1996 Rochester Forth Conference
June 19-22, 1996, Ryerson Polytechnic University, Toronto, Ontario, Canada

The Sensor Driven Airborne Replanner Simulation: Building The World in a Box

Richard C. Wagner
Radical Novelties

Abstract

From 1989 to 1992, a small team of Navy Engineers developed the world's first fully robotic aircraft control system, the Sensor Driven Airborne Replanner. The simulation written to aid in the laboratory development and testing of this system is the most comprehensive avionics simulation in existence. Written in 16-bit PolyForth, the simulation accurately emulates an aircraft, an autopilot, a Global Positioning System receiver, an Inertial Navigation System, a camera, a camera gimbal, an imaging system, the complete suite of sensors found on the aircraft, ships on the ocean surface, and wind. In its most refined form, the simulation runs twice real-time on a 20 MHz 80386 PC, requires about 64 Kbytes of memory, and was nominated by the Office of Naval Technology for inclusion on SIMNET.

Introduction

The Sensor Driven Airborne Replanner (SDAR--pronounced, "ess-dar") was developed in response to a need expressed by the U.S. Navy. The objective of the SDAR project was to produce a control system which would provide for the robotic operation of Naval Unmanned Aerial Vehicles (UAVs) used for short- and medium-range reconnaissance. The objective of this type of reconnaissance is to provide a deeper knowledge of the surrounding environment, helping to improve a force's situational awareness. Systems providing this function are used to see over the horizon, giving advanced warning of enemy (or friendly) ships operating in the area, determining what those ships are doing, providing spotting services for shipboard artillery, and providing battle damage assessment of opposing forces. The requirement leading to a fully robotic UAV capability was the elimination of the radio up- and downlinks, which are normally used to control the aircraft, and to receive data from the aircraft. In battle, these act as beacons, announcing a force's presence to the enemy and acting as localizers for any type of radiation seeking weapon. The SDAR system eliminates these radio links by putting the intelligence of the shipboard mission planner, pilot, and camera operator into the aircraft in the form of an Artificial Intelligence system. (For more on the Sensor Driven Airborne Replanner, see reference 1.)

In its final form, SDAR allows the mission planner to program a provisional search path into the UAV before launch. He can also tell the SDAR system what kind of ships he finds interesting or uninteresting using any observable ship parameters (length, speed, course, and position). He can, for example, specify a great interest in ships between 500 and 700 feet long, sailing north at more than 20 knots. He can also specify, for example, that he has absolutely no interest at all in ships shorter than 150 feet. After the UAV is launched, the SDAR system flies the aircraft along the search path while scanning the ocean surface for ships using either a visible wavelength or infrared camera. Should it encounter a ship, the ship's parameters are immediately entered into the SDAR database. The ship's position will be updated automatically, even when the camera is not pointed at it. When the camera does point at the ship, the data in the ship's database entry is updated with the latest and most accurate measurements. If the ship is interesting (according to the mission planner's entries before launch) SDAR will fly the aircraft covertly to the beam of the ship, aim the camera at the ship, and either transmit or record a series of pictures. With each frame, the interest value of the ship is decremented, until it eventually becomes uninteresting. At that point, SDAR flies the aircraft covertly away from the ship, and back to the search path. If SDAR finds the ship to be uninteresting from the start, it will ignore the ship, or actually avoid it, if it is in the way. After the encounter, the ship's database entry will be maintained for a few minutes. Should SDAR encounter this ship again, its position will correlate with the entry in the database, indicating that this ship has already been "prosecuted". It is now uninteresting and SDAR will ignore or avoid it. After SDAR makes its last search path "waypoint", it returns the aircraft to the ship, where it is recovered. Thus, an entire surveillance mission can be flown without any radio transmissions at all, making UAV's operational in battle--a condition under which they were sorely needed, but grounded in the past.

