Final Approach – An Automated Landing System for the X-Plane Flight Simulator

Oren Hazi
CS 229 Final Project

Introduction and Motivation

Automated aircraft landing systems are typically employed by commercial flights that routinely operate at airports with poor visibility. The best systems currently in use are autopilots designed to work in conjunction with a Category IIIc Instrument Landing System (ILS). An ILS is a ground-based radio navigation system that provides an aircraft with precision guidance along the approach path to an ILS runway. ILS hardware must be installed at the approach end of every ILS runway, and typically costs on the order of tens of millions of dollars to install (even more for Cat IIIc systems). As a result, they are usually only installed at airports with scheduled airline traffic, and are therefore not as available to general aviation (GA) pilots. Autopilots with automated landing capability are also quite expensive, and for similar reasons, are almost never installed in GA aircraft.

The quality of instrumentation systems installed in GA aircraft has been improving rapidly in recent years, largely due to Moore's law. GPS receivers with wide area augmentation system (WAAS) support have become ubiquitous. “Glass cockpit” systems (typically composed of several bright LCD screens) provide improved situational awareness and are now offered as standard features in many light aircraft. These systems can render simulated 3D views of outside terrain with “highway in the sky” style navigational overlays. It is also fairly common to see accurate inertial measurement units, radar altimeters, FLIR cameras, traffic avoidance systems, and weather radars installed in new high performance piston and light turboprop aircraft. Modernized flight management and automation systems that take full advantage of these instruments will likely play a crucial role in GA aircraft of the future.

The goal of this project is to use apprenticeship learning to build an autopilot capable of landing a small airplane at a specific airport from a final approach at traffic pattern altitude, relying only on instruments installed in the aircraft. The X-Plane flight simulator is used to test and demonstrate the autopilot, as testing in a real aircraft would be dangerous and prohibitively expensive. X-Plane is a commercial flight simulator developed by Laminar Research that uses blade element theory to model aerodynamic forces on the various parts of an aircraft in real time, resulting in realistic behavior, even in complex aircraft. X-Plane is used extensively by NASA and by aircraft manufacturers to simulate real aircraft during research and development.

Apprenticeship Learning

Markov Decision Processes (MDPs) provide a useful framework for finding optimal behavior using reinforcement learning. Policy exploration in systems that have large state spaces can be computationally challenging, especially if the dynamics of the system are unknown or difficult to model. Additionally, in order to guarantee full coverage of the state space, efficient exploration policies often tend to preferentially explore states that are not yet well-characterized. There are many systems for which this kind of exploration would be dangerous, such as nuclear plant controllers or aircraft autopilots.

Apprenticeship learning is an alternative to optimal exploration policies where a human expert guides

policy exploration by manually performing a desired task. This has been shown to result in nearly optimal performance relative to the performance of the human, and is computationally efficient [1]. Apprenticeship learning has successfully been used to learn the dynamics of a radio controlled helicopter, which exhibits a flight model that is arguably more complex than that of an airplane [2].

State Variables

The primary unknown state variable when landing an aircraft in instrument meteorological conditions (IMC) is the position of the aircraft relative to the desired touchdown point. Until recently, this information was difficult to obtain accurately without complex radio-navigation hardware and expensive installations near the runway and along the approach path.

WAAS enabled GPS receivers have a lateral accuracy specification of 25 ft, but have an accuracy of about 3 ft in practice. The measured accuracy is more than adequate for aligning an aircraft with a runway, and even the nominal specification would be a reasonable estimate at most airports in an emergency.

The vertical accuracy of WAAS GPS is specified at 25 ft, and measured at about 4 – 5 ft. Misjudging the touchdown height by 25 ft would likely result in disaster (either by beginning the flare too early and stalling the aircraft at a deadly altitude, or by extending the approach and crashing the airplane into the runway). Pressure altimeters can provide an accurate differential height, but the sea level reference pressure must be constantly adjusted. Forgetting or incorrectly adjusting the reference pressure can result in altitude variations of hundreds of feet. In this project, the simulated aircraft will be equipped with a radar altimeter, which can easily be made accurate to within 1 ft, which is adequate for landing safely.

