Continuous Pursuit-Evasion with Aerodynamic Constraints

Part II Computer Science Project Proposal

Brace Godfrey, dg681

October 9, 2025

Project Originator:	Brace Godfrey
Day-To-Day Supervisor:	Ramsey Faragher
Marking Supervisor:	Alastair Beresford
Director of Studies:	Ramsey Faragher

1 Introduction

Pursuit-Evasion is a problem domain in which one or more pursuers attempt to capture one or more evaders in some environment. This field has applications and links to game theory, spatial algorithms and air flight kinematics. In this project I will consider a three-dimensional continuous space populated by aerodynamically-accurate entities and investigate how a Markov Decision Process (MDP) can be used to model and solve pursuit-evasion problems in this space.

The core of my project will incorporate work from [1] where I will implement their `FastMDP` algorithm for efficiently solving pursuit-evasion problems using a Markov Decision Process. I will extend this work by considering how I can improve the visualisation of the simulation, consider additional kinematic and environmental constraints and introduce additional agents to the problem.

2 Structure of the Project

Project Core

The project core can be broken down into the following components. For each component I describe the work required and my implementation approach.

The project will be implemented in Python 3.13.

Visualisation

Initially the program's output will be a static image showing the paths taken by the pursuers and evaders by plotting their positions at each timestep. This is sufficient to demonstrate the basic functionality of the simulation. I will use the python library `matplotlib` to create these plots.

Modelling Agents Kinematically

Each agent will be modelled using a pseudo-6DoF (degrees of freedom) kinematic model inspired by the formulation in [1]. Each agent includes parameters:

- V: Airspeed in metres per second
- (x, y, z): Current position
- γ: Flight path angle in radians
- ϕ: Roll angle in radians
- ψ: Yaw angle in radians
- α: Angle of attack in radians

The inputs to the model will be the thrust n_x, the rate of change of ϕ, and the rate of change of α. The equations of motions of the agents can be calculated from these parameters and used for forward projection in the MDP.

Markov Decision Process and `FastMDP` Algorithm

Markov Decision Processes (MDPs) are a mathematical framework for decision-making in situations where outcomes are partly random and partly under the control of an agent. In pursuit-evasion, each agent's position and velocity define the state, possible manoeuvres define the actions, and the environment responds probabilistically. The `FastMDP` algorithm described in [1] provides an efficient way to solve pursuit-evasion problems using MDPs using forward projection of the simulation and policy evaluation.

Extensions

I will consider several possible extensions to the project core that explore additional constraints or complexities in the problem space. These include:

- Introducing terrain and obstacles into the 3D environment.
- Adding additional agents and investigating collaborative strategies among them.
- Implementing advanced aerodynamic limitations, such as stalling and maximum turn rates.
- Creating a dynamic visualisation, outputting a video or interactive 3D model of the simulation at each timestep and representing the agents visually with 3D models.

3 Starting Point

I have some experience with using and building Markov models in Python (in particular Markov Chains and Hidden Markov Models) from my Part IA Machine Learning and Real-World Data course, but I have not specifically implemented a Markov Decision Process or the `FastMDP` algorithm before.

I do not have experience with pursuit-evasion problems or explicit kinematic models, but I have written Python code to simulate physical three-dimensional systems before with a focus on linear algebra and vector mathematics rather than aerodynamics.

I am confident with the Python programming language, including the plotting library `matplotlib` and numerical libraries such as `numpy` and `pandas` which I will use to improve the efficiency of my code.

4 Evaluation

4.1 Success Criteria

For the project to be deemed a success, the following must be successfully completed:

1. The implemented `FastMDP` algorithm produces valid pursuer and evader behaviour.
2. The simulation computes agent actions for each timestep within a reasonable time frame, ensuring the algorithm is computationally efficient.
3. Any explored extensions, such as additional agents or constraints, produce valid and consistent results.

4.2 Evaluation Plan

The evaluation will focus on the following:

- **Trajectory validity:** Check distances and angles between agents at each timestep to ensure realistic flight paths.
- **Performance:** Measure CPU time and memory per timestep, and see how this changes with more agents or longer simulations.

5 Timetable and Milestones

Weeks 1 to 2 (13 Oct 25 - 26 Oct 25)

Submit Proposal.

Initialise a Git repository for the project and set up a basic python project structure.

Install any dependencies or libraries required and familiarise myself with the basic syntax and usage.

