ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2613425

Achieving Fully-Automated Aircraft Flight with Limited Resources

Article · August 1996

Source: CiteSeer

CITATIONS
0

READS
10

3 authors, including:

Ella M Atkins
University of Michigan
263 PUBLICATIONS **5,882** CITATIONS
SEE PROFILE

Edmund H. Durfee
University of Michigan
345 PUBLICATIONS **9,082** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Cloud-enabled automotive decision-making systems View project

NASA's Generalized Trajectory Modeling and Prediction for Unmanned Aircraft Systems View project

All content following this page was uploaded by Ella M Atkins on 15 August 2013.

The user has requested enhancement of the downloaded file.

Achieving Fully-Automated Aircraft Flight with Limited Resources[1]

Ella M. Atkins **Edmund H. Durfee** **Kang G. Shin**

University of Michigan AI Lab
1101 Beal Ave.
Ann Arbor, MI 48109

{marbles, durfee, kgshin}@umich.edu

1 Introduction

Fully-automating aircraft flight presents a number of challenges for researchers. Perhaps most important is the criticality of real-time responses -- an aircraft cannot remain safe indefinitely once it has left the ground. Additionally, aircraft flight dynamics are complex and not fully-understood, so computations are often time-consuming and approximate at best. Current aircraft use low-level controllers to handle ordinary situations, but rely on human pilot commands to respond when emergencies or unexpected situations (e.g., course changes) arise. We are working to expand the role of autopilots to also handle emergency situations. Unfortunately, there are many possible emergencies, ranging from a failed sensor to collision-course traffic to unexpectedly flying into a tornado. In fact, the comprehensive set of possible emergencies is so large that it is infeasible to build a database of pre-planned responses. Instead, we approach the problem by building control plans to handle “expected”, or highly-probable, situations. Then, we detect and replan to handle unexpected situations or emergencies as they arise.

Flexible computation is very important during these replanning phases. Ideally, the planner could carefully expand all reachable states, selecting actions to keep the plane safe while reaching the goal. However, a quick response is usually necessary because the aircraft cannot just “stop” and wait for a new plan. For example, if all engines fail, the new plan must be completed in time for the aircraft to fly to a desirable landing location (e.g., nearby airport or open field) before it loses too much altitude. Thus, we should restrict planning time to allow production of an approximate plan in time to execute successfully. If planning time permitted, this plan may be able to reach the desired goal. Alternatively, if allowable planning time was brief, this approximate plan may simply maintain a safe set of states, buying the time required to develop a complete plan that can reach the goal.

We have investigated replanning for unexpected situations during aircraft flight in the context of CIRCA (Cooperative Intelligent Real-time Control Architecture) [Musliner], which combines a planner, scheduler, and separate real-time plan execution module such that plans are built, scheduled, then executed with real-time guarantees of system safety. Section 2 briefly describes CIRCA, including recent enhancements that admit probabilistic knowledge specification as well as the detection and handling of unexpected situations. Section 3 describes flight simulation tests, followed by a discussion of future additions we feel are required to achieve a sufficient level of robustness when replanning must occur rapidly (Section 4).

2 CIRCA Background

Figure 1 shows the general architecture of the CIRCA system. The AI subsystem (AIS) contains both the planner and the scheduler. The "shell" around all AIS operations consists of meta-rules controlling a set of knowledge areas. Working memory contains tasks that are ready to be executed, including planning, downloading plans from the AIS to the real-time subsystem (RTS), and processing feedback from the RTS. The CIRCA domain knowledge base specifies a list of goals which will enable the system to successfully reach its final goal. The world model is created incrementally based on the initial state set and the set of temporal and action state transitions. The planner selects actions based on estimated "cost vs. benefit" and backtracks if the action does not ultimately help achieve a goal or avoid failure (e.g., striking an obstacle or crashing an airplane). CIRCA minimizes its use of memory and time by expanding only states explicitly produced by transitions from initial states or their descendants. State expansion terminates whenever all specified goals have been reached while avoiding failure states.

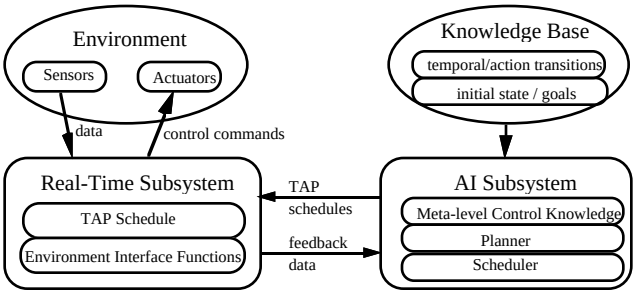

Figure 1. CIRCA Architecture.