Since radar altimeters report the distance from the aircraft to the ground, it is only usable once the aircraft is over the runway. Before then, it will be affected by any buildings and trees that are directly below the aircraft. The pressure altimeter will therefore be used on final approach to maintain the desired rate of descent.

The remaining instruments (attitude, airspeed, RPM, manifold pressure) are accurate and precise enough to use directly. Vertical speed is calculated by differentiating the pressure altitude measurements, as the vertical speed indicator tends to lag by several seconds.

The following state variables were measured and used by the learning algorithm for this project: pressure altitude, radar altitude, pitch, roll, yaw, heading, vertical speed, pitch derivative, roll derivative, yaw derivative. The actions used were: yoke pitch position, yoke roll position, throttle position. In addition to these variables, the GPS latitude and longitude were used by the planning algorithms. I used a PC joystick to fly the demonstration approaches in X-Plane, but did not have access to a set of rudder pedals, so rudder position measurements were excluded from the experiments.

Training

I wrote an X-Plane plug-in that can export flight parameters from within the simulator. When enabled, the plug-in samples the appropriate flight instruments and control positions at a rate of 5 Hz and writes them to a text file. This allowed me to fly multiple training approaches and experiment with various

post processing techniques afterwards, and perform the regression analysis and model estimation in a high level programming language (The X-Plane SDK is written in C, data processing was done using Python).

I flew approximately 100 training approaches and landings from within X-Plane. These approaches were made in a Cessna 172 Skyhawk from ten miles out to runway 5L at Edwards Air Force Base (KEDW). The sample data was separated by altitude into two phases: approach and flare. The approach phase included any portion of the state and action data where the aircraft was higher than 10 ft above ground level (agl) as measured by the radar altimeter, and the remaining data was used for the flare phase.

The state variables were finely binned in order to make the problem more tractable. The discretized states were then linked to result states (the state sampled 1/5 second later) via the action performed in the original state. Whenever multiple actions connected an initial state and a result state, the values of each of the action measurements were averaged. This resulted in more accurate model estimation, as some state transitions were caused by transient effects (such as wind gusts) rather than pilot input.

Training for the approach and flare phases was performed separately. The goal for the approach phase was to learn as much about the dynamics of the aircraft as possible (in normal flight) so that arbitrary approaches could be made to any runway at different speeds, angles, and power settings. The flare is performed when the aircraft is in ground effect and flying at or near its stall speed, so we expect the dynamics to be significantly different from the normal flight dynamics found during the approach phase.

In the approach phase, the pressure altitude and radar altitude measurements were not included as state variables, as these are not pertinent to the dynamics of the aircraft and would only serve to spread similar behavior among a multitude of states. The dynamics do not change very much when the aircraft is flying at 1500 ft vs. 150 ft. If action a_1 connects states s_1 and s_2, and action a_2 connects states s_3 and s_4, but (s_1, s_3) and (s_2, s_4) differ only in their altitude, then the similar actions a_1 and a_2 would be separated if altitude was part of the state. Excluding altitude allows us to learn much faster, since we do not have to wait to reach a specific altitude during the next approach in order to see another potential action for a given state. New potential actions for a given state can occur at any time during the approach.

In order to record control input actions from as many states as possible, the approaches were flown with severe turbulence enabled in the simulator. As the turbulence knocked the aircraft into situations with unusual attitudes and accelerations, the data logger recorded my response inputs to states that would not have been explored while making an approach in calm conditions.

Planning

Once the dynamics of the aircraft have been estimated, a planning algorithm specifies costs for given states. The model estimate is used to guide the aircraft to the states associated with the lowest costs. Separate planning algorithms were used for the approach and flare phases of the flight. The auto-land plug-in transitions from the approach algorithm to the flare algorithm when the aircraft reaches 10 ft agl.

The planning algorithm for the flare simply specifies that the wings should be kept level while power is gradually reduced and the pitch angle is raised. The values for power reduction rate, pitch rotation rate,

and maximum pitch angle were determined experimentally by hand flying the approach and having the auto-land plug-in take over at 10 ft agl. The cost for states that were not along the ideal trajectory were increased linearly with the varying parameter.