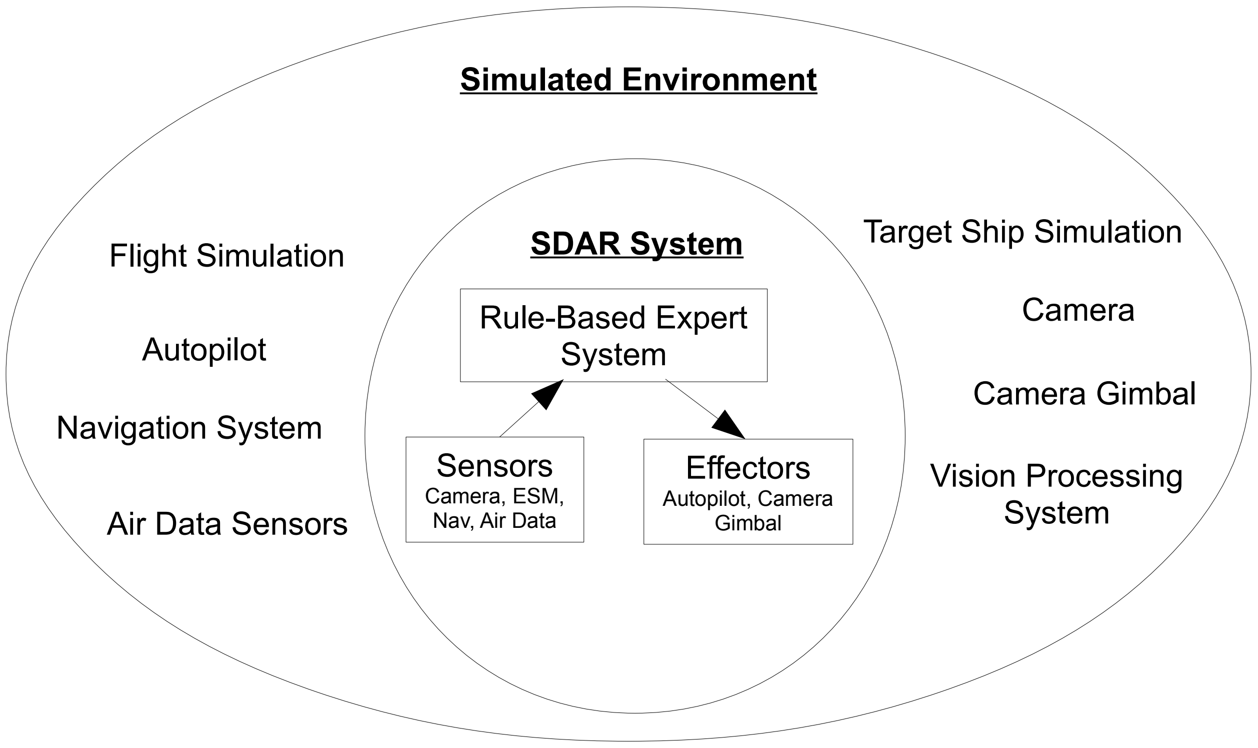

Figure 1: The SDAR system immersed in the simulated environment

Obviously, one doesn't go about developing a system like this directly on the aircraft--at least not if one wants to be successful. To develop and test SDAR, the team constructed a simulation into which they could immerse the SDAR system, emulating the data environment in which SDAR would operate. Figure 1 illustrates this idea. System development against a simulation has many advantages. First: proximity--neither the environment nor the airborne system ever get out of sight. When testing in a real aircraft, a huge capital investment is required to maintain some semblance of what one could call "situational proximity"--keeping abreast of what is going on in the air. (Remember, this aircraft is a UAV, you remain behind, on the ground.) But with a simulation, time can be stopped, and the developers can sit back, scratch their heads and think, "...okay, based on the situation at this point, why do you suppose *that* happened?" Second, if the simulation and airborne system are written well, they can both be run *faster* than real-time. At its fastest, the combined simulation/SDAR system ran about three-and-one-half times real-time. (As complex as the SDAR system is, this speed was always limited by the simulation. The SDAR system has the capability of running 650 times faster than it actually does.) This means that the typical 20-minute mission could be flown in less than six minutes in the lab, and provided for fast development/testing cycles. Third, a simulation offers a controlled environment. If, during a simulated flight, some kind of error occurs, the situation causing the error can be reliably reproduced by using the same starting parameters. Conversely, by randomizing starting parameters, thousands of different missions can be flown--supporting the development of a robust airborne system. Last, by developing against a simulation, there is no risk of losing the aircraft. Since these UAV's cost around one million dollars apiece, one doesn't actually flight test until *all* of the kinks are ironed-out.

The simulation was never a static entity. As more and different capabilities needed to be developed and tested within the SDAR system, the simulation evolved as necessary, acquiring new features, and having deeper detail written into existing features. Altogether, the SDAR simulation went through four major evolutionary phases.

The First Cut

SDAR didn't take its first, tentative steps within the confines of an embedded system. Instead, it was developed on a PC, where the brunt of a full-blown Forth system could be brought to bear upon it. The navigation routines were the first to be written. But these required some form of simulation for their testing. To provide for this, a very rudimentary flight simulation was written. The dynamic model embodied within this simulation considered the aircraft as a point-mass. This sounds very elementary, and it is, but it got the job done. In fact, this simple point-mass simulation proved far more useful than one might assume. It was retained through the entire development of the inference engine, AI rulebase, and support routines, and was only replaced after about two years, when SDAR was required to provide the new feature of camera gimbal stabilization.