[1] This research was supported by NSF Grant IRI-9209031.

CIRCA's control plans are represented as cyclic schedules of test-action pairs (TAPs). Typical tests involve reading sensors and comparing the sensed values with certain preset thresholds, while actions involve sending actuator commands or transferring data between CIRCA modules. When the AIS planner creates a TAP, it stores an associated worst-case execution time and execution deadline to enable safety guarantees. These TAP attributes are used by a deadline-driven scheduler [Liu] to create a periodic TAP schedule. If the scheduler is unable to create a schedule to support all deadlines, the AIS backtracks to the planner, which then removes states or selects different actions. The AIS downloads a successfully-scheduled TAP plan to the RTS, which then executes it.

Recently, we have improved CIRCA in two respects. First, we sought to relax the absolute 100% safety guarantee requirement, at least when planning and scheduling become prohibitively difficult. To do this, we have implemented a model in which transition probabilities (specified as functions of time) are used to approximately compute state probabilities [Atkins]. These probabilities are used by the planner so that state expansion occurs in decreasing order of probability. This provides two advantages: (1) highly-probable goal paths are identified, saving the planner time by not requiring expansion of all potentially goal-reaching states, and (2) highly-improbable states may be removed from consideration when scheduling is impossible.

We also have incorporated algorithms for detecting and handling certain classes of “unplanned-for” states. Figure 2 shows the relationship between subclasses of possible world states. Modeled states have distinguishing features and values represented in the planner knowledge base; we have not considered methods (e.g., discovery) to handle unmodeled states. The planned-for set are states from which failure is avoided. Handled states are on a goal path, while deadend are not. The planner can model other states, including those that are reachable but “removed” due to resource limitations, and “imminent-failure” that are not considered reachable but, if reached, lead directly to failure.

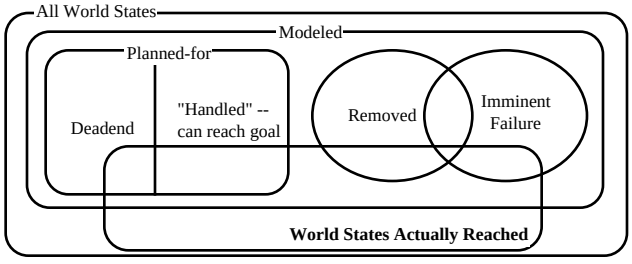

Figure 2. World State Classification Diagram.

As shown in Figure 2, states actually reached may include any subclass. We detect all deadend, removed, and imminent-failure states, and replan to handle each if it is reached. While our algorithms successfully build minimized tests (using ID3 [Quinlan]) to detect these states, we seek ways to improve efficiency. With large sets of unplanned-for states, creating detection tests might take longer than the planner can afford. We are considering methods to build approximate detection tests quickly. Such tests will not identify all unplanned-for states, but may be required if the CIRCA planner is to admit execution time bounds, as discussed below.

3 Flight Simulation Tests

To test our local probabilistic model and unhandled state detection capabilities in the aircraft domain, we interfaced the ACM F-16 flight simulator [Rainey] to a low-level Proportional-Derivative (P-D) control system which calculated the proper actuator commands to achieve a commanded altitude and heading. CIRCA's RTS issued commands to the low-level controller, which then computed actuator commands. Modeled state features include altitude, heading, location (or “FIX”), gear and traffic status, and navigation sensor data. CIRCA successfully controlled the aircraft during normal "flight around a pattern" (illustrated in Figure 3) from initial takeoff through a full-stop landing on the runway.

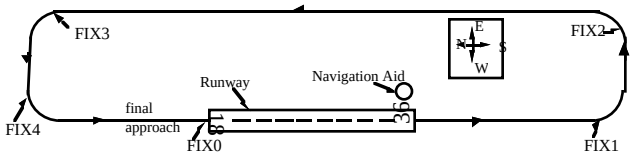

Figure 3. Flight Pattern Flown during Simulation.

We have tested our new algorithms using two emergencies: “gear fails on final approach”, and “collision-course traffic on final approach” [Atkins]. In either situation, failure to notice and react to the problem will result in a crash. A series of test runs provided examples of each possible unhandled state type: deadend, removed, and imminent-failure. A comparison of tests with and without the unhandled state detection/reaction routines demonstrated that the aircraft had a better chance to land safely when the unhandled states were detected than when they were ignored.