Continue to research Markov Decision Processes and the `FastMDP` algorithm and consider how it can be adjusted for my project.

Weeks 3 to 4 (27 Oct 25 - 9 Nov 25)

Implement a timestep system to update the positions of the agents in the simulation and create a movement path.

Connect the timestep system to a basic visualisation using `matplotlib`, initially testing with pre-defined paths.

Milestone: Generate static plots capable of displaying the path of an agent in a 3D environment.

Weeks 5 to 6 (10 Nov 25 - 23 Nov 25)

Create agent classes to represent the pursuers and evaders with plottable positions.

Begin to consider the reward functions used in the MDP for the pursuers and evaders considering how they will affect movements.

Weeks 7 to 8 (24 Nov 25 - 7 Dec 25)

Begin to implement kinematic constraints and calculations for agent positions.

Implement the reward functions for the pursuers and evaders based on agent positions and velocities.

Milestone: Agents capable of independent movement based on kinematic inputs and constraints and path correctly plotted.

Weeks 9 to 10 (8 Dec 25 - 21 Dec 25)

Finish implementation of kinematic constraints and calculations for agent positions.

Begin implementing the FastMDP algorithm using the kinematic properties of agents, and apply positional updates to the agents at each timestep.

Weeks 11 to 12 (22 Dec 25 - 4 Jan 26)

Slow implementation progress for Christmas and New Year holidays.

Take slack time to ensure all milestones have been met and that the project is on track.

Weeks 13 to 14 (5 Jan 26 - 18 Jan 26)

Continue implementing the FastMDP algorithm.

Start to consider how the MDP or FastMDP algorithm could be adjusted to introduce additional constraints or agents.

Milestone: Kinematic models of agents complete and accurate.

Weeks 15 to 16 (19 Jan 26 - 1 Feb 26)

Finish implementing FastMDP, ensuring that it uses the correct kinematic inputs and applies the correct positional updates to the agents.

Begin writing Implementation draft in dissertation.

Write progress report draft.

Weeks 17 to 18 (2 Feb 26 - 15 Feb 26)

Begin evaluating the project core against the success criteria and tests.

Create progress report presentation.

Finish writing Implementation draft in dissertation.

Milestone: Submit progress report.

Weeks 19 to 20 (16 Feb 26 - 1 Mar 26)

Take slack time to ensure all milestones have been met and that the project is on track.

Begin writing Evaluation draft in dissertation.

Weeks 21 to 22 (2 Mar 26 - 15 Mar 26)

Finish writing Evaluation draft in dissertation.

Begin exploring extensions, focussing on introducing additional constraints to the simulation and decision process.

Milestone: Implementation and Evaluation drafts complete.

Weeks 23 to 24 (16 Mar 26 - 29 Mar 26)

Continue exploring extensions, focussing on introducing additional agents to the simulation and promoting collaborative strategies.

Write Introduction draft in dissertation.

Weeks 25 to 26 (30 Mar 26 - 12 Apr 26)

Finish exploring extensions, and extend Implementation and Evaluation sections in dissertation to discuss explored extensions.

Write Conclusion draft in dissertation.

Milestone: Have a final code repository with the project core implementation and explored extensions complete.

Weeks 27 to 28 (13 Apr 26 - 26 Apr 26)

Combine written dissertation sections into a first full draft, ensuring correct formatting and presentation.

Submit draft to supervisors for feedback and make improvements.

Weeks 29 - 30 (27 Apr 26 - 10 May 26)

Continue to submit dissertation drafts to supervisors for feedback and make improvements.

Ensure that the codebase and the dissertation document are presentable and ready for submission.

Week 31 (11 May 26 - 15 May 26)

Submit final dissertation within deadline.

Milestone: Dissertation and code submitted.

6 Resource Declaration

I will be using my personal laptop (Macbook Pro 2021 — 16GB RAM, Apple M1 Pro) as my primary machine for software development. As a backup, I will use my personal desktop computer (32GB RAM, Intel 11th Gen Core i9) or a remote server provided by SRCF (student run computing facility). I will continuously backup my code and dissertation with Git version control and push to a remote GitHub repository.

References

[1] J. Bertram and P. Wei. “An Efficient Algorithm for Multiple-Pursuer-Multiple-Evader Pursuit/Evasion Game”. In: *AIAA Scitech 2021 Forum*. 2021.