In an aircraft dynamic model, the simple balance of lift, weight, thrust, and drag accurately produces the long-term, macroscopic motions that the aircraft generates in flight. In a real aircraft, there is always an energy balance between altitude and airspeed. As an airplane dives, it loses altitude. Potential energy, formerly in the form of altitude, is traded for airspeed (kinetic energy) as the airplane accelerates downhill. However, as the airspeed increases, so does the lift provided by the wing. This causes the airplane to level-off, raising the nose up, toward the horizon. These two actions, altitude loss and raising of the nose, are typically about ninety degrees out of phase. This means that, when the airplane's nose passes through the horizon, it has gained extra airspeed, causing the nose to rise above the horizon, initiating a climb. As the airplane climbs, it trades airspeed for altitude. It will eventually level-off, nose-over, and begin a new descent. This long-term rising and falling motion is called the "phugoid" mode of the aircraft. For most airplanes, the period of this motion is on the order of thirty seconds. The phugoid mode is very lightly damped (by nothing more than the drag of the aircraft), and will continue for many minutes after the aircraft is perturbed from straight-and-level flight. The point-mass dynamic model is capable of reproducing this phugoid motion from three second-order, linear, partial differential equations. In the simulation, the system was integrated with a simple Euler integrator, using a time step of one second. The natural frequency of the equation system is around 30 seconds, and a one-second time step is small enough to ensure numeric stability. In fact, it provides perfectly acceptable fidelity.

After the flight simulation was complete, development of the artificial intelligence system began in earnest. With both the simulation and the airborne system being developed on the same machine (in fact, within the same Forth system), concerns arose about the possibility of the two separate systems becoming intertwined. Each of the developers had experienced an occasion where, while developing two separate systems in one environment, routine "A", which was only to be used within one system, was also used in the other. When enough of these code and architecture crossovers occur, separating the two systems can be quite a chore. Thus, it was decided that a hard interface (well, architecturally hard, anyway) would be built between the simulation and SDAR. Any data going from the simulation to the SDAR system would only do so through a "sensor". And any data going from the SDAR system to the simulation would only do so through the autopilot, or the camera gimbal controller. This decision had the ancillary effect of keeping the developers honest. If some data was required for a routine, it could only be obtained by going through a sensor. Going and getting the "truth data" from the simulation was forbidden. Later on, this decision would turn out to be downright prophetic, but more on that later. Most of the sensors were very simple--fetching data from a simulation variable and storing it into an SDAR variable. Other sensors were more complex, taking raw data from the simulation and refining it for consumption within SDAR. An example of two simulated sensors is in Figure 2. The first line in the figure is the simulated altimeter. This is a simple sensor, where the altitude truth data is fetched from its simulation state variable (a floating-point variable) and stored into the SDAR altitude variable (an integer variable). The second line represents the SDAR compass, which is a more complex sensor. The heading data is fetched from its simulation state variable (floating point), changed from radians to degrees (`R>D`), then from a single length integer to an unsigned integer. Last, a large multiple of 360 degrees is added, and the modulus of 360 degrees is taken. The result is then stored into the SDAR heading variable (an integer).

```
          ALTITUDE S@ N> AURA-ALT !
YAW S@ R>D N>U 14400 + 360 MOD AURA-HEADING !
```

Figure 2: The simulated altimeter and compass

Note that the names of the SDAR state variables each begin with `AURA`. When the SDAR project was initiated, its was called the Autonomous Unmanned Reconnaissance Aircraft--AURA. This name seemed beautifully appropriate, being that it denotes something that is there, but remains unseen. Unfortunately, a bigger, more heavily funded project (A directed energy weapon--talk about an appropriate name!) also used this name. To avoid confusion, the name SDAR was adopted. References to the original name could be found throughout the SDAR code even at project completion--fossil remnants of an earlier time. (In the government defense arena, it's not uncommon to see a project's name change, or evolve over time. Part of the reason for this is that, when the funding for a project runs dry, the project can be renamed, the "new" project. receiving new funding.)

