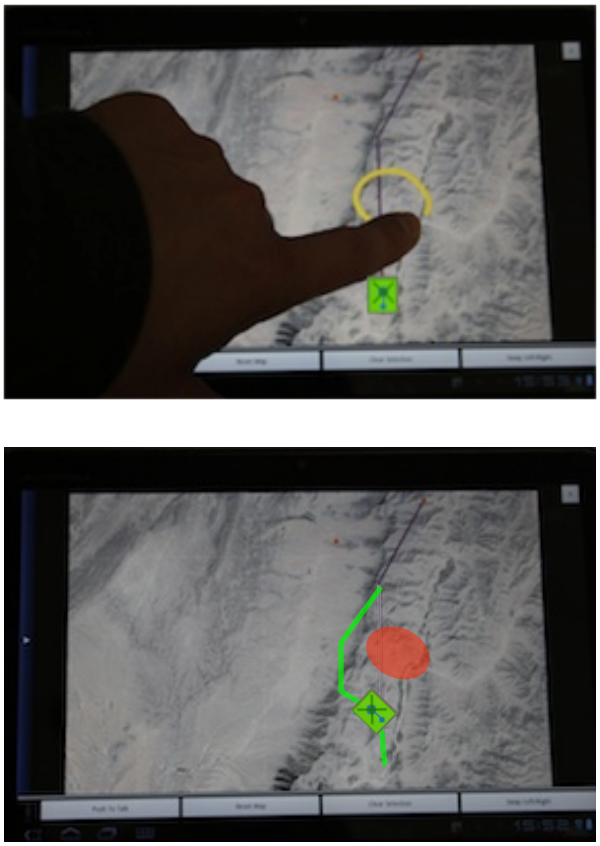

Figure 6: Example of SID instantiated with a tablet computer, interacting with a simulated helicopter.

In the case of a real UAV, we have connected to a Parrot ARDrone [23], to which we have added video processing to find particular landing site markers. Here, the interaction is remote via speech and via a marker on the ground indicating the position of the landing site. The dialogue begins when the aircraft is near the landing zone, and the user can tell the aircraft to look for a particular type of marker. When the marker is found, SID notifies the user and the user can then direct the aircraft to land. Figure 7 illustrates the UAV in action. The left panel shows the view from the aircraft's forward-looking camera, with processed video marking the found landing site. The right panel shows the aircraft about to land on the marker.

Figure 7: SID directing a Parrot ARDrone UAV to a landing site

VII. Conclusions and Future Work

We have described a multi-modal intelligent user interface called the Smart Interaction Device (SID) that allows a user to interact with unmanned platforms in more natural ways. SID

serves as a facilitator between the user and the unmanned platform, interpreting the user's high-level commands and converting them into low-level commands for the platform, and provided user-level feedback based on low-level status information from the vehicle. SID fuses a variety of input modalities (sketch, speech, gesture) to infer user's intent. SID also engages in a dialogue with the user to ensure that it understands the user; for example, asking for clarification when the user's input is ambiguous. By acting as a smart facilitator, SID moves unmanned platforms in the direction of supervisory control, making them easier to use and freeing up users to perform other tasks. SID is related to systems like WITAS [17] and QuickSet [11], but explores how a cognitive architecture (Soar [9]) can be used as the basis for an intelligent user interface, and broadens the modalities of interaction from those earlier systems to include speech, sketch, and gesture for interaction with different types of unmanned platforms.

We have not yet completed a formal evaluation of SID to gauge quantitatively its performance in terms of the goals suggested earlier: natural interaction, increased user situational awareness, and supervisory control. However, a number of the features we have demonstrated are necessary precursors to achieving these goals, including using multi-modal dialogue without requiring the user to be heads-down in the display. While we do not have firm results yet, some pilot studies indicate that dialogue with SID helps to reduce the amount of work the operator must do and the amount of stress that the operator feels in performing a task. We have had informal evaluations with representative users to get feedback on the system, and have been able to roll in some suggested improvements. Our next steps in this work are to perform a full usability evaluation and further iterate on the dialogue strategies and interaction modalities to make a more robust, user-friendly system.

REFERENCES

[1] R. Nullmeyer, *et al.*, "Birds of Prey: Training Solutions to Human Factors Issues," in *IITSEC*, Orlando, FL, 2007.

[2] H. A. Yanco and J. Drury, "Where am I?" Acquiring situation awareness using a remote robot platform," presented at the IEEE International Conference on Systems, Man and Cybernetics, 2004.

[3] T. B. Sheridan, *Telerobotics, Automation and Human Supervisory Control*. Cambridge, MA: MIT Press, 1992.

[4] J. Brown, *et al.*, "Soldier Experiments and Assessments using SPEAR Speech Control System for UGVs," presented at the Axxociation for Unmanned Vehicle Systems International (AUVSI), 2010.

[5] T. Kollar, *et al.*, "Toward Understanding natural Language Directions," in *IEEE International Conference on Human-Robot Interaction*, Osaka, Japan, 2010.

[6] M. Skubic, *et al.*, "Using Spatial Language in a Human-Robot Dialog," in *IEEE International Conference on Robotics and Automation*, Washington, DC, 2002.

[7] O. Lemon, *et al.*, "Collaborative Activities and Multi-tasking in Dialogue Systems," presented at the 3rd SIGdial Workshop on Discourse and Dialogue, 2002.

[8] J. E. Laird, *et al.*, "Soar: An Architecture for General Intelligence," *Artificial Intelligence,* vol. 47, pp. 289-325, 1991.

[9] J. E. Laird, "Extending the Soar Cognitive Architecture," in *Artifical General Intelligence*, Memphis, TN, 2008.

[10] S. Oviatt, *et al.*, "Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions," *Human-Computer Interaction,* vol. 15, 2000.

[11] P. R. Cohen, *et al.*, "Quickset: Multimodal Interaction for Distributed Applications," presented at the 5th ACM International Conference on MultiMedia, 1997.

[12] M. Johnson, *et al.*, "Unification-based multimodal integration," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1998.

[13] Sanna A., *et al.*, "A Kinect-based natural interface for quadrotor control.," presented at the 4th International ICST Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN2011), Genoa, IT, 2011.

[14] S. Oviatt, *et al.*, "When Do We Interact Multimodally? Cognitive Load and MultiModal Communication Patterns," presented at the ICMI, State College, PA, 2004.

[15] M. Maybury, "Intelligent User Interfaces: An Introduction," presented at the 4th International Conference on Intelligent User Interfaces, 1999.

[16] J. Allen, *et al.*, "Toward Conversational Human-Computer Interaction," *AI Magazine,* vol. 22, 2001.

[17] O. Lemon, *et al.*, "A Multi-modal Dialogue System for Human-Robot Conversation," in *NAACL*, 2001.

[18] D. Traum, "Semantics and Pragmatics of Questions and Answers for Dialogue Agents," in *International Workshop on Computational Semantics*, 2003, pp. pp 380-394.

[19] R. Doorenbos, "Production Matching for Large Learning Systems," PhD, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1995.

[20] R. Dale and E. Reiter, *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press, 2000.

[21] E. Olson, *et al.*, "Progress towards multi-robot reconnaissance and the MAGIC 2010 competition. ," *Journal of Field Robotics,* (to appear).

[22] SoarTech. (2010). *SimJr*. Available: http://code.google.com/p/simjr/

[23] Parrot. (2011). *Parrot ARDrone*. Available: http://ardrone.parrot.com