Intelligent Pilot Intent Analysis System Using Artificial Intelligence Techniques

Richard Stottler[1] and Bonnie Schwartz[2]
Stottler Henke Associates, Inc., San Mateo, CA, 94404 and Air Force Research Lab/RBCC, Wright-Patterson AFB, OH, 45433-7542

Randy Jensen[3]
Stottler Henke Associates, Inc., San Mateo, CA, 94404

This paper presents the design, prototyping and testing results for a software system that automatically analyzes Air Traffic Control (ATC) communications that have been converted to speech along with location and velocity data of other aircraft from onboard sensors to determine pilot intent and predict their trajectories. The design includes novel knowledge representation techniques for this domain such as procedure graphs, dialog graphs, and grammar fragments. The design was prototyped and tested on actual ATC communications from San Jose International Airport (SJC). Results of this testing are also presented.

Nomenclature

ATC	=	Air Traffic Control
ATP	=	Aircraft Trajectory Prediction
BNs	=	Bayesian Networks
IPIAS	=	Intelligent Pilot Intent Analysis System
NAS	=	National Airspace System
SJC	=	San Jose International Airport
SR	=	Speech Recognition
TAO	=	Terminal Area of Operations
UAV	=	Unmanned Aerial Vehicle
U/DP	=	Utterance/Dialog Processing

I. Introduction

For Unmanned Aerial Vehicles (UAVs) to operate safely in the National Airspace System (NAS), they must be capable of operating with an equivalent level of safety to manned aircraft. One method to achieve this goal is to predict what other aircraft will do. No area is more crowded than the Terminal Area of Operations (TAO). The purpose of the Intelligent Pilot Intent Analysis System (IPIAS) is to provide an onboard capability that enables a UAV to predict trajectories of other aircraft in the TAO. This ability can only be realized with an automatic capability for recognizing the intent of the pilots of other aircraft. A significant factor in understanding intent is Air Traffic Control (ATC) communications to and from the other pilots, supplemented with sensors onboard the UAV.

There is currently a pressing desire in military and civilian communities for UAV airspace integration. This will require unmanned aircraft to coexist with manned aircraft at the same airfield, at the same time without sacrificing operational tempo or safety. Currently, unmanned operations are based separately and operational tempo is severely reduced during unmanned missions.

UAVs encounter unique problems in the terminal area. As they move away from being remotely piloted, technology development is necessary to safely and accurately navigate the terminal area. It is very difficult for a

[1] President, 951 Mariners Island Blvd., San Mateo, CA 94404, AIAA Member.
[2] USAF AFMC AFRL/RBCCX, AFRL/RBCC, 2130 8th Street, Bldg. 45 Rm. 283A, Wright-Patterson AFB, OH, 45433-7542, AIAA Contributor.
[3] Group Manager, 951 Mariners Island Blvd., San Mateo, CA 94404, AIAA Contributor.

UAV operator to maintain the same situational awareness and tempo of activities as an onboard pilot. In order for UAVs to truly integrate into the manned airspace, they must act autonomously and react like a manned aircraft.

There are many challenges to achieving a human-like reaction for a UAV in the terminal area. The Air Force Research Laboratory has identified several of these challenges, including understanding the intent of other aircraft. In the terminal area, under ATC control, human pilots keep a mental database of other aircraft locations and the commands they have received from ATC. This database along with basic knowledge of terminal area operating procedures enables the human pilot to accurately predict other pilots' intentions. For UAVs to integrate into terminal area operations with manned aircraft, the UAV must also have an intent analysis capability.

II. Description

A. System Overview

For ground operations, one of the most important (and easily generalizable to other airports) knowledge representation structures is the procedure graph. The procedure graph represents the expected dialogs and ground movements of the aircraft along with alternatives (e.g.,, additional, optional portions such as rejections of requests, or holds and releases). The dialogs in the procedure graph are themselves graphs of expected dialog events. For example, the taxi instructions in a departure procedure have four individual dialog events. The normal sequence includes a Taxi-Request event, Taxi-Approval (with route) event, and Taxi-Acknowledgement event but also includes an optional Taxi-Rejection event. By reasoning over the procedure and dialog graphs, IPIAS can determine the next expected events based on the last ones that occurred.

Associated with each type of dialog event are a set of fractional dialog grammars that each apply to just one fraction of the specific type of dialog event (though some are reused across different events because they appear in more than one type of event). This proves to be a very powerful technique to take advantage of the structure of important communication events while also ignoring portions of the same utterances that were unstructured and unimportant. For example, although a push request might contain a lot of extraneous information with various unimportant words inserted almost anywhere, all push requests include the word "Ground," an aircraft identifier (tail or flight number), and some form of the word "Push" in that order but not necessarily consecutively. Fractional grammars can, therefore, check utterances first for each of these three components and then for their relative order. The structure of these requests enabled the creation of a robust system that is tolerant to extraneous words and does not create false positives, regardless of the dozens of different dialog event types.