Two planning algorithms were written for the approach phase. The first was a naïve local planner. This algorithm simply keeps the wings level and maintains a constant airspeed and vertical descent rate, again with a cost that increases linearly with the varying parameter for any non ideal states.

The second was a Geo-referenced planner. This algorithm computes a straight line between the initial position of the aircraft (when the auto-land plug-in is enabled) and the desired touchdown point. The costs again increase linearly for states where the vertical or horizontal distances to the path are nonzero. The cost increases faster in the vertical direction.

Results

The auto-lander performed well when using the local planner and the flare planner. Since this configuration did not include any directional control, the aircraft heading would very gradually drift to the right. This behavior is almost certainly due to P-factor (asymmetric propeller thrust at nonzero angles of attack). As a result, this system could not be used to reliably land on a runway, and was demonstrated by landing in an empty part of Rogers Dry Lake near KEDW. I attempted to add an additional linear constraint on the aircraft heading to keep it pointed in a consistent direction, but this caused the aircraft to undergo divergent rolling behavior that eventually led to a crash.

The Geo-referenced planner exhibited a similar oscillatory roll behavior, and I strongly suspect that this was due to the lack of rudder input in the system (since my joystick lacked a yaw axis). As the rudder was not being controlled at all by this system, and I had no way to train it to use the rudder, I designed a separate PID control loop to drive the rudder. The control position was set to the direction of the straight-line-to-runway path computed by the Geo-referenced planner. This resulted in some improvement, but the aircraft would still occasionally roll erratically. Adding an additional PID loop to damp the aileron input fixed the roll problem entirely and allowed the Geo-referenced planner to easily make the runway from a half-mile approach. It occurred to me later that the behavior of the PID controllers could easily be simulated using additional linear cost constraints on the roll angle, yaw angle, roll derivative, and yaw derivative, but I didn't have time to test this before the project was due.

Strong crosswinds were also problematic. This is almost certainly due to lack of rudder control, as proper rudder technique is crucial for making a safe crosswind landing. This problem cannot be fixed with a simple yaw damper, but would need a more complex flare planning algorithm that can compensate for lateral drift.

During the flare, the aircraft would occasionally “balloon” to an altitude above 10 ft agl, which caused the auto-lander to return to using the approach planner. This resulted in porpoising around 10 ft, but this was never observed to occur for more than two iterations. This problem can easily be overcome by using a more complex plan transition model, but this was not tested due to time constraints.

Finally, although the Geo-referenced planner can successfully land an aircraft on a runway, its behavior is still fairly naïve. If the auto-lander is enabled on a very long final approach, the aircraft will follow a shallow descent to the touchdown point. In reality, this can be dangerous, as obstacles such as buildings, trees, towers, power lines, and even terrain can intersect with the planned path when the

aircraft gets low enough. It would be safer to maintain altitude until a specified distance from the airport and then begin a normal descent. Conversely, if the auto-lander is enabled too close to the airport, the aircraft will follow an approach that is too steep, and the aircraft will crash into the runway. The correct behavior here is to go around and attempt the approach again from a safer distance. The planner is also unable to fly a standard “box” traffic pattern if the aircraft is not aligned with the runway when the auto-lander is enabled.

There is obviously significant room for improvement in the planning algorithms for a real autopilot, but this project has shown that the dynamics of the aircraft can be estimated using apprenticeship learning, and that it is possible to design planners that are capable of making safe, repeatable landings.

References

1. Exploration and apprenticeship learning in reinforcement learning, Pieter Abbeel and Andrew Y. Ng. In *Proceedings of the Twenty-second International Conference on Machine Learning*, 2005.

2. An application of reinforcement learning to aerobatic helicopter flight, Peter Abbeel, Adam Coates, Morgan Quigley, Andrew Y. Ng. In *NIPS*, 2007.

3. Autonomous Autorotation of an RC Helicopter, Pieter Abbeel, Adam Coates, Timothy Hunter and Andrew Y. Ng. In *11th International Symposium on Experimental Robotics (ISER)*, 2008.