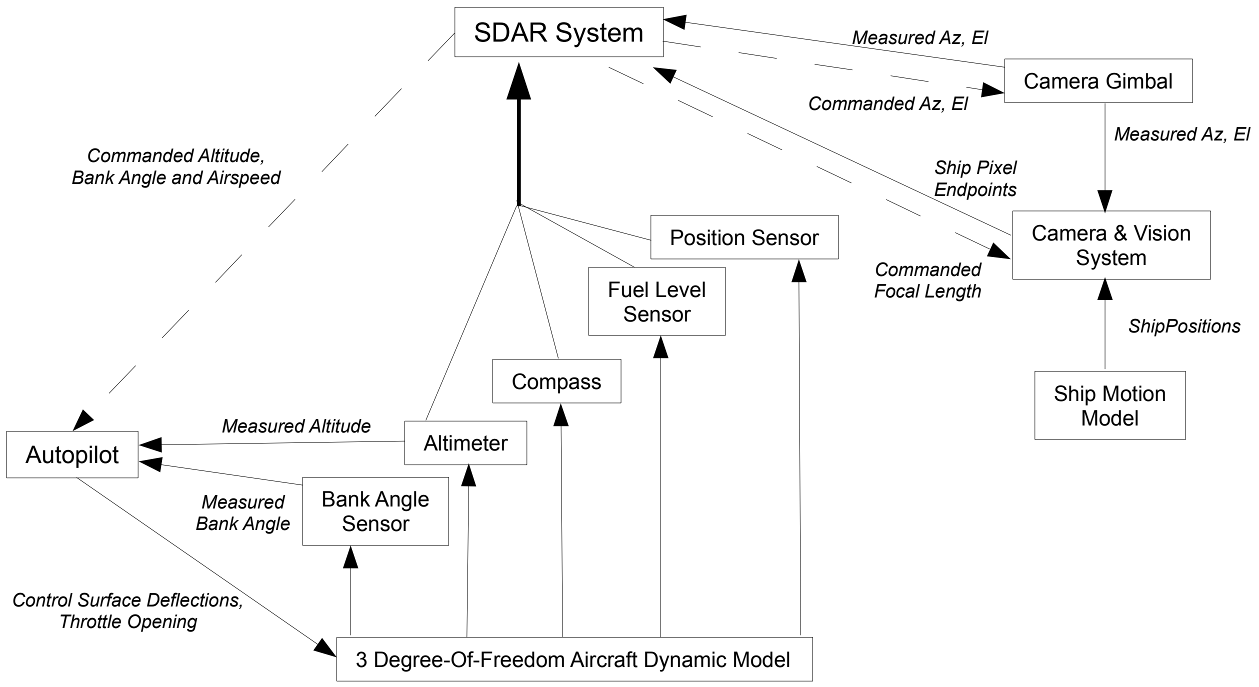

Figure 3: The initial simulation architecture

Figure 3 illustrates the state of the simulation at this point. The basis of the simulation is the aircraft dynamic model (the flight simulation). Around (or perhaps above?) this, because they all use its truth data, lie the sensors and actuators. The altimeter, fuel level sensor, and position sensing system (at this point, the type of system was undecided, but assumed to be GPS) are all simple sensors. The compass and bank angle sensors are each of the more complex type. The autopilot receives its data from the sensors, and its commands from the SDAR system. Its output goes back to the flight simulation, manipulating the aircraft's control surfaces and throttle. The gimbal is assumed to be inertially stabilized. This means that gimbal azimuth plus aircraft heading yields the direction the camera is pointing. Gimbal elevation is identical to the direction the camera is pointing, with respect to the downward vertical. The gimbal model receives its azimuth and elevation commands from SDAR, and sends the actual measurements (The gimbal doesn't move instantaneously. It's modeled by two second-order differential equations.) to both

SDAR and the camera model. The camera/vision system model has a number of inputs, and only one output. Using aircraft position, heading, and altitude, combined with gimbal angles and focal length (commanded by SDAR), a calculation is made to determine where the projection of the camera's field-of-view (FOV) lands on the ocean surface. Ship positions (from a first-order ship model) are then checked to determine which ships the camera "sees". The endpoints (on the ocean surface) of any ships within the camera FOV are then transformed back up (using a non-cartographic lens model) to yield camera pixel coordinates. This last transformation models the SDAR vision processing system, which only sends SDAR the pixel coordinates of ship endpoints within the camera FOV. (SDAR runs these same calculations in reverse. It takes the ship endpoints in camera pixel coordinates from the vision system, and transforms them, using aircraft position, heading, altitude, gimbal angles, and focal length, down to the ocean surface, again, using a non-cartographic lens model).

In a single simulation "frame", processing starts with the actuators (see Figure 4). Using the command inputs sent by SDAR during the last processing frame, the autopilot and gimbal controller are executed, generating their respective control outputs. Next, the flight and gimbal simulations (`DYNAMICS`) are each executed. A one-second integration step yields the aircraft and gimbal responses to this new set of control inputs. The sensor models are each executed at this point, generating the data that will be required by both SDAR and the camera model. The last dynamic model to run during the simulation frame is the ship simulation, updating each of the simulated ship's positions to the latest second. Last, the camera/vision system model is executed, generating the last pieces of data required for SDAR processing. The SDAR system is then run, resulting in SDAR generating a new set of commands for each of the control systems.

```
: SIMULATION ( --)
        AUTOPILOT      GIMBAL-CONTROL      DYNAMICS
        NAV-SENSORS    SENSE-GIMBAL
        SHIP-SIM       SENSE-IMAGE         SDAR ;
```

Figure 4: The top level simulation word

The Second Cut

From its conception, SDAR was intended to run on a stand-alone, single-board computer. More specifically, the development team intended to write the system using CMForth, and run it on an RTX2000-based machine. After a year to a year-and-a-half of development, SDAR had reached a state of robust maturity, successfully completing thousands of random missions. The next step in the development process was to realize the separation of the SDAR system and the simulation. What had previously been an architectural interface between the two systems, would now become a physical interface, as the SDAR system was moved to its own PC. Previously, interaction between SDAR and the simulation occurred in memory, as a data value moved from its simulation state variable to its SDAR equivalent through a sensor model. This interaction would now occur over a serial link. After running a one-second frame, the simulation transmits its data over the serial line to SDAR. SDAR then runs through one execution cycle, and sends its commands back to the simulation. As such, the simulation (and SDAR, of course) acquired a new data transmission and reception subsystem.