Figure 1 shows the high level architecture for the IPIAS. Speech Recognition (SR) and Sensor Processing, shown in gray, provide inputs to IPIAS but are being developed under other efforts. For each ATC utterance, the SR engine provides possible text parses and associated uncertainties. These are input to the Utterance/Dialog Processing (U/DP) module that uses procedure graphs and dialog graphs along with information specific to the airport (e.g., a local map and local vocabulary) to classify the utterance, extract relevant information (e.g., tail number and taxi route), and update the estimated state of the aircraft objects that it has created. The state includes current step of the procedure, location information derivable from ATC utterances, and the aircraft's planned route, if available. The aircraft objects and associated states are sent to the Aircraft Trajectory Prediction (ATP) module that correlates them with sensor reports from the Sensor Processing module and periodically updates its predictions on each aircraft's expected trajectories.

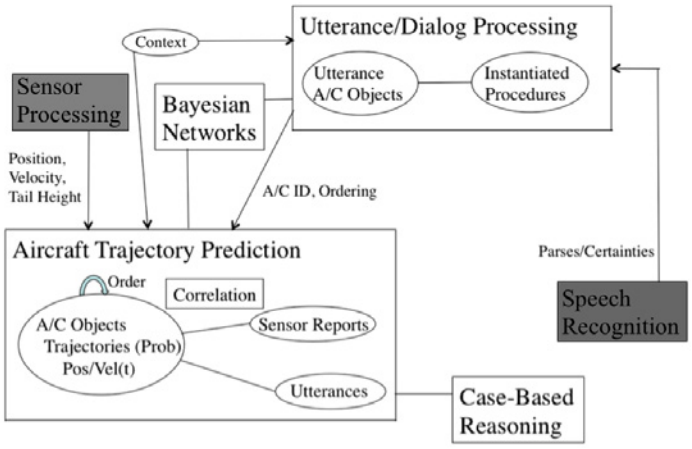

Figure 1. Intelligent Pilot Intent Analysis System Architecture.

B. Utterance Processing

Utterance processing makes use of the concept of grammar fragments. Not every word of every utterance has to be understood. There are often superfluous words in an utterance and sometimes unnecessary utterances or whole dialogs. Important dialog events need to be correctly classified and significant information must be extracted from them. Even in important dialog events in the terminal area, there is a fair amount of freedom in the structure and

content but evidence can be gathered enabling classification into a specific type of dialog event. This evidence is primarily in the form of executing dialog fragments, which are small regular expressions that capture phrases expected to be in the dialog event. For example, a typical push request starts with the word "Ground," followed by the aircraft's identifier (e.g., "Southwest 1062"), followed by its location (e.g., "at Gate 20"), followed by the request (e.g., "requesting push").

When determining if a specific text string should be classified as a specific dialog event, the grammar fragments are applied to the string. Associated with each dialog event is the set of allowed ordering of the phrases; these orderings are checked to see if they hold for the specific utterance being analyzed. (Note that sometimes it is acceptable for one phrase to be embedded within another). Based on which phrases were recognized (and with what probability), and their adherence to the allowed set of orderings, the probability that this utterance should be classified as a specific event is calculated, first in isolation, and later in the context of previous and subsequent dialog events for the same aircraft identifier.

1. Real-Time Utterance Processing

The U/DP module keeps a global and an aircraft-specific context for each channel. These include the last few utterances spoken; when and by whom they were spoken; whether they were intelligible; and the possible parses and interpretations (dialog events), if they were intelligible. When a new utterance is processed by the SR module and the resulting parses and certainties are sent to the U/DP module, it first checks the global context to see whether the utterance is likely part of a specific, ongoing dialog. This check include the time between the last utterance on the same channel and the new utterance; whether the last utterance's dialog event was a last event in its dialog graph, and the aircraft identifier(s) extracted from the new and last utterance. If one of the possible parses of the new utterance is potentially part of a dialog or it includes a previously encountered aircraft identifier, two steps are taken. First, the probabilities that the previous parses with the same aircraft identifier are the correct parse are increased, especially in the case of the last utterance being part of the same dialog as the new utterance. Second, the procedure graph and dialog graph instances for the specific aircraft are accessed.

