Exploring the Unified Planning Framework for a More Integrated and Flexible Fault-Tolerant Flight Path Planning System

Sondes Morchedi[1], Prakash Jamakatel[1] and Jane Jean Kiam[1,†]

[1]*Universität der Bundeswehr München, Institute for Flight Systems, Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany*

Abstract

This paper reports the ambitions of an ongoing work intended to exploit the Unified Planning Framework (UPF) to enhance and extend a flight path planning system we developed using an automated planner to compute flight trajectories in non-nominal situations. We identify i) multiple benefits to draw using UPF, specifically to overcome the shortcomings of our previous implementation and ii) possible extensions of the flight path planning system thanks to the functions the UPF offers.

Keywords

Planning, Fault-tolerant Flight Planning, Flight Path Planning, Unified Planning Framework, PDDL+

1. Introduction

For more sustainable mobility solutions, the next generation aircraft is expected to reduce emissions significantly during flights while providing a better overall aircraft performance and safety [1]. In line with this, future intelligent flight systems are required to adapt to varying conditions, to the system's dynamics, and to handle faults while keeping the pilot informed. A fault-tolerant flight path planning system is an essential part of an intelligent flight system. In addition to increasing fuel efficiency, it is also expected to reduce damage and casualties in emergency situations caused by unforeseen events, e.g. faulty mechanics or human errors [1].

In the previous work [4], an automated flight path planning system was developed to assist the pilot in a single-pilot ultralight aircraft. The system was implemented using a PDDL+ compatible automated planner [5], which uses a forward state-space search and heuristics capable of coping with non-linear mathematical operations. In this paper, we report an ongoing work aimed at extending the previous work, by exploring the benefits of the UPF [6]. We first provide a brief summary of the previous work, followed by an overview of the adapted system

10th Italian Workshop on Planning and Scheduling (IPS-2022), November 28–02 December, 2022, Udine, IT

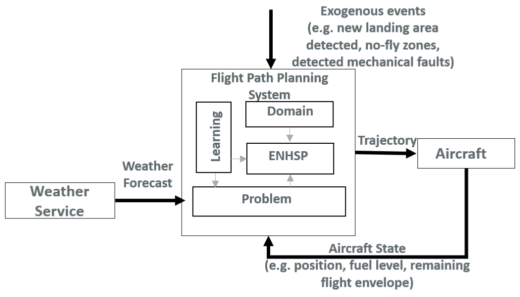
sondes.morchedi@unibw.de (S. Morchedi); prakash.jamakatel@unibw.de (P. Jamakatel); jane.kiam@unibw.de (J. J. Kiam)

0000-0002-2754-7095 (P. Jamakatel); 0000-0002-6830-9931 (J. J. Kiam)

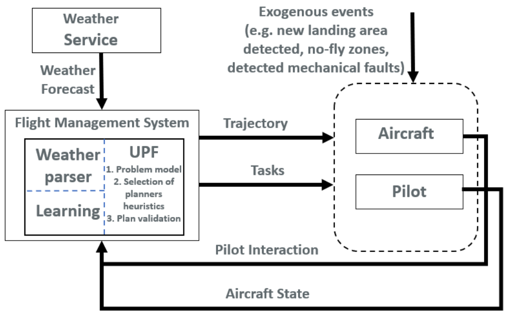
[1]Especially in non-commercial aviation, the majority of the accidents can be credited to human errors [2], due mainly to the less rigorous demand set on the pilot for flight licence, i.e. lesser number of training hours and laxer medical requirements [3].

architecture and underline the expected benefits of exploiting the UPF. Subsequently, we discuss the ongoing progress and challenges faced, alongside with the future work considered.

2. Exploring the UPF for the Flight Path Planning System



(a) Automated fault-tolerant flight path planning with ENHSP.



(b) A more flexible flight path planning system with the UPF

Figure 1: Comparing the flight path planning system architectures.

The aim of the automated fault-tolerant flight path planning system is to determine a flight trajectory meant to guide an aircraft (e.g. a single-pilot ultralight in [4]) from its current position to a goal position placed within the vicinity of a safe landing area. This is helpful especially when the pilot is overwhelmed, typically when an emergency landing is required due to low-fuel level, or to a mechanical fault. Figure 1a illustrates the architecture adopted in the previous work [4], in which the automated flight-path planning system considers weather information known at planning time[2], the current state (i.e. position and fault detection state variables) of the aircraft, and the goal position, all encoded as part of the (PDDL+) problem instance. The aircraft's physical model is encoded as a problem domain adapted from [8]. The ENHSP planner solver [5] is exploited to compute an executable temporal plan $\pi =< a_0(t_0), a_1(t_1), \cdots, a_n(t_n) >$, with each a_* representing a set of control parameters, namely the turn rate and the climb rate, which will be used to predict the sequence of waypoints $< x_0(t_0), x_1(t_1), \cdots, x_n(t_n) >$, setting therefore a feasible reference trajectory for the autopilot to guide the airplane, in case the pilot is incapable of steering the aircraft. A learning module built on k-means clustering is part of the planning system to determine the optimal planning parameters for non-nominal situations (i.e. emergency landing with low fuel or maneuvering a faulty aircraft) using flight data generated from flight simulations, for tuning the ENHSP planner solver, as well as for selecting the ranges of the state variables in the planning problem, in order to simplify the search [9].

Nevertheless, there are several shortcomings worth mentioning in the previous works [4, 9]. To call the ENHSP planner solver, another bash instance is created within the flight path planning system, and run as an external process, limiting therefore the interaction with the planner solver. Furthermore, minimum changes in the problem instance, due to either a new

[2]The weather information used is from the National Oceanic and Atmospheric Administration (NOAA) [7].

goal condition, e.g. if a better landing area is detected, or to minimal changes in the weather information, the problem domain and instance have to be parsed and grounded entirely, which can be time-consuming for a more complex problem model. Therefore, we have identified the UPF as a more compact and efficient framework that can overcome these shortcomings, and offer more meaningful extensions to the fault-tolerant flight path planning system.

2.1. The Unified Planning Framework (UPF)

The UPF [6] is a collaborative effort under development within the AIPlan4EU project, with the purpose of providing flexibility in the planning problem definition (similar to *Tarski* [10]) and making a wide range of planning technologies accessible (similar to *Planutils* [11]), via their integration as Python libraries within a single framework.

2.2. Benefits of using UPF for the Flight Path Planning System

Based on the published information and objectives of the UPF in [6] and [12], we identified some features of the UPF that can be exploited to develop a more flexible and integrated flight path planning system, of which the architecture is as depicted in Figure 1b. The benefits drawn from the UPF-based flight path planning system architecture are described below.
Overcoming the shortcomings of the previous work: Without having to invoke an external bash process, it is more flexible to interact with the planning engines within the same Python framework, which is particularly useful when we consider a human-in-the-loop planning system, in which the pilot can modify a solution plan, for example by adding an intermediate waypoint. In this case, the `PlanValidator` can be called programmatically.

A more prominent improvement the UPF will bring forth is the coping with a dynamic environment, as it allows to change state variables programmatically without having to parse the entire problem instance completely, reducing thereby the overall computational time. With this flexible parsing, a partial planning problem containing typically weather data and airspace constraints can be pre-encoded, later completed by the pilot (with goals or additional constraints), who interacts with the flight path planning system via a cockpit user interface. Domain-specific validator can check the pilot's inputs for conflicts (with the constraints of the existing partial problem) before including them into the planning problem. Having this implemented, the flight path planning system will conform with the EASA Roadmap for the use of AI in a cockpit: the system is pilot-centered, with AI playing an assisting role [13].
Other potential extensions: To date, the only planner solver used for the (kinodynamic) flight path planning problem at hand is ENHSP. The main hindrance in using other planners is the lack of support for non-linear mathematical operations encoded in the problem domain. With the parser of UPF supporting the encoding of non-linear mathematical operations, it is likely that more planner solvers capable of solving this class of problem will be included in the UPF, leading to the possibility of creating multiple planning instances in parallel for different planners or for different heuristics of the same planner, emulating thereby "diverse planning" to obtain multiple solutions that can be presented to the pilot for assisting him/her in decision-making.

Since the current implementation relies only on one solver (ENHSP), plan repair, although can be considered, can only be done externally to the AI planning module, and in a "programmatic" and domain-specific manner, as in [8], where partial plans are "stitched" together by adding an artificial "bridge" to ensure continuity between them. With the UPF, even if the planner solver does not include plan repair capabilities, another planning engine can be used for repairing a plan, which is relevant to our use case, given the dynamic environment, e.g. a no-fly-zone is determined during flight due to detected moving objects nearby (such as drones, cranes, etc.).

The learning module in the previous work was based on an unsupervised learning method, with the aim to select the set of planning and control parameters to optimize the likelihood of a safe landing in non-nominal situations. This learning module can be extended to include inverse reinforcement learning to learn aircraft maneuvers from an experienced pilot in emergency landing, and use these "pre-learned" maneuvers as actions in the planning problem. Having planning libraries implemented in Python enables the exploitation of Python libraries such as scikit-learn [14] or OpenAI Gym [15] within the same Python-based system.

3. Discussion and Future Work

Having identified the benefits of the UPF for our current flight path planning system that was developed in Matlab in [4] and in C++ in [8], we are currently working towards porting the parser for weather data, the automated encoding of problem models, and the learning module into Python3. As soon as the integration of PDDL+ syntax and *global constraints*, which have been proven to be extremely handy in defining no-fly zones in a compact manner, is realized in the UPF, the architecture shown in Figure 1b can be fully implemented without much adaptation and its performance can be compared to the previous work. Subsequently, the improvements and extension described in Section 2.2 can also be included.

3.1. Future Work to Include Hierarchical Task Network Planning

As pointed out in the "Future Work" of [6], hierarchical structures will be added into the UPF, enabling thereby the solving of Hierarchical Task Network (HTN) planning problems. This will further benefit the extension of the flight path planning system to an even more versatile flight management system capable of assisting the pilot with tasks of higher abstraction levels[3] while simultaneously communicating a flight path plan to reach the desired goal location. With this extension, not only that (high-level) task and motion planning can be supported by the same framework and communicated to the pilot in a hierarchical manner (that is more comprehensible for human cognition), plan and goal recognition can also be performed in an automated fashion using HTN planning engines [17], resulting in a close-loop assisting system.

4. Acknowledgments

This work is done within the research group "Intelligent Control of Highly Over-Actuated Flight Systems" funded by Munich Aerospace and within the MOREALIS project funded by the German Federal Ministry of Economic Affairs and Climate Action.

[3]Pilots are in general instructed with tasks that are hierarchical in nature, and abstract away minute details, as domain-level knowledge is assumed [16].

5. Citations and Bibliographies

References

[1] European Union Aviation Safety Agency, Study on the societal acceptance of urban air mobility in europe, 2021.

[2] A. De Voogt, F. Chaves, E. Harden, M. Silvestre, P. Gamboa, Ultralight accidents in the us, uk, and portugal, Safety 4 (2018) 23.

[3] European Union Aviation Safety Agency, Light sport aircraft, 2022. Accessed: 2022-10-18.

[4] B. S. León, J. J. Kiam, A. Schulte, Model-based automated flight path planning for an ultralight aircraft (2020).

[5] E. Scala, P. Haslum, S. Thiébaux, M. Ramirez, Interval-based relaxation for general numeric planning, in: ECAI 2016, IOS Press, 2016, pp. 655–663.

[6] A. A. Micheli, A. Alexandre Bit-Monnot, al., Unified planning: A python library making planning technology accessible, in: International Conference on Automated Planning and Scheduling: System Demonstrations, 2022.

[7] G. Rutledge, J. Alpert, S. Ronald, B. Lawrence, The noaa operational model archive and distribution system (nomads) (2003).

[8] J. J. Kiam, E. Scala, M. R. Javega, A. Schulte, An ai-based planning framework for haps in a time-varying environment, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 30, 2020, pp. 412–420.

[9] B. S. León, J. J. Kiam, A. Schulte, A fault-tolerant automated flight path planning system for an ultralight aircraft, in: M. Baldoni, S. Bandini (Eds.), AIxIA 2020 – Advances in Artificial Intelligence, Springer International Publishing, Cham, 2021, pp. 175–190.

[10] M. Ramırez, G. Frances, Tarski, URL https://github. com/aig-upf/tarski. Accessed on (2021) 05–17.

[11] S. J. K. M. Muise Christian, Pommerening Florian, Planutils: Bringing planning to the masses, International Conference on Automated Planning and Scheduling: System Demonstrations (2022).

[12] A. Micheli, A. Arnold, A. Bit-Monnot, and other contributors, Unified-planning documentation, https://unified-planning.readthedocs.io/en/latest/, 2022. Accessed: 2022-10-21.

[13] European Union Aviation Safety Agency (EASA), Artifial intelligence roadmap: A human-centric approach to ai in aviation, 2020.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540 (2016).

[16] Cessna Aircraft Company, Cessna 172 1974 skyhawk owner's manual: Pilot operating handbook (poh) / aircraft flight manual (afm), Independently Published, 1972.

[17] D. Höller, G. Behnke, P. Bercher, S. Biundo, Plan and goal recognition as htn planning, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2018, pp. 466–473.