1992 Rochester Forth Conference
June 17-21, 1992, University of Rochester, Rochester, NY

The Sensor Driven Airborne Replanner: Artificial Intelligence for Unmanned Air Vehicles

Richard C. Wagner
Aerospace Engineer
Advanced Processors Branch, Code 5051
Naval Air Warfare Center, Aircraft Division
Warminster, PA 18974-5000

Abstract

The Sensor Driven Airborne Replanner is a Forth computer hardware and software implementation providing autonomous control of Unmanned Air Vehicles (UAV's). The SDAR system employs artificial intelligence techniques to impress the interests and intentions of a human mission planner onto the real-time behavior of a fully autonomous surveillance UAV. Based on CMForth and the Harris RTX2000 processor, the system executes an AI expert system, image processing, database management, aircraft sensor data processing, aircraft control, and camera gimbal control/stabilization in real-time. The product of a three-man, three-year effort, SDAR has been successfully flying in a high-fidelity environment simulation (which was also written in Forth) for over a year. Flight testing is scheduled for the summer of 1992.

Introduction

Unmanned Air Vehicles (UAV's) are unmanned aircraft used by the armed forces to perform missions considered too dangerous or too expensive to justify the use of a manned aircraft. UAV's have been active in the military service for many years, successfully performing a variety of missions from attack (cruise missiles) to surveillance, and also as airborne gunnery/missile targets. Most of these aircraft are of the fixed-wing variety and vary in wingspan (relative size) from about five feet, the size of a typical radio-controlled model, to over one hundred feet. There are also a few rotary-wing (helicopter) UAV's, one of which may be entering service with the military in the near future.

The method used to control UAV's in flight varies. Cruise missiles and most other offensive UAV's either fly a pre-programmed path autonomously from their point of launch to their target, or use sensors to home-in on the target. Offensive missions are relatively simple, since the location of the target is known before launch. The offensive UAV must simply fly from point to point along a simple, predetermined path. The surveillance mission, on the other hand, is fairly complex. The external situation is typically unknown and is, in fact, being analyzed through the UAV. As information about the external environment is made available by the aircraft, the mission must continually change in order to optimize the assimilation of data. Thus, all surveillance UAV's are currently controlled from the ground. This provides the UAV operator with complete control of the aircraft and permits unconstrained mission modification.

The Problem

There are a number of problems brought about by the ground-based control of surveillance UAV's. The current state of the art in surveillance UAV control requires maintaining a continuous command/control uplink to the aircraft. This uplink provides commands to the autopilot for conventional control of the vehicle. The ground-based pilot issues commands through the uplink to control the flightpath (direction, altitude, airspeed) of the UAV.

The uplink also carries commands to the camera and its gimbal. These commands allow the operator to point the camera where needed, and to zoom in and out from a target of interest. The UAV, in turn, operates a continuous data downlink. Since the operator is on the ground, and since the aircraft will typically be operating beyond line-of-sight, data regarding the UAV's position, attitude, altitude, and airspeed must all be transmitted down to the operator for display. The camera signal is also carried on the downlink so that real-time imagery can be analyzed at the ground control station.

These up- and downlinks make the UAV/ground-station system extremely vulnerable. In a hostile situation, the links act as beacons announcing the presence of both the UAV and the ground station. The links also act as localizers for any number of homing weapons that could be brought to bear against them.

The objective of the SDAR project is to provide autonomous control of a Naval UAV, eliminating the control and data links without limiting the flexibility of the surveillance mission or the quality of the data returned to the operator.

The SDAR Solution

To provide useful, autonomous control of the UAV, a hardware/software architecture was developed which would allow a human mission planner to impress his intentions and interests onto the real-time behavior of the aircraft system. The architecture provides:

- A waypoint database allowing the mission planner to pre-program a provisional search path.
- A ship interest matrix allowing the mission planner to specify his interest in a given ship based on a set of UAV-collected ship parameters. (For example, the mission planner can specify that he is greatly interested in ships longer than 700 feet. He could also specify, for example, that he has no interest at all in ships shorter than 200 feet.)
- A transmitter interest matrix allowing the mission planner to specify his interest in certain radio transmitters.
- An AI expert system to analyze the current situation based on data from the UAV's sensors and alter the behavior of the aircraft and its systems to best adapt to and capitalize on the situation. The expert system also directs the main mission sensor (in this case a camera) to optimize the quality of the data retained or sent to the operator for his analysis.

The SDAR architecture allows the human mission planner to impress his interests and intentions onto the real-time behavior and decisions of the UAV. Before launching the UAV, the mission planner begins by pre-programming a provisional, or preferred, search path that the UAV will follow. The search path is defined by an ordered list of "waypoints" in space. He then enters his ship-interests into a ship interest matrix, based on observable ship parameters and radio transmitter types. The UAV is then launched, carrying a camera and a communications frequency Electronic Support Measures (ESM) receiver. (The receiver reports the bearing to any transmission, plus the type of transmitter that sent the transmission.) Following the provisional search path, SDAR flies the UAV into the area of interest.