For aircraft previously referenced, the U/DP module accesses the graph instances to determine the next dialog event. For example, if the last dialog event was a Push Request event and that event was very recent, then the next dialog event should either be a Push Approval event (with high probability) or a Push Rejection event (with much lower probability). However, if the last dialog event was a Push Acknowledgement and enough time has passed that the aircraft could be done pushing back, then the next expected event is the first Taxi Dialog event.

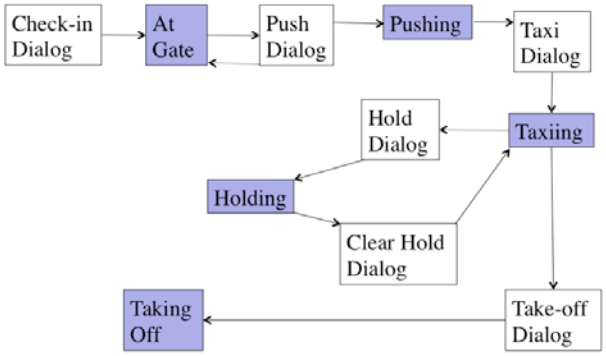

Figure 2. Departure Procedure Graph.

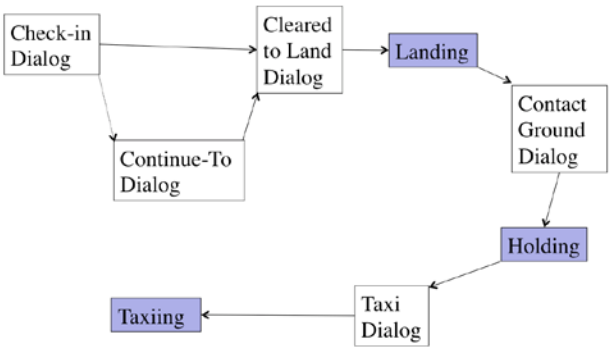

Figure 3. Arrival Procedure Graph.

The U/DP module may receive multiple text strings (with associated probabilities) for each utterance from the SR module. For each test string whose probability is above the threshold, given the next expected dialog event, the set of grammar fragments associated with that event type are evaluated and a Bayesian calculation is used to determine the probability that that dialog event is the correct one. If the probability is below the threshold, the grammar fragment sets for successive dialog events are evaluated under the assumption that one or more dialog events have been missed. The probability of one dropped dialog event (especially given a low probability for the expected next event) is not very low. However, the probability decreases exponentially for the assumption of an increasing number of dropped events in a row unless the context information indicates that IPIAS's UAV has been out of radio communication. The probability also depends on the time since the last recognized event for this aircraft. For example, it is more likely that IPIAS has missed several events if enough time has passed for those

events to have reasonably occurred. Conversely, if little time has passed since the last recognized event, it is very unlikely that many events have occurred.

When the next event is recognized as expected, a Bayesian calculation increases the probability that the aircraft identifier is correct (as previously described) and that the aircraft is at the indicated step, thus increasing the probability of future predicted events from this aircraft's procedure and dialog graph instances. By the time an aircraft has progressed several dialog steps through the procedure there will be a lot of evidence and therefore both a high probability and a resistance to small amounts of contrary evidence for both the specific aircraft identifier and for the current step in the procedure. When an aircraft identifier from a correct parse has not been referenced before, a new utterance aircraft object is created for it and the appropriate (departure or arrival) procedure graph and dialog graphs are instantiated for it and initialized to the correct procedure and dialog step as evidenced by the current utterance's parse.

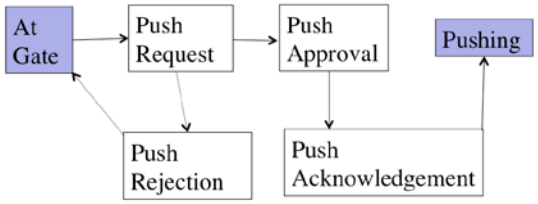

Figure 4. Push Dialog Graph.

The U/DP module must also handle abbreviations, route extraction, and ordering. Abbreviations only apply to tail numbers because flight numbers are never abbreviated. Abbreviated tail numbers are straightforward because they are not parsed any differently than nonabbreviated tail numbers. They consist of an optional model name followed by a sequence of numbers and letters. Additionally, ATC will not use abbreviations that are ambiguous, i.e., if two tail numbers are similar, they will always use the entire tail number and will even announce that there are similar tail numbers present and that abbreviations for the affected tail numbers should not be used. Handling abbreviations becomes a matching problem with straightforward rules. If the aircraft model is spoken, it has to match for the identifiers to match; if no model is present in either identifier being matched, the model should be ignored for matching purposes. Separate from the model, it is acceptable to leave off the beginning, end, or both of the sequence of numbers and letters so that to match, one sequence must be a substring of the full tail number.

