Real–Time UAV Maneuvering via Automated Planning in Simulations

Miquel Ramirez\(^1\), Michael Papasimeon\(^2\), Lyndon Benke\(^2\), Nir Lipovetzky\(^1\),
Tim Miller\(^1\), Adrian R. Pearce\(^1\)
\(^1\)School of Computing and Information Systems, University of Melbourne, Australia
\(^2\)Defence Science and Technology Group, Australia
miquel.ramirez@gmail.com, \{nir.lipovetzky, tmiller, adrianrp\}@unimelb.edu.au
\{firstname.lastname\}@dst.defence.gov.au

Abstract

The automatic generation of realistic behaviour such as tactical intercepts for Unmanned Aerial Vehicles (UAV) in air combat is a challenging problem. State-of-the-art solutions propose hand–crafted algorithms and heuristics whose performance depends heavily on the initial conditions and specific aerodynamic characteristics of the UAVs involved. This demo shows the ability of domain–independent planners, embedded into simulators, to generate on–line, feed–forward, control signals that steer simulated aircraft as best suits the situation.

1 Application Domain

In computational operations research (OR), multi-agent simulations (MAS) are often used to model, analyse and understand complex socio-technical systems [Heinze et al., 2008]. In the defence domain, such simulations are used to support the acquisition of new aircraft, to evaluate system upgrades, to assess tactical behaviour [Heinze et al., 1998; Tidhar et al., 1998] and to explore future operational concepts such as employment of autonomous systems [Byrnes, 2014].

Multi-agent simulations of air combat are challenging due to both the highly dynamic and adversarial nature of the domain and the complexity in the systems and the team tactics being modelled. These challenges manifest themselves across the entire spectrum of the software engineering and operational analysis processes, from specifying complex team tactical behaviour [Heinze et al., 2000; Evertsz et al., 2015], up to representing these complex behaviours within agent reasoning frameworks for verification and validation.

2 Problem Scenario

In this demo, we consider a simulated adversarial scenario consisting of two UAVs. The goal of each UAV is to maneuver itself behind the other and to maintain this for a certain period of time. In the terminology of air combat this is known as a stern conversion [Shaw, 1985]. The purpose of a stern conversion is to put the target aircraft in the right position to satisfy specific engagement criteria. These include constraints on the distance to the target, relative angles between the directions of aircraft motions, speed (both in absolute and relative terms) and altitude, which all need to be upheld over a given period of time. The purposes of maneuvering and maintaining a relative astern position go beyond the engagement of a weapon system. These may include employing a sensor to positively identify a target aircraft, positioning for flying in formation, or following ground vehicles or surface vessels in civilian surveillance operations.

3 ACE Multi-Agent Simulation Environment

In this demonstration we show how automated planning [Ghallab et al., 2004] can be used in the context of a MAS environment called ACE (Air Combat Environment) [McDonald et al., 2015]. ACE is a team-oriented MAS currently under development by the Australian Defence Science and Technology (DST) Group. ACE is designed to simulate teams of aircraft in adversarial n-versus-m air combat missions to conduct OR studies. ACE is used to both inform the acquisition of new aerospace systems and explore how best to employ them\(^1\).

A typical adversarial scenario modeled in ACE consists of two UAVs on opposing sides, blue and red, with the goal of each UAV being to successfully engage the other. Each UAV can be modeled either with simplified or high–fidelity flight dynamics, sensor and decision making models, which can be selected dynamically when the simulation is initialised. In addition to the core simulation kernel, ACE also provides analysts with tools for specifying scenarios as well as the capability to export the histories resulting from the simulation in formats amenable for 3D visualisation and statistical analysis [McDonald and Papasimeon, 2015]. Regarding decision making frameworks, at the timing of writing this, ACE allows the simulated pilots to be implemented via scripts, finite–state machines (FSMs), two–player hybrid game controllers [Isaacs, 1965; Park et al., 2016], and model–based predictive control [Camacho and Bordons, 2013] via hybrid planning [Fox and Long, 2006]. This demonstration focuses on the last type of controller, whose technical foundations and interest from an innovation perspective are discussed next.

\(^1\)For more details we refer the reader to the DST Group website for the research team responsible for ACE-2: https://www.dst.defence.gov.au/capability/aerospace-capability-analysis.
4 PDDL+ Domain Predictive Control

Model (Based) Predictive Control (MPC) refers to a range of control methods, rather than a specific control strategy, which make explicit use of models of processes — aircraft dynamics in our case — to obtain the control signal by minimizing an objective function [Camacho and Bordons, 2013]. While MPC is a general framework, most existing approaches have trouble dealing with systems where dynamics can reconfigure spontaneously or where they are required to handle constraints that rule out specific combinations of control inputs.

The Domain Predictive Control (DPC) framework [Löhr et al., 2012] exploits the observation that both of these aspects are a central part of domain–independent automated planning [Ghallab et al., 2004]. Like MPC, DPC uses an explicit model to predict future states, but instead of relying on ad–hoc descriptions of states and transitions, these are compactly described by means of a domain description given in a formal abstract language. This effectively decouples the model from the algorithms used to seek sequences of control signals that steer aircraft towards goal states. The ACE MPC module implements L¨ohr’s DPC framework with a twist. Instead of relying on linear dynamics that can be solved analytically and then used to construct a numeric planning domain description [Fox and Long, 2003], PDDL+ [Fox and Long, 2006] is used instead. This allows for the representation of arbitrary hybrid dynamical systems [Goebel et al., 2009] directly, to model the simulated aircraft dynamics. Since the dynamics, control inputs and associated constraints are given in a symbolic, declarative form, direct manipulations like that of “relaxing” the fidelity of the dynamics used by the simulation do not require any programming.

At every time step of the simulation where the pilot agent is required to generate a control signal, a call is made to a hybrid planner2, which seeks plans for goals, in our case, steering aircraft to be astern of the target, by means of heuristic search. Plans are then interpreted into control signals in a straightforward manner, projecting the trajectories induced by them over the variables that keep track of the evolution of control signals over time. While in principle, any hybrid planner could be used off-the-shelf, we have found it necessary to develop our own planner3 that operates in a different manner than existing systems. The reason for this follows from observing that the temporal distance between current simulation states and those where the controlled aircraft is astern of the target, typically correspond to hundreds of simulation time steps. In turn, this requires hybrid planners to navigate huge search trees4, so run–times become long, in the orders of thousands of seconds, when the planner finds a solution. Since we seek high performance simulation execution, we bound the length of the sequence of control signals considered, as is the standard practice in existing approaches [Gibbens and Medagoda, 2011] to UAV guidance based on MPC.

Bounding the search in this manner corresponds with seeking solutions to a net–benefit planning problem [Keyder and Geffner, 2009], where the reward or utility function is derived automatically from the symbolic description of the goal G. This reward function, first proposed in [Löhr et al., 2012], generalises the well known idea in planning that heuristic guidance is readily available from measuring to what degree each of the conditions in G are true in a given state. For the stern conversion task we consider in this demo, G is a quadratic equation encoding constraints on relative distances, angles and speeds between aircraft. Interestingly, in this setting, the problem of maintaining the goal over time is implicitly addressed. It can be shown that sequences of control signals, maximizing the utility function derived from G, necessarily model the LTL formula □\(\neg G\), as long as G is reachable from each state that follows from the selected sequence of signals. Guidance is obtained from performing a limited lookahead search guided by the structural novelty of states [Lipovetzky and Geffner, 2012] selecting control inputs on trajectories that end in state maximising the utility function. Our current implementation uses the simplest algorithm by Lipovetzky and Geffner, IW(1), as it has already shown great performance in deterministic and non-deterministic discrete games [Lipovetzky et al., 2015; Geffner and Geffner, 2015]. IW(1) is a plain breadth–first search, guaranteed to run in linear time and space, that prunes states based on how novel they are, where a state is novel if and only if it encounters a value of a state variable that it has not seen before.

5 Demo Overview

The demonstration will consist in showing a set of simulation histories, computed off–line, that can be readily played for interested passers–by on ACE 3D visualization of air-to–air combat. The action is rendered both on a screen and also via a pair of Microsoft Hololens. The selected histories illustrate how the hybrid planning controller compares with hand–programmed and game theoretic opponents, over a diverse set of initial conditions.

6 Discussion

The proposed demo system contains a number of contributions and innovations, which are discussed next. First, the notion of planning over simulators [Lipovetzky et al., 2015], previously only exercised over video games, is integrated with a realistic, professional simulation environment, ACE. Second, a compelling and realistic application of non–linear PDDL+ is presented, capable of performing close to real–time. Third, we present a practical example of how to instantiate the MPC framework using the tools, languages and theory proposed and developed by the domain–independent planning community. Last, by showing how efficient and robust model–based controllers can be, we look forward to dispelling the entrenched notion that the model–based approach to AI, while valid and interesting, is too expensive to be used in systems deployed in the real world [Geist, 2017].

2See [Piotrowski et al., 2016; Scala et al., 2016] for two recent hybrid planners.

3For an overview of how continuous change is handled we refer the reader to Ramirez et al. [2017].

4A similar problem with conducting complete search with A* up to the horizon becoming unfeasible is also reported by L¨ohr [2012].

5This formula is referred to as “weak maintenance” or “infinitely often” in the literature on LTL model checking [Pnueli, 1977].
References