As the UAV flies along the search path, SDAR searches visually for ships by scanning the on-board camera. It also listens for ship radio transmissions using the ESM receiver. If SDAR sees a ship, it quickly inserts the ship's length, speed, course and position into its ship database and compares the pertinent ship data to the mission planner's entries in the ship interest matrix. The ship's data entries will be updated automatically, so that its position is always known, even when the system is not looking at it. Should the ship be determined to be of interest to the mission planner, SDAR then flies the UAV covertly to the beam of the ship using an aft approach (see Fig. 1). Once abeam the ship, SDAR either records or transmits a number of

frames of image data from the camera along with the pertinent ship data. This is the SDAR payoff: autonomous location of a potentially interesting target, with low-probability-of-intercept transmission of data back to the operator for his analysis. With each frame of data, the interest value of the ship is decremented, causing the ship to become uninteresting after a proportionate number of photos have been taken. At this point, SDAR leaves the ship and returns to searching along the predefined search path. SDAR continues to update the ship's data entries for some time after the encounter. Should it be encountered again, the ship's actual location will likely correlate with its updated position in the database and SDAR will know that it has already prosecuted the ship. Since the ship is now uninteresting, SDAR will ignore it, or avoid it if it is in the way.

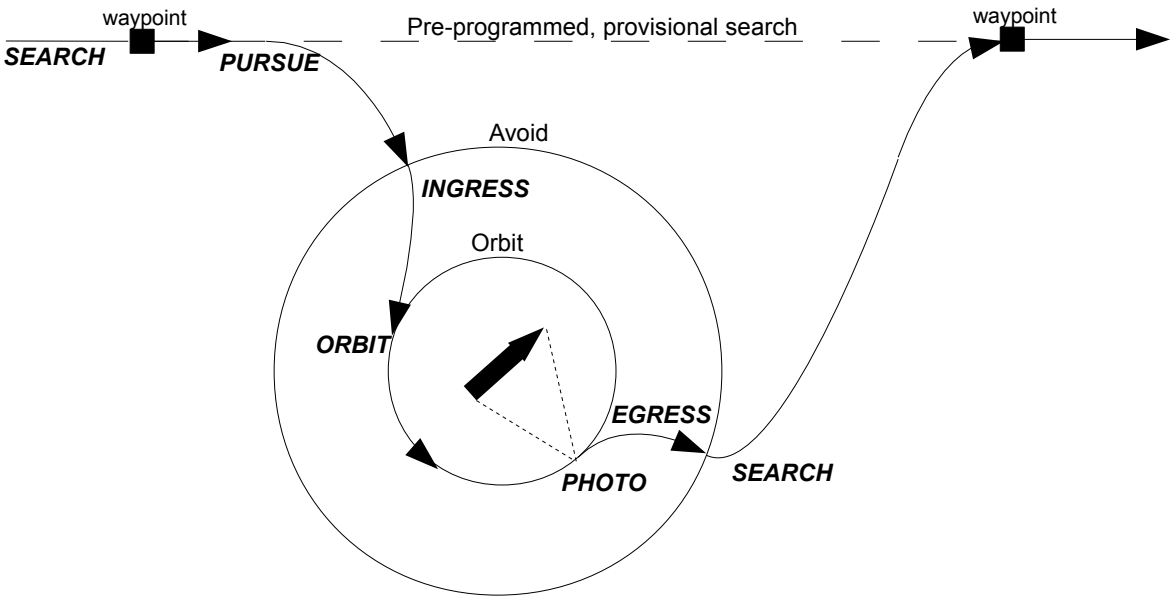

Figure 1: SDAR behaviors (in caps) during a typical encounter

If the system intercepts a radio transmission with the ESM receiver, the transmitter type and the bearing to the transmitter are entered into SDAR's transmitter database. SDAR then compares the transmitter type to the mission planner's entries in the transmitter interest matrix. If the transmitter is interesting, SDAR executes a "VERTICAL-PAN" behavior by turning the camera to the bearing of the transmitter, and then panning the camera vertically to the horizon. This is called "sensor cross-cueing" by the military, where one sensor is used to direct another to a subject of interest. If, during the vertical pan maneuver, the ship is seen in the camera's field of view, SDAR then prosecutes the ship in its normal fashion. If the ship isn't seen, SDAR reverts back to its previous behavior after the end of the vertical pan maneuver. If a transmitter is found to be uninteresting, SDAR simply ignores it.

SDAR Software

The SDAR system is written using CMForth, a Forth model developed by Chuck Moore to run on the Novix NC4000 Forth processor.

The heart of the SDAR system is the inference engine. The SDAR inference engine is forward-chaining and is unique in that it allows parameter passing on the stack between the

antecedent and consequent phrases of a rule. Although this sounds rather trivial, it does allow for a very logical flow of data down a rule. Running on the Harris RTX2000 processor, the inference engine is capable of 20,000 simple inferences per second.