As Figure 3 illustrates, there are quite a few data values that get transferred from the simulation to SDAR. The number coming back is somewhat smaller. The process of keeping track of the set of variables to be transmitted and received threatened to become complex, error prone, and a pain. Additionally, it appeared that the transmitting and receiving routines would have to be modified each time a new variable needing transmission was used. To deal with this problem, a new "transmission" `VARIABLE` was defined. This new variable works, at run time, just as a normal variable does, leaving a pointer to its data on the stack. However, the data does not reside within the body of the definition, as it does in a normal variable. Instead, the data is kept in a contiguous area of memory, along with the data for all of the other transmission variables (see Figure 5). This way, the transmission variables can be used in the normal manner. However, instead of transmitting the data one value at a time, the entire, contiguous data block can be transmitted in one fell swoop.

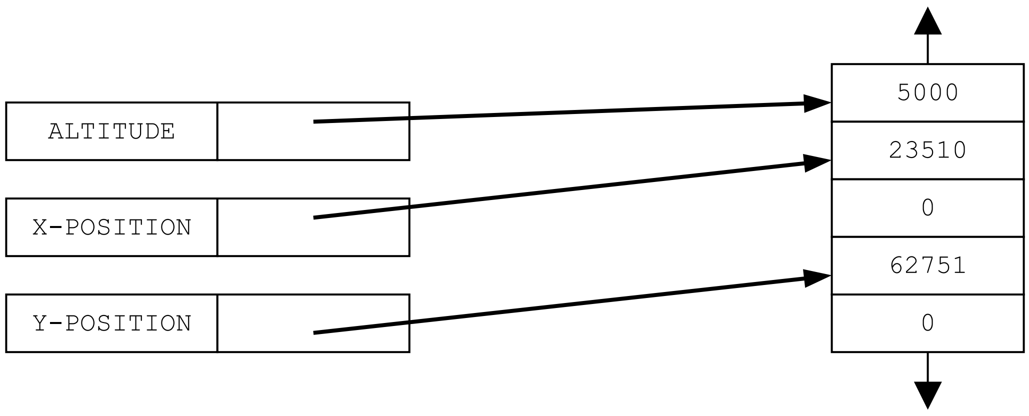

Figure 5: The transmission VARIABLE architecture

After simulation processing completes, the data transmission routine is entered. This begins with a loop in which the simulation continuously sends a "request to send" code and waits for an "acknowledge" code from the SDAR system. After this initial handshake is complete, the simulation streams the entire data block across the serial line, finishing off with a checksum. If SDAR calculates the same checksum, it sends an acknowledge code, and the process is complete. If the checksums do not match, SDAR sends an error code, an alarm is sounded, and the process repeats. In the two years in which this scheme was employed, the alarm was heard about three times. After the simulation data is successfully transmitted, the simulation enters its data reception routine, waiting for SDAR to finish its processing. When SDAR reaches its transmission routine, the entire process is repeated in reverse, filling the transmission variables associated with the simulation's control models.

With the attainment of the most significant milestone of the project (proving the SDAR concept works through the establishment of a fully functioning system), there were high-fives all around the SDAR lab. Development slowed dramatically for a few weeks, as a schedule of talks and demonstrations was undertaken. During this period, the sponsor (the UAV Joint Program Office--part of the Cruise Missile Program Office) decided that SDAR should incorporate a second mission sensor (the first being a camera). If this could be done, SDAR would be the first system to demonstrate a new concept called "Sensor Cross-Cueing", where one sensor is used to direct the actions of a second. This second mission sensor was a communications frequency Electronic Support Measures (COM/ESM) receiver.

Since World War II, radar has played a critical role in both offensive and defensive weapon systems. In many cases, it has become the only sensor employed to both identify and track a target. With the deployment of a new radar in the field by one force, there is usually a deployment, by the opposing force, of a parallel system used to jam, fool, confuse, exploit weaknesses, and generally counter the new radar. Systems like this, designed to deprive the enemy of his radar or radio capability, are called Electronic CounterMeasures, or ECM systems. Other systems with a similar, somewhat anti-parallel functionality have also been developed. These, rather than countering an enemy radar or radio installation, are designed to provide useful information on the offending system. In its insatiable quest to give everything in the world a three-letter acronym, the military has deemed these Electronic Support Measures, or ESM systems. The COM/ESM system employed by SDAR was a relatively new product at the time, its origins coming from the Tacit Rainbow missile program. The system provides two major pieces of information on any communications frequency transmitter broadcasting within range: the type of transmitter broadcasting, and the bearing, plus or minus ten degrees, to the transmitter. The SDAR system uses both of these pieces of data. First, to determine whether or not a transmitter is interesting to the mission planner, and second, to turn the camera to the bearing of the transmitter, to see if a potential target exists.