4 Limiting Planning Execution Time

CIRCA's reaction to unhandled states is coincidentally real-time, and was successful during our relatively simple tests. However, this may not be acceptable when unhandled states quickly lead to failure. Timely reactions may be achieved either by bounding replanning execution time or by building reactions in advance.

As planning technology progresses, more architectures employ methods for computing and adhering to planner

execution time bounds, as discussed in works such as [Dean], [Horwitz], [Ingrand], and [Zilberstein]. We feel a combination of these and other innovative ideas is required to achieve a near-optimal balance between planning time and plan quality. We would ideally like to use a planner's initial state(s) to quickly compute an initial estimate of a planning time limit, then modify this estimate based on environment changes during planning. Assuming an algorithm to compute a planning time limit, we could use a type of design-to-time algorithm [Garvey]. We are now expanding states in inverse order of probability, so the most probable states will be handled first. Whenever the planner's execution time expires, the probability of the next state to be expanded will be the maximum probability of reaching any single unplanned-for state.

There are still potential quality problems with this algorithm. Perhaps most important, with a short planning deadline, high probability states may remain unexpanded. The planner may make use of its knowledge about probabilities and temporal delays, as well as knowledge about relative "importance" among states, to make tradeoffs that may improve planning. State expansion may be ordered by decreasing utility u(s), as shown in equation (1), where p(s) = probability of reaching state s, t(s) = minimum time before the system can reach state s, pf(s,n) = probability of reaching failure in n (or fewer) steps from state s, and a, b, and c are (as yet undetermined) scaling constants. By expanding states in this order, we hope to plan for the most "important" states, achieving a balance between state probability, system safety (i.e., prioritizing expansion to handle states that can reach failure), and the time horizon considered by the planner (i.e., near-term states are handled; far-term states will be handled by subsequent plans). Unfortunately, even the best u(s) will not guarantee a high-quality plan when the planner must execute quickly, so we continue to search for ways to achieve the ever-elusive balance between quality and execution time.

$$u(s) = a * p(s) + b * t(s) + c * pf(s, n) \quad (1)$$

5 Conclusion

Aircraft control is not a "solved problem" -- uncertain or unexpected events such as collision-course traffic may occur. Most emergency situations are now handled by pilots, and even the best pilots may select suboptimal responses. Fully-automated aircraft flight will require approximate knowledge, and quite a large knowledge base will be necessary. Limiting planning time and carefully scheduling planned actions to guarantee critical responses will be crucial for safe, fully-automated flight in which all expected time-critical operations are part of a scheduled plan and have near 100% chance of executing properly, while emergency situations are handled by some combination of pre-planned reflex actions and fast replanning. We have begun to investigate methods to build plans that handle such emergency situations, including necessary feature-value specifications for aircraft, types of time-dependent temporal transition probabilities to use, and available sources of statistical data (e.g., FAA accident reports). We have tested simple situations such as "collision-course traffic" and "landing gear fails on final approach", but will need to significantly enhance CIRCA's planning and temporal reasoning capabilities before a full-scale aircraft control knowledge base could be handled with sufficient speed and accuracy to be trusted during emergencies.

6 References

E. M. Atkins, E. H. Durfee, and K. G. Shin, "Plan Development using Local Probabilistic Models," to appear in *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference*, August 1996.

T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning with Deadlines in Stochastic Domains," *Proc. of AAAI*, pp. 574-579, July 1993.

A. Garvey, M. Humphrey, and V. Lesser, "Task Interdependencies in Design-to-time Real-time Scheduling," *Proc. of AAAI*, pp. 580-585, July 1993.

E. J. Horvitz, "Reasoning under varying and uncertain resource constraints," *Proceedings AAAI-88*, AAAI, pp 111-116, 1988.

F. F. Ingrand and M. P. Georgeff, "Managing Deliberation and Reasoning in Real-Time AI Systems," in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 284-291, November 1990.

C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46-61, January 1973.

D. J. Musliner, E. H. Durfee, and K. G. Shin, "World Modeling for the Dynamic Construction of Real-Time Control Plans", *Artificial Intelligence*, vol. 74, pp. 83-127, 1995.

J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

R. Rainey, *ACM: The Aerial Combat Simulation for X11*. February 1994.

S. Zilberstein, "Real-Time Robot Deliberation by Compilation and Monitoring of Anytime Algorithms," *AAAI Conferenc*e, pp. 799-809, 1994.

View publication stats