The SDAR rulebase is also somewhat unique. The rules are all written in very high-level code. SDAR is a state machine -- there are a number of states into which the system can be transitioned. Each state embodies a unique aircraft system behavior. Writing state transition rules in high level code makes them extremely readable. This simplifies the development of state diagrams and promotes the construction of a stable rulebase that doesn't thrash between states.

There are two sets of words written to support the rulebase. The first set, "conditional attachments", are words that are executed in the antecedent phrase of the rule. These words typically execute sensor data processing algorithms whose details need not be seen. Thus, a level of abstraction is introduced which makes a rule more readable. An example of a conditional attachment is the word which determines whether or not there are any ships in the field of view of the camera, ?SHIPS-IN-FOV. The second set of words, "procedural attachments", are executed in the consequent phrase of a rule. These words execute routines that either control the physical hardware of the aircraft, or act upon the internal configuration of the SDAR system itself. An example of a procedural attachment is the word which configures the aircraft for searching, SEARCH-PLATFORM. SDAR's current mission requires about 70 rules to transition the system's behavior through 13 different states.

CMForth is a very small, clean Forth environment. As such, it is easy to understand and very, very fast. However, it does lack some of the amenities to which many Forth programmers using large, PC-based systems are accustomed. To support the development of SDAR, a very complete on-line development environment was written. The environment includes a nearly complete Forth-83 overlay which loads on top of CMForth, and a well rounded set of debugging aids: SEE, a decompiler featuring source code viewing and indenture; VIEW; TRAP; and STEP. The communication routines used to talk to and load the SDAR single board computers are written in polyForth and, with the on-line development environment, provide a nearly transparent interface between the host PC and the target machine. The entire SDAR system, including the on-line development environment, currently occupies about 34K of RAM on the target machines.

SDAR Hardware

The SDAR system runs on Silicon Composers' RTX2000 serial and parallel single board computers. The parallel boards mount directly into the backplane of the host PC, mapping into the PC's memory. This permits very fast memory-to-memory transfer of source code from the PC's hard disk to the target machine's block buffer, where it is compiled under CMForth. The serial boards are used for the actual airborne systems and because of the greatly reduced data transfer rate, require much longer to load from source code. However, code development is always complete before the serial boards are used, so the source code need only be compiled once. After that, the dictionary can be copied out and reloaded straight into RAM later on. Thus, only about 34K bytes of data need to be transferred over the serial line, as opposed to about 600K bytes of raw source code. Both boards operate at 10 MHz, and running optimized CMForth produce an average throughput of about 15 MIPS.

Laboratory Development Environment

Development of such a complex system on a real aircraft would obviously be very expensive. During the testing of new code, there would always be a risk of losing the aircraft, and the separation of the developers from the real hardware of the system during flight would limit their ability to assimilate information. This partially opens the tight development/testing

loop that Forth programmers enjoy so much. One other drawback of using a real aircraft for development is that the real world only works in real time. Consequently, a very complete, high-fidelity environment simulation was written in polyForth to support the development of SDAR in the lab. The simulation was designed to completely emulate the environment that the SDAR computer would experience should it be installed in a real aircraft. At the time of this writing, SDAR has been flying the simulation for nearly two years.

The simulation runs on a 20 MHz 80386 PC. The heart of the simulation is a high-fidelity, 6 degree-of-freedom aircraft dynamic model. This is the same type of model used in manned simulators and is capable generating each of the in-flight motions that are exhibited by a real aircraft. The simulation has been validated against flight test data from a number of real aircraft (the McDonnel-Douglas T-45, Cessna 152 and Piper Cherokee to name a few) and generates responses with a high degree of correlation.

To realize the simulated illusion of the UAV, a complete control/sensor suite was written on top of the aircraft simulation. Since SDAR is eventually to be flown on a real UAV, the suite is modeled after the real control systems and sensors expected for the flight test. The flight-test UAV has an autopilot on board which acts as the interface between the pilot and the aircraft. This autopilot was modeled and is the interface through which SDAR flies the simulated UAV. A complete sensor suite from the real UAV was also modeled. This includes the air data sensors, a Global Positioning System receiver, an Inertial Navigation System, the ESM receiver, a camera and a gimbal. Each of the sensor models is complete to the point of actually producing the measurement errors that would be generated by the real instrument in flight. Since there is no real camera image to work with, the vision system was also simulated and takes image data from the simulated camera. A simulation of wind and ships on the ocean surface completes the virtual world in which SDAR flies.

The simulation and SDAR run in lockstep. The simulation runs a one-second frame, then transmits all of the sensor data over to SDAR. SDAR runs one frame and transmits its control commands back to the simulated autopilot and camera gimbal controller.

The entire environment simulation runs about twice real-time. This promotes fast development/testing cycles very reminiscent of Forth. The same aircraft dynamic model was used by another group who programmed it in FORTRAN. Their simulation requires a Gould SEL (about $100,000 of computer) just to run real-time. The SDAR aircraft simulation, by itself, runs at about three times real-time -- on a $2,000 PC. Compare and save.