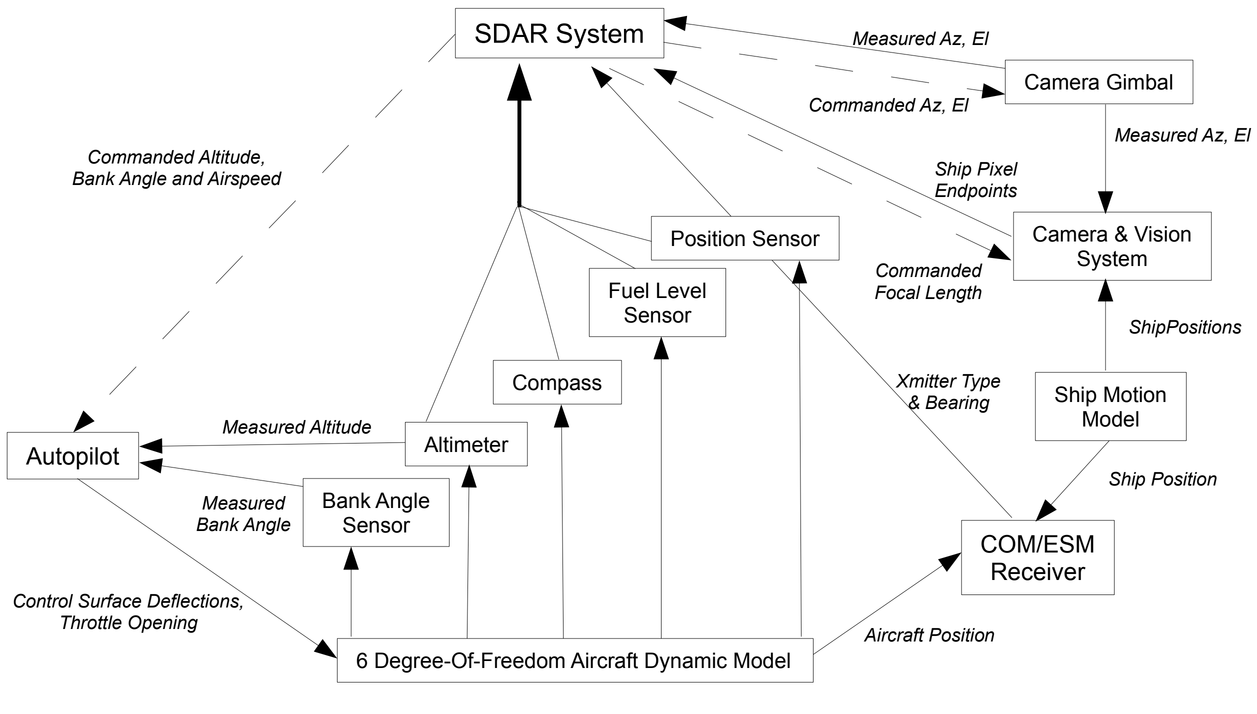

Figure 6: The simulation architecture with the COM/ESM receiver model

The relatively poor directional accuracy of the COM/ESM system concerned the development team. Up to this point, all of the sensors and actuators were considered to generate an insignificant amount of error. With this piece of equipment, however, this was certainly not the case. Questions arose: How would SDAR respond to such inaccurate measurements? Would this system provide enough accuracy to actually direct the camera to unseen ships, providing the sensor cross-cueing functionality the sponsor was looking for? The best way to find out was to make the simulated COM/ESM receiver as realistic as possible. Two new additions to the simulation accomplished this. First, each ship received a new transmitter model. This was simple. Each ship had two bytes added to its entry in the simulation database. The first byte indicates the type of transmitter the ship is carrying. The second indicates the transmitter status--on or off. A single word controls the transmitters, turning them on or off at statistical intervals, or on a schedule with the simulation time. The simulation architecture, at this point, is illustrated in Figure 6. The receiver model takes position truth data from the aircraft dynamic model, along with ship position data from the ship dynamic model, to determine the bearing from the aircraft to a transmitting ship. A 20-degree, Gaussian-distributed error is superimposed over the bearing truth data to arrive at the measured, or sensor data. During a simulation frame, the transmitter model is executed, turning the necessary transmitters on or off. Next, the receiver model is run, generating a list of transmitter bearings and types, which is sent to SDAR with the rest of the sensor data.

The COM/ESM receiver model was the first sensor model in the simulation to generate operationally representative errors. Were it not for the decision to include sensor models in the simulation, the addition of this new receiver model would have been far more difficult. As it was, the addition required only a few days of work.

The Third Cut

From the start, the development team knew the limits (as all good engineers should) of the simulation's aircraft dynamic model, and understood just how far it could be trusted to emulate the real world. Although the original flight simulation faithfully reproduced the large-scale body translations generated by the aircraft, it ignored any rotational dynamics that the real airplane would experience. A set of small-scale, translational dynamics, linked to the rotational dynamics, was also missing.

In a real aircraft, all motion occurs about the center of gravity (c.g.). If a force is applied to the aircraft, the new balance of moments about the c.g. will cause the aircraft to rotate. The rotation eventually ceases when other, rotation-sensitive moments come to balance the disturbing moment. With the change in orientation, the aircraft experiences changed body forces. This force imbalance causes a linear acceleration, changing the aircraft's flightpath--the direction it is flying. This process can also happen in reverse--a change in flightpath causing body rotations. These effects combine to cause an airplane to do a complex, delicate dance as it moves through the air.

About the lateral axis, the axis going from wingtip to wingtip, the airplane experiences the aforementioned phugoid oscillation. This is a slow, lightly damped oscillation in flightpath with no change in angle of attack (body rotation). A second type of motion about this axis is the "short period" mode. This is a change in angle of attack with little to no change in flightpath. This motion is highly damped, rarely lasting for even one complete period. The period of motion, for most airplanes, is on the order of tenths of a second. Motion about the vertical axis is linked to motion about the longitudinal (nose to tail) axis. A perturbation about the vertical axis will lead to a motion called "Dutch roll". This is roughly akin to the motion of a drunk as he staggers down the street. The airplane turns and banks left, and then right, as it snakes through the air. This motion has a period on the order of a few seconds, and is lightly damped. Once disturbed, an airplane can Dutch roll for as much as a minute.

To properly model all of this motion, the simulation was given a full, six degree-of-freedom (DOF) dynamic model. This is the same type of model used in manned simulators, and is capable of accurately reproducing all of the motions the real aircraft generates in flight. To be sure the model yielded acceptable fidelity, it was validated against flight test data from the Douglas T-45 Goshawk, the Cessna 152, and the Piper Cherokee 140. Correlation was excellent for all three aircraft. This model is far more complex than the original 3-DOF model. It is comprised of 9 second-order, non-linear, partial differential equations. An Adams-Bashforth 2 integrator was used to integrate the equation system, having been shown to provide the greatest fidelity (Cardullo, Reference 2). Since the shortest period of motion is on the order of a fraction of a second, an integration step of 0.02 seconds is used (with an Euler integrator, the timestep has to be even smaller). A larger timestep leads to poor fidelity. Larger still, leads to numeric instability. Because of this, the upgraded flight simulation runs much, much slower than the original.

At about the same time the simulation received its new flight model, the SDAR system was migrated from the PC to the RTX2000-based single board computer. A single SDAR frame requires only 1/650th of a second to execute on this hardware (compare this with the 1/3 of a second it required on the PC). This means that, as far as the human is concerned, the simulation runs wide-open, about twice real-time. Thus, there was only slightly noticeable performance degradation after the switch to the 6-DOF dynamic model.

With the new dynamic model, the effects of body rotation on the camera were fully realized. This forced development of more complex camera pointing and controlling algorithms, and culminated in the development of an SDAR subsystem capable of stabilizing an unstabilized camera.

The Last Cut:

Working for the Government is a rather unique experience. Projects like SDAR are usually developed at defense laboratories run by the Army, Navy, or Air Force (SDAR was developed at the Naval Air Development Center, Warminster, PA), or under commercial contracts let by these labs. While an Engineer or Scientist might be employed by the Lab where he works, the funding for his project, and consequently his paycheck, rarely, if ever, comes from the Lab itself. Instead, the funding comes from a "sponsor". This is an office, usually below the Pentagon level, that is responsible for a given type of work, weapon, or project. Each year, after The Budget is approved, offices within the Pentagon each get their allotment of the defense appropriation. These offices each have offices below them, which may have offices below them, and again, and again...who knows how many levels there may be? Eventually, what's left of the funding that started at the top makes its way down to the offices which fund the labs, and then into the labs themselves, where the work (sometimes) gets done. In an effort to try to make sure that something is being done for their money, most sponsoring offices hold a yearly "Program Review". Lucky sponsors get to see what has been done with their funding, as groups they've sponsored show off what they've accomplished over the year. Far too often, however, a sponsor receives a thick report, full of less-than-useful information he doesn't have the time to read anyway.

The SDAR team was scheduled for its 1991 yearly review shortly before Christmas. Over the year, SDAR had matured dramatically. It was now a robust, flight-worthy system, having been running on its airborne hardware for almost a year. It had also received a new state--the ESCAPE state--under which it could deftly negotiate a multi-ship encounter. With this last state, SDAR was bulletproof. It had been thrown into thousands of random simulations, most involving multi-ship scenarios, and was able to prosecute every ship without overflying or getting too close to a single one. SDAR had become so robust that the development team, even when trying, couldn't find a scenario that would cause SDAR to fail. True tests came as demonstrations were given. Any developer knows that, if a system is going to fail, it will do so when it's being demoed. But in demonstration after demonstration, SDAR just kept on running.

For the review, the team decided it would give a demonstration right in the sponsor's office. To accomplish this, they took an SDAR airborne system, a laptop to talk to the system, and a single floppy disk with a compiled version of the simulation on it, to Washington, D.C. The simulation was run on the sponsor's office PC with an overhead projection screen. By the end of the demo, the sponsors were beside themselves. This was the first time in their experience that a development team had something tangible (real hardware, that is) to show for the expended funding. They had one or two reservations, however. Since development of SDAR had been done strictly against a simulation, could the development team really guarantee that the system would function properly the first time out, when it would be attached to real hardware? They pointed out that many of the sensors onboard the real aircraft produced small but significant measurement errors. Might these errors cause the SDAR system to fail, or do something unexpected in flight? They also pointed out that the GPS satellite constellation was still incomplete. This meant that a GPS receiver would sometimes lose its line-of-sight with a satellite, going offline and returning no navigation data. These were valid concerns, and the SDAR team intended to address them.

Returning home, the team set to work realizing each of the sensor models that already existed within the simulation. With the data sheets on the actual airborne sensors SDAR would be attached to, the team added error models to each of the simulated sensors. Nearly all of the real sensors exhibited Gaussian distributed measurement error, and so this was used initially. However, the team soon realized that an even worse case would exist if they modeled the sensors using uniform random measurement errors. And so, most of the simulated sensors received a uniform error model. The sensors and their error models are listed in Figure 7.

Sensor	Error Model
COM/ESM Receiver	Gaussian, +/- 10 deg.
Altimeter	Uniform, +/- 20 ft.
Heading	Uniform, +/- 3 deg.
Bank	Uniform, +/- 3 deg.
Gimbal Azimuth	Uniform, +/- 3 deg.
Gimbal Elevation	Uniform, +/- 3 deg.
GPS	Statistical Dropouts; Gaussian, +/- 15 ft.; Gaussian, +/- 2 ft/sec
INS	Drift

Figure 7: The sensors and their error models

The GPS system was a more complicated problem. On the real aircraft, when GPS was offline, the SDAR system would have to revert to an Inertial Navigation System (INS). This type of system uses accelerometers and gyros to measure the attitude and accelerations being experienced by the aircraft, and integrates them to yield velocity and position data. To model this, the old position sensor was broken into two sub-sensors. At the base lies the INS system. This was modeled much like a dynamic simulation. A set of accelerometer offsets (a complication of INS is that the accelerometers don't necessarily read zero when the aircraft is not accelerating) were integrated to yield velocities, and then drift distances. These drifts would then be added to the flight simulation truth data, yielding a realistic, drifting INS position fix. The GPS receiver was modeled using two main parameters: the average time the receiver would be online, and the average time the receiver would be offline. With this, the receiver transitions from an online to an offline state and back at statistical intervals. When the receiver is online, position and velocity truth data is taken from the flight simulation and given a Gaussian error distribution to arrive at GPS sensor data. As long as the GPS system is online, the position sensor takes its data from the GPS receiver. When GPS drops offline, the position sensor initializes the INS drift distances to zero, and begins using INS position data. On the simulation monitor, two aircraft tracks appear when GPS goes offline. The first is where SDAR really is. The second, because of INS drifting, is where SDAR thinks it is.

Because the sensor models already existed, the process of upgrading them to model their airborne counterparts only required about a week of work. Fearing for the worst, the SDAR team began running simulated missions with the new, very complete simulation. Happily, the SDAR system chugged merrily along, shrugging off the new, error-riddled data, almost as if the development team was trying to pull a prank on it, and it knew better. The greatest surprise was the system's response to the GPS dropouts. When the GPS goes offline, the measured position of the aircraft slowly drifts away from its actual position. This process is slow and smooth, and so the team didn't expect to see any wild gyrations on SDAR's part when it happened. However, when GPS comes back online, the system experiences a sudden discontinuity. In the period of one second, its geographic position can change thousands of feet. The SDAR system tolerates this very well, however, simply plodding along, while making a course correction to get back to where it thinks it should be.

Conclusion

After the success of the yearly review, word of the SDAR system began to spread around the Center and to other Navy labs. For the next month or two, the team was back on the demonstration trail, showing off the system to interested groups, prospective sponsors, and even a few Admirals from other Navy bases, who had heard about the new, gee-whiz robotic system. Having a room full of technically trained people sit with their mouths open, marveling at SDAR, was very gratifying for the development team. Soon, they became rather accustomed to this type of reaction. During one demo, however, a member of the audience seemed perfectly oblivious to the SDAR system, and instead, asked a continual stream of questions about the simulation. This fascinated the development team, who had never considered the simulation as a product. Instead, it was only a tool they needed to get the job done. Attention, during development and especially during demos, was always focused keenly on SDAR. In fact, most audiences never gave the simulation a second thought. Yet this man couldn't seem to hear enough about it. In the end, it turned out that he was from the Office of Naval Technology. He was responsible for pushing the simulation envelope, in an effort to offer better, more comprehensive development and testing environments to groups around the country. Up to this point, he hadn't seen this complete a simulation, even on a mainframe. Here, it was running twice real-time, on a PC! And with that, to the astonishment of the development team, the simulation was nominated for inclusion on SIMNET.

References

1. Wagner, Richard C.; "The Sensor Driven Airborne Replanner: Artificial Intelligence for Unmanned Air Vehicles"; Proceedings of the 1992 Rochester Forth Conference on Biomedical Applications; pages 104-106; Institute for Applied Forth Research; Rochester, NY; 1992; ISBN #0-914593-12-9

2. Cardullo, Frank M.; "Mathematical Modeling"; Course notes for the Fifth Annual Flight Simulation Update; Thomas J. Watson School of Engineering, Applied Science and Technology; State University of New York at Binghamton; 1989