Given the use of grammar fragments, route extraction is a straightforward process. Once the dialog event type has been determined, if it is one that may have a route included with it (Push-Approval, Push-Acknowledge, Departure Taxi-Approval, Departure Taxi-Acknowledge, Arrival Taxi-Approval, and Arrival Taxi-Acknowledge), the grammar fragment for the route is applied. In the case of the push dialog events, this will be the phrase "PUSH <to the> direction," where direction will be different for different airports. For taxi dialog events, the expected phrase is "location <taxi> via route" or "route <to> location," where location is a runway, specified by a number and name; a gate, specified by a door or gate number; or a named area of the airport, specified by name only.

A route is a series of letters that correspond to straight taxiway segments that cross and lead from the aircraft's current position to the destination location. The U/DP module contains a graph of these named segments that describes which segments cross other segments, the precise latitude/longitude of those intersections, and the precise end point locations of the segments. The segments are typically roughly parallel or orthogonal to the runways. To generate the predicted route from the uttered route, the aircraft's current position (segment(s)) must first be determined. Successive segments are usually orthogonal to each other. The first named route segment will be the segment that the aircraft is currently on so that the first predicted route segment will be from the aircraft's current position along the first segment. The second named route segment will intersect the first one, and therefore the second predicted route segment joins at that intersection with the end of the first predicted route segment. This process repeats until the last segment. The process is complicated by the fact that ATC may omit crossing segments, in which case the successive named route segments will be parallel instead of crossing. When this occurs a number of different predicted routes have to be created, corresponding to taking any possible route segment that connects the two named parallel segments. If the aircraft is pointed in the approximate direction of its destination, segments behind it and that overshoot the destination can be eliminated, or have their probability greatly decreased.

Ordering information is indicated in the dialog by the keywords "BEHIND" or "NUMBER 2." Extracting ordering information is more complicated because usually the aircraft that is to be followed is not specified by identifier but by type or occasionally by airline. "NUMBER 2" always refers to taking off or landing, so it is straightforward to determine the aircraft that is currently cleared to take-off or land on the same runway. During ground taxi, ordering information typically references an aircraft in close proximity. It may not always be possible to immediately determine which aircraft is being referred, but it should be resolvable once the aircraft has started to move.

C. Aircraft Trajectory Prediction (ATP)

Predicted aircraft trajectories are represented as a series of segments where each segment represents the expected movement of the center of the aircraft. Each endpoint has an error in its width and expected time that the aircraft will be at that point. All of the forward projection, trajectory generation, and simulation are done in the ATP module. The ATP module includes an internal simulation of the aircraft, trucks, airport/taxiways, and queuing (including expected gap distances between planes in a line). It simulates all trajectories forward in time for the next X minutes or to the end of the current path and noting expected centerline location with error widths and time errors at the ends of each segment. In particular, the simulation is able to simulate the behavior of planes waiting in the queue to take off, accelerating (decelerating) down the runway during take-off (landing), and taxiing to parking areas. The simulation will rerun immediately after new data is received. Specifically, it uses the most up-to-date sensor information and correlations.

The ATP module receives inputs from the U/DP module (Utterance Aircraft Objects) and the Sensor Processing (SP) module (position/velocity/tail height), correlates them, creates and updates Aircraft Objects with associated trajectories, and outputs these as the primary IPIAS system output. After Aircraft Objects are created (usually as a result of an initial Utterance Aircraft Objects but possibly from sensor reports), they will be updated both from correlated sensor data and the updating of the associated Utterance Aircraft Object. As additional sensor data comes in, it will be correlated with predictions from the Aircraft Objects in the ATP module. The Aircraft Objects include the (possibly multiple hypothesized) predicted trajectory for each aircraft along with estimates of location and velocity that come from utterance processing and sensed position and velocity. If sensor data is unavailable between utterances, estimates of time delays between procedure steps and speed of taxi will be made. These estimates are based on general heuristics or similar past events. The Aircraft Objects use the trajectory, estimated past location and current speed to estimate where the aircraft is along the trajectory and its velocity vector at any given time. These estimates become inputs to the sensor correlation algorithm, along with the sensor data.

1. Sensor correlation

The sensor correlation algorithm processes Utterance Objects by decreasing certainty of location. Specifically, the following locations are in order of decreasing certainty: at the gate, cleared for takeoff, landing, cleared to cross a runway, on parking spot, in takeoff queue order, all other locations. For each Utterance Aircraft Object, the sensor correlation algorithm finds the most likely sensor report (the closest that shares most attributes) and marks both as correlated if not already correlated. If a sensor report was previously correlated, the correlation with the highest probability is applied. If there are other probable correlations, the associated sensor reports are recorded.

Aircraft Utterance Objects cleared to land or take off, are correlated with anything on the runway heading in the correct direction and within a reasonable range of the expected velocity. The expected velocity is calculated as a function of position on the runway. The sensor correlation assumes that aircraft very rarely move backward after pushback. Any specific aircraft object may have multiple predicted trajectories and therefore multiple predicted positions. Once a sensor report correlates with one trajectory, any other possibilities can be eliminated. In some cases, when the trajectories have not yet diverged significantly, one sensor report may correlate with multiple predicted positions so multiple hypothesized trajectories will be maintained until the trajectories diverge. When one aircraft's multiple predicted positions correlate with multiple different sensor reports, the possible associations have to be maintained until an attempt has been made to correlate all sensor reports. Since the sensor correlation algorithm processes in decreasing order of position certainty, this problem mostly takes care of itself.

2. Bayesian Networks (BNs)

Bayesian Networks (BNs) will be used extensively to calculate probabilities and, in most cases, the required BN will be dynamically copied from a master BN assembled during initialization when a new aircraft is discovered. The procedure graphs can be combined with the dialog graphs to create one large graph for each of the two procedures (arrival and departure) that define the allowed types of dialog events, their relative relationships, and various aircraft states. For example, in the requesting taxi state while stopped at the final pushback location or in the pushing back (moving) state. For each unique state, a BN exists that shows the probability that observable evidence will be received. The evidence can consist of both dialog evidence and sensed evidence. In the case of dialog events, the expected evidence includes dialog fragments, their relative order, and the time since the last dialog event for this aircraft, as evidenced by the aircraft identifier or specific location. Dialog fragments also include extractable parameters like tail number and taxi route. There are two BN nodes for each of these parameters. One BN represents evidence from parsing the utterance at the bottom of the BN. The other BN represents the actual parameter of the aircraft (e.g., the actual tail number) at the top of the BN. For a single aircraft, the node representing the aircraft's identifier and other parameters will typically include more than one value.

There are connections between the different dialog event BNs, specifically between the parameter nodes at the top. There are also logical connections where values for some of the top nodes are set based on values from other top nodes from other BNs. For example, the expected current state of the aircraft, which is based on the previous time of the last sensed state, may be set by an actual BN connection or by software code. One (easily altered) design decision is which dialog event BNs to connect together directly versus which ones to connect logically (i.e., assigning a value through software or receiving it from another BN). The advantage to providing a direct connection between the BNs is that evidence flows back and forth naturally. The disadvantage is that larger BNs may have computation time issues. An advantage to making the connection logically is that it can involve additional logic and calculations. Current efforts include experimenting with different BN structures.

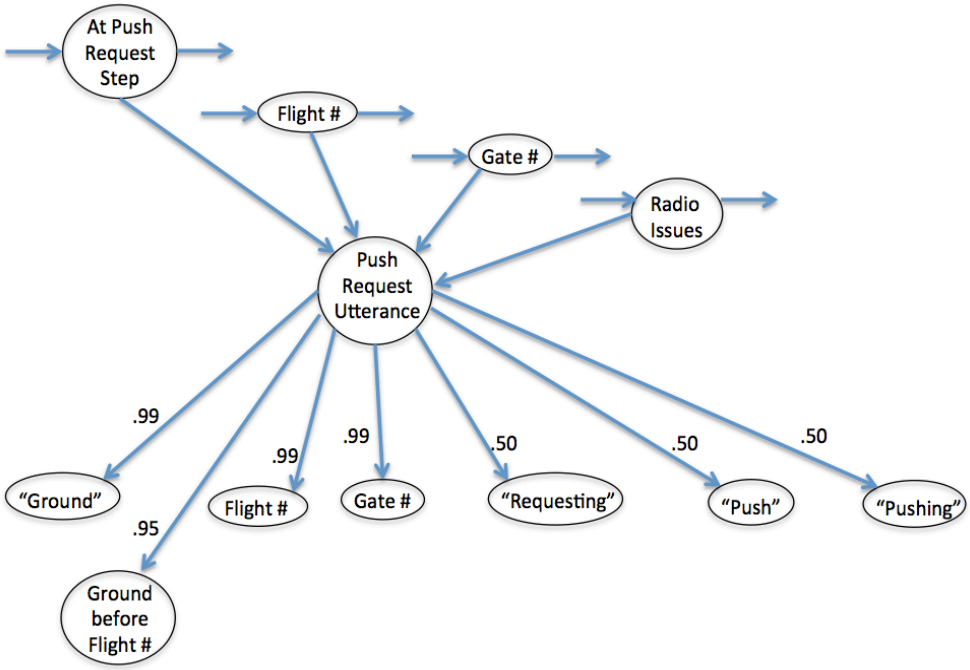

Figure 5. Push Request BN.

An example of a BN for a dialog event, Push Request, is given in Fig. 5 above. In this case, the center node represents the event of a Push Request being broadcast to ATC and received at the UAV. It depends upon the aircraft being at this step in the procedure, the aircraft's flight number, gate, and whether all radios are currently working, configured properly and away from dead zones. The evidence for this utterance will include whether the dialog fragments found specific words ("Ground," "Gate," "Requesting," "Push," and "Pushing"), found specific parameters using the correct syntax (flight number and gate number), and proper orderings of dialog fragments. Based on evidence input at the bottom, the parameters and probabilities can be calculated at the top. The top nodes may also include either input from earlier event BNs (e.g., At Push Request Step or Flight #) or be transmitted to top nodes of other, future event BNs.

III. Results

A. Prototype Implementation of the Design

This design was initially prototyped and applied to a simple airport simulation based on the San Jose International Airport (SJC). The prototype processed dialog relating to commercial airline and general aviation departures and arrivals at SJC; predicted the current position of the aircraft based on the dialog; correlated that to simulated sensor data; and predicted its future trajectory (ground route). The ground truth simulation of SJC ground traffic, including pushbacks, taxiing, take-offs and landings, was used to drive simulated sensors that provided estimates of the position, velocity, and tail height, each with randomized error. The prototype displayed the aircrafts' sensed locations with red circles, their ground truth locations with icons (white airplanes on black squares), trajectory predictions (red lines and arcs) and predicted locations and headings (green triangles).

The utterances came from a time-stamped text file extracted from actual SJC ATC transcripts that were correct except that utterances and specific words in utterances might have been dropped or unintelligible. The simple simulation feeds out each utterance at the proper simulation time. The simulation and IPIAS combined prototypes ran on a basic desktop processor easily in real-time. The ground truth simulation read the scripted movement of aircraft from a scenario file and updated their position and velocity every second. It sent this information to the sensor model, which added random error to produce a sensed position, speed, heading, and tail height every three seconds. This sensed data was sent to the Correlation module, which also received predicted positions and velocities from the ATP module. The Correlation module correlated the sensed data to the best matching predicted data. Based on this matching, updated position, velocity, and tail height were sent back to the ATP module, which updated its predictions based on these sensed quantities. Tail height is a special case where the ATP model initially has no information until a sensor data match was made based entirely on position and velocity. At the time of the first match, the sensed tail height was assigned as the predicted tail height. Thereafter, this tail height could be used to resolve correlation ambiguities.

Finally, the ground truth data was displayed with aircraft icons (see Fig. 6). Each aircraft icon indicates the ground truth position and approximate heading, shown by the direction of the airplane icon, as well as the aircraft identifier, overlaid in red text in Fig. 6. (The white arrow points to an example.) The position sensor data is displayed as red circles, indicated by the yellow arrow. The predicted positions and headings are indicated by the green triangles, an example of which is pointed to by the green arrow. This example also shows two predicted trajectories in red, a red arc for a push back and red line down the runway for a landing.

Figure 6. Prototype Data Display.

The prototype display also gives a glimpse into the prototype's reasoning, specifically its use of the procedure and dialog graphs. Figure 7 shows an example. The collapsible tree display starts empty but adds aircraft as each is mentioned during ATC communications. Underneath each aircraft as it steps through the procedure and dialog graphs, it keeps track of the events that have passed, the current state, and the next expected event. In Fig. 7, the Cessna has (reading upward through past events since the most recent is at the top) received landing approval and has acknowledged that approval and therefore its current state is Landing-In-Progress and its next expected dialog event is Taxi-to Gate-Request since after it finishes landing, rolling out, and pulling off the active runway.

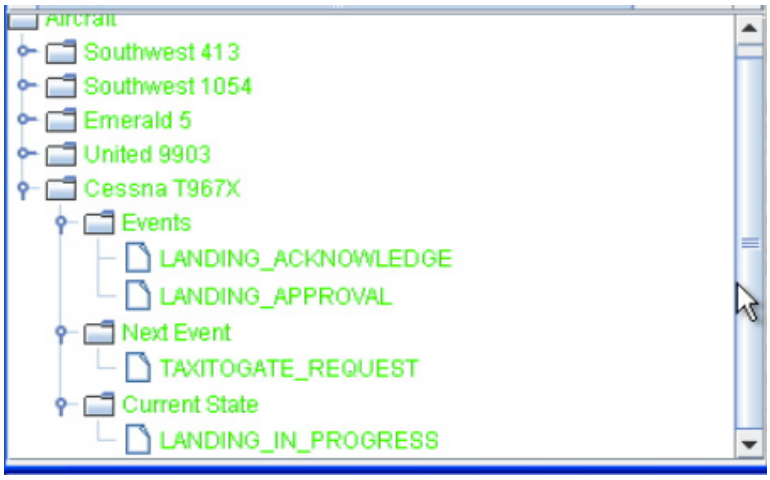

Figure 7. Past and Predicted Events.

B. Prototype Results

The prototype worked well on the complex scenarios and actual SJC transcripts. Below is a segment from the ground channel transcript of the demonstration scenario with bolding added related to the portions of the transcript that are discussed herein. There are several interesting aspects to point out that were all correctly handled by the prototype. Note the extraneous conversation at 15:29:12 and 15:29:15, which the prototype correctly ignores. Similarly note the "good afternoon" at 15:30:25 was also ignored. The pilot-spoken lines at 15:28:15 ("requesting push"), 15:30:05 ("pushing"), and 15:30:25 ("push with Foxtrot") all use different ways to request a push. Note the two different route specifications, all from the gate area to the preparation for takeoff area. The first, at 15:28:57, includes instructions about both cross segments, "Runway 30R taxi via Golf Yankee Alpha," whereas the second, at 15:30:37, omits both crossing segments giving the pilot the freedom to choose and requiring IPIAS to maintain several competing hypotheses, "Runway Three-Zero-Left, taxi via Yankee." Similarly, an unusual push approval at 15:30:30: "San Jose Ground push approved." This is one of the few without a direction and it requires IPIAS to maintain competing pushback trajectories, one to the North and one to the South, until the true one can later be resolved with sensor data. Lines 15:29:55, 15:29:58, and 15:30:02 include three different ways to reference the same aircraft, "Cessna Tango 967 X-Ray," "Cessna 67," and "Cessna 67 X-Ray." Finally, at 15:30:45 by referring to Runway 12 Left, the script demonstrates how the prototype is able to adapt to changing runway directions (taking off south instead of north).

Pilot 02/28/2011 15:28:15 Ground, Southwest 1054, Gate 24 with Foxtrot, **requesting push**
ATC 02/28/2011 15:28:17 Southwest 1054, push your tail to the south
Pilot 02/28/2011 15:28:20 Push tail to the south, Southwest 1054
Pilot 02/28/2011 15:28:45 San Jose Ground, Southwest 1054, taxiing Foxtrot
ATC 02/28/2011 15:28:57 Southwest 1054, **Runway 30R taxi via Golf Yankee Alpha**
Pilot 02/28/2011 15:29:05 Golf Yankee Alpha 30R Southwest 1054
ATC 02/28/2011 15:29:12 **Nice job out there, sir. Thanks for the help**
Pilot 02/28/2011 15:29:15 **No, thank you**
...
Pilot 02/28/2011 15:29:55 San Jose Ground **Cessna Tango 967 X-Ray**, we're clear at 30 Left Juliet, we are going to Atlantic
ATC 02/28/2011 15:29:58 **Cessna 67** Ground taxi to Atlantic via Juliet Victor
Pilot 02/28/2011 15:30:02 Juliet Victor to Atlantic, **Cessna 67 X-Ray**
Pilot 02/28/2011 15:30:05 Ground, Southwest 581, Gate 21, **pushing**
...
Pilot 02/28/2011 15:30:25 Ground, **good afternoon** Southwest 2989 Gate 20 **push with Foxtrot**
ATC 02/28/2011 15:30:30 Southwest 2989 San Jose Ground **push approved**
Pilot 02/28/2011 15:30:35 Ground, Southwest 581, ready to taxi from 21 with Foxtrot
ATC 02/28/2011 15:30:37 Southwest 581 **Runway Three-Zero-Left, taxi via Yankee**
Pilot 02/28/2011 15:30:40 Yankee 30 left, Southwest 581
ATC 02/28/2011 15:30:45 American 504, **Runway 12 Left** taxi via Kilo Yankee November
Pilot 02/28/2011 15:30:48 Kilo Yankee November 12 Left American 504

Figure 8. Scenario Transcript from SJC ATC-Pilot Communications.

Figure 9. Prototype Display Screen Capture.

In Fig. 9, the green text in the upper left, "Southwest 2989 San Jose Ground push approved", has just been processed. As described above, this has led to two different push back trajectories; one is shown in black and indicated with a yellow arrow and one is shown in red and happens to intersect with a taxi trajectory. Ten seconds later this ambiguity was resolved when sensor data reveal that the aircraft is pushing back to the North. The long straight red line corresponds to the predicted trajectory for flight 9903, which was previously cleared to take off (fourth text line from the bottom in the figure). Other predicted trajectories that are shown were extracted from the dialog. They include the lower-leftmost red 3-segment path for Cessna T967X from the runway to the "Atlantic" area, the red 3-straight-segment path on the upper right for an aircraft that has almost completed its ATC-specified route from gate area to takeoff area, and a curved pushback path for flight 504, which has almost finished its pushback maneuver. At a later time, when Southwest 581 is given a vague taxi path by ATC, 6 different cross over points (6 different trajectories) are initially shown by the prototype. Since the aircraft ends up taking the last one, the final trajectory is not resolved until close to the end. However, each individual trajectory is eliminated one by one as the aircraft passes the crossover points corresponding with each different trajectory without taking them.

The demonstration scenario had 17 aircraft and 11 takeoffs and landings in a 12 minute period. It included aircraft that went through the entire procedure during the scenario and aircraft that were at various stages in their procedures when the scenario started. The dialog was taken directly from SJC ATC-pilot transcripts and all of it was handled correctly. The simple sensor data correlation algorithms worked well with miscorrelations being rare and automatically corrected within two additional sensor sweeps.

The sensor model was designed to model sensor processing that would not be too challenging to achieve. Attributes were a 3 second sensor processing time (for all aircraft in view), approximately +/- 1% error in reported position, +/- 5 degrees of error in reporting heading, and +/- 10% error in reported speed and tail height. Because current vision processing techniques exceed these values, IPIAS is expected to perform well when integrated with realistic sensors.

IV. Future Work

The next step is to develop the full-scale IPIAS to support autonomous UAV ground operations and integrate it to the degree possible with an appropriate UAV system. The full-scale IPIAS will analyze text recognized with a speech recognition engine from actual airport ATC-Pilot communications and use simulated sensor data to predict routes on the ground for aircraft on the ground, taking off, and landing. IPIAS is intended to eventually be an integral part of an operational ground-autonomous UAV.

The goals of the current research are to develop IPIAS to automatically provide aircraft ground trajectories; integrate it into the larger architecture and the appropriate simulation and testing environment; show near-perfect performance within reasonable computational bounds on realistic scenarios with real ATC communications; and

demonstrate significant benefit for the autonomous UAV ground operations. In support of these goals, pilot intent recognition cognitive processes are being elicited from experienced, commercial, small aircraft pilot to create full-scale procedure and dialog graphs.

V. Conclusion

The prototyping effort provided very good support for full-scale implementation. The most important aspect was the development and design of the techniques and knowledge representation schemes for ATC-Pilot dialog processing and analysis and the prediction of aircraft trajectories to be used in the ultimate system and implemented in the prototype. For ground operations, one of the most important (and easily generalizable to other airports) knowledge representation structures was the procedure graph. The procedure graph represents the expected dialogs and ground movements of the aircraft along with alternatives. The dialogs in the procedure graph are themselves graphs of expected utterance events. By reasoning over the procedure and dialog graphs, IPIAS can determine the next expected events based on the last ones that occurred.

Associated with each type of dialog event are a set of fractional dialog grammars that each apply to one fraction of the specific type of dialog event (though some are re-used across different events because they appear in more than one type of event). This proved to be a very powerful technique in the prototype to take advantage of the structure of important communication events while also ignoring portions of the same utterances that were unstructured and unimportant. For example, although a Push-Request might contain a lot of extraneous information with various unimportant words inserted almost anywhere, all the Push-Requests include the word “Ground,” an aircraft identifier (tail or flight number), and some form of the word “Push” in that order but not necessarily consecutively. By having separate fractional grammars check utterances first for each of these three components, then for their relative order, a system that was robust, tolerant to extraneous words, but did not create false positives was created in the context of a limited prototype.

The prototype was successfully demonstrated with a complex scenario including actual ATC-Pilot dialog from San Jose Airport (SJC). The prototype also included a simple simulation, sensor models, and sensor data correlation capability. The prototype worked well on the complex scenarios and actual SJC transcripts. The demonstration scenario had 17 aircraft and 11 takeoffs and landings in a 12 minute period. It included both aircraft that went through the entire procedure during the scenario and aircraft that were at various stages in their procedures when the scenario started. The dialog was taken directly from actual SJC ATC-pilot transcripts and all of it was handled correctly. The relatively simple sensor data correlation algorithms worked well with miscorrelations being rare and automatically corrected within two additional sensor